



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
У НОВОМ САДУ




Игор Шикуљак

**Систем за праћење кретања
мобилних уређаја са
омогућеном употребом Вај-фај
технологије**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2021

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА	Лист/Листова:
		1/1

(Податке уноси предметни наставник - ментор)

Врста студија:	<input checked="" type="checkbox"/> Основне академске студије
Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	Проф. др Милан Видаковић

Студент:	Игор Шикунљак	Број индекса:	RA117/2017
Област:	Рачунарске мреже, безбедност система, имбедед системи		
Ментор:	Проф. др Милан Видаковић		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: <ul style="list-style-type: none"> - проблем – тема рада; - начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; - литература 			

НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:

Систем за праћење кретања мобилних уређаја са омогућеном употребом Вај-фај технологије
--

ТЕКСТ ЗАДАТКА:

Задатак рада представља креирање система за прикупљање података од значаја који ће омогућити детекцију и праћење кретања уређаја који користе Вај-Фај технологију, без обзира на статус њихове конекције на неку мрежу. Стога, студент треба да: <ul style="list-style-type: none"> - проучи релевантне теоријске концепте, из ИЕЕЕ802.11 технологије те пронађе и осмисли методе који ће омогућити прикупљање потребних података - изабере имбедед платформу на којој ће реализовати осмишљени систем - имплементира осмишљени систем и изврши тестирање истог - дискутује реализовано рјешење и предложи даља унапређења
--

Руководилац студијског програма:	Ментор рада:

Примерак за: <input type="checkbox"/> - Студента; <input type="checkbox"/> - Ментора
--

Sadržaj

1	Uvod.....	7
2	Teorijske osnove	9
2.1	IEEE 802.11 standard	9
2.1.1	Distribucija kanala u 2,4GHz frekvencijskom spektru (IEEE 802.11 b/g/n/ax) ...	9
2.1.2	IEEE 802.11 frame	10
2.1.3	Probe Request paketi.....	11
2.1.4	Request to Send (RTS) i Clear to Send (CTS) paketi	12
3	Arhitektura sistema	15
3.1	Opšta struktura predloženog rješenja	15
3.2	Platforma za realizaciju rješenja i njena ograničenja	15
3.2.1	ESP8266 SoC i NodeMCU platforma	15
3.2.2	Dodatni moduli – periferije.....	16
3.2.3	Razvojni alati, šema sistema, platformska ograničenja i njihovo rješenja	16
4	Implementacija	21
4.1	SdFat biblioteka	21
4.2	Konfiguracija glavnog-upravljačkog NodeMCU-a	21
4.3	NTP sinhronizacija vremena	22
4.4	Rad sa RTC satom.....	24
4.4.1	Inicijalizacija RTC sata sa tačnim vremenom (u preciznosti sekunde)	24
4.4.2	Praćenje vremena SQW interrupt-om (u preciznosti milisekunde).....	24
4.5	Inicijalizacija sistema.....	25
4.6	Aktivni režim rada glavnog NodeMCU uređaja	27
4.7	Prikupljanje 802.11 paketa i njihova obrada	29
4.8	Slanje komandi RTS odašiljaču.....	31
4.9	RTS odašiljač	31
5	Rezultati i diskusija	35
5.1	Praćenje mobilnih uređaja nezavisno od RTS odašiljača	35
5.2	RTS odašiljač	36
5.3	MAC randomizacija	37
6	Budući rad	39
7	Zaključak.....	41
8	Literatura	43

Biografija.....	45
КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА.....	47
KEY WORDS DOCUMENTATION	49

1 Uvod

Mobilni uređaji koji upotrebljavaju WiFi tehnologiju [1], u svom normalnom režimu rada neizbježno odaju određene identifikacione podatke svojoj okolini. Kako je medijum kroz koji se prenose podaci zajednički za sve uređaje u okolini, tj. radi se o fizičkom prostoru u kome se podaci prenose bežično, tako jedini način za sprovođenje uspješne komunikacije jeste da svaki set prenošenih podataka sa sobom nosi informaciju/identifikaciju porijekla i destinacije. Jedinstveni identifikator svakog fizičkog uređaja u cijeloj IEEE 802 [2] grupi standarda jeste MAC (Media Access Control) adresa.

Cilj ovog rada jeste kreiranje sistema koji će omogućiti praćenje kretanja mobilnih uređaja (WiFi stanica, WiFi station) u prostoru. Ovaj rad se ograničava na praćenje uređaja koji rade u okviru IEEE 802.11 b/g/n bežičnih mreža, što odgovara uobičajenim, opšteprisutnim verzijama 2.4Ghz WiFi tehnologije. Detekcija uređaja se postiže slušanjem komunikacije u neposrednoj okolini jedne jedinice našeg sistema te prikupljanjem relevantnih podataka za identifikaciju uređaja. Od interesa su specifični paketi u okviru komunikacije između mobilnih uređaja i WiFi pristupnih tački (Access Point, AP), a to su Probe Request paketi. Probe Request pakete šalju mobilni uređaji sa ciljem pronalaženja WiFi pristupnih tački u okolini kao i utvrđivanja kvaliteta njihovog signala.

Dodatno se razmatra i aktivna transmisija RTS (Request to Send) paketa sa ciljem povećanja broja i učestalosti slanja Probe Request paketa od strane mobilnih uređaja, što bi omogućavalo i precizniju lokaciju i praćenje istih.

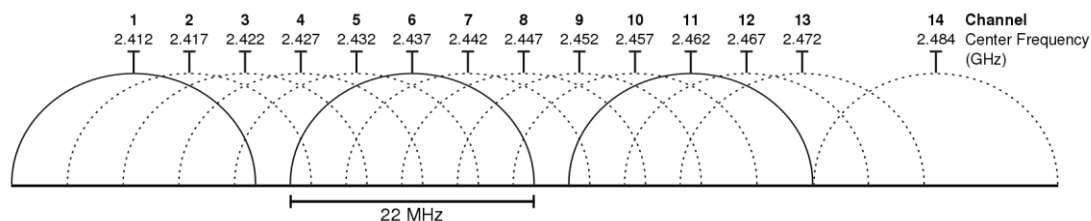
2 Teorijske osnove

2.1 IEEE 802.11 standard

IEEE 802.11 je set standarda koji definiše komunikacione protokole u bežičnom aspektu mreža lokalnog nivoa (Wireless Local Area Networks, WLAN). U nastavku će biti objašnjeni relevantni dijelovi tog standarda za ovaj rad.

2.1.1 Distribucija kanala u 2,4GHz frekvencijskom spektru (IEEE 802.11 b/g/n/ax)

Kolizije elektromagnetnih signala narušavaju integritet poslatih podataka, te je u tim slučajevima neophodno izvršiti retransmisiju datog podatka. Najjednostavniji primjeri za to jesu ako dva uređaja pokušaju da pošalju pakete u istom trenutku na istom frekvencijskom opsegu, ili jednostavno prirodna ili vještačka interferencija iz okoline. Učestalost potrebe za retransmisijom paketa utiče na performanse uređaja. Ovaj problem je značajan u okolnostima kada postoji više WiFi pristupnih tački na istom mjestu, te se onda date mreže međusobno ometaju. Zbog toga, alocirani frekvencijski spektar za mreže koje prate IEEE 802.11 b/g/n ili ax standarde je podijeljen na više kanala. Svaka mreža radi na najmanje jednom kanalu, i tako se može obezbijediti potpuno nesmetana komunikacija u datoj tački prostora za najmanje tri mreže. Spektar je podijeljen na ukupno 14, međusobno djelimično preklapajućih kanala širine 22 MHz (slika 1), sa idejom da ako nije moguće potpuno eliminisati koliziju između mreža, onda se ona može makar djelimično umanjiti.



Slika 1 - Kanali u 2,4GHz spektru, njihovo preklapanje i širina ¹

Kanali 1-13 su kanali koji su dozvoljeni zakonskim regulativama u većem dijelu svijeta, dok to nije slučaj za kanal 14. Zbog toga, kanal 14 nije uzet u obzir u implementaciji ovog rada.

¹ Izvor:

Michael Gauthier, *Wireless Networking in the Developing World*
KelleyCook, dorada slike
CC BY-SA 3.0

Mreže obično funkcionišu po HT20 (High Throughput 20) principu što znači da za svoje funkcionisanje koriste jedan od pomenutih kanala u opsegu od 20Mhz (preostala 2 MHz su raspoređena na krajevima opsega i ne koriste se – služe rasipanju, slabljenju signala ka narednom neometanom signalu). IEEE 802.11n standard definiše i HT40 princip upotrebe alociranog frekvencijskog spektra i on podrazumjeva istovremenu upotrebu dva nepreklapajuća kanala od strane jedne WiFi pristupne tačke. U tom slučaju se jedan kanal koristi za prenos paketa sa podacima, a drugi za ostale pakete – ukupno 40MHz iskorišćenog frekvencijskog opsega, te otuda i naziv HT40.

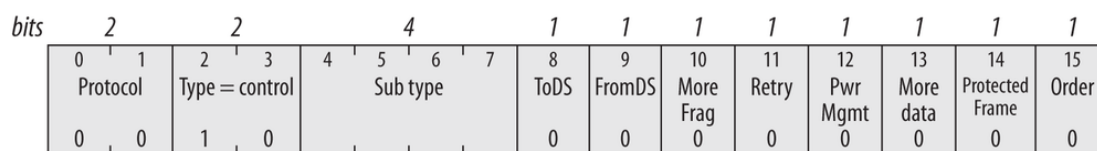
2.1.2 IEEE 802.11 frame

Sva komunikacija koja odvija preko WiFi mreže, bez obzira na protokol, se enkapsulira u 802.11 frame. Polja u tzv. MAC zaglavlju definišu razne osobine specifičnog paketa, kao što su npr. tip paketa, adresa porijekla, adresa destinacije, stanje paketa, kontrolni podaci... [3] Na slici 2 je prikazano MAC zaglavlje sa dužinom polja izraženom u bajtima.



Slika 2 - Polja u generičkom 802.11 frame-u ²

Kao što se može vidjeti, postoji više adresnih polja od samo porijekla i destinacije. To je zbog toga što konfiguracija mreže može biti znatno kompleksnija od tipa stanica-pristupna tačka, i može da uključuje posrednike na obje strane te veze. Da bi se adresna polja mogla ispravno tumačiti, u okviru Frame Control (slika 3) polja se definiše distribicioni sistem vezan za dati paket.



Slika 3 - Polja prisutna unutar Frame Control polja u 802.11 frame-u ³

Polja ToDS i FromDS definišu okruženje u kakvom je dati paket koji je u tranzitu. [4] U slučaju direktne komunikacije oba bita (polja, zastavice) neće biti setovana, dok u slučajevima kada paket napušta/namijenjen je za distribicioni sistem, tada će respektivno FromDS i ToDS biti setovani. Moguća je situacija i kada su oba bita setovana.

Adresna polja mogu imati neka od sledećih značenja:

- RA (Receiver Address) – adresa primaoca,
- TA (Transmitter Address) – adresa pošiljaoca,

² Izvor: [3]

³ Izvor: [3]

- DA (Destination Address) – adresa destinacije (krajnji primaoc paketa),
- SA (Source Address) – adresa izvora (originalni pošiljalac paketa) i
- BSSID (Basic service set identifier) – adresa nadležne mreže.

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	RA = DA	TA = SA	BSSID	N/A
0	1	RA = DA	TA = BSSID	SA	N/A
1	0	RA = BSSID	TA = SA	DA	N/A
1	1	RA	TA	DA	SA

Tabela 1 - Značenje sadržaja adresnih polja na osnovu sadržaja DS polja ⁴

2.1.3 Probe Request paketi

WiFi stanice emituju Probe request pakete sa ciljem otkrivanja dostupnih pristupnih tačaka na datoj lokaciji. Paketi su obično broadcast tipa, što znači da su usmjereni ka svim pristupnim tačkama, i od svih njih koji su u dometu se očekuje odgovor. Broadcast paketi se identifikuju po jedinstvenoj adresi (MAC adresa) destinacije (DA) – FF:FF:FF:FF:FF:FF. Sa druge strane, Probe request paketi mogu biti usmjereni i ka specifičnoj pristupnoj tački. WiFi stanice emituju takve pakete uglavnom sa razlogom da otkriju promjene u kvalitetu signala od određene pristupne tačke.

U skladu sa prethodno iznijetim informacijama, mobilni uređaji emituju Probe request pakete da bi otkrili pristupne tačke u blizini i povezali se na prethodno poznatu mrežu, ili da bi mogli inicirati novu konekciju. Osim toga, uređaji periodično emituju Probe request pakete bez obzira da li su u datom trenutku konektovani na određenu mrežu ili ne, sa ciljem utvrđivanja kvaliteta signala pristupnih tačaka. To omogućuje automatizovani prelazak sa jedne mreže na drugu poznatu mrežu u zavisnosti od kvaliteta signala i time pružanje bolje usluge.

Probe request paketi pripadaju Management tipu paketa (u 802.11 frame-u oznaka tipa u frame control sekciji nosi vrijednost 0) i imaju oznaku podtipa u 802.11 Frame Control sekciji sa vrijednošću 4 (slika 4).

⁴ Izvor: [4]

wlan.fc.type_subtype == 0x4						
No.	Time	Source	Destination	Protocol	Length	Info
11	0.820504	c6:a4:1a:c6:48:23	Broadcast	802.11	112	Probe Request
<div> <div>IEEE 802.11 Probe Request, Flags:C</div> <div>Type/Subtype: Probe Request (0x0004)</div> <div> <div>Frame Control Field: 0x4000</div> <div> <div>.... 00 = Version: 0</div> <div>.... 00.. = Type: Management frame (0)</div> <div>0100 = Subtype: 4</div> </div> <div> <div>Flags: 0x00</div> <div> <div>.... 00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x0)</div> <div>.... 0.. = More Fragments: This is the last fragment</div> <div>.... 0... = Retry: Frame is not being retransmitted</div> <div>...0 = PWR MGT: STA will stay up</div> <div>..0. = More Data: No data buffered</div> <div>.0.. = Protected flag: Data is not protected</div> <div>0... = +HTC/Order flag: Not strictly ordered</div> </div> <div> <div>.000 0000 0000 0000 = Duration: 0 microseconds</div> <div>Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)</div> <div>Destination address: Broadcast (ff:ff:ff:ff:ff:ff)</div> <div>Transmitter address: c6:a4:1a:c6:48:23 (c6:a4:1a:c6:48:23)</div> <div>Source address: c6:a4:1a:c6:48:23 (c6:a4:1a:c6:48:23)</div> <div>BSS Id: Broadcast (ff:ff:ff:ff:ff:ff)</div> <div>.... 0000 = Fragment number: 0</div> <div>1010 0011 1010 = Sequence number: 2618</div> <div>Frame check sequence: 0x60000000 [unverified]</div> <div>[FCS Status: Unverified]</div> </div> </div> <div>IEEE 802.11 Wireless Management</div> <div> <div>Tagged parameters (69 bytes)</div> <div> <div>> Tag: SSID parameter set: Wildcard SSID</div> <div>> Tag: Supported Rates 1, 2, 5.5, 11, 6, 9, 12, 18, [Mbit/sec]</div> <div>> Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]</div> <div>> Tag: DS Parameter set: Current Channel: 11</div> <div>> Tag: HT Capabilities (802.11n D1.10)</div> <div>> Tag: Extended Capabilities (9 octets)</div> <div>> Tag: Vendor Specific: Microsoft Corp.: Unknown 8</div> </div> </div> </div> </div>						
0000	00 00 0f 00 2a 00 00 00	10 00 00 00 00 00 00 40*C		
0010	00 00 00 ff ff ff ff ff	ff c6 a4 1a c6 48 23 ffH#			
0020	ff ff ff ff ff a0 a3 00	00 01 08 02 04 0b 16 0c			
0030	12 18 24 32 04 30 48 60	6c 03 01 0b 2d 1a 2d 01	..\$2.0H` 1	----		
0040	17 ff 00 00 00 00 00 00	00 00 00 00 00 00 00 00			
0050	00 00 00 00 00 00 00 00	7f 09 01 00 08 00 00 00			
0060	00 40 80 dd 07 00 50 f2	08 00 11 00 00 00 00 60	..@.....P`		

Slika 4 - Primjer Probe request paketa

2.1.4 Request to Send (RTS) i Clear to Send (CTS) paketi

RTS predstavlja mehanizam za razrješavanje problema sa kolizijama u slanju paketa. Osnovna ideja jeste da, kada se desi kolizija, umjesto ponovnog slanja samog paketa, pošiljalac primaocu prvo pošalje Request to Send (zahtjev za slanje) paket, na koji će primaoc da odgovori sa Clear to Send (dozvoljeno slanje) paketom. Nakon te razmjene, pošiljalac može da pošalje paket primaocu, a primaoc neće preduzimati nikakve akcije slanja na određeno vrijeme. Takođe, drugi uređaji na mreži koji nisu uključeni u datu komunikaciju će takođe obustaviti slanje svojih paketa odmah po primanju RTS paketa, a nastaviće svoj normalni rad tek nakon registrovanja CTS paketa, odnosno kada prođe unaprijed određeni vremenski interval.

Request to send paketi pripadaju Control tipu paketa (u 802.11 frame-u oznaka tipa u frame control sekciji nosi vrijednost 1) i imaju oznaku podtipa u 802.11 frame control sekciji sa vrijednošću 11 (slika 5).

wlan.fc.type_subtype == 0x1b				
No.	Time	Source	Destination	Protocol
13859	881.689600	MurataMa_cc:4b:a5 (b0:72:bf:cc:4b:a5)...	TaicangT_03:7d:19 (f8:6c:e1:03:7d:19)...	802.11
<				
> Frame 13859: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface \Device\NPF_{A96511F6-C				
> Radiotap Header v0, Length 15				
> 802.11 radio information				
IEEE 802.11 Request-to-send, Flags:C				
Type/Subtype: Request-to-send (0x001b)				
Frame Control Field: 0xb400				
.... ..00 = Version: 0				
.... 01.. = Type: Control frame (1)				
1011 = Subtype: 11				
Flags: 0x00				
.... ..00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0)				
.... .0.. = More Fragments: This is the last fragment				
.... 0... = Retry: Frame is not being retransmitted				
...0 = PWR MGT: STA will stay up				
..0. = More Data: No data buffered				
.0.. = Protected flag: Data is not protected				
0... = +HTC/Order flag: Not strictly ordered				
.000 0110 0010 1110 = Duration: 1582 microseconds				
Receiver address: TaicangT_03:7d:19 (f8:6c:e1:03:7d:19)				
Transmitter address: MurataMa_cc:4b:a5 (b0:72:bf:cc:4b:a5)				
Frame check sequence: 0x00002000 [unverified]				
[FCS Status: Unverified]				
0000	00 00 0f 00 2a 00 00 00	50 00 00 00 00 00 00	b4*... P.....
0010	00 2e 06 f8 6c e1 03 7d	19 b0 72 bf cc 4b a5	00	...1...}...r...K...
0020	20 00 00			..

Slika 5 - Primjer Request to Send paketa

Za ovaj rad je značajno je napomenuti da CTS paketi u svom 802.11 zaglavlju ne sadrže adresu pošiljaoca, već samo adresu primaoca.

Clear to send paketi pripadaju Control tipu paketa (u 802.11 frame-u oznaka tipa u frame control sekciji nosi vrijednost 1) i imaju oznaku podtipa u 802.11 frame control sekciji sa vrijednošću 12 (slika 6).

wlan.fc.type_subtype == 0x1c				
No.	Time	Source	Destination	Protocol
13886	887.108841		98:5f:98:06:85:58 (98:5f:98:06:85:58)...	802.11
<				
> Frame 13886: 29 bytes on wire (232 bits), 29 bytes captured (232 bits) on interface \Device\NPF_{A96511F6-}				
> Radiotap Header v0, Length 15				
> 802.11 radio information				
IEEE 802.11 Clear-to-send, Flags:C				
Type/Subtype: Clear-to-send (0x001c)				
Frame Control Field: 0xc400				
.... ..00 = Version: 0				
.... 01.. = Type: Control frame (1)				
1100 = Subtype: 12				
Flags: 0x00				
.... ..00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0)				
.... .0.. = More Fragments: This is the last fragment				
.... 0... = Retry: Frame is not being retransmitted				
...0 = PWR MGT: STA will stay up				
..0. = More Data: No data buffered				
.0.. = Protected flag: Data is not protected				
0... = +HTC/Order flag: Not strictly ordered				
.001 1011 0111 0100 = Duration: 7028 microseconds				
Receiver address: 98:5f:98:06:85:58 (98:5f:98:06:85:58)				
Frame check sequence: 0x21000000 [unverified]				
[FCS Status: Unverified]				
0000	00 00 0f 00 2a 00 00 00	50 00 00 00 00 00 00	c4*... P.....
0010	00 74 1b 98 5f 98 06 85	58 00 00 00 21		.t... ..X...!

Slika 6 - Primjer Clear to Send paketa

3 Arhitektura sistema

3.1 Opšta struktura predloženog rješenja

Ideja za realizaciju sistema se sastoji u postavljanju više malih uređaja – jedinica sistema, raspoređenih na određenom prostoru. Svaki od tih uređaja će imati sposobnost da sluša mrežni saobraćaj iz svoje okoline, izdvaja iz njega sve Probe Request pakete i bilježi relevantne identifikacione podatke. Naknadno se okupljaju podaci od svih pojedinačnih uređaja, povezani sa informacijom o lokaciji svakog uređaja koji je prikupljao podatke i vremenskim trenutkom kada je svaki podatak registrovan. Na osnovu tih podataka je moguće rekonstruisati kretanje mobilnih uređaja prostim utvrđivanjem kada je određeni mobilni uređaj bio u okolini naše jedinice sistema – preko činjenice da li je jedinica sistema tada registrovala Probe Request-ove od datog mobilnog uređaja.

Pored praćenja Probe Request paketa, razmatra se i ciljano slanje RTS paketa. Od tog procesa se kao posljedica očekuje povećan broj Probe Requestova od ciljanog mobilnog uređaja (WiFi stanice), jer se očekuje jedan od dva scenarija. Ako je mobilni uređaj povezan na WiFi mrežu, postojanje RTS paketa može prouzrokovati potragu za novom mrežom koja će biti manje opterećena i time pružati bolju uslugu, ili, ako je mobilni uređaj nepovezan i u periodičnoj potrazi za mrežama, ubrzaće dolazak novog ciklusa emitovanja Probe Request-ova na red. Sa druge strane, takođe bi bilo moguće slušanje na CTS poruke što bi potvrdilo prisustvo mobilnog uređaja u neposrednoj blizini naše jedinice sistema. Kao što je navedeno u poglavlju 2.4.1, CTS paketi ne sadrže adresu porijekla paketa, pa to blago komplikuje pristup praćenju kretanja više uređaja, jer se CTS odgovor mora vezivati za posljednji poslati RTS paket (vezivanje za njegovu destinaciju). Pored toga, ne bi bilo moguće brzo slanje velikog broja RTS paketa namijenjenih različitim ciljnim uređajima, jer bi to moglo dovesti do preplitanja odgovora, te je potrebno praviti sitnije pauze između slanja RTS paketa različitim ciljanim WiFi stanicama.

3.2 Platforma za realizaciju rješenja i njena ograničenja

3.2.1 ESP8266 SoC i NodeMCU platforma

Za implementaciju predloženog rješenja je izabran SoC (System on a Chip) ESP8266 koji proizvodi Espressif Systems [5], odnosno ESP8266 Dev Kit platforma poznata pod nazivom NodeMCU [6]. Dati izbor je napravljen jer pruža podršku za većinu funkcionalnosti koje su potrebne za implementaciju predstavljenog rješenja, a i zbog svoje izrazito niske cijene.

3.2.2 Dodatni moduli – periferije

Prikupljene podatke je potrebno potkrijepiti tačnim vremenom njihovog prikupljanja i skladištiti ih. Za precizno praćenje vremena se koristi RTC (Real Time Clock) modul sa DS3231 čipom. Bitno je napomenuti da je za ovaj projekat neophodna verzija čipa DS3231S(N) [7] jer ona ima programabilan generator signala (SQW, Square Wave Generator), dok druge verzije, kao npr. DS3231M [8], ga nemaju. Za povezivanje RTC sata na NodeMCU se koristi I2C (Inter-Integrated Circuit) protokol. Skladištenje podata se obavlja na (mikro) SD memorijsku karticu. SD kartice imaju SPI interfejs, pa se direktno povezuju na NodeMCU i sa njima se komunicira SPI (Serial Peripheral Interface) protokolom.

3.2.3 Razvojni alati, šema sistema, platformska ograničenja i njihovo rješenja

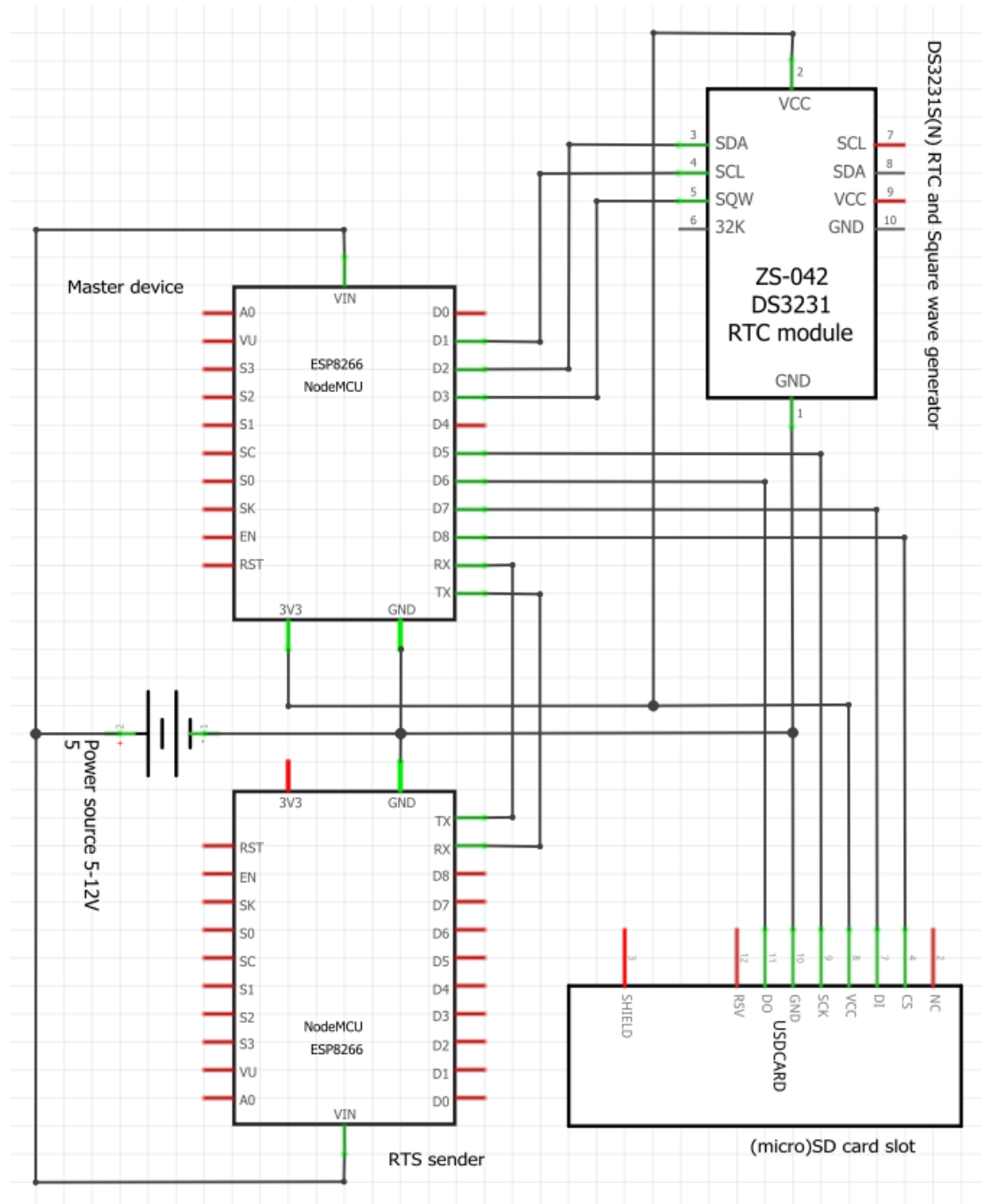
Za programiranje ESP8266 postoji ESP8266 NON-OS SDK [9], ali programiranje u njemu za veći dio sistema se radi u Arduino Framework-u (ESP8266 core for Arduino) [10] koji omogućava značajno jednostavniju implementaciju. ESP8266 core for Arduino u svojoj pozadini koristi ESP8266 NON-OS SDK, pa sve njegove funkcionalnosti su dostupne i unutar samog ESP8266 core for Arduino Framework-a.

ESP8266 zajedno sa svojim ESP8266 NON-OS SDK ima podršku za tzv. promiscuous mode rada na mreži, tj. može da prati mrežne pakete koji se šalju u njegovoj okolini, a nije ograničen na samo pakete koji pripadaju mreži sa kojom je trenutno asociran. Prema tehničkoj specifikaciji [11] i API dokumentaciji [12] podržani su samo paketi u HT20 principu rada, i postoji ograničenje da se prihvataju na korektnu obradu samo paketi koji prolaze određene kriterijume. Jedan od njih jeste da MAC zaglavlje (a preko toga i 802.11 paket u cjelosti) mora imati minimalnu dužinu od 24 bajta. Međutim, kao što se to može vidjeti na primjeru za CTS paket u poglavlju 2.1.4 (slika 6), dužina CTS paketa je manja od toga. To znači da nije moguće registrovanje CTS paketa.

U API dokumentaciji [12] se može vidjeti da ESP8266 ima mogućnost slanja tzv. freedom paketa, tj. moguće je slanje programski definisanih 802.11 paketa bez potrebe da ESP8266 bude asociran ili u procesu asocijacije sa nekom WiFi mrežom. Nažalost, ta funkcionalnost je ograničena na slanje neenkriptovanih Data paketa, Beacon, Probe Request i Probe Response paketa. Dodatno ograničenje (koje je vrijedno pomena u ovom radu) je da poslati paket mora imati minimalnu dužinu od 24 bajta, iskućujući FCS polje. Zbog navedenog, nije moguće slanje RTS paketa. Međutim, u verziji 1.3.0 ESP8266 NON-OS SDK [13][14], se prvi put pojavila metoda za slanje freedom paketa. U tom trenutku je prisustvo te funkcije bilo nezvanično, nije se pominjala u dokumentaciji, i očigledno je da data funkcija u svojoj implementaciji za tu verziju ne vrši provjeru koji je tip (tip i podtip iz Frame Control polja) paketa u pitanju. To nam omogućava da ipak šaljemo RTS pakete, zbog čega u sistem ubacujemo još jedan

NodeMCU, koji će imati ulogu RTS sendera, a komadne će dobijati od centralnog NodeMCU uređaja, preko UART (Universal asynchronous receiver-transmitter) komunikacije.

Šema jedinice sistema koja je u skladu sa prethodno napisanim je predstavljena na slici 7:



Slika 7 - Šema jedinice sistema

Raspored veza (priključaka) između komponenti i glavnog NodeMCU-a određuju karakteristike izvedenih GPIO (General Purpose Input/Output) veza sa ESP8266 na njega. U tabeli 15 su prikazani pinovi, njihove karakteristike, namjene, moguće upotrebe i ograničenja:

Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Tabela 2 - Ograničenja upotrebe dostupnih ESP8266 GPIO pinova ⁵

Veze sa modula (RTC i SD kartica) su povezani na podrazumijevane pinove (one od njih koje su morali biti neophodno vezani za određene). Izuzetak su CS veza (Chip Select) za SD karticu (koja može biti vezana za bilo koji pin koji omogućava digitalni izlazni signal) I SQW (programabilni generator periodičnog digitalnog signala) veza sa RTC sata koja će se koristiti za izazivanje sistemskog prekida (interrupt). SQW žica generiše signal u svakom trenutku, pa je taj signal prisutan i u toku boot

⁵ Izvor: [15]

procesa na ESP8266, što može izazivati probleme. Za te dvije žice postoje sledeći slobodni pinovi sa datim ograničenjima [15]:

- D0 – slobodno za CS, nemoguće za SQW,
- D3 – slobodno za CS, invertovan signal za SQW, neuspješan boot za SQW
vrijednost = logička 0,
- D4 – slobodno za CS, invertovan signal za SQW, neuspješan boot za SQW
vrijednost = logička 0, na liniji prikopčana dioda! i
- D8 – podrazumijevano za CS, nemoguće za SQW.

Na osnovu prethodnih podataka, vidno je da je nemoguće da SQW signal ponekad (očekivanje = 0.5) izazove neuspješan boot na ESP8266. Bez obzira na to, izabrana je najbolja moguća opcija, a to je CS na D8, a SQW na D3.

4 Implementacija

4.1 SdFat biblioteka

Komunikacija sa memorijskom karticom se obavlja uz pomoć SdFat [16] biblioteke, verzija 1.1.4. SdFat biblioteka u okviru sebe ima definisanu File klasu koja je u koliziji sa File klasom iz Arduino Core biblioteke FS.h [10]. FS.h je korišćena u okviru ESP8266WiFi.h [10] biblioteke koja se koristi za uspostavljanje konekcije na određenu WiFi mrežu.

Navedeni problem je razriješen tako što je SdFat biblioteka modifikovana – klasa File je preimenovana na SdFatLibFile, i sve reference na tu klasu su prepravljene [17].

4.2 Konfiguracija glavnog-upravljačkog NodeMCU-a

Softver dozvoljava oređeni nivo parametrizacije koji se tiče omogućavanja njegovog normalnog rada i prilagođavanja specifičnim potrebama u datoj situaciji. Sve te konstante se definišu u cfg.hpp zaglavlju (listing 1).

```
#ifndef _CONFIG_HPP_
#define _CONFIG_HPP_

#define MILLISECOND_PRECISION true

#define INTERNAL_LED_BLINK_INTERVAL 200
#define CHANNEL_SWITCH_INTERVAL 200
#define FILE_SAVE_INTERVAL 10000
#define RTS_COMMAND_SWITCHING_INTERVAL 16000

#define SD_CS_PIN D8
#define SQW_INTERRUPT_PIN D3
#define ESP8266_ONBOARD_LED D4

#define WIFI_SSID "Igor"
#define WIFI_PASSWD ""

#endif
```

Listing 1 – Konfiguracioni fajl (cfg.hpp)

Sistem dobija tačno vrijeme tako što se pri pokretanju sinhronizuje sa određenim NTP (Network Time Protocol) serverom. Za tu sinhronizaciju je potrebna konekcija na internet, pa se pomoću konstanti sa WIFI_ prefiksom definiše naziv (SSID) mreže i njena lozinka (Pre-Shared Key). Sinhronizacija i kasnije praćenje vremena je moguće

u preciznosti na nivou milisekunde. U zavisnosti od potreba sistema, moguće je birati praćenje podataka na nivou milisekunde, ili, u suprotnom, na nivou sekunde, pomoću konstante `MILLISECOND_PRECISION`. Sinhronizacija vremena do nivoa milisekunde se obavlja zahvaljujući biblioteci `ESPNTpClient` [18], a kasnije precizno praćenje protoka milisekundi se može obezbijediti pomoću ranije pomenutog programabilnog generatora signala na DS3231 RTC satu, preko njegovog SQW pina. Konfiguracija omogućava podešavanje veza za SQW pin i CS pin, koji su unaprijed podešeni na vrijednosti koje su izabrane kao optimalne u poglavlju 3.2.3.

U poglavlju 3.2.3 je takođe utvrđeno da se sistem neće pri svakom pokretanju pokrenuti ispravno, pa se iz tog razloga uvodi indikator koji će pokazivati stanje sistema. Za te potrebe je izabrana dioda koja je ugrađena na ESP8266, i povezana na D4 pin na Dev Kit-u. Moguće je signal stanja sistem preusmjeriti na neki drugi slobodni pin. Kada sistem nije u zdravom stanju, moguće ga je ručno dovesti u zdravo stanje pomoću pritiska na RST (Reset) taster (efektivno izaziva ponovno pokretanje sistema) koji je prisutan na NodeMCU ploči.

Ostale konfiguracione konstante se odnose na stanje sistema u kome je on već prešao u aktivni rad i prikupljanje podataka. Tu se definišu vremenski intervali za:

- Indikator zdravlja signala – brzina promjene stanja indikatora – lampice,
- Skok na drugi kanal – zadržavanje na pojedinačnom kanalu u 2.4GHz spektru,
- Čuvanje podataka na memorijskoj kartici – aktivnost oduzima mnogo sistemskog vremena pa se radi periodično i
- Promjenu ciljnog uređaja na RTS odašiljaču – interval izdavanja nove komande sekundarnom NodeMCU-u.

4.3 NTP sinhronizacija vremena

`ESPNTpClient` biblioteka obavlja sinhronizaciju vremena asinhronim putem. To nije željeno ponašanje u našem slučaju, jer prikupljanje podataka u našem sistemu može da počne tek kada je sinhronizacija vremena uspješno urađena. Zbog toga je bilo potrebno napraviti mehanizam koji će omogućiti čekanje na sinhronizaciju.

Kreirana je enumeracija sa osnovnim stanjima NTP klijenta koji su relevantni za naše potrebe (listing 2).

```
typedef enum {  
    TIME_SYNCED,  
    TIME_SYNCING,  
    ERROR,  
    IDLE  
} NTPClient_State_t;
```

Listing 2 – Enumeracija stanja NTP klijenta

U funkciji pokretanja NTP klijenta, prijavimo metodu koja će, između ostalog, pozivati metodu za procesiranje svakog događaja u `ESPNTpClient` biblioteci (listing 3).

```

void startNTPClient(uint16_t milli_ntp_timeout, int sync_period,
                  const char * TZ, const char * ntp_server_nullable) {
    NTP.onNTPSyncEvent ([]) (NTPEvent_t event) {
        ntpEvent = event; // hold last triggered event
        processSyncEvent(event);
    });
    ...
}

```

Listing 3 – Registracija obrađivača događaja u metodi za pokretanje NTP klijenta

Metoda za procesiranje, na osnovu događaja u biblioteci definiše trenutno stanje iz prethodno predstavljene enumeracije (listing 4).

```

NTPClient_State_t ntpClientState = IDLE;

void processSyncEvent(NTPEvent_t ntpEvent) {
    switch (ntpEvent.event) {
        case timeSyncd:
        case syncNotNeeded:
            ntpClientState = TIME_SYNCED;
            break;
        case requestSent:
        case partlySync:
            ntpClientState = TIME_SYNCING;
            break;
        default:
            ntpClientState = ERROR;
            break;
    }
    if(additionalEventHandler != nullptr) additionalEventHandler(ntpEvent);
}

```

Listing 4 – Obradivač događaja dobijenih iz ESPNtpClient biblioteke

Sada, na osnovu stanja zabilježenog u ntpClientState promjenljivoj možemo implementirati funkciju čekanja na uspješnu sinhronizaciju vremena sa NTP serverom (listing 5).

```

#define SYNCING_PRINT_INTERVAL 1000

void waitNTPClientSync() {
    unsigned long time = millis();
    if(Serial) Serial.println("NTP Client Sync in progress.");
    while(getNTPClientState() != TIME_SYNCED) {
        if(Serial && millis() - time > SYNCING_PRINT_INTERVAL) {
            Serial.printf(".");
            time = millis();
        }
    }
}

```

Listing 5 – funkcija koja implementira čekanje na sinhronizaciju vremena sa NTP serverom

4.4 Rad sa RTC satom

DS3231 ima mogućnost praćenja vremena u preciznosti sekunde, bez direktne podrške za milisekunde. Međutim, moguće postaviti generisanje pobudnog signala na frekvenciji 1024Hz (1024 pobude u toku jedne sekunde). Taj izlaz, SQW, je moguće koristiti kao izazivač sistemskih prekida. Obradivač tog sistemskog prekida će pratiti protok vremena na osnovu prekida i tako sam čin preciznog praćenja vremena u preciznosti milisekunde prebacuje na glavni NodeMCU. U radu sa DS3231 satom je korištena istoimena biblioteka [23].

4.4.1 Inicijalizacija RTC sata sa tačnim vremenom (u preciznosti sekunde)

ESPNTpClient biblioteka ne pruža dovoljnu fleksibilnost u radu sa dobijenim vremenom. Zbog toga se iz nje vremenski podaci uvoze u biblioteku TimeLib [19]. Potrebno je sačekati uspješnu sinhronizaciju TimeLib biblioteke sa vremenom uzetim iz NTP klijenta (listing 6).

```
// setting internal timelib helper
setSyncProvider(NTPUnixTics);
while (timeStatus() != timeSet);
```

Listing 6 – Sinhronizacija vremena u TimeLib biblioteci

Nakon toga se prelazi na postavljanje vremena na RTC satu, gdje se koriste funkcije i makroi iz TimeLib biblioteke za dobavljanje podataka o vremenu koji se predaju pozivima metoda iz DS3231 biblioteke (listing 7).

```
void setDS3231Time() {
    Clock.setYear(tmYearToY2k(CalendarYrToTm(year())));
    Clock.setMonth(month());
    Clock.setDate(day());
    Clock.setDoW(weekday());
    Clock.setClockMode(false); // set to 24h
    //setClockMode(true); // set to 12h
    Clock.setHour(hour());
    Clock.setMinute(minute());
    Clock.setSecond(second());
    ...
}
```

Listing 7 – Postavljanje vremena na RTC satu (DS3231)

4.4.2 Praćenje vremena SQW interrupt-om (u preciznosti milisekunde)

U trenutku kada utvrdimo tačno vrijeme, imamo broj koji predstavlja broj milisekundi koje su protekle od početka epohe pa do baš tog trenutka. Početak epohe je definisan kao 01.01.1970. 00:00:00 0ms. Ako tog trenutka, kada smo sinhronizovali vrijeme, počnemo da pratimo broj milisekundi koje protiču kao rezultat pobudnog signala na

svaku milisekundu, zbirom ta dva broja uvijek dobijamo tačno vrijeme izraženo u milisekundama od početka epohe.

Kako je frekvencija pobudnog (interrupt) signala 1024Hz, to znači da ipak nemamo pobudu tačno svake milisekunde, već svake 0.9765625ms. Taj problem rješavamo programski, tako što pri prijemu interrupt-a, funkcija koja vrši obradu sistemskog prekida (ISR, Interrupt Service Routine), odlučuje da li će dati interrupt računati kao proticanje jedne milisekunde, ili će je preskočiti. Odluka se donosi tako što se razmatra koja vrijednost će biti bliža realnoj vrijednosti, ako se doda jedna milisekunda, ili ako se preskoči (listing 8).

```
uint16_t int_freq = 1024;
double milli_period = double(int_freq) / 1000;
uint16_t floor_freq = floor(milli_period);
uint16_t ceil_freq = ceil(milli_period);

volatile unsigned long long counter = 0;
volatile unsigned long long millisecond_isr_counter = 0;
volatile unsigned long long milli_isr_total = 0;

void IRAM_ATTR isr_func() {
    ++counter;
    ++millisecond_isr_counter;
    if (millisecond_isr_counter == ceil_freq ||
        (millisecond_isr_counter == floor_freq &&
         (double(counter) - milli_period * (milli_isr_total + 1))
         >= 0.0)) {
        millisecond_isr_counter = 0;
        ++milli_isr_total;
        ++espRTCTime;
    }
}
```

Listing 8 – Obradivač sistemskog prekida izazvan od strane RTC sata (SQW generatora)

ISR funkcija mora biti označena sa `IRAM_ATTR`, što znači da će ona biti smještena u RAM, a ne u flash memoriju ESP8266.

4.5 Inicijalizacija sistema

U inicijalizaciji sistema se obavljaju sve pripreme dovođenja sistema u aktivno stanje prikupljanja podataka. Cjelokupni kod te funkcije je prikazan na listingu 9, i obuhvata:

- inicijalizaciju veza – pripremanje pinova za komunikaciju,
- otvaranje serijske komunikacije (sa drugim NodeMCU-om),
- povezivanje na WiFi mrežu,
- sinhronizacija vremena sa NTP serverom,
- postavljanje vremena na RTC sat i početak praćenja protoka vremena,

- povezivanje sa memorijskom karticom i otvaranje fajla za logovanje na njoj i
- registracija metode za prikupljanje podataka.

```
void setup() {
    // setting up pin modes
    pinMode(SD_CS_PIN, OUTPUT);
    pinMode(SQW_INTERRUPT_PIN, INPUT_PULLUP);
    pinMode(ESP8266_ONBOARD_LED, OUTPUT);
    digitalWrite(ESP8266_ONBOARD_LED, LOW);
    digitalWrite(SD_CS_PIN, LOW);

    // setting up serial comms
    Serial.begin(115200); while (!Serial);

    // connecting to wifi
    clientSetupWiFi();
    clientConnectWiFi(WIFI_SSID, WIFI_PASSWD);

    // setup I2C interface
    Wire.begin();
    //Clock.enable32kHz(true);
    if(!Clock.oscillatorCheck()) clockPresent = false;

    // interfacing with RTC oscillators
    if(clockPresent) {
        Clock.enable32kHz(false);
        Clock.enableOscillator(true, false, 1);
    }

    // synchronising time
    registerNTPEventHandler(addEHSerialPrintEvent);
    startNTPClient();
    waitNTPClientSync();

    // setting internal timelib helper
    setSyncProvider(NTPUnixTics);
    while (timeStatus() != timeSet);

    // setting RTC
    if(clockPresent) {
        attachInterrupt(digitalPinToInterrupt(SQW_INTERRUPT_PIN),
                        isr_func, RISING);

        setDS3231Time();
    }

    // stopping ntp client's resync jobs
    stopNTPClient(false);
}
```

```

// disconnecting from wifi
clientDisconnectWiFi();

// SPI interface to SD card
sdCardPresent = SD.begin(SD_CS_PIN, SPI_HALF_SPEED);

// open file and prepare for writing
sprintf(filename, "%llu.txt", getPrefferedTime());
openFile();

// start logging
logTime();
rts_switch_command();
sniffer_init(channel, probe_request_cb_handler);
}

```

Listing 9 – Inicijalizacija sistema

Na početku se pinovi CS, SQW i pin vezan za diodu na ESP8266 ploči, pripremaju za upotrebu u skladu sa svojom namjenom. U poglavlju 3.2.3, na tabeli 2 je pokazano da pinovi D3 i D4 (SQW i onboard LED) vezani za pull up otpornik, što znači da se na njih primjenjuje invertovana logika. Zbog toga se upotreba SQW pina označava sa INPUT_PULLUP. U poglavlju 4.2 je opisana upotreba LED diode vezane na pin D4. U toku inicijalizacione procedure ta lampica treba da bude upaljena, a s obzirom na invertovanu logiku i na tom pinu (poglavlje 3.2.3), postavljamo na nju vrijednost logičke nule (LOW).

Priprema RTC sata za upotrebu podrazumijeva gašenje 32kHz oscilatora (jer nam on nije potreban), i paljenje oscilatora sa oznakom 1, što je 1.024kHz oscilator koji je nama potreban. On postaje odmah dostupan na SQW pin-u. Uz postavljanje vremena na RTC satu, registrujemo interrupt za SQW pin tako što na njega povezujemo našu ISR metodu (isr_func) i definišemo da se okidanje registruje na rastućoj ivici square wave signala.

Bitno je zaustaviti NTP klijent i diskonektovati se sa WiFi mreže, jer samo u tom slučaju se smije registrovati metoda za obradu pristiglih 802.11 paketa. Zaustavljanje NTP klijenta je neophodno jer se ne smije dozvoliti ponovni pokušaj osvježavanja vremena – ponovni pokušaj pristupa internetu u toku slušanja paketa prouzrokuje fatalnu grešku.

4.6 Aktivni režim rada glavnog NodeMCU uređaja

Slušać 802.11 paketa je prijavljen u inicijalizacionoj fazi i on funkcioniše u pozadini preko sistemskih prekida. O njemu će biti riječi tek u narednom poglavlju.

Glavna linija izvršavanja sistema se bavi periodičnim obrađivanjem poslova (listing 10) koji su već pomenuti u okviru poglavlja 4.1 gdje je bila riječ o konfiguraciji. Tu su bili konfigurisani periodi za periodične poslove, a to su: indikacija stanja sistema, skokovi

na druge kanale, čuvanje podataka na memorijskoj kartici i izmjena komande RTS odašiljaču.

```
void loop() {
    probe_request_final_handler();
    loop_led_blinking();
    channel_hopping();
    sd_file_saving();
    rts_switching();
    yield();
}
```

Listing 10 – Periodično obrađivanje poslova u beskonačnoj petlji sistema

Sve četiri te funkcije su implementirane na sličan način, pomoću generičkog tajmera koji se izvršava željenu metodu samo ako je od njenog prethodnog izvršavanja prošao definisani vremenski interval. Implementacija generičkog tajmera se može vidjeti na listingu 11.

```
void generic_timer(unsigned long * last, unsigned long current,
                  unsigned long desired_interval, void (*handler)()) {
    if((current - *last) < desired_interval) return;
    *last = current;
    handler();
}
```

Listing 11 – Implementacija generičkog tajmera

Upotreba tajmera se može demonstrirati na primjeru LED lampice koja je indikator zdravog stanja sistema. U toku zdravog stanja sistema ugrađena LED dioda treba da miga, što znači da se implementacija toga svodi na periodično invertovanje njenog stanja (listing 12).

```
void led_blinker() {
    static uint8_t next_state = HIGH;
    digitalWrite(ESP8266_ONBOARD_LED, next_state);
    next_state = next_state == LOW ? HIGH : LOW;
}

void loop_led_blinking() {
    static unsigned long last = millis();
    generic_timer(&last, millis(), INTERNAL_LED_BLINK_INTERVAL, led_bli
nker);
}
```

Listing 12 – Upotreba generičkog tajmera za periodičnu promjenu stanja indikatora zdravlja jedinice sistema

Generičkom tajmeru se predaje adresa na promjenljivu statičkog vijeka trajanja koja nosi vrijednost posljednjeg izvršavanja ciljanog posla, sledeći parametar je trenutno vrijeme, treći vremenski interval – period izvršavanja, i četvrti je pokazivač na void metodu bez parametara koja obavlja taj posao.

4.7 Prikupljanje 802.11 paketa i njihova obrada

Obrada primljenih paketa u promiscuous režimu se obavlja tako što se registruje funkcija koja će biti obrađivač paketa. U [11] je naglašeno da trajanje ovih funkcija ne treba biti dugo, jer to izaziva propuštanje paketa koji bi mogli biti primljeni u toku obrađivanja nekog paketa. Čuvanje izdvojenih podataka je dugotrajna operacija, tako da je svakako potrebno implementirati baferovanje podataka prije njihovog čuvanja na memorijsku karticu. Postoji opcija i baferovanja samih paketa, prije njihove obrade, ali to nije uzeto u obzir u implementaciji, jer bi baferovanje bafera paketa bilo izrazito memorijski skupocjeno, a korist je vjerovatno ograničena.

Prikupljeni podaci se čuvaju u strukturi koja sadrži sve podatke iz Probe Request paketa koji mogu biti od značaja pri praćenju i identifikaciji, a uz to sadrži i polje u koje se upisuje vrijeme prijema paketa (listing 13).

```
struct probe_request {
    uint64_t millitime;
    RxControl radiotap_header;
    uint8_t source_address[6];
    uint8_t transmission_address[6];
    uint8_t destination_address[6];
    uint8_t receiver_address[6];
    uint8_t bssid_address[6];
    struct frame_sequence_control_t seq_ctrl;
};
```

Listing 13 – Struktura za čuvanje prikupljenih podataka od značaja iz Probe Request paketa

Po prijemu paketa, poziva se tzv. callback metoda koja započinje obradu paketa (listing 14).

```
void sniffer_handler_func(uint8_t *buff, uint16_t len) {
    // Filter unreliable packets (HT40, LDPC)
    if(len <= sizeof(struct RxControl)) return;

    const struct promiscuous_mode_pkt * ppkt = (struct promiscuous_mode_pkt *) buff;
    struct probe_request pr;
    bool success = probe_request_filter(ppkt, len, &pr);
    if(success && sniffer_pr_cb != nullptr) {
        pr.radiotap_header = ppkt->rx_ctrl;
        sniffer_pr_cb(&pr);
    }
}
```

Listing 14 – Funkcija obrađivač primljenih 802.11 paketa

Bitno je napomenuti da se callback metoda poziva za sve primljene tipove 802.11 paketa, a čak i za nevalidne, pa se prvo mora pristupiti filtriranju paketa. Odmah počinje da se konstruiše probe_request struktura i da se popunjava podacima. Ako se ispostavi da je paket validan Probe Request, tada će se pozvati metoda (u prethodnom listingu

označena kao `sniffer_pr_cb`) koja je u inicijalizacionoj fazi sistema prijavljena kao funkcija koja će dovršavati izdvajanje podataka. Njene obaveze se svode na bilježenje tačnog vremena i snimanje podataka (listing 15). Kao što je već navedeno, snimanje podataka se ne vrši direktno, već se baferuje (pomoćnom standardnom generičkom strukturom deque).

```
std::deque<struct probe_request> prbuff;

void probe_request_cb_handler(struct probe_request * ppr) {
    ppr->millitime = getPrefferedTime();
    prbuff.push_back(*ppr);
}
```

Listing 15 – Obradivač koji registruje vrijeme i smješta podatke u bafer (STL struktura Deque)

Od tog trenutka se nastavlja prikupljanje paketa, a krajnje čuvanje podataka se prepušta glavnoj liniji izvršavanja. U poglavlju 4.6 se u glavnoj izvršnoj funkciji loop vidi da s pri svakom pozivu iste, poziva i `probe_request_final_handler`. On vrši finalno čuvanje podataka i prazni obrađene Probe Request-ove iz deque-a (listing 16).

```
void probe_request_final_handler() {
    if(prbuff.empty()) return;
    struct probe_request pr = prbuff.front();
    char temp[256] = {0};
    char recv[20] = {0};
    char dest[20] = {0};
    char trans[20] = {0};
    char src[20] = {0};
    char bssid[20] = {0};
    macprint(pr.receiver_address, recv);
    macprint(pr.destination_address, dest);
    macprint(pr.transmission_address, trans);
    macprint(pr.source_address, src);
    macprint(pr.bssid_address, bssid);
    sprintf(temp, "PR timestamp %llu channel %d
                |recv->%s|dest->%s|trans->%s|src->%s|bssid->%s|",
            pr.millitime, channel, recv, dest, trans, src, bssid);
    prbuff.pop_front();
    logln(temp);
}
```

Listing 16 – Krajnji obrađivač prikupljenih podataka, formiranje loga

Samo ako ima Probe Request-ova na raspolaganju, pokreće se čuvanje jednog (najstarijeg) od njih. Nema potrebe za čuvanjem više njih, jer se ova metoda u programskoj petlji poziva u svakoj njenoj iteraciji, a osim toga, jedinično čuvanje je bolje jer omogućuje preciznije izvršavanje periodičnih poslova.

4.8 Slanje komandi RTS odašiljaču

Podaci se šalju periodično RTS odašiljaču. Ideja je da se u svakom period izda komanda koja će da se tiče jednog uređaja. U svakoj sledećoj periodi se ciklično prolazi kroz listu ciljanih uređaja. Primjer niza koji sadrži jednu takvu komandu se može vidjeti na listing 17.

```
{'D',0x40,0x4e,0x36,0x3a,0x45,0x1b, 'S',0x00,0x11,0x22,0x33,0x44,0x55}
```

Listing 17 – Komanda (niz) koja se šalje RTS odašiljaču

Komanda se sastoji iz dva dijela. Prvi počinje sa znakom „D” i ukazuje na to da slijedi MAC adresa destinacije RTS paketa. Zatim, drugi dio započinje znakom „S” koji ukazuje da u nastavku slijedi MAC adresa porijekla RTS paketa koji treba poslati.

4.9 RTS odašiljač

U poglavlju 3.2.3 je opisano kako je ipak moguće slati 802.11 freedom RTS pakete u određenoj implementaciji koja ne vrši provjeru tipa paketa koji se šalje. Ipak, prisutna je provjera da paket mora imati minimalnu dužinu od 24 bajta, što je znatno više od običnog RTS paketa, kao što je prikazano u poglavlju 2.1.4 na slici 5 gdje je ukupna dužina paketa 16 bajta.

```
// RTS Packet buffer
uint8_t packet[128] = { 0xb6, 0xb0, 0x00, 0x00,
                        0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
                        0xee, 0xee, 0xee, 0xee, 0xee, 0xee,
                        0xaa, 0xaa, 0xaa, 0xaa,
                        0xaa, 0xaa, 0xaa, 0xaa,
                        0xaa, 0x00};
```

Listing 18 – RTS paket

Tu prepreku prevazilazimo tako što u paket dodajemo beznačajne podatke u obliku 0xaa bajta, kao što se može vidjeti na listing 18, u četvrtom i petom redu paketa. Prva dva bajta paketa predstavljaju Frame Control polje, u kome je definisan tip - RTS paket. Sledeća dva bajta predstavljaju Duration polje, koje ima vrijednost 0. Drugi red je prostor za MAC adresu primaoca, a treći je prostor za MAC adresu primaoca.

RTS odašiljač sluša serijsku komunikaciju na kojoj očekuje komande od glavnog NodeMCU-a. Kada primi podatke koji su u formatu koji je opisan u poglavlju 4.8 i prikazan na listing 17, dakle počinje sa znakom „D”, on tumači te podatke i upisuje ih na odgovarajuća polja u paketu (listing 19). To znači da prvu adresu kopira na lokacije počev od bajta 4 (drugi red), a drugu adresu kopira na adrese počev od reda 3 (treći red).

```
#define PACKET_DEST_IND 4
#define BUFFLEN 6
int iter = 0;
```

```

void ICACHE_FLASH_ATTR uart_rx_task(os_event_t *events) {
    if (events->sig == 0) {
        // Sig 0 is a normal receive.
        // Get how many bytes have been received.
        uint8_t rx_len = (READ_PERI_REG(UART_STATUS(UART0)) >>
                        UART_RXFIFO_CNT_S) & UART_RXFIFO_CNT;

        uint8_t rx_char;
        uint8_t ii;
        uint8_t srcnow = 0;
        for (ii=0; ii < rx_len; ii++) {
            rx_char = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;
            if(ii==0 && rx_char!='D') break;
            else if(ii == 14) break;
            else if(ii == 7 && rx_char=='S') {
                srcnow = 1;
                continue;
            }
            else if(ii==0 || ii==7) continue;
            iter = ii-1-srcnow;
            packet[PACKET_DEST_IND + iter] = rx_char;
            os_printf("iter %d SERIAL %02x = %c and datalen %d \n",
                    iter, rx_char, rx_char, rx_len);
        }
        // Clear the interrupt condition flags and
        // re-enable the receive interrupt.
        WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR |
UART_RXFIFO_TOUT_INT_CLR);
        uart_rx_intr_enable(UART0);
    }
}

```

Listing 19 – Funkcija obrađivač primljenih poruka serijskim putem (UART)

U verzijama SDK u kojima je funkcija za slanje freedom paketa zvanično prisutna, postoji i mehanizam koji potvrđuje da je paket poslat. To je realizovano tako što se pomoću funkcije

```
int wifi_register_send_pkt_freedom_cb(freedom_outside_cb_t cb); ,
```

gdje je

```
typedef void (*freedom_outside_cb_t)(uint8 status); ,
```

registruje callback funkcija koja će biti pozvana tek kada je 802.11 freedom paket uspješno poslat. To je bitno, jer u slučaju kada se pokuša slanje paketa, a prethodni paket još nije poslat, posljedica je da nijedan paket neće biti poslat. Nažalost, u SDK 1.3.0, ne postoji ova funkcionalnost, tako da se između slanja mora praviti pauza da bi se osiguralo stvarno isporučivanje paketa. U ovom radu nije istraživano koliki je minimalni vremenski interval, već je izabrana vrijednost od 1250ms koja se pokazala kao više nego dovoljna da svaki paket bude isporučen.


```

void sender(void *arg) {
    const char * pmac = packet + PACKET_DEST_IND;
    const char * pmac1 = packet + PACKET_DEST_IND + 6;
    os_printf("Send status: %d on channel %d
               dest %02x:%02x:%02x:%02x:%02x:%02x and
               src %02x:%02x:%02x:%02x:%02x:%02x\n",
               wifi_send_pkt_freedom(packet, 25, 0),
               channel, pmac[0], pmac[1], pmac[2],
               pmac[3], pmac[4], pmac[5], pmac1[0],
               pmac1[1], pmac1[2], pmac1[3],
               pmac1[4], pmac1[5]);
    next_channel();
    light_switch();
}

```

Listing 20 – Funkcija koja vrši slanje RTS paketa

Funkcija slanja paketa (listing 20) se poziva periodično, sa već naglašenom pauzom, i nakon slanja paketa se odmah prelazi na sledeći kanal na kome će ponovo biti poslat isti paket, ili, ako je u međuvremenu došla nova komanda, novi paket. Zbog ovoga, bitno je da na glavnom NodeMCU interval slanja komandi RTS odašiljaču bude minimalno jednak broju kanala (u ovom radu 13) * pauzi između slanja RTS paketa (ovdje, 1250).

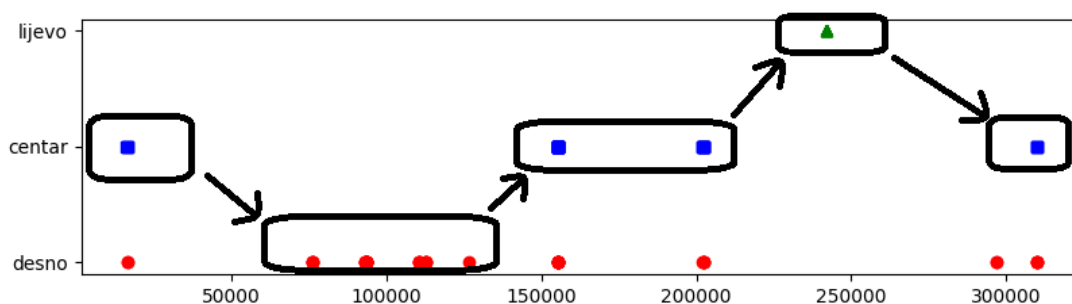
5 Rezultati i diskusija

5.1 Praćenje mobilnih uređaja nezavisno od RTS odašiljača

Cjelokupni sistem je testiran tako što su tri jedinice postavljene na tri lokacije koje su u kolinearnom odnosu. Te lokacije će u nastavku biti označene kao „centar“, „lijevo“ i „desno“. Lokacija „desno“ se nalazi relativno bliže lokaciji „centar“, nego što je to slučaj sa lokacijom „lijevo“. Zbog toga, „centar“ i „desno“, često imaju preklapanja u detekciji uređaja, jer često oba detektuju isti Probe Request paket. Ipak, moguće je razlučiti kojoj lokaciji je mobilni uređaj bliži na osnovu broja detektovanih Probe Request-ova.

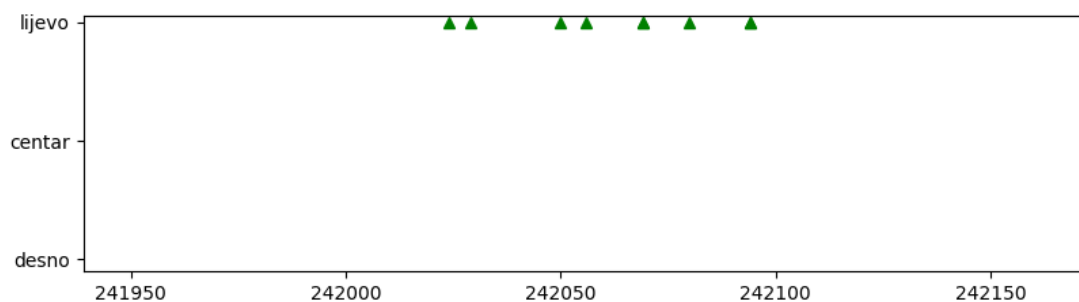
Test je sproveden nošenjem mobilnih uređaja između lokacija u sledećem redoslijedu:

centar → desno → centar → lijevo → centar.



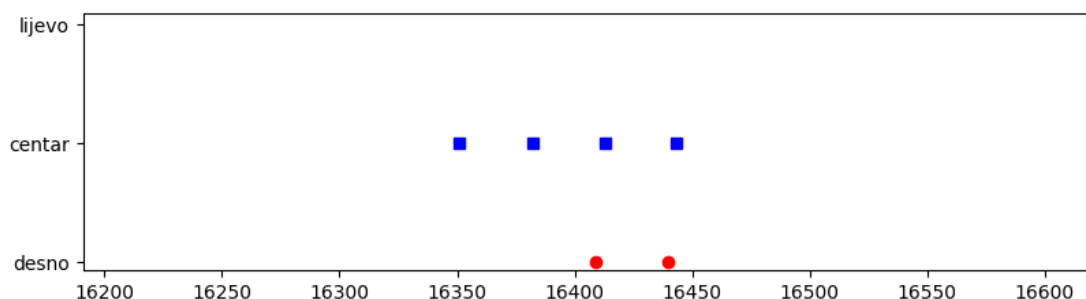
Grafikon 21 – Rezultati testa praćenja jednog mobilnog uređaja

Na x osi sa grafikona 21 je prikazan protok vremena u milisekundama u toku sprovođenja testa, dok su na y osi prikazane lokacije jedinica sistema. Svaka oznaka na grafiku predstavlja informaciju da je u datom trenutku na datoj lokaciji detektovano prisustvo praćenog mobilnog uređaja. Jedna oznaka na grafiku predstavlja jedan ili više Probe Request paketa. To se može vidjeti na grafikonu 22 gdje se u većem nivou detalja prikazuje detekcija uređaja na lokaciji „lijevo“ sa prethodnog grafika.



Grafikon 22 – Prikaz broj Probe Request-ova u jednoj grupaciji

Već je pomenuto da zbog blizine lokacija „centar“ i „desno“ se mobilni uređaj detektuje na obje lokacije, međutim, lako je uočljivo kojoj lokaciji je uređaj bliži jednostavnim prebrojavanjem broj registrovanih Probe Request-ova. Ako posmatramo početni trenutak testa kada se nalazimo u lokaciji „centar“, vidimo da je jedinica „centar“ detektovala više paketa od jedinice „desno“. To je prikazano na grafikonu 3.



Grafikon 23 – Prikaz istovremeno primljenih Probe Request-ova na dvije jedinice sistema

Navedeni rezultati nam ukazuju da je praćenje kretanja mobilnih uređaja veoma izvodljivo sa predloženim sistemom.

5.2 RTS odašiljač

Testiranje je sprovedeno više puta sa različitim konfiguracijama RTS odašiljača, da bi se utvrdilo njegovo dejstvo na broj registrovanih Probe Request paketa. Postojale su tri konfiguracije različite konfiguracije.

U prvoj konfiguraciji RTS odašiljač je bio isključen, to nam omogućava praćenje uređaja potpuno pasivnom metodom i predstavlja referentno mjerenje za testiranje uticaja RTS odašiljača.

Druga konfiguracija definiše aktivno slanje RTS paketa ka ciljanim uređajima, ali se na mjestu porijekla paketa nalazi uređajima nepoznata MAC adresa.

Treća konfiguracija definiše ponovo aktivno slanje RTS paketa, ali se na mjestu porijekla paketa stavlja MAC adresa mreže koja je ciljanom uređaju poznata – povezan je na nju.

Rezultati ne pokazuju promjene u broju poslatih Probe Request-ova koje bi mogle dovesti do jednoznačnog zaključka. Vidan je manji pad u broju Probe Request-ova u trećoj konfiguraciji, što, u slučaju da to nije samo greška u mjerenju, bi se moglo objasniti time da su ovi paketi direktno ometanje mreže na koju je ciljani uređaj povezan, i komunikacije između mreže i ciljanog uređaja. Moguće je da mobilni uređaj nije poštuovao Duration polje u RTS paketu (koje ima vrijednost 0), pa da je ipak čekao na prijem paketa što je oduzelo vrijeme u toku kojeg bi inače bio poslat Probe Request paket.

Bitno je napomenuti da su za precizno testiranje uticaja RTS odašiljača potrebni posebni uslovi – izolovano okruženje od smetnji i veoma precizno praćenje mrežnog saobraćaja. Te uslove nije bilo moguće obezbijediti u ovom testiranju.

5.3 MAC randomizacija

MAC randomizacija je još uvijek nestandardizovani metod kojim mnogi proizvođači mobilnih uređaja pokušavaju da onemoguće praćenje. MAC randomizacija u paketima mijenja MAC adresu porijekla paketa, te tako ne postoji jedinstveni identifikator uređaja. Ta tehnika veoma dobro radi kada mobilni uređaj nije asociran sa WiFi mrežom, međutim, rezultati u našim testovima su pokazali da kada je mobilni uređaj povezan na neku WiFi mrežu, tada se randomizovana MAC adresa zadržava dok god je uređaj povezan na mrežu. Zbog takvog ponašanja, ovaj sistem je u mogućnosti da prati uređaje i u takvoj situaciji.

U opisanoj implementaciji MAC randomizacije, ovaj sistem nije efektivan za praćenje uređaja u okruženju kada oni nisu povezani na bilo koju WiFi mrežu. Ovaj sistem, ipak, u svojim izdvojenim podacima pruža dovoljno informacija tako da bi praćenje bilo moguće ostvariti. O tome će biti riječi u poglavlju 6 – Budući rad.

Standardizacija MAC randomizacije je u toku, trenutno se nalazi u Internet-Draft fazi [20].

6 Budući rad

Kao što je u poglavlju 5.3 već implicirano, postoje metodi za prevazilaženje MAC randomizacije. U toku implementacije ovog sistema to je uzeto u obzir, i u okviru `probe_request` strukture, koja je predstavljena u poglavlju 4.7 na listingu 13, su dodavani svi podaci koji mogu dati informacije koji bi mogli pomoći u identifikaciji, nevezano za samu MAC adresu porijekla paketa.

Sa MAC randomizacijom, problem identifikacije uređaja se svodi na klasterovanje više MAC adresa koje određeni uređaj koristi u jednu grupu koja će predstavljati taj uređaj.

Jedan način bi bio unapređivanje metoda predloženog u radu [21], gdje bi se RSS (Received Signal Strength) informacija mogla koristiti za grupisanje. RSS informacija se nalazi u okviru `RxControl` podstrukture unutar `probe_request` strukture.

Drugi način bi bio implementacija metoda izloženog u radu [22], gdje se koristi tačno vrijeme prijema `Probe Request` paketa za klasterovanje istih u grupe pomoću metoda baziranog na KNN (k-Nearest Neighbors) algoritmu. U ovom radu je značajna pažnja posvećena preciznom praćenju vremena, pa bi implementacija metoda prikazana u tom radu trebala dati i u ovom slučaju veoma dobre rezultate.

Dodatno je moguće unaprijediti rad tako što će jedinice sistema periodično samostalno slati prikupljene podatke na neku centralnu lokaciju. Prepreka tome jeste što bi takav sistem zahtijevao mrežnu konekciju u toku svog cjelokupnog režima rada, i to što bi se prikupljanje podataka moralo prekinuti da bi se jedinica povezala na mrežu i poslala podatke. Tek nakon slanja podataka se mogu ponovo prikupljati informacije.

7 Zaključak

Rad je pokazao da je praćenje kretanja mobilnih uređaja moguće i da je veoma lako izvodljivo sa izrazito jeftinom i pristupačnom platformom kao što je ESP8266. Način na koji je postignuto praćenje jeste primarno pasivnim slušanjem IEEE 802.11 komunikacije iz okoline i izvajanjem MAC adresa porijekla iz Probe Request paketa.

Postoji dosta prostora za dalji razvoj i istraživanje. Nadogradnje sistema bi omogućile praćenje i onih uređaja koji vrše MAC randomizaciju, ali i povećale autonomnost cjelokupnog sistema pa tako omogućile komercijalizaciju ovakvog rješnja.

Ovaj rad, i implementacija samog projekta, su dostupni na:

<https://github.com/FmasterofU/bachelors-thesis> .

8 Literatura

- [1] Wi-Fi Alliance - <https://www.wi-fi.org/>
- [2] IEEE 802 LAN/MAN Standards Committee - <https://www.ieee802.org/>
- [3] Matthew S. Gast, „802.11 Wireless Networks: The Definitive Guide, 2nd Edition”, O’Reilly Media, Inc. - <https://www.oreilly.com/library/view/80211-wireless-networks/0596100523/>
- [4] IEEE 802.11-05/0710r 0, “WDS” Clarifications - http://www.ieee802.org/1/files/public/802_architecture_group/802-11/4-address-format.doc
- [5] Espressif Systems - <https://www.espressif.com/>
- [6] NodeMCU - https://www.nodemcu.com/index_en.html
- [7] DS3231S(N) Data Sheet - <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>
- [8] DS3231M Data Sheet - <https://datasheets.maximintegrated.com/en/ds/DS3231M.pdf>
- [9] ESP8266 NON-OS SDK - https://github.com/espressif/ESP8266_NONOS_SDK
- [10] ESP8266 core for Arduino - <https://github.com/esp8266/Arduino>
- [11] ESP8266 Technical Reference - https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
- [12] ESP8266 Non-OS SDK API Reference - https://www.espressif.com/sites/default/files/documentation/2c-esp8266_non_os_sdk_api_reference_en.pdf
- [13] ESP8266_NONOS_SDK_v1.3.0_15_08_08 Release Note - <https://bbs.espressif.com/viewtopic.php?p=3092#p3092>
- [14] ESP-OPEN-SDK - <https://github.com/pfalcon/esp-open-sdk>
- [15] ESP8266 Pinout Reference - <https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>
- [16] SdFat, Greiman - <https://github.com/greiman/SdFat>
- [17] Modifikovana verzija SdFat 1.4.1 biblioteke - <https://github.com/FmasterofU/SdFat>
- [18] ESPNtpClient, Germán Martín - <https://github.com/gmag11/ESPNtpClient>
- [19] TimeLib, Paul Stoffregen - <https://github.com/PaulStoffregen/Time>
- [20] MAC address randomization, draft-zuniga-mac-address-randomization-01, IETF Internet Draft - <https://datatracker.ietf.org/doc/html/draft-zuniga-mac-address-randomization-01>
- [21] Indoor Occupancy Tracking in Smart Buildings Using Passive Sniffing of Probe Requests, Edwin Vattapparamban et al. - https://www.researchgate.net/publication/305674790_Indoor_occupancy_tracking_in_smart_buildings_using_passive_sniffing_of_probe_requests
- [22] Defeating MAC Address Randomization Through Timing Attacks, Célestin Matte et al. - <https://hal.inria.fr/hal-01330476/document>
- [23] DS3231, NorthernWidget - <https://github.com/NorthernWidget/DS3231>

Biografija

Igor Šikuljak je rođen u 22.07.1998. godine u Sokocu, Bosna i Hercegovina. Osnovnu školu „Vuk Karadžić” u Vlasenici je završio 2013. godine kao učenik generacije. Potom upisuje gimaziju u SŠC „Milorad Vlačić” u Vlasenici koju završava 2017. godine, takođe kao učenik generacije. Uporedo sa srednjom školom, polaznik je na seminarima računarstva u IS Petnica u Valjevu. Nakon završene srednje škole, upisuje Fakultet tehničkih nauka u Novom Sadu, smjer Računarstvo i automatika, a postaje i asistent na programu računarstva u IS Petnica, gdje je i dalje angažovan.

Položio je sve ispite predviđene planom i programom svog smijera, na modulu Primijenjene računarske nauke i informatika i studijskoj grupi Internet i elektronsko poslovanje.

Kontakt: igorsikuljak@gmail.com



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	монографска публикација	
Тип записа, ТЗ:	текстуални штампани документ	
Врста рада, ВР:	дипломски-бечелор рад	
Аутор, АУ:	Игор Шидуљак	
Ментор, МН:	проф. др Милан Видаковић	
Наслов рада, НР:	Систем за праћење кретања мобилних уређаја са омогућеном употребом Вај-фај технологије	
Језик публикације, ЈП:	српски	
Језик извода, ЈИ:	српски / енглески	
Земља публиковања, ЗП:	Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2021	
Издавач, ИЗ:	ауторски репринт	
Место и адреса, МА:	Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6	
Физички опис рада, ФО:	бр. поглавља: 8 / страница: 49 / цитата: 23 / слика: 7 / листинга: 20 / прилога: 0 / табела: 2 / графикона: 3	
Научна област, НО:	Рачунарске науке	
Научна дисциплина, НД:	Рачунарске мреже, безбједност система, имбедед системи	
Предметна одредница / Кључне речи, ПО:	Праћење мобилних уређаја, MAC адресе, 802.11 Probe Request, 802.11 RTS/CTS, Espressif ESP8266	
УДК:		
Чува се, ЧУ:	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	У овом раду је представљено рјешење за просторно праћење кретања мобилних уређаја. Систем подразумијева скуп ESP8266 уређаја распоређених по простору у коме желимо да пратимо мобилне уређаје. ESP8266 чворови слушају мрежни саобраћај из своје околине и детектују Probe Request пакете који откривају присуство појединачних уређаја. Рјешење показује одличне резултате за мобилне уређаје који не користе MAC рандомизацију, као и за уређаје који су повезани на одређену Вај-Фај мрежу јер се у тим околностима MAC рандомизације налази у статичној фази.	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:		
Председник:	др Мирослав Зарић, ванредни проф., ФТН Нови Сад	
Члан:	др Жељко Вуковић, ФТН Нови Сад	Потпис ментора:
Члан,ментор:	др Милан Видаковић, ред. проф., ФТН Нови Сад	



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO :		
Identification number, INO :		
Document type, DT :	monographic publication	
Type of record, TR :	textual material	
Contents code, CC :	BSc thesis	
Author, AU :	Igor Šikuljak	
Mentor, MN :	Milan Vidaković, PhD, prof., FTN Novi Sad	
Title, TI :	A system for spatial movement tracking of mobile devices with enabled WiFi technology	
Language of text, LT :	Serbian	
Language of abstract, LA :	serbian / english	
Country of publication, CP :	Serbia	
Locality of publication, LP :	Vojvodina	
Publication year, PY :	2021	
Publisher, PB :	author's reprint	
Publication place, PP :	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6	
Physical description, PD :	no. of chapters: 8 / pages: 49 / citations: 23 / images: 7 / listings: 20 / appendices: 0 / tables: 2 / graphs: 3	
Scientific field, SF :	Computer science	
Scientific discipline, SD :	Computer networks, security, embedded systems	
Subject / Keywords, S/KW :	Mobile device tracking, MAC addresses, 802.11 Probe Request packets, 802.11 RTS/CTS packets, Espressif ESP8266	
UDC :		
Holding data, HD :	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad	
Note, N :		
Abstract, AB :	<p>In this paper a solution for spatial movement tracking of mobile devices is presented. The system consists of several ESP8266 chips that are spread out in a space in which we want to track mobile device movement. ESP8266 nodes are listening to network traffic in their vicinity and they filter out Probe Request packets that reveal the nearby presence of a specific device. The solution shows excellent results for mobile devices that do not use MAC Randomization, but also for devices that are connected to a WiFi network, regardless of MAC Randomization, because in those circumstances, MAC Randomization is in its static phase.</p>	
Accepted by sci. board on, ASB :		
Defended on, DE :		
Defense board, DB :		
President:	Miroslav Zarić, PhD, assoc. prof., FTN Novi Sad	
Member:	Željko Vuković, PhD, FTN Novi Sad	Mentor's signature:
Mentor:	Milan Vidaković, PhD, prof., FTN Novi Sad	