Mjerenje stanja Li-Ion baterije

Šumanovac, Leon

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

SVEUČILIŠNI STUDIJ

MJERENJE STANJA LI-ION BATERIJE

Završni rad

Leon Šumanovac

Osijek, 2021.

SADRŽAJ

1. UVOD

Cilj ovog završnog rada je određivanje postotka napunjenosti baterije. Baterije su izvor električne energije koje se koriste u svakodnevnoj upotrebi i nije moguće zamisliti život bez njih. Većina njih same na sebi nemaju način pomoću kojeg se očita trenutno stanje napona ili postotka napunjenosti. Očitavanje trenutnog stanja napona na bateriji moguće je jedino korištenjem voltmetra, a postotak napunjenosti baterije nije moguće točno iščitati iz napona, jer baterija nema linearnu krivulju pražnjenja.

U ovom završnom radu će se prikazati 2 moguća načina očitavanja stanja baterije: metoda aproksimacije po naponu i metoda pomoću look-up tablice. Kako bi to bilo moguće, spojena je shema te su izrađeni programi koji će pomoću Arduina očitavati napon baterije u određenim vremenskim intervalima, zapisivati ih u EEPROM (*Electrically Erasable Programmable Read-Only Memory*) i Excel tablicu kako bi bilo moguće prikazati ovisnost napona baterije o konstantnom trošilu i odrediti preostalo vrijeme rada baterije.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je dizajnirati mikroupravljački sustav za mjerenje stanja Li-Ion baterije. Kada se mikroupravljač upali treba očitati stanje baterije i prikazati ga na ekranu računala.

2. PREGLED POSTOJEĆIH RJEŠENJA

U današnje vrijeme se za očitavanje napona baterije koriste BMS (*Battery management system*) i zaštitne sklopke koji osim za očitavanje, služe i za onemogućavane rada baterije izvan dopuštenih granica te ukoliko su više baterija zajedno spojene, balansiranje njihovog napona [1].

2.1. Battery Management System

BMS omogućuje praćenje pojedinih ćelija baterije u spoju više (najčešće serijski) spojenih baterija. Dopušta konstantno praćenje i sakupljanje informacija, te komunikaciju sa vanjskim sučeljem koje je korisniku jasno i pristupačno. Preko tog sučelja može pratiti svaku ćeliju zasebno ili cijeli spoj zajedno. Njegova funkcija je zaštititi bateriju, produljit joj životni vijek i zadržati ju unutar predviđenog radnog napona. Te funkcije su bitne za sigurnost, efikasnost i pouzdanost baterije i cijelog spoja.

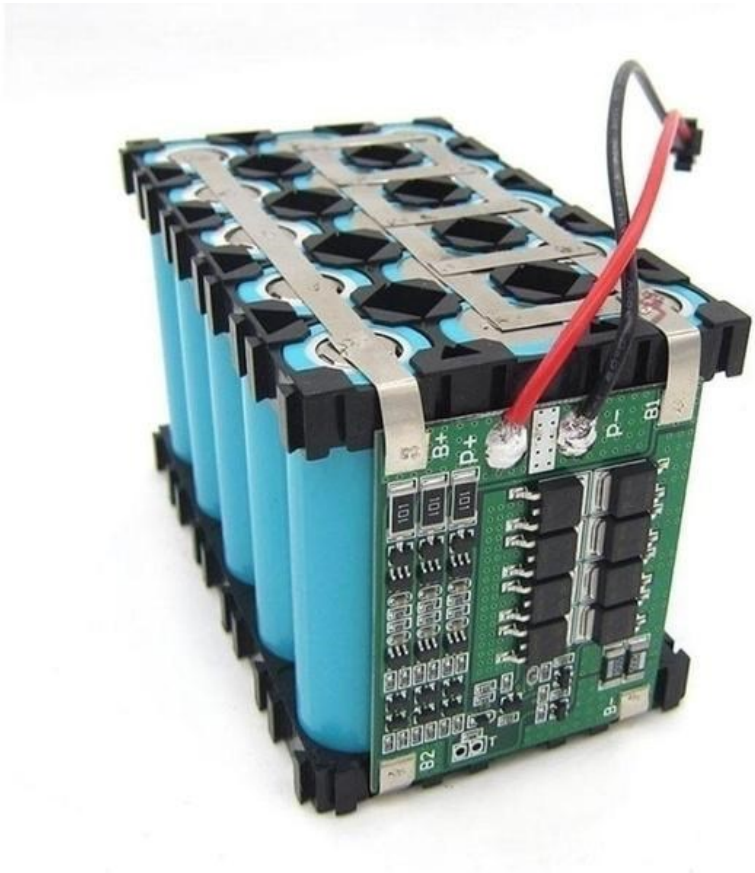
Pomoću njega se mogu očitavat razne vrijednosti, kao što su: napon, struja i temperatura. Pomoću tih vrijednosti je moguće procijeniti stanje baterije i prilagoditi operacije kako bi se baterija zaštitila. BMS omogućuje određivanje stanja napunjenosti i stanja zdravlja, što su važni pokazatelji uporabljivosti i sposobnosti baterije. Stanje napunjenosti je najjednostavnija mjera zdravlja baterije, što se mjeri i u ovom završnom radu, međutim stanje napunjenosti se može promijeniti s godinama jer se kapacitet baterija smanjuje s godinama pa je moguće očitati 100%, no tih 100% je manje nego kada je baterija bila nova [2]. Ukoliko su baterije spojene zajedno, bitno je držati sve baterije na jednakom postotku, što je omogućeno korištenjem BMS-a. Primjeri komercijalnih BMS-a: TLE9012AQU_TR_BMS2 i TLE9015QU_TRX_BRG tvrtke Infineon.

2.2. Zaštitne sklopke

Ukoliko nije potrebno mjerenje temperature, struje i napona, također se mogu koristiti i zaštitne sklopke. One štite baterije od prekomjernog punjenja, pražnjenja, prekomjerne struje i struje kratkog spoja. Mogu se naći za različite brojeve ćelija i razne zaštitne funkcije. Cilj im je zaštititi baterije, kao i BMS, i pružiti visoku točnost i nisku potrošnju baterija, te omogućavaju i spajanje više serijskih baterija za svoj rad. Jedan od nedostataka zaštitnih sklopki je što ne mogu u isto vrijeme i puniti i koristiti bateriju, jer sklopke u isto vrijeme mogu nadgledati samo struju kojom se baterija puni ili prazni. Zaštitne sklopke imaju i svoje prednosti, a neke od njih su: mogućnost samostalnog rada bez

potrebe za programiranjem i samostalno praćenje stanja baterije. Primjer komercijalne zaštitne sklopke: BQ294506 tvrtke Texas Instruments.

Slika 2.1. Primjer zaštitne sklopke korištene na više međusobno spojenih baterija.



3. IMPLEMENTACIJA SUSTAVA

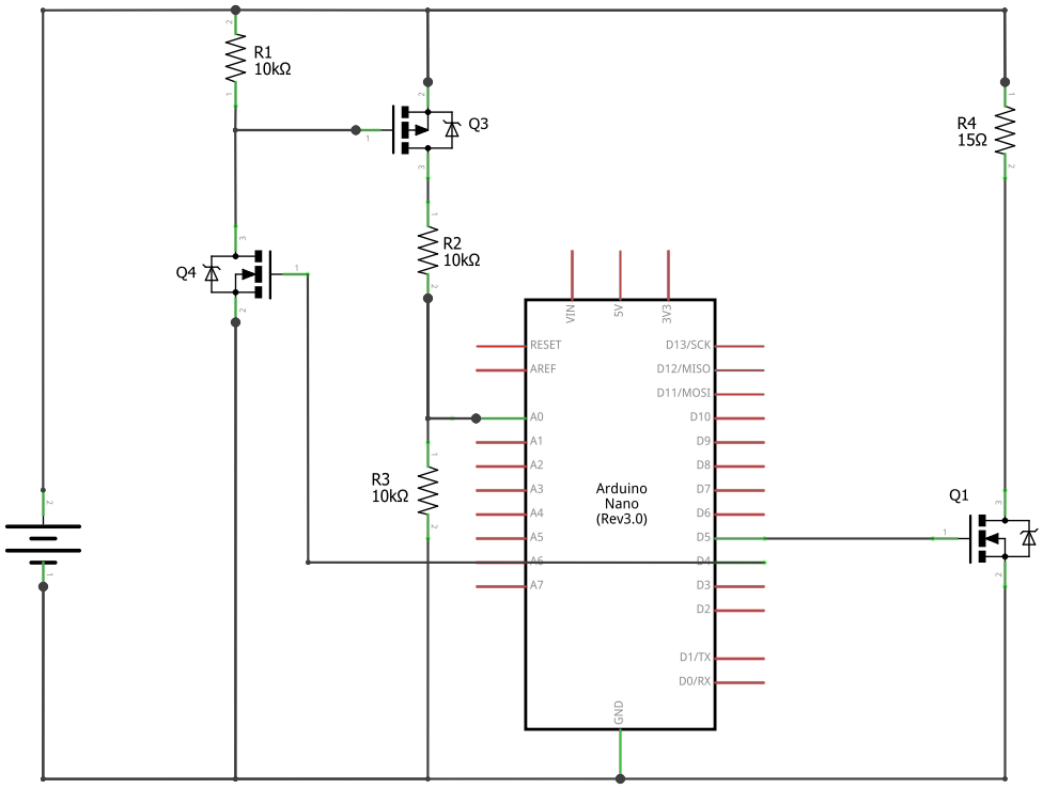
Napon pražnjenja baterije nije linearna karakteristika, nego je krivulja (Slika 3.1.), te će stoga biti potrebno napraviti spoj koji će prazniti bateriju uz mogućnost zaustavljanja pražnjenja te program koji će očitavati napon na bateriji.

Slika 3.1. Krivulja pražnjenja baterije.

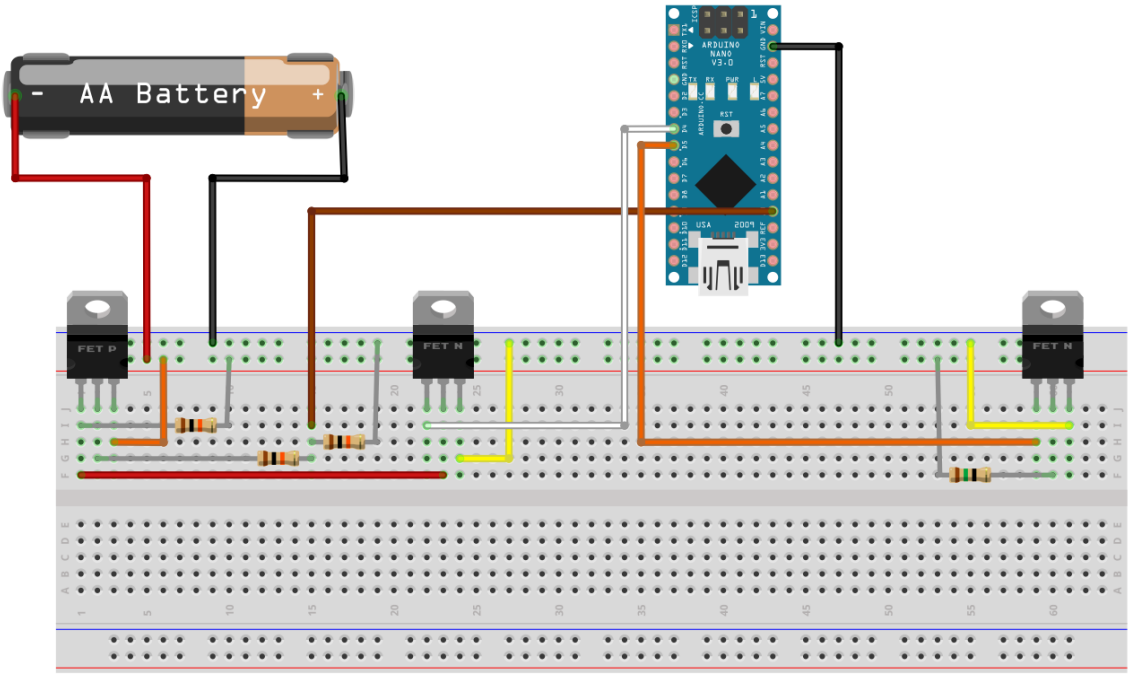
Kako bi bilo moguće odrediti napon iz baterije potrebno je napraviti elektroničku shemu, spojiti ju i napraviti program koji će pomoću Arduina moći odrediti trenutno stanje baterije i spremati podatke u Excel.

3.1. Elektronička shema

Prvi korak u realizaciji sustava je izrada elektroničke sheme. Kako bi ona bila pregledna, napravljena je u programu Fritzing koji ne samo da omogućuje izradu sheme, nego i pravi primjer spajanja na *breadboard*-u u drugom prozoru. Također, omogućuje i izradu PCB-a (engl. *Printed Circuit Board)*, no on nije korišten u ovom završnom radu. Shema pomoću koje se radio ovaj završni radi je prikazana na slici 3.2, a modificirani prikaz automatsko generiranog spoja na *breadboard*-u od Fritzing-a na slici 3.3.



Slika 3.2. Shematski prikaz spoja za očitavanje napona na bateriji.



Slika 3.3. Prikaz spoja na *breadboard*-u koristeći Fritzing.

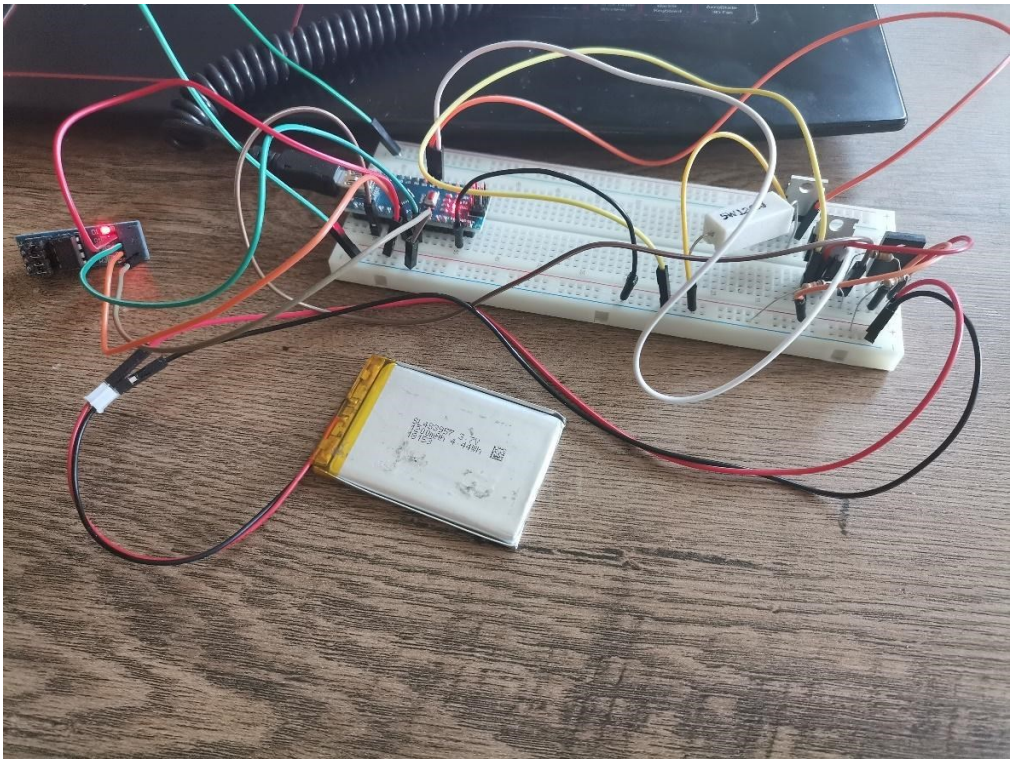
Shema sadrži djelitelj napona, tako da Arduino može podržavati i očitavati baterije većeg napona od 5V. Djelitelj napona čine otpornici R1, R2 i R3, te jedan P i jedan N kanalni MOSFET (engl. *Metal-Oxide-Semiconductor Field-Effect Transistor*). MOSFET-i služe kao sklopke kako bi Arduino mogao upravljati sa sklopom, što će biti opisano kasnije u radu.

Otpornik R4 označava trošilo, a N kanalni MOSFET još jednu sklopku pomoću koje Arduino može upravljati radom trošila. Bitno je da Arduino ima mogućnost zaustavljanja strujnog kruga trošila da se baterija ne isprazni ispod dozvoljene razine.

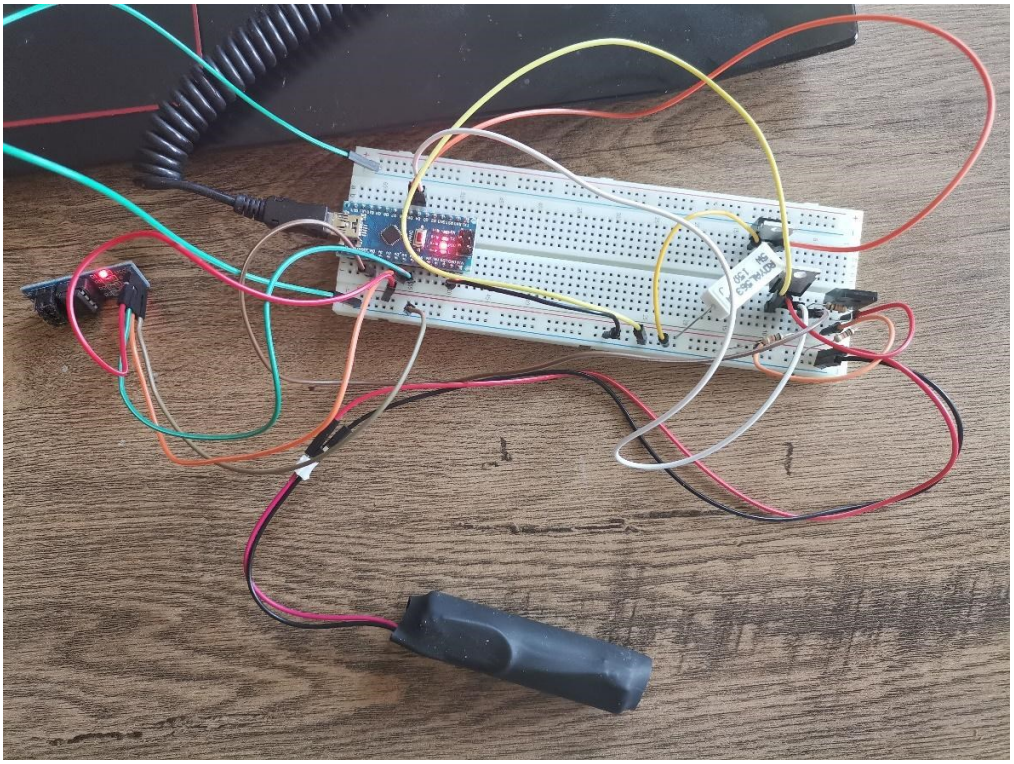
3.2. Rješenje sklopa

Nakon što se odredila elektronička shema po kojoj će se sklop spajati, potrebno je pravilno odabrati komponente. Za otpornike R1, R2 i R3 su odabrani žičani otpornici od 10k ohm-a. MOSFET-i koji su korišteni za spoj su P4NK80ZFP (P kanalni) i IRF540N (N kanalni). Otpornici koji su se koristili za trošilo su: 12 Ohm, 5 Watt i 15 ohm 5 Watt, a baterije koje su se koristile kao izvori i na kojima se mjerio napon su: 1200 mAh i 1300 mAh. Također, koristila se razvojna ploča Arduino Nano koji je bio spojen na računalo putem Mini USB kabla, te kako bi se sve spojilo u jednu cjelinu na *breadboardu* su se koristili kratkospojnici (engl. *jumper cable*). Da bi uspješno napravili look-up tablicu potrebno je i spojiti EEPROM koji ima 4 *pin*-a: GND (engl. *ground*), SCL (engl. *Serial Clock Line*), SDA (engl. *Serial Data Line*) i Vcc. SDA i SCL *pin*-ovi su spojeni na A4 i A5 *pinove* Arduina, jer oni označavaju SDA i SCL *pin*-ove Arduina.

Pošto je u ovom završnom radu potrebno prazniti baterije nekoliko puta, potrebno ih je i napuniti. Za punjenje baterija se koristio Croduino Nova 2, jer na sebi ima JST (engl. *Japan Solderless Terminal*) konektor, kao što imaju i baterije korištene u ovom radu.



Slika 3.4. Fizička realizacija sheme sa slike 3.1.



Slika 3.5. Fizička realizacija sheme sa slike 3.1.

3.3. Programsko rješenje

Nakon što je sklop uspješno spojen, napisani su programi koji će uspješno zapisivati i očitavati napon sa baterije. Kako bi mjerenja bila što preciznija, pomoću MOSFET-a Q4 sa slike 3.1. se upravlja sa očitavanjem napona baterije. MOSFET je otvorena sklopka sve dok ne treba izmjeriti napon, te kada je potrebno izmjeriti napon na bateriji, MOSFET dobije logičku "1" na *gate* i time propusti struju do analognog *pin*-a A0 i čim se napon baterije očita MOSFET dobiva logičku "0", te time opet prelazi u način rada otvorene sklopke. Time se postižu točnija mjerenja i štedi baterija jer očitavanje ne troši bateriju stalno nego samo u trenutku kada je potrebno obaviti očitavanje. Očitavanje i zapisivanje napona se obavlja svakih 10 sekundi, za što se koristi biblioteka TimerOne.h. Ona je isprogramirana kako bi svaku sekundu povećala brojač za jedan, te kada brojač dođe na deset, resetirala ga i pozvala funkciju *measure* koja je prikazana na slici 3.6.

```
void measure(){
  digitalWrite(4, HIGH);
  int analogValue = analogRead(A0);
  voltage = analogValue * (5.0 / 512.0);
  digitalWrite(4, LOW);
  Serial.println(voltage);
}
```

Slika 3.6. Funkcija *measure* za očitavanje i zapisivanje napona.

Poziv *Serial.println* omogućuje zapisivanje trenutnog napona na *Serial monitor*, ili ukoliko se postavi, zapisivanje u Excel-ovu tablicu pomoću dodatka za Excel – *Data Streamer*. Mjerenja su izvršena 3 puta za svaku kombinaciju baterija – otpornik, te je uzeta srednja vrijednost mjerenja i prikazana grafovima na slikama u nastavku.

Bitno je bilo implementirati i način za zaustavljanje pražnjenja baterije kako bi ostala u području rada. To je odrađeno u funkciji *loop* koja se stalno ponavlja dok Arduino radi. Ona stalno ispituje razinu napona izmjerenu u funkciji *measure* i uspoređuje ju sa minimalnom radnom vrijednosti napona, te ukoliko je ona manja ili jednaka, zaustavlja pražnjenje baterije i stavlja Arduino u *sleep*. U programu je definirana minimalna radna vrijednost napona od 2.55V. Funkcije za ispisivanje stanja baterije će biti naknadno objašnjene u poglavljima metoda u kojima se koriste.

3.4. Metoda aproksimacije po naponu

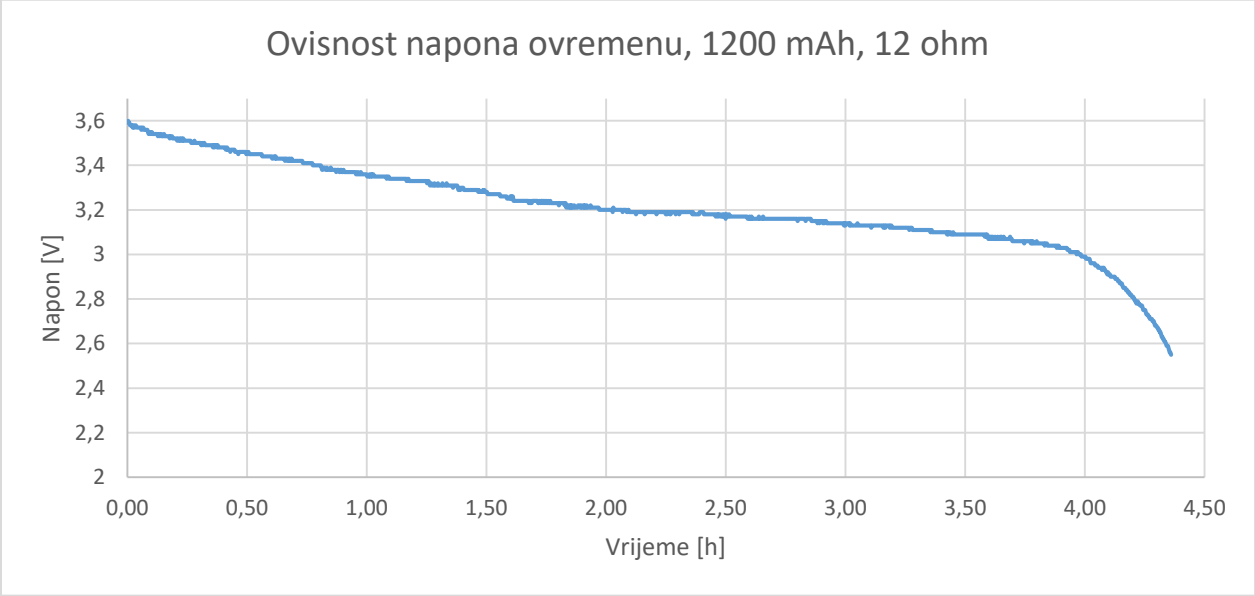
Pomoću metode aproksimacije po naponu se dobiva ista krivulja kao i krivulja pražnjenja baterije. Slike 3.7. i 3.8. prikazuju ovisnost napona i postotka napunjenosti baterije o vremenu koristeći bateriju od 1200 mAh pri korištenom trošilu od 12 ohm-a. Pošto obije baterije imaju jednaki maksimalni napon (U_{max}) i minimalni napon (U_{min}), pomoću formule 3-1 se računa vrijednost napona koja označava 1% napunjenosti baterije.

$$U_p = \frac{U_{max} - U_{min}}{100} \tag{3-1}$$

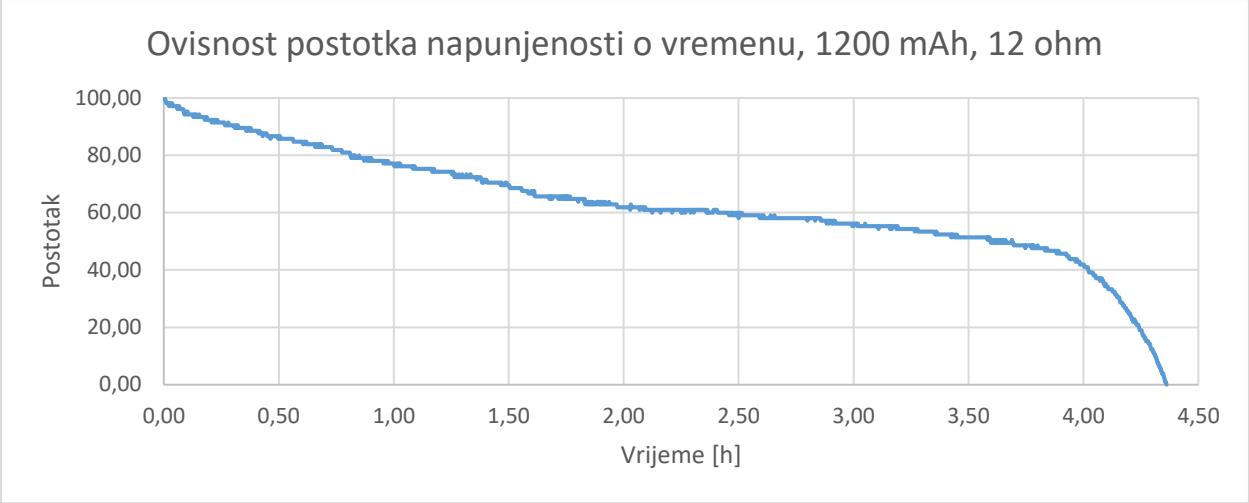
$$U_p = \frac{3.6 - 2.55}{100} = 0.0105$$

Nakon izračunavanja napona od 1% napunjenosti baterije, formulom 2-2 računamo trenutni postotak napunjenosti baterije.

$$\frac{U_{trenutni} - U_{min}}{U_p} \tag{3-2}$$

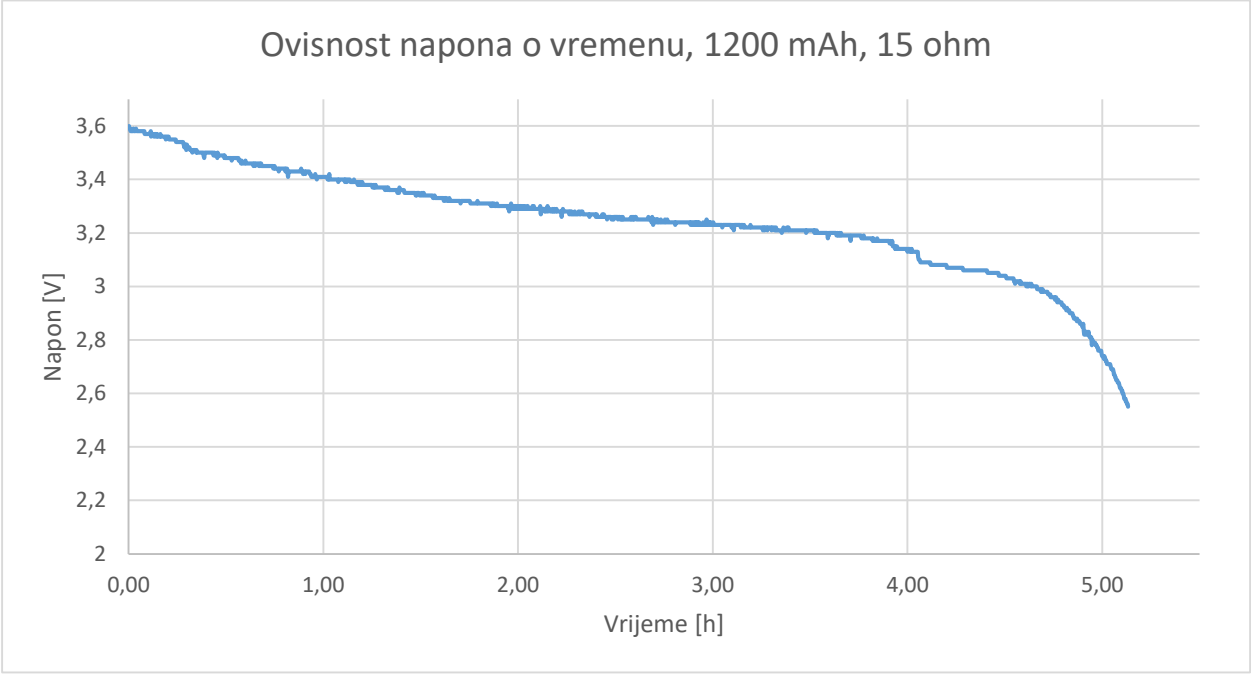


Slika 3.7. Ovisnost napona o vremenu kod baterije 1200mAh, koristeći otpornik od 12 ohm-a.

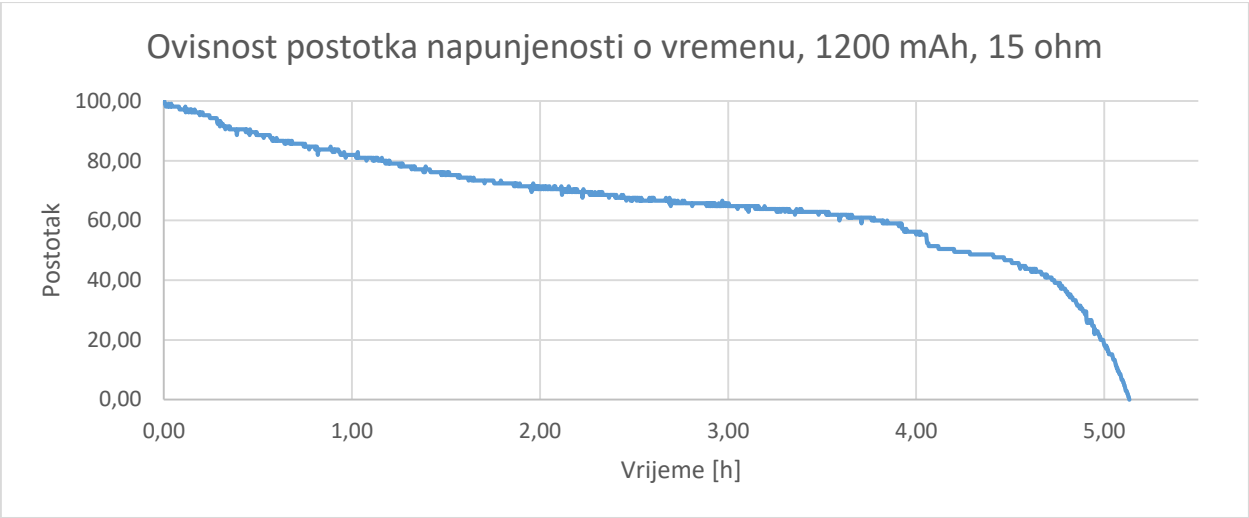


Slika 3.8. Ovisnost postotka napunjenost o vremenu kod baterije 1200 mAh, koristeći otpornik od 12 ohm-a.

Iz grafova na slikama 3.7. i 3.8. je moguće primijetiti kako grafovi imaju slične izglede, što je i bilo za očekivat pošto je postotak isključivo ovisan o naponu baterije. Također iz tih grafova se zaključuje da nije moguće pravilno iščitati napon baterije jer pri konstantnom trošilu ne gubi linearno napon, tj. prva 4 sata rada izgubi 60% napona, dok u zadnjih 20 minuta izgubi 40%.

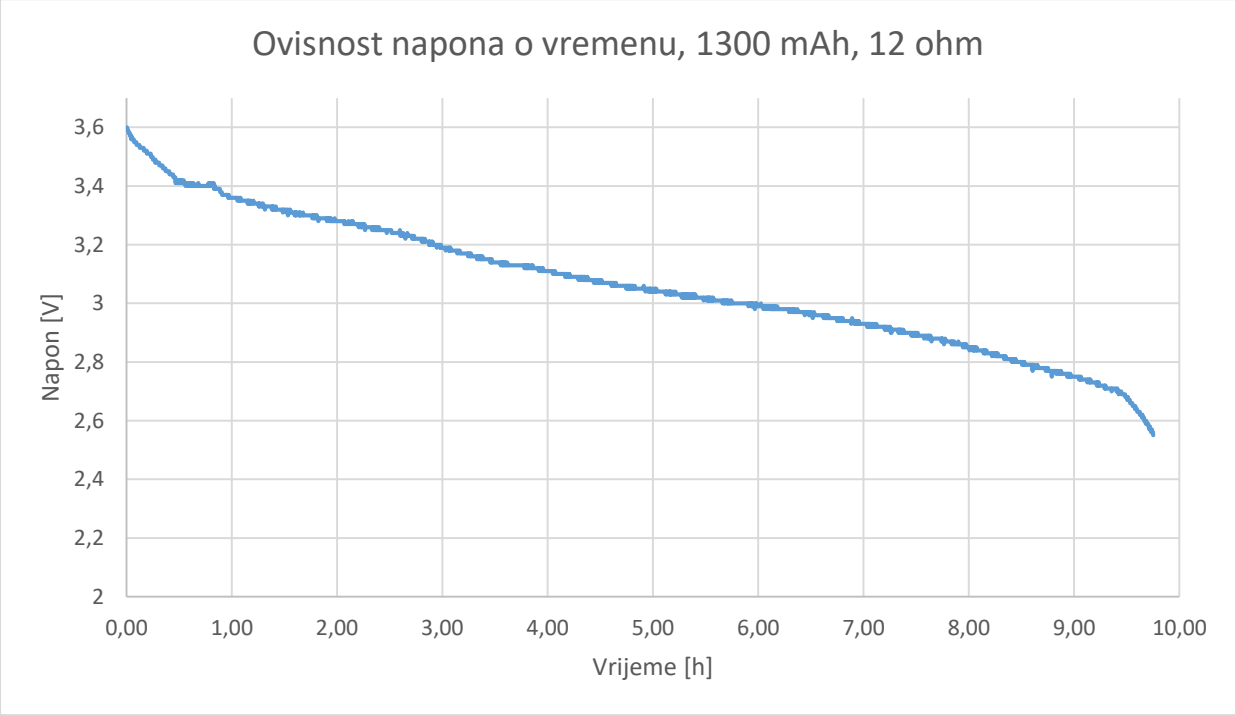


Slika 3.9. Ovisnost napona o vremenu kod baterije 1200 mAh, koristeći otpornik od 15 ohm-a.

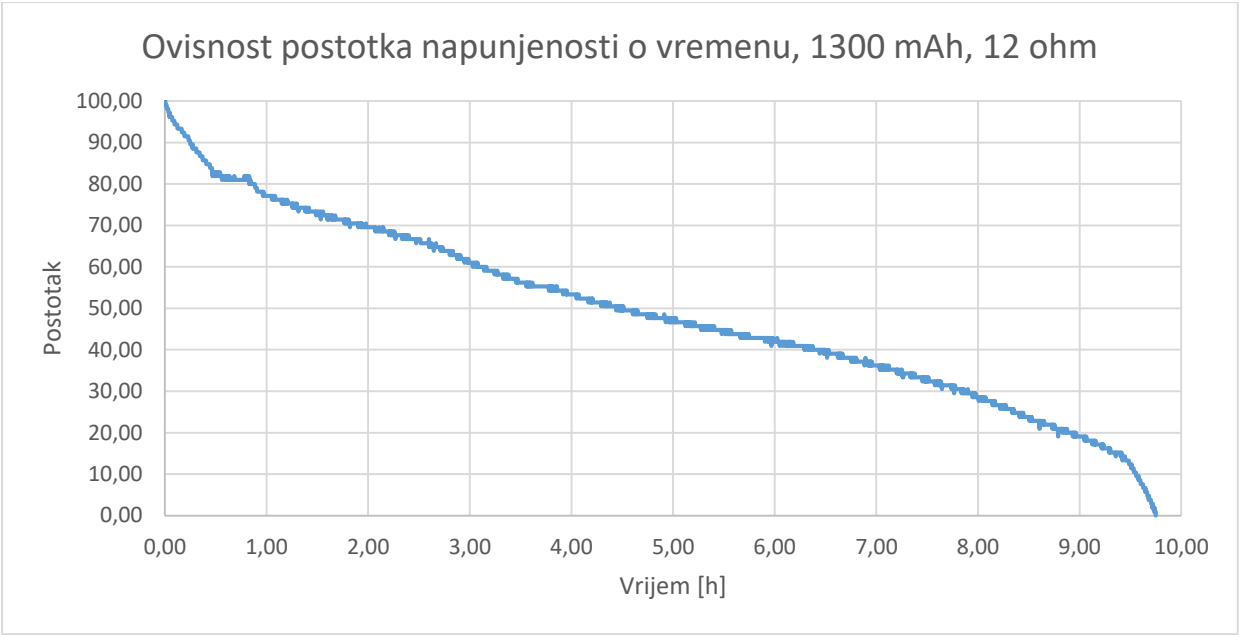


Slika 3.10. Ovisnost postotka napunjenost o vremenu kod baterije 1200 mAh, koristeći otpornik od 15 ohm-a.

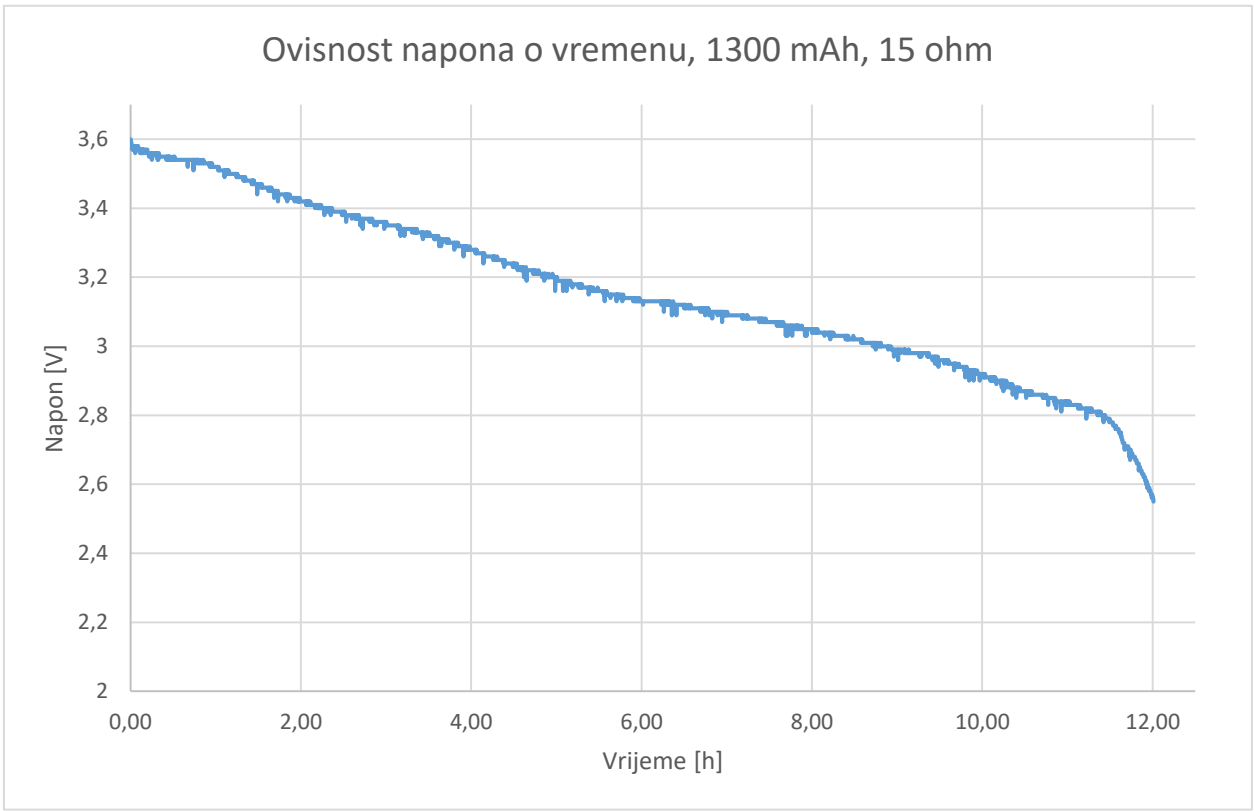
Iz grafova na slikama 3.9. i 3.10. je opet moguće primijetiti kako su grafovi slični, te iz uzorka od 2 baterije i 2 otpornika svi grafovi imaju slične krivulje različitih dužina zbog kapaciteta baterija i vrijednosti otpora korištenih otpornika. U nastavku su priložene slike ostalih mjerenja.



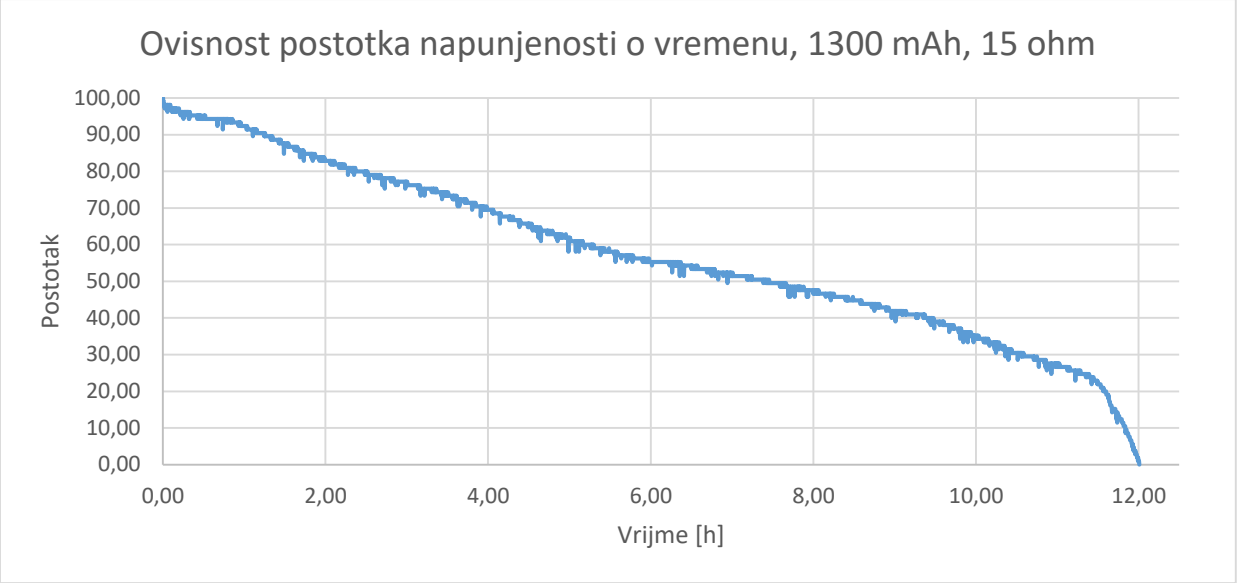
Slika 3.11. Ovisnost napona o vremenu kod baterije 1300 mAh, koristeći otpornik od 12 ohm-a.



Slika 3.12. Ovisnost postotka napunjenost o vremenu kod baterije 1300 mAh, koristeći otpornik od 12 ohm-a.



Slika 3.13. Ovisnost napona o vremenu kod baterije 1300 mAh, koristeći otpornik od 15 ohm-a.



Slika 3.14. Ovisnost postotka napunjenost o vremenu kod baterije 1300 mAh, koristeći otpornik od 15 ohm-a.

Napravljen je i program koji prikazuje trenutno stanje napunjenosti baterije, no kao što se može primijetiti iz grafova ovisnosti postotka napunjenosti o vremenu, taj podatak ne znači puno jer graf nije linearan, te nije moguće precizno odrediti koliko će dugo baterija nastavit raditi prije nego se isprazni. Program radi na temelju *interrupt* poziva kako bi se u željeno vrijeme prikazalo stanje baterije. Funkcije koje određuje postotak baterije su prikazane na slici 3.15., a primer ispisa na *Serial monitor* na slici 3.16.

```
void ocitaj(){
  measure();
  float postotak = (voltage - emptyVoltage) / percent;
  Serial.print("Baterija se nalazi na ");
  Serial.print(postotak, 2);
  Serial.print("% napunjenosti.");
  Serial.print('\n');
  delay(50);
}
```

Slika 3.15. Funkcije za očitavanje postotka baterije iz trenutnog napona.

```
14:07:08.946 -> Baterija se nalazi na 79.87% napunjenosti.
```

Slika 3.16. Primjer ispisa funkcije ocitaj iz slike 3.14.

3.5. Metoda pomoću look-up tablice

Pomoću metode sa look-up tablicom su ponovljena ista mjerenja kao i u metodi aproksimacije po naponu, samo je još dodan modul EEPROM koji je spojen na način da su SDA i SCL *pinovi* EEPROM-a spojeni na *pinove* A4 i A5 Arduina. EEPROM u sebi ima memoriju za 1024 bajtova (engl. *bytes*), ali su vrijednosti napona zapisane u varijablu tipa *float* koja zauzima 4 bajta memorije, što znači da ukupno možemo zapisati 256 vrijednosti napona u EEPROM look-up tablicu. Podaci su istovremeno bili zapisivani i u Excel-ov Data Streamer, no dobiveni grafovi su jednaki kao i grafovi iz poglavlja 3.4., pa stoga nisu prikazani u ovom poglavlju. Kako bi mogli uspješno očitavati vrijeme dok se baterija ne isprazni, potrebno je zapisati podatke u memoriju EEPROM-a. Napravljen je program koji će svakih deset minuta zapisati trenutno stanje baterije u trajnu memoriju EEPROM-a (Slika 3.17.).

```
void brojac(){
  sekunde++;
  if(sekunde == 10){
    sekunde = 0;
    brojacDeset++;
    measure();
  }

  if(brojacDeset == 60){
    brojacDeset = 0;
    EEPROM.put(adresa, voltage);
    adresa+=sizeof(float);
  }
}
```

Slika 3.17. Funkcija za zapisivanje podataka u EEPROM.

Kako bi se moglo točno svakih deset minuta zapisati stanje u memoriju, koristi se biblioteka *TimerOne.h*, koja pomoću *timer-a* sa korakom svaku jednu sekundu povećava brojač sekundi dok on ne dođe na deset. Nakon deset sekundi brojač se postavi na nula, drugi brojač se poveća za jedan i pozove se funkcija koja zapisuje podatke u Excel (Slika 3.6). Kada taj brojač dođe na 60, postavi se na nulu i vrijednost napona se zapiše u EEPROM. Odabran je vremenski interval od deset minuta zato što EEPROM ima ograničen broj podataka koji može biti zapisan u njemu, a pošto se za mjerenja koriste 2 baterije i 2 trošila, postoje 4 moguće kombinacije baterija-trošilo koje trebaju biti upisane u EEPROM. . Pošto su se baterije praznile između 4 i 12 sati, podatke nije bilo moguće zapisati u memoriju EEPROM-a sa jednakim rasponom adresa dodijeljenih svakoj kombinaciji baterija-otpornik, nego su bili zapisivani redom: baterija 1200 mAh s otpornikom od 15 ohma (adresa 1 – 125), baterija 1300 mAh s otpornikom od 15 ohma (adresa 129 – 421), baterija 1200 mAh s

otpornikom 12 od ohma (adresa 425 – 533) i baterija 1300 mAh s otpornikom od 12 ohma (adresa 537 – 773).

Kako bi se očitalo stanje baterije kada bude potrebno, koristi se *interrupt* poziv spojen na digitalni *pin* D2 koji je postavljen na načinu rada *INPUT_PULLUP* i tipkala spojenog na nulti potencijal. Program ispisuje na *serial monitor* trenutni postotak baterije i očekivano vrijeme prije nego što se baterija isprazni.

```
void ocitaj(){
  measure();
  while(voltage < trenutniNapon){
    prethodniNapon = trenutniNapon;
    adresa+=sizeof(float);
    EEPROM.get(adresa, trenutniNapon);
  }
  if((prethodniNapon - voltage) < (voltage - trenutniNapon)){
    float preostaloVrijeme = maxVrijemeBaterije - ((adresa - sizeof(float) - startAdresa) / sizeof(float) * 10);
    float postotak = (preostaloVrijeme / maxVrijemeBaterije) * 100;
    Serial.print("Baterija se nalazi na ");
    Serial.print(postotak, 2);
    Serial.print("%, sa otprilike ");
    Serial.print(preostaloVrijeme, 0);
    Serial.print(" preostalih minuta rada.");
    Serial.print('\n');
    delay(50);
  }
  else{
    float preostaloVrijeme = maxVrijemeBaterije - ((adresa - startAdresa) / sizeof(float) * 10);
    float postotak = (preostaloVrijeme / maxVrijemeBaterije) * 100;
    Serial.print("Baterija se nalazi na ");
    Serial.print(postotak, 2);
    Serial.print("%, sa otprilike ");
    Serial.print(preostaloVrijeme, 0);
    Serial.print(" preostalih minuta rada.");
    Serial.print('\n');
    delay(50);
  }
}
```

Slika 3.17. Funkcije za očitavanje stanja baterije iz trenutnog napona pomoću look-up tablice.

Vrijednosti imaju malu pogrešku jer se zbog ograničenosti memorije EEPROM-a zapisivao napon svakih deset minuta. Na slici 3.17. je prikazana funkcija za ispis postotka baterije i očekivanog vremena prije nego što se baterija isprazni. Funkcija sadrži naredbu *if* koja pomoću pamćenja prijašnjeg stanja uspoređuje je li trenutno stanje bliže vrijednosti zapisanoj na trenutnoj adresi u EEPROM-u ili na prijašnjoj, te ispisuje podatke za postotak i preostalo vrijeme ovisno o bližoj vrijednosti.

4. ZAKLJUČAK

U ovom završnom radu su uspoređene dvije metode za očitavanje stanja napunjenosti baterije (engl. *State of Charge*). Obije metode imaju i prednosti i mane, te je potrebno znati sa kojom se metodom očitalo stanje prije nego je moguće donijeti zaključak o preostalom iskoristivnom vremenu baterije.

Metoda aproksimacije stanja pomoću napona je jednostavna za izvest, ali se zadnjih 30% isprazni u vrlo kratkom vremenu zbog krivulje pražnjenja baterije koja nije linearna. Sve što je potrebno za aproksimaciju po naponu je maksimalna vrijednost napona, tj. vrijednost napona kada je baterija puna, i vrijednost napona kada isključujemo bateriju kako napon baterije ne bi pao na nulu. Ova metoda nije idealna zbog navike na linearni pad postotka napunjenosti baterije.

Metoda aproksimacije pomoću look-up tablice je nešto zahtjevnija od metode aproksimacije po naponu, jer je potrebno imati dodatni modul EEPROM-a stalno uključen kao vanjsku memoriju. Također, potrebno je i u EEPROM zapisati podatke iz kojeg će se iščitavati stanje baterije. Pomoću ove metode se dobiva znatno linearnija karakteristika pražnjenja, te je moguće prikazati očekivano preostalo vrijeme trajanja baterije pomoću početne i krajnje adrese u EEPROM-u sa poznatim i konstantnim vremenskim razmacima između dvije adrese. EEPROM se mogao koristiti u ovom završnom radu samo iz razloga jer je trošilo bilo konstantno, te ukoliko se promijeni trošilo, potrebno je njegove vrijednosti zapisati u EEPROM kako bi se dobio točan, približno linearan graf pražnjenja.

LITERATURA

1. Y. Baruskov, J. Qian, Battery Power Management for Portable Devices, Artech House, London, 2013

2. G. L. Plett (2015): Battery Management Systems, Volume 1: Battery Modeling Battery Modeling, [Internet], dostupno na https://pdfroom.com/books/battery-management-systems-volume-1-battery-modeling-battery-modeling/0q2JQW6egxE

SAŽETAK

U ovom završnom radu su prikazana 2 jednostavna načina očitavanja stanja napunjenosti baterije, a to su metoda aproksimacije po naponu i metoda pomoću EEPROM-ove look-up tablice. Metode su međusobno uspoređene kako bi se odredila preciznost i točnost.

Ključne riječi – Arduino, EEPROM, look-up tablica, postotak napunjenosti baterije

This final paper presents 2 simple ways of reading battery's state of charge, and those are voltage approximation method and method using EEPROM's look-up table. Methods are compared to each other to determine precision and accuracy.

Key words – Arduino, EEPROM, look-up table, battery's state of charge