IoT sustav za nadzor potrošnje energenata u kućanstvu s naglaskom na sigurnost

Marđetko, Goran

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Goran Marđetko

IOT SUSTAV ZA NADZOR POTROŠNJE ENERGENATA U KUĆANSTVU S NAGLASKOM NA SIGURNOST

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Goran Marđetko

JMBAG: 0016141750

Studij: Informacijski sustavi

IOT SUSTAV ZA NADZOR POTROŠNJE ENERGENATA U KUĆANSTVU S NAGLASKOM NA SIGURNOST

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Igor Tomičić

Varaždin, rujan 2023.

Goran Marđetko

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Tema ovog rada je razvoj sustava za potrošnju energenata u kućanstvu. Naglasak je na korištenju IoT tehnologija te na sigurnosnim elementima sustava. Cilj rada je korisniku sustava na jednostavan i pregledan način prikazati informacije o potrošnji energije. Uz razvoj sustava, opisane su i tehnologije potrebne za ostvarivanje ovog projekta. Uz to, cilj rada je nadograditi postojeće znanje iz područja baza podataka, dizajna i razvoja mobilnih te web aplikacija.

Ključne riječi: Internet stvari, programiranje, ESP32 Cam, sigurnost, aplikacija

Sadržaj

1. Uvod

U današnjem digitalnom dobu IoT (eng. *Internet of Things*) neophodan je koncept koji se sve češće koristi. Susrećemo ga kao ugrađen koncept u već postojeće tehnologije (npr. pametan televizor, pametan frižider...) ili kao dodatak postojećim uređajima (npr. pametna utičnica, pametni prekidač...) [1].

Cilj projekta je razviti uređaj i program koji će koristeći razne tehnologije omogućiti korisnicima jasan pregled potrošnje energenata. Razvojem IoT uređaja za nadzor potrošnje energenata u kućanstvu nastoji se smanjiti potrošnja istih. Glavna pretpostavka je da će se više pažnje pridavati na količinu potrošenih energenata ako je pregled potrošnje lako dostupan i razumljiv. Glavna svrha ovog sustava je informirati korisnike o količini potrošene energije, a njegova implementacija u kućanstvu pruža niz prednosti. Korisnici dobivaju predodžbu o potrošnji energije, a uvođenjem navika za štednju mogu smanjiti račune za istu te povećati energetsku učinkovitost u kućanstvu. Osim toga, takav IoT sustav pridonosi održivosti i zaštiti okoliša jer pomaže korisnicima da jasnije vide njihov utjecaj na okoliš.

Ovaj sustav je moguće nadograditi te ga potencijalno koristiti u svrhu očitanja brojila na daljinu, a kao rezultat toga, tvrtke koje pružaju energiju potrošačima ne bi morale izlaziti na teren kod očitanja i kontrole brojila. Time bi se moglo uštedjeti vrijeme potrebno za očitanje, gorivo potrebno za dolazak na lokaciju brojila, a isto tako izbjegle bi se situacije kao što je nemogućnost pristupa brojilu (potrošač energije nije kod kuće) što zahtjeva ponovni izlazak na teren ili javljanje stanja brojila putem telefonskog poziva.

Motivacija za odabir ove teme je velik interes vezan za projekte koji uključuju softverske i hardverske komponente. Ova dinamična interakcija između softvera i hardvera omogućuje stvaranje inovativnih proizvoda i rješenja koja oblikuju način na koji živimo, radimo i komuniciramo. Jedan od razloga velike zainteresiranosti za ovu temu jest činjenica da su mnoge svakodnevne tehnološke naprave rezultat spoja softverskih i hardverskih elemenata. Pametni telefoni, računalni sustavi, pametni uređaji za domaćinstvo i autonomna vozila samo su neki od primjera. Ovakvi projekti zahtijevaju znanje iz različitih područja kako bi se osiguralo usklađivanje između programskih rješenja i fizičkih komponenti.

2. Metode i tehnike rada

U ovom poglavlju su opisane metode i tehnike koje se koriste prilikom razvoja sustava za nadzor potrošnje energenata. Uz to, opisani su programi, aplikacije i tehnologije koje su potrebne za uspješan razvoj sustava.

Kod odabira programa i tehnologija obraćao sam pažnju na činjenicu da su oni otvorenog koda (eng. *open source*). To omogućuje korisnicima da imaju slobodan pristup izvornom kodu te da ga prema potrebi izmjenjuju. Bitno je naglasiti da su programi otvorenog koda sigurni i transparentni jer omogućuju korisnicima pregled i provjeru koda koji čini sam program. To omogućuje kontinuirani rad i poboljšanje programa te osigurava korisniku mogućnost nastavka korištenja tog programa, iako ga izvorni kreator više neće nadograđivati i podržavati [2].

2.1. KiCad

KiCad je besplatan program otvorenog koda koji je bio potreban za izradu električne sheme, dizajna tiskane pločice te za izradu datoteke za preslikavanje sheme na bakrenu pločicu. *KiCad* je skraćenica za "Kicad Electronic Design Automation" te se može koristiti na mnogim poznatim operativnim sustavima. Program pruža sve alate i funkcionalnosti potrebne za dizajn i razvoj tiskane pločice. U programu su dostupne sve potrebne elektroničke komponente te njihove dimenzije što osigurava da korisnik ne može postaviti komponente na način na koji bi se preklapale u fizičkom svijetu [3].

2.2. Arduino IDE

Arduino IDE je besplatno softversko okruženje otvorenog koda koje se koristi za programiranje mikrokontrolera. Ovaj program je odabran zbog pozitivnog prethodnog iskustva te je on najbolje rješenje za ovu primjenu. Ovaj softver omogućuje pisanje, uređivanje i prevođenje programskog koda što rezultira jednostavnim i brzim prototipiranje i razvojem projekata s hardverskim i softverskim komponentama [4].

2.3. Web tehnologije

Za ostvarenje ovog projekta korišteno je više tehnologija koje se mogu svrstati u kategoriju web tehnologija. U nastavku su opisane te objašnjene primjene svake od njih.

2.3.1. XAMPP

XAMPP je softverski paket koji pruža okruženje za razvoj i testiranje web aplikacija na lokalnom računalu. Naziv *XAMPP* označava komponente od kojih se on sastoji a to su *Apache*, *MySQL*, *PHP* i *Perl*. U nastavku su opisane komponente *XAMPP-a* koje su korištene pri izradi ovog projekta [5].

2.3.2. Apache

Apache je web poslužitelj koji omogućuje rad i razvoj na web stranicama koje putem njega pokrećemo na našem lokalnom računalu. Smatra se najpoznatijim i najkorištenijim web poslužiteljom otvorenog koda. Razvijen je kao projekt Apache Software Foundation te se može koristiti na mnogim platformama (Windows, Unix, Linux, macOS). Podržava protokole kao što je HTTPS koji sam koristio u ovom projektu [6].

2.3.3. mySQL

Za spremanje podataka potrebnih za ostvarenje projekta odabran je *MySQL*. To je jedan on najpopularnijih sustava za upravljanje relacijskim bazama. Ovaj sustav je također otvorenog koda. Koristi se za pohranjivanje, upravljanje i baratanje podacima u relacijskom formatu. Kao što ime predlaže, *MySQL* koristi "Structured Query Language" za baratanje podacima u bazi. Omogućuje korisnicima izvršavanje raznih operacija kao što su upiti, umetanje, ažuriranje i brisanje podataka [7].

2.3.4. PHP

PHP je skriptni jezik otvorenog koda koji se koristi za obavljanje aktivnosti na strani servera. *PHP* ima ugrađene funkcije i biblioteke koje omogućuju komuniciranje i rad s bazama podataka, rad s mrežnim protokolima (HTTP,HTTPS), obrada i analiza slike te ostale funkcionalnosti. U ovom projektu *PHP* je korišten za slanje upita bazi podataka te za komunikaciju s mobilnom aplikacijom i samim uređajem za nadzor potrošnje energije putem HTTPS protokola [8].

2.4. Android studio

Za razvoj mobilne aplikacije korišten je programski paket *Android Studio*. To je integrirano razvojno okruženje (eng*. integrated development environment*) koje služi razvoju aplikacija za uređaje koji koriste Android operacijski sustav. Razvijen je od strane Google-a te sadrži komponente potrebne za izradu, testiranje te ispravljanje grešaka u razvoju aplikacije. Omogućuje vizualno i programsko stvaranje grafičkog sučelja, nudi mogućnost za automatsko dovršavanje koda te ima mogućnost korištenja ugrađenog emulatora za testiranje aplikacije na raznim uređajima [9].

3. Dizajn i razvoj uređaja

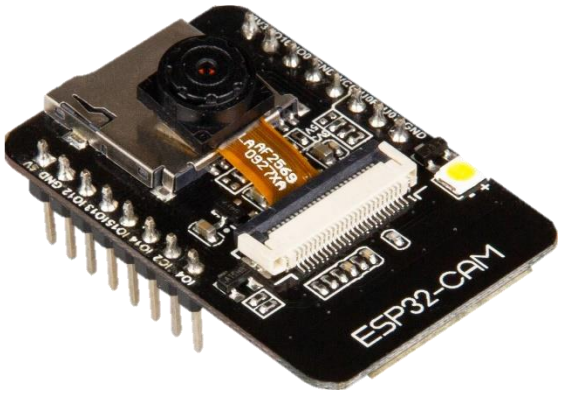
Nakon opisa tehnologija potrebnih za ostvarivanje ovog projekta krećemo s razvojem. U sljedećim odlomcima bit će opisane hardverske komponente te softverska rješenja koja čine ovaj projekt.

3.1. Hardverske komponente

Za ostvarenje ovog projekta korišteno je nekoliko hardverskih komponenti. U nastavku je opisana svrha te specifikacije svake od njih.

3.1.1. ESP32-CAM

Glavnu hardversku komponentu ovog sustava čini mikrokontroler ESP32-Cam. To je nasljednik ESP8266 mikrokontrolera te sadrži sve što je potrebno za razvoj ovog IoT projekta. Omogućuje korištenje niskoenergetske Wi-Fi vezu, sadrži integriranu kameru i blic te digitalne i analogne ulaze/izlaze potrebne za komunikaciju s ostalim komponentama.



Slika 1: ESP32-Cam (Izvor: https://asset.conrad.com/media10/isa/160267/c1/-/hr/002332111PI00/image.jpg)

Tablica 1-Potrošnja ESP32-Cam-a

Mode	Potrošnja
Rad bez blica	180mA na 5V
Rad s blicom	310mA na 5V
Deep-Sleep	6mA na 5V
Modern-Sleep	20mA na 5V
Light-Sleep	6.7mA na 5 V

(Izvor: https://docs.ai-thinker.com/en/esp32-cam)

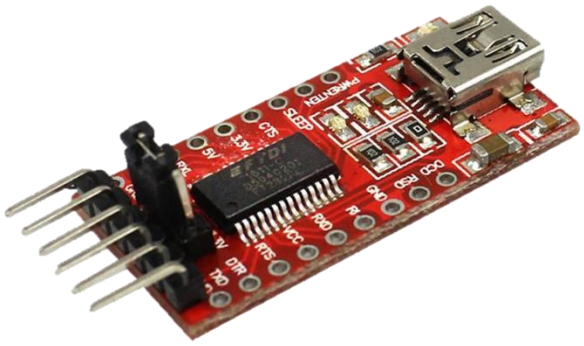
Tablica 2- Specifikacije ESP32-Cam-a

Wi-Fi modul	ESP32
Flash memorija	32 Mbit
Antena	Ugrađena
Wi-Fi protokol	IEEE 802.11 b/g/n/e/i
Izlazno ulazni portovi	9
Napajanje	5V
Sigurnost	WPA/WPA2/WPA2-Enterprise/WPS
Sučelje za komunikaciju	UART/SPI/I2C/PWM

(Izvor: https://docs.ai-thinker.com/en/esp32-cam)

3.1.2. FTDI FT232R

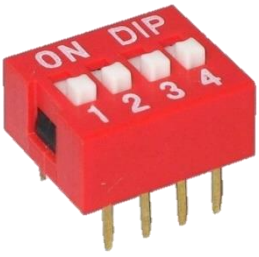
U ovom projektu je korišten FTDI FT232R. To je komponenta koja omogućuje komunikaciju između računala i uređaja (ESP32-Cam) putem USB priključka. S obzirom na to da ESP32-Cam nema USB priključak bilo je neizbježno koristiti ovu komponentu. Koristeći tu komponentu prenosio sam program s računala na ESP32-Cam te sam putem nje mogao na računalu vidjeti odgovore i greške sa ESP32-Cam-a te ih tako ispraviti.



Slika 2: FTDI FT232R (Izvor: behind-the-scenes.net/wp-content/uploads/FTDI-USB-UART-module.jpg)

3.1.3. DIP switch

Za potrebe isključivanja i uključivanja nekih veza na pločici odabrana je komponenta naziva „DIP switch“. To je komponenta koja služi kao mehanički prekidač u elektronici te omogućuje ručno postavljanje stanja prekidača u svrhu mijenjanja postavki uređaja. Sastoji se od jednog ili više prekidača smještenih u plastičnom kućištu. Svaki prekidač se može nalaziti u jednom od dva stanja - uključeno ili isključeno.

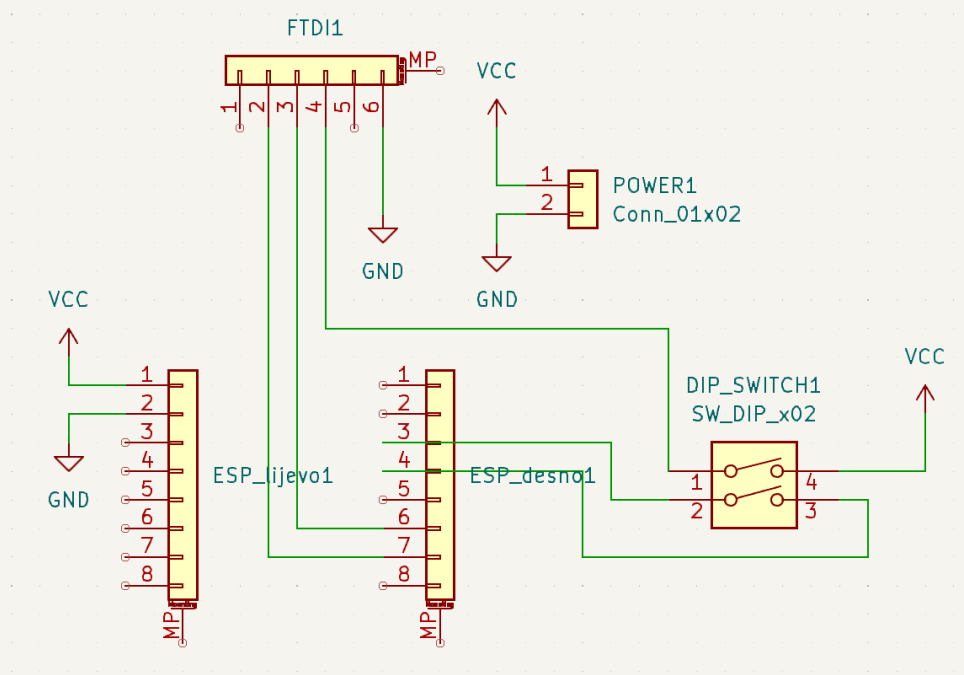


Slika 3: DIP Switch (Izvor: smarttronik.com/1506-large_default/red-4-position-dip-switch.jpg)

3.2. Izrada uređaja

3.2.1. Shema uređaja

Nakon uspješne izrade prototipa na eksperimentalnoj pločici slijedi izrada tiskane pločice koja će sadržavati sve komponente i priključke potrebne za rad uređaja. Za to sam koristio prethodno opisani program *KiCad* [1] . Nakon dodavanja svih potrebnih komponenti u program spojene su kao što su bile spojene na eksperimentalnoj pločici.

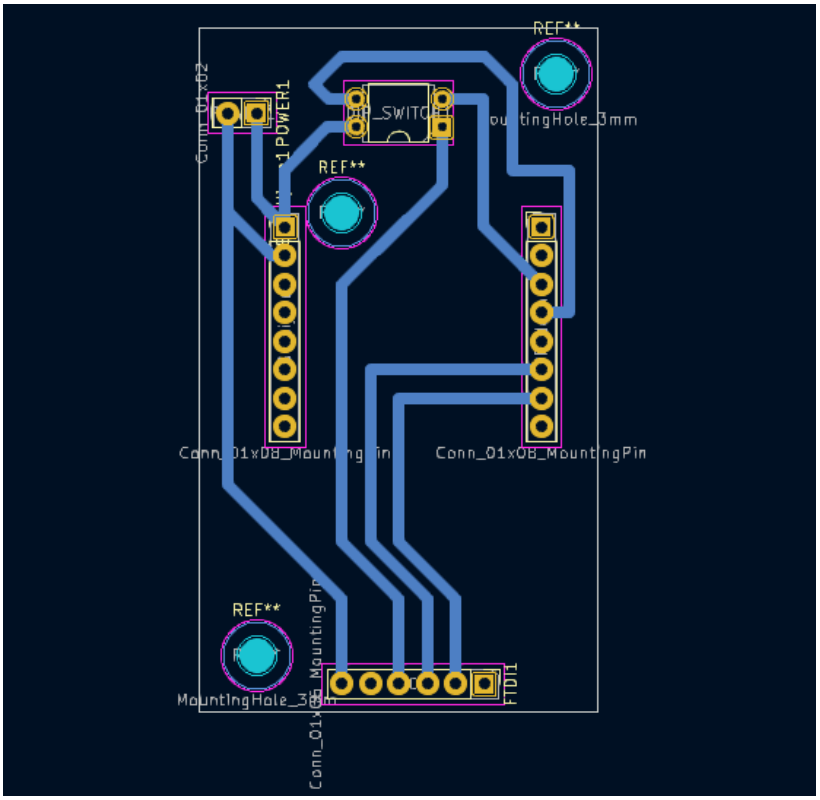


Slika 4: Shema uređaja (Autorski rad)

Priključak za napajanje „POWER1“ spojen je na izvor napajanja od 5V te na uzemljenje. Pinovi 1 i 2 na headeru „ESP_lijevo1“ služe za napajanje uređaja, pin 1 je spojen na izvor napajanja od 5V dok je pin 2 spojen za zajedničko uzemljenje. Pinovi 6 i 7 na headeru „ESP_desno1“ služe za serijsku komunikaciju ESP32-Cam-a sa FTDI FT232R uređajem. Pin 6 spojen je na pin 3 na headeru „FTDI1“ dok je pin 7 spojen na pin 2. Pin 4 na headeru „FTDI1“ spojen preko dip switcha na izvor napajanja od 5V dok je pin 6 spojen na uzemljenje. Pinovi 3 i 4 na headerima „ESP_desno1“ spojeni su u briku preko dip switcha. Te dvije veze spojene su na dip switch za potrebe programiranja i rada uređaja. Ako su prekidači na dip switchu uključeni onda je uređaj u modu za programiranje, ako ih isključimo onda je uređaj u modu za rad.

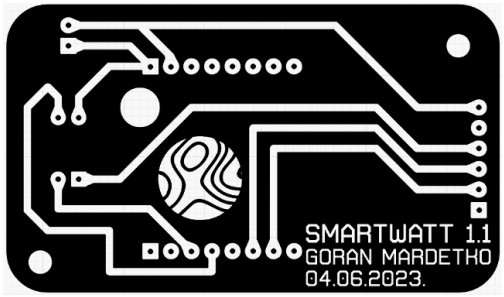
3.2.2. Izrada tiskane pločice

Prvi korak je postavljanje komponenata na skici tiskane pločice. Komponente su posložene tako da zauzimaju što manje prostora. Nakon toga su prema prije napravljenoj shemi dodane veze između komponenti.

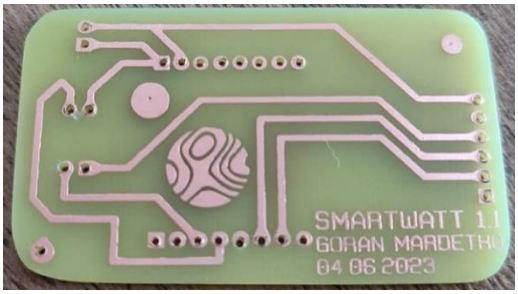


Slika 5: Dizajn tiskane pločice (Autorski rad)

Potom je dizajn izvezen (eng. *exported*) u obliku SVG datoteke te je dorađen u smislu dodavanja imena pločice, kreatora i datuma. Nakon toga je negativna verzija dizajna laserom prenesena na bakrenu pločicu. Bitno je prije toga vertikalno okrenuti dizajn jer je bakrena strana s donje strane tiskane pločice dok je komponente potrebno umetnuti s gornje strane. Koristeći željezov(III) klorid uklonjen je bakar na onim mjestima gdje nema veze između pinova te su svrdlom promjera 1 milimetar izbušene rupe na predviđenim mjestima.



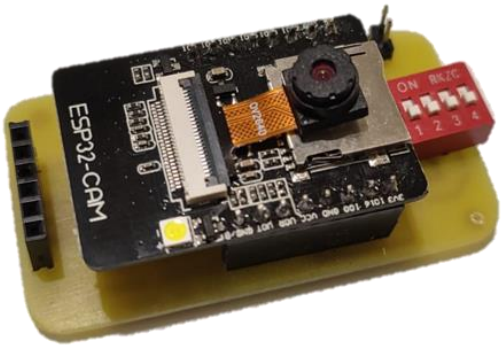
Slika 6: Dizajn za jetkanje (Autorski rad)



Slika 7: Gotova pločica (Autorski rad)

Nakon toga su komponente postavljene i zalemljene na predviđena mjesta te je pločica spremna za rad.

Slika 9: Uređaj u kućištu (Autorski rad)



Slika 8: Gotov uređaj (Autorski rad)

4. Programski kod uređaja

U ovom djelu će biti objašnjeni glavni dijelovi koda koji čine programski dio (eng. *firmware*) uređaja. Kod je pisan i preveden (eng. *compiled*) u prethodno opisanom *Arduino* [2] integriranom okruženju za razvoj.

Program počinje spremanjem podataka za povezivanje na Wi-Fi mrežu. Spremama se naziv i lozinka mreže na koju se uređaj povezuje. Također je spremljena adresa lokalnog servera kojem će uređaj kasnije slati fotografije. Spremljen je i port 443 preko kojeg se odvija HTTPS komunikacija. Definiran je pin za korištenje integriranog blica, vrijeme spavanja uređaja odnosno interval slikanja te faktor pretvorbe iz mikrosekundi u sekunde.

```
const char * ssid = "*********";
const char * password = "*********";
String mojServer = "192.168.1.X";
String putanjaSlika = "/upload.php";
const int serverPort = 443;

int flashPin = 4;

#define uS_TO_SEC 1000000ULL
#define VRIJEME_SPAVANJA  86400
```

4.1.1. Postavljanje postavki uređaja za rad

U ovoj cjelini bit će opisani i objašnjeni kod koji čini Setup() funkciju. Cilj funkcije je postavljanje postavki za rad uređaja. Na početku je postavljena brzina serijske komunikacije s uređajem te brzina za prijenos podataka. Definiran je „flashPin“ kao izlazni pin. Uređaj je postavljen u način rada „WIFI_STA“ što predstavlja „Station mode“ odnosno omogućuje da se uređaj može spojiti na WiFi mrežu te komunicirati s ostalim uređajima na mreži (u ovom slučaju s računalom koje ima ulogu servera). Na serijski monitor je ispisan naziv mreže na koju se uređaj spaja te je nakon uspješnog spajanja ispisana IP adresa samog uređaja.

```
Serial.begin(115200);
pinMode(flashPin, OUTPUT);
WiFi.mode(WIFI_STA);
Serial.println();
Serial.print("Spajanje na ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
Serial.println();
```

```
Serial.print("ESP32-CAM IP adresa: ");
Serial.println(WiFi.localIP());
```

Nakon toga postavljena je konfiguracija kamere, odabrana je veličina slike XGA (1024 x 768 piksela) te je kvaliteta postavljena na 5 (raspon 0-63) s time da manji broj predstavlja veću kvalitetu. U odnosu na klasičnu ESP32 pločicu koja ima 512 KB RAM memorije, ESP32 Cam sadrži dodatan vanjski RAM veličine 4MB koji joj omogućuje fotografiranje. Moguće je odabrati i veću rezoluciju (npr. UXGA ili SXGA) no onda slike nisu potpune zbog nedostatka RAM memorije. Ove postavke odabrane su zbog najboljeg omjera kvalitete slike i pouzdanosti stvaranja uspješne fotografije.

```
config.frame_size = FRAMESIZE_XGA;
config.jpeg_quality = 5;
config.fb_count = 2;
```

Stvorena je varijabla tipa esp_err_t, naziva err koja će zabilježiti grešku kod inicijalizacije kamere. Ako greška postoji, odnosno varijabla err je različita od „ESP_OK“, ispisuje se ispisuje greška te je uređaj resetiran.

```
esp_err_t err = esp_camera_init( & config);
if (err != ESP_OK) {
  Serial.printf("Greska kod inicijalizacije kamere 0x%x", err);
  delay(1000);
  ESP.restart();
}
```

Fotografiranje brojila će se obavljati jednom dnevno te taj proces traje oko 5 sekundi što znači da ostalih 86395 sekundi u danu uređaj miruje. Tijekom rada uređaj troši oko 180mA što nije zanemarivo zbog toga što je izvor napajanja baterija. Iz tog razloga je uređaj tijekom vremena mirovanja postavljen u „deep sleep“ način rada te troši samo 150 μA. Kod korištenja „deep sleep-a“ funkcija „loop“ se ne izvršava. Nakon poziva funkcija fotografiraj postavljeno je vrijeme buđenja uređaja (nakon 24 sata) te započinje deep sleep.

```
fotografiraj();
esp_sleep_enable_timer_wakeup(VRIJEME_SPAVANJA * uS_TO_SEC);
Serial.printf("Ulazim u deepsleep");
esp_deep_sleep_start();
```

4.1.2. Fotografiranje brojila

Pin flashPin je postavljen na vrijednost HIGH (upaljen blic) te je dodana vremenska odgoda od 70ms prije i nakon snimanja fotografije što osigurava dobro osvjetljenje. Između te dvije odgode snima se fotografija. Naredbom esp_camera_fb_get() snimljena fotografija se sprema u memoriju. Varijabla fb je pokazivač na memorijsku lokaciju na kojoj se nalazi fotografija. Nakon toga se provjerava status varijable fb te ako je ona prazna ispisana je greška te je uređaj resetiran.

```
camera_fb_t * fb = NULL;
digitalWrite(flashPin, HIGH);
delay(70);
fb = esp_camera_fb_get();
delay(70);
digitalWrite(flashPin, LOW);

if (!fb) {
  Serial.println("Greska kod slikanja");
  delay(1000);
  ESP.restart();
}
```

Nakon uspješno spremljene fotografije potrebno je poslati ju na web server. Povezujemo se na server čije smo podatke ranije spremili. U zaglavlju je naveden naziv i vrsta datoteke koju šaljemo. U ovom slučaju naziv je „esp32-cam.jpg“ a vrsta datoteke je „image/jpeg“. Izračunata je veličina podataka koji se šalju (veličina fotografije, zaglavlja i „tail“). HTTPS POST zahtjevom podaci su poslani na server.

```
  if (client.connect(mojServer.c_str(), serverPort)) {
    Serial.println("Connection successful!");
    String head = "--ESP32\r\nContent-Disposition: form-data;
name=\"imageFile\"; filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
    String tail = "\r\n--ESP32--\r\n";

    uint32_t imageLen = fb -> len;
    uint32_t extraLen = head.length() + tail.length();
    uint32_t totalLen = imageLen + extraLen;

    client.println("POST " + putanjaSlika + " HTTPS/1.1");
    client.println("Host: " + mojServer);
    client.println("Content-Length: " + String(totalLen));
    client.println("Content-Type: multipart/form-data; boundary=ESP32");
    client.println();
    client.print(head);
```

Nakon slanja podataka na server, snimljena fotografija izbrisana je iz memorije uređaja.

```
esp_camera_fb_return(fb);
```

Na serijski monitor ispisan je odgovor servera. Ako slanje nije uspjelo ispisuje se poruka o neuspjelom slanju.

```
  Serial.println(podaciBody);
  } else {
    podaciBody = "Povezivanje na " + mojServer + " neuspjesno.";
    Serial.println(podaciBody);
  }
```

5. Programski kod web aplikacije

Na serveru se nalazi nekoliko *PHP* [6] datoteka koje obrađuju primljene fotografije te služe za komunikaciju mobilne aplikacije s bazom podataka. U sljedećim odlomcima opisan je rad svake od njih.

5.1.1. Komunikacija između mobilne aplikacije i baze podataka

Ova datoteka služi za komunikaciju između mobilne aplikacije i baze podataka. Preko lokalnog servera aplikacija šalje HTTPS zahtjev serveru te on na temelju podataka iz zahtjeva šalje upit bazi podataka. Nakon toga baza vraća odgovor serveru koji ga šalje aplikaciji te se na temelju primljenih podataka prikazuju razne statistike o potrošnji energenata.

Kod započinje spremanjem podataka potrebnih za spajanje na bazu podataka. Spremljen je naziv servera, korisničko ime, lozinka te naziv baze. Nakon toga uključene su postavke za prikaz grešaka kod rada s bazom podataka. Stvorena je nova vezu s bazom podataka koristeći prethodno spremljene podatke o bazi. Izvršena je provjera greške kod povezivanja te ako postoji je ispisana.

```
$servername = "localhost";
$username = "********";
$password = "********";
$dbname="********";

mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
if ($conn->connect_error) {
  die("Povezivanje nije uspijelo: " . $conn->connect_error);}
```

Stvorene su varijable upitKod te datum. One su potrebne za određivanje upita koji će se slati bazi. Program čeka POST zahtjev s podacima o identifikacijom broju uređaja i kodu za upit. Ako su ti podaci uspješno poslani POST zahtjevom, spremljeni su u odgovarajuće varijable za daljnju upotrebu.

```
$upitKod=0;
$datum=date("Y-m-d");

if( $_POST["upitKod"] && $_POST["id_uredaj"])
  {

        $upitKod= $_POST["upitKod"];
        $id_uredaj=$_POST["id_uredaj"];

//ostatak  koda u nastavku
}
```

Kroz aplikaciju se šalje četiri zahtjeva te su oni obrađeni u kodu ispod. Prvi zahtjev se odnosi na trenutno stanje brojila. Za slanje upita bazi korišteni su pripremljeni izrazi (eng. *prepared statements*) o kojima će više biti rečeno u odjeljku o sigurnosti. Prvi upit odabire redak s najvećim identifikacijskim brojem u tablici, odnosno najnoviji zapis. Također taj zapis mora imati i odgovarajući identifikacijski broj uređaja. Varijabla id_uredaj vezana je naredbom „bind_param“ kako bi ona popunila znak „?“ u upitu. Upit je poslan te je rezultat spremljen u varijablu „result“. Također je pozvana i funkcija rezultat koja je kasnije objašnjena.

```
if( $upitKod==1){
      $stmt = $conn->prepare("SELECT * FROM mjerenja WHERE id=(SELECT
      max(id) FROM mjerenja) and id_uredaja=?");
      $stmt->bind_param("s",$id_uredaj);
      $stmt->execute();
      $result=$stmt->get_result();
      rezultat($result);
      }
```

Sljedeći upit služi za dohvaćanje stanja brojila na početku trenutnog mjeseca. Odabran je redak koji ima najmanji id, odgovarajući id uređaja te je zabilježen u trenutnom mjesecu.

```
$mjesec=date('m');
$stmt = $conn->prepare(SELECT * FROM mjerenja WHERE id=(SELECT min(id) FROM
mjerenja where datum like CONCAT('%-', ?, '-%')) and id_uredaja=?);
$stmt->bind_param("ss",$mjesec,$id_uredaj);
$stmt->execute();
$result=$stmt->get_result();
rezultat($result);
```

Sljedeći upit služi za dohvaćanje stanja brojila na početku trenutnog tjedna. Odabran je redak koji ima najmanji id, odgovarajući id uređaja te je zabilježen u trenutnom tjednu. Prethodna dva upita u kombinaciji s prvim omogućuju izračun i prikaz podataka o tjednoj i mjesečnoj potrošnji.

```
$tjedan=date("W", strtotime($datum));
$stmt = $conn->prepare("SELECT * FROM mjerenja WHERE id=(SELECT min(id)
FROM mjerenja where WEEK(mjerenja.datum)>=?) and id_uredaja=?");

$stmt->bind_param("ss",$tjedan,$id_uredaj);
$stmt->execute();
$result=$stmt->get_result();
rezultat($result);
```

Posljednji upit zadužen je za dohvaćanje podataka koji popunjavaju graf potrošnje po mjesecima. Broj upita koje šaljemo ovisi tome koliko mjeseci je uređaj u upotrebi. Za svaki mjesec poslano je dva upita. Prvi upit dohvaća stanje brojila na početku mjeseca dok drugi upit dohvaća stanje na kraju mjeseca. Koristeći varijablu startS koja je kasnije poslana funkciji rezultatSpecial saznajem znao jesu li dohvaćeni podaci s početka ili kraja mjeseca.

```
$godina=date("Y", strtotime($datum));
$trenutniMjesec = date('m');
$trenutniMjesecBrojac = date('n');

for($i=1;$i<=$trenutniMjesecBrojac;$i++){
      if($i<10){
            $mjesecZaUpit="0$i";
      }else{ $mjesecZaUpit=$i;}

      $zaUpit=$godina."-".$mjesecZaUpit."-%";
      $stmt = $conn->prepare("SELECT * FROM mjerenja WHERE id=(SELECT
      min(id) FROM mjerenja where datum LIKE ?) and id_uredaja=?");
      $stmt->bind_param("ss",$zaUpit,$id_uredaj);
      $stmt->execute();
      $result=$stmt->get_result();

      $startS=true;
      rezultatSpecial($result,$startS);

      $stmt = $conn->prepare("SELECT * FROM mjerenja WHERE id=(SELECT
      max(id) FROM mjerenja where datum LIKE ?) and id_uredaja=?");
      $stmt->bind_param("ss",$zaUpit,$id_uredaj);
      $stmt->execute();
      $result=$stmt->get_result();
      $startS=false;
      rezultatSpecial($result,$startS);
}
```

Funkcija rezultat koristi se za ispis rezultata odnosno odgovora baze na poslan upit. Dohvaća sve retke koji odgovaraju poslanom upitu te ispisuje vrijednosti koje su zabilježene. Kod ispisa korišten je znak „x“ koji kasnije u mobilnoj aplikaciji pomaže s razdvajanjem podataka.

```
function rezultat($rezultat){
      if ($rezultat->num_rows > 0) {
            while($row = $rezultat->fetch_assoc()) {
                  echo "id: x" . $row["id"]. " - vrijednost: x" .
                  $row["vrijednost"]. "x datum: x" . $row["datum"];
            }
      }else { echo "0 rezultata";}
}
```

Funkcija rezultatSpecial ima istu namjenu kao i funkcija rezultat. Razlika je u tome da prima dva parametara, jedan je rezultat a drugi je varijabla StartS tipa bool. Pomoću nje program određuje je li dohvaćeni podatak mjerenje s početka ili kraj mjeseca. To je potrebno kako bi u aplikaciji mogli točno prikazati grafove za pregled potrošnje.

Na kraju programa prekida se veza s bazom podataka.

```
$conn->close();
```

5.1.2. Obrada primljene fotografije

Svrha ove datoteke je primanje i obrada fotografije koju šalje uređaj. Fotografija se šalje HTTPS POST zahtjevom te se privremeno sprema u direktorij na serveru. Nakon toga se iz naziva datoteke izvlači ID uređaja koji je poslalo fotografiju. Poziva se funkcija koja kao parametar prima fotografiju spremljenu u direktoriju. Nakon toga se izvršava naredba koja preko naredbenog retka pokreće Python program te kao parametar prima fotografiju.

```
function ocr($image) {
$tekst=shell_exec('"C:\\Users\\korisnik\\Desktop\\analiza.py"
"'.$image.'"');
return $tekst;
}
```

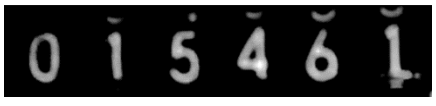
Koristeći *Python* i *OpenCV* fotografija se obrađuje kako bi se lakše očitali podaci koji se kasnije spremaju u bazu podataka. Za lociranje samog brojčanika na brojilu detektira se karakteristična bijelu liniju na brojilu (označeno crvenom linijom na slici ispod) koristeći Houghovu transformaciju. Nakon toga dodaju se još dvije linije (označene zelenom bojom) koje su pomaknute prema dolje po y osi za određeni broj piksela. One služe za izdvajanje samog brojčanika s fotografije. Prema zelenim linijama fotografija se sužava te je pretvorena u crno-bijelu varijantu zbog lakše detekcije brojeva. Na tu obrađenu sliku primijenjeni su filteri te je prilagođena svjetlina i kontrast što uvelike povećava točnost detekcije brojeva.

Koristeći *tesseract* detektiraju se brojeve sa slike. Poziva se naredba navedena ispod. U postavkama naredbe dozvoljena je detekcija brojeva od 0-9. Također je korišten mode rada „PSM 7“ (eng. *Page Segmentation Mode*) što programu govori da tretira sliku kao jednu liniju teksta.

```
text = pytesseract.image_to_string(crop,config='--psm 7 -c
tessedit_char_whitelist=0123456789')
```

Slika 10: Detekcija elemenata na brojilu (Autorski rad)

Slika 12: Izdvojeni brojčanik (Autorski rad)



Slika 11: Izdvojeni brojčanik s efektima i obradom (Autorski rad)

6. Android aplikacija

Za pregled potrošnje osmišljena je android aplikacija. Pomoću nje korisnik ima uvid u mjesečnu, tjednu te ukupnu potrošnju. Također ima mogućnost postavljanja mjesečnog cilja potrošnje te ima uvid u grafičku statistiku potrošnje po mjesecima. Aplikacija je napravljena koristeći *Android Studio* [7].

6.1.1. Aktivnost zaslona učitavanja aplikacije

Ova aktivnost se prva pokreće kod otvaranja aplikacije. Korisniku je prikazan zaslon učitavanja dok se u pozadini dohvaćaju svi podaci potrebni za rad aplikacije. Prvi korak kod otvaranja aplikacije je provjera internetske veze koja je neophodna za dohvaćanje podataka sa servera. U varijablu povezan tipa boolean spremljen je rezultat funkcije ProvjeriVezu().

```
boolean povezan=ProvjeriVezu();

boolean ProvjeriVezu(){
    ConnectivityManager connectivityManager=(ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo=connectivityManager.getActiveNetworkInfo();
        if(networkInfo!=null){
            if(networkInfo.isConnected()){
                return true;
            }else return false;
        }else return false;
    }
```

Ako je varijabla povezan istinita, znamo da korisnik ima vezu s internetom te pozivamo funkciju PosaljiHTTPzahtjev(). U suprotnom, koristeći SharedPreferences dohvaćamo zadnje zabilježene podatke te pozivamo funkciju OtvoriPocetniZaslon kojoj prosljeđujemo zadnje uspješno dohvaćene podatke.

```
if(povezan==true){
    PosaljiHTTPzahtjev();
}else{

    SharedPreferences offline = this.getSharedPreferences("mypref", 0);
    SharedPreferences.Editor editor2 = offline.edit();
    offline = getSharedPreferences("mypref", MODE_PRIVATE);
    String offlineUkupno = offline.getString("OfflineUkupno", "");
    String offlineMjesecno = offline.getString("OfflineMjesecno", "");
    String offlineTjedno = offline.getString("OfflineTjedno", "");
```

```
    OtvoriPocetniZaslon(offlineUkupno,offlineMjesecno,offlineTjedno);
}
```

Ako korisnik ima vezu s internetom, pozvana je funkcija PosaljiHTTPzahtjev(). Najprije su dohvaćeni i spremljeni podaci potrebni za slanje zahtjeva. Zamišljeno je da svaki fizički uređaj za mjerenje brojila ima svoj jedinstveni QR kod. Nakon što ga korisnik skenira on je spremljen u aplikaciji te služi za dohvaćanje točnih podataka iz baze. Spremljen je tekst QR koda u varijablu id_uredaj te adresa lokalnog servera kojem će upit biti poslan.

```
public void PosaljiHTTPzahtjev(){

SharedPreferences pref2 = this.getSharedPreferences("kod", 0);
SharedPreferences.Editor editor2 = pref2.edit();
String QRkod = pref2.getString("kod", "");
id_uredaj=QRkod;

String url = "https://192.168.1.9/sqlUpitiNovo.php";
RequestQueue MyRequestQueue = Volley.newRequestQueue(this);

//ostatak koda funkcije u nastavku
}
```

U nastavku je formiran novi zahtjev za slanje serveru. Taj dio se sastoji od tri funkcije. Prva se zove onResponse te je zadužena za slušanje odgovora koji je poslao server, u našem slučaju su to zapisi iz baze podataka. Također slušamo moguće greške u slanju ili primanju upita te ako postoje ispisujemo ih u konzolu. Zadnji dio je priprema podataka za slanje, spremljen je kod upita te id uređaja.

```
StringRequest NoviZahtjev1 = new StringRequest(Request.Method.POST, url,
new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        rezultat=response;
    }
},new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.d("ERROR", error.toString() );
    }
}) {
    protected Map<String, String> getParams() {
        Map<String, String> Podaci = new HashMap<String, String>();
        Podaci.put("upitKod", "1");
        Podaci.put("id_uredaj", id_uredaj);
        return Podaci;
    }
};
```

Nakon formiranja zahtjeva dodan je u red za slanje. Isti postupak je napravljen i za ostale zahtjeve koji se šalju serveru.

```
MyRequestQueue.add(NoviZahtjev1);
```

Nakon uspješnog dohvaćanja podataka potrebnih za rad aplikacije pozvana je funkcija OtvoriPocetniZaslon(). Stvorena je nova aktivnost koristeći intent sa proslijeđenim ranije dohvaćenim podacima. Nakon je pokrenuta aktivnost.

```
private void OtvoriPocetniZaslon(String rezultat, String rezultat2, String
rezultat3){

final Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        Intent i = new Intent(LoadingScreen.this, MainActivity.class);
        i.putExtra("ukupnoStanje", rezultat);
        i.putExtra("mjesecnoStanje", rezultat2);
        i.putExtra("tjednoStanje", rezultat3);
        startActivity(i);
    }
}, 0);
```

6.1.2. Glavna aktivnost pregleda potrošnje

Kod pokretanja aktivnosti pozivaju se tri funkcije. Prva je postavljanjeMjerila() te je zadužena za obradu podataka dohvaćenih sa servera te popunjavanja grafičke elemente s obrađenim podacima. Sljedeća funkcije je spremiNovePodatke() te je zadužena za spremanje novo dohvaćenih podataka koji se koriste u slučaju da korisnik kod sljedećeg pokretanja aplikacije nema internetsku vezu. Posljednja funkcija postavlja podatke vezane uz mjesečni cilj potrošnje korisnika. U nastavku će ukratko biti opisana svrha svake od njih.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    postavljanjeMjerila();
    spremiNovePodatke();
    postaviCilj();
}
```

U nastavku je primjer koda koji se izvršava kod poziva funkcije postavljanjeMjerila(). Varijabla ukupnoStanje sadrži odgovor servera na ranije poslan HTTPS upit. Odgovor je u formatu „id: x200 - vrijednost: x102619x datum: x2023-07-27 23:00:00“. Znak „x“ korišten je kao pomoć kod razdvajanja podataka. Koristeći ugrađenu funkciju „split“ odgovori servera su rastavljeni na lakše razumljive podatke. Iz tih podataka izvlači se zapisano stanje brojila (vrijednost) te datum i vrijeme bilježenja stanja. Popunjavaju se prije pripremljeni tekstualni prozori s dobivenim podacima.

```
if (ukupnoStanje != "" && ukupnoStanje.contains("x")) {
    spStgUkupno = ukupnoStanje.split("x");
    if (spStgUkupno[3] != null) {
        textViewUkupno.setText(spStgUkupno[2]);
        TextView textViewZadnjeMjerenje = (TextView)
findViewById(R.id.textViewZadnjeMjerenje);
        textViewZadnjeMjerenje.setText(spStgUkupno[4]);
        String[] spStgDatumVrijeme=new String[0];
        spStgDatumVrijeme=spStgUkupno[4].split(" ");

        String[] spStgDatum=new String[0];
        spStgDatum=spStgDatumVrijeme[0].split("-");

        String spStgVrijeme=spStgDatumVrijeme[1];

        String finalDatumVrijeme="Zadnje mjerenje: "+ spStgDatum[2]+".
"+spStgDatum[1]+". "+spStgVrijeme;
        textViewZadnjeMjerenje.setText(finalDatumVrijeme);
    }
}
```

Funkcija spremiNovePodatke nakon uspješnog dohvaćanja podataka sa servera sprema iste u lokalnu memoriju korisnika aplikacije. Koriste se „shared preferences“ zbog jednostavnog i brzog spremanja i dohvaćanja podataka.

```
private void spremiNovePodatke(){
    SharedPreferences pref = this.getSharedPreferences("mypref", 0);
    SharedPreferences.Editor editor = pref.edit();

    editor.putString("OfflineUkupno", ukupnoStanje).commit();
    editor.putString("OfflineMjesecno", mjesecnoStanje).commit();
    editor.putString("OfflineTjedno", tjednoStanje).commit();

}
```

Za postavljanje cilja potrošnje koriste se podaci koje je korisnik unio u postavkama aplikacije. To su podaci o cijeni energenata te o željenom cilju potrošnje na mjesečnoj bazi. Izračunava se iznos u eurima te se popunjavaju tekstualni prozor s odgovarajućim podacima. Korisnik aplikacije ima uvid u iznos potrošene energije u eurima i mjernoj jedinici te vidi koliko energije mu je još dostupno za korištenje prije prekoračenja mjesečnog cilja.

```
private void IzracunCijene(float cijenaKwh, String potrosnjaS,TextView
postaviText){
    int potrosnja= Integer.parseInt(potrosnjaS);

    float izracun =potrosnja * cijenaKwh;
    NumberFormat formatter = NumberFormat.getNumberInstance();
    String output ="Potrošnja: " + formatter.format(izracun)+ "€";
    postaviText.setText(output);
}
```

Slika 13: Glavna aktivost u aplikaciji
(Autorski rad)

6.1.3. Postavke aplikacije

Ova aktivnost se koristi za postavljanje postavka aplikacije. Korisnik unosi podatke koji su potrebni za rad aplikacije. Neki od tih podakta su QR kod uređaja, cijena energenata, vrsta energenata te postavljanje cilja potrošnje.

Funkcija spremiCilj aktivira se pritiskom na gumb „Spremi cilj“. Dohvaća se tekst koji je korisnik unio te se sprema u varijablu „cilj“ tipa String. Taj podatak se pohranjuje u lokalnu memoriju uređaja kao dijeljene preference (eng. *shared preferences*) te korisnika obavještavamo o novo postavljenom cilju koristeći Toast obavijest.

```
public void spremiCilj(View view){
    EditText editTextCilj= (EditText) findViewById(R.id.editTextCilj);
    String cilj=editTextCilj.getText().toString();

    SharedPreferences pref = this.getSharedPreferences("mypref", 0);
    SharedPreferences.Editor editor = pref.edit();
    editor.putString("cilj", cilj).commit();

    String zaToast = "Spremljeno! Novi cilj je:  " + cilj + " €.";
    Toast toast = Toast.makeText(this, zaToast, duration);
    toast.show();

}
```

U postavkama korisnik ima mogućnost odabira energenata kojeg želi nadzirati. Odabir se vrši pritiskom na neki od ponuđenih radio buttona. To je potrebno kako u aplikaciji bile prikazane ispravne mjerne jedinice te odgovarajuće grafičke podloge. Nakon pritiska na neki od ponuđenih energenata korisnikov odabir se sprema u lokalnu memoriju uređaja kako bi se kod ponovnog otvaranja aplikacije automatski odabrao željeni energent.

```
private void popuniRadio(){
    SharedPreferences pref = this.getSharedPreferences("mypref", 0);
    String spremljeniIznos = pref.getString("checkKod", "");
    RadioGroup rGroup = (RadioGroup)findViewById(R.id.radioGroup1);
    int proba=Integer.parseInt(spremljeniIznos);
    rGroup.check(proba);
}
```

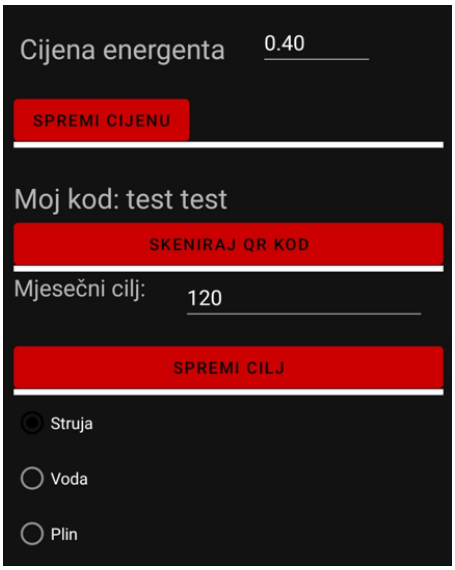
Klikom na gumb „Skeniraj“ otvara nova aktivnost koja služi za skeniranje QR koda. Kao skener koristi se *ZXing decoder* koji je pouzdan i jednostavan za implementaciju. Nakon pritiska na

gumb „Skeniraj“ otvara se nova aktivnost za skeniranje. Nakon uspješnog skeniranja podaci izvučeni iz koda spremaju se u lokalnu memoriju uređaja.

```
public void skeniraj(View view) {
    Intent myIntent = new Intent(Postavke.this, SkenirajKod.class);
    Postavke.this.startActivity(myIntent);
}


private void spremiKod(String kod){
    SharedPreferences pref = this.getSharedPreferences("kod", 0);
    SharedPreferences.Editor editor = pref.edit();
    editor.putString("kod", kod).commit();
}
```

Pritiskom na gumb „Spremi“ sprema se cijena po jedinici energenta koji je korisnik odabrao. Obavještavamo korisnika o izmjeni.

```
public void spremi(View view) {
    EditText uneseniIznos=(EditText)findViewById(R.id.cijenaStruje);
    SharedPreferences pref = this.getSharedPreferences("mypref", 0);
    SharedPreferences.Editor editor = pref.edit();

    cijenaStrujeString = uneseniIznos.getText().toString();

    editor.putString("cijena_struje", cijenaStrujeString).commit();
    String zaToast = "Spremljeno! Nova zadana cijena energenta je "
+ cijenaStrujeString + " €.";
    Toast toast = Toast.makeText(this, zaToast, duration);
    toast.show();

}
```



Slika 14: Postavke aplikacije (Autorski rad)

6.1.4. Grafički pregled potrošnje

Cilj aktivnosti je korisniku na pregledan način prikazati prethodno zabilježene podatke o potrošnji energije. Podaci se prikazuju u obliku stupčastog grafa na mjesečnoj bazi.

Aktivnost započinje slanjem HTTPS zahtjeva serveru. Nakon toga server šalje upit bazi podataka te ona vraća odgovarajuće podatke mobilnoj aplikaciji. Poziva se funkcija ObradaPodataka() koja kao argument prima odgovor baze podataka u varijabli „rezultat“ tipa string. Vraćeni podaci se sastoje od stanja brojila na početku i kraju mjeseca. Razdvojeni su znakom „/“ kako bi ih lakše programski obradio. Za svaki set podataka izračunava se razlika stanja brojila s kraja i početka mjeseca te se stvara novi stupac s tom vrijednosti. Nakon što su obrađeni svi setovi podataka (mjeseci) poziva se funkcija koja stvara listu s potrebnim podacima za prikaz te se poziva funkcija za stvaranje samog grafa.

```
private void ObradaPodataka(String rezultat){


    String[] spStgRezultat = rezultat.split("/");
    for(int i=0;i<spStgRezultat.length;i++){
        String[] spStgRezultatFinal = spStgRezultat[i].split("x");

        int rezulat=0;
        if(spStgRezultatFinal.length>1) {
            rezulat = Integer.parseInt(spStgRezultatFinal[1]) -
Integer.parseInt(spStgRezultatFinal[0]);
        }
        mjesecnaPotrosnja.add(new BarEntry(i, rezulat));

    }
    getmjesecnaPotrosnja();
    StvoriGraf();
}
```

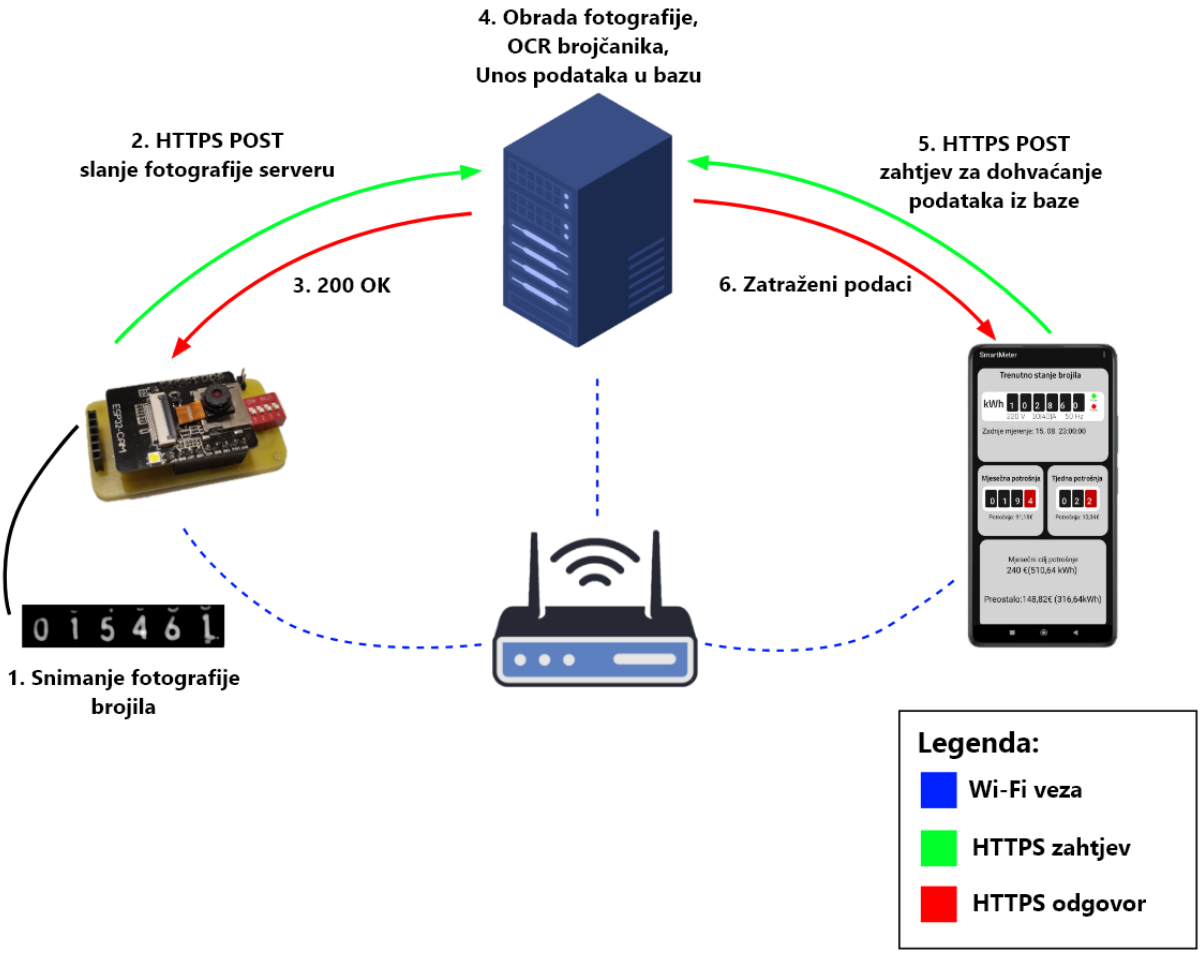
Za stvaranje grafova koristi se gotova biblioteka *MPAndroidChart*. Postavljaju se nazivi stupaca koji su u ovom slučaju mjeseci u godini. Dohvaćaju se podaci o mjesečnoj potrošnji te se stvaraju setovi podataka koji čije stupce na grafu.

```
private void StvoriGraf(){
    ArrayList<IBarDataSet> dataSets = new ArrayList<>();
    List<String> xAxisValues = new ArrayList<>(Arrays.asList("Siječanj",
"Veljača", "Ožujak"…));
    List<BarEntry> mjesecnaPotrosnja = getmjesecnaPotrosnja();


    dataSets = new ArrayList<>();
    BarDataSet set1;

    set1 = new BarDataSet(mjesecnaPotrosnja, "Potrosnja");
    dataSets.add(set1);
…}
```

Slika 15: Prikaz potrošnje koristeći stupičaste grafove (Autorski rad)

7. Pregled sustava

U ovom poglavlju pregledat ćemo interakciju između komponenata koje čine sustav. Ilustracija na slici ispod opisuje komponente te tok podata između njih (s lijeva na desno). Sustav se sastoji od tri bitnije komponente koje su spojene na Wi-Fi mrežu. Prva komponenta je sam uređaj koji ima zadatak snimiti fotografiju brojila te je nakon toga poslati na server putem HTTPS POST zahtjeva. Ako je slanje uspješno obavljeno, server vraća odgovor „200 OK“. Sljedeća komponenta je server, njegov je zadatak privremeno spremanje te obrada primljene fotografije. Nakon toga se vrši OCR nad fotografijom te se dobiveni podaci spremaju u bazu podataka. Treća komponenta je mobilni uređaj, odnosno aplikacija na uređaju. U trenutku kad korisnik pokrene aplikaciju, šalje se nekoliko HTTPS POST zahtjeva prema serveru (bazi podataka) te server aplikaciji odgovara s traženim podacima te su oni prikazani na uređaju korisnika.



Slika 16: Pregled sustava (Autorski rad)

8. Sigurnost

Sigurnost ima veliku važnost u svim IT projektima pa tako i u ovom. Svaki dio projekta se oslanja na očuvanje sigurnosti podataka te će u ovom poglavlju biti pojašnjena važnost i implementacija sigurnosnih rješenja kako bi cijeli sustav bio siguran [10].

8.1.1. HTTPS ili HTTP

Kao protokol za slanje podataka između web servera i mobilne aplikacije koristi se HTTPS. HTTP (Hypertext Transfer Protocol) i HTTPS (Hypertext Transfer Protocol Secure) su dva različita protokola kojima je glavna i najveća razlika razina sigurnosti koju pružaju. Kod HTTP protokola podaci se prenose u obliku običnog teksta te nisu šifrirani što znači da lako mogu biti iščitani prilikom prijenosa.

HTTPS protokol je sigurnija verzija HTTP-a koja koristi dodatni sloj sigurnosti pod nazivom SSL/TLS (Secure Sockets Layer/Transport Layer Security) kako bi šifrirala podatke prije slanja. Kada se koristi HTTPS, svi podaci koji se prenose između web preglednika i web poslužitelja su šifrirani, što znači da su zaštićeni od prisluškivanja i ne mogu se lako pročitati tijekom prijenosa.

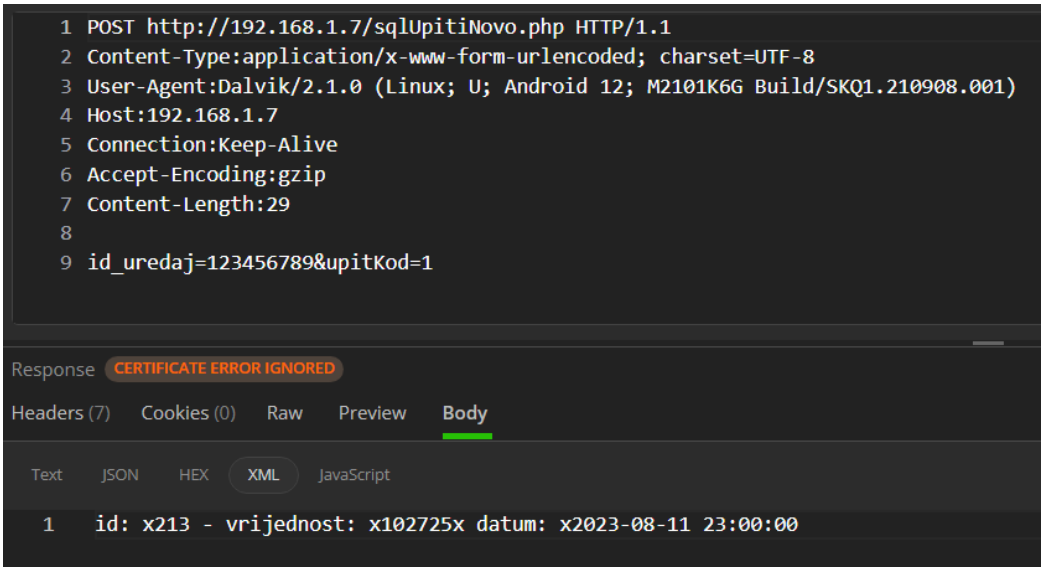
8.1.2. SQL injection

SQL umetanje (*eng. SQL injection*) je vrsta napada koja iskorištava sigurnosni propust u web aplikacijama. Omogućuje napadačima izvršavanje neovlaštenih SQL upita nad bazom podataka. Ovo je jedna od najčešćih i opasnijih vrsta napada na web aplikacije no može se spriječiti implementiranjem i pridržavanjem sigurnosnih pravila.

Kad web aplikacija nema implementaciju sigurnosnih pravila napadač može iskoristiti ovaj propust i umetnuti zlonamjerni SQL kod u polje za pretraživanje, obrazac za prijavu ili slanjem HTTP zahtjeva. U slučaju da aplikacija ne provjeri i ne obrađuje korisnički unos, SQL upit može biti iskorišten kako bi napadaču omogućio ispis, izmjenu ili brisanje podataka iz baze.

S obzirom na to da ova web aplikacija nema korisničko sučelje pa tako nema ni polja za unos podatka preko kojih bi mogli napraviti SQL umetanje odlučio sam to napraviti slanjem HTTP zahtjeva serveru. Ovaj primjer također prikazuje sigurnosni nedostatak kod korištenja HTTP-a umjesto HTTPS-a. Promet između klijenta i servera nije šifriran te smo u mogućnost uhvatiti i kopirati HTTP zahtjev. Za pregled prometa te slanje zahtjeva koristio sam program *Fiddler Everywhere*.

Pokrenuo sam snimanje mrežnog prometa te sam otvorio android aplikaciju za pregled potrošnje energenata. U programu za nadzor filtriram POST zahtjeve radi lakšeg pregleda. Nakon pronalaska zahtjeva koji šalje neke podatke prema serveru (i bazi podataka) kopiram taj zahtjev te ga kasnije mijenjam kako bi napravio napad na server.



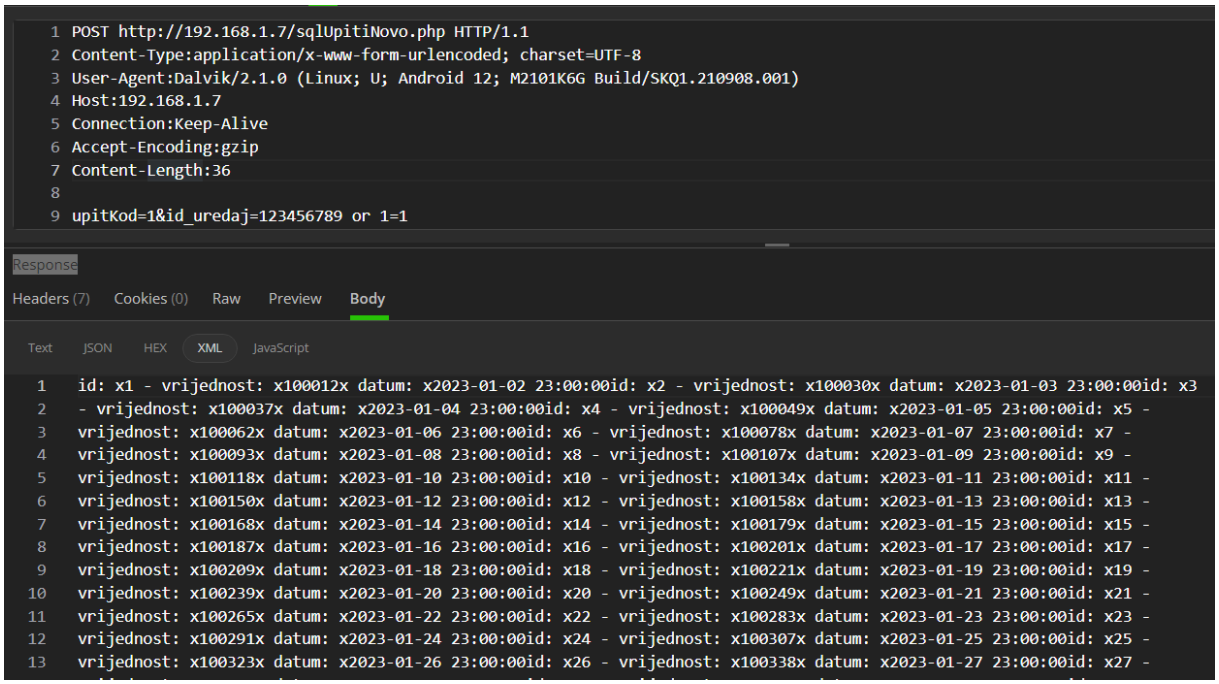
Slika 17: Uhvaćeni HTTPS zahtjev (Autorski rad)

Zahtjev sam promijenio na načina da sam podacima koji se šalju serveru nadodao „or 1=1“. Poslani podaci se na serveru spremaju u varijable koje su nužne za ispravno izvršavanje zadanog SQL upita. Dodavanjem izraza „or 1=1“ u varijablu te time i u SQL upit uspjeli smo iz tablice ispisati sve zapise. Baza izvršava upit kao i inače no vraća sve zapise jer je izraz „1=1“ uvijek istinit.

Ovaj napad je uspješno izvršen zbog sigurnosnog propusta na web serveru. Prilikom sastavljanja upita za bazu podataka nisu korišteni pripremljeni izrazi te se u varijablu koja popunjuje SQL upit dodaju svi podati dobiveni iz HTTP POST zahtjeva bez naknadne kontrole. U nastavku je kod sa web servera korišten prilikom izvedbe ovog napada.

```
$upit="SELECT * FROM mjerenja WHERE id=(SELECT max(id) FROM mjerenja) and
id_uredaja=".$id_uredaj."";
$result = $conn->query($upit);
rezultat($result);
```

Na slici ispod možemo vidjeti uređeni HTTP zahtjev te odgovor servera, odnosno sve zapise iz tablice.

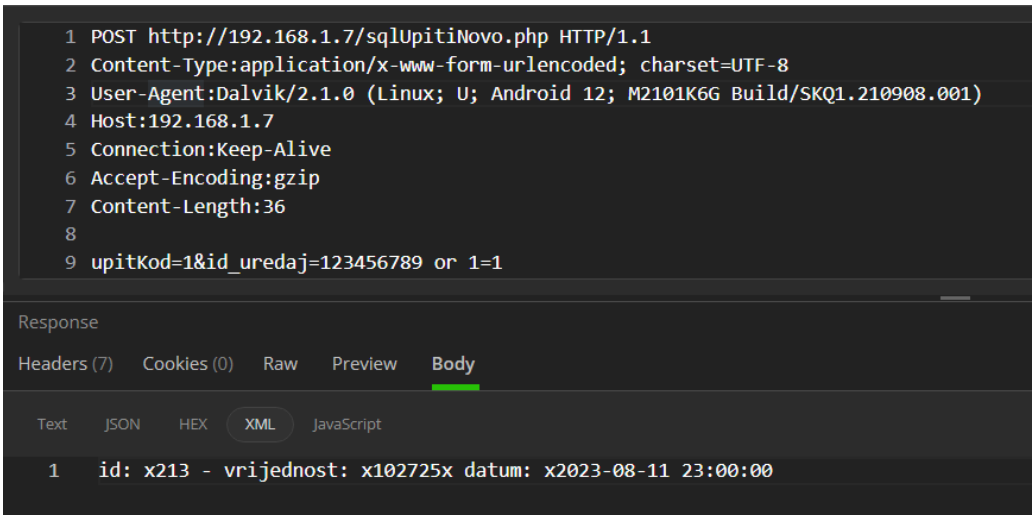


Slika 18: Uređeni HTTPS zahtjev te odgovor servera (Autorski rad)

Kako bi onemogućili napadačima izvođenje ovakvih napada, potrebno je promijeniti način na koji podatke primljene iz zahtjeva stavljamo u SQL upit. Jedan od načina je korištenje pripremljenih izraza. Koristeći ovu metodu varijable nisu izravno umetnute u SQL upit. Umjesto varijabli u upit postavljamo simbol „?“ koji služi kao privremena zamjena za varijablu. Nakon toga se SQL upit prekompajlira te nakon toga više nije moguće mijenjati logiku upita. Na kraju su na mjesta upitnika stavljeni podaci primljeni iz zahtjeva te se upit izvršava. Nakon ovih promjena kod izvedbe istog napada iz prethodnog primjera baza će doslovno tražiti id uređaja koji je jednak zapisu „1=1“. U nastavku je kod koji je korišten kod izvedbe ovog primjera.

```
$stmt = $conn->prepare("SELECT * FROM mjerenja WHERE id=(SELECT max(id)
FROM mjerenja) and id_uredaja=?");
$stmt->bind_param("s",$id_uredaj);
$stmt->execute();
$result=$stmt->get_result();
rezultat($result);
```

Možemo vidjeti da prilikom slanja istog zahtjeva kao i u gornjem primjeru kao odgovor ne dobivamo sve podatke iz tablice već samo zapis koji odgovara poslanom upitu.



Slika 19: Uređeni HTTPS zahtjev te odgovor servera nakon implementacije sigurnosnih mjera (Autorski rad)

9. Zaključak

Cilj ovog rada bio je razvoj sustava za nadzor potrošnje energenata s naglaskom na sigurnost. Razvoj takvog sustava zahtijeva multidisciplinarni pristup koji obuhvaća aspekte programiranja, elektronike, sigurnosti podataka te komunikacijskih protokola. Koristeći znanje stečeno tijekom školovanja te poznavanje i istraživanje potrebnih tehnologija uspio sam ostvariti ovaj projekt.

Sigurnosni aspekt ovog sustava je ključan jer podaci o potrošnji energenata i navikama kućanstva postaju osjetljivi. Implementirana sigurnosna rješenja zaštite podataka osiguravaju da privatnost korisnika ostane zaštićena. Sprečavanje neovlaštenog pristupa i zloupotrebe informacija treba biti glavni cilj kod razvoja takvog proizvoda [10].

U konačnici, IoT sustav za nadzor potrošnje energenata s fokusom na sigurnost predstavlja ključan korak prema energetski učinkovitijem kućanstvu. Kroz inteligentno upravljanje i svjesnost o potrošnji, ovi sustavi imaju potencijal da doprinesu očuvanju okoliša, financijskoj uštedi te zadovoljstvu korisnika.

10. Popis literature

[1] Priya Malta, Bhaskar Pant and Minit Arora, "*All you want to know about internet of things*", (ICCCA 2017), [Na internetu]
Dostupno: https://ieeexplore.ieee.org/abstract/document/8229999 [pristupano 18.6.2023.]

[2] M. Himansh and V. M. Manikandan, "A Statistical Study and Analysis to Identify the Importance of Open-source Software," (ICITIIT, 2022.), [Na internetu]
Dostupno: https://ieeexplore.ieee.org/document/9744176 [pristupano 25.6.2023.]

[3] KiCad Development Team. KiCad (verzija 7.0.7) (2023) [Na internetu]
Dostupno: https://www.kicad.org/ [pristupano 1.7.2023.]

[4] Arduino SRL. Arduino IDE (verzija 2.1.1) (2023) [Na internetu]
Dostupno: https://www.arduino.cc/en/software [pristupano 1.7.2023.]

[5] BitRock. XAMPP (verzija 8.0.28) (2023) [Na internetu]
Dostupno: https://www.apachefriends.org/download.html [pristupano 10.7.2023.]

[6] Apache Corporation. Apache (verzija 2.4) (2023) [Na internetu]
Dostupno: https://httpd.apache.org/ [pristupano 20.7.2023.]

[7] Oracle Corporation. MySQL (verzija 8.0) (2023) [Na internetu]
Dostupno: https://www.mysql.com/downloads/ [pristupano 5.8.2023.]

[8] PHP Foundation. PHP (verzija 8.2. 6) (2023) [Na internetu]
Dostupno: https://www.php.net/downloads.php [pristupano 5.8.2023.]

[9] Google. Android Studio. (verzija 2022.3.1 RC 1) (2023) [Na internetu]
Dostupno: https://developer.android.com/studio [pristupano 10.7.2023.]

[10] Sulabh Bhattarai and Yong Wang, "End-to-End Trust and Security for Internet of Things Applications", (IEEE computer society, 2018.) [Na internetu]
Dostupno: https://ieeexplore.ieee.org/document/8352081 [pristupano 25.6.2023.]

11. Popis slika

12. Popis tablica