Univerzitet u Nišu
Elektronski fakultet u Nišu

Stefan Ilić

Ispitivanje mogućnosti korišćenja komercijalnog tranzistora sa plivajućim gejtom kao detektora jonizujućeg zračenja u režimu visoke osetljivosti

MASTER RAD

Studijski program: Elektronika i mikrosistemi

Kandidat:
Stefan Ilić, br. ind. 929

Mentor:
prof. dr Goran Ristić

Ovaj master rad urađen je u okviru projekta ELICSIR koji je finansiran od strane Horizont 2020 programa za istraživanja i inovacije Evropske Unije prema sporazumu o grantu broj 857558

Niš, oktobar 2020.

Ispitivanje mogućnosti korišćenja komercijalnog tranzistora sa plivajućim gejtom kao detektora jonizujućeg zračenja u režimu visoke osetljivosti

Master rad

Zadatak:

1. *Realizacija programatora treće generacije električno programabilnog MOS tranzistora sa plivajućim gejtom.*
2. *Eksperiment ožračivanja uzoraka, u posebnom režimu visoke osetljivosti, jonizujućim zračenjem korišćenjem gama izvora.*
3. *Prikaz i diskusija dobijenih rezultata.*

Kandidat:
Stefan Ilić, br. ind. 929

Mentor:
prof. dr Goran Ristić

Datum prijave rada: 06.07.2020.

Datum predaje rada: 14.10.2020.

Datum odbrane rada: 19.10.2020.

Komisija:

prof. dr Goran Ristić

prof. dr Zoran Prijić

doc. dr Vojkan Davidović

Ispitivanje mogućnosti korišćenja komercijalnog tranzistora sa plivajućim gejtom kao detektora jonizujućeg zračenja u režimu visoke osetljivosti

SAŽETAK

U ovom radu je predstavljena upotreba komercijalnog MOS tranzistora sa plivajućim gejtom, takozvanim EPAD-om (električno programabilnog MOS tranzistora sa plivajućim gejtom), kao detektora jonizujućeg zračenja u posebnom režimu visoke osetljivosti (HIGH-SENS). Ideja je da se dizajnira linearni senzor sa konstantnom vrednošću osetljivosti na jonizujuće zračenje. Tehnologija sa plivajućim gejtom omogućava da se napravi poluprovodnički detektor sa znatno tanjim oksidom i na taj način se eliminišu neželjeni efekti kao što su gubljenje informacija tokom vremena (eng. fading), nekompatibilnost sa CMOS tehnologijom, velika temperaturna osetljivost itd. Ranija zapažanja pokazuju da se osetljivost tokom ozračivanja smanjuje jer se količina elektrona na plivajućim gejtom smanjuje. Režim visoke osetljivosti je osmišljen kako bi se povećala osetljivost detektora tako što se plivajući gejt tokom rada u polju zračenja ciklično puni elektronima i na taj način stalno održava napunjen plivajući gejt i samim tim postiže veća osetljivost. Takođe, zavisnost promene napona praga od doze nije linearna, ali se ovim režimom može stalno održavati linearnost ako se uzme dovoljno mali opseg u kome se kreće vrednost napona praga. Ovaj proces se konroliše od strane pomoćne elektronike koja se nalazi pored detektora. Eksperiment je sadržao dve varijante režima visoke osetljivosti: sa nultom i pozitivnom polarizacijom, pri čemu nulta polarizacija daje veću osetljivost. Rezultati pokazuju znatno veću osetljivost EPAD-a u režimu visoke osetljivosti i nulte polarizacije tokom ozračivanja nego bez takvog režima rada. Takođe svi EPAD-i u režimu visoke osetljivosti pokazuju linearnu zavisnost sa apsorbovanom dozom. Problem postoji u ponovljivosti eksperimenta, pretpostavlja se da je uzrok tunel oksid kroz koji se puni plivajući gejt elektronima koji nema zadovoljavajući kvalitet za ovakav režim rada. Takođe, funkcionisanje sistema je dosta složeno i zahteva obimnu kalibraciju softvera pre početka rada.

Ključne reči: Plivajući gejt; MOS tranzistor; jonizujuće zračenje; detektor;

Investigation of the possibility of using a commercial floating gate transistor as an ionizing radiation detector in high sensitivity mode

ABSTRACT

This paper presents the use of a commercial MOS transistor with a floating gate, the so-called EPAD (Electrically Programmable Analog Device), for ionizing radiation detectors in a special high-sensitivity mode (HIGHSENS). The idea is to design a linear sensor with a constant value of sensitivity to ionizing radiation. Floating gate technology provides a semiconductor detector with a much thinner oxide and thus eliminates side effects such as fading, CMOS incompatibility, high-temperature sensitivity, etc. Earlier observation shows that sensitivity decrease because the amount of electrons at the floating gate decreases. The high sensitivity mode is designed to increase the sensitivity of the detector by cyclically charging the floating gate with electrons during operation in the radiation field and thus constantly maintaining a charged floating gate and thus higher sensitivity. Also, the dependence of the threshold voltage shift on the dose is not linear, but this mode can constantly maintain linearity if a sufficiently small linear area is selected where the value of the threshold voltage oscillates. This process is controlled by auxiliary electronics located next to the detector. The experiment contained two variants of the high-sensitivity regime: with zero and positive bias, where zero bias gives higher sensitivity. The results show a significantly higher sensitivity of EPAD in the mode of high sensitivity and zero bias during irradiation than without such mode of operation. Also, all EPADs in the high-sensitivity mode show a linear dose-dependence. The problem exists in the repeatability of the experiment, it is assumed that the cause is the tunnel oxide through which the floating gate is charged with the electrons, which does not have a satisfactory quality for this kind of work. Also, the operation of the system is quite complex and requires extensive calibration of the software.

Keywords: Floating gate; MOS transistor; ionizing radiation; detector;

ZAHVALNICA

Zahvaljujem Goranu Ristiću, redovnom profesoru Elektronskog fakulteta, Univerziteta u Nišu, na strpljivoj analizi svakog koraka u izradi master rada. Velika posvećenost profesora Ristića kako u nastavi tako i u mentorisanju odigrala je značajnu ulogu u mom napredovanju i školovanju. Veliko hvala Aleksandru Jakšiću, Nikoli Vasoviću i Russell-u Duane-u, istraživačima sa Nacionalnog Instituta Tyndall u Irskoj na nesebičnoj pomoći i dragocenim sugestijama tokom izrade projekta koji je doprineo u tumačenju navedenih rezultata. Marku Anđelkoviću, istraživaču na Institutu za mikroelektroniku (IHP) u Nemačkoj dugujem zahvalnost za osmišljavanje koncepta evropskog projekta ELICSIR i savete u pisanju master teze.

SPISAK KORIŠĆENIH SKRAĆENICA

Skraćenica	Puni naziv
EPAD	Electrically Programmable Analog Device
DUT	Device Under Test
SMU	Source Measure Unit
GPIB	General Purpose Interface Bus
IEEE	Institute of Electrical and Electronics Engineers
ZTC	Zero Temperature Coefficient
USB	Universal Serial Bus
PMMA	Poly(Methyl Methacrylate)
QFN	Quad-Flat No-leads
TID	Total Ionizing Dose

Sadržaj

1. Uvod

Civilizacija na planeti zemlji se konstantno razvija. Pokretač svakog razvoja je tehnološki napredak. Prema Kardaševoj skali ljudska civilizacija se nalazi između tipa 0 i 1. Kardaševa skala je metoda merenja civilizacijskog nivoa tehnološkog napretka na osnovu količine energije koju je u stanju civilizacija da koristi. Meru je predložio sovjetski astronom Nikolaj Kardašev 1964. godine. Da bismo postigli tip 1 potrebno je da koristimo svu energiju koja je dostupna na planeti Zemlji. Današnji naučnici procenjuju da ako svake godine uvećamo iskorišćenje energije na Zemlji za 3 procenta, da bi za 100 do 200 godina postigli tip 1 civilizacijskog napretka.

Dakle, razvoj civilizacije se meri količinom energije koja se može iskoristiti. Jedan od ključnih izvora predstavlja nuklearna energija i njen razvoj. U Francuskoj nuklearne elektrane služe kao primarni izvor energije. Moderna civilizacija je stalno izložena sve većem uticaju jonizujućeg zračenja. Kada se postigne tip 1, prelazi se na tip 2 koji predstavlja osvajanje Sunčevog solarnog sistema i konačno potpuno iskorišćenje Sunčeve energije koja se emituje. Tip 3 razvoja civilizacije predstavlja osvajanje galaksije i iskorišćenje energije u matičnoj galaksiji. Kako bi civilizacija pravilno napredovala nauka se mora razvijati u svim aspektima. Ako napravimo analogiju i posmatramo nauku kao čovekovo telo, svaki deo tela je potrebno pravilno razvijati kako bi čovek imao zdravo telo i zdrav duh. Ni jedan deo tela ne sme biti zakržljao, tako ni jedan deo nauke ne sme biti nedovoljno razvijen, jer onda nauka neće biti na zdravoj osnovi.

Jedan nezaobilazni aspekt nauke predstavlja nauka o senzorima pomoću koje mi proširujemo naše čulne mogućnosti. Zahvaljujući raznim senzorima, možemo videti, čuti i detektovati razne fizičke pojave koje su van opsega naših čula. Jonizujuće zračenje predstavlja upravo jednu od pojava u prirodi koju naša čula ne mogu osetiti. Zato je potrebno razvijati nove senzore u ovoj oblasti koji mogu doprineti tehnološkom napretku i uzdizanju civilizacijskog nivoa.

Uvek postoji težnja za svaku fizičku pojavu napraviti idealni senzor. Pod idealnim senzorom se najčešće smatra senzor koji na datu fizičku pojavu ima linearni odziv, što veću osetljivost, ponovljivost merenja, što veći opseg itd. Idealni detektor jonizujućeg zračenja treba da ispunjava prethodno spomenute osobine.

Sedamdesetih godina prošlog veka PMOS tranzistori (RADFET-i) su postali pretpoznatljivi kao senzori jonizujućeg zračenja. Usavršavani su sve do sada i njihova najveća primena je u svemirskim istraživanjima. Pored toga koriste za praćenje nivoa zračenja u radioterapiji i laboratorijama u kojima je povećan nivo zračenja. Kako postoji stalna težnja ka idealnom senzoru jonizujućeg zračenja koji će imati sve bolje karakteristike, potrebno je prevazići nedostatke PMOS dozimetara, kao što su gubitak informacija o primljenoj dozi, nekompatibilnost sa CMOS tehnologijom, nelinearnost, itd. Sve veći broj radova ukazuje na mogućnost korišćenja MOS tranzistora sa plivajućim gejtom u dozimetrijske svrhe.

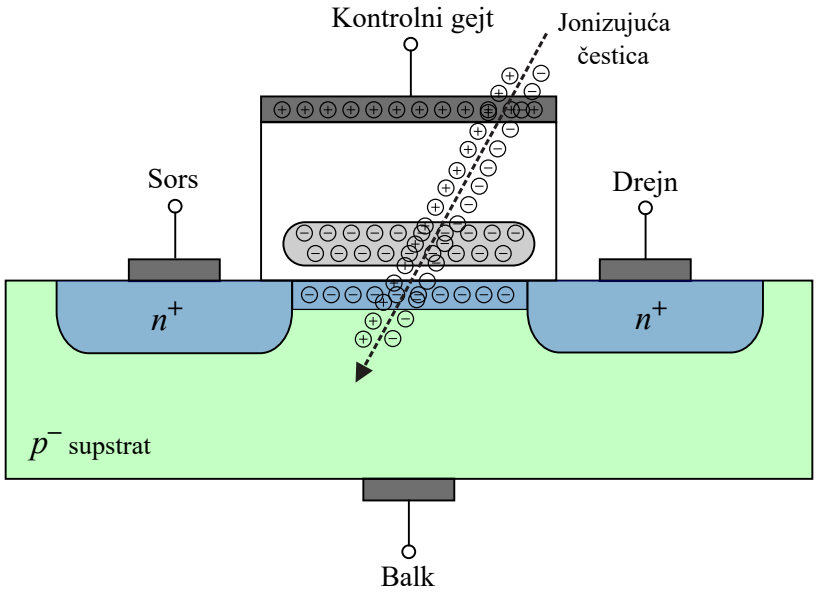
Zadatak ovog rada je ispitivanje mogućnosti korišćenja MOS tranzistora sa plivajućim

gejtom kao detektora jonizujućeg zračenja u režimu visoke osetljivosti. MOS tranzistor sa plivajućim gejtom koji je korišćen u ovom radu predstavlja komercijalnu komponentu namenjenu za potpuno druge svrhe, kao što su strujna ogledala, strujni izvori i svuda gde je potrebno precizno podešavanje napona praga. Iako nisu namenjeni za ovu svrhu, ovde će biti predstavljena ideja da ovi tranzistori rade u posebnom režimu koji može obezbediti određene osobine idealnog senzora kao što su linearnost i velika osetljivost.

Pored uvodne, rad sadrži još sedam glava. U drugoj glavi je opisan MOS tranzistor sa plivajućim gejtom, objašnjen je njegov princip rada kao detektor jonizujućeg zračenja i detaljno je prikazan režim rada visoke osetljivosti MOS tranzistora sa plivajućim gejtom. Zatim, u trećoj glavi je opisan programator treće generacije koji je dizajniran specijalno za ove tranzistore, prikazane su njegove karakteristike i poboljšanja u odnosu na prethodne generacije. Nakon toga u četvrtoj glavi je predstavljen eksperiment koji se odvijao na Institutu za nuklearne nauke "Vinča", potom je opisana postavka samog eksperimenta, kao i oprema koja je korišćenja. U petoj glavi predstavljeni su rezultati eksperimenta i diskusija dobijenih rezultata. U šestoj glavi su dati zaključci izvedeni na osnovu rezultata istraživanja koji su prethodnoo predstavljeni. Sedma glava sadrži spisak korišćene literature, a osma glava je dodatak koji sadrži fotografije eksperimenata i listing kodova.

2. MOS tranzistor sa plivajućim gejtom u režimu visoke osetljivosti

MOS tranzistor sa plivajućim gejtom poseduje dva gejta. Jedan gejt je standardni gejt kao i kod ostalih MOS tranzistora koji se naziva kontrolni gejt, i njemu je moguće pristupiti preko izvoda. Drugi gejt je dodatni gejt koji je oksidom izolovan sa svih strana, tzv. plivajući gejt (eng. *floating gate*). On se nalazi između kanala i kontrolnog gejta i njemu nije moguće pristupiti. Kako bi se dovelo naelektrisanje na plivajući gejt potrebno je polarisati kontrolni gejt tako da se stvori jako električno polje koje će privući elektrone iz kanala ka plivajućem gejtu. Oksid između kanala i plivajućeg gejta je tanak tako da elektroni mogu da dođu do plivajućeg gejta tunelovanjem ili injekcijom vrućih nosilaca blizu drejna [1]. Kada se elektroni nađu na plivajućem gejtu oni tu ostaju zarobljeni sve dok ih jako električno polje suprotne polarizacije sa kontrolnog gejta ne vrati nazad u kanal. Kada na plivajućem gejtu nema naelektrisanja, tada će pri naponu na kontrolnom gejtu jednakom naponu praga (V_{th}) tranzistor provesti i detektovaće se struja na izlazu. Nakon dovođenja naelektrisanja na plivajući gejt, prenosna karakteristika se pomera za ΔV_{th}, tako da je potrebno dovesti veći napon na gejt kako bi se formirao kanal.



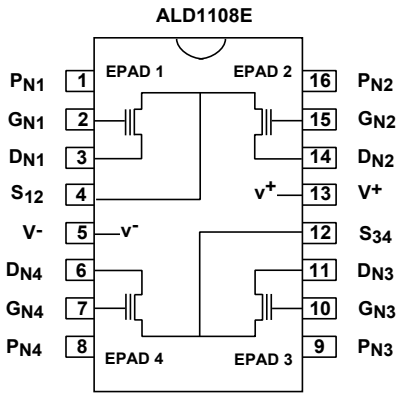
Slika 2.1: N-kanalni MOS tranzistor sa napunjenim plivajućim gejtom pod dejstvom jonizujućeg zračenja.

Razlog pomeraja prenosne karakteristike leži u tome što elektroni koji se nalaze na plivajućem gejtu odbijaju elektrone koji treba da oforme kanal. Samim tim je potrebno dovesti veći napon na kontrolni gejt kako bi se nadjačalo električno polje koje potiče od

elektrona na plivajućem gejtu i tako došlo do formiranja kanala [2]. Dejstvom jonizujućeg zračenja na klasičan MOS tranzistor bez plivajućeg gejta dolazi do formiranja defekata u oksidu, čije električno polje utiče na otežano ili olakšano formiranje kanala, u zavisnosti da li je reč o PMOS ili NMOS tranzistoru [3].

Međutim, delovanje jonizujućeg zračenja kod NMOS tranzistora sa plivajućim gejtom ispoljava se smanjenjem naelektrisanja na plivajućem gejtu. Na slici 2.1 je prikazan N-kanalni MOS tranzistor sa plivajućim gejtom koji je napunjenim elektronima. Pri dejstvu jonizujućeg zračenja generišu se parovi elektron-šupljina u oksidu MOS tranzistora. Električno polje sa plivajućeg gejta razdvaja generisana naelektrisanja privlačenjem šupljina. Šupljine se rekombinuju sa elektronima koji se nalaze na plivajućem gejtu i time se smanjuje količina naelektrisanja što dovodi do smanjenja vrednosti napona praga tranzistora.

EPAD predstavlja električno programabilni MOS tranzistor sa plivajućim gejtom (eng. Electrically Programmable Analog Device). To je patentirana tehnologija koja je nastala devedesetih godina u Americi od strane *Advanced Linear Devices* [4]. Na slici 2.2 je prikazan raspored pinova integrisanog kola ALD1108E koje je korišćeno u radu. ALD1108E sadrži četiri EPAD-a, svaki od njih ima poseban pin za programiranje koji prilikom korišćenja tranzistora može da bude vezan za masu ili da pliva. Od četiri EPAD-a, svaka dva imaju zajednički sors. Dovođenje negativnog napona na kontrolni gejt i električno brisanje plivajućeg gejta nije moguće zbog dodatne diode čija je anoda vezana za sors, a katoda za kontrolni gejt tranzistora. Moguće je samo punjenje plivajućeg gejta elektronima.



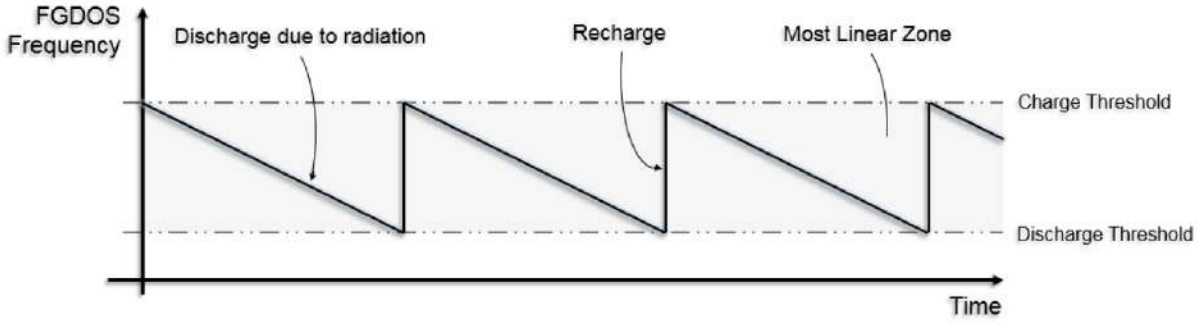
Slika 2.2: Konfiguracija pinova za integrisano kolo ALD1108E koje sadrži četiri EPAD-a [5].

Senzor jonizujućeg zračenja baziran na MOS tranzistoru sa plivajućim gejtom koji poseduje visoku osetljivost je nova tehnologija razvijena od strane *iC Malaga* iz Španije. Ovaj senzor se može koristiti za različite vrste aplikacija kao na primer u medicini, ličnoj dozimetriji, svemirskim istraživanjima, vojnoj odbrani, u oblasti fizike visokih energija itd. Tehnologija plivajućeg gejta primenjena u svrhe dozimetra jonizujućeg zračenja omogućava da se postignu četiri osnovna dozimetrijska zahteva:

- Linearnost: do 98% sa izvorom ^{60}Co
- Integracija: dva senzora u pakovanju QFN od 5 mm x 5 mm
- Laka upotreba: očitavanje preko serijskog perifernog interfejsa (SPI)
- Niska potrošnja energije: dostupan je pasivni režim rada

Minimalna količina radijacije koja se može detektovati pomoću ovog senzora iznosi 0.1 mGy, dok je maksimalna ukupna apsorbovana doza (TID) koju sistem može da izdrži do 200 Gy. Senzor ima mogućnost da detektuje jonizujuće zračenje i bez napajanja tada radi u takozvanom pasivnom režimu. Jedino se sistem mora napajati prilikom očitavanja podataka. Drugi režim rada je naravno aktivni kada je moguće merenje apsorbovane doze jonizujućeg zračenja u realnom vremenu [6].

Januara 2016. godine potpisana je saradnja između CERN-a i iCMalage o razvoju i karakterizaciji senzora sa plivajućim gejtom. Nakon toga grupa autora iz CERN-a su testirali dozimetar sa plivajućim gejtom [7]. Integrisano kolo (čip) se sastoji iz FPGA i dozimetra sa plivajućim gejtom. FPGA poseduje kontroler koji upravlja dozimetrom. Pomoću njega se vrši proces očitavanja trenutne vrednosti apsorbovane doze i proces punjenja (programiranja) plivajućeg gejta elektronima. Na slici 2.3 predstavljen je funcionalni proces automatskog punjenja i pražnjena plivajućeg gejta pod uticajem jonizujućeg zračenja.



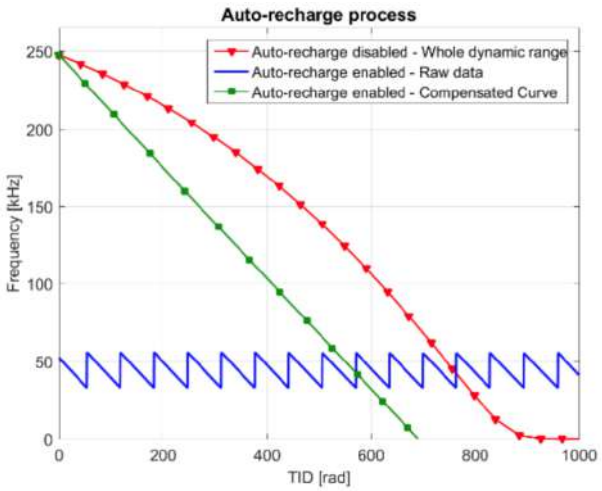
Slika 2.3: Funcionalna operacija automatskog procesa reprogramiranja dozimetra sa plivajućim gejtom tokom ozračivanja [7].

Izlazna vrednost ovog dozimetra je frekvencija, dok gornja i donja granica na grafiku 2.3 predstavljaju vrednosti frekvencije za koje je plivajući gejt napunjen do maksimuma (najveća vrednost napona praga) i donja granica dozvoljene vrednost napona praga do koje se prazni plivajući gejt. Možemo uočiti da vreme punjenja plivajućeg gejta elektronima jako malo. Takođe primećujemo da je linearna zavisnost promene frekvencije tokom vremena, pošto je konstanta jačina jonizujućeg zračenja (^{60}Co), takođe je linearna zavisnost i sa apsorbovanom dozom. Na slici 2.4 prikazana je realna izlazna karakteristika dozimetra tokom ozračivanja.

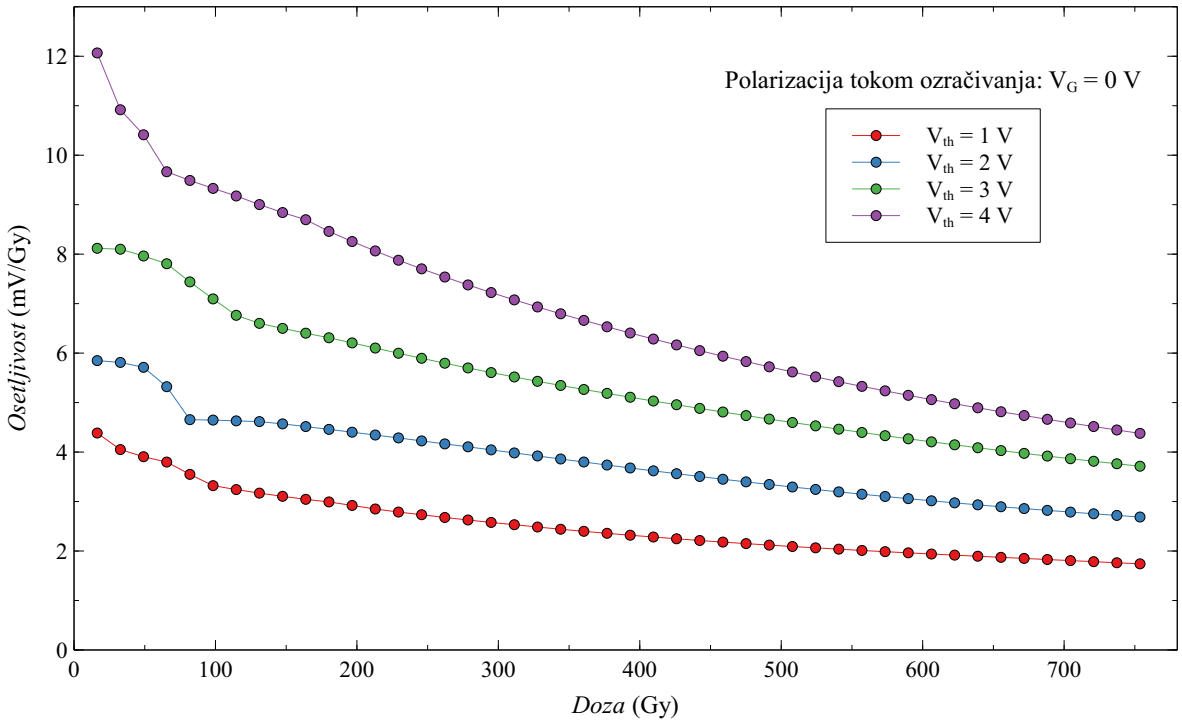
Korisni rezultat predstavlja zelena kriva na slici 2.4 koja je dobijena tako što su uzeti linearni delovi plave krive i spojeni u jednu. Proces reprogramiranja traje manje od dve sekunde. Može se zaključiti na osnovu rezultata u ovom radu da senzor ne radi samo dve sekunde na svakih 0.59 Gy-a (59 rada) jer se tada odbija proces punjenja plivajućeg gejta elektronima i nije moguće pratiti vrednost napona praga. Korišćenjem dva ovakva senzora koja su uparena na čipu moguće je prevazići ovaj kratak period i omogućiti stalno praćenje u realnom vremenu. Ovakav senzor se pokazao kao obećvajuć u dozimetrijskim aplikacijima [8, 7, 9].

U diplomskom radu analizirano je ponašanje EPAD-a sa različitim tipovima polarizacije u polju jonizujućeg zračenja [10]. Posmatrajući grafik 2.5 za EPAD-e sa nultom polarizacijom, možemo uočiti da na početku (u malom opsegu) tranzistor ima veliku osetljivost.

Uočavamo da prva tačka za EPAD sa $V_{th} = 4\ V$ ima vrednost 12.067 mV/Gy za

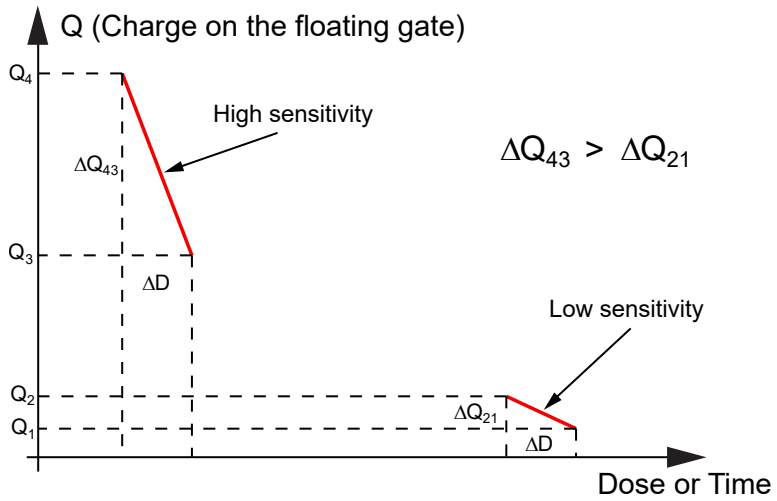


Slika 2.4: Zavisnost izlazne frekvencije od ukupna apsorbovane doze (TID); Crvena boja predstavlja pražnjenje plivajućeg gejta bez automatskog reprogramiranja. Plava boja predstavlja pražnjenje i punjenje plivajućeg gejta pomoću auto-reprogramiranja. Zelena boja predstavlja podatke koji su obeleženi plavom bojom ali sada spojeni u kompenzacionu krivu. [7].



Slika 2.5: Osetljivost EPAD-a sa nultom polarizacijom tokom ozračivanje za različite vrednosti početnog napona praga.

apsorbovanu dozu 16.384 Gy. Pretpostavljamo da bi za manji opseg doze EPAD imao veću osetljivost, ako pratimo zavisnost na grafiku možemo primetiti eksponencijalnu pravilnost. Kao što je poznato kod PMOS dozimetara osetljivost opada sa dozom, što je i ovde slučaj kada je gejt nulto polarisan. Videli smo da dinamički polarisane komponente imaju gotovo konstantnu osetljivost na merenom opsegu doza, ali je njihova osetljivost relativno mala (do 7.55 $\frac{mV}{Gy}$). Ako bi pravili stekovanu strukturu dinamički polarisanih EPAD-a to bi svakako povećalo osetljivost ali bi unelo i veći šum, što verovatno nije dobro. Međutim, ako bi "držali" EPAD stalno u oblasti visoke osetljivosti onda bi postigli linearnu zavisnost sa apsorbovanom dozom što je cilj idealnog dozimetra, da ima konstatno visoku osetljivost. Najveći problem ovde je taj što se struktura MOS tranzistora sa plivajućim gejtom tokom stalnog procesa pražnjenja i punjenja oštećuje, pa je pitanje do koje doze tako osmišljen dozimetar može da funkcioniše. Takođe, pitanje je i ponovljivosti procesa pražnjenja EPAD-a, jer u idelnom slučaju dozimetar će se uvek prazniti istom zavisnošću ako je napunjen do iste vrednosti početnog napona praga. U praksi verovatno neće biti takav slučaj jer postoji degradacije strukture i sistem može izdržati samo određeni broj ciklusa sa približno istim karakteristikama. Bitno je naglasiti da proces punjenja na zamišljenom opsegu mora biti, u polju zračenja, brži od procesa pražnjenja u istom opsegu kako bi bilo moguće izvesti reprogramiranje tokom ozračivanja. Na slici 2.6 možemo videti ilistraciju pražnjenja dozimetra sa plivajućim gejtom u oblasti visoke i niske osetljivosti.



Slika 2.6: Ilustracija naelektrisanja na plivajućim gejtom u zavisnosti od vremena ili doze. Predstavljene su visoka i niska osetljivost.

Ako poredimo visoku i nisku osetljivost na grafiku (Sl. 2.6) uočavamo da za istu vrednost apsorbovane doze (ΔD) kod visoke osetljivosti naelektrisanje se smanji sa Q_4 na Q_3 (ΔQ_{43}), a kod niske osetljivosti sa Q_2 na Q_1 (ΔQ_{21}). Dakle, što je veća apsolutna količina naelektrisanja na plivajućem gejtu, to je veća promena naelektrisanja za malu količinu doze.

Ideja eksperimenta je testirati EPAD u režimu rada visoke osetljivosti i pratiti njegovo ponašanje, što podrazumeva: pouzdanost ovakvog rada, pomeraj napona praga i brzina programiranja tokom zračenja. U idealnom slučaju ovaj eksperiment bi trebalo sprovesti do otkaza EPAD-a i tako videti koliko on može maksimalnim broj ciklusa da ostvari u ovakvom režimu. Ključno je da izabrani opseg bude takav da programiranje oduzima malu količinu vremena, a za to vreme da drugi EPAD na čipu pokriva takozvani slepi deo senzora (eng. blind time). Sa strane izvodljivosti, najveći je zahtev napisati dobar

program koji će automatski prelazati sa praćenja jednog EPAD-a na drugi. Opseg u visokoj osetljivosti mora biti jako mali, jer programiranje traje duže kada je plivajući gejt napunjen, nego kada je prazan [11].

Velike poteškoće su prilikom testiranja softvera i podešavanja parametara programiranja EPAD-a, zato što se sva kalibracija sistema mora odvijati direktno dok se komponenta ozračuje. Dakle, nije moguće simulirati sistem i testirati ga u laboratoriji bez izvora.

3. EPAD programator treće generacije

Razvijena je treća generacija EPAD programatora koju ćemo predstaviti u ovom radu. Naime, prva generacija programatora je predstavljena na konferenciji ETRAN 2018. godine [12]. U ovoj verziji vreme programiranja se unosilo ručno preko Arduino IDE softvera pomoću koga je Arduino Nano za zadato vreme uključivao MOS tranzistor putem koga se dovodila povorka signala na EPAD. Način programiranja je na osnovu velikog broja pokušaja empirijski utvrđen. Između prve i druge generacije izvršena je dekapsulacija i reverzno inženjerstvo integrisanog kola ALD1108E [13]. Zahvaljujući tome znanje o EPAD tehnologiji je znatno uvećano i ubrzo je nastala druga generacija programatora koja automatizovano vrši punjenje EPAD-a do željene vrednosti napona praga i ona je predstavljena u diplomskom radu [10]. U drugoj generaciji programator naizmenično meri napon praga i programira EPAD po prethodno određenoj funkciji približavajući se tako željenoj vrednosti. Pošto proces programiranja nije reverzibilan programator ne sme da pređe zadatu vrednost napona praga. Programator druge generacije može da programira samo jedan EPAD na čipu. Da bi se programirao sledeći EPAD na čipu potrebno je ručno prespojiti žice na test fikturi (eng. test fixture). EPAD programator treće generacije ima mogućnost automatizovanog programiranja svih EPAD-a na čipu ALD1108E. Dizajnirano električno kolo za programiranje poseduje četiri nezavisna SMU kanala koja su preko GPIB-a povezana na računar. U našem slučaju korišćeni su sledeći uređaji: Keithley 2636A koji ima dva SMU kanala, Keithley 2400 i Keithley 2410 koji imaju po jedan SMU kanal. Naravno, mogu se koristiti i drugi uređaji za potrebe EPAD programatora, potrebno je samo u programu podesiti komande za kontrolu datih uređaja. Program je napisan u C#-u i prikazan je u dodatku.

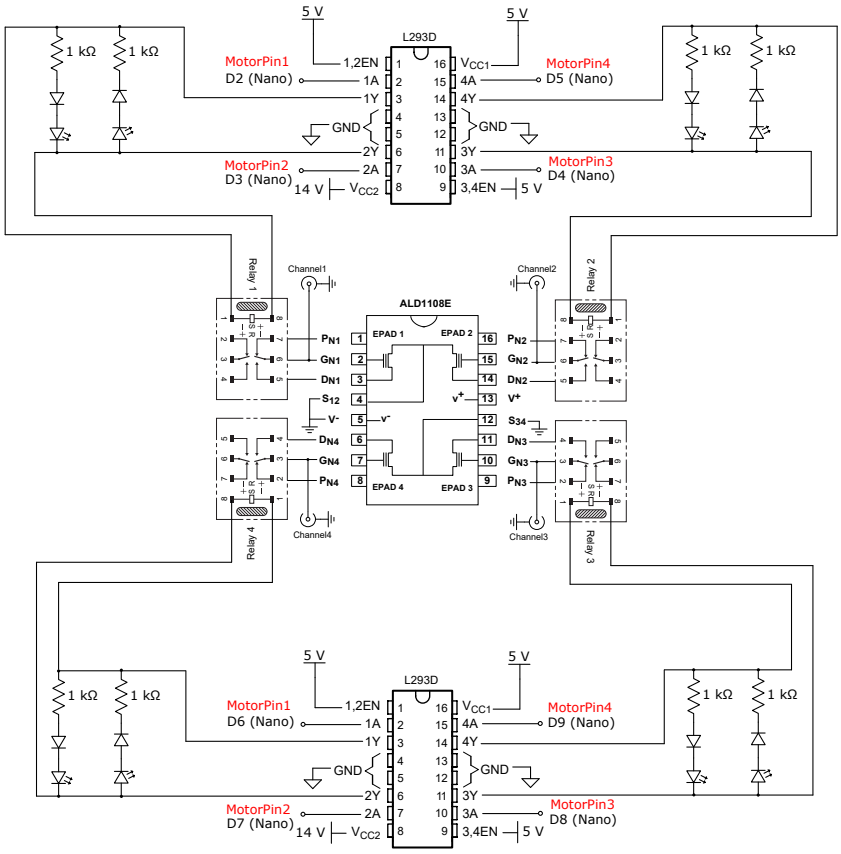
EPAD programator treće generacije poseduje sledeće mogućnosti:

- Korisnik na početku unosi vrednosti napona praga svih EPAD-a na čipu ALD1108E koje želi da programira,
- Programator redom programira EPAD-e na čipu,
- Tokom procesa programiranja vrše se očitavanja napona praga koje se ispisuju na displej za svaki EPAD posebno,
- Kada programator postigne zadatu vrednost prikazuje se funkcija programiranja datog EPAD-a na grafiku i nakon toga izvršava se premeravanje cele prenosne karakteristike tranzistora (potpragovska i nadpragovska oblast).

Prenosna karakteristika se meri tako što se propušta struja i meri napon. Snima se potpragovska i nadpragovska karakteristika od 50 pA do 3 mA. Pre snimanje karakteristike

propušta se struja od 1 µA kako bi se odredio napon praga EPAD-a, a zatim i struja od 68 µA, kako bi se odredio napon u ZTC tački.

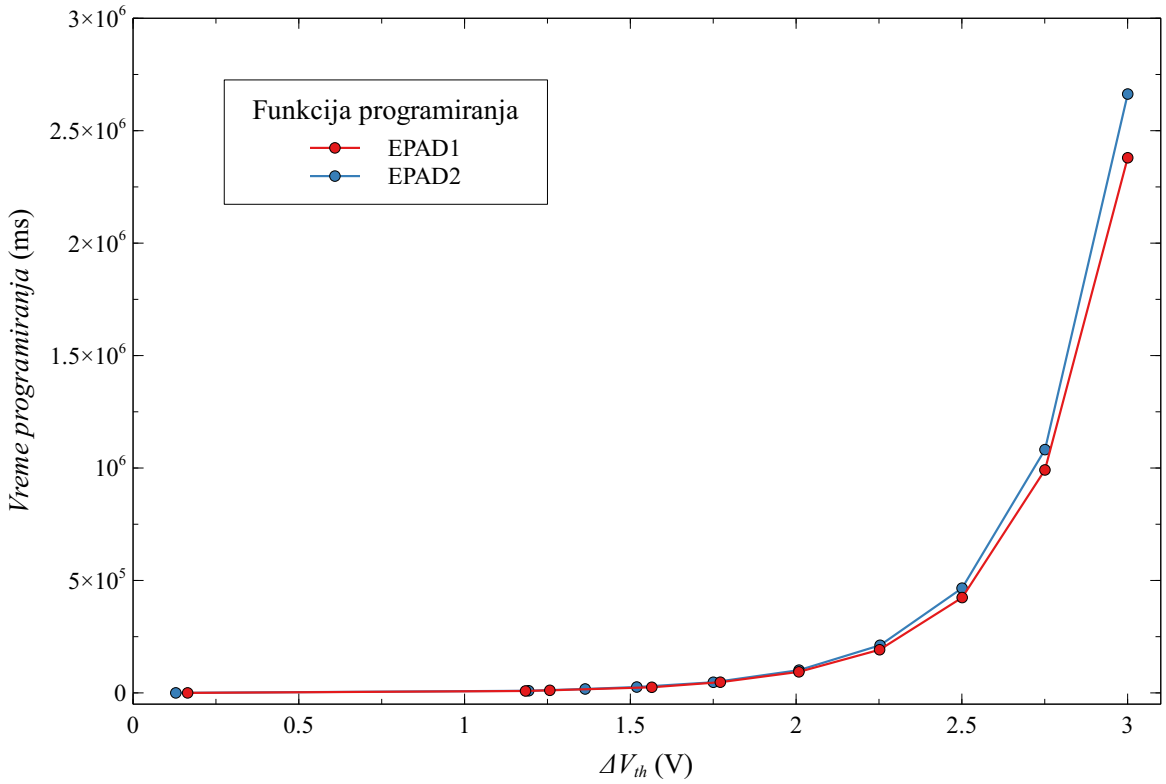
Sistem je kontrolisan preko GPIB 488 IEEE protokola, pomoću Windows aplikacije napisane u C#. Svaki od kanala je povezan za gejt EPAD-a i preko releja se u jednoj konfiguraciji vrši programiranje EPAD-a, a u drugoj merenje prenosne karakteristike EPAD-a. Releji su kontrolisani preko motor drajvera integrisanog kola L293D koji putem Arduino Nano jedinice i serijskog porta komunicira sa Windows aplikacijom. Električna šema programatora za EPAD-e treće generacije je prikazana na slici 3.1.



Slika 3.1: Električna šema EPAD programatora treće generacije

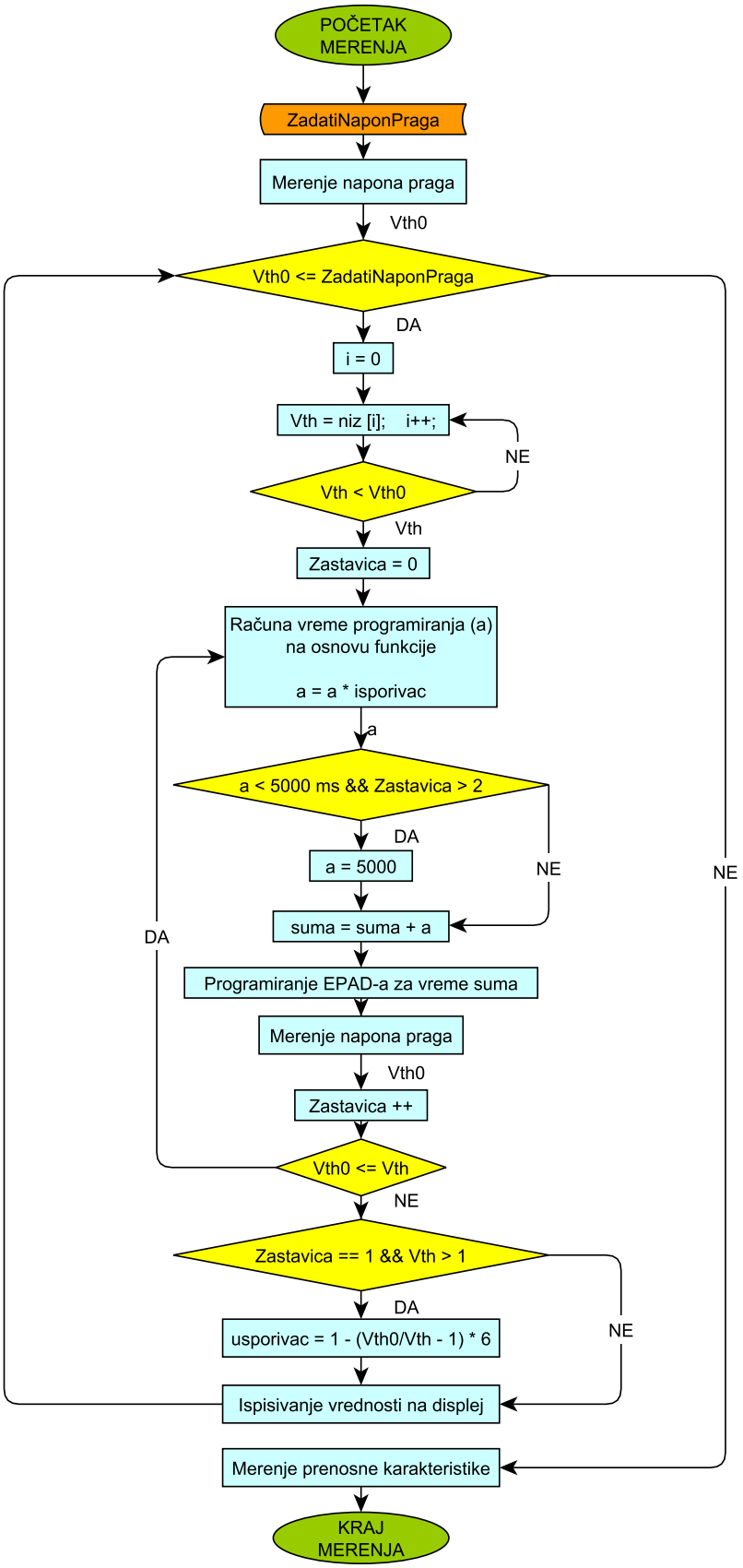
Algoritam za EPAD treće generacije je unapređen u odnosu na prethnodnu generaciju, tako da su sve greške koje su primećene u radu otklonjene. Povrh toga, novi algoritam sada beleži podatke o funkciji programiranja svakog EPAD-a. Naime, u programu postoji zadat niz vrednosti napona praga koje u toku procesa programiranja treba postignuti i za svaku vrednost upisati vreme u poseban fajl koje je potrebno da se do te vrednosti napona praga dođe kao i samu vrednost napona praga. Na osnovu ovih podataka moguće je nacrtati funkciju programiranja datog EPAD-a i uporediti je sa funkcijama drugih EPAD-a. Na slici 3.2 predstavljen je grafik zavisnosti vremena programiranja i promene napona praga za dva EPAD-a na istom čipu. Uočavamo da funkcija ima isti oblik, što važi za sve EPAD-e, ali da krajnje vrednosti nisu identične, dakle i na istom čipu funkcija programiranja nije ista za sve EPAD-e.

Novi algoritam za programator treće generacije je prikazan na slici 3.3. Na početku korisnik zadaje željeni napon praga EPAD-a koji se programira, zatim se izmeri trenutni napon praga ($Vth0$) pomoću uređaja. Sledi *while* petlja sa uslovom da je trenutni napon praga manji ili jednak zadatom. Sve dok ovaj uslov važi odvija se proces programiranja.



Slika 3.2: Funkcije programiranja EPAD-a.

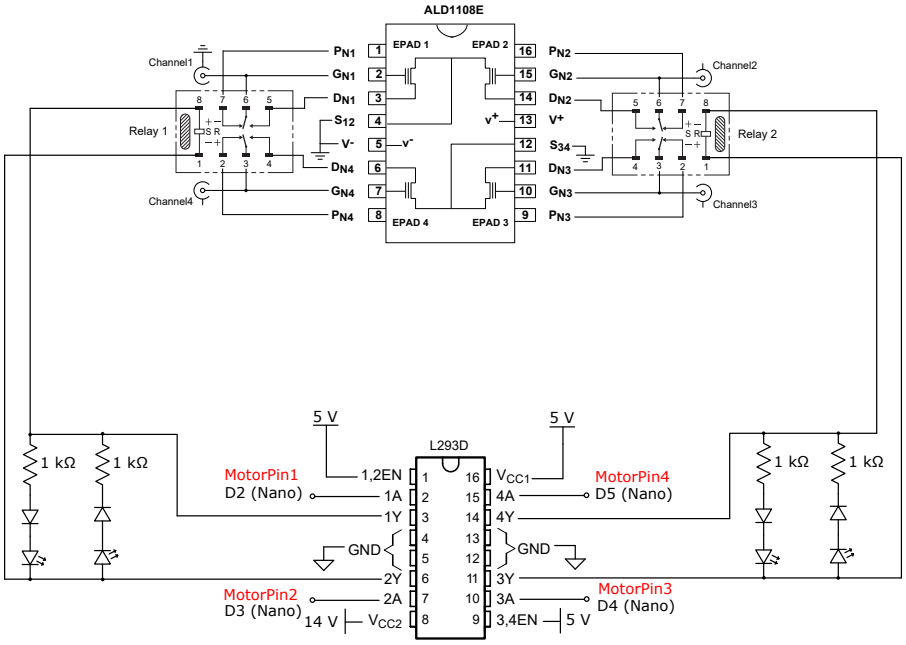
U petlji se prvo odvija traženje veće vrednosti napona praga od trenutne iz zadatog niza. Ovaj deo koda služi da se proces programiranja odvija sa određenim korakom kako bi se beležili podaci o funkciji programiranja svakog EPAD-a. Vrednosti niza su date u dodatku. Potom se pomoću funkcije koja je detaljno opisana u diplomskom radu generiše vreme programiranja [10]. Promenljiva usporivač ima početnu vrednost jedinicu. Njena uloga je da smanji vreme programiranja ako program primeti da je promena napona praga veća nego što se to funkcijom očekuje. Upravo je uloga promenljive zastavica da zabeleži da li je program prošao samo jednom kroz malu *do while* petlju Vth0 ¡= Vth. Nakon toga se *if* naredbom ispituje da li je vreme programiranja "a"manje od 5000 milisekundi i da li je kroz petlju program prošao više od dva puta. Ako su oba uslova ispunjena vreme programiranja dobija vrednost 5000 ms. Ova naredba ima ulogu da anulira jako male vrednosti vremena programiranja koje mogu da se generišu kada se vrednost napona praga znatno približi traženoj vrednosti. U takvim slučajevima programiranje može trajati satima, *if* naredba eliminiše takvu mogućnost. Brojka od 5000 milisekundi je emirijski utvrđena kao dovoljno dugačak period programiranja kako bi se uočila promena vrednosti napona praga. Nakon *do while* petlje nailazi se na *if* naredbu koja može promeniti vrednost usporivača datim empirijskim izrazom. Ovaj izraz je dobijen na osnovu obrade rezultata funkcija programiranja za algoritam koji nije sadržao usporivač. Na kraju se ispisuju vrednosti na displeju, proverava se glavni uslov, ako je ispunjen izvršava se finalno merenje prenosne karakteristike glavnog i pomoćnog EPAD-a. Ceo kod u C# je predstavljen u dodatku B.1.



Slika 3.3: Algoritam za EPAD programator treće generacije.

4. Eksperiment

Osmišljen je eksperiment za ispitivanje MOS trazistora sa plivajućim gejtom kao detektor jonizujućeg zračenja u posebnom režimu visoke osetljivosti po ugledu na dozimetar sa plivajućim gejtom koji je napravila iC Malaga. Pomoću komercijalne komponente ALD1108E koji se sastoji iz četiri EPAD-a na čipu i pomoćne elektronike sa softverskom kontrolom realizovan je eksperiment. Električna šema štampane ploče je data na slici 4.1. EPAD se sastoji iz dva integrisana NMOS tranzistora sa zajedničkim plivajućim gejtom koji se nazivaju glavni i pomoćni tranzistor (jer su pod tim nazivima otkriveni), pri čemu je uloga pomoćnog tranzistora punjenje plivajućeg gejta elektronima [10]. U ovom slučaju režima visoke osetljivosti, tranzistori će nositi nazive prateći (eng. monitoring) i programirajući (eng. charging) tranzistor jer glavni tranzistor ima ulogu praćenja promene napona praga (monitoring), a pomoćni ulogu punjenja plivajućeg gejta elektronima. Kako je potrebno pomoću SMU kanala mernog uređaja (Keithley 2636A) vršiti praćenje, a onda u određenom trenutnku preći na programiranje tranzistora, pored EPAD-a je potrebno postaviti leč (eng. latch) relej koji će svoje trenutno stanje zadržati tako da nije potrebno stalno opterećivati kalem polarizacijom, što povećava pouzdanost sistema kada je u pitanju ovakav način rada.

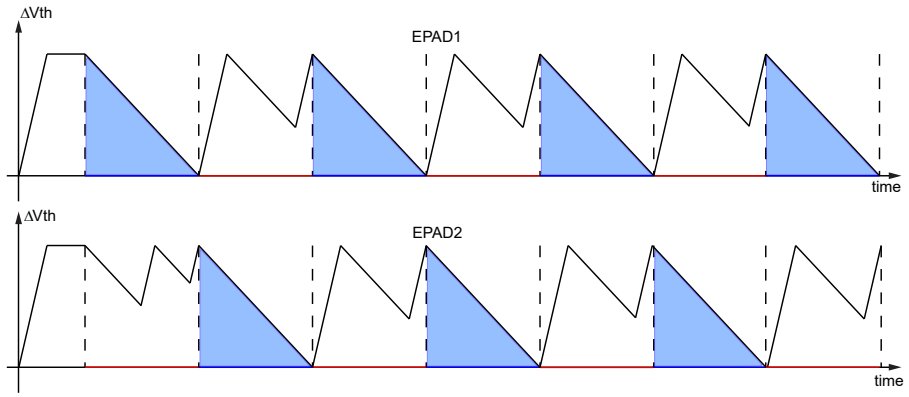


Slika 4.1: Električna šema ploče za ozračivanje sa integrisanim kolom ALD1108E i leč (eng. latch) relejima, kao i dodatno kolo za uključivanje releja.

Na električnoj šemi sa slike 4.1 može se videti motor drajver L293D koji kontroliše leč releje koji su povezani na čip ALD1108E tako da svaki leč relej kontroliše po dva EPAD-a

na čipu. Kada se sistem koristi za režim visoke osetljivosti koristi jedan EPAD (glavni i pomoćni tranzistor) za jedan leč relej. Ploča je dizajnirana tako da se mogu pratiti EPAD1 i EPAD2 kao upareni tranzistori ili EPAD3 i EPAD4. Pošto je korišćen SMU koji poseduje dva kanala, ako se posmatraju EPAD3 i EPAD4 potrebno je prevezati sa kanala 1 i 2 na kanal 3 i 4.

Dakle, dva EPAD-a će se naizmenično puniti i prazniti i time će se obezbediti stalno praćenje apsorbovane doze. Ciklusi programiranja i pražnjenja su prikazani na slici 4.2. Plavom bojom je obeleženo kada je dati EPAD aktivan, tj. kada radi u režimu praćenja apsorbovane doze (eng. monitoring). EPAD1 i EPAD2 su upareni i oni obezbeđuju stalno praćenje apsorbovane doze.



Slika 4.2: Ilustracija režima visoke osetljivosti za dva uparena EPAD-a na istom čipu.

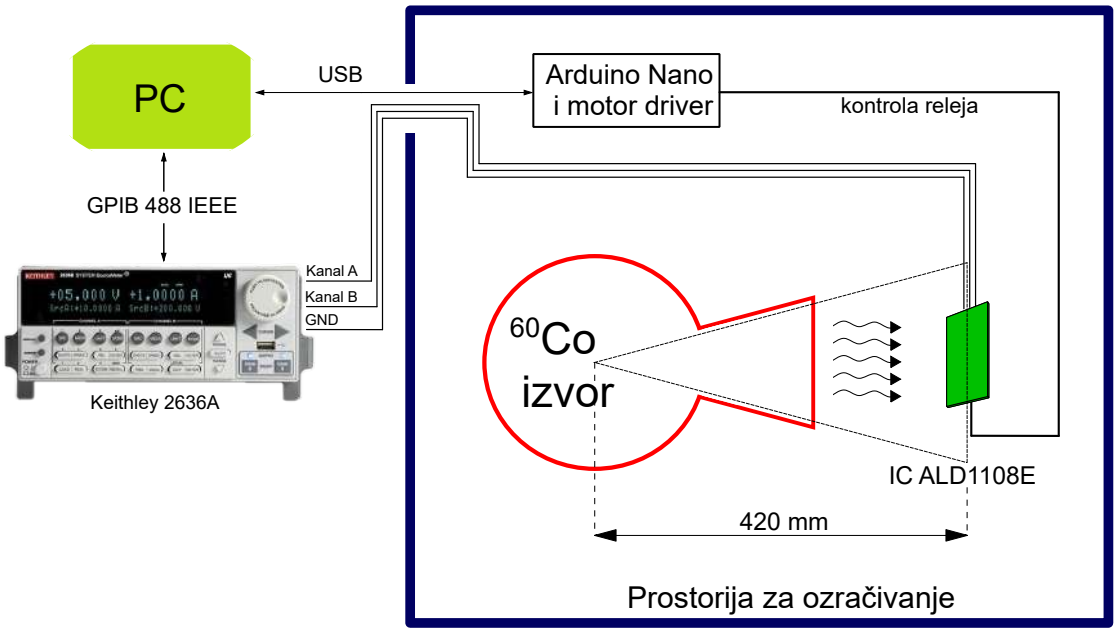
Na slici 4.2 uočavamo da su prvo EPAD-i napunjeni do iste vrednosti napona praga i kada se pojavi jonizujuće zračenje EPAD-i se identično prazne pošto imaju iste početne vrednosti napona praga i nalaze se na istom čipu (imaju identične tehnološke parametre) pa imaju iste karakteristike. Sa jednim EPAD-om se prati apsorbovana doza promenom napona praga, a drugi EPAD se programira tokom ozračivanja kako bi postigao početnu vrednost napona praga. Kada se prvi EPAD isprazni do određene vrednosti napona praga završava se ciklus i onda drugi EPAD uzima ulogu prvog, sada se pomoću njega prati apsorbovana doza, a prvi EPAD se programira tokom ciklusa do početne vrednosti napona praga. Kada se drugi EPAD isprazni do određene vrednosti napona praga zadate u softveru završava se ciklus i ponovo se menjaju uloge, sada se prvi EPAD koristi kao detekor zračenja, a drugi se programira i tako ciklično.

Utvrđeno je višesatnim testiranjima izvorom gama zračenja, da je potrebno prilagoditi struju i napon programiranja EPAD-a kako bi proces programiranja bio dominantniji u odnosu na ozračivanje. Zato treba uzeti opseg tako da vreme programiranja bude što manje kako bi sistem bio sposoban da brzo vrati napon praga na početnu vrednost ako su jačine doza veće ili ako je degradacija komponente postala značajna. Sledi postavka eksperimenta koji se odvijao na Institutu za nuklearne nauke "Vinča".

4.1 Postavka eksperimenta

Eksperimentalna postavka se sastoji iz štampane ploče na kojoj se nalazi jedan čip ALD1108E (sadrži četiri EPAD-a na čipu) i dva leč (eng. latch) releja, zatim pored ploče sa nalazi pomoćna ploča sa Arduino Nano jedinicom i motor drajverom koji kontrolišu releje na štampanoj ploči, dvokanalna forsirajuća-merna jedinica (eng. dual channel SMU)

Keithley 2636A koja svojim kanalima vrši merenje napona praga i programiranje EPAD-a na čipu i računara sa Windows aplikacijom napisanom u C#-u putem koje se kontroliše Kethley, Arduino Nano i beleže podaci u računar. Keithley i računar komuniciraju preko GPIB kartice, dok Ardunio Nano jedinica komunicirao preko USB-a sa računarom. Kod za Arduino Nano napisan u Arduino IDE razvojnom okruženju dat je u dodatku B.2. Na slici 4.3 je prikazana eksperimentalna postavka u sobi za ozračivanje u Laboratoriji za zaštitu od jonizujućeg zračenja na Institutu za nuklearne nauke „Vinča“.



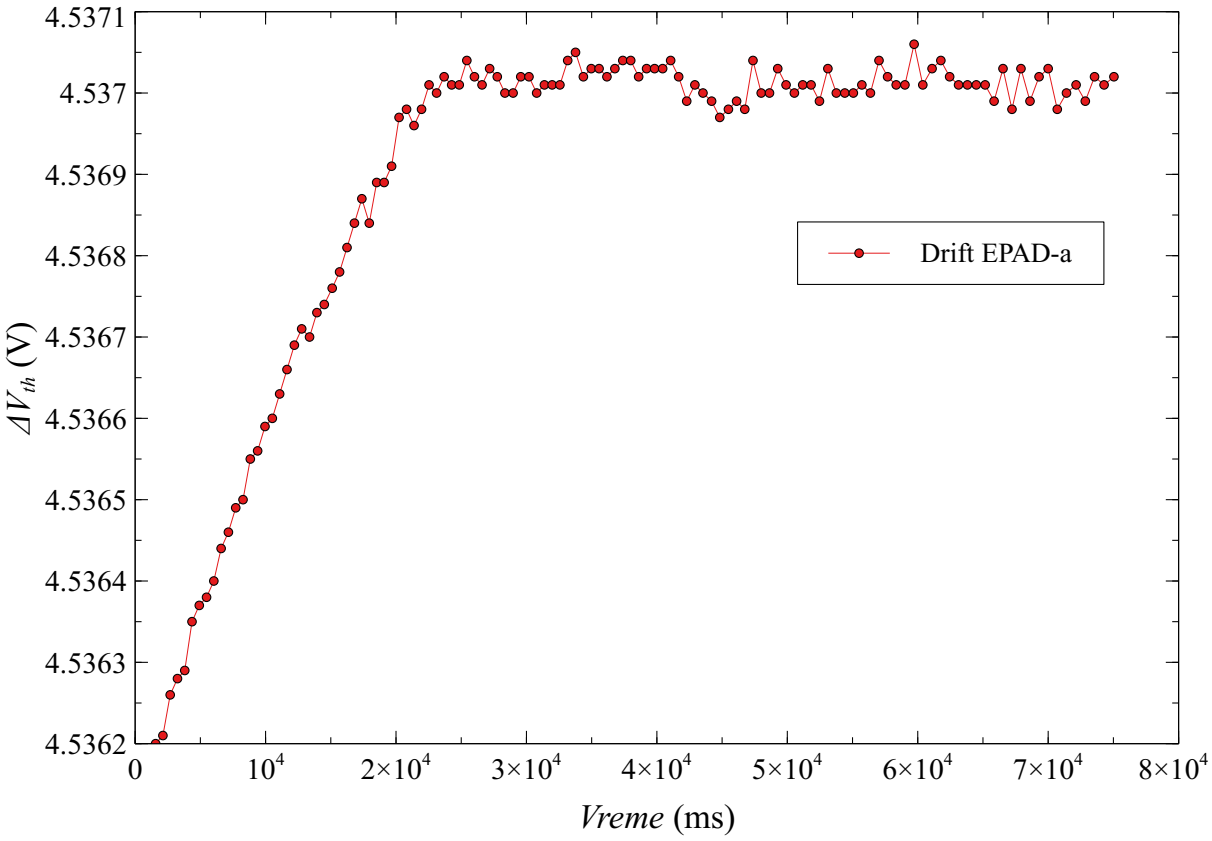
Slika 4.3: Postavka eksperimenta na Institutu za nuklearne nauke "Vinča".

Ceo eksperiment je automatizovan zahvaljujući dobro napisanom programu koji na osnovu zadatih graničnih uslova beleži cikluse režima visoke osetljivosti i menja uloge tranzistora (praćenje i programiranja) između dva EPAD-a na čipu. Neophodno je pre pokretanja programa zadati parametre programiranja: vreme, napon i struja programiranja. Takođe potrebno je zadati i opseg napona praga u kome sme da radi tranzistor. Ceo kod programa HIGHSENS napisan u C#-u dat je u dodatku B.3.

Ispred DUT-a je postavljen PMMA pleksiglas debljine 3 mm koji se montira na štampanu ploču pomoću odstojnika tako da kućište DUT-a prianja na pleksiglas (standard prilikom ozračivanja gama zracima), a pleksiglas se fiksira na usta kolimatora tako da se DUT nalazi u centru snopa radiaktivnog zračenja koje dolazi od izvora, gde se smatra da je polje homogeno. Ozračivanje je vršeno sa radioaktivnim izotopom kobalta ^{60}Co koji emituje gama fotone 1.17 MeV i 1.33 MeV energije, tj. srednja enerija fotona je 1.25 MeV. Komponente su se nalazile 420 mm od izvora. U polju zračenja koji čini snop visokoenergetskih fotona direktno je bila izložena samo štampana ploča sa DUT-om. Brzina doze pri ozračivanju je imala vrednost od 28.42 Gy/h u vodi, temperatura u prostoriji je bila 25 $°C$.

5. Rezultati i diskusija

MOS tranzistor po svojoj prirodi ima osobinu da prilikom stalnog proticanja struje kroz kanal oslobađa neka popunjena stanja blizu međupovršine kanala i oksida, ta stanja se nazivaju border trapovi [14]. Oslobađanjem zahvaćenih stanja povećava se struja drejna, sa samim tim i pad napona na tranzistoru, što dovodi do povećanja očitavanih vrednosti. Ako su u pitanju veoma precizna očitavanja kao što je to slučaj u dozimetrijskim aplikacijama, porast napona može dovesti do pogrešnog očitavanja doze. Promena napona u vremenu se naziva drift. Što je oksid u MOS tranzistoru deblji to je veći drift koji nastaje prilikom stalnog očitavanja napona, jer postoji veći broj zahvaćenih naelektrisanja i njihovo prostiranje je dublje u oksidu (gledano od međupovršine). Tranzistori sa plivajućim gejtom imaju tanak oksid što ukazuje da će imati mali drift. Na slici 5.1 prikazan je drift neozračenog EPAD-a.

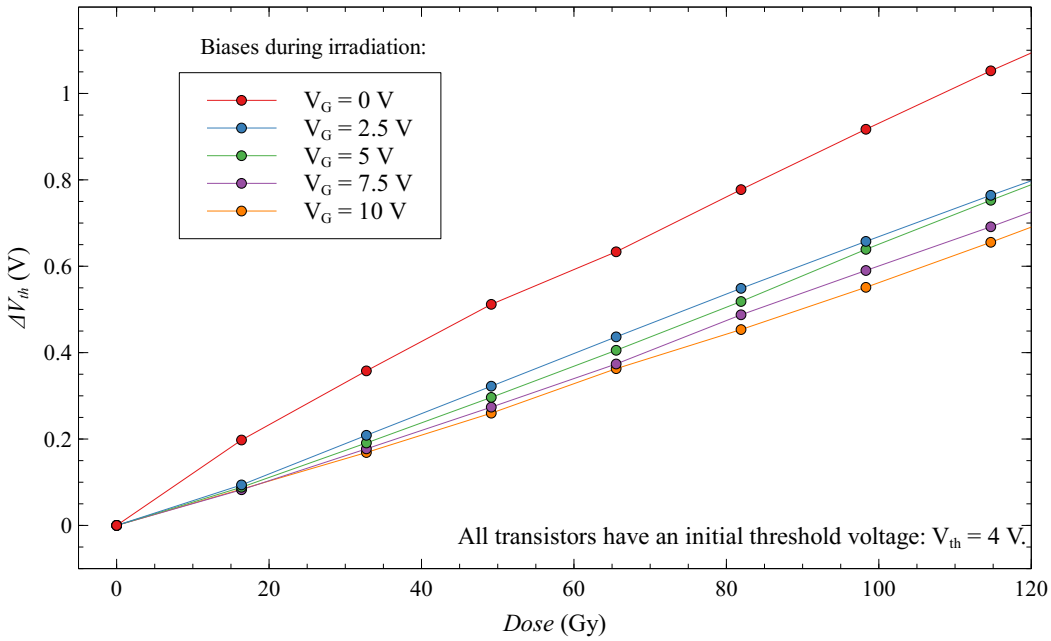


Slika 5.1: Promena napona (drift) EPAD-a prilikom stalnog forsiranja ZTC struje kroz kanal tranzistora.

Primećujemo da posle 25 sekundi dolazi do saturacije očitavanog napon. Veći broj merenja na različitim uzorcima pokazuju da drift EPAD-a traje oko 25 do 35 sekundi.

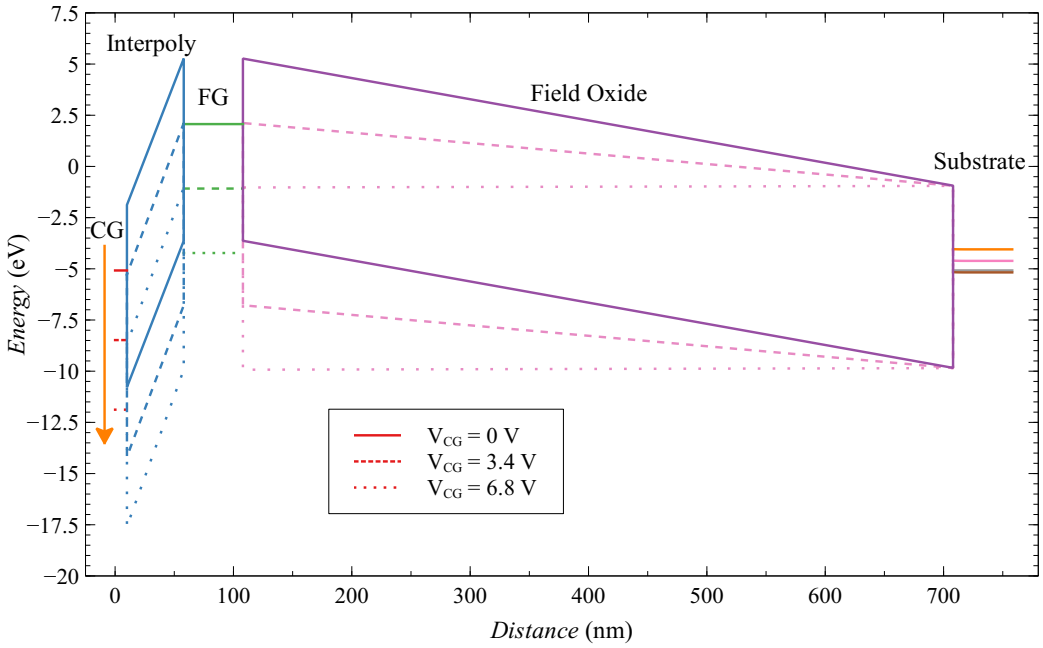
U diplomskog radu je zaključeno da EPAD-i sa nultom polarizacijom imaju veću osetljivost nego sa stalnom polarizacijom tokom ozračivanja kada je plivajući gejt znatno napunjen elektronima (tj. na početku eksperimenta) [10].

Analizirajmo grafik 2.5 koji smo ranije spomenuli u drugom poglavlju za prvih 120 Gy eksperimenta i predstavimo ga na novom grafiku 5.2 gde je na ordinati promena napona praga. Možemo uočiti da veći napon na kontrolnom gejtu daje manju osetljivost u prvom delu eksperimenta.



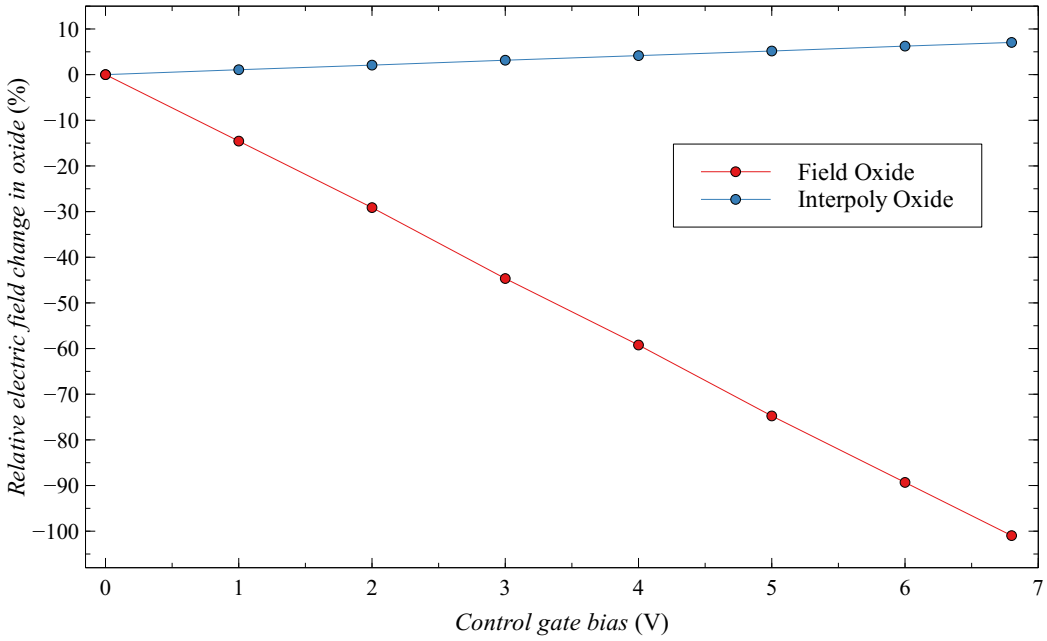
Slika 5.2: Interesantan fenomen veće osetljivosti sa nulto polarisanim EPAD-om.

Kako bi se objasnio ovaj veoma interesantan fenomen da nulto polarisan MOS tranzistor sa plivajućim gejtom ima veću osetljivost nego sa stalnom pozitivnom polarizacijom, moramo prvo da analiziramo ovaj problem u pogledu električnog polja u oksidu tranzistora tokom ozračivanja. Ovo možemo učiniti energetskim dijagramom MOS tranzistora sa plivajućim gejtom i za to je korišćen besplatan program Multi-Dielectric Energy Band Diagram koji je napravljen od strane Boise State University [15, 16]. Ključno je da odredimo koji deo EPAD strukture treba posmatrati. Velika oblast plivajućeg gejta se nalazi iznad debelog oksida (eng. field oxide) koji je zbog svoje debljina značajan za generisanje parova elektron-šupljina tokom ozračivanja. Ipak, pošto je EPAD komercijalna komponenta, nisu nam poznate debljine slojeva u poprečnom preseku MOS strukture. Pošto su ti podaci neophodni kao ulazne informacije za formiranje energetskog dijagrama MOS tranzistora sa plivajućim gejtom, iskoristili smo podatke veoma slične strukture koja je predstavljena od strane Tarr-a [17]. Takva struktura ima debeo oksid (eng. field oxide) ispod plivajućeg gejta debljine 600 nm, dok je oksid između kontrolnog i plivajućeg gejta, takozvani međuoksid (eng. interpoly oxide) debljine 48 nm. Pretpostavimo da je plivajući gejt debljine 50 nm, a da je kontrolni gejt 10 nm, oba gejta su napravljena od n+ polisilicijuma. Na slici 5.3 je prikazan energetski dijagram EPAD strukture [18].



Slika 5.3: Energetski dijagram EPAD strukture.

Kada povećamo napon na kontrolnom gejtu dolazi do smanjenja energije u MOS strukturi sa plivajućim gejtom, pri čemu se vrednosti električnih polja u međuoksidu i debelom oksidu menjaju. Pošto je oksid polja značajno deblji od međuoksida, promene u vrednostima električnog polja su drastičnije. Zato je napravljen grafik (Slika 5.4), gde se može videti relativna promena električnog polja u zavinosti od promene napona na kontrolnom gejtu.

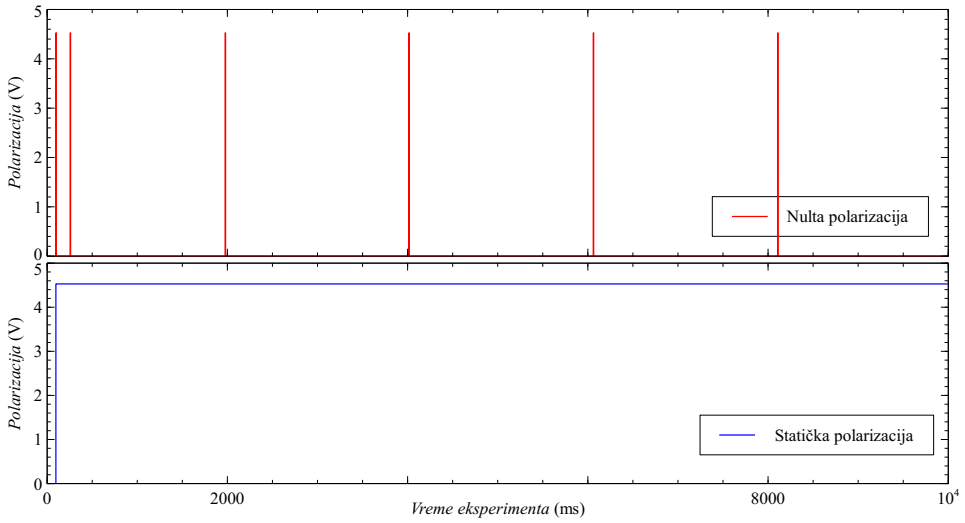


Slika 5.4: Električno polje u debelom oksidu i međuoksidu.

Jačina električnog polja u debelom oksidu opada za više od 100%, dok u međuoksidu jačina polja raste samo 7% za 6.8 V na kontrolnom gejtu. Možemo smatrati da promena

jačine električnog polja u međuoksidu je gotovo zanemarljiva u poređenju na promenu električnog pola u debelom oksidu. Ovo može objasniti zašto polarizacija kontrolnog gejta dovodi do smanjenja osetljivosti EPAD-a. Tokom ozračivanja plivajući gejt se prazni i njegova jačina električnog polja se smanjuje, pri čemu uticaj polja sa kontrolnog gejta postaje sve jači. Objašnjenje ove pojave će dosta značiti za rezultate eksperimenta u ovom master radu.

Eksperiment koji je realizovan, osmišljen je tako da se prate naponi na uparenim EPAD-ima koji imaju isti napon praga (npr. $V_{th} = 4\ V$) forsiranjem ZTC struje od 68 µA pri kojoj napon na gejtu iznosi u slučaju $V_{th} = 4\ V$ oko 4.5 V. Nulti temperaturni koeficijent (ZTC) je definisan za vrednost struje $I_D = 68$ µA [5]. Kako bi se u režimu visoke osetljivosti ostvarila što veća osetljivost potrebno je da EPAD bude nulto polarisan. Međutim, sam režim visoke osetljivosti zahteva stalno praćenje vrednosti napona na EPAD-ima koji su upareni. Iz tog razloga smo osmislili dve grupe eksperimenata, prva sa stalnom polarizacijom, a druga grupa sa nultom polarizacijom i povremenim očitavanjem. Polarizacije EPAD-a tokom ozračivanja su prikazane na slici 5.5.



Slika 5.5: Polarizacije EPAD-a u režimu visoke osetljivosti sa nultom i stalnom (statičkom) polarizacijom.

Naravno, zbog koncepta samog eksperimenta koji zahteva stalno praćenje napona praga tokom ozračivanja nije moguće uvek imati nultu polarizaciju. Možemo uočiti da se kod EPAD-a sa nultom polarizacijom očitavanje vrši na oko 2000 ms. Samo očitavanje traje oko 5 ms što je trenutni minimum koji se može postići datom opremom koju poseduje Laboratorija za primenjunju fiziku. Dakle, 0.25 % od vremena ukupnog eksperimenta nije nulta polarizacija na kontrolnom gejtu što se može zanemariti i smatrati da je nulta polarizacija tokom celog eksperimenta. Kod EPAD-a sa stalnom polarizacijom očitavanje traje na svakih 250 ms, dok je polarizacija konstantna tokom ozračivanja. Na osnovu prethodno objašnjenje pojave smanjenja osetljivosti zbog stalne polarizacije tokom ozračivanja EPAD-a, očekujemo veću osetljivost sa nultom polarizacijom nego sa stalnom.

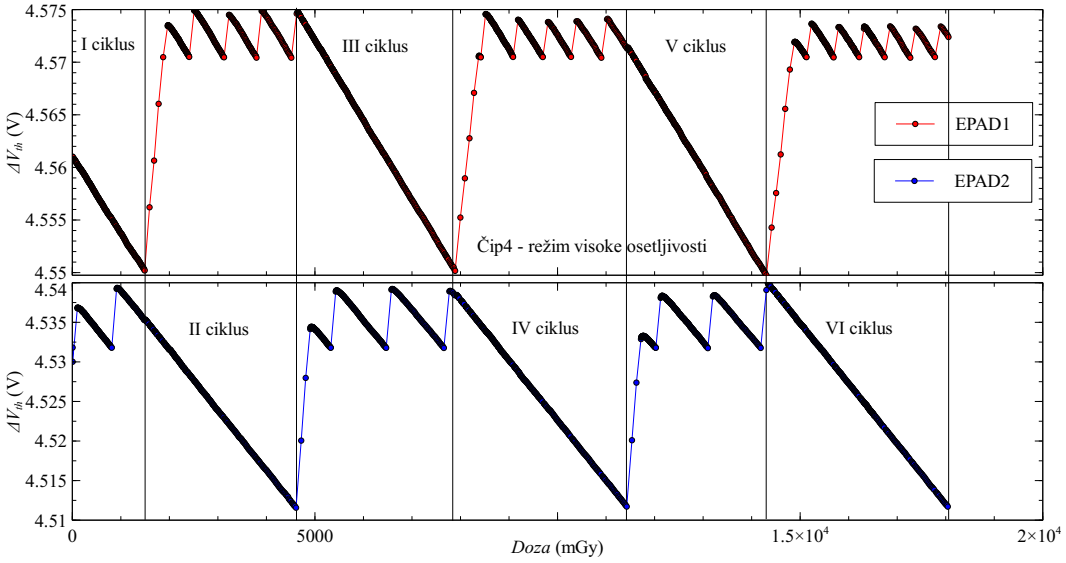
Obrada rezultata je vršena u programu Origin-u. Zbog velike kompleksnosti u rezultatima nije bilo moguće za relativno kratko vreme napisati program u Matlab-u koji bi obradio rezultate. Odlučili smo da je brže obraditi ih ručno i svaki od rezultata pojedinačno pogledati i analizirati. Posebno zato što ne znamo kakav rezultat očekujemo, jer ovakav eksperiment nikada nije realizovan do sada. Izvršena je određena filtracija loših merenja koja se javljaju kod EPAD-a sa kojim se prati apsorbovana doza zbog velike struje

injekcije tokom programiranja uparenog EPAD-a. Većina rezultata nije bilo upotrebljivo za prikazivanje u radu. Međutim, takvi rezultati su dosta značili u boljem razumevanju i planiranju eksperimenata.

Kao što je već rečeno, za ovaj eksperiment je napisan program u C#-u. Program ima određene parametre koje je potrebno definisati. Vreme, napon i struja programiranja su ključni parametri koje je potrebno podešavati kako bi programiranje tokom ozračivanja bilo dominatnije u odnosu na uticaj zračenja. Takođe, pored ovih parametara za režim visoke osetljivosti je potrebno i definisati opseg rada, što je učinjeno jednim parametrom koji predstavlja koeficijent manji od jedinice. Program uzima početni napon praga i množi sa koeficijentom koji pokazuje do koje vrednosti napona praga sme EPAD da se isprazni u toku ciklusa.

U ovom radu su predstavljena tri ozračena integrisana kola ALD1108E, jedan je ozračen pri stalnom polarizacijom (Čip 4) sa početnim naponom praga $V_{th} = 4\ V$, a ostala dva su ozračena pri nultom polarizacijom, pri čemu jedan od njih ima početni napon praga $V_{th} = 4\ V$ (Čip 10), a drugi $V_{th} = 4.5\ V$ (Čip 2).

Na slici 5.6 je prikazan grafik promene napona praga uparenih EPAD-a na čipu ALD1108E, koji u eksperimentu nosi oznaku četiri, pri stalnom polarizacijom tokom ozračivanja.



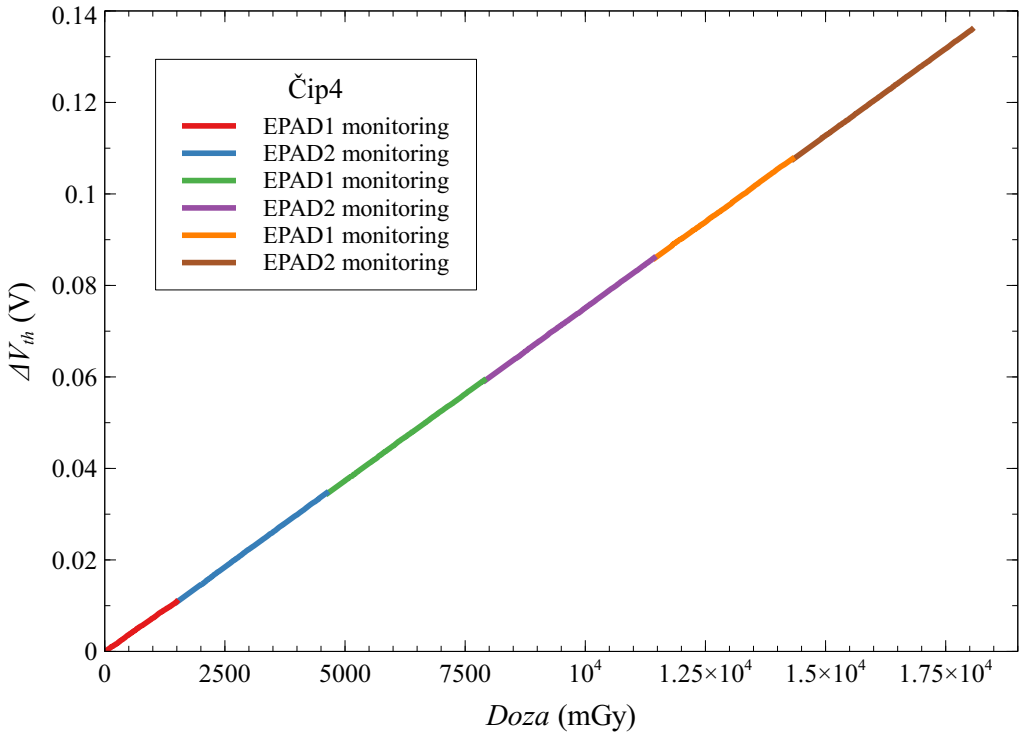
Slika 5.6: Čip 4 - Upareni EPAD1 i EPAD2 u režimu visoke osetljivosti pri stalnom polarizacijom tokom ozračivanja.

Uočavamo na ordinati opseg promene napona praga u kome se nalazi EPAD tokom eksperimenta. Ovaj opseg je definisan početnim naponom praga pre ozračivanja i koeficijentom koji se zadaje u programu. Vertikalnim linijama na grafiku su odvojeni ciklusi. Na ovoj slici je prikazano prvih šest ciklusa dok je ovaj čip napravio ukupno 76 ciklusa pre nego što je otkazao. Tokom ciklusa jedan EPAD služi za praćenje apsorbovane doze i njega nazivamo prateći tranzistor, dok se drugi EPAD programiranjem održava spreman da u sledećem ciklusu preuzme ulogu pratećeg tranzistora. Naziv ciklusa na slici 5.6 je postavljen kod EPAD-a koji je u tom ciklusu prateći tranzistor.

Proces programiranja u jednom ciklusu sastoji se iz pomeraja između dve uzastopne vrednosti napona praga (korak) tokom programiranja pre nego što se postigne željena vrednost napona praga EPAD-a. Korak programiranja direktno zavisi od vremena, na-

pona i struje programiranja. Ovi parametri se unose u program pre početka eksperimenta. Promena napona praga (korak) u vremenu se naziva brzina programiranja. Brzina se menja sa promenom jačine doze, i ako je jačina doze prevelika neće doći do pomeraja napona praga, tj. neće se vršiti programiranje. Ako se veća količina elektrona u plivajućem gejtu smanji zbog zračenja nego što se poveća zbog programiranja za isti vremenski interval, programiranje EPAD-a nije moguće postići. Dakle, brzina programiranja zavisi i od jačine doze, i zato je potrebno prilagoditi parametre programiranja jačini doze, tako da brzina programiranja ne bude ni prevelika ni premala. Ako je brzina programiranja prevelika, javiće se veliki premašaji u vrednostima napona praga, jer će jedan korak programiranja biti toliko veliki da će prebaciti definisan opseg rada EPAD-a u režimu visoke osetljivosti, što će dovesti do nesinhronizovanog rada uparenih EPAD-a. Ovaj efekat je više puta primećen tokom eksperimenata. Zato je jako bitno da se parametri programiranja dobro isprojektuju pre početka eksperimenta, a to je moguće samo uz prethnodno iskustveno znanje o ponašanju EPAD-a tokom ozračivanja.

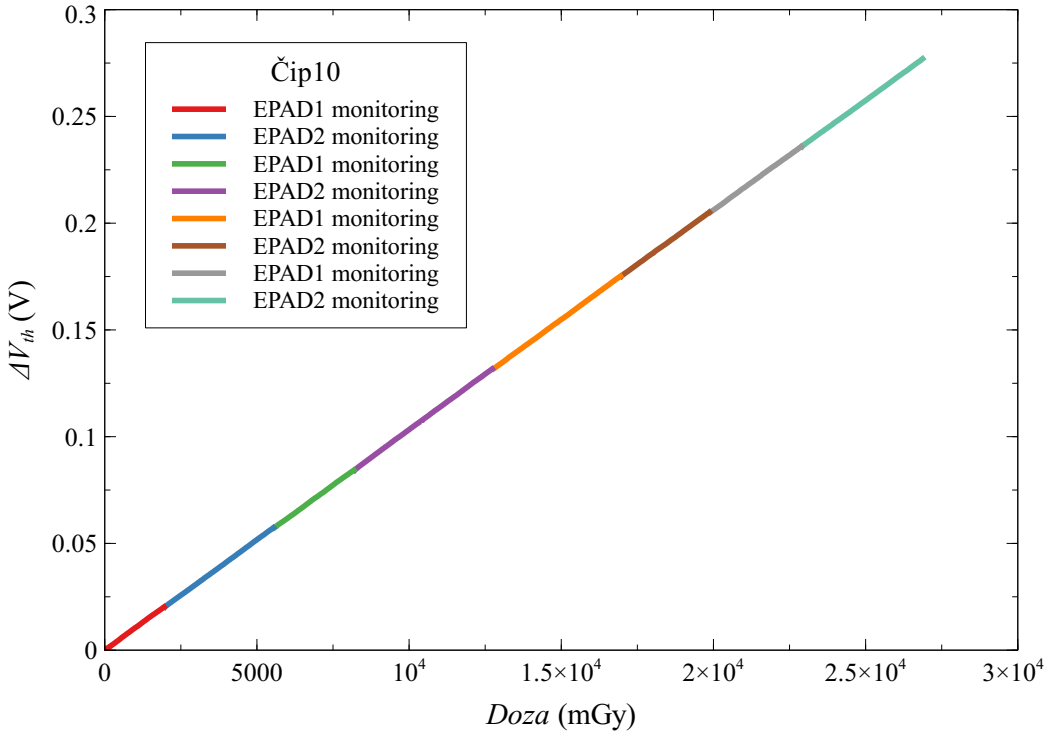
Na slici 5.7 je prikazana zavisnost promene napona praga pratećeg tranzistora od apsorbovane doze pri stalnom polarizacijom tokom ozračivanja i početnim naponom praga $V_{th} = 4\ V$ (Čip 4).



Slika 5.7: Čip 4 - Zavisnost promene napona praga od apsorbovane doze pratećeg tranzistora u režimu visoke osetljivosti pri stalnom polarizacijom i početnim naponom praga $V_{th} = 4\ V$.

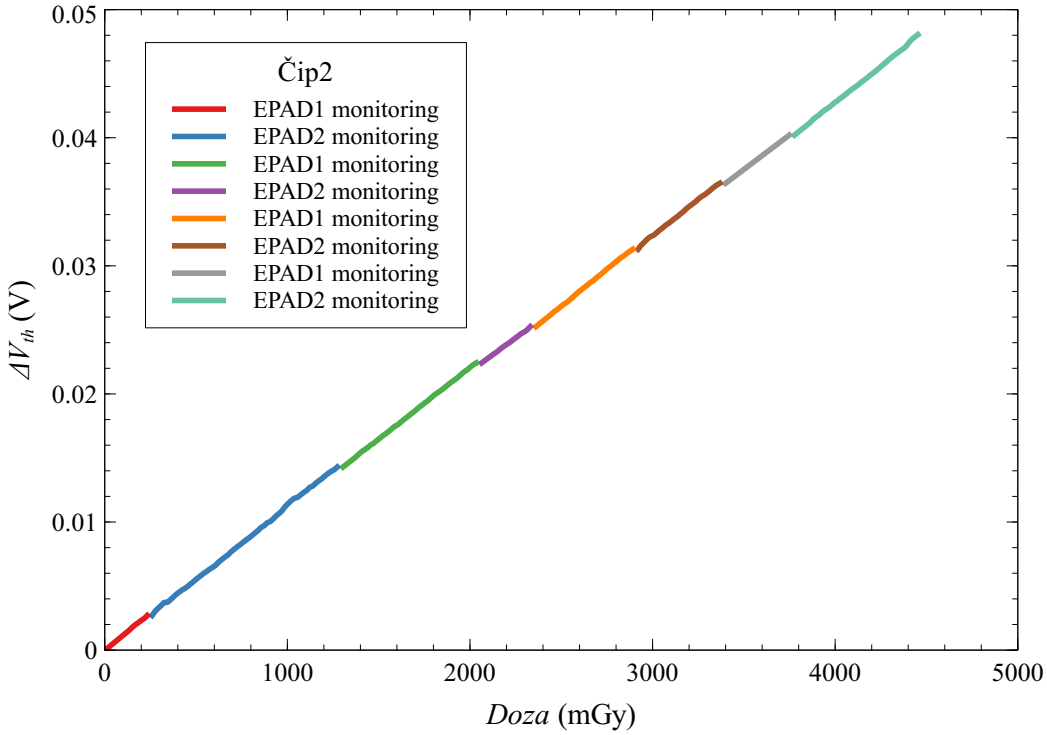
Grafik prikazan na slici 5.7 je dobijen na osnovu ciklusa pratećih tranzistora EPAD-a 1 i EPAD-a 2. Može se primetiti da je zavisnost linearna, što potvrđuje našu pretpostavku da se pomoću režima visoke osetljivosti postiže linearna zavinost promene napona praga od apsorbovane doze. Pošto je zavisnost linearna možemo da provučemo pravu i izračunamo osetljivost na celom opsegu. Osetljivost čipa 4 je $S = 7.535\ \frac{mV}{Gy}$, dok je osetljivost EPAD-a sa stalnom polarizacijom koji nisu u režimu visoke osetljivosti manja i ne prelazi vrednost od $S = 7\ \frac{mV}{Gy}$ [10].

Dva čipa koja su tokom ozračivanja imali nultu polarizaciju su čip 10 sa $V_{th} = 4\ V$ i čip 2 sa $V_{th} = 4.5\ V$ njihove zavisnosti promene napona praga pratećeg tranzistora od apsorbovane doze su prikazane na slikama 5.8 i 5.9.



Slika 5.8: Čip 10 - Zavisnost promene napona praga od apsorbovane doze pratećeg tranzistora u režimu visoke osetljivosti pri nutlom polarizacijom i početnim naponom praga $V_{th} = 4\ V$.

Može se uočiti da su zavisnosti promene napona praga od apsorbovane doze linearne kao i što je očekivano za režim visoke osetljivosti. Fitovane su prave za oba čipa i izračunate su osetljivosti. Čip 10 ima vrednost $S = 11.65\ \frac{mV}{Gy}$, dok čip 2 ima $S = 12.03\ \frac{mV}{Gy}$. Osetljivost ovih čipova je znatno veća od osetljivosti čipa koji je stalno polarisan tokom ozračivanja, što je očekivano na osnovu prethodnih rezultata [10]. Najveću osetljivost ima čip koji je nulto polarisan i ima najveći početni napon praga. Problem postoji u otkazivanju oba čipa nakon 9 ciklusa.



Slika 5.9: Čip 2 - Zavisnost promene napona praga od apsorbovane doze pratećeg tranzistora u režimu visoke osetljivosti pri nultom polarizacijom i početnim naponom praga $V_{th} = 4.5\ V$.

Tokom programiranja EPAD-a komponente su izložene stresu, a najveći stres se dešava u tunel oksidu kroz koji elektroni putuju iz kanala do plivajućeg gejta. Stres se značajno povećava kada se programiranje vrši tokom zračenja, što je slučaj za režim visoke osetljivosti, zato što je potrebno ostvariti veliku injekciju elektrona kroz tunel oksid u plivajući gejt. Uvek se tokom programiranja odvija degradacija tunel oksida zbog injekcije elektrona, ali je zbog ozračivanja potrebna znatno veća injekcija i samim tim nastaje i znatno veća degradacija strukture. To je razlog većeg otkaza tranzistora koji je primećen u eksperimentu.

Za ovaj eksperiment je ukupno upotrebljeno 15 čipova ALD1108E, od njih su najbolje rezultate koji su predstavljeni u ovom radu dala tri čipa. Nažalost, najveći problem režima visoke osetljivosti za EPAD-e je mali prinos od samo 20 %. Smatramo da se prinos može znatno povećati ako se tehnologija EPAD-a prilagodi potrebama poluprovodničkih dozimetra. Pre svega, pretpostavljamo da je najvažnije napraviti kvalitetniji i uniformniji tunel oksid, čija degradacija direktno utiče na životni vek EPAD-a kao detektora jonizućeg zračenja u režimu visoke osetljivosti.

6. Zaključak

Koncept režima visoke osetljivosti koji se koristi za dozimetre sa plivajućim gejtom je primenjen na EPAD-e. Korišćen je čip ALD1108E na kome se nalaze četiri MOS tranzistora sa plivajućim gejtom (EPAD-i). Za režim visoke osetljivosti korišćena su dva EPAD-a sa čipa. Rezultati su pokazali da je osetljivost povećana u poređenju sa ranijim eksperimentom sa EPAD-ima prikazanim u diplomskom radu. Takođe, postignuta je linearna zavisnost promene napona praga od apsorbovane doze. Najveću osetljivost koja iznosi $S = 12.03\ \frac{mV}{Gy}$ imaju upareni EPAD-i na čipu br. 2 koji su nulto polarisani tokom ozračivanja i imaju početni napon praga $V_{th} = 4.5\ V$. Nulto polarisan čip sa manjom vrednošću napona praga ($V_{th} = 4\ V$) ima manju osetljivost, odakle se može zaključiti da MOS tranzistor sa plivajućim gejtom treba da poseduje što veći početni napon praga tj. da plivajući gejt ima što veću količinu elektrona, jer time se postiže veća osetljivost.

Ranije je utvrđeno da uticaj jonizujućeg zračenja dovodi do degradacije oksida u MOS tranzistoru. Takođe prilikom punjenja plivajućeg gejta elektronima dolazi do oštećenja tunel oksida. Prilikom programiranja EPAD-a pomoću programatora vrednosti napona i struje programiranja su niži nego u režimu rada visoke osetljivosti EPAD-a. Veće vrednosti napona i struje programiranja dovode do većeg oštećenja tunel oksida. Zbog toga dolazi do otkaza EPAD-a nakon malog broja ciklusa što predstavlja ozbiljan problem koji je potrebno prevazići kako bi EPAD u režimu visoke osetljivosti postao komercijalni poluprovodnički dozimetar.

Režimom visoke osetljivosti su poboljšane određene dozimetrijske osobine EPAD-a, kao što su osetljivost i linearnost. Očigledno da su neophodne određene izmene u dizajnu EPAD-a. Prilikom programiranja su primećene razlike u kvalitetu tunel oksida od čipa do čipa, a čak i na istom čipu između tranzistora jer se vreme programiranja znatno razlikovalo. Svakako da jedan od problema predstavlja nedovoljni kapacitet plivajućeg gejta koji treba značajno uvećati i samim tim postići veću osetljivost EPAD-a.

MOS tranzistor sa plivajućim gejtom kao detektor jonizujućeg zračenja je relativno nova dozimetrijska oblast u kojoj se dosta vrše istraživanja. U ovom radu je pokazano da EPAD-i imaju veliki potencijal kao senzori jonizujućeg zračenja i da treba dalje nastaviti sa ovi istraživanjima.

Literatura

[1] R. Edgecock, J. Matheson, M. Weber, E. G. Villani, R. Bose, A. Khan, D. Smith, I. Adil-Smith, and A. Gabrielli, "Evaluation of commercial programmable floating gate devices as radiation dosimeters," *Journal of Instrumentation*, vol. 4, no. 02, p. P02002, 2009.

[2] G. Campardo, R. Micheloni, D. Novosel *et al.*, *VLSI-design of non-volatile memories*. Springer, 2005.

[3] A. Holmes-Siedle, "The space-charge dosimeter: General principles of a new method of radiation detection," *Nuclear Instruments and Methods*, vol. 121, no. 1, pp. 169 – 179, 1974.

[4] Advanced Linear Devices, Inc. [Online]. Available: http://www.aldinc.com/

[5] ALD, "QUAD/DUAL Electrically Programmable Analog Device," 2012. [Online]. Available: http://www.aldinc.com/pdf/ALD1110E.pdf

[6] iCMalaga, "Floating Gate Sensor." [Online]. Available: http://www.ic-malaga.com/servicios_rad_en.html

[7] R. Ferraro, S. Danzeca, M. Brucoli, A. Masi, M. Brugger, and L. Dilillo, "Design of a radiation tolerant system for total ionizing dose monitoring using floating gate and RadFET dosimeters," *Journal of Instrumentation*, vol. 12, no. 04, pp. C04 007–C04 007, apr 2017.

[8] M. Brucoli, S. Danzeca, M. Brugger, A. Masi, A. Pineda, J. Cesari, L. Dusseau, and F. Wrobel, "Floating Gate Dosimeter Suitability for Accelerator-Like Environments," *IEEE Transactions on Nuclear Science*, vol. 64, no. 8, pp. 2054–2060, 2017.

[9] M. Brucoli, J. Cesari, S. Danzeca, M. Brugger, A. Masi, A. Pineda, L. Dusseau, and F. Wrobel, "Investigation on Passive and Autonomous Mode Operation of Floating Gate Dosimeters," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1620–1627, 2019.

[10] S. Ilic, "Electrically programmable floating gate MOS transistor as a radiation detector," Sep. 2019, Diploma thesis.

[11] S. Ilic, A. Jevtic, S. Stankovic, and V. Davidovic, "Feasibility of Applying an Electrically Programmable Floating-Gate MOS Transistor in Radiation Dosimetry," in *Proceedings IEEE 31st International Conference On Microelectronics*. IEEE Serbia and Montenegro Section - ED/SSC Chapter, Sep. 2019, pp. 67 – 70.

[12] A. Jevtic, S. Ilic, V. Davidovic, Z. Prijic, and A. Prijic, "Characterization of electrically programmable floating gate MOS transistor," in *62nd Annual Meeting of ETRAN Society*, Palic, Serbia, 2018, pp. 304–307, in Serbian.

[13] A. Jevtic, S. Ilic, and V. Davidovic, "EPADs decapsulation and analysis," in *11th Student Project Conference "IEEESTEC 2018"*, Nis, Serbia, 2018, pp. 329–333, in Serbian.

[14] Z. Savic, B. Radjenovic, M. Pejovic, and N. Stojadinovic, "The contribution of border traps to the threshold voltage shift in pMOS dosimetric transistors," *IEEE Transactions on Nuclear Science*, vol. 42, no. 4, pp. 1445–1454, 1995.

[15] R. G. Southwick and W. B. Knowlton, "Stacked dual-oxide MOS energy band diagram visual representation program (IRW student paper)," *IEEE Transactions on Device and Materials Reliability*, vol. 6, no. 2, pp. 136–145, 2006.

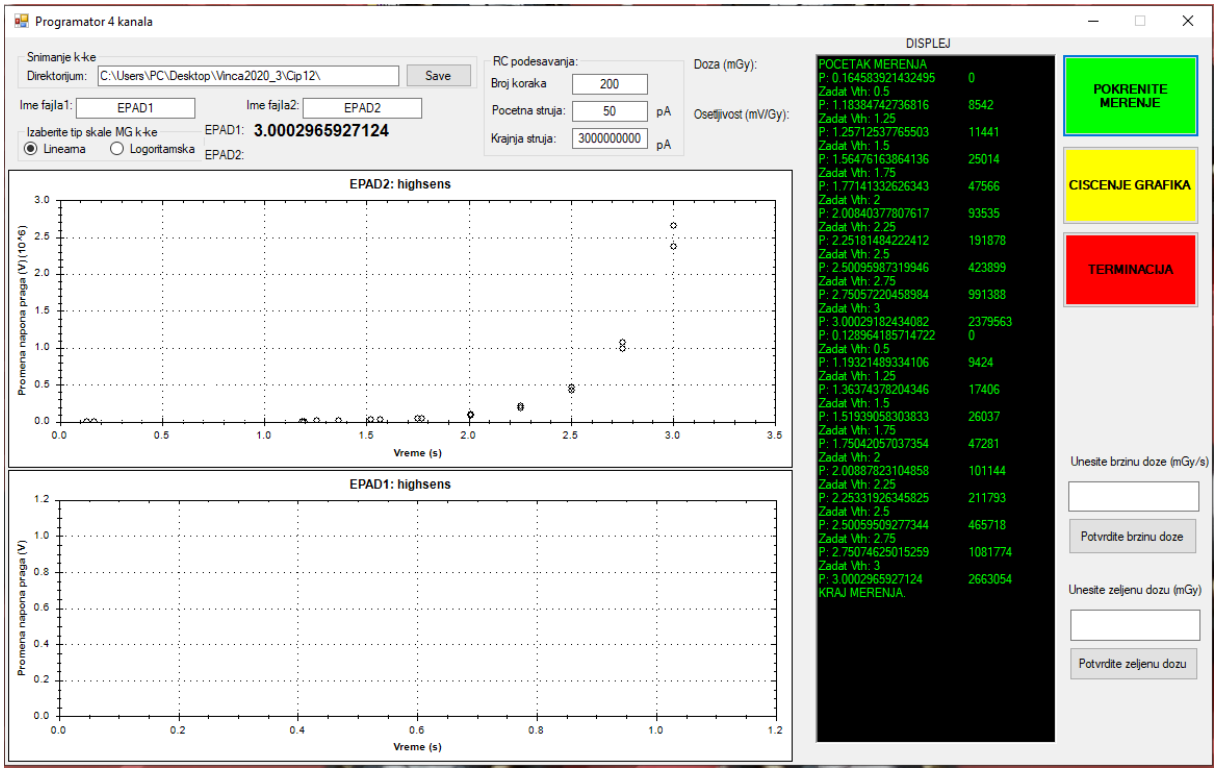
[16] R. G. Southwick, A. Sup, A. Jain, and W. B. Knowlton, "An interactive simulation tool for complex multilayer dielectric devices," *IEEE Transactions on Device and Materials Reliability*, vol. 11, no. 2, pp. 236–243, 2011.

[17] N. Garry Tarr, "Method of monitoring radiation using a floating gate field effect transistor dosimeter, and dosimeter for use therein," U.S. Patent US 6,172,368 B1, 2001.

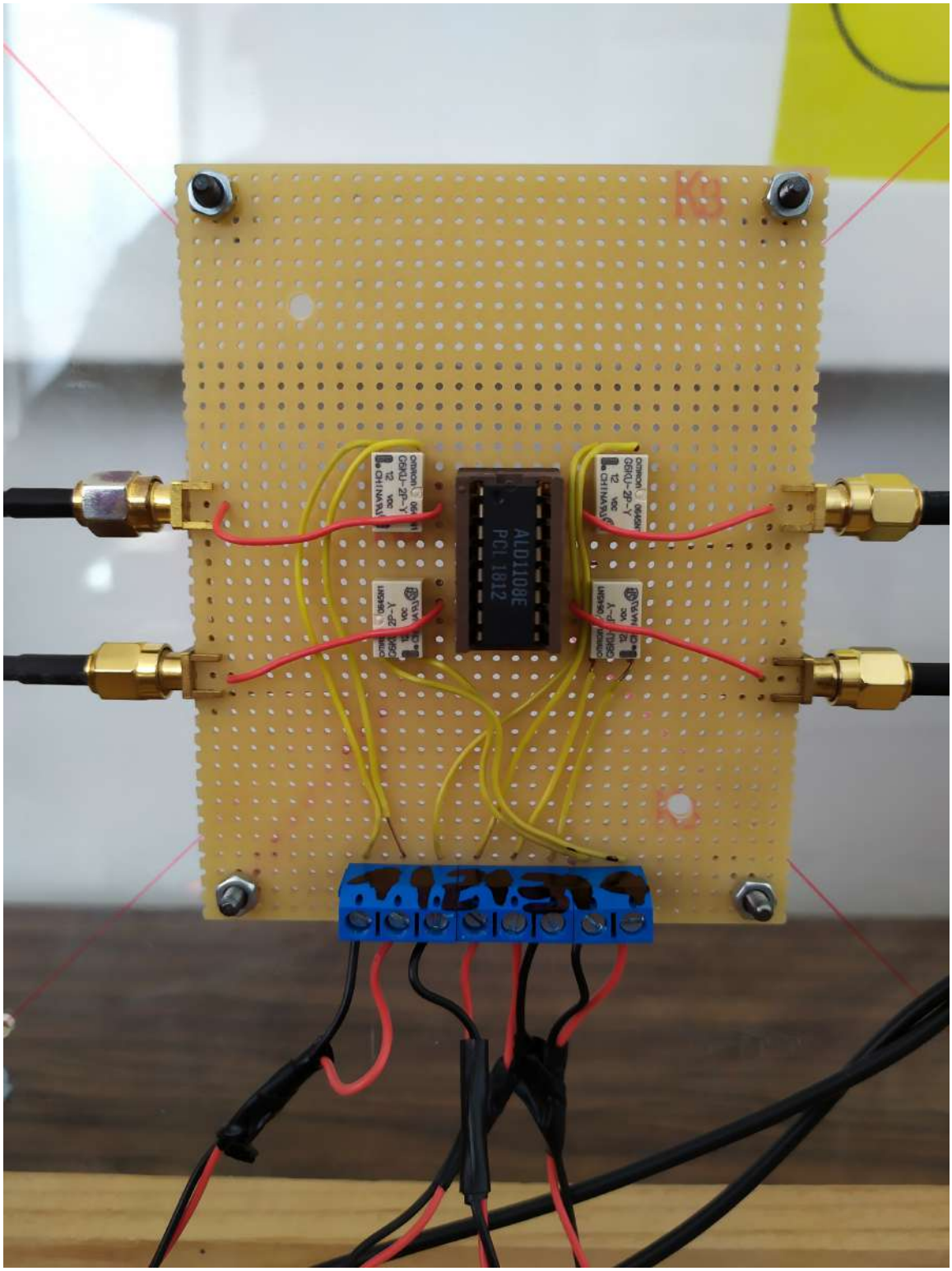
[18] S. Ilic, A. Jevtic, S. Stankovic, and G. Ristic, "Floating-gate mos transistor with dynamic biasing as a radiation sensor," *Sensors*, vol. 20, no. 11, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/11/3329

Dodaci

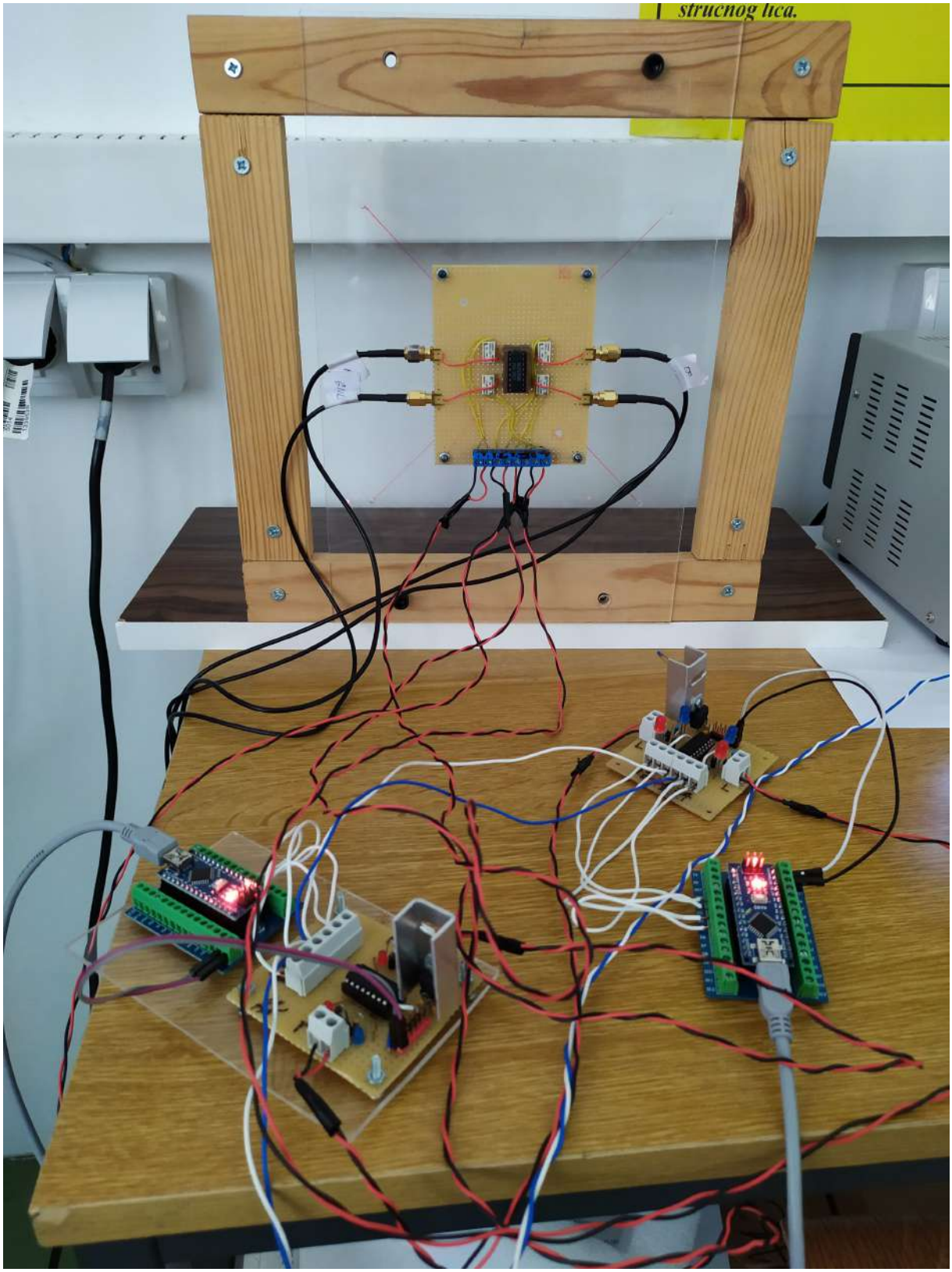
A. Dodatne fotografije



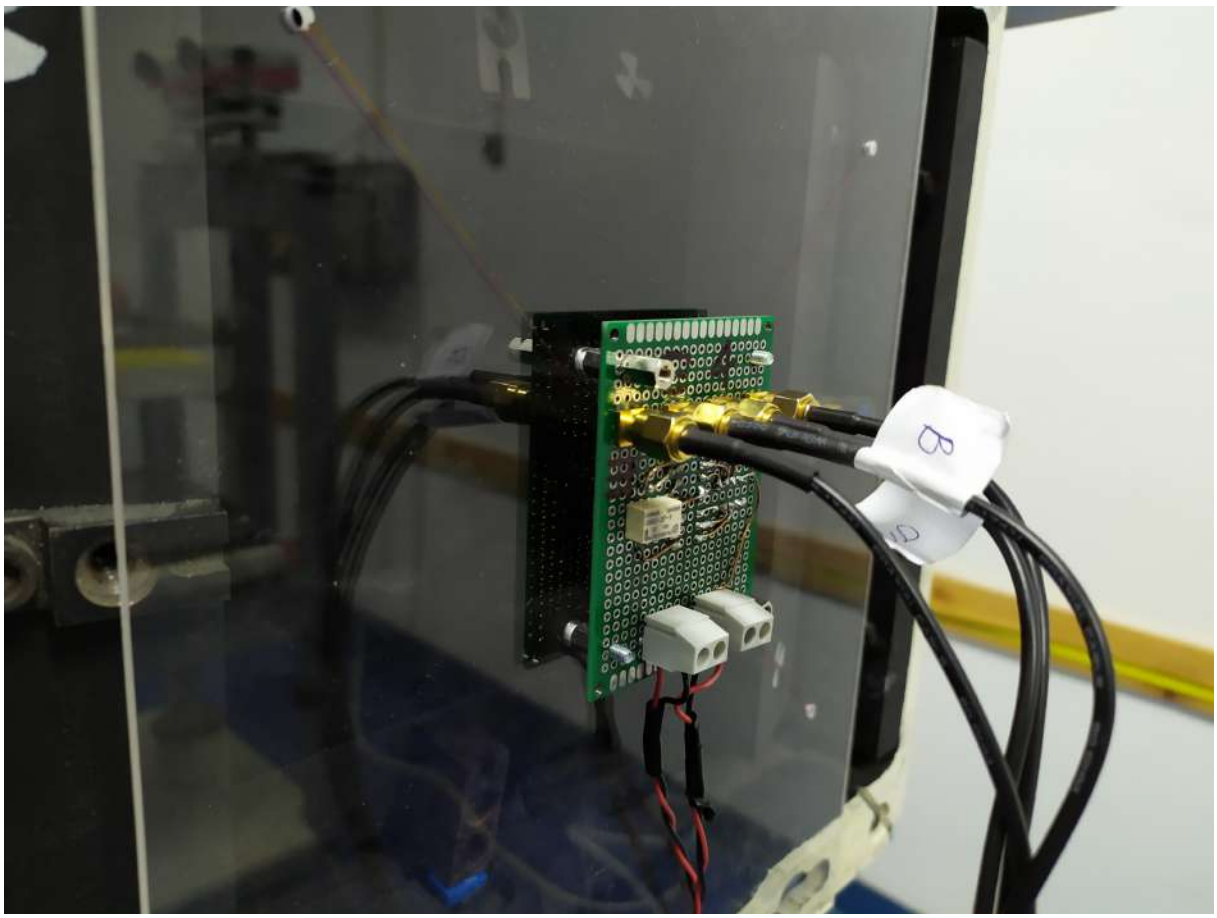
Slika A.1: Windows aplikacija EPAD programatora treće generacije nakon programiranja dva EPAD-a na čipu.



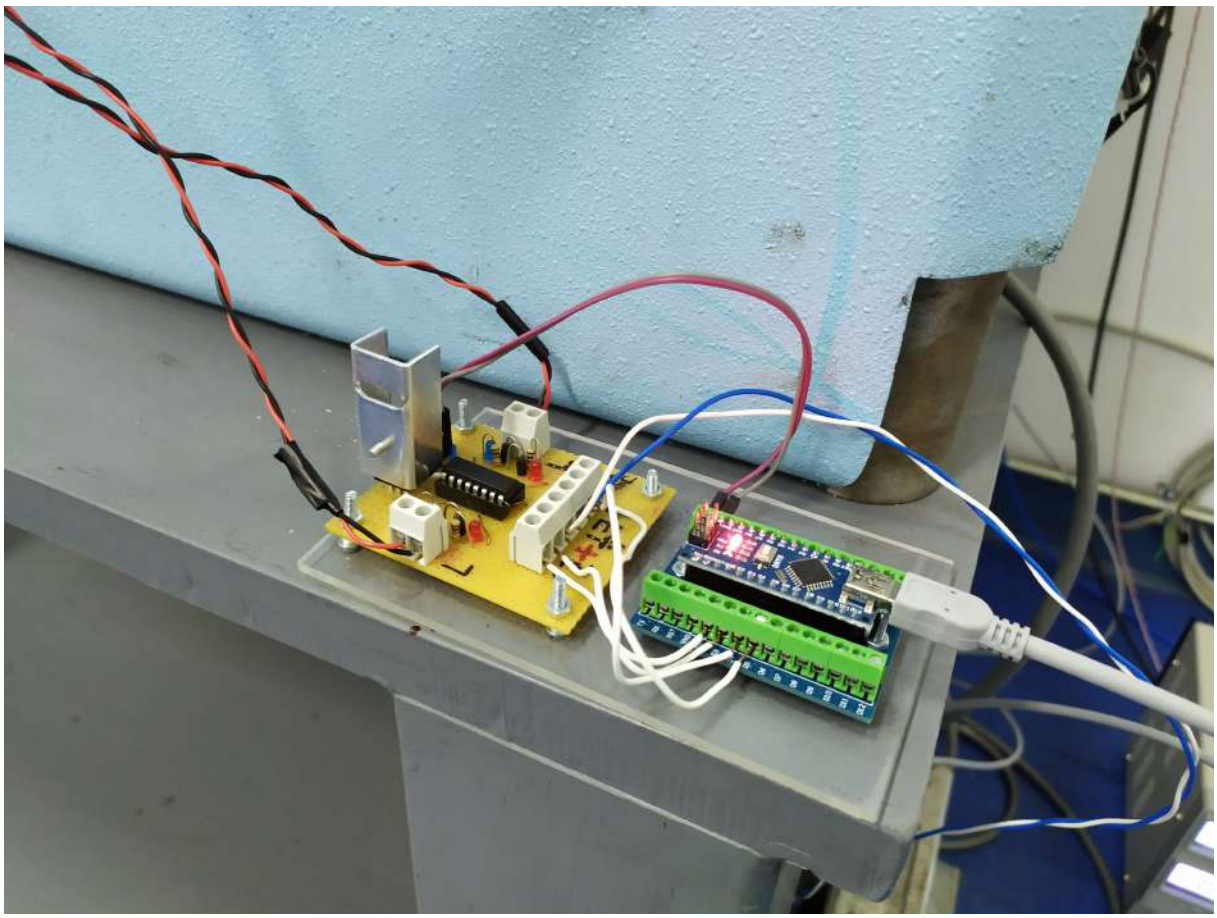
Slika A.2: Fotografija EPAD programatora treće generacije.



Slika A.3: Fotografija EPAD programatora treće generacije kontrolisanog pomoću dva motor drivera i dve Arduino Nano jedinice.



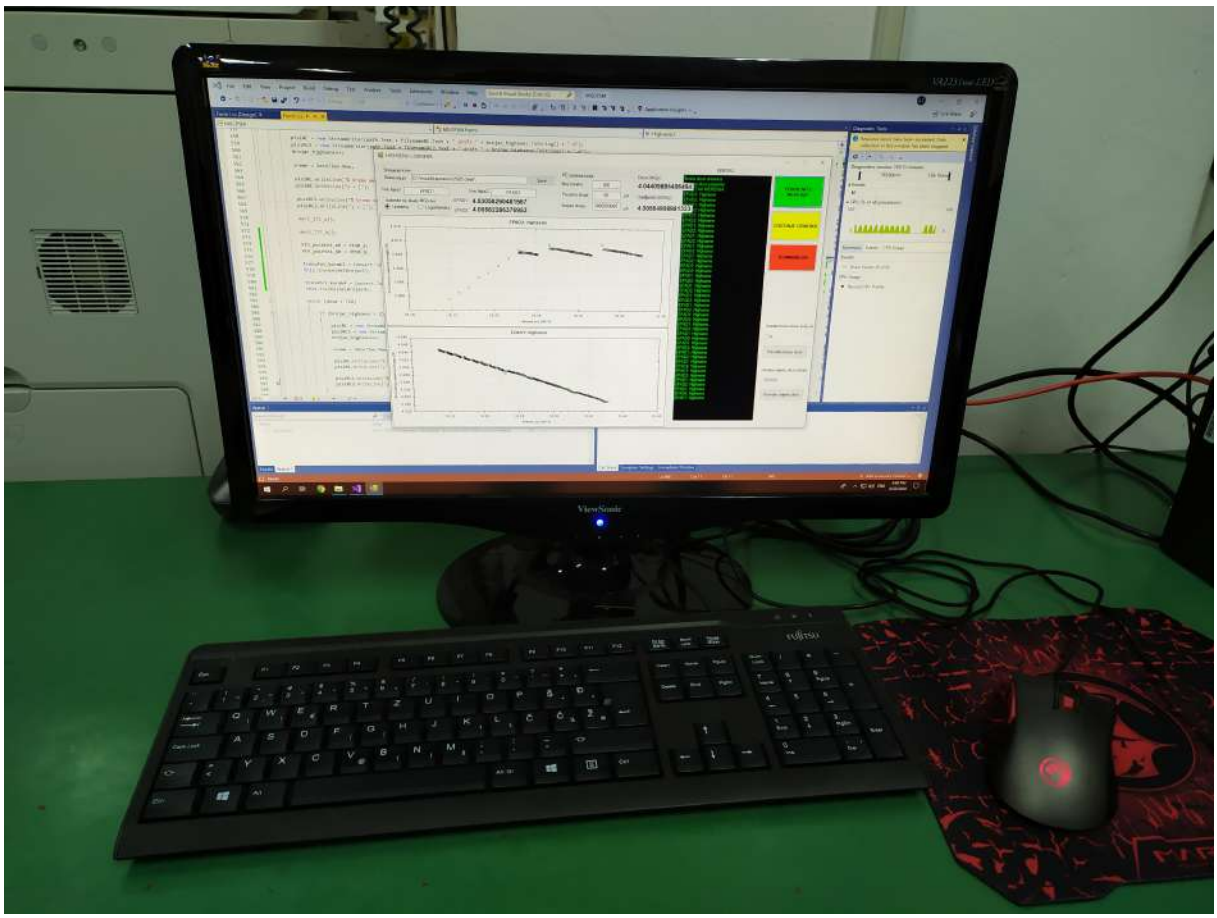
Slika A.4: Fotografija sistem za merenje EPAD-a u režimu visoke osetljivosti montiranog preko pleksiglasa na usta kolimatora u Laboratoriji za zaštitu od jonizujućeg zračenja na Institutu za nuklearne nauke „Vinča“.



Slika A.5: Fotografija motor drajvera i Arduino Nano jedinice u prostoriji za ozračivanje ispod usta kolimatora u Laboratoriji za zaštitu životne sredine od jonizujućeg zračenja, na Institutu za nuklearne nauke "Vinča".

Slika A.6: Fotografija linearnog izvora napajanja koji se nalazi u prostoriji za ozračivanje u Laboratoriji za zaštitu životne sredine od jonizujućeg zračenja, na Institutu za nuklearne nauke "Vinča".

Slika A.7: Fotografija prostorije za ozračivanje sa našom ekperimentalnom postavkom u Laboratoriji za zaštitu životne sredine od jonizujućeg zračenja, na Institutu za nuklearne nauke "Vinča".



Slika A.8: Fotografija računara sa programom "Highsens"napisanom u C#-u na kontrolnom punktu u Laboratoriji za zaštitu životne sredine od jonizujućeg zračenja, na Institutu za nuklearne nauke "Vinča".

B. Listing kodova

Listing B.1: Glavni deo koda za programator u C#-u

```
public void meri_epad_a()
        {
            device_2636.Write("smua.reset()");
            device_2636.Write("smua.sense = smua.SENSE_LOCAL");
            device_2636.Write("smua.source.func = smua.OUTPUT_DCAMPS");
            device_2636.Write("smua.source.autorangei = smua.AUTORANGE_ON");
            device_2636.Write("smua.source.leveli = 0");
            device_2636.Write("smua.measure.autorangev = smua.AUTORANGE_ON");
            device_2636.Write("smua.measure.nplc = 1"); //DODATAK BRZINA MERENJA
            device_2636.Write("smua.source.limitv = 10");
            device_2636.Write("smua.measure.v()");
            device_2636.Write("smua.source.output = smua.OUTPUT_ON");
            device_2636.Write("format.asciiprecision = 16"); //vraca merene
                vrednosti sa 16 cifara

            Vth_struja = 0.000001; // 1 uA
            ZTC_struja = 0.000068; // 68 uA

            if (choose == 1)
            {
                pisiRC = new StreamWriter(path.Text + filenameRC.Text +
                    "_Acharging_" + brojAcharging.ToString() + ".m"); //merenje
                    programirajuceg tranzistora
                brojAcharging++;
            }
            else
            {
                pisiRC = new StreamWriter(path.Text + filenameRC.Text +
                    "_Amonitoring_" + brojAmonitoring.ToString() + ".m");
                brojAmonitoring++;
            }

            vreme = DateTime.Now;

            pisiRC.WriteLine("% Vreme merenja : " + vreme);

            device_2636.Write("smua.source.leveli = " + Vth_struja.ToString());
            device_2636.Write("print(smua.measure.v())");
            naponn = device_2636.ReadString();
```

```
        pisiRC.WriteLine("Vth = [" + naponn + "];");

        device_2636.Write("smua.source.leveli = " + ZTC_struja.ToString());
        device_2636.Write("print(smua.measure.v())");
        naponn = device_2636.ReadString();
        pisiRC.WriteLine("ZTC = [" + naponn + "];");

        device_2636.Write("smua.source.leveli = 0"); //da vrati
            karakteristiku na pocetak

        pisiRC.WriteLine("y = [");

        pocetna_struja = System.Double.Parse(textBox3.Text) / 1000000000000;
        //pocetna_struja = 0.000001; //A prilikom merenja ZTC-a
        krajnja_struja = System.Double.Parse(textBox1.Text) / 1000000000000;
        if (choose == 1)
        {
            krajnja_struja = 0.001; //merenje programirajuceg tranzistora
        }
        //krajnja_struja = 0.000150; //A prilikom merenja ZTC-a
        broj_koraka = System.Double.Parse(textBox2.Text);

        konstanta = Math.Pow(10,
            (Math.Abs(Math.Log10(Math.Abs(krajnja_struja)) -
            Math.Log10(Math.Abs(pocetna_struja))) / (broj_koraka - 1)));

        for (brojac = 0; brojac <= broj_koraka - 1; brojac = brojac + 1)
        {
            device_2636.Write("smua.source.leveli = " + (pocetna_struja *
                Math.Pow(konstanta, brojac)).ToString());
            device_2636.Write("print(smua.measure.v())");
            naponn = device_2636.ReadString();
            pisiRC.Write((Math.Abs(pocetna_struja) * Math.Pow(konstanta,
                brojac)).ToString() + "\t" + naponn);
            listRC.Add(System.Double.Parse(naponn),
                (Math.Abs(pocetna_struja) * Math.Pow(konstanta, brojac)));
                // x y
        }

        pisiRC.WriteLine("];");
        pisiRC.WriteLine("");
        pisiRC.Close();

        device_2636.Write("smua.reset()");
        device_2636.Write("smua.source.output = smua.OUTPUT_OFF");
        device_2636.Write("smub.reset()");
        device_2636.Write("smub.source.output = smub.OUTPUT_OFF");
    }

    public void meri_epad_b()
    {
```

```
device_2636.Write("smub.reset()");
device_2636.Write("smub.sense = smub.SENSE_LOCAL");
device_2636.Write("smub.source.func = smub.OUTPUT_DCAMPS");
device_2636.Write("smub.source.autorangei = smub.AUTORANGE_ON");
device_2636.Write("smub.source.leveli = 0");
device_2636.Write("smub.measure.autorangev = smub.AUTORANGE_ON");
device_2636.Write("smub.measure.nplc = 1"); //DODATAK BRZINA MERENJA
device_2636.Write("smub.source.limitv = 10");
device_2636.Write("smub.measure.v()");
device_2636.Write("smub.source.output = smub.OUTPUT_ON");
device_2636.Write("format.asciiprecision = 16"); //vraca merene
    vrednosti sa 16 cifara

Vth_struja = 0.000001; // 1 uA
ZTC_struja = 0.000068; // 68 uA

if (choose == 1)
{
    pisiRC = new StreamWriter(path.Text + filenameRC2.Text +
        "_Bcharging_" + brojBcharging.ToString() + ".m"); //merenje
        programirajuceg tranzistora
    brojBcharging++;
}
else
{
    pisiRC = new StreamWriter(path.Text + filenameRC2.Text +
        "_Bmonitoring_" + brojBmonitoring.ToString() + ".m");
    brojBmonitoring++;
}

vreme = DateTime.Now;

pisiRC.WriteLine("% Vreme merenja : " + vreme);

device_2636.Write("smub.source.leveli = " + Vth_struja.ToString());
device_2636.Write("print(smub.measure.v())");
naponn = device_2636.ReadString();
pisiRC.WriteLine("Vth = [" + naponn + "];");

device_2636.Write("smub.source.leveli = " + ZTC_struja.ToString());
device_2636.Write("print(smub.measure.v())");
naponn = device_2636.ReadString();
pisiRC.WriteLine("ZTC = [" + naponn + "];");

device_2636.Write("smub.source.leveli = 0"); //da vrati
    karakteristiku na pocetak

pisiRC.WriteLine("y = [");

pocetna_struja = System.Double.Parse(textBox3.Text) / 1000000000000;
//pocetna_struja = 0.000001; //A prilikom merenja ZTC-a
```

```
        krajnja_struja = System.Double.Parse(textBox1.Text) / 1000000000000;
        if (choose == 1)
        {
            krajnja_struja = 0.001; //merenje programirajuceg tranzistora
        }
        //krajnja_struja = 0.000150; //A prilikom merenja ZTC-a
        broj_koraka = System.Double.Parse(textBox2.Text);

        konstanta = Math.Pow(10,
            (Math.Abs(Math.Log10(Math.Abs(krajnja_struja)) -
            Math.Log10(Math.Abs(pocetna_struja))) / (broj_koraka - 1)));

        for (brojac = 0; brojac <= broj_koraka - 1; brojac = brojac + 1)
        {
            device_2636.Write("smub.source.leveli = " + (pocetna_struja *
                Math.Pow(konstanta, brojac)).ToString());
            device_2636.Write("print(smub.measure.v())");
            naponn = device_2636.ReadString();
            pisiRC.Write((Math.Abs(pocetna_struja) * Math.Pow(konstanta,
                brojac)).ToString() + "\t" + naponn);
            listRC.Add(System.Double.Parse(naponn),
                (Math.Abs(pocetna_struja) * Math.Pow(konstanta, brojac)));
                // x y
        }

        pisiRC.WriteLine("];");
        pisiRC.WriteLine("");
        pisiRC.Close();

        device_2636.Write("smub.reset()");
        device_2636.Write("smub.source.output = smub.OUTPUT_OFF");
        device_2636.Write("smua.reset()");
        device_2636.Write("smua.source.output = smua.OUTPUT_OFF");
    }

    public void meri_Vth_a()
    {
        Delegate delBrojac = new delBrojKoraka(TrenutnoBrojKoraka);
        device_2636.Write("smua.reset()");
        device_2636.Write("smua.sense = smua.SENSE_LOCAL");
        device_2636.Write("smua.source.func = smua.OUTPUT_DCAMPS");
        device_2636.Write("smua.source.autorangei = smua.AUTORANGE_ON");
        device_2636.Write("smua.source.leveli = 0");
        device_2636.Write("smua.measure.autorangev = smua.AUTORANGE_ON");
        device_2636.Write("smua.measure.nplc = 1"); //DODATAK BRZINA MERENJA
        device_2636.Write("smua.source.limitv = 10");
        device_2636.Write("smua.measure.v()");
        device_2636.Write("smua.source.output = smua.OUTPUT_ON");
        //device_2636.Write("format.asciiprecision = 16"); //vraca merene
            vrednosti sa 16 cifara
```

```
    Vth_struja = 0.000001; // 1 uA

    device_2636.Write("smua.source.leveli = " + Vth_struja.ToString());
    device_2636.Write("print(smua.measure.v())");
    naponn = device_2636.ReadString();
    //pisiVth.WriteLine("Vth0 = [" + naponn + "];"); //moze se
        koristiti kao provera

    Vth0 = Convert.ToDouble(naponn);
    Vth0 = Vth0 - 1; // prebacivanje u delta Vth0

    device_2636.Write("smua.source.output = smua.OUTPUT_OFF");

    trenutni_korak = Convert.ToString(Vth0);
    this.Invoke(delBrojac);
}

public void meri_Vth_b()
{
    Delegate delBrojac = new delBrojKoraka(TrenutnoBrojKoraka);
    device_2636.Write("smub.reset()");
    device_2636.Write("smub.sense = smub.SENSE_LOCAL");
    device_2636.Write("smub.source.func = smub.OUTPUT_DCAMPS");
    device_2636.Write("smub.source.autorangei = smub.AUTORANGE_ON");
    device_2636.Write("smub.source.leveli = 0");
    device_2636.Write("smub.measure.autorangev = smub.AUTORANGE_ON");
    device_2636.Write("smub.measure.nplc = 1"); //DODATAK BRZINA MERENJA
    device_2636.Write("smub.source.limitv = 10");
    device_2636.Write("smub.measure.v()");
    device_2636.Write("smub.source.output = smub.OUTPUT_ON");
    //device_2636.Write("format.asciiprecision = 16"); //vraca merene
        vrednosti sa 16 cifara

    Vth_struja = 0.000001; // 1 uA

    device_2636.Write("smub.source.leveli = " + Vth_struja.ToString());
    device_2636.Write("print(smub.measure.v())");
    naponn = device_2636.ReadString();
    //pisiVth.WriteLine("Vth0 = [" + naponn + "];"); //moze se
        koristiti kao provera

    Vth0 = Convert.ToDouble(naponn);
    Vth0 = Vth0 - 1; // prebacivanje u delta Vth0

    device_2636.Write("smub.source.output = smub.OUTPUT_OFF");

    trenutni_korak = Convert.ToString(Vth0);
    this.Invoke(delBrojac);
}

public void programiranj_epad_a()
```

```
{
    ZadatiNaponPrag = ZadatiNaponPrag - 1; //prebacivanje u Delta Vth
    Delegate delIspisProg = new delIspisMerenja(obavestiProgramiranje);
    Delegate delIspisZadatNapon = new
        delIspisZadatNapon(obavestiZadatNapon);
    pisiVth = new StreamWriter(path.Text + filenameRC.Text +
        "_AprogEPAD1_" + iteratorZaMerenja.ToString() + ".m");
    vreme = DateTime.Now;
    pisiVth.WriteLine("% Vreme merenja : " + vreme);
    pisiVth.WriteLine("y = [");
    suma = 0;
    usporivac = 1;
    ZastavicaPrva = 0;

    serialPort1.Write("B"); //Spaja G i D EPAD-a 1
    Thread.Sleep(350);
    meri_Vth_a();
    pisiVth.WriteLine(Vth0 + "\t" + suma);
    listPrag.Add(Vth0, suma);       // x y
    this.Invoke(delIspisProg);

    while (Vth0 <= ZadatiNaponPrag)
    {
        int i = 0;

        serialPort1.Write("B"); //Spaja G i D EPAD-a 1
        Thread.Sleep(350);
        meri_Vth_a();

        do
        {
            Vth = niz[i];
            i++;
        } while (Vth < Vth0);

        this.Invoke(delIspisZadatNapon);

        Zastavica = 0;
        do
        {
            t1 = 220.511575393758 * Vth0 * Math.Exp(2.48798745664686 *
                Vth0) + Vth0 * Math.Exp(4.25896818690299 * Vth0) +
                24469.8926918828 * Vth0 + 2075.73597614034;
            t2 = 220.511575393758 * Vth * Math.Exp(2.48798745664686 *
                Vth) + Vth * Math.Exp(4.25896818690299 * Vth) +
                24469.8926918828 * Vth + 2075.73597614034;
            Dt = t2 - t1;
            Dt = Dt * usporivac; //usporava po potrebi
            a = Convert.ToInt32(Dt);

            if (a < 5000 && Zastavica > 2)
```

```
    {
        a = 5000;
    }

    suma = suma + a;

    serialPort1.Write("A"); //Spaja P i G EPAD-a 1
    Thread.Sleep(350);

    device_2636.Write("smua.reset()");
    device_2636.Write("smua.sense = smua.SENSE_LOCAL");
    device_2636.Write("smua.source.func = smua.OUTPUT_DCVOLTS");
    device_2636.Write("smua.source.rangev = 30");
    device_2636.Write("smua.source.levelv = 12.5");
    device_2636.Write("smua.source.limiti = 0.00325");
    device_2636.Write("smua.measure.autorangei =
        smua.AUTORANGE_ON");
    device_2636.Write("smua.measure.i()");
    device_2636.Write("smua.source.output = smua.OUTPUT_ON");

    Thread.Sleep(a); // vreme programiranja u ms

    device_2636.Write("smua.source.output = smua.OUTPUT_OFF");

    serialPort1.Write("B"); //Spaja G i D EPAD-a 1
    Thread.Sleep(350);
    meri_Vth_a();

    Zastavica++;
} while (Vth0 <= Vth);

if (Zastavica == 1 && Vth > 1)
{
    usporivac = 1 - (Vth0 / Vth - 1) * 6;

    if (ZastavicaPrva == 0)
    {
        usporivac_stari = usporivac; //prvi put prg premasi
            vrednost
        ZastavicaPrva = 1;
    }

    if (usporivac_stari < usporivac)
    {
        usporivac = usporivac_stari;
    }
}

pisiVth.WriteLine(Vth0 + "\t" + suma);
listPrag.Add(Vth0, suma);       // x y
this.Invoke(delIspisProg);
```

```
    }

    pisiVth.WriteLine("];");
    pisiVth.WriteLine("");
    pisiVth.Close();

    //premeravanje prenosnih karakteristika glavnog i pomocnog EPAD-a
    serialPort1.Write("B"); //Spaja G i D EPAD-a 1
    Thread.Sleep(350);
    choose = 0;
    meri_epad_a();

    serialPort1.Write("A"); //Spaja P i G EPAD-a 1
    Thread.Sleep(350);
    choose = 1;
    meri_epad_a();
}

public void programiranj_epad_b()
{
    ZadatiNaponPrag = ZadatiNaponPrag - 1; //prebacivanje u Delta Vth
    Delegate delIspisProg = new delIspisMerenja(obavestiProgramiranje);
    Delegate delIspisZadatNapon = new
        delIspisZadatNapon(obavestiZadatNapon);
    pisiVth = new StreamWriter(path.Text + filenameRC2.Text +
        "_BprogEPAD2_" + iteratorZaMerenja.ToString() + ".m");
    vreme = DateTime.Now;
    pisiVth.WriteLine("% Vreme merenja : " + vreme);
    pisiVth.WriteLine("y = [");
    suma = 0;
    usporivac = 1;
    ZastavicaPrva = 0;

    serialPort1.Write("D"); //Spaja G i D EPAD-a 2
    Thread.Sleep(350);
    meri_Vth_b();
    pisiVth.WriteLine(Vth0 + "\t" + suma);
    listPrag.Add(Vth0, suma);       // x y
    this.Invoke(delIspisProg);

    while (Vth0 <= ZadatiNaponPrag)
    {
        int i = 0;

        serialPort1.Write("D"); //Spaja G i D EPAD-a 2
        Thread.Sleep(350);
        meri_Vth_b();

        do
        {
```

```
    Vth = niz[i];
    i++;
} while (Vth < Vth0);

this.Invoke(delIspisZadatNapon);

Zastavica = 0;
do
{
    t1 = 220.511575393758 * Vth0 * Math.Exp(2.48798745664686 *
        Vth0) + Vth0 * Math.Exp(4.25896818690299 * Vth0) +
        24469.8926918828 * Vth0 + 2075.73597614034;
    t2 = 220.511575393758 * Vth * Math.Exp(2.48798745664686 *
        Vth) + Vth * Math.Exp(4.25896818690299 * Vth) +
        24469.8926918828 * Vth + 2075.73597614034;
    Dt = t2 - t1;
    Dt = Dt * usporivac; //usporava po potrebi
    a = Convert.ToInt32(Dt);

    if (a < 5000 && Zastavica > 2)
    {
        a = 5000;
    }

    suma = suma + a;

    serialPort1.Write("C"); //Spaja P i G EPAD-a 2
    Thread.Sleep(350);

    device_2636.Write("smub.reset()");
    device_2636.Write("smub.sense = smub.SENSE_LOCAL");
    device_2636.Write("smub.source.func = smub.OUTPUT_DCVOLTS");
    device_2636.Write("smub.source.rangev = 30");
    device_2636.Write("smub.source.levelv = 12.5");
    device_2636.Write("smub.source.limiti = 0.00325");
    device_2636.Write("smub.measure.autorangei =
        smub.AUTORANGE_ON");
    device_2636.Write("smub.measure.i()");
    device_2636.Write("smub.source.output = smub.OUTPUT_ON");

    Thread.Sleep(a); // vreme programiranja u ms

    device_2636.Write("smub.source.output = smub.OUTPUT_OFF");

    serialPort1.Write("D"); //Spaja G i D EPAD-a 2
    Thread.Sleep(350);
    meri_Vth_b();

    Zastavica++;
} while (Vth0 <= Vth);
```

```
            if (Zastavica == 1 && Vth > 1)
            {
                usporivac = 1 - (Vth0 / Vth - 1) * 6;

                if (ZastavicaPrva == 0)
                {
                    usporivac_stari = usporivac; //prvi put prg premasi
                        vrednost
                    ZastavicaPrva = 1;
                }

                if (usporivac_stari < usporivac)
                {
                    usporivac = usporivac_stari;
                }

            }

            pisiVth.WriteLine(Vth0 + "\t" + suma);
            listPrag.Add(Vth0, suma);       // x y
            this.Invoke(delIspisProg);

        }

        pisiVth.WriteLine("];");
        pisiVth.WriteLine("");
        pisiVth.Close();

        //premeravanje prenosnih karakteristika glavnog i pomocnog EPAD-a
        serialPort1.Write("D"); //Spaja G i D EPAD-a 2
        Thread.Sleep(350);
        choose = 0;
        meri_epad_b();

        serialPort1.Write("C"); //Spaja P i G EPAD-a 2
        Thread.Sleep(350);
        choose = 1;
        meri_epad_b();
    }

    public void meri_Vth_2410()
    {
        Delegate delBrojac = new delBrojKoraka(TrenutnoBrojKoraka);
        device_2410.Write("*RST");
        device_2410.Write(":SOUR:FUNC CURR");
        device_2410.Write(":SOUR:CURR:MODE FIX");
        device_2410.Write(":SOUR:CURR:RANG:AUTO ON");
        device_2410.Write(":SOUR:CURR:LEV 1E-6");
        device_2410.Write(":SENS:VOLT:PROT 10");
        device_2410.Write(":SENS:FUNC 'VOLT'");
        device_2410.Write(":SENS:VOLT:RANG:AUTO ON");
```

```
    device_2410.Write(":FORM:ELEM VOLT");
    device_2410.Write(":OUTP ON");

    device_2410.Write(":READ?");
    naponn = device_2410.ReadString();

    Vth0 = Convert.ToDouble(naponn);
    Vth0 = Vth0 - 1; // prebacivanje u delta Vth0

    device_2410.Write(":OUTP OFF");

    trenutni_korak = Convert.ToString(Vth0);
    this.Invoke(delBrojac);
}

public void meri_Vth_2400()
{
    Delegate delBrojac2 = new delBrojKoraka2(TrenutnoBrojKoraka2);
    device_2400.Write("*RST");
    device_2400.Write(":SOUR:FUNC CURR");
    device_2400.Write(":SOUR:CURR:MODE FIX");
    device_2400.Write(":SOUR:CURR:RANG:AUTO ON");
    device_2400.Write(":SOUR:CURR:LEV 1E-6");
    device_2400.Write(":SENS:VOLT:PROT 10");
    device_2400.Write(":SENS:FUNC 'VOLT'");
    device_2400.Write(":SENS:VOLT:RANG:AUTO ON");
    device_2400.Write(":FORM:ELEM VOLT");
    device_2400.Write(":OUTP ON");

    device_2400.Write(":READ?");
    naponn = device_2400.ReadString();

    Vth0 = Convert.ToDouble(naponn);
    Vth0 = Vth0 - 1; // prebacivanje u delta Vth0

    device_2400.Write(":OUTP OFF");

    trenutni_korak2 = Convert.ToString(Vth0);
    this.Invoke(delBrojac2);
}

public void programiranj_epad_2410()
{
    ZadatiNaponPrag = ZadatiNaponPrag - 1; //prebacivanje u Delta Vth
    Delegate delIspisProg = new delIspisMerenja(obavestiProgramiranje);
    Delegate delIspisZadatNapon = new
        delIspisZadatNapon(obavestiZadatNapon);
    pisiVth = new StreamWriter(path.Text + filenameRC.Text +
        "_AprogEPAD3_" + iteratorZaMerenja.ToString() + ".m");
    vreme = DateTime.Now;
    pisiVth.WriteLine("% Vreme merenja : " + vreme);
```

```
pisiVth.WriteLine("y = [");
suma = 0;
usporivac = 1;
ZastavicaPrva = 0;

serialPort2.Write("B"); //Spaja G i D EPAD-a 3, Nano2
Thread.Sleep(2000);
meri_Vth_2410();
pisiVth.WriteLine(Vth0 + "\t" + suma);
listPrag.Add(Vth0, suma);       // x y
this.Invoke(delIspisProg);

while (Vth0 <= ZadatiNaponPrag)
{
    int i = 0;

    serialPort2.Write("B"); //Spaja G i D EPAD-a 3, Nano2
    Thread.Sleep(2000);
    meri_Vth_2410();

    do
    {
        Vth = niz[i];
        i++;
    } while (Vth < Vth0);

    this.Invoke(delIspisZadatNapon);

    Zastavica = 0;
    do
    {
        t1 = 220.511575393758 * Vth0 * Math.Exp(2.48798745664686 *
            Vth0) + Vth0 * Math.Exp(4.25896818690299 * Vth0) +
            24469.8926918828 * Vth0 + 2075.73597614034;
        t2 = 220.511575393758 * Vth * Math.Exp(2.48798745664686 *
            Vth) + Vth * Math.Exp(4.25896818690299 * Vth) +
            24469.8926918828 * Vth + 2075.73597614034;
        Dt = t2 - t1;
        Dt = Dt * usporivac; //usporava po potrebi
        a = Convert.ToInt32(Dt);

        if (a < 5000 && Zastavica > 2)
        {
            a = 5000;
        }

        suma = suma + a;

        serialPort2.Write("A"); //Spaja P i G EPAD-a 3, Nano2
        Thread.Sleep(2000);
```

```
            device_2410.Write("*RST");
            device_2410.Write(":SOUR:FUNC VOLT");
            device_2410.Write(":SOUR:VOLT:MODE FIXED");
            device_2410.Write(":SOUR:VOLT:RANG 20");
            device_2410.Write(":SOUR:VOLT:LEV 12.3");
            device_2410.Write(":SENS:CURR:PROT 3.25E-3");
            device_2410.Write(":SENS:FUNC 'CURR'");
            device_2410.Write(":SENS:CURR:RANG 10E-3");
            device_2410.Write(":OUTP ON");

            Thread.Sleep(a); // vreme programiranja u ms

            device_2410.Write(":OUTP OFF");

            serialPort2.Write("B"); //Spaja G i D EPAD-a 3, Nano2
            Thread.Sleep(2000);
            meri_Vth_2410();

            Zastavica++;
        } while (Vth0 <= Vth);

        if (Zastavica == 1 && Vth > 1)
        {
            usporivac = 1 - (Vth0 / Vth - 1) * 6;

            if (ZastavicaPrva == 0)
            {
                usporivac_stari = usporivac; //prvi put prg premasi
                    vrednost
                ZastavicaPrva = 1;
            }

            if (usporivac_stari < usporivac)
            {
                usporivac = usporivac_stari;
            }
        }

        pisiVth.WriteLine(Vth0 + "\t" + suma);
        listPrag.Add(Vth0, suma);       // x y
        this.Invoke(delIspisProg);

    }

    pisiVth.WriteLine("];");
    pisiVth.WriteLine("");
    pisiVth.Close();
}

public void programiranj_epad_2400()
{
```

```
ZadatiNaponPrag = ZadatiNaponPrag - 1; //prebacivanje u Delta Vth
Delegate delIspisProg = new delIspisMerenja(obavestiProgramiranje);
Delegate delIspisZadatNapon = new
    delIspisZadatNapon(obavestiZadatNapon);
pisiVth = new StreamWriter(path.Text + filenameRC2.Text +
    "_AprogEPAD4_" + iteratorZaMerenja.ToString() + ".m");
vreme = DateTime.Now;
pisiVth.WriteLine("% Vreme merenja : " + vreme);
pisiVth.WriteLine("y = [");
suma = 0;
usporivac = 1;
ZastavicaPrva = 0;

serialPort2.Write("D"); //Spaja G i D EPAD-a 4, Nano2
Thread.Sleep(2000);
meri_Vth_2400();
pisiVth.WriteLine(Vth0 + "\t" + suma);
listPrag.Add(Vth0, suma);       // x y
this.Invoke(delIspisProg);

while (Vth0 <= ZadatiNaponPrag)
{
    int i = 0;

    serialPort2.Write("D"); //Spaja G i D EPAD-a 4, Nano2
    Thread.Sleep(2000);
    meri_Vth_2400();

    do
    {
        Vth = niz[i];
        i++;
    } while (Vth < Vth0);

    this.Invoke(delIspisZadatNapon);

    Zastavica = 0;
    do
    {
        t1 = 220.511575393758 * Vth0 * Math.Exp(2.48798745664686 *
            Vth0) + Vth0 * Math.Exp(4.25896818690299 * Vth0) +
            24469.8926918828 * Vth0 + 2075.73597614034;
        t2 = 220.511575393758 * Vth * Math.Exp(2.48798745664686 *
            Vth) + Vth * Math.Exp(4.25896818690299 * Vth) +
            24469.8926918828 * Vth + 2075.73597614034;
        Dt = t2 - t1;
        Dt = Dt * usporivac; //usporava po potrebi
        a = Convert.ToInt32(Dt);

        if (a < 5000 && Zastavica > 2)
        {
```

```
            a = 5000;
        }

        suma = suma + a;

        serialPort2.Write("C"); //Spaja P i G EPAD-a 4, Nano2
        Thread.Sleep(2000);

        device_2400.Write("*RST");
        device_2400.Write(":SOUR:FUNC VOLT");
        device_2400.Write(":SOUR:VOLT:MODE FIXED");
        device_2400.Write(":SOUR:VOLT:RANG 20");
        device_2400.Write(":SOUR:VOLT:LEV 12.3");
        device_2400.Write(":SENS:CURR:PROT 3.25E-3");
        device_2400.Write(":SENS:FUNC 'CURR'");
        device_2400.Write(":SENS:CURR:RANG 10E-3");
        device_2400.Write(":OUTP ON");

        Thread.Sleep(a); // vreme programiranja u ms

        device_2400.Write(":OUTP OFF");

        serialPort2.Write("D"); //Spaja G i D EPAD-a 4, Nano2
        Thread.Sleep(2000);
        meri_Vth_2400();

        Zastavica++;
    } while (Vth0 <= Vth);

    if (Zastavica == 1 && Vth > 1)
    {
        usporivac = 1 - (Vth0 / Vth - 1) * 6;

        if (ZastavicaPrva == 0)
        {
            usporivac_stari = usporivac; //prvi put prg premasi
                vrednost
            ZastavicaPrva = 1;
        }

        if (usporivac_stari < usporivac)
        {
            usporivac = usporivac_stari;
        }
    }

    pisiVth.WriteLine(Vth0 + "\t" + suma);
    listPrag.Add(Vth0, suma);       // x y
    this.Invoke(delIspisProg);

}
```

```
            pisiVth.WriteLine("];");
            pisiVth.WriteLine("");
            pisiVth.Close();
        }

            private void backgroundWorker1_DoWork(object sender,
                DoWorkEventArgs e)
        {
            if (backgroundWorker1.CancellationPending)
                e.Cancel = true;

            Delegate del = new delegat(iscrtajGrafik);
            Delegate delCrtajFunkciju = new delPrag(iscrtajGrafikPrag);

            serialPort1.Open(); //otvaranje komunikacija sa Arduinom 1
            Thread.Sleep(2000);

            serialPort2.Open(); //otvaranje komunikacija sa Arduinom 2
            Thread.Sleep(2000);


            ZadatiNaponPrag = 4.00;
            programiranj_epad_a();
            this.Invoke(delCrtajFunkciju); //crta grafik

            ZadatiNaponPrag = 4.00;
            programiranj_epad_b();
            this.Invoke(delCrtajFunkciju); //crta grafik

            ZadatiNaponPrag = 4.0;
            programiranj_epad_2410();
            this.Invoke(delCrtajFunkciju); //crta grafik

            ZadatiNaponPrag = 4.0;
            programiranj_epad_2400();
            this.Invoke(delCrtajFunkciju); //crta grafik

            serialPort1.Close();
            serialPort2.Close();
        }
```

Listing B.2: Program za pokretanje releja napisan u Arduino IDE

```
//L293D
//Motor A
const int SetRelay1 = 2; //D2 - Pin 2, L293D
const int ResetRelay1 = 3; //D3 - Pin 7, L293D
//Motor B
const int ResetRelay2 = 4; //D4 - Pin 10, L293D
const int SetRelay2 = 5; //D5 - Pin 15, L293D
```

```
String data;

void setup(){

    Serial.begin(115200); //Serijska komunikacija

    //Izlazni pinovi
    pinMode(SetRelay1, OUTPUT);
    pinMode(ResetRelay1, OUTPUT);
    pinMode(ResetRelay2, OUTPUT);
    pinMode(SetRelay2, OUTPUT);
}

void loop(){
 if (Serial.available())
  {
    receivedCommand();
  }
}

void receivedCommand (){
        data = Serial.readString();

         if (data == "A") //Red LED up (Set Relay 1)
         {
             digitalWrite(SetRelay1, HIGH);
             delay(20);
             digitalWrite(SetRelay1, LOW);
         }

         if (data == "B") //Blue LED up (Reset Relay 1)
         {
             digitalWrite(ResetRelay1, HIGH);
             delay(20);
             digitalWrite(ResetRelay1, LOW);
         }

         if (data == "C") //Red LED down (Set Relay 2)
         {
             digitalWrite(SetRelay2, HIGH);
             delay(20);
             digitalWrite(SetRelay2, LOW);
         }

         if (data == "D") //Blue LED down (Reset Relay 2)
         {
             digitalWrite(ResetRelay2, HIGH);
             delay(20);
             digitalWrite(ResetRelay2, LOW);
         }
```

```
}
```

Listing B.3: Program za merenje u C#-u, glavni deo koda

```
public void Highsens() //funkcija za merenje drifta dva EPAD-a istovremeno,
    stalnim forsiranjem ZTC struja i naizmenicnim merenjem napona
        {
            Delegate delBrojac = new delBrojKoraka(TrenutnoBrojKoraka);
            Delegate delBrojac2 = new delBrojKoraka2(TrenutnoBrojKoraka2);
            Delegate delBrojac3 = new delBrojKoraka3(TrenutnoBrojKoraka3);
            Delegate delBrojac4 = new delBrojKoraka4(TrenutnoBrojKoraka4);
            Delegate delIspisProg = new delIspisMerenja(obavestiProgramiranje);
            Delegate delIspisZadatNapon = new
                delIspisZadatNapon(obavestiZadatNapon);
            Delegate del = new delegat(iscrtajGrafik);
            Delegate delGrafikTemp = new delPrag(iscrtajGrafikPrag);
            Delegate delOcistiGraf = new
                delCistiGrafik(ocistiGrafikIIsprazniListu);

            brojac_highsens = 1;
            Zastavica = 0;
            ZastavicaHighsens = 0;
            ZastavicePrekidEksperimenta = 0;

            Delta_a = 0.516142845153808; //UNOSE SE VREDNOSTI
            Delta_b = 0.517146587371827; //UNOSE SE VREDNOSTI
            Vth_pocetni_a0 = 5.094103813171387 - Delta_a; //UNOSE SE VREDNOSTI
            Vth_pocetni_b0 = 5.078830718994141 - Delta_b; //UNOSE SE VREDNOSTI

            pisiRC = new StreamWriter(path.Text + filenameRC.Text + "_drift_" +
                brojac_highsens.ToString() + ".m");
            pisiRC2 = new StreamWriter(path.Text + filenameRC2.Text + "_drift_"
                + brojac_highsens.ToString() + ".m");
            brojac_highsens++;

            vreme = DateTime.Now;

            pisiRC.WriteLine("% Vreme merenja : " + vreme);
            pisiRC.WriteLine("y = [");

            pisiRC2.WriteLine("% Vreme merenja : " + vreme);
            pisiRC2.WriteLine("y = [");

            meri_ZTC_a();

            meri_ZTC_b();

            trenutni_korak3 = Convert.ToString(Vth_pocetni_a0);
            this.Invoke(delBrojac3);

            trenutni_korak4 = Convert.ToString(Vth_pocetni_b0);
```

```
this.Invoke(delBrojac4);

while (ZastavicePrekidEksperimenta == 0)
{
    if (brojac_highsens > 2) //prvi put upisuje se izvan while
        petlje, zato je stavljen if
    {
        pisiRC = new StreamWriter(path.Text + filenameRC.Text +
            "_drift_" + brojac_highsens.ToString() + ".m");
        pisiRC2 = new StreamWriter(path.Text + filenameRC2.Text +
            "_drift_" + brojac_highsens.ToString() + ".m");
        brojac_highsens++;

        vreme = DateTime.Now;

        pisiRC.WriteLine("% Vreme merenja : " + vreme);
        pisiRC.WriteLine("y = [");

        pisiRC2.WriteLine("% Vreme merenja : " + vreme);
        pisiRC2.WriteLine("y = [");
    }

    while ((Vth0_a > Vth_pocetni_a0 * 0.9995) || (Vth0_b <=
        Vth_pocetni_b0)) //((Vth0_a > 3.985) || (Vth0_b <= 3.999))
    {
        if (ZastavicaHighsens == 0)
        {
            this.Invoke(delIspisProg);
            ZastavicaHighsens = 1;
        }

        if (sw3.ElapsedMilliseconds > 10000 && Zastavica == 1)
        {
            device_2636.Write("smub.source.output =
                smub.OUTPUT_OFF");
            sw3.Stop();
            sw3.Reset();

            serialPort1.Write("D"); //Spaja G i D EPAD-a 2
            Thread.Sleep(350);
            meri_ZTC_b();

            Zastavica = 0;
        }
        else if ((Vth0_b < Vth_pocetni_b0) && Zastavica == 0)
        {
            serialPort1.Write("C"); //Spaja P i G EPAD-a 2
            Thread.Sleep(350);

            device_2636.Write("smub.reset()");
            device_2636.Write("smub.sense = smub.SENSE_LOCAL");
```

```
        device_2636.Write("smub.source.func =
            smub.OUTPUT_DCVOLTS");
        device_2636.Write("smub.source.rangev = 30");
        device_2636.Write("smub.source.levelv = 13");
        device_2636.Write("smub.source.limiti = 0.0035");
        device_2636.Write("smub.measure.autorangei =
            smub.AUTORANGE_ON");
        device_2636.Write("smub.measure.i()");
        device_2636.Write("smub.source.output = smub.OUTPUT_ON");

        sw3.Start();
        Zastavica = 1; //Zastavica ukazuje na to da li je
            programiranje u toku ili ne
    }
    else
    {
        device_2636.Write("smua.source.output = smua.OUTPUT_ON");
        device_2636.Write("print(smua.measure.v())");
        device_2636.Write("smua.source.output =
            smua.OUTPUT_OFF");
        naponn = device_2636.ReadString();

        proteklo = sw.ElapsedMilliseconds;
        pisiRC.Write(proteklo + "\t" + naponn);
        Vth0_a = Convert.ToDouble(naponn) - Delta_a;
        trenutni_korak = Convert.ToString(Vth0_a);
        this.Invoke(delBrojac); //ispisuje trenutnu vrednost
            napona EPAD1
        proteklo = proteklo / 1000;
        listRC.Add(proteklo, Vth0_a);
        this.Invoke(del); //iscrtavace tacku na grafiku EPAD1

        if (Zastavica == 0)
        {
            device_2636.Write("smub.source.output =
                smub.OUTPUT_ON");
            device_2636.Write("print(smub.measure.v())");
            device_2636.Write("smub.source.output =
                smub.OUTPUT_OFF");
            naponn2 = device_2636.ReadString();

            proteklo2 = sw2.ElapsedMilliseconds;
            pisiRC2.Write(proteklo2 + "\t" + naponn2);
            Vth0_b = Convert.ToDouble(naponn2) - Delta_b;
            trenutni_korak2 = Convert.ToString(Vth0_b);
            this.Invoke(delBrojac2); //ispisuje trenutnu
                vrednost napona EPAD2
            proteklo2 = proteklo2 / 1000;
            listPrag.Add(proteklo2, Vth0_b);
            this.Invoke(delGrafikTemp); //iscrtavace tacku na
                grafiku EPAD2
```

```
        }

        if ((Vth0_b < Vth_pocetni_b0) && Zastavica == 0)
            //utvrdjeno da EPAD2 treba puniti, nema spavanja
        {
            Thread.Sleep(0);
        }
        else if (Zastavica == 1) //dok se programira EPAD2,
            EPAD1 se meri na po 2 sek
        {
            Thread.Sleep(2000);
        }
        else //ako je sve normalno
        {
            Thread.Sleep(2000);
        }

    }
}
ZastavicaHighsens = 0;

pisiRC.WriteLine("];");
pisiRC.WriteLine("");
pisiRC.Close();

pisiRC2.WriteLine("];");
pisiRC2.WriteLine("");
pisiRC2.Close();

this.Invoke(delOcistiGraf); //cisti grafike i liste

pisiRC = new StreamWriter(path.Text + filenameRC.Text +
    "_drift_" + brojac_highsens.ToString() + ".m");
pisiRC2 = new StreamWriter(path.Text + filenameRC2.Text +
    "_drift_" + brojac_highsens.ToString() + ".m");
brojac_highsens++;

vreme = DateTime.Now;

pisiRC.WriteLine("% Vreme merenja : " + vreme);
pisiRC.WriteLine("y = [");

pisiRC2.WriteLine("% Vreme merenja : " + vreme);
pisiRC2.WriteLine("y = [");

while ((Vth0_b > Vth_pocetni_b0 * 0.9995) || (Vth0_a <=
    Vth_pocetni_a0)) //((Vth0_b > 3.985) || (Vth0_a <= 3.999))
{
    if (ZastavicaHighsens == 0)
    {
        this.Invoke(delIspisZadatNapon);
```

```
        ZastavicaHighsens = 1;
    }

    if (sw3.ElapsedMilliseconds > 10000 && Zastavica == 1)
    {
        device_2636.Write("smua.source.output =
            smub.OUTPUT_OFF");
        sw3.Stop();
        sw3.Reset();

        serialPort1.Write("B"); //Spaja G i D EPAD-a 1
        Thread.Sleep(350);
        meri_ZTC_a();

        Zastavica = 0;
    }
    else if ((Vth0_a < Vth_pocetni_a0) && Zastavica == 0)
    {
        serialPort1.Write("A"); //Spaja P i G EPAD-a 1
        Thread.Sleep(350);

        device_2636.Write("smua.reset()");
        device_2636.Write("smua.sense = smua.SENSE_LOCAL");
        device_2636.Write("smua.source.func =
            smua.OUTPUT_DCVOLTS");
        device_2636.Write("smua.source.rangev = 30");
        device_2636.Write("smua.source.levelv = 13");
        device_2636.Write("smua.source.limiti = 0.0035");
        device_2636.Write("smua.measure.autorangei =
            smua.AUTORANGE_ON");
        device_2636.Write("smua.measure.i()");
        device_2636.Write("smua.source.output = smua.OUTPUT_ON");

        sw3.Start();
        Zastavica = 1; //Zastavica ukazuje na to da li je
            programiranje u toku ili ne
    }
    else
    {
        device_2636.Write("smub.source.output = smub.OUTPUT_ON");
        device_2636.Write("print(smub.measure.v())");
        device_2636.Write("smub.source.output =
            smub.OUTPUT_OFF");
        naponn2 = device_2636.ReadString();

        proteklo2 = sw2.ElapsedMilliseconds;
        pisiRC2.Write(proteklo2 + "\t" + naponn2);
        Vth0_b = Convert.ToDouble(naponn2) - Delta_b;
        trenutni_korak2 = Convert.ToString(Vth0_b);
        this.Invoke(delBrojac2); //ispisuje trenutnu vrednost
            napona EPAD2
```

```
            proteklo2 = proteklo2 / 1000;
            listPrag.Add(proteklo2, Vth0_b);
            this.Invoke(delGrafikTemp); //iscrtavace tacku na
                grafiku EPAD2

            if (Zastavica == 0)
            {
                device_2636.Write("smua.source.output =
                    smua.OUTPUT_ON");
                device_2636.Write("print(smua.measure.v())");
                device_2636.Write("smua.source.output =
                    smua.OUTPUT_OFF");
                naponn = device_2636.ReadString();

                proteklo = sw.ElapsedMilliseconds;
                pisiRC.Write(proteklo + "\t" + naponn);
                Vth0_a = Convert.ToDouble(naponn) - Delta_a;
                trenutni_korak = Convert.ToString(Vth0_a);
                this.Invoke(delBrojac); //ispisuje trenutnu vrednost
                    napona EPAD1
                proteklo = proteklo / 1000;
                listRC.Add(proteklo, Vth0_a);
                this.Invoke(del); //iscrtavace tacku na grafiku EPAD1
            }

            if ((Vth0_a < Vth_pocetni_a0) && Zastavica == 0)
                //utvrdjeno da EPAD1 treba puniti, nema spavanja
            {
                Thread.Sleep(0);
            }
            else if (Zastavica == 1) //dok se programira EPAD2,
                EPAD1 se meri na po 2 sek
            {
                Thread.Sleep(2000);
            }
            else //ako je sve normalno
            {
                Thread.Sleep(2000);
            }

        }
    }
    ZastavicaHighsens = 0;

    pisiRC.WriteLine("];");
    pisiRC.WriteLine("");
    pisiRC.Close();

    pisiRC2.WriteLine("];");
    pisiRC2.WriteLine("");
    pisiRC2.Close();
```

```
            this.Invoke(delOcistiGraf); //cisti grafike i liste
        }

        sw.Stop();
        sw2.Stop();

        device_2636.Write("smua.reset()");
        device_2636.Write("smua.source.output = smua.OUTPUT_OFF");
        device_2636.Write("smub.reset()");
        device_2636.Write("smub.source.output = smub.OUTPUT_OFF");
    }
```