Android aplikacija za praćenje parametara rada SoC uređaja

Pristavnik, Ivana

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:141645

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-07**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ANDROID APLIKACIJA ZA PRAĆENJE PARAMETARA RADA SOC UREĐAJA

Završni rad

Ivana Pristavnik

Osijek, 2017. godina.

SADRŽAJ

1. UVOD 3
1.1. Zadatak završnog rada 3
2. POSLUŽITELJ 4
2.1. Single Board Computer (SBC) i System on Chip (SoC) 4
2.1.1. Raspberry Pi 5
2.2. Postavljanje poslužitelja 5
2.2.1. Bežična pristupna točka 6
2.2.2. Statička IP adresa 8
2.2.3. LAMP poslužitelj 8
2.3. Praćeni parametri rada SoC uređaja 9
2.3.1. Očitavanje parametara rada SoC uređaja pomoću PHP-a 9
3. KLIJENT 12
3.1. Izrada korisničkog sučelja 12
3.2. Funkcionalnost aplikacije 14
3.2.1. Povezivanje s Internetom 14
4. TESTIRANJE APLIKACIJE 18
4.1. Usporedba s postojećim rješenjima 23
4.1.1. *PhpSysInfo* 23
4.1.2. *RPi-Monitor* 23
4.1.3. *Raspcontrol* 23
4.1.4. *Raspberry Stats* 24
5. ZAKLJUČAK 25
LITERATURA 26
SAŽETAK 27
ABSTRACT 28
ŽIVOTOPIS 29

1. UVOD

Pojavom SoC uređaja otvaraju se mogućnosti za eksponencijalan razvoj prenosivih uređaja malih dimenzija i niske potrošnje električne energije.

U ovom radu bit će prikazano kako pomoću Android aplikacije pratiti parametre rada SoC uređaja. Korišteni SoC uređaj je Raspberry Pi, malo računalo razvijeno u edukacijske svrhe. Osim što se Raspberry Pi može koristiti kao osobno računalo priključivanjem monitora, miša i tipkovnice, našao je primjenu čak i u robotici i elektronici. Pogodan je za mrežnu pohranu podataka te se može koristiti kao poslužitelj. Cilj ovog rada je pratiti parametre rada danog uređaja kao što su opterećenje procesora, radne memorije, pohrane i prijenosa podataka putem mreže. Budući da je jedna od najvećih kvaliteta bilo kojeg SoC uređaja njegova veličina, korisniku nije u interesu imati uređaj spojen na monitor samo radi nadziranja njegovih performansi. Racionalnije rješenje je korištenje aplikacije na pametnom telefonu koji se bežično povezuje s uređajem kojega se nadzire.

U sljedećem, poglavlju 2, opisana je poslužiteljska strana okruženja izrađenog u okviru rada. Pojašnjava se korištenje PHP programskog jezika u skripti pokrenutoj na poslužitelju koja prikuplja podatke i formira ih u JSON oblik. Za povezivanje klijentske i poslužiteljske strane je korišteno bežično povezivanje putem WiFi-ja te je na poslužitelju postavljena bežična pristupna točka da bi se omogućilo korištenje aplikacije izvan pristupa Internetu. U poglavlju 3 će biti opisana klijentska strana koju čini uređaj na kojemu je pokrenuta Android aplikacija, izrađena pomoću razvojnog okruženja *Android Studio*. Aplikacija pomoću Java programskog jezika šalje zahtjev poslužitelju za dohvaćanje vrijednosti traženih parametara rada i te podatke prikazuje na grafičkom korisničkom sučelju izrađenom pomoću XML opisnog jezika. Konačno se, u poglavlju 4, pojašnjava način rada aplikacije, od njenog pokretanja, sve do njenog uništavanja te se ista uspoređuje s već postojećim rješenjima.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je izraditi Android aplikaciju koja na zaslonu mobilnog uređaja prikazuje parametre rada nekog od dostupnih SoC uređaja, poput Raspberry Pi, ODROID U3+ i CubieBoard2. Potrebno je omogućiti povezivost Android uređaja s navedenim SoC uređajem putem WiFi ili Bluetooth veze. Također je potrebno instalirati i podesiti poslužiteljsku stranu na SoC-u, instalirati sve potrebne alate te dokumentirati svaki korak.

2. POSLUŽITELJ

Model klijent-poslužitelj se može primijeniti na svakom sustavu koji se sastoji od dijela koji traži obavljanje nekog zadatka i dijela koji obavlja traženi zadatak. Klijentska strana mora osigurati korisničko sučelje, pomoću kojega korisnik može postaviti upit, slati upit poslužitelju i primiti poslužiteljev odgovor, koji se prikazuje korisniku. Poslužitelj prihvaća klijentov zahtjev, obrađuje ga i šalje rezultat klijentu.

Poslužitelj za potrebe ovog rada je SoC (eng. *System on Chip*) uređaj čiji se parametri očitavaju te se te vrijednosti šalju klijentu, koji će biti opisan u sljedećem poglavlju.

2.1. Single Board Computer (SBC) i System on Chip (SoC)

SBC (eng. *Single Board Computer*) je potpuno funkcionali računalni sustav izveden pomoću jedne pločice. SBC ima mikroprocesore, memoriju, ulaz/izlaz i ostale mogućnosti koje pružaju minimalističku funkcionalnost računala. Za razliku od osobnog računala, većina SBC-a nemaju mogućnost proširenja. Zbog napretka u elektronici i razvoju poluvodičkih komponenata, cijene su im vrlo niske, a efektivnost, kao i područje primjene, veliko.

Svi SBC su pretežno SoC uređaji. SoC je integrirani krug koji posjeduje sve komponente jednog računala na jednom čipu. SoC se često koristi u mobilnim uređajima zbog široke primjene i niske potrošnje energije.

Prema [1], najveća prednost korištenja SoC-a je njegova veličina. Ukoliko se koristi CPU, teško je napraviti kompaktno računalo zbog velikog broja individualnih čipova i ostalih komponenti koje se moraju postaviti na pločicu. Ukoliko se koristi SoC, tada se ima dovoljno mjesta za postavljanje baterije, antene i ostalih dodataka.

Popularne serije SBC uređaja su:

- Raspberry Pi (Raspberry Pi Foundation),
- Banana Pi i Banana Pro,
- Intel Edison i Galileo,
- Cubieboard,
- Beaglebone i Beagleboard.

2.1.1. Raspberry Pi

Raspberry Pi je serija SBC uređaja, veličine kreditne kartice. Raspberry Pi je razvila tvrtka Raspberry Pi Foundation s ciljem promicanja učenja osnova računalnih znanosti u školama pružanjem jeftine računalne platforme. Raspberry Pi je pušten u prodaju 2012. godine te je doživio veliki uspjeh. Najnoviji model je Raspberry Pi 3 Model B. Njegove specifikacije su dane u tablici 2.1.

Tab. 2.1. Specifikacije uređaja Raspberry Pi 3 Model B[2]

Datum izdavanja	Veljača 2016.
Arhitektura	ARMv8
SoC Broadcom	BCM2837
CPU	1.2 GHz 64-bit quad-core ARM Cortex-A53
GPU	Broadcom VideoCore IV (3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz
Memorija	1 GB (dijeljeno s GPU)
USB	4 2.0 ulaza
Video izlaz	HDMI rev 1.3 i Composite Video RCA jack
Pohrana	Micro SDHC slot
Mreža	10/100 Mbps Ethernet, Bluetooth, WiFi
Izvor napajanja	5V MicroUSB
Ocjene snage	800 mA (4W)

2.2. Postavljanje poslužitelja

Da bi se mogao koristiti Raspberry Pi, potrebno je na microSD karticu, koja se koristi kao vanjska memorija, staviti operacijski sustav. U ovom radu je korišten službeni NOOBS (eng. *New Out Of the Box Software*). Pomoću NOOBS-a se jednostavno instalira Raspbian na Raspberry Pi.

Raspbian je operacijski sustav temeljen na Debian operacijskom sustavu te je primarni operacijski sustav za Raspberry Pi. On koristi PIXEL (eng. *Pi Improved Xwindows Environment, Lightweight*) koji pruža okruženje radne površine. Za potrebe rada je korištena inačica Raspbian Jessie + PIXEL, izdana u travnju 2017. godine.

Za bežično povezivanje Android uređaja i Raspberry Pi potrebno je koristiti WiFi ili Bluetooth. Usporedba WiFi i Bluetooth tehnologija je dana u tablici 2.2.

Tab. 2.2. Usporedba WiFi i Bluetooth bežičnog povezivanja uređaja[3]

	WiFi	**Bluetooth**
Frekvencija	2,4/3,6/5 GHz	2,4 GHz
Propusnost	Visoka (11 Mbps).	Niska (800 Kbps).
Maksimalna udaljenost uređaja	802.11b/g 32m do 95m. 802.11n ima veći domet. 2.5GHz WiFi ima veći domet od 5GHz WiFi. Antene mogu dodatno povećati domet.	5 do 30 metara.
Potrošnja energije	Visoka.	Niska.
Jednostavnost korištenja	Kompleksnija, traži dodatnu konfiguraciju hardvera i softvera.	Jednostavno za korištenje, moguće je koristiti za spajanje do 7 uređaja istovremeno.
Potrebna konfiguracija	Bežični adapter na svim uređajima u mreži, bežični usmjerivač i/ili bežična pristupna točka.	Bluetooth adapter na svim uređajima koji se spajaju.

Budući da je potrebno često razmijenjivati podatke te u što kraćem vremenu dobiti informaciju o parametrima rada Raspberry Pi uređaja, brzina prijenosa je bitna te je korišten WiFi za bežično povezivanje i prijenos podataka. Ukoliko bi se koristilo povezivanje preko usmjerivača na kojega se putem WiFi-ja spajaju i Android uređaj i Raspberry Pi, aplikacija ne bi radila izvan dometa tog usmjerivača, tj. bez pristupa internetu.

2.2.1. Bežična pristupna točka

Prema [4], Raspberry Pi se može koristiti kao bežična pristupna točka koja pokreće samostalnu mrežu te tada nije potrebno imati pristup internetu. To je moguće postaviti pomoću ugrađenog WiFi-ja u Raspberry Pi 3. Da bi Raspberry Pi radio kao bežična pristupna točka, potrebno je instalirati sustav za bežično pristupanje te DHCP poslužitelj da bi se uređajima koji se spajaju na mrežu dala mrežna adresa.

Prvi korak prema postavljanju bežične pristupne točke je instalacija potrebnih paketa, prikazana na slici 2.1.

```
sudo apt-get install dnsmasq hostapd
```

Sl. 2.1. Instalacija paketa dnsmasq i hostapd

Prema [5], **Dnsmasq** je DNS (eng. *Domain Name System*) usmjerivač i DHCP (eng. *Dynamic Host Configuration Protocol*) poslužitelj za male računalne mreže. Podržava statičko i dinamičko IP adresiranje. DHCP u kombinaciji s DNS-om omogućava lokalnim uređajima, s DHCP dodijeljenim adresama, da se prikazuju u DNS-u.

Hostapd je paket koji omogućava upotrebu pristupne točke pomoću ugrađenog WiFi-ja te se koristi za autentifikaciju poslužitelja.

Nakon instalacije je DHCP ugašen. Da bi se uključio, potrebno mu je dati raspon IP adresa koje može davati.

```
sudo nano /etc/dnsmasq.conf
```

Sl. 2.2. Otvaranje datoteke /etc/dnmasq.conf

Korištenjem naredbe na slici 2.2. se otvara datoteka te ju se tada može mijenjati. Na kraju datoteke se definira raspon IP adresa, počinjući s 10.0.0.2 i završavajući s 10.0.0.5 te se postavlja da te četiri IP adrese budu dostupne 12 sati.

Potrebno je stvoriti novu datoteku za konfiguraciju hostapd paketa te nakon otvaranja datoteke /etc/hostapd/hostapd.conf u nju upisati sljedeće:

```
interface=wlan0
hw_mode=g
channel=10
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
rsn_pairwise=CCMP
wpa_passphrase=password
ssid=raspberry
```

Sl. 2.3. Datoteka /etc/hostapd/hostapd.conf

Na slici 2.3. se pomoću uputa iz [6] postavlja ime WiFi sučelja, definira korištenje 2,4 GHz raspona frekvencija te korištenje kanala 10. Nadalje, definira se korištenje WPA (eng. *WiFi Protected Access algoritma*), algoritma za sigurnu komunikaciju putem IEEE 802.11 bežičnih mreža. WPA-Personal ili WPA-PSK (eng. *Wi-Fi Protected Access - pre-shared key*) je

dizajniran za male mreže. Svaki mrežni uređaj pomoću tog algoritma šifrira mrežni promet koristeći 256 bitni ključ. Definira se korištenje CCMP (eng. *Counter Mode Cipher Block Chaining Message Authentication Code Protocol, Counter Mode CBC-MAC Protocol*) protokola za šifriranje podataka koji vrši ovjeru. Također, postavljaju se lozinka i ime bežične pristupne točke.

2.2.2. Statička IP adresa

Razlika između statičkih i dinamičkih IP adresa je u tome što se statičke adrese ne mijenjaju, a dinamičke se mijenjaju. Prednost statičkih IP adresa je što su povoljne za kreiranje računalnih poslužitelja. Da bi se postavila statička IP adresa poslužitelja, potrebno je u datoteci /etc/network/interfaces dodati sljedeće:

```
allow-hotplug wlan0
iface wlan0 inet static
address 10.0.0.1
netmask 255.255.255.0
network 10.0.0.0
broadcast 10.0.0.255
```

Sl. 2.4. Dio datoteke /etc/network/interfaces

Statička IP adresa poslužitelja je, nakon promjena na slici 2.4., 10.0.0.1, a to je jedna od adresa iz raspona privatnih IP adresa. Privatne IP adrese nisu direktno povezane s internetom te zbog toga te adrese ne moraju biti jedinstvene. Danas se rasponi privatnih adresa koriste u zaštićenim mrežama. Na kraju, da bi se primijenile sve promjene, potrebno je ponovno pokrenuti Raspberry Pi.

2.2.3. LAMP poslužitelj

Prema [7], LAMP je popularna *open source* internetska platforma, često korištena za internetske stranice i poslužitelje. Uključuje Linux operacijski sustav i Apache HTTP poslužitelj. Podaci se spremaju u MySQL bazu podataka, a dinamični sadržaj se obrađuje pomoću PHP-a.

Apache HTTP poslužitelj je jedan od najpopularnijih internetskih poslužitelja. Dobro je dokumentiran što ga čini idealnim za posluživanje internetske stranice. Apache, i njegove dodatne alate, je moguće instalirati pomoću naredbe sa slike 2.5.

```
sudo apt-get install apache2 apache2-doc apache2-utils
```

Sl. 2.5. Preuzimanje i instalacija Apache poslužitelja

PHP je komponenta koja obrađuje programski kod i prikazuje dinamički sadržaj. Može pokretati skripte, spajati se na MySQL bazu podataka, uzimati podatke iz baze, obraditi ih i proslijediti internetskom poslužitelju koji ih prikazuje. Ako poslužitelj podržava PHP, on će automatski analizirati .php datoteke koje su postavljene na njega. Moguće ga je instalirati pomoću naredbe sa slike 2.6.

```
apt-get install libapache2-mod-php5 php5 php-pear php5-xcache
```

Sl. 2.6. Preuzimanje i instalacija PHP biblioteka

Pomoću naredbe sa slike 2.7. je moguće instalirati dodatne pakete za povezivanje s bazom podataka.

```
apt-get install php5-mysql
```

Sl. 2.7. Preuzimanje i instalacija dodatnih PHP paketa

MySQL se koristi za upravljanje bazom podataka te ga je moguće instalirati pomoću naredbe sa slike 2.8.

```
apt-get install mysql-server mysql-client
```

Sl. 2.8. Preuzimanje i instalacija MySQL baze podataka

2.3. Praćeni parametri rada SoC uređaja

Parametri rada SoC uređaja čija će se vrijednost prikazivati pomoću Android aplikacije izrađene u ovom radu su opterećenje procesora, popunjenost radne memorije, popunjenost tvrdog diska (memorijske kartice) te količina primljenih i poslanih podataka putem mreže.

2.3.1. Očitavanje parametara rada SoC uređaja pomoću PHP-a

Prema [8], PHP je programski jezik opće primjene koji je stvoren radi lakog načina obrade podataka pri stvaranju internetskih stranica. Može biti pokrenut u naredbenoj liniji na bilo kojem računalu s instaliranim PHP-om. Obično se koristi na poslužiteljskoj strani te podržava komunikaciju s bazom podataka.

U tekstualnu datoteku se upisuju naredbe koje se trebaju izvršiti pomoću PHP koda te se ta datoteka sprema unutar mape /var/html/www. Potrebno je datoteci dati ime s proširenjem .php kako bi poslužitelj prepoznao programski jezik pomoću kojega je datoteka pisana.

Prema [9], naredbe za preuzimanje vrijednosti svakog od traženih parametara se mogu dobiti korištenjem funkcije *shell_exec()*. Njome se izvodi naredba u ljusci te se dobiva povratni podatak u obliku stringa. Dobivene podatke je potrebno pretvoriti u odgovarajući oblik da bi se dobio valjani prikaz koji je jednostavno dalje obrađivati.

```
$prevVal = shell_exec("cat /proc/stat");
$prevArr = explode(' ',trim($prevVal));
$prevTotal = $prevArr[2] + $prevArr[3] + $prevArr[4] + $prevArr[5];
$prevIdle = $prevArr[5];
usleep(0.15 * 1000000);
$val = shell_exec("cat /proc/stat");
$arr = explode(' ', trim($val));
$total = $arr[2] + $arr[3] + $arr[4] + $arr[5];
$idle = $arr[5];
$intervalTotal = intval($total - $prevTotal);
$stat['cpu'] =  intval(100 * (($intervalTotal - ($idle - $prevIdle)) /
$intervalTotal));
```

Sl. 2.9. Dohvaćanje podataka o radu procesora

Slikom 2.9. je prikazan način korištenja *shell_exec()* funkcije. Tim dijelom koda se dohvaćaju vrijednosti koje vraća *shell_exec()* funkcija te se one pohranjuju u varijablu `$prevVal`. Čeka se jedan vremenski period, koji iznosi 0,15 sekundi, te se tada ponovno očitavaju vrijednosti i uspoređuju se s prošlima. Konačno se izračunava postotak nekorištenja procesora u ukupnom vremenu. Dani podaci se obrađuju da bi se prikazali u odgovarajućem obliku kao cjelobrojni tip.

Na sličan način se dohvaćaju i ostale vrijednosti potrebnih parametara koji će se nadalje prikazivati u aplikaciji.

JSON (eng. *JavaScript Object Notation*) je sintaksa za pohranu i razmjenu podataka. JSON može biti realiziran pomoću dvije strukture:

1. Neporedana lista vrijednosti, za što se koriste objekti.
2. Poredana lista vrijednosti, za što se koristi niz.

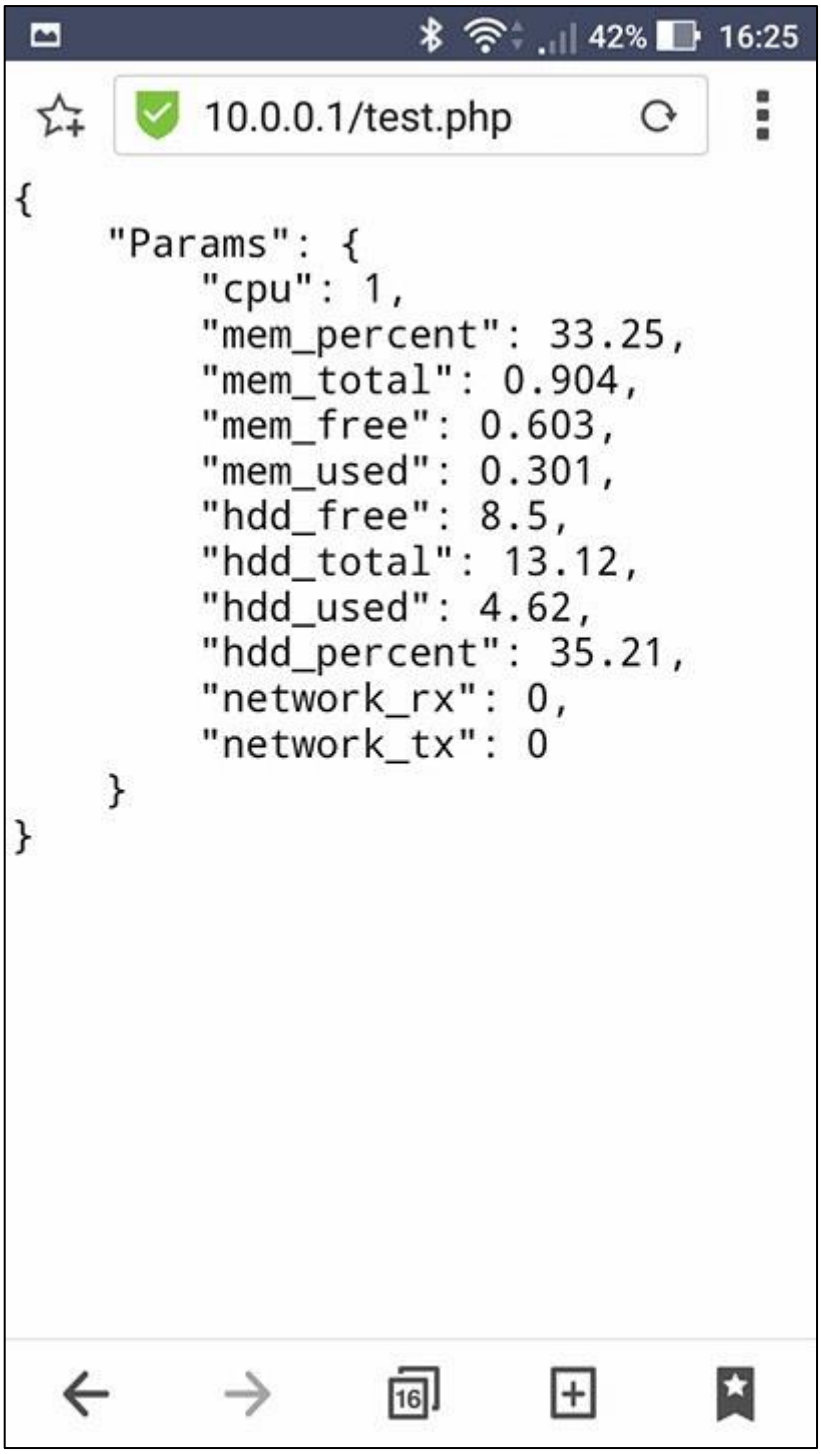
Da bi se vrijednosti pohranjene u objektu ili nizu u PHP kodu prikazale u JSON obliku, potrebno je u .php datoteku dodati sljedeće:

```
header('Content-type: text/json');
header('Content-type: application/json');
```

Sl. 2.10. Određivanje prikaza JSON oblika

Za pretvorbu PHP elemenata u JSON elemente se koristi funkcija *json_encode(),* a za ispis se koristi PHP funkcija *echo()*.[10]

Kada se otvori adresa internetskog poslužitelja s putanjom do PHP skripte, prikažu se parametri rada u obliku prikazanom na slici 2.1.



Sl. 2.11. Parametri rada SoC uređaja prikazani na internetskom poslužitelju

Params je ime JSON objekta koji sadrži ključeve, koji predstavljaju nazive očitavanih parametara rada, i vrijednosti tih ključeva. Vrijednost ključa *cpu* se prikazuje u postotcima, kao i vrijednosti ključeva *mem_percent* i *hdd_percent.* Vrijednosti ključeva *network_rx* i *network_tx* su prikazane u megabajtima (MB), a sve ostale u gigabajtima (GB).

3. KLIJENT

Uređaj na kojemu je pokrenuta Android aplikacija je klijent. Aplikacija od poslužitelja traži podatke te ih prikazuje na sučelju.

Android Studio je službeno okruženje za razvoj i testiranje Android aplikacija. Prema [11], izrada Android aplikacija se provodi kroz pet koraka:

1. Postavljanje softvera i stvaranje novog projekta.
2. Pisanje aplikacije i dizajniranje korisničkog sučelja.
3. Gradnja i pokretanje aplikacije. U ovom koraku se stvara APK (eng. *Android Package Kit*)[1] paket koji se može instalirati u emulatoru ili na Android uređaju.
4. Uklanjanje pogrešaka i testiranje aplikacije s ciljem optimiziranja performansi.
5. Objavljivanje aplikacije čime se daje mogućnost korištenja aplikacije drugim korisnicima.

3.1. Izrada korisničkog sučelja

Korisničko sučelje je sve što korisnik može vidjeti i s čime komunicira. Android pruža širok spektar već izrađenih komponenata koje omogućavaju izradu grafičkog korisničkog sučelja za aplikaciju. Za izradu korisničkog sučelja se koristi XML (eng. *EXtensible Markup Language*), opisni jezik za označavanje podataka te se izgled aplikacije definira pomoću activity_main.xml datoteke.

Radi jednostavnosti i preglednosti aplikacije, za izradu korisničkog sučelja korišten je *TableLayout*. To je *ViewGroup*, poseban prikaz, koji sadrži druge elemente te njih prikazuje u stupcima i retcima. Obrubi *TableLayout* ćelija, redaka i stupaca se ne prikazuju. Također, ćelije mogu ostati prazne.

TableRow je jedan redak unutar *TableLayout* elementa. Svaki redak može imati nula ili više ćelija, a svaka ćelija je definirana nekim drugim *View* elementom. To znači da je unutar jedne ćelije moguće postaviti i *ImageView*, *TextView*, čak i *Button* elemente.

U ovom radu je korištena tablica sa 16 redaka, od čega su četiri nazivi većih sličnih skupina podataka koje se prikazuju, 11 redaka podijeljenih u dva stupca za prikaz vrijednosti određenog parametra te jedan redak s gumbom za dohvaćanje podataka.

[1] Paket kojega koristi Android operacijski sustav za distribuciju i instalaciju mobilnih aplikacija.

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_margin="0.5dip"
        android:background="#42adf4"
        android:gravity="start"
        android:padding="5dip"
        android:text="@string/usage"
        android:textColor="#000000"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/cpu"
        android:gravity="end"
        android:padding="5dip"
        android:textColor="#000000" />
</TableRow>
```

Sl. 3.1. Primjer prikaza jednog retka u tablici

Pomoću XML koda na slici 3.1. se prikazuje jedan redak tablice, koji se sastoji od dva *TextView* elementa. Prvi *TextView* prikazuje natpis „OPTEREĆENJE“, a u drugome će se prikazivati određena vrijednost. Na sličan način su izrađeni ostali retci tablice. Slika 3.2. prikazuje konačan izgled korisničkog sučelja.

Sl. 3.2. Izgled korisničkog sučelja

3.2. Funkcionalnost aplikacije

Android aplikacije su pisane u Java programskom jeziku. Prema [12], Java je objektno orijentirani programski jezik, što znači da koristi koncept uporabe programskih objekata. Objektno orijentiranim pristupom se programerima skraćuje vrijeme pisanja aplikacije. Također, Java omogućava izvršavanje više procesa istovremeno korištenjem višenitnog pristupa.

Unutar `onCreate()` metode se postavlja *layout* definiran unutar activity_main.xml datoteke i postavlja se gumb koji je povezan s tom XML datotekom. Definira se stvaranje novog objekta klase *ConnectedToServer* i izvršavanje njenih metoda pritiskanjem gumba.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    refresh = (Button) findViewById(R.id.refresh);
    refresh.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new ConnectedToServer().execute();
        }
    });
}
```

Sl. 3.3. `OnCreate()` metoda

3.2.1. Povezivanje s Internetom

Da bi se aplikacija mogla povezati s Internetom, potrebno je postaviti dopuštenje unutar AndroidManifest.xml datoteke kao što je prikazano na slici 3.4.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Sl. 3.4. Dopuštenje povezivanja s Internetom

Budući da se nije moguće povezivati s Internetom u glavnoj niti programa, potrebno je koristiti ili Java višenitni pristup ili *AsyncTask* Android klasu da bi se omogućilo preuzimanje podataka s poslužitelja. U ovom radu je korištena klasa *AsyncTask* zbog toga što je, prema [13], optimalna za postupke u mreži koji ne zahtijevaju preuzimanje velike količine podataka.

AsyncTask je klasa koja omogućuje pokretanje pozadinskih funkcija i prikazivanje rezultata na korisničkom sučelju. Ona se provodi u četiri koraka:

1. `onPreExecute()`, prikazuje se na korisničkom sučelju prije nego je zadatak izveden. Taj korak se najčešće koristi za postavljanje zadatka, na primjer pokazivanje trake napretka.

2. `doInBackground(Params…)`, izvodi se u pozadini odmah nakon `onPreExecute()`. Koristi se za izvođenje pozadinskih operacija koje zahtijevaju više vremena.
3. `onProgressUpdate(Progress…)`, koristi se za prikazivanje napretka na korisničkom sučelju dok se pozadinske operacije izvršavaju.
4. `onPostExecute(Result)`, izvršava se nakon završavanja pozadinskih operacija.

Klasa *ConnectedToServer* nasljeđuje *AsyncTask* klasu. U metodi `doInBackground()` *ConnectedToServer* klase se provjerava može li aplikacija pristupiti URL-u poslužitelja. Ukoliko je aplikacija uspjela pristupiti poslužitelju, metoda prosljeđuje „1“, a ukoliko nije, tada prosljeđuje „0“.

```
@Override
protected Integer doInBackground(Void... voids) {
    try {
        URL myUrl = new URL("http://10.0.0.1/");
        URLConnection connection = myUrl.openConnection();
        connection.setConnectTimeout(500);
        connection.connect();
        return 1;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return 0;
}
```

Sl. 3.5. `doInBackground()` metoda *ConnectedToServer* klase

U `onPostExecute()` metodi *ConnectedToServer* klase se poziva metoda `getData()`, odnosno, poziva se metoda `alert()` ako se aplikacija ne može povezati s poslužiteljem.

```
protected void onPostExecute(Integer check) {
    if (check == 1) {
        getData();
    } else {
        alert();
    }
}
```

Sl. 3.6. `onPostExecute()` metoda *ConnectedToServer* klase

Metoda `getData()`, koja je prikazana na slici 3.7., stvara novi objekt klase *JSONParse*, pokreće metode te klase i, na isti način kao što je opisano u `onCreate()` metodi, stvara novi objekt *ConnectedToServer* klase da bi provjerila povezanost s poslužiteljem.

```
private void getData() {
    try {
        new JSONParse().execute();
        new ConnectedToServer().execute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Sl. 3.7. `getData()` metoda

Metoda `alert()` stvara novi *alertDialog* koji prikazuje poruku da aplikacija nije povezana s poslužiteljem i gumb koji zatvara *alertDialog* i poziva metodu `clearTextView()` koja briše sve vrijednosti promatranih parametara sa zaslona.

Klasa *JSONParse* također nasljeđuje *AsyncTask* klasu. Unutar `onPreExecute()` metode se povezuje ime *TextView* elementa s elementom u XML datoteci, kao što je to na slici 3.7. prikazano za primjer povezivanja imena TextView elementa „cpu1“ i odgovarajućeg elementa XML datoteke.

```
@Override
protected void onPreExecute() {
    super.onPreExecute();

    cpu1 = (TextView)findViewById(R.id.cpu);
}
```

Sl. 3.7. `onPreExecute()` metoda *JSONParse* klase

Unutar `doInBackground()` metode, prikazane na slici 3.8., *JSONParse* klase se definiraju operacije koje se izvode u pozadini. Prvo se stvara novi objekt klase *JSONParser*. To je klasa koja pomoću HTTP protokola i URL-a (eng. *Uniform Resource Locator*) dohvaća podatke ispisane na poslužitelju pomoću PHP skripte. Ona parsira string i stvara JSON objekt, koji se prosljeđuje u sljedeću metodu.

```
@Override
protected JSONObject doInBackground(String... args) {
    JSONParser jParser = new JSONParser();

    JSONObject json = jParser.getJSONFromUrl(url);
    return json;
}
```

Sl. 3.8. `doInBackground()` metoda *JSONParse* klase

Unutar `onPostExecute()` metode *JSONParse* klase se za svaki od ključeva objekta dohvaća njegova vrijednost. Ta vrijednost se tada postavlja u *TextView* element XML datoteke na način prikazan na slici 3.9.

```
@Override
protected void onPostExecute(JSONObject json) {
        String cpu = json.getJSONObject("Params").getString("cpu");
        cpu1.setText(cpu);
}
```

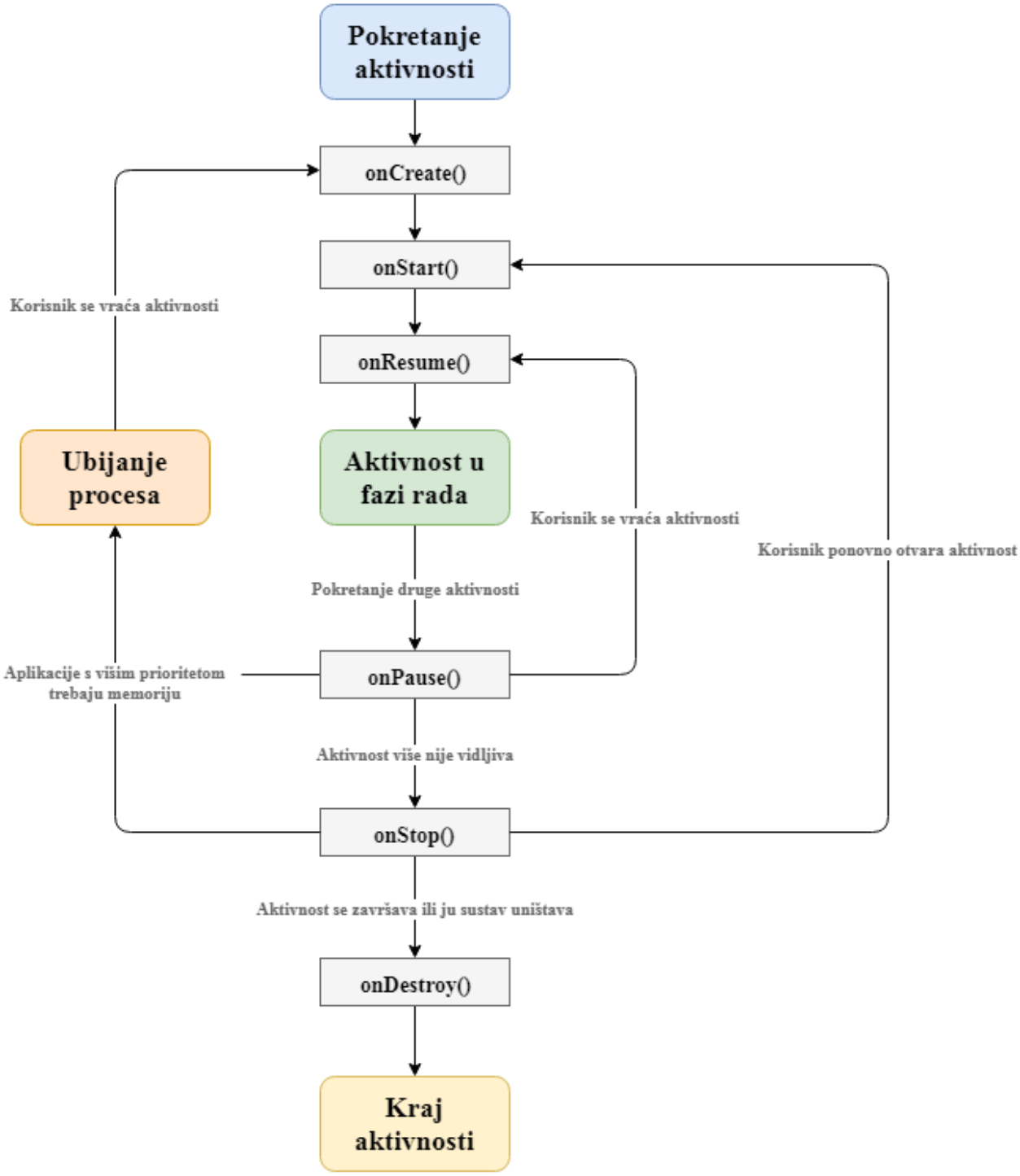
Sl. 3.9. `onPostExecute()` metoda *JSONParse* klase

4. TESTIRANJE APLIKACIJE

Prema [14], Android sustav ima više različitih faza u kojim svaka od aplikacija može biti. Ovisno o fazi, sustav određuje na koji način korisnik vidi aplikaciju i vidi li ju uopće. Android posjeduje te faze radi raspodjele resursa za trenutno korištene aplikacije. Za navigaciju između različitih faza životnog ciklusa aplikacije, klasa *Activity* pruža šest funkcijskih poziva, prikazanih na slici 4.1.

Faze životnog ciklusa su sljedeće:

- Stvaranje (*engl. Being created*),
- Pokretanje (*engl. Starting*),
- Nastavljanje (*engl. Resuming*),
- Rad (*engl. Running*),
- Pauziranje (*engl. Pausing*),
- Zaustavljanje (*engl. Stopping*),
- Uništavanje (*engl. Being destroyed*).

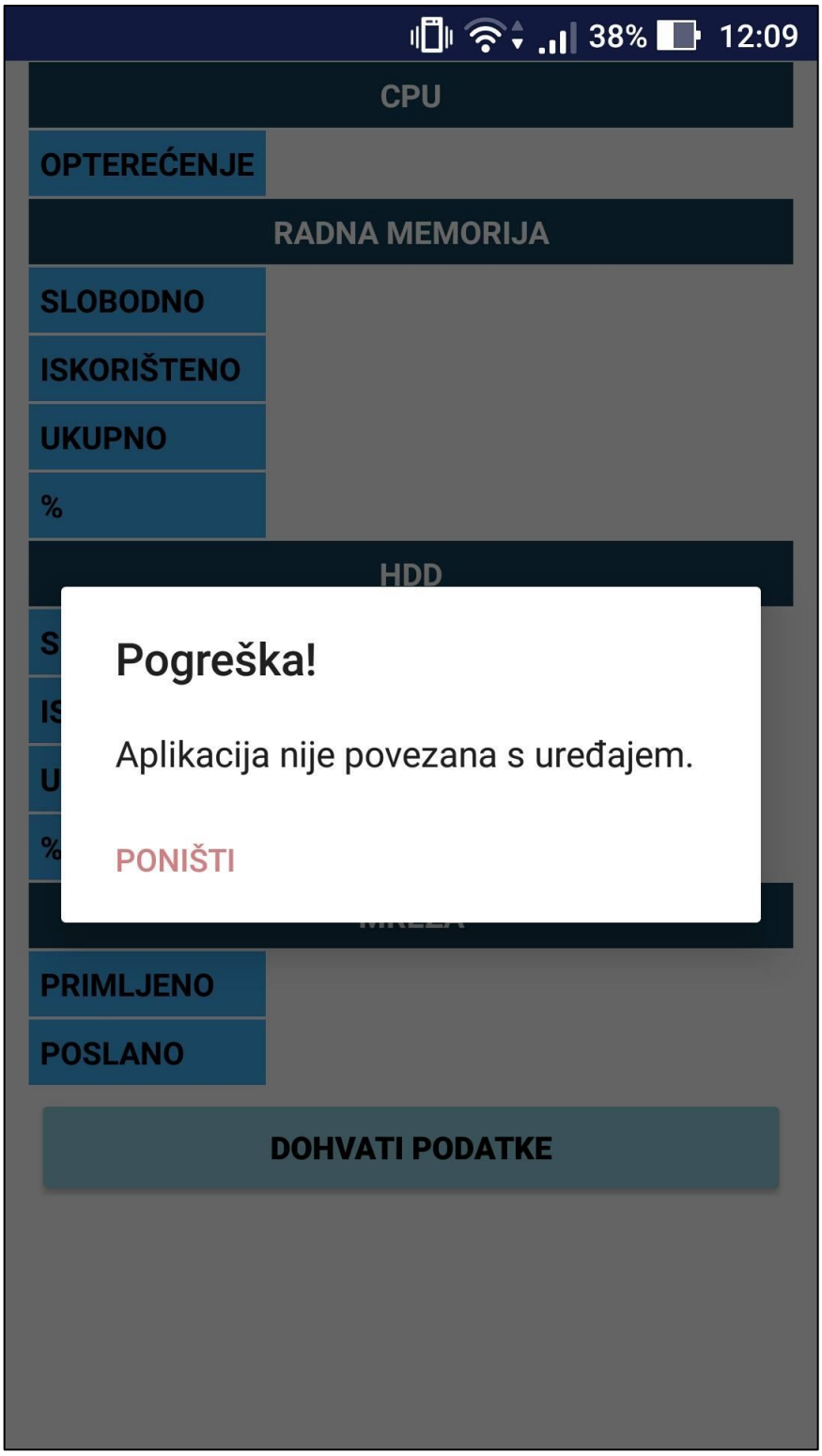


Sl. 4.1. Faze životnog ciklusa Android aplikacije

Kada se aplikacija pokrene, prikaže se korisničko sučelje, takvo da korisnik vidi nazive parametara koje može očitati pomoću aplikacije te prazna polja pored naziva, što je prikazano na slici 4.2.

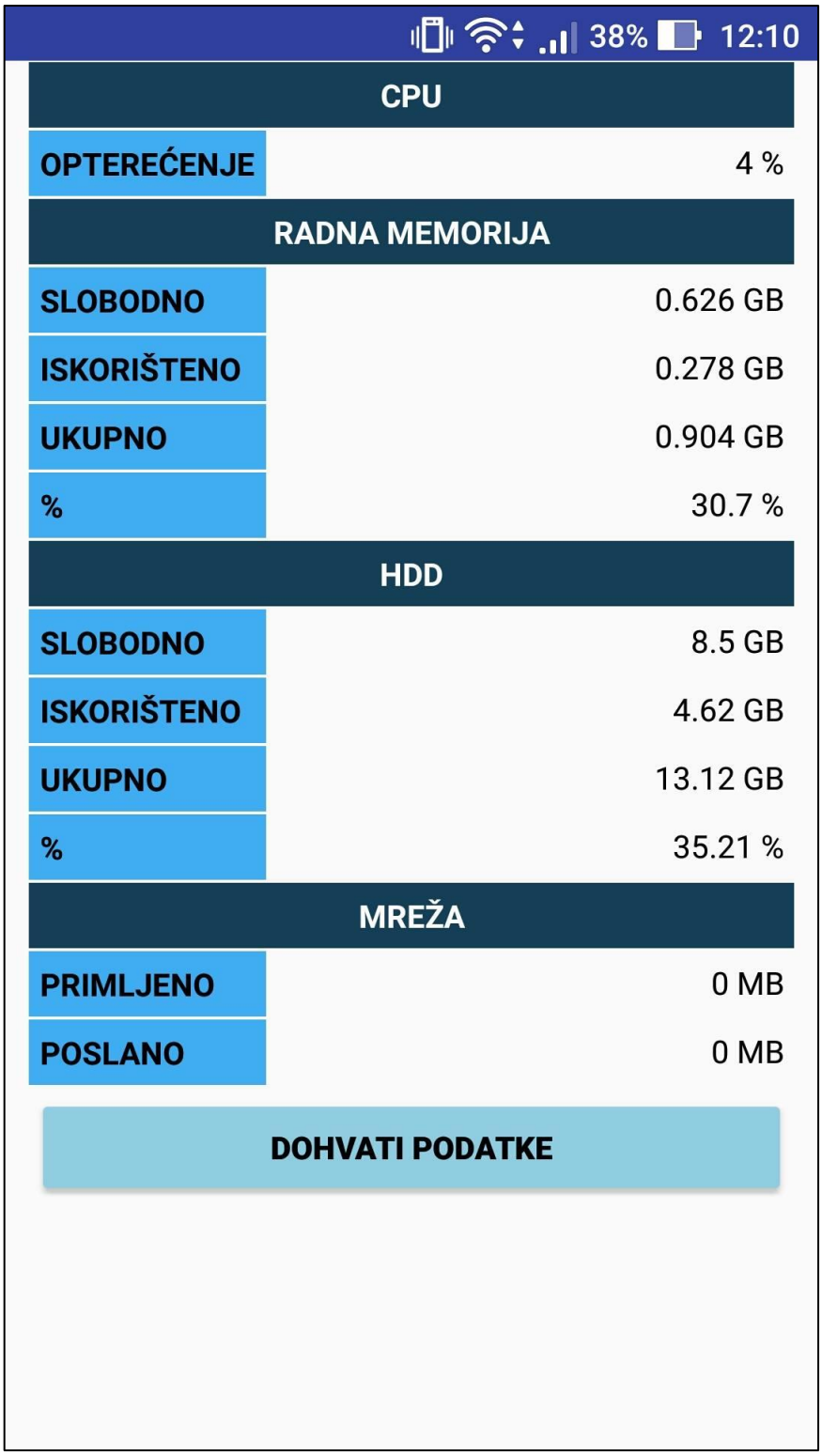
Sl. 4.2. Prikaz aplikacije pri njenom otvaranju

Kada se aplikacija otvara, ona prolazi kroz faze stvaranja, pokretanja i nastavljanja. Pritiskom gumba „DOHVATI PODATKE“, aplikacija provjerava povezanost s Raspberry Pi-jem.



Sl. 4.3. Prikaz aplikacije pri učitavanju podataka

Ukoliko je korisnik nije povezan WiFi-jem s Raspberry Pi-jem, prikazuje mu se poruka koja ga o tome obavještava, prikazano na slici 4.3. Nakon poništavanja poruke, korisnik može ponovno pokušati dohvatiti podatke.



Sl. 4.4. Prikaz aplikacije s učitanim parametrima

Ukoliko je korisnik povezan WiFi-jem s Raspberry Pi-jem, tada mu se prikazuju vrijednosti promatranih parametara. Budući da se pri svakom dohvaćanju podataka provjerava povezanost s Raspberry Pi-jem, ako se korisnik u bilo kojem trenutku, dok se provodi dohvaćanje podataka,

odspoji s uređaja, prikazuje se obavijest prikazana na slici 4.3. te se poništavanjem prikazuje korisničko sučelje prikazano na slici 4.2.

Ukoliko je provjera uspješno izvršena i korisnik je još uvijek povezan s uređajem, tada se dohvaćaju nove vrijednosti parametara. Takvim pristupom korisniku je pruženo automatsko osvježavanje promatranih podataka bez mogućnosti naglog prestanka rada aplikacije.

Kada se aplikacija uništi, oslobađa se memorija koja je bila potrebna za njen rad te se pri njenom ponovnom pokretanju korisniku prikazuje izgled prikazan na slici 4.2.

4.1. Usporedba s postojećim rješenjima

Budući da aplikacija, čija je izrada opisana u ovom radu, nije jedina takve vrste, nužno je objasniti prednosti i nedostatke već postojećih rješenja.

4.1.1. *PhpSysInfo*

PhpSysInfo je program koja nije namijenjen isključivo za Raspberry Pi, ali odlično radi i za njegovo nadziranje. Pisan je u PHP-u te, kao što je opisano u ovom radu, potrebno je imati instaliran Apache HTTP poslužitelj te PHP na promatranom uređaju.

PSIAndroid je aplikacija koja omogućava korištenje *PhpSysInfo* programa na Android uređaju. Aplikacija pruža grafičke prikaze i vrlo je jednostavna za korištenje.

4.1.2. *RPi-Monitor*

Iako je *PhpSysInfo* odličan program za nadziranje podataka, često korisniku nije potrebno znati sve informacije koje on nudi.

RPi-Monitor je jednostavna aplikacija koja pruža uvid u osnovne parametre. Budući da je distribuirana kao DEB paket, moguće ju je lako pokrenuti. Osim osnovnih informacija, prikazuje promatrane parametre i pomoću grafova.

Iako *RPi-Monitor* nije Android aplikacija, već web aplikacija, prikaz na malim zaslonima je vrlo dobar.

4.1.3. *Raspcontrol*

Raspcontrol je jedno od prvih rješenja izrađeno isključivo za Raspberry Pi. Slično kao i aplikacija opisana u ovom radu, pisana je u PHP-u te zahtjeva instalaciju Apache internetskog poslužitelja i PHP-a.

Aplikacija pruža jednostavno korisničko sučelje s prikazom osnovnih podataka o radu uređaja te korištenje grafova.

Slično kao *RPi-Monitor*, to je web aplikacija koja dobro radi i na malim zaslonima.

4.1.4. *Raspberry Stats*

Raspberry Stats je jednostavna aplikacija u smislu funkcionalnosti. Odlična je za korisnike kojima trebaju isključivo osnovne informacije. Izrađena je pomoću Node.js-a te je za njeno korištenje također potrebno instalirati potrebne pakete.

Međutim, aplikacija radi isključivo na portu 80, što znači ukoliko korisnik već ima pokrenut poslužitelj na Raspberry Pi-ju koji koristi drugi port, potrebno je to prilagoditi aplikaciji.

5. ZAKLJUČAK

Zadatak ovog rada je bio izraditi Android aplikaciju pomoću koje će biti moguće očitavati parametre rada SoC uređaja. Tražena Android aplikacija treba biti jednostavna za korištenje, kod čega je potrebno obratiti pozornost na jednostavnost i dokumentiranost postavljanja poslužitelja. Da bi se olakšalo korištenje aplikacije na drugom SoC uređaju, bilo je potrebno koristiti poznate pakete i tehnologije, kompatibilne s raznim platformama i operacijskim sustavima. Zadatak je izvršen pomoću PHP skripte na SoC uređaju te same Android aplikacije, pisane u Java programskom jeziku, koja podatke očitava i ispisuje.

Na Raspberry Pi-ju je postavljen Apache HTTP poslužitelj. Za povezivanje klijentske i poslužiteljske strane, korišteno je bežično povezivanje putem WiFi-ja jer omogućuje brzi prijenos potrebnih podataka koji su prikazani na Apache HTTP poslužitelju. Da bi se parametri rada mogli očitavati izvan pristupa Internetu, na poslužitelju je postavljena bežična pristupna točka. PHP skriptom se jednostavno očitavaju parametri rada te se uređaj ne opterećuje. Budući da je JSON lak za parsiranje, korišten je za prikaz rezultata. Sama aplikacija je izrađena pomoću razvojnog okruženja *Android Studio*. Da bi bilo moguće izvršavati operacije pomoću mreže, bilo je potrebno koristiti *AsyncTask* klasu koja s asinkronim pristupom, za razliku od višenitnog pristupa, ne stvara novu nit za izvršavanje programa te je pogodna za jednostavne operacije u mreži koje ne zahtjevaju preuzimanje velike količine podataka. *JSONParser* klasa dohvaća JSON vrijednosti pomoću URL-a poslužitelja, njih parsira i ispisuje u tablici XML datoteke.

Programsko rješenje izvedeno u sklopu rada omogućuje povezivanje s poslužiteljem. Nakon uspješnog povezivanja je moguće poslati zahtjev za dohvaćanje podataka o performansama poslužitelja. Također, prije svakog slanja zahtjeva za dohvaćanje podataka se provjerava povezanost klijentske i poslužiteljske strane. Ovo rješenje se može, osim na Raspberry Pi-ju, primijeniti na drugim SoC uređajima. Po uzoru na rješenje prikazano u ovom radu, mogu se izraditi aplikacije za rad na drugim operacijskim sustavima ili web aplikacija optimizirana za rad na malim zaslonima. Prednosti navedenog programskog rješenja su jednostavno postavljanje i mogućnost korištenja izvan pristupa Internetu. Rješenje prikazano u ovom radu je pogodno za najveći broj korisnika, budući da je Android najpopularniji operacijski sustav za pametne telefone te je *Raspberry Pi 3 Model B* trenutno najpopularniji SBC. Navedeno programsko rješenje moguće je, po uzoru na već postojeća rješenja, nadograditi s grafičkim prikazom podataka te pružiti prikaz podataka korištenjem ugrađenih senzora u Raspberry Pi-ju.

LITERATURA

[1] A. Pajankar, Raspberry Pi Supercomputing and Scientific Programming: MPI4PY, NumPy, and SciPy for Enthusiasts. Apress, 2017.

[2] „Raspberry Pi - Teach, Learn, and Make with Raspberry Pi“, Raspberry Pi. [Na internetu]. Dostupno na: https://www.raspberrypi.org/. [Pristupljeno: 02-lip-2017].

[3] „Bluetooth vs Wi-Fi - Difference and Comparison | Diffen“. [Na internetu]. Dostupno na: http://www.diffen.com/difference/Bluetooth_vs_Wifi. [Pristupljeno: 02-lip-2017].

[4] R. Golden, Raspberry Pi Networking Cookbook. Packt Publishing Ltd, 2016.

[5] D. Lowe, Networking All-in-One Desk Reference For Dummies. John Wiley & Sons, 2008.

[6] P. J. Kula, Raspberry Pi 2 Server Essentials. Packt Publishing Ltd, 2016.

[7] „Easy LAMP Server Installation“, Linux.com | The source for Linux information, 23-velj-2010. [Na internetu]. Dostupno na: https://www.linux.com/learn/easy-lamp-server-installation. [Pristupljeno: 08-lip-2017].

[8] J. Valade, PHP and MySQL For Dummies. John Wiley & Sons, 2009.

[9] „PHP: shell_exec - Manual“. [Na internetu]. Dostupno na: http://php.net/manual/en/function.shell-exec.php. [Pristupljeno: 10-lip-2017].

[10] „PHP: JSON - Manual“. [Na internetu]. Dostupno na: http://php.net/manual/en/book.json.php. [Pristupljeno: 10-lip-2017].

[11] „Android Developers“. [Na internetu]. Dostupno na: https://developer.android.com/index.html. [Pristupljeno: 14-lip-2017].

[12] D. Lowe i B. A. Burd, Java All-In-One Desk Reference For Dummies. John Wiley & Sons, 2011.

[13] H. Vasconcelos i S. Liles, Asynchronous Android Programming. Packt Publishing, Limited, 2016.

[14] „The Activity Lifecycle | Android Developers“. [Na internetu]. Dostupno na: https://developer.android.com/guide/components/activities/activity-lifecycle.html. [Pristupljeno: 25-lip-2017].

SAŽETAK

Najveća kvaliteta nekog SoC uređaja, kao što je Raspberry Pi, je njegova veličina. Zbog male veličine dodavanjem većih perifernih uređaja gubi se na njegovoj kompaktnosti i prenosivosti. Iz tog razloga korisniku nije u interesu imati uređaj spojen na monitor radi nadziranja njegovih performansi. Stoga je idealno rješenje imati Android aplikaciju koja će omogućiti povezivanje s takvim uređajem i prikaz korisniku potrebnih podataka. Aplikacija i SoC uređaj se bežično povezuju putem WiFi-ja. Njihova povezanost se, radi ispravnosti rada aplikacije, provjerava prije svakog dohvaćanja podataka.

U ovom radu je opisan postupak podešavanja poslužiteljske strane, što uključuje postavljanje poslužitelja na SoC uređaju i stvaranje PHP skripte za očitavanje potrebnih podataka te njihovo formatiranje u JSON oblik. Nadalje, opisana je izrada Android aplikacije pomoću razvojnog okruženja *Android Studio* korištenjem Java programskog jezika i XML opisnog jezika.

Ključne riječi: System on Chip, Raspberry Pi, Android, PHP, bežična pristupna točka

ABSTRACT

Android Application for Monitoring SoC Device Performance

The biggest quality of a SoC device, such as Raspberry Pi, is it's size. Due to its small size by connecting larger peripherals it could lose its compactness and portability. Therefore, it is not in the user's interest to have the device connected to the monitor in order to observe its performance. The ideal solution is to have an Android application that is able to connect to the SoC device and show the user desired performance statistics. The application and the SoC device connect wirelessly via WiFi. Their connectivity is checked before each data fetching due to the application's functionality.

In this thesis, the procedure of setting up the server side is described, which includes setting up the server on the SoC device and creating the PHP script for reading the desired parameters and converting them into JSON format. Furthermore, the making of the Android application using Android Studio development environment, Java programming language and XML markup language is described.

Keywords: System on Chip, Raspberry Pi, Android, PHP, wireless access point

ŽIVOTOPIS

Ivana Pristavnik je rođena 18. listopada 1995. godine u Osijeku, Hrvatska. Srednju školu, II. Gimnaziju, završava 2014. godine u Osijeku. Iste godine upisuje sveučilišni preddiplomski studij Računarstvo na Elektrotehničkom Fakultetu u Osijeku.

Ivana Pristavnik