



UNIVERZITET SINGIDUNUM
FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Sinteza sopstvenog izvora slučajnosti - TRNG

- Diplomski rad -

Mentor:

Doc. dr Saša Adamović

Student:

Igor Fermevc

Br. Indeksa:2012/201044

Beograd, 2015.



FAKULTET ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Beograd, Danijelova 32

Kandidat: Igor Fermevc

Broj indeksa: 2012/201044

Smer: Informatika i računarstvo

Tema: Sinteza sopstvenog izvora slučajnosti - TRNG

Zadatak:

Datum odobrenja rada:
Beograd, __.__.____.

MENTOR

Doc. dr Saša Adamović

DEKAN

Prof. dr Dragan Cvetković

Sinteza sopstvenog izvora slučajnosti - TRNG

Sažetak: U ovom radu će biti prikazan pregled nekoliko postojećih rešenja na polju generatora slučajnosti realizovanih pomoću komercijalno dostupnih elektronskih komponenti. U poređnom analizom tehničkih karakteristika i načina na koji navedeni elektronski sklopovi generišu „potpuno“ slučajne sekvence binarnih vrednosti, biće objašnjene prednosti i mane predstavljenih rešenja. Nakon zaključka izvedenog analizom već postojećih rešenja, biće prikazana izgradnja elektronskog uređaja za generisanje slučajnosti po uzoru na uređaj koji predlaže Julien Thomas, entuzijasta na polju elektronike iz Nemačke, a prikazan je na njegovoj Internet prezentaciji [1]. Generator slučajnosti je realizovan u obliku prenosnog USB fleš diska. Ovaj uređaj se napaja preko USB priključka računara, kojem se nakon priključivanja prijavljuje kao serijski komunikacioni port. Primenom ovakvog koncepta, postignuta je maksimalna prenosivost uređaja i olakšana je komunikacija sa istim. Opisani uređaj će biti podvrgnut statističkim testovima koji su namenjeni za testiranje izvora slučajnosti, a definiše ih NIST (eng. National Institute of Standards and Technology). U nastavku ovog rada, biće prikazani rezultati testiranja uređaja i analiza dobijenih rezultata. Komunikacija sa uređajem i preuzimanje generisanih slučajnih sekvenci će biti izvršeni upotrebom namenske aplikacije koju je razvio Julien Thomas, a testiranje dobijenih podataka i prikaz rezultata statističkih testova biće izvršeni korišćenjem odgovarajuće Java aplikacije koja je razvijena na Univerzitetu Singidunum. Na kraju rada navodimo moguća poboljšanja funkcionalnosti uređaja i načine za rešenje problema uočenih prilikom testiranja i rada sa uređajem.

Ključne reči: generator slučajnosti, kriptologija, elektronski šum

Synthesis of personal source of randomness

Abstract: This paper will present an overview of several hardware based randomness source solution realized by the use of commercially available electronic components. By comparing technical characteristics of these devices and the ways they generate randomness, we'll elaborate the good and the bad sides of these solutions. To be able to conduct statistical tests, we'll explain the process of developing our own true RNG solution based on the work of Julien Thomas, electronic engineer and "open source" enthusiast from Germany, which is presented at his Internet site [1]. Randomness generator is realized in the form of USB flash memory (USB stick). Device is powered by host computer and is recognized by the host as serial communication port. These features enable maximum portability of the device and make the communication easier. The values generated by this device will be submitted to battery of statistical tests defined by NIST. These results will be presented and further analyzed. In order to communicate with the device and to use generated values, we're using a custom software application developed by Julien Thomas. For statistical tests evaluation, we're using an Java software application developed by University Singidunum. Finally, we'll point to possible improvements of the device and we'll propose the ways to overcome the problems observed during the use and testing of the device.

Keywords: sources of randomness, cryptology, electronic noise

SADRŽAJ

Spisak slika	- 3 -
Spisak tabela.....	- 3 -
1 Uvod	- 4 -
1.1 Predmet istraživanja	- 4 -
1.2 Doprinosi rada	- 4 -
1.3 Struktura rada.....	- 4 -
2 Pregled u oblasti istraživanja	- 5 -
2.1 Izvori slučajnosti	- 5 -
2.1.1 Pseudo-izvori slučajnosti – PRNG	- 5 -
2.1.2 Izvori slučajnosti – TRNG.....	- 6 -
2.2 Analiza komercijalno dostupnih izvora slučajnosti (TRNG).....	- 6 -
2.2.1 Araneus - Alea II.....	- 6 -
2.2.2 TectroLabs - TL200.....	- 7 -
2.2.3 Ublid.it – TrueRNG2.....	- 7 -
2.2.4 Moonbase Otago – OneRNG	- 8 -
3 Teorijske osnove istraživanja	- 9 -
3.1 Pojam informacije.....	- 9 -
3.2 Entropija	- 9 -
3.3 Signal šuma kao izvor slučajnosti	- 10 -
3.3.1 Pojam eelektronskog šuma	- 10 -
3.3.2 Vrste i karakteristike elektronskog šuma.....	- 10 -
3.4 Karakteristike izvora slučajnosti (NIST).....	- 12 -
3.4.1 NIST baterija statističkih testova	- 12 -
3.4.2 Ispitivanje učestalosti u nizu	- 12 -
3.4.3 Ispitivanje učestalosti u bloku	- 12 -
3.4.4 Ispitivanje uzastopnih ponavljanja istih bitova u nizu	- 13 -
3.4.5 Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku	- 14 -
3.4.6 Ispitivanje stanja binarne matrice	- 15 -
3.4.7 Ispitivanje diskretne Furijerove transformacije	- 15 -
3.4.8 Ispitivanje nepreklapajućih uzoraka	- 16 -

3.4.9 Ispitivanje preklapajućih uzoraka	- 17 -
3.4.10 Maurerov univerzalni statistički test	- 18 -
3.4.11 Ispitivanje linearne kompleksnosti	- 18 -
3.4.12 Serijski test	- 19 -
3.4.13 Ispitivanje približne entropije.....	- 19 -
3.4.14 Ispitivanje slučajnih zbirova	- 20 -
4 Pregled predloženog rešenja	- 22 -
4.1 Generička šema predloženog rešenja.....	- 22 -
4.2 Opis izrade generatora sklučajnosti.....	- 25 -
4.3 Postavka i objašnjenje eksperimentalnog okruženja	- 27 -
4.3.1 Režim rada 1.....	- 28 -
4.3.2 Režim rada 2.....	- 28 -
4.3.3 Režim rada 3.....	- 29 -
4.3.4 Režim rada 4.....	- 29 -
4.4 Performanse predloženog okruženja i prikaz eksperimentalnih rezultata....	- 30 -
4.4.1 Test ispitivanja učestanosti u nizu	- 31 -
4.4.2 Serijski test	- 32 -
4.4.3 Poker test	- 33 -
4.4.4 Test uzastopnih ponavljanja iste vrednosti.....	- 34 -
4.4.5 Entropijski test (sa i bez preklapanja)	- 35 -
4.5 Oblast primene predloženog rešenja	- 36 -
5 Zaključak.....	- 36 -
5.1 Sumarni ciljevi rada i ostvareni rezultati.....	- 36 -
5.2 Predlog za budući rad	- 37 -
6 Literatura	- 37 -
PRILOG 1 Listing izvornog programskog koda mikrokontrolera	- 38 -
PRILOG 2 Izvorni kod XR232NT aplikacije	- 45 -

Spisak slika

Slika 2.1-1 Bitmapa sekvence koju generiše PHP rand() metod na Windows OS-u	- 5 -
Slika 2.1-2 Bitmapa sekvence koju generiše „pravi“ izvor slučajnosti.....	- 6 -
Slika 2.2-1 Araneus Alea II TRNG	- 7 -
Slika 2.2-2 TectroLabs TL200	- 7 -
Slika 2.2-3 Ubld.it TrueRNG2	- 8 -
Slika 2.2-4 Moonbase Otago OneRNG	- 8 -
Slika 4.1-1 Blok šema uređaja.....	- 22 -
Slika 4.1-2 Analogni signal šuma na ulazu prvog dela komparatora	- 22 -
Slika 4.1-3 „Digitalni šum“ koji se dovodi na ulaz mikroprocesora	- 23 -
Slika 4.1-4 Uprošćena šema programskog koda uređaja.....	- 24 -
Slika 4.2-1 Električna šema generatora slučajnosti	- 25 -
Slika 4.2-2 Izgled štampane ploče sa rasporedom komponenti.....	- 26 -
Slika 4.3-1 Parametri za podešavanje komunikacionog porta (Windows OS)	- 27 -
Slika 4.3-2 Kontinualni ispis generisanih bajtova na monitoru	- 28 -
Slika 4.3-3 Prikaz generisanih bajtova pomoću bitmape 512x384 piksela	- 28 -
Slika 4.3-4 Histogram kumulativnih vrednosti frekvencije generisanih bajtova.....	- 29 -
Slika 4.3-5 Upisivanje generisanih bajtova u datoteku.....	- 29 -
Slika 4.4-1 Rezultati testa „ispitivanje učestanosti“ – uzorak TRNG	- 31 -

Spisak tabela

Tabela 2.2-1 Tabelarni prikaz tehničkih karakteristika	- 9 -
Tabela 4.2-1 Vrednosti i količine upotrebljenih elektronskih komponenti	- 26 -

1 Uvod

1.1 *Predmet istraživanja*

Nepovezani i nepredvidljivi događaji – nasumičnost, slučajnost ili nešto treće? Čovek je kroz vekove bio zaintrigiran pozadinom nekog događaja za čiju pojavu nije imao logično objašnjenje. Često povezivani sa sudbinom, verom u nešto apstraktno ili jednostavno pripisivani srećnim okolnostima, definicija slučajnosti ili nepredvidljivosti neke pojave ili rezultata neke aktivnosti i danas su predmet rasprave u velikom broju naučnih oblasti. Od filozofije, preko psihologije, medicine i ostalih prirodnih nauka pa sve do fizike i matematike, mnogo je pokušaja da se definiše slučajnost.

U ovom radu se nećemo baviti apstraktnim odlikama neodređenosti već ćemo se osloniti na strogo definisane matematičke principe kako bismo vrednosti koje generiše neki izvor slučajnosti mogli da deklariramo kao čisto slučajne, ničim povezane, nezavisne događaje. Sa stanovišta zaštite informacionih sistema, mogućnost generisanja čisto slučajnih vrednosti ima presudan značaj. Nemoguće je zamisliti neki od algoritama za šifrovanje ili računarske programe, naročito na polju raznih simulacija i veštačke inteligencije, u kojima se ne koristi slučajno generisana vrednost. Imajući u vidu činjenicu da u modernoj kriptologiji implementacija nekog sistema zaštite direktno zavisi od kvaliteta, tj. jačine slučajno generisanih vrednosti koje se koriste kao ključevi ili materijal za generisanje kriptološki jakih ključeva, lako možemo zaključiti koliki značaj imaju generatori slučajnosti te ne čudi koliko se vremena i energije ulaže u istraživanja u ovoj oblasti.

1.2 *Doprinosi rada*

- Prikaz i objašnjenje tehničkih karakteristika i performansi postojećih, komercijalno dostupnih, rešenja na polju generatora slučajnosti.
- Usporedna analiza postojećih, komercijalno dostupnih, rešenja.
- Razvoj sopstvenog hardverskog generatora slučajnosti po uzoru na rešenje koje predlaže J.Thomas [1]
- Detaljan opis izgradnje i prikaz rada uređaja sa uputstvom za upotrebu uređaja i postojećeg pratećeg softvera.
- Provera kvaliteta sopstvenog rešenja primenom NIST statističkih testova nad podacima koje generiše naš izvor slučajnosti.
- Predlog za poboljšanje rada uređaja u cilju povećanja sigurnosti i brzine rada.

1.3 *Struktura rada*

U uvodnom poglavlju smo objasnili pojam slučajnosti i ukratko opisali ulogu i značaj generatora slučajnosti sa aspekta zaštite informacionih sistema.

U drugom poglavlju, izvršen je pregled aktuelne situacije u oblasti od interesa za ovaj rad i prikazani su detalji vezani za nekoliko komercijalno dostupnih generatora slučajnosti.

U trećem poglavlju je prikazan kratak osvrt na teorijsku osnovu koja stoji iza generatora slučajnosti. Opisane su karakteristike signala šuma i njihova upotreba u generatorima slučajnosti i predstavljena je baterija NIST testova kojima se statistički dokazuje kvalitet generatora.

U četvrtom poglavlju opisano je postavljeno eksperimentalno okruženje i prikazan je detaljan proces razvoja sopstvenog generatora slučajnosti, počev od generičke šeme uređaja, preko električne šeme, opisa rada elektronskog sklopa, a poglavlje završavamo prikazom rezultata NIST testova kojima smo podvrgli vrednosti koje generiše naš izvor.

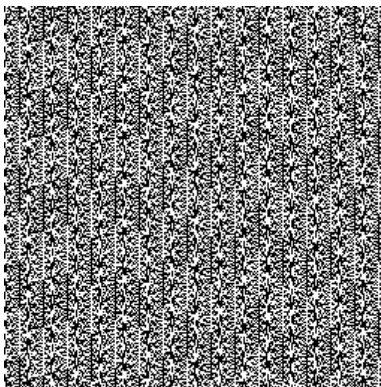
Na kraju rada, u petom poglavlju, izvodimo zaključke i predlažemo moguća poboljšanja našeg rešenja kao osnov za budući rad.

2 Pregled u oblasti istraživanja

2.1 Izvori slučajnosti

2.1.1 Pseudo-izvori slučajnosti – PRNG

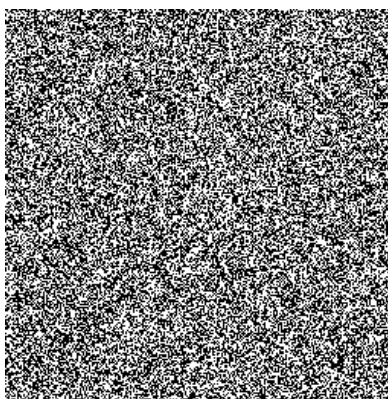
Ovi generatori su bazirani na algoritmima koji u osnovi koriste matematičke formule ili unapred definisane liste ili tabele vrednosti. Svaki od ovih načina implementacije koristi inicijalnu (engl. Seed), za algoritam eksternu, slučajnu vrednost. Ovakav pristup je ujedno i jedno od ograničenja PRNG sistema. Bez obzira na to koliko slučajno izgledaju sekvence koje upotrebljeni algoritam generiše, interna stanja ovih generatora su potpuno određena inicijalnom vrednošću. Za istu inicijalnu vrednost, algoritam proizvodi iste vrednosti na izlazu. Drugi problem koji se pojavljuje kod PRNG sistema je periodičnost. Ukoliko bi u dovoljno dugom vremenskom periodu posmatrali generisanu sekvencu, u nekom trenutku vremena, ta sekvenca bi počela da se ponavlja. Slika 2.1-1 prikazuje sekvencu generisanu od strane lošeg pseudo-slučajnog izvora konvertovanu u bitmap-u radi lakšeg uočavanja ponavljajućeg šablona.



Slika 2.1-1 Bitmapa sekvence koju generiše PHP rand() metod na Windows OS-u

2.1.2 Izvori slučajnosti – TRNG

Polazeći od pretpostavke da su samo procesi koji se dešavaju u prirodi potpuno neodređeni, ovi generatori slučajnosti oslanjaju se na razne fizičke događaje i procese. Za razliku od pseudo-izvora slučajnosti, ovde izlaz nije ni u kakvoj vezi za nekim od prethodnih stanja generatora ili sa nekim od prethodno generisanih vrednosti na izlazu. Iako deluju idealno ovi izvori slučajnosti imaju i svoje nedostatke. Pored problema da se prirodni izvor slučajnosti priključi na računar, generisane sekvence po pravilu pokazuju neravnomernu raspodelu verovatnoća generisanih vrednosti u posmatranom uzorku dobijenom na izlazu generatora. Kako bi se ovaj problem rešio neophodno je izvršiti neki vid naknadnog procesiranja dobijenih vrednosti. Slika 2.1-2 prikazuje sekvencu generisanu od strane hardverskog generatora slučajnosti konvertovanu u bitmapu i na njoj je teško uočiti bilo kakve šablone i ponavljanja.



Slika 2.1-2 Bitmapa sekvence koju generiše „pravi“ izvor slučajnosti

2.2 Analiza komercijalno dostupnih izvora slučajnosti (TRNG)

U nastavku rada biće opisani principi rada i navedene tehničke karakteristike nekoliko komercijalno dostupnih uređaja za generisanje slučajnosti. Sva prikazana rešenja realizovana su po istom principu. Ovi uređaji kao izvor slučajnosti koriste određena fizička svojstva poluprovodnika ili druge fizičke pojave u elektronskim komponentama. Imajući u vidu karakteristike signala koji generišu ovi elektronski sklopovi, svi uređaji ovog tipa implementiraju neki vid naknadne obrade istog sa ciljem obezbeđivanja kriptološki upotrebljivih slučajnih vrednosti. Svi uređaji su koncipirani tako da se napajaju i ostvaruju komunikaciju sa sistemom putem USB priključka računara na koji su priključeni.

2.2.1 Araneus - Alea II

Alea II je proizvod kompanije Araneus iz Finske [4]. Predstavlja poboljšanu varijantu Alea I generatora slučajnih brojeva čiji je izvor slučajnosti se oslanja na šum dobijen usled fizičkih procesa obrnuto polarisanog PN spoja. Na ovaj način je generisan beli šum u širokom frekventnom opsegu. Princip dalje obrade analognog signala šuma se ogleda u pojačavanju i konvertovanju istog u digitalni signal, nakon čega se daljim procesiranjem otklanjaju eventualne nepravilne raspodele slučajnih binarnih vrednosti i

koleracije binarnih sekvenci. Prema podacima proizvođača, ovaj uređaj generiše slučajne vrednosti brzinom od 100kb/s i u radu troši maksimalno 250mW. Proizvođač uz uređaj isporučuje upravljački softver koji omogućava upotrebu uređaja na svim vrstama operativnih sistema. Izvorni kod USB protokola je javno dostupan i dokumentovan što omogućava krajnjim korisnicima da isti prilagođavaju svojim potrebama. Uzorak podataka generisanih pomoću ovog uređaja dostupni su za preuzimanje na prezentaciji proizvođača kao i dokumenti sa rezultatima statističkih testova kojima je uređaj podvrgnut. Prodajna cena uređaja je oko 120 eura.



Slika 2.2-1 Araneus Alea II TRNG

2.2.2 TectroLabs - TL200

TL200 je uređaj proizveden u SAD od strane kompanije TectroLabs LLC [5]. Naslednik modela TL100 koristi dva nezavisna generatora slučajnosti baziranih na principu „efekta lavine“ kod obrnuto polarisane Zener diode. Rezultat je RF šum koji se AD konvertorima digitalizuje i pre dalje obrade konstantno se vrše dva tipa testiranja prema NIST (eng. Repetition Count Test, Adaptive Proportion Test). U nastavku procesiranja, dva dobijena bajta na ulazu prolaze XOR logičku operaciju. Rezultujući bajt se dalje obrađuje HASH funkcijom što na kraju daje čisto slučajne nizove bajtova. U zavisnosti od izabranog profila rada uređaja postiže se maksimalna brzina generisanja slučajnih vrednosti od 2Mb/s uz maksimalnu potrošnju struje od 98mA. Uređaj je dostupan po ceni od 199 dolara i isporučuje se sa potrebnim upravljačkim programima i dokumentacijom. Na prezentaciji proizvođača dostupni su za preuzimanje rezultati statističkih testova za izvore slučajnosti. Dodatno, proizvođač je upotrebom FTDIChip-ID tehnologije omogućio implementaciju dodatne zaštite na nivou komunikacije sa uređajem.



Slika 2.2-2 TectroLabs TL200

2.2.3 Ubld.it – TrueRNG2

Ovaj uređaj se može naručiti po ceni od 50 dolara i proizvodi se u SAD [6]. Izvor slučajnosti je dvostruki obrnuto polarisan PN spoj, a naknadno procesiranje se vrši uzorkovanjem sekvenci nakon procesa digitalizacije šuma. Podeljene sekvence prolaze XOR logičku operaciju. Na osnovu podataka sa prezentacije proizvođača, nakon XOR operacije u odnosu 20:1, zadržana je visoka entropija generisanih sekvenci uz brzine

generisanja veće od 350kb/s. Proizvođač obezbeđuje odgovarajuće upravljačke programe za uređaj i oni su dostupni sa preuzimanje na prezentaciji proizvođača. Nažalost, osim vrlo pozitivnih kritika kupaca ovog uređaja, ne postoji nikakva detaljnija dokumentacija o samom proizvodu.



Slika 2.2-3 Ublid.it TrueRNG2

2.2.4 Moonbase Otago – OneRNG

Proizvod kompanije Moonbase Otago iz Novog Zelanda za čiju masovnu proizvodnju je nedavno pokrenuta „Kickstarter“ kampanja [7]. Kao izvor slučajnosti koristi kombinaciju obrnuto polarisanog PN spoja i RF prijemnik. Proizvođač je kompletan elektronski sklop i izvorni kod softvera i firmvera učinio dostupnim javnosti sa primarnim ciljem da krajnji korisnici mogu proveriti rad i svaki deo ovog uređaja. Na ovaj način omogućena je svaka vrsta prepravke ili poboljšanja uređaja od strane korisnika koji može biti siguran da uređaj nema skrivenih opcija koje mogu narušiti sigurnost. Jedina komponenta kojoj „poklanjamo poverenje“ po pitanju sigurnosti je mikroprocesor. Brzina generisanja slučajnih sekvenci je 350kb/s, generisane vrednosti prolaze sve predviđene statističke testove za izbore slučajnosti. Rezultati testova dostupni su na prezentaciji proizvođača. Prodaja uređaja planirana je za maj 2015. godine po ceni od 50 dolara.



Slika 2.2-4 Moonbase Otago OneRNG

U tabeli 2.2-1 date su karakteristike opisanih rešenja sa ciljem bolje preglednosti. Osim pomenutih rešenja na tržištu postoji daleko veći broj uređaja koji, kako svojom cenom, tako i brzinom generisanja slučajnosti daleko premašuju okvire koje smo postavili u ovom radu, a odnose se na cenu i dostupnost upotrebljenih komponenti.

Model	Brzina	Izvor slučajnosti	Potrošnja el.en.	Cena
Alea II	100 kb/s	PN spoj	max 250 mW	120 EUR
TL200	2 Mb/s	PN spoj	max 400 mW	199 USD
True RNG2	350 kb/s	PN spoj	-	50 USD
One RNG	350 kb/s	PN spoj + RF p-p	-	50 USD
XR232USB	200 kb/s	Oscilator (an. Komp.)	max 100 mW	*
ZIS XR232	72 kb/s	Oscilator (an. Komp.)	max 100 mW	15 EUR

Tabela 2.2-1 Tabelarni prikaz tehničkih karakteristika

3 Teorijske osnove istraživanja

3.1 Pojam informacije

Informacija kao pojam može se definisati na više načina. U svakodnevnom životu, veoma često upotrebljavamo pojam „informacija“. U tom opštem smislu, informacija mnoštvu generisanih podataka daje neki smisao i upotrebljivost, tačnije primalac iz informacije saznaje nešto novo čime uvećava svoje znanje. Informacija je danas toliko značajna da je možemo posmatrati kao dominantni resurs u svim oblastima društvenog i naučnog života [8]. Posmatrano sa tehničkog aspekta, iako je bilo naučnih radova koji su se bavili ovom materijom [9], tek je 1948. godine Klod Šenon (Claude E. Shannon 1916-2001) u svom čuvenom radu [10] postavio matematičke osnove za teoriju informacije kao naučne discipline koja se u mnogome oslanja na teoriju verovatnoće i statistike a njen uticaj je ogroman i omogućio je realizaciju nekih od najvećih tehnoloških dostignuća današnjice.

3.2 Entropija

U teoriji informacije, po Šenonu, entropija je mera za količinu neodređenosti, tačnije, odnosi se na prosečnu količinu informacije sadržane u porukama koju generiše posmatrani informacioni izvor. Svaka poruka se posmatra kao događaj ili uzorak izvučen iz skupa svih mogućih događaja za posmatrani izvor informacije. Entropija dakle, karakterizuje neodređenost izvora informacije. Informacioni izvor takođe karakterizuje i raspodela verovatnoće kojom je neka poruka generisana, tačnije, što je manja verovatnoća da se generiše određena poruka ili događaj, to ona sadrži veću

* Sve informacije o uređaju su javno dostupne na [1], uključujući i softver potreban za rad uređaja. Uređaj nije moguće kupiti kao gotov proizvod već ga je potrebno samostalno proizvesti.

količinu informacije. Kada posmatramo binarni izvor informacije, tj. izvor koji može generisati samo dve poruke 0 ili 1, entropija H će zavisi od verovatnoća p kojom se poruke generišu i data je formulom:

$$H = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

Obzirom da entropija zavisi od raspodele verovatnoća, možemo zaključiti da će ista biti maksimalna ako je ova raspodela „normalna“, tj. ukoliko su verovatnoće da se generiše 0 i 1 jednake. Glavni cilj prilikom projektovanja generatora slučajnosti je upravo postizanje ravnomerne raspodele verovatnoća događaja za svaki element polaznog skupa. U slučaju binarnog izvora informacije, za osnovu logariamske funkcije je uzet broj 2, a mera za entropiju je bit.

3.3 *Signal šuma kao izvor slučajnosti*

3.3.1 Pojam eelektronskog šuma

Šum ili smetnja, može se definisati kao neželjena pojava koja degradira ili na neki drugi način utiče na signal koji prenosi neku informaciju. Po mestu nastajanja, a u odnosu na elektronski sklop koji posmatramo, šum možemo podeliti na spoljni i unutrašnji. Spoljni šum potiče od izvora kao što su atmosferska pražnjenja, elektromagnetna zračenja u okolini i najčešće su prolaznog karaktera. Za potrebe ovog rada, fokusiraćemo se na unutrašnje šumove koji su posledica protoka naelektrisanja kroz elektronske komponente ili nastaju usled nesavršenosti materijala upotrebljenih za izradu elektronskih komponenti ili samog procesa proizvodnje istih. Prilikom projektovanja elektronskih sklopova, ulažu se ogromni napor u pronalaženje tehničkih rešenja koja za cilj imaju maksimalno smanjivanje uticaja šuma na koristan signal koji prenosimo. Suprotno ovoj tezi, u ovom radu ćemo specifične karakteristike šuma upotrebiti kao izvor slučajnosti na način koji će biti objašnjen u daljem tekstu.

3.3.2 Vrste i karakteristike elektronskog šuma

3.3.2.1 Termički šum

Ova vrsta šuma je prisutna u svim provodnicima i uzrokuje je nepredvidljivo kretanje elektrona kao posledica promene temperature u elektro-provodnim komponentama i ne zavisi od primenjenog napona niti od površine samog provodnika. Ova vrsta šuma se još naziva i „Jonson-Nyquist“ šum po naučnicima koji su okarakterisali i matematički definisali [11][12] njegova svojstva. Karakteristično za ovu vrstu šuma je skoro ujednačena raspodela snage u, teoretski, neograničenom frekventnom opsegu. Upravo ova karakteristika kvalifikuje signal ove vrste šuma za upotrebu u generatorima slučajnosti. U praksi ipak postoji gornja granica frekventnog opsega koja je određena karakteristikama upotrebljenih komponenti u elektronskom sklopu.

3.3.2.2 Fliker šum ($1/f$)

Za razliku od termičkog šuma, raspodela snage fliker šuma opada na višim frekvencijama. Fliker šum je direktna posledica protoka jednosmerne struje i zavisi od njene jačine. Snaga ovog šuma je obrnuto proporcionalna frekvenciji odakle potiče i skraćeni naziv „ $1/f$ “. Uopšteno posmatrajući karakteristike ovog šuma, treba posvetiti posebnu pažnju izvedbi elektronskog sklopa ukoliko je njegova namena generisanje slučajnosti.

3.3.2.3 Šum usled efekta „sačme“

Ova vrsta šuma, kao termalni šum, nastaje u svim provodnicima kada kroz njih teče struja. Karakteristično za ovu vrstu šuma je da, za razliku od termalnog, ne zavisi od temperature. Kao i termalni šum, raspodela snage pojedinačnih komponenti signala je normalna u širokom frekventnom opsegu iz čega zaključujemo da je ovaj signal pogodan za dalju obradu u generatorima slučajnosti. Potencijalni problem kod generatora koji koriste ovu vrstu šuma kao izvor slučajnosti je u poluprovodničkim spojevima koji rade u posebnom režimu, blizu proboja PN spoja što nakon dužeg vremena upotrebe dovodi ili do otkaza upotrebene komponente ili do, sa aspekta entropije, pada u kvalitetu generisanih sekvenci.

3.3.2.4 Šum dobijen upotrebom analognog komparatora

Analogni komparator je elektronski sklop čija je najčešća primena u AD konverziji signala. Ako posmatramo mogućnosti pojačanja signala, komparator je sklop vrlo sličan operacionom pojačavaču i ima veliki koeficijent pojačanja ulaznog signala. Ukoliko postavimo standardni, komercijalno dostupni, komparator u kolo u kojem su vrednosti pratećih elektronskih komponenti tako odabrane da se iskoristi maksimalni faktor pojačanja uz eliminisanje ugrađene „histerezis“ funkcije za rezultat dobijamo haotično ali ipak kontrolisano oscilovanje sklopa. Iako se prilikom projektovanja elektronskih sklopova teži da se smanji uticaj šuma na rad i izbegnu neželjene oscilacije, u slučaju generisanja slučajnosti upravo nam ovakvo ponašanje odgovara. Na izlazu komparatora dobijamo signal šuma širokog frekventnog opsega, čija je raspodela snage svih komponenti normalna čime ispunjavamo polazni osnov za dalju obradu ovakvog signala. Još jedna dobra karakteristika ovog koncepta je mogućnost korišćenja jedne vrednosti napona za ceo sklop bez potrebe za dodatnim komponentama koje „podizuju“ napon kao u slučaju rešenja sa inverzno polarisanim PN spojem, što se na kraju odražava i na smanjenje ukupne potrešnje uređaja.

3.4 Karakteristike izvora slučajnosti (NIST)

3.4.1 NIST baterija statističkih testova

NIST-ov paket testova predstavlja paket koji se sastoji od 15 statističkih testova razvijenih za testiranje karakteristika binarnih sekvenci koje generišu hardverski ili softverski genratori slučajnosti na bazi kriptografski slučajnih ili pseudoslučajnih brojeva. Ovi testovi se fokusiraju na one osobine testiranih sekvenci koje odstupaju od očekivanih obeležja slučajnosti a koje bi mogle da postoje u testiranom nizu. Neki testovi su podeljeni u više podtestova a opisi testova i formule matematičkih funkcija koji slede su preuzete u celosti iz NIST dokumenta [13].

3.4.2 Ispitivanje učestalosti u nizu

Ovaj test vrši ispitivanje broja ponavljanja vrednosti u binarnoj sekvenci, tj. jedinica i nula u nizu bitova. Karakteristika dobrog izvora slučajnosti je da broj nula i jedinica u posmatranom uzorku bude približno jednak, uz minimalna odstupanja. Ukoliko testirana sekvenca ne prođe ovaj test, rezultati svih narednih testova se dovode u pitanje ili su nezadovoljavajući. Preporuka je da sekvenca koja se testira sadrži najmanje stotinu bitova.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Duzina binarne sekvence n

Elementi binarne sekvence se konveruju u vrednosti „1“ i „-1“. Zatim se svi elementi sabiraju i dobija se S_n . $S_n = X_1 + X_2 + \dots + X_n$, ($X_i = 2\epsilon_i - 1$)

Izračunava se vrednost S_{obs} - statistika testa: $S_{obs} = \frac{|S_n|}{\sqrt{n}}$

Izračunavanje izlaznog parametra P – rezultat testa:

erfc – komplementarna funkcija greške: $erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-u^2} du$

$$P = erfc\left(\frac{|S_{obs}|}{\sqrt{2}}\right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost P manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.3 Ispitivanje učestalosti u bloku

Test proverava da li je u M-bitnom bloku broj jedinica približan $M/2$. Ulazna sekvenca koja se testira, deli se na blokove od M bita bez preklapanja, a eventualni višak bitova se odbacuje. Preporuka je da sekvenca koja se testira sadrži najmanje stotinu bitova.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina binarne sekvence n
- Dužina svakog bloka binarne sekvence M

Binarna sekvenca se deli na blokove: $N = \frac{n}{M}$.

Određuju se proporcija jedinica svakog M bloka: $\pi_i = \frac{\sum_{j=1}^M \epsilon_{(i-1)M+j}}{M}$, za $1 \leq i \leq N$.

Izračunava se χ^2 – statistika testa: $\chi^2(obs) = 4M \sum_{i=1}^N (\pi_i - \frac{1}{2})^2$.

Izračunavanje izlaznog parametra P – rezultat testa:

igamc – nekompletna gama funkcija:

$$P(a, x) \equiv \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt, \text{ gde je } P(a, 0) = 0 \text{ a } P(a, \infty) = 1.$$

$$P = \text{igamc} \left(\frac{N}{2}, \frac{\chi^2(obs)}{2} \right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa P manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.4 Ispitivanje uzastopnih ponavljanja istih bitova u nizu

Posmatrana karakteristika u ovom testu je ukupan broj uzastopnih ponavljanja istih vrednosti u nizu. Svrha ovog testa je da se odredi da li promenljivi broj uzastopnih ponavljanja nula i jedinica odgovara karakteristikama koje se očekuju u slučajnoj sekvenci. Još preciznije rečeno, test prikazuje da li je oscilacija u promenama vrednosti iz 0 u 1 prebrza ili prespora. Preporuka je da sekvenca koja se testira sadrži najmanje stotinu bitova.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina binarne sekvence n

Izračunati proporciju jedinica u ulaznoj sekvenci: $\pi = \frac{\sum j \epsilon_j}{n}$

Izračunati $V(\text{obs})$ – statistika testa: $V_n(\text{obs}) = \sum_{k=1}^{n-1} r(k) + 1$, gde je $r(k)=0$ ako je $\varepsilon_k = \varepsilon_{k+1}$, u suprotnom $r(k)=1$.

Izračunavanje izlaznog parametra **P** – rezultat testa:

$$P = \text{erfc}\left(\frac{|V_n(\text{obs}) - 2n\pi(1-\pi)|}{2\sqrt{2n\pi(1-\pi)}}\right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa **P** manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.5 Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku

U ovom testu se posmatra najduže uzastopno ponavljanje jedinica u M -bitnim blokovima. Svrha je određivanje da li se dužina najdužeg uzastopnog ponavljanja poklapa sa dužinom koja bi se očekivala u nizu slučajnih brojeva. Treba imati u vidu da neregularnost u očekivanom broju ponavljanja jedinica implicira i neregularnost u očekivanom broju ponavljanja nula iz čega zaključujemo da je dovoljno proveriti samo ponavljanje jedinica.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ε
- Dužina binarne sekvence n
- M – dužina svakog bloka. Test je podešen tako da za M koristi tri vrednosti u skladu sa dužinom parametra n prema tabeli:

Minimum n	M
128	8
6272	128
750,000	10^4

- N – broj blokova

Ulazna sekvenca se deli na M -bitne blokove.

Kategorizacija frekvencije najdužeg ponavljanja jedinica u blokovima prema tabeli:

v_i	$M = 8$	$M = 128$	$M = 104$
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

$$\text{Izračunati } \chi^2(\text{obs}) - \text{statistiku testa: } \chi^2(\text{obs}) = \sum_{i=0}^K \frac{(V_i - N\pi_i)^2}{N\pi_i}$$

Vrednosti za π_i su prethodno izračunati za odgovarajuće M

Vrednosti za K i N se određuju prema tabeli:

M	K	N
8	3	16
128	5	49
104	6	75

Izračunavanje izlaznog parametra **P** – rezultat testa:

$$P = \text{igamc}\left(\frac{K}{2}, \frac{\chi^2(\text{obs})}{2}\right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa P manja od 0.01, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.6 Ispitivanje stanja binarne matrice

Svrha ovog testa je provera linearne zavisnosti između podnizova fiksne dužine iz originalnog niza.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Duzina binarne sekvence n
- Broj redova u pod-matrici **M**
- Broj kolona u pod-matrici **Q**

Ulazna sekvenca se deli na MxQ neprekidajuće blokove nakon čega se dobija **N** blokova ($N=n/MQ$). Preostali bitovi se odbacuju. Dobijeni MxQ blokovi se ubacuju u MxQ pod-matrice.

Određuje se binarni rang R_l za svaku pod-matricu ($l=1, \dots, N$)

Izračunati $\chi^2(\text{obs})$ – statistiku testa

Izračunavanje izlaznog parametra **P** – rezultat testa:

$$P = e\chi^{2(\text{obs})/2} = \text{igamc}(1, \chi^{2(\text{obs})/2})$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa **P** manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.7 Ispitivanje diskretne Furijerove transformacije

Fokus ovog testa predstavlja vršne vrednosti diskretne Furijerove transformacije testirane sekvence. Svrha ovog testa je da otkrije periodične funkcije, tj. ponavljajuće šablone u testiranom nizu što ukazuje na odstupanje od pretpostavke o slučajnosti.

Namera je da se otkrije da li je broj vršnih vrednosti koje prelaze vrednost praga 95% značajno različit od preostalih 5%.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Duzina binarne sekvence n

Vrednosti ulazne binarne sekvence se pretvaraju u vrednosti -1 i 1 i dobijamo sekvencu X ($X = x_1, x_2, \dots, x_n$ gde je $x_i = 2\epsilon_i - 1$)

Na sekvencu X primenjujemo diskretnu Furijeovu transformaciju i dobijamo S :

$$S = \text{DFT}(X)$$

Izračunavamo M po formuli $M = \text{modulus}(S') \equiv |S'|$, gde je S' pod-sekvencu koja sadrži prvu polovinu S sekvence

Izračunavamo T po formuli $T = \sqrt{(\log \frac{1}{0.05})n}$ što predstavlja vrednost praga koju ne sme preći 95% vrenosti dobijenih ovim testom.

Izračunavamo N_0 po formuli $0.95n/2$ što predstavlja očekivani teorijski broj maksimalnih vrednosti koje su manje od izračunatog praga T

Izračunavamo N_1 što predstavlja stvarno uočen broj vršnih vrednosti u M koje su manje od T

$$\text{Izračunavamo } d \text{ po formuli } d = \frac{(N_1 - N_0)}{\sqrt{\frac{n(0.95)(0.05)}{4}}}$$

Izračunavanje izlaznog parametra P – rezultat testa:

$$P = \text{erfc}\left(\frac{|d|}{\sqrt{2}}\right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa P manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.8 Ispitivanje nepreklapajućih uzoraka

Akcent ovog testa je na broju koji pokazuje koliko se puta ponovila prethodno definisa sekvenca. Test bi trebalo da otkrije da li je broj pojava uzoraka koji se ne preklapaju približno jednak broju koji se očekuje za niz slučajnih brojeva. Ukoliko se testirani uzorak poklopi sa predefinisanim šablonom, okvir koji obuhvata testirani uzorak preskače uzorak, a ukoliko nema poklapanja, okvir se pomera za jedan bit u testiranom bloku.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina šablona za poređenje m
- Duzina binarne sekvence n

- Predefinisana sekvenca dužine m bita - **B**
- Dužina podniza binarne sekvence ε koji se poredi sa $B - \mathbf{M}$
- Broj nezavisnih blokova predefinisana na $8 - \mathbf{N}$

Ulazna sekvenca se deli na $\mathbf{N}$ nezavisnih blokova dužine $\mathbf{m}$.

Definišu se brojači $\mathbf{W}$ za svaki blok $\mathbf{N}$ koji označavaju broj poklapanja sa šablonom $\mathbf{W}_i$, gde je $i = (1, \dots, \mathbf{N})$.

Izračunava se aritmetička sredina $\mu = \frac{(M - m + 1)}{2^m}$ i varijansa

$$\sigma^2 = M \left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}} \right).$$

Izračunati $\chi^2(\text{obs})$ – statistiku testa: $\chi^2(\text{obs}) = \sum_{j=1}^N \frac{(W_j - \mu)^2}{\sigma^2}$.

Izračunavanje izlaznog parametra **P** – rezultat testa:

$$\mathbf{P} = \text{igamc} \left(\frac{\mathbf{N}}{2}, \frac{\chi^2(\text{obs})}{2} \right)$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa **P** manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.9 Ispitivanje preklapajućih uzoraka

Karakteristika koja se posmatra u ovom testu je učestalost pojave svih mogućih m -bitnih uzoraka gde dolazi do preklapanja u celom ispitivanom nizu. Test bi trebalo da otkrije da li je broj pojava uzoraka koji se preklapaju približno jednak broju koji se očekuje za niz slučajnih brojeva. Za razliku od prethodnog testa, ukoliko se poređena sekvenca poklopi sa šablonom, okvir se pomera za jedan bit bez preskakanja.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ε
- Dužina šablona za poređenje $\mathbf{m}$
- Dužina binarne sekvence $\mathbf{n}$
- Predefinisana sekvenca dužine m bita - **B**
- Dužina podniza binarne sekvence ε koji se poredi sa $B - \mathbf{M}$
- Broj nezavisnih blokova predefinisana na $8 - \mathbf{N}$
- Broj stepeni slobode **K** (predefinisano u kodu testa, $K=5$)
- Niz $\mathbf{V}_i$ čiji elementi predstavljaju brojače za poklapanje uzorka sa šablonom

Ulazna sekvenca se deli na $\mathbf{N}$ nezavisnih blokova dužine $\mathbf{m}$.

Vrši se poređenje blokova i ukoliko nema poklapanja, uvećava se $\mathbf{V}_0$, ukoliko ima jedno poklapanje uvećava se $\mathbf{V}_1$..., na kraju ako ima pet poklapanja uvećava se $\mathbf{V}_5$.

Izračunati aritmetičku sredinu $\lambda = \frac{(M - m + 1)}{2^m}$ i $\eta = \frac{\lambda}{2}$

Izračunati $\chi^2(\text{obs})$ – statistiku testa: $\chi^2(\text{obs}) = \sum_{i=0}^K \frac{(V_i - N\pi_i)^2}{N\pi_i}$, vrednosti π_i su

prethodno izračunate, a date su u tabeli tehničkih opisa testova.

Izračunavanje izlaznog parametra **P** – rezultat testa:

$$P = \text{igamc}\left(\frac{K}{2}, \frac{\chi^2(\text{obs})}{2}\right).$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa **P** manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.10 Maurerov univerzalni statistički test

Ovaj test je 1992. godine objavio švajcarski naučnik Ueli Maurer [14]. Test ispituje mogućnosti kompresije testiranog niza bez gubitka informacije. Za niz koji se može značajno kompresovati smatra se da nije niz slučajnih bitova zbog činjenice da je kompresija niza efikasnija ukoliko niz pokazuje periodična svojstva.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina podniza ulazne sekvence $L - 6 \leq L \leq 16$.
- Dužina podniza inicijalnog bloka $Q - Q = 10 \cdot 2^L$
- Dužina binarne sekvence $n - n \geq (Q + K)L$

Ulazna sekvenca se deli na odgovarajući broj inicijalnih, nepreklapajućih blokova Q dužine L i odgovarajući broj testnih, nepreklapajućih K blokova ($K = \frac{n}{L} - Q$) dužine L (višak bitova ulazne sekvence se odbacuje).

Koristeći inicijalizacioni blok, kreira se tabela čije kolone predstavljaju sve moguće vrednosti bita u L bloku. U prvi red tabele upisuju se indeksi podniza L u kojem je pronađeno poklapanje L vrednosti u Q bloku. Redovi tabele se dalje popunjavaju poređenjem vrednosti L podnizova u K blokovima. Izračunava se kumulativna suma distanci između blokova u kojima se vrednosti poklapaju $sum = sum * \log_2(i - T_j)$, gde je i broj bloka, a vrednost u j koloni tabele.

Izračunava se statistika testa: $f_n = \frac{1}{K} \sum_{i=Q+1}^{Q+K} \log_2(i - T_j)$.

Izračunavanje izlaznog parametra **P** – rezultat testa:

$P = \text{erfc}\left(\frac{f_n - \text{expectedValue}(L)}{\sqrt{2\sigma}}\right)$, gde su vrednosti za **expectedValue(L)** i σ unapred

poznate i nalaze se u odgovarajućim tabelama.

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa **P** manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.11 Ispitivanje linearne kompleksnosti

Ovaj test se koristi za dobijanje dužine linearnog pomeračkog registra (eng. *Linear feedback shift register* - LFSR). Svrha ovog testa je utvrđivanje da li je sekvenca dovoljno kompleksa da se može smatrati slučajnom.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina podniza ulazne sekvence M - $500 \leq M \leq 5000$
- Dužina binarne sekvence n - $n \geq 10^6$
- Broj stepeni slobode K – predefinisano u testu $K=6$
- Broj podnizova dužine M bita N - $N \geq 200$

Ulazna sekvenca se deli na N broj podnizova dužine M bitova ($n=NM$). Upotrebom Berlekamp-Massey algoritma određuje se linearna kompleksnost L_i svakog N bloka, gde je $i=1, \dots, N$.

Izračunava se $\chi^2(obs) = \sum_{i=0}^K \frac{(v_i - N\pi_i)^2}{N\pi_i}$, gde su vrednosti π_i unapred izračunate i

date u odgovarajućoj tabeli.

Izračunavanje izlaznog parametra P – rezultat testa:

$$P = igamc\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right).$$

PRAVILO ODLUČIVANJA:

Ukoliko je vrednost testa P manja od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.12 Serijski test

Test se koristi za utvrđivanje učestalosti svih mogućih preklapanja m -bitnog šablona u čitavoj testiranoj sekvenci.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina binarne sekvence n
- Dužina podniza m - $m < |\log_2 n| - 2$

Na kraj ulazne sekvence dodajemo $m-1$ bita sa početka ulazne sekvence. U indekse niza V smeštamo broj ponavljanja određenog šablona m . Za svaku vrednost parametra m postoji poseban niz v i izračunava se ψ_m^2 po formuli:

$$\psi_m^2 = \frac{2^m}{n} \sum_{i_1 \dots i_m} v_{i_1 \dots i_m}^2 - n, \text{ na isti način i } \psi_{m-1}^2 \text{ i } \psi_{m-2}^2.$$

$$\text{Izračunavamo } \nabla \psi_m^2 = \psi_m^2 - \psi_{m-1}^2 \text{ i } \nabla^2 \psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2.$$

Izračunavanje izlazne parametre P_1 i P_2 – rezultati testa:

$$P_1 = igamc(2^{m-2}, \nabla \psi_m^2),$$

$$P_2 = igamc(2^{m-3}, \nabla^2 \psi_m^2)$$

PRAVILO ODLUČIVANJA:

Ukoliko su vrednosti testa P manje od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.13 Ispitivanje približne entropije

U ovom testu posmatra se takođe učestalost pojave svih mogućih preklapajući m -bitnih uzoraka u nizu. Cilj je poređenje učestalosti preklapajućih blokova sa očekivanim rezultatima.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina binarne sekvence n
- Dužina podniza m - $m < \lfloor \log_2 n \rfloor - 5$

Na kraj ulazne sekvence dodajemo $m-1$ bita sa početka ulazne sekvence. Izračunavamo broj ponavljanja svake podsekvence i rezultate smeštamo u niz C^m_i , gde je i vrednost bita u m okviru.

Za svaku vrednost i , izračunavamo $C^m_i = \frac{\#i}{n}$.

Izračunavamo $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i * \log \pi_i$, gde je $\pi_i = C^m_j$, a $j = \log_2 i$. Uvećavamo m za jedan i

ponavljamo prethodne korake kako bismo izračunali $\varphi^{(m+1)}$.

Izračunavamo statistiku testa: $\chi^2 = 2n[\log 2 - ApEn(m)]$, gde je $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$.

Izračunavanje izlaznog parametra P – rezultati testa:

$$P = \text{igamc}(2^{m-1}, \frac{\chi^2}{2}).$$

PRAVILO ODLUČIVANJA:

Ukoliko su vrednosti testa P manje od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

3.4.14 Ispitivanje slučajnih zbirova

Svrha ovog testa je potreba da se utvrdi da li je kumulativni zbir parcijalnih sekvenci koje se javljaju u nizu koji se testira u mnogome odstupa u odnosu na vrednosti kumulativnog zbira očekivane kod slučajnih sekvenci.

OPIS TESTA:

Ulazni parametri za testiranje su:

- Binarna sekvenca ϵ
- Dužina binarne sekvence n
- Smer prolaska kroz sekvencu prilikom sabiranja **mode** (0 – unapred, 1 – unazad)

Ulazna sekvenca se konvertuje u niz koji sadrži vrednosti 1 i -1 ($X_i = 2\epsilon_i - 1$). Sabiramo vrednosti elemenata niza i memorišemo maksimalnu apsolutnu vrednost koju ćemo koristiti za izračunavanje statistike testa.

Izračunavamo statistiku testa z , $z = \max_{1 \leq k \leq n} |S_n|$.

Izračunavanje izlaznog parametra P – rezultati testa:

$$P = 1 - \sum_{k=\frac{(-n/z+1)}{4}}^{\frac{(n/z-1)}{4}} [\Phi((4k+1)z) - \Phi((4k-1)z)] + \sum_{k=\frac{(-n/z-1)}{4}}^{\frac{(n/z-3)}{4}} [\Phi((4k+3)z) - \Phi((4k+1)z)] \text{ gde}$$

je Φ standardna normalna funkcija $\Phi_{(z)} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-u^2/2} du$.

PRAVILO ODLUČIVANJA:

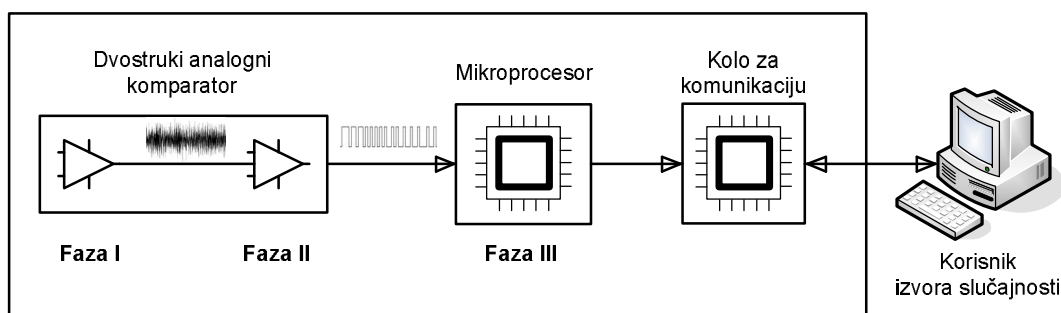
Ukoliko su vrednosti testa **P** manje od **0.01**, testirana sekvenca nije slučajna, u suprotnom, sekvenca je slučajna.

4 Pregled predloženog rešenja

U ovom poglavlju biće predstavljen elektronski sklop koji je zasnovan na principima upotrebe elektronskog šuma kao izvora slučajnosti. Rad sklopa će biti objašnjen preko generičkih šema, a zatim će biti objašnjena i mogućnosti primene ovog elektronskog sklopa u kriptografiji. Nakon toga, biće prikazane performanse sklopa i rezultati dobijeni nakon nekoliko statističkih testova kojima će generisane slučajne vrednosti biti podvrgnute.

4.1 Generička šema predloženog rešenja

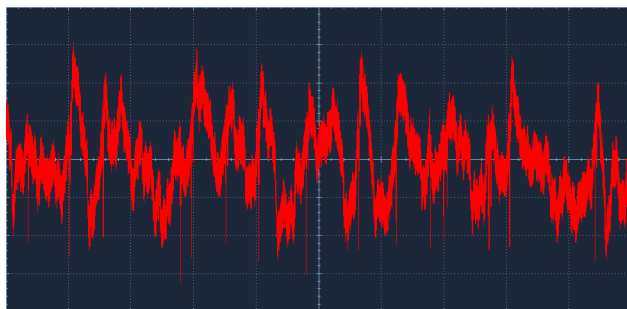
Razvijeno rešenje se sastoji iz tri faze: generisanje signala šuma, pojačavanje i AD konverzija signala šuma i na kraju uzorkovanje digitalnog signala, kontrola kvaliteta uzoraka i naknadno procesiranje.



Slika 4.1-1 Blok šema uređaja

Faza I – Generisanje signala šuma

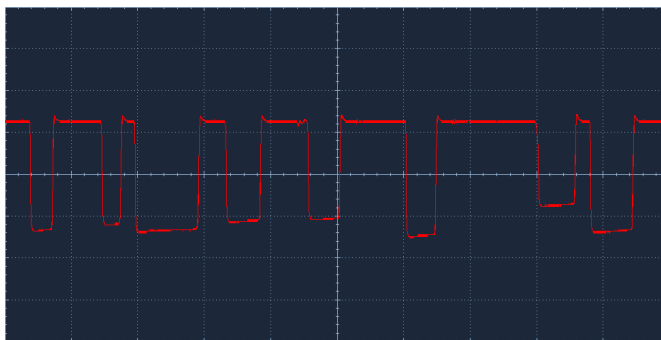
U prvoj fazi, analogni komparator je na nestandardan način, koji će biti detaljno opisan u nastavku rada, doveden u stanje u kojem se manifestuju haotične oscilacije koje za rezultat imaju signal po karakteristikama sličan signalu belog šuma. Izborom odgovarajućih komponenti u elektronskom sklopu, postiže se konzistentno generisanje šuma u opsegu bliskom graničnim frekvencijama komparatora. Vrednosti upotrebljenih elektronskih komponenti u sklopu nemaju teorijsku potporu već se do istih došlo eksperimentalnim putem i testiranjem.



Slika 4.1-2 Analogni signal šuma na ulazu prvog dela komparatora

Faza II – Pojačavanje signala i AD konverzija

U drugoj fazi, analogni komparator dodatno pojačava i vrši konverziju analognog signala šuma u digitalni signal. U ovom delu sklopa, komparator radi u standardnom režimu i poredi signal šuma sa referentnim naponom koji je pomoću razdelnika napona postavljen na polovinu napona napajanja ($V_{CC} = 5V$, $V_{ref} = 2.5V$). Ako je napon na invertujućem ulazu veći od referentnog napona na ne-invertujućem ulazu komparatora, napon na izlazu komparatora će biti jednak naponu napajanja (logička jedinica). U suprotnom slučaju, napon na izlazu komparatora će biti jednak nuli (logička 0).

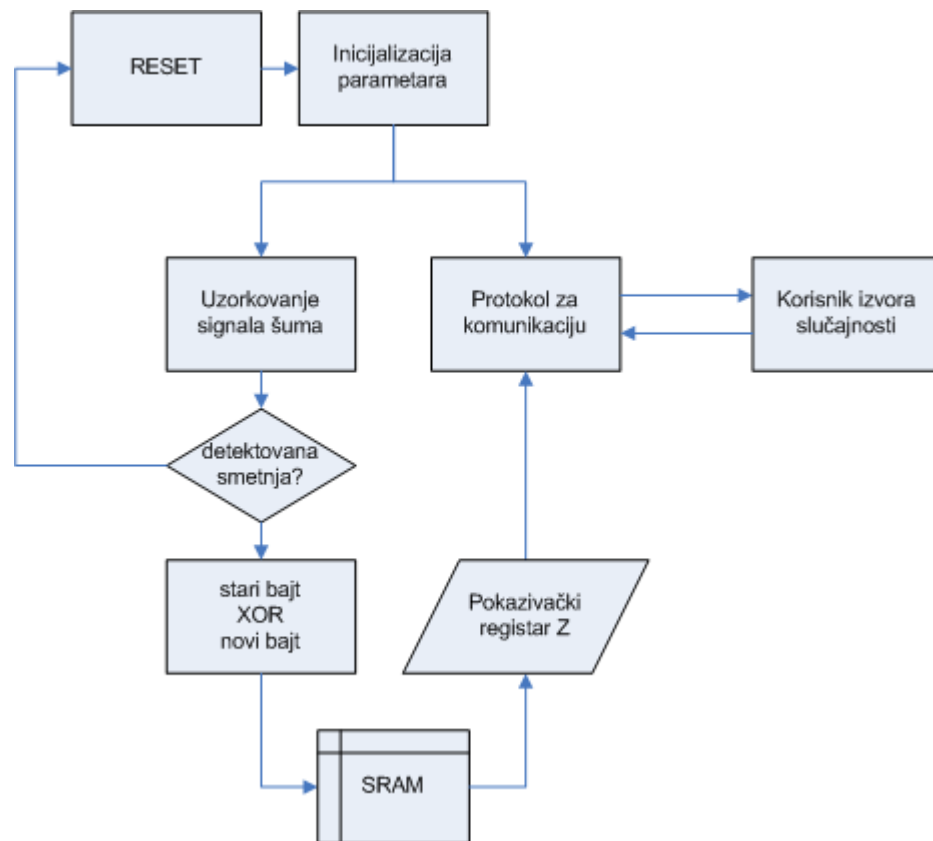


Slika 4.1-3 „Digitalni šum“ koji se dovodi na ulaz mikroprocesora

Faza III – Uzorkovanje digitalnog signala, kontrola kvaliteta uzorka i naknadno procesiranje

U trećoj fazi, digitalni signal promenljive širine impulsa, dovodi se na odgovarajući ulaz mikroprocesora. Na osnovu instrukcija zadatih u programskom kodu koji se izvršava u mikroprocesoru, signal na ulazu se uzorkuje maksimalnom mogućom učestanošću koja zavisi od izabranih postavki unutrašnjeg oscilatora i karakteristika samog mikroprocesora, a u ovom slučaju je 16MHz. Ova vrednost unutrašnjeg oscilatora nam omogućava da pored čestog uzorkovanja signala šuma paralelno implementiramo i proveru kvaliteta dobijenih uzoraka i kontrolu komunikacionog protokola između izvora i korisnika slučajnosti. Koristeći vrednost brzog internog brojača u mikrokontroleru, proverava se učestalost promena vrednosti uzorkovanog bita. Ukoliko se vrednosti ponavljaju to se tretira kao greška i izvršava se procedura restarta u kojoj se pored inicijalizacije odgovarajućih registara ponovo osvežava kompletna memorija od 512 bajta za skladištenje slučajnih vrednosti. Na ovaj način smo obezbedili nove slučajne vrednosti jer postoji teoretska mogućnost da SRAM memorija i nakon gubitka napajanja zadrži prethodne vrednosti. Komunikacioni protokol neće omogućiti isporuku slučajnih vrednosti korisniku istih ukoliko je detektovana anomalija nakon uzorkovanja ili je skladište slučajnosti prazno. Kada se detektuje promena vrednosti uzorkovanog signala, tj. prelazak iz „0“ u „1“ ili obrnuto, u 8-mo bitni registar se upisuje vrednost bita sa najmanjom težinom iz registra brojača. Ova procedura se ponavlja dok se ne formira kompletan bajt nakon čega se primenjuje se XOR logička operacija nad vrednošću dobijenog bajta i vrednošću bajta koju smo poslednju upisali u SRAM, inkrementira se pokazivač memorijske lokacije SRAM-a, a rezultat XOR operacije se upisuje u novu lokaciju u memoriji. Bitno je napomenuti da se u svakom trenutku izvršavanja koda u

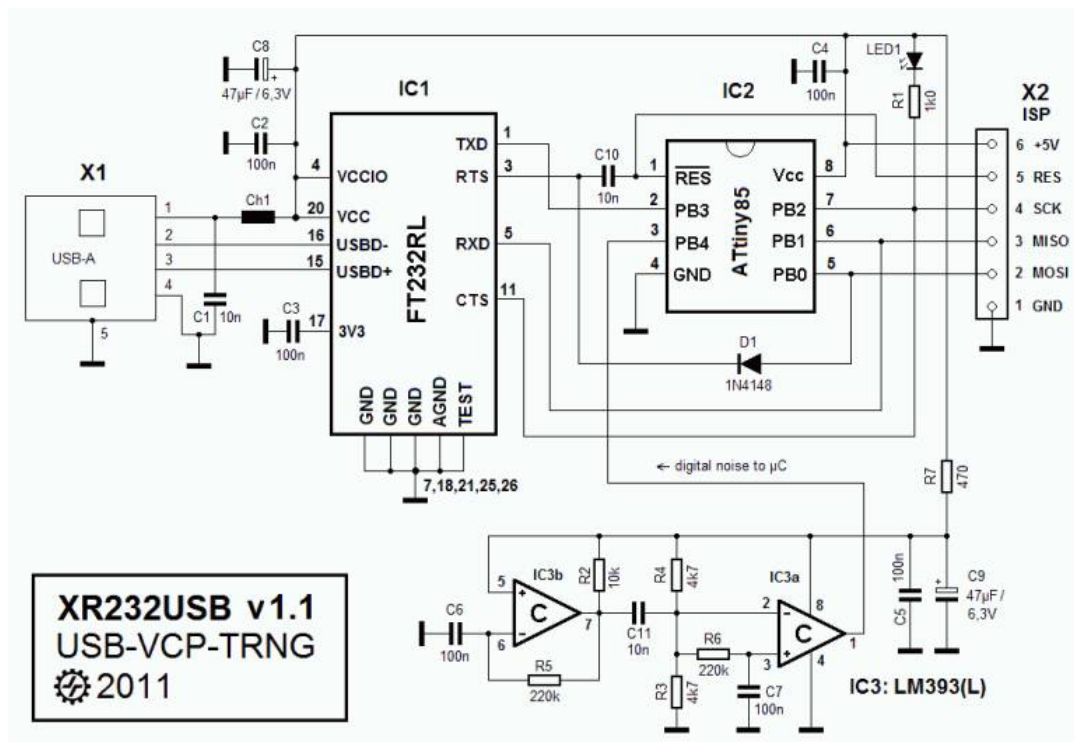
mikroprocesoru, takoreći paralelno obavlja proces uzorkovanja i obrade signala šuma, kontrola kvaliteta dobijenih vrednosti u cilju zaštite korisnika slučajnosti i na kraju i obrada signala komunikacionog protokola. Na slici 4.1-4 je prikazana uprošćena šema rada programskog koda uređaja.



Slika 4.1-4 Uprošćena šema programskog koda uređaja

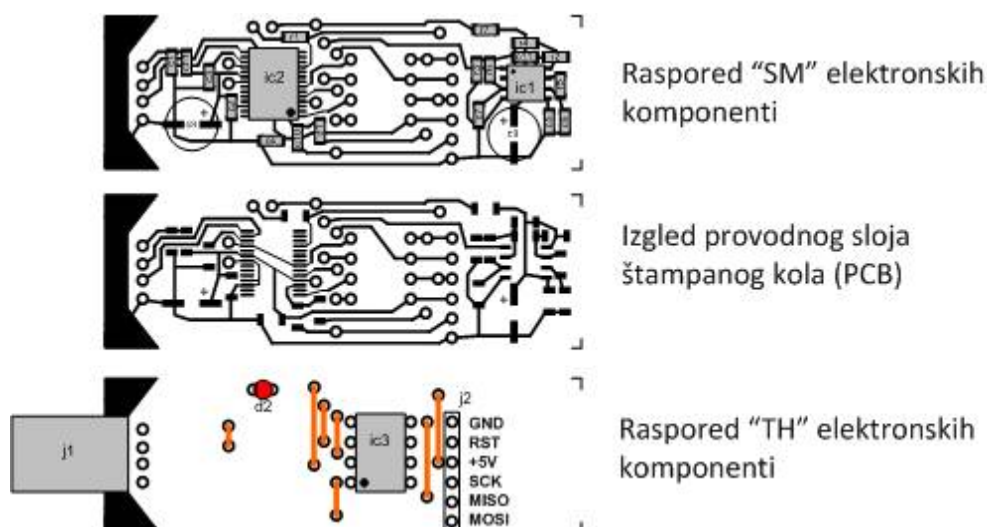
4.2 Opis izrade generatora slučajnosti

Generator slučajnosti je realizovan upotrebom diskretnih elektronskih komponenti. Sa ciljem smanjenja dimenzija uređaja izabrane su komponente za površinsku montažu (SMD – surface mount device). Štampana ploča (engl. PCB – printed circuit board) je izrađena na jednoslojnoj pertinaks ploči dimenzija 6 x 2 centimetra. Električna šema na osnovu koje je izrađena štampana ploča prikazana je na slici 4.2-1.



Slika 4.2-1 Električna šema generatora slučajnosti

U trenutku izrade uređaja, na domaćem tržištu nije bio dostupan odgovarajući mikrokontroler u SMD izvedbi pa je štampano kolo prilagođeno DIP8 varijanti istog. Na štampanoj ploči se nalazi i 6-polni konektor koji predstavlja interfejs za pristup mikrokontroleru (engl. ISP - In-System Programming). Ova opcija omogućava povezivanje sa odgovarajućim uređajem za programiranje sa ciljem lakšeg pristupa mikrokontroleru u slučaju reprogramiranja istog. Na slici 4.2-2 prikazan je izgled štampanog kola sa rasporedom i oznakama okmponenti, a u tabeli 4.2-1 prikazane su vrednosti i količine upotrebljenih elektronskih komponenti.



Slika 4.2-2 Izgled štampane ploče sa rasporedom komponenti

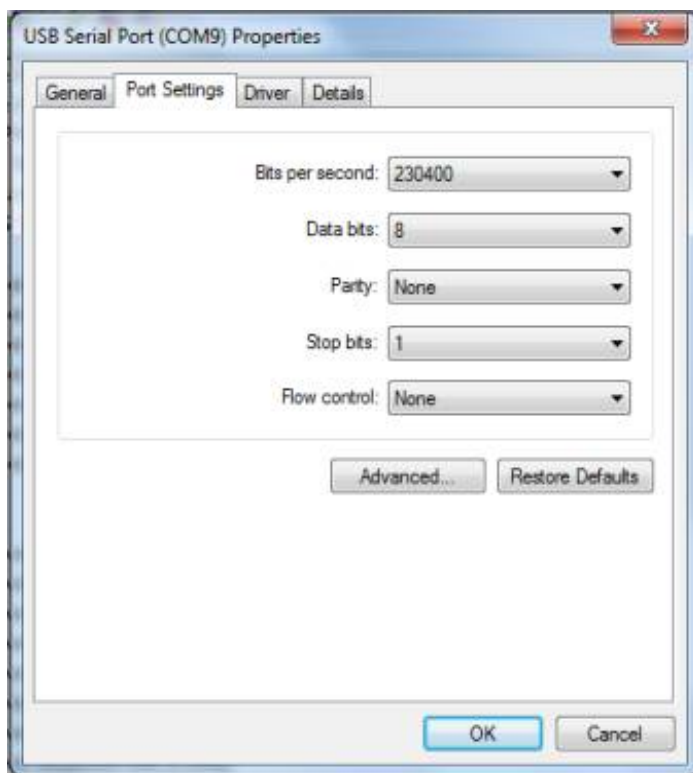
R.br.	Oznaka u el. šemi	Tip/Vrednost	Količina
1	IC1	LM393	1
2	IC2	FT232RL	1
3	IC3	Attiny85-20PU	1
4	D1	1N4148	1
5	D2	LED	1
6	J1	USB f	1
7	J2	6pin f	1
8	C1, C10, C11	10nF	3
9	C2, C3, C4, C5, C6, C7	100nF	6
10	C8, C9	47uF/6.3V	2
11	R1	1K	1
12	R2	10K	1
13	R3, R4	4K7	2
14	R5, R6	220K	2
15	Ch1 (ferite bead)	0 ohm	1

Tabela 4.2-1 Vrednosti i količine upotrebljenih elektronskih komponenti

Ukupna cena komponenti upotrebljenih za izradu ovog uređaja je oko 1500,00 RSD, tj. maksimalno 15 EUR ako uračunamo i sav potrošni materijal. Imajući u vidu dobre statističke karakteristike generisanih vrednosti i jednostavnost upotrebe uređaja, moglo bi se razmisliti o ideji komercijalizacije istog, a u tom slučaju bi se sa proizvodnjom daleko većeg broja jedinica mogao očekivati i drastičan pad osnovnih troškova izrade.

4.3 *Postavka i objašnjenje eksperimentalnog okruženja*

Generator slučajnosti se priključuje na USB interfejs računara. Operativni sistem računara, uređaj automatski detektuje kao serijski komunikacioni port bez potrebe za instaliranjem posebnih upravljačkih programa, tačnije, upravljački programi su u velikom broju slučajeva već sastavni deo većine operativnog sistema. Potrebno je podesiti parametre komunikacionog porta kao u primeru prikazanom na slici 4.3-1.



Slika 4.3-1 Parametri za podešavanje komunikacionog porta (Windows OS)

Testiranje rada uređaja i prikupljanje generisanih vrednosti vrši se pomoću namenske aplikacije, slanjem odgovarajuće komande uređaju putem serijskog komunikacioni port. Aplikacija omogućava nekoliko režima za prikaz ili prikupljanje vrednosti koje generiše izvor slučajnosti. Prvi režim prikazuje generisane vrednosti u vidu niza bajtova koji se prikazuju na monitoru računara na kojem je pokrenuta aplikacija. Drugi režim rada, u posebnoj formi, na monitoru računara, prikazuje generisane vrednosti konvertovane u mapu bitova rezolucije 512x384 piksela. Treći režim rada, u posebnoj formi, prikazuje histogram kumulativnih vrednosti generisanih bajtova. Četvrti režim rada, upisuje generisane bajtove u datoteku a samim tim omogućeno je i naknadno testiranje dobijenih uzoraka korišćenjem NIST baterije testova.

4.3.1 Režim rada 1

Komanda: XR232NT COM9:230400 -h

```

C:\Windows\system32\cmd.exe

C:\Users\Igor\Desktop\software>xr232nt com9:230400 -h

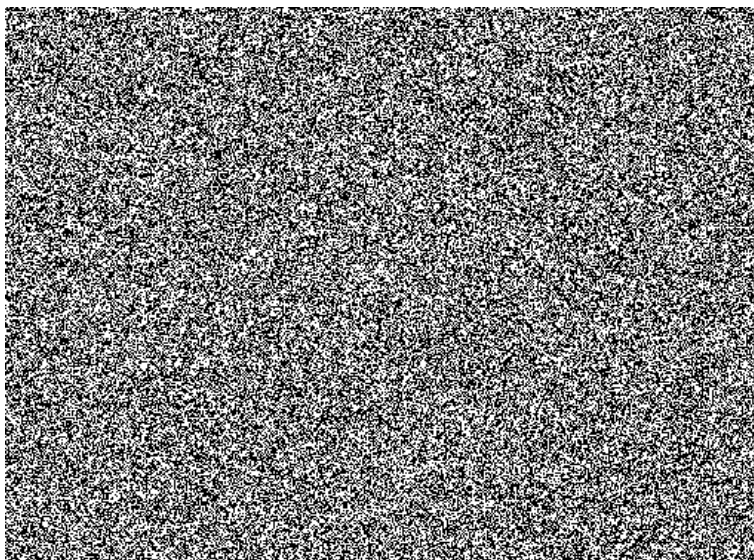
COM9:230400
D5990EBE6DC98BD7B5BD00F00E1B49918D387D66F428C1AFA80CDF31E4EAC45E247E2E2FC99686C
FDCB1728E56097EC2A642CF787B534ABR3DBF8E77E782A755DFC6FA1617A9E9EC2C13DE33597811
FECFEC22058461EF2DEC0714FC5B6DDDB244963291DE74C5160054138A2A6D6CA9B3CB0B1448477
26491C5815E2BBB1181681AACFDACEAFA8FF6A8A9EB2C7942B30FF796F3A3563ABBCD5F99632
703669A340769E389924402E76AEDB1F03F1F3C888BBA6211BA76CDB714EB89B827729A0C4906B14
F61AB61877A90749F94F51960AC52950EA73486BCE8892124FC8BDDBF18D269CA7DB05E8FF8C81A
1F0A332BD265382CC92FDCAB037D7358527576DF44FPA8C2CAD94B45F815C016BB0E8773505030F
36A919B7D05CC349DD86117C617298DC3F1F50751P7DEE33F6C7D7E04824BDD3F99BBE97735A62
5C44E79B608162F98528EC39F0FD85C15C37DB617E9B578EF853DFE0E71D5859257B3DE2AA7B96A
A95B076096CBEF7D862B7D3F0491ED8F8125E970E0604CAAF5AD017EEAF580DEB73035437C178C72
E9362F0B2DB77152AC54B33857A93C191E5248700569D3E21D4C300439B11085C7FEF5BC956F235C
619E135292F999A9AC52C3892CCB178228A8FDC8D06BC82A872C5B60EB1A67206E292FEB64876E
510B3FC82C41A5E6EECD879B52063B0CEB71EAA1602B236D156A2C0D5C7C2CE781C58475D1CF62
4959888CF9502837D20D744540ACC4201C11A8BA6A385E268E8A49AEF9B5F9581AA7D95F8F8C2849
4FA6B038620DF40FDFCBA802F9CCC4E68189D8A6106FEF450375070C46291A427556B171B05F5E9
D3332626818A3A172D4EF28719B63F1A0FAF0CD3B3D53C6B8E3A7E54596B5D37F1A1EC4C245EA72
348CB1F3984D1B80D9D64C6AE6A0CAB9CD2B2F8A7816ED6A7B6AF0FCE88DA76D3DE7EEF7C00A8C4E
D053B5D4F5F7D500455D29D7D58135309F00AD31B059BBA34985173A1494528D957648BFC50
A340BDA8B2FDB0D2DA10DFF58B8E78B5468598BDA93F17E31F8472239FD2B8A914A9E3591A6496
2762FC7371FDF1D0F1B953ADD7A28DD354AE04BD4ED9966AB7CCF0EEE1CDD446240FD1D4F042E870
8C515DB707BA292A37C51F40A714F96F68FEDB8539D52B2B828AC337F099B1EB29D618FCF031B138
4282D117E1AACDF57D0208799DB31468B440EB4CA20F8F9C31414250CE7F283B77C0205C8942306
1A937E84539BA156AA145AFA0D62875BC6D8C95BBD5C3ED787C586B5DC5FC3145D90BB890AC156F
60240CC0A8B764009A8B8366AFC427E539267A0CAB7B6C3935220454C87C69C9D062085728B7
2779AEDF5DDEC11B26290148734F33DD9F1A778F6B8F00E21CA0EACF3406F72628CA931F607C3
C4412171C7BB2FFB42137110F1416E85F2B8E0D4FD3F44272112C8EA9E9EEB6A78B910D2C20C94D
0F7B9BD7A06445E5FA929C3F09DD722950DB4DF53FC6167009D0251F6FA581F0A454A0FAEAE0DD
42B6E17CB13BD98A8C36421168E78E725494CA7D5E76725F49A99ACE9230B48DE7FB9PB0C887CD2E
827B37B921878EE1C46B7P727E5A6DD03670278C44464DFDA0B0FED1062E89FBB75BA0F5DD312FB6
4B751A6189EB02A100286CE30FBD88FA101481ECD5983605EE1E6229FA8A4C8947F7B7A35DF7F83817
26D9B1859BA1DDF27CC7A8799FA8C6C8ECB2768EF85E640BF3074D748FBF296282C96B1F7D4CCA75
358287E4B74C40810039486CC73DE9B7C5294B984EA11A9A890967FEE67879A2ADD3FF8F363543
4FDF0F78DB28961B31A0E52574FF63B2AE4666719C51D60EBC09C42780A80E67B89E79239FAC0
A2FF50D4B3630BF780A031D0A2E4767358198CDA020710815CD7B5F97060561300D8F5EB09811EE
794CB963BDB4FD3998B4C77A4100214F109C58691D2AB70670025C6BCEA6BE76959664C37291F916
21B8A01AEDB5A8A32CA89786B852840035176682F9369810178D5515224DAC545CCFE164BD033150
0DFA9EC7949FEA0ADC8434EA88D4B50209EAD995D6D5C2348F051D75813A28E0B90F762D4B08C915
225B20D4D5C7878E7B25DE6C10978ADEFE0D7EB52885213E245BA2076EES1F722F30E587DC66FDD9
A1D6F60FF7FC043FEED95B37D123B6932EEFEC97B05EEF7E379458A78C9D418D272B2AD11D810
9329CE124C00F27868E7A657A280E7B37142C9C4672PB6E6C667B06FB900DD8242EA4FE15C0FD
0855DD31AB271D886669596D855EEB8AC6BA612E0A281FB1757947E70C6EEBA0021D4AD0E7D7C76
CEE176362910A152F95DD83883E1B6C61F348AAD3F713EBEAA0DDE92C8819F3363618P73FDD0B13F
620958B8D72F53804722A7848D98A6E991E178964C3DC1B63BFF7674E85C3D7BDDA7745C76D2FFDE
C:\Users\Igor\Desktop\software>

```

Slika 4.3-2 Kontinualni ispis generisanih bajtova na monitoru

4.3.2 Režim rada 2

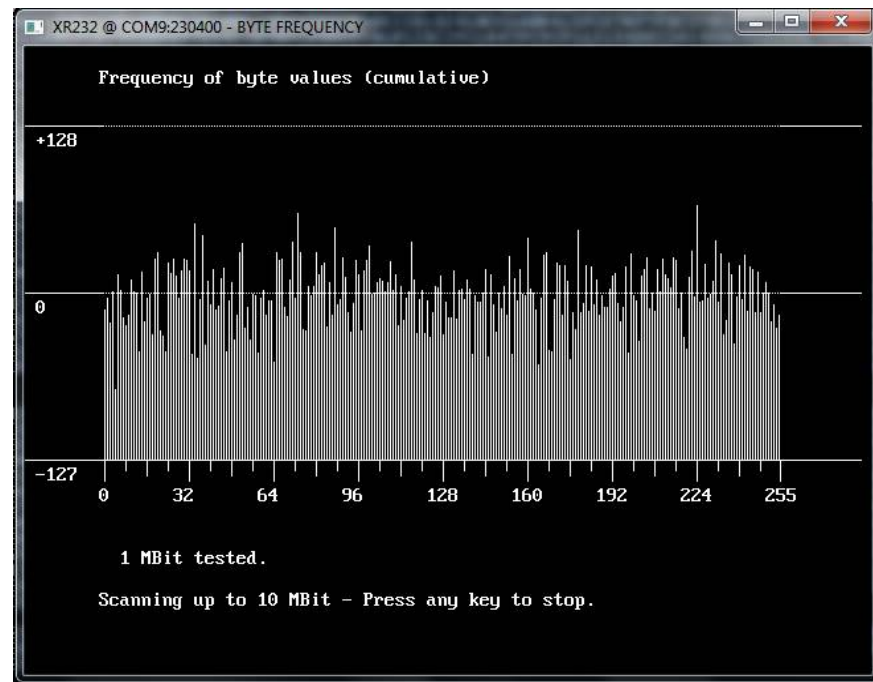
Komanda: XR232NT COM9:230400 -b



Slika 4.3-3 Prikaz generisanih bajtova pomoću bitmape 512x384 piksela

4.3.3 Režim rada 3

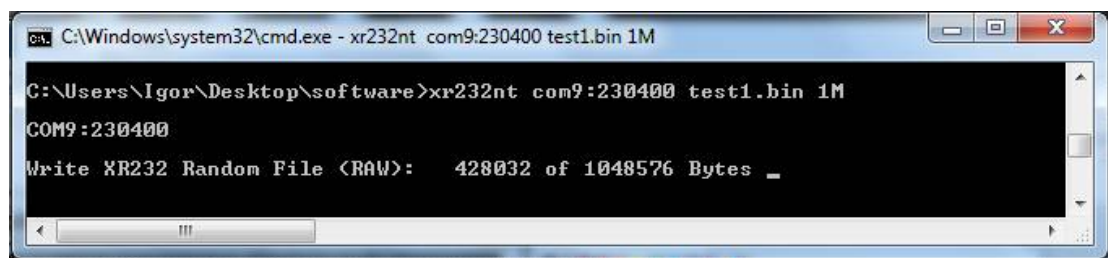
Komanda: XR232NT COM9:230400 -f



Slika 4.3-4 Histogram kumulativnih vrednosti frekvencije generisanih bajtova

4.3.4 Režim rada 4

Komanda: XR232NT COM9:230400 test1.bin 1M



Slika 4.3-5 Upisivanje generisanih bajtova u datoteku

4.4 *Performanse predloženog okruženja i prikaz eksperimentalnih rezultata*

Programski kod koji se izvršava u mikroprocesoru uređaja koncipiran je tako da u svakom trenutku paralelno obrađuje dva procesa. Glavna uloga mikroprocesora je uzorkovanje digitalizovanog signala šuma maksimalnom mogućom frekvencijom od 16MHz, tj. vrednost signala na ulazu u mikrokontroler se uzorkuje više od milion puta u sekundi. Obzirom da je prosečna frekvencija signala šuma merena frekvenc-metrom oko 140kHz, može se zaključiti da je naš izvor slučajnosti sposoban da generiše oko 17KB podataka u sekundi. Ako uzmemo u obzir da se uzorkovane vrednosti dodatno obrađuju i da mikroprocesor koristimo i za izvršavanje instrukcija koje se odnose na komunikacioni protokol sa računarom tj. aplikacijom koja koristi generisane podatke, realna brzina generisanja pada na 9KB/s što je i potvrđeno prostim merenjem vremena potrebnog za generisanje i upis u datoteku 1MB podataka koje genriše izvor slučajnosti.

$$\frac{1MB}{113s} = \frac{1024KB}{113s} = 9,062KB / s = \mathbf{72\ kb/s}$$

U nastavku će biti prikazani rezultati nekoliko statističkih testova urađenih sa uzorkom od 16KB. Ova količina je odabrana sa ciljem da rezultate testova možemo uporediti sa rezultatima testova urađenih na podacima preuzetim sa sajta [15], gde postoji ograničenje u veličini uzorka i ono iznosi 16KB za fajl sa sirovim bajtovima. Testiranje je izvršeno pomoću Java aplikacije razvijene na Univerzitetu Singidunum koja implementira nekoliko najvažnijih NIST testova.

4.4.1 Test ispitivanja učestanosti u nizu

Statistical Tests

File Edit

Input From File: Manually Input:

Generate Random Sequence
Number of bits: 8388603 Java Secured Ra...

Tests
Frequency Test Serial Test Poker Test Runs Test Maurer Entropy Test

NIST Frequency (Monobit) Test:

Sum Observed: 0.7733980419227863
P-Value: 0.43928684844454247
P-Value (Decimal): 0.43928684844454246594125

CryptTool/Christian Schiestl Frequency (Monobit) Test:

Zeros: 65396
Ones: 65676
X1: 0.59814453125
P-Value: 0.3824913014660403
P-Value (Decimal): 0.38249130146604032320922

Minimum 100 bits. Minimum 10000 bits.

Status: Using Array from File for Tests. Read 131072 bits.

Slika 4.4-1 Rezultati testa „ispitivanje učestanosti“ – uzorak TRNG

Statistical Tests

File Edit

Input From File: Manually Input:

Generate Random Sequence
Number of bits: 8388603 Java Secured Ra...

Tests
Frequency Test Serial Test Poker Test Runs Test Maurer Entropy Test

NIST Frequency (Monobit) Test:

Sum Observed: 2.1047475283755825
P-Value: 0.03531329087107253
P-Value (Decimal): 0.03531329087107253072996

CryptTool/Christian Schiestl Frequency (Monobit) Test:

Zeros: 65917
Ones: 65155
X1: 4.429962158203125
P-Value: 0.02068978594218813
P-Value (Decimal): 0.02068978594218812833149

Minimum 100 bits. Minimum 10000 bits.

Status: Using Array from File for Tests. Read 131072 bits.

Slika 4.4-2 Rezultati testa „ispitivanje učestanosti“ – uzorak random.org

4.4.2 Serijski test

The screenshot shows the 'Statistical Tests' application window. The 'Serial Test' tab is selected. The 'Test' button is highlighted. The results for the NIST Serial Test (m=3) and the CryptTool/Christian Schiestl Serial Test are displayed.

NIST Serial Test (m=3):		CryptTool/Christian Schiestl Serial Test:	
000s: 16459	001s: 16332	Number Of 00s: 32791	
010s: 16216	011s: 16389	Number Of 01s: 32605	
100s: 16333	101s: 16273	Number Of 10s: 32605	
110s: 16389	111s: 16681	Number Of 11s: 33070	
P-Value1: 0.39033840952607723		X2: 3.8229936305142473	
P-Value2: 0.8587978815920743		P-Value: 0.07392945170728989	
P-Value1 (Decimal): 0.390338409526077234		P-Value (Decimal): 0.07392945170728988657504	
P-Value2 (Decimal): 0.858797881592074263			
Minimum: $m < \log_2(n)-2$ m=3, Minimum 17 bits.		Minimum 128 bits.	

Status: Using Array from File for Tests. Read 131072 bits.

Slika 4.4-3 Rezultati serijskog testa – uzorak TRNG

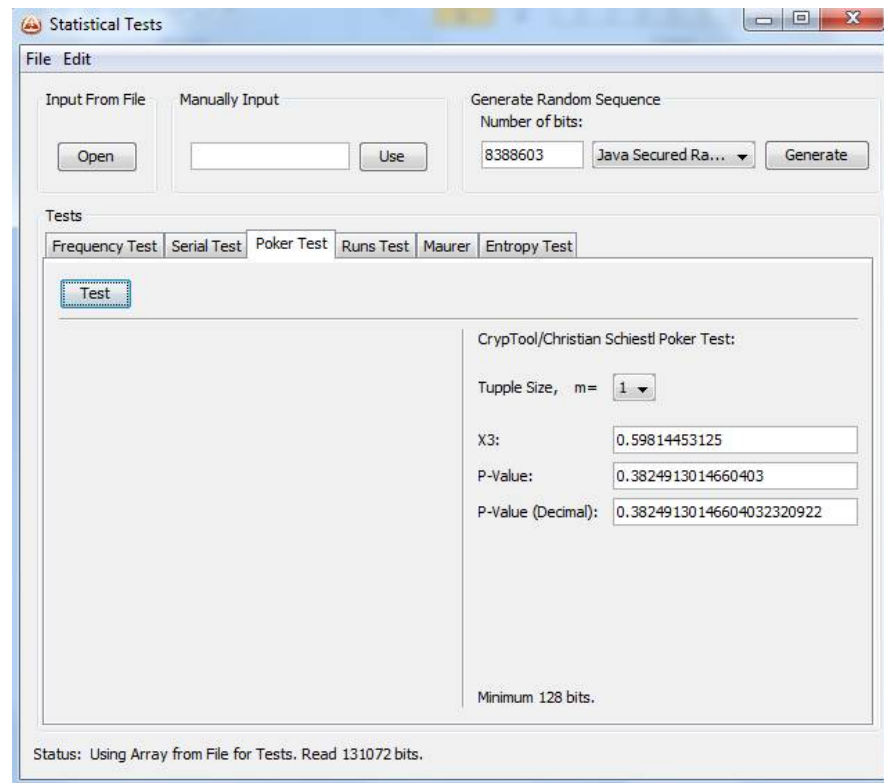
The screenshot shows the 'Statistical Tests' application window. The 'Serial Test' tab is selected. The 'Test' button is highlighted. The results for the NIST Serial Test (m=3) and the CryptTool/Christian Schiestl Serial Test are displayed.

NIST Serial Test (m=3):		CryptTool/Christian Schiestl Serial Test:	
000s: 16770	001s: 16532	Number Of 00s: 33301	
010s: 16308	011s: 16307	Number Of 01s: 32615	
100s: 16533	101s: 16082	Number Of 10s: 32615	
110s: 16307	111s: 16233	Number Of 11s: 32540	
P-Value1: 0.08787409630113242		X2: 7.2550253676308785	
P-Value2: 0.6651177823489514		P-Value: 0.013291110317984831	
P-Value1 (Decimal): 0.087874096301132423		P-Value (Decimal): 0.01329111031798483125165	
P-Value2 (Decimal): 0.665117782348951358			
Minimum: $m < \log_2(n)-2$ m=3, Minimum 17 bits.		Minimum 128 bits.	

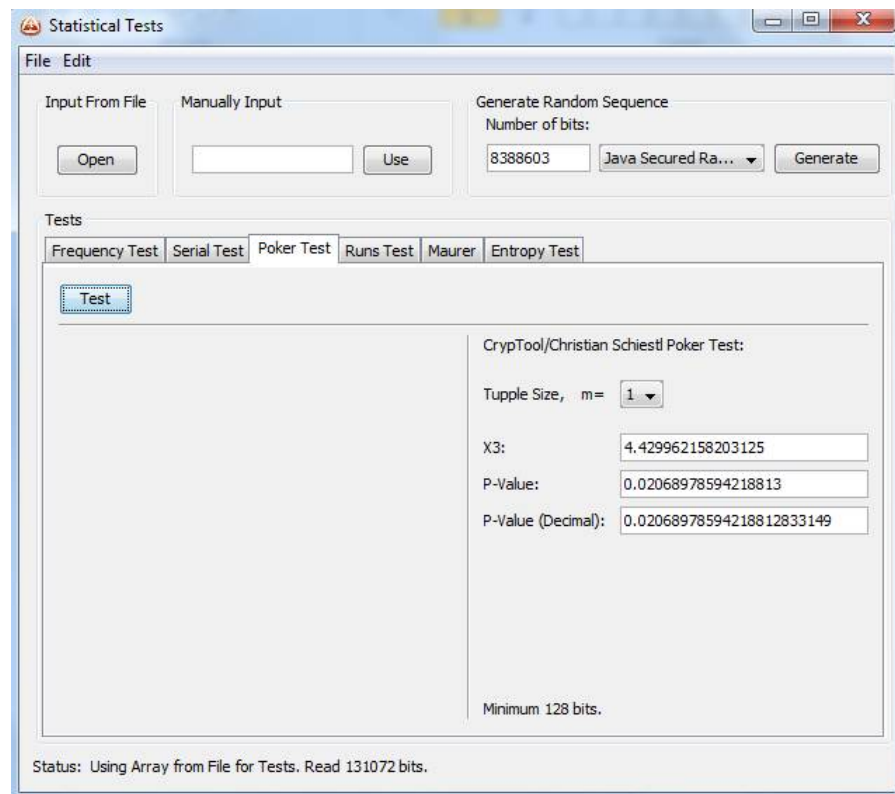
Status: Using Array from File for Tests. Read 131072 bits.

Slika 4.4-4 Rezultati serijskog testa – uzorak random.org

4.4.3 Poker test



Slika 4.4-5 Rezultati poker testa – uzorak TRNG



Slika 4.4-6 Rezultati poker testa – uzorak TRNG

4.4.4 Test uzastopnih ponavljanja iste vrednosti

Statistical Tests

File Edit

Input From File: Manually Input:

Generate Random Sequence
Number of bits: 8388603 Java Secured Ra...

Tests
Frequency Test Serial Test Poker Test **Runs Test** Maurer Entropy Test

NIST Runs Test:

pi: 0.501068115234375
tau: 0.005524271728019902
Status: Applicable.
V-Observed: 65211
P-Value: 0.07285398445992658
P-Value (Decimal): 0.07285398445992657956438

Minimum 100 bits.

CrypTool/Christian Schiestl Runs Test:

Length of Blocks/Gaps, k= 3
Expected number of 1bit Blocks/Gaps: 16384.25
Expected number of 2bit Blocks/Gaps: 8192.0625
Expected number of 3bit Blocks/Gaps: 4096.0
X4: 6.3710315308524414
P-Value: 0.06587154414441267
P-Value (Decimal): 0.06587154414441266647717

Minimum 128 bits.

Status: Using Array from File for Tests. Read 131072 bits.

Slika 4.4-7 Rezultati testa uzastopnih ponavljanja istih vrednosti – uzorak TRNG

Statistical Tests

File Edit

Input From File: Manually Input:

Generate Random Sequence
Number of bits: 8388603 Java Secured Ra...

Tests
Frequency Test Serial Test Poker Test **Runs Test** Maurer Entropy Test

NIST Runs Test:

pi: 0.49709320068359375
tau: 0.005524271728019902
Status: Applicable.
V-Observed: 65231
P-Value: 0.09438178917535944
P-Value (Decimal): 0.09438178917535944290051

Minimum 100 bits.

CrypTool/Christian Schiestl Runs Test:

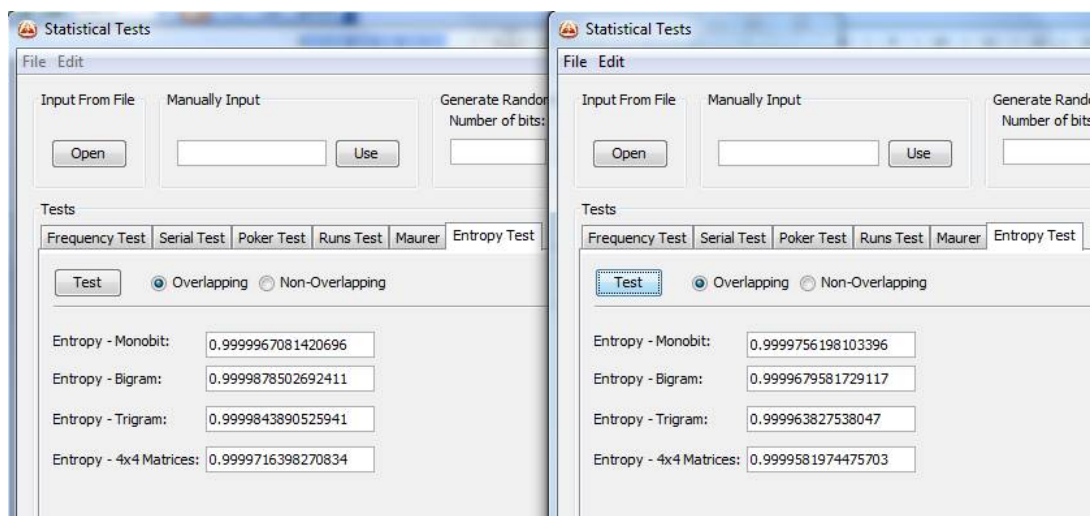
Length of Blocks/Gaps, k= 3
Expected number of 1bit Blocks/Gaps: 16384.25
Expected number of 2bit Blocks/Gaps: 8192.0625
Expected number of 3bit Blocks/Gaps: 4096.0
X4: 19.127430492466104
P-Value: 3.35836490742362E-4
P-Value (Decimal): 0.00033583649074236200443

Minimum 128 bits.

Status: Using Array from File for Tests. Read 131072 bits.

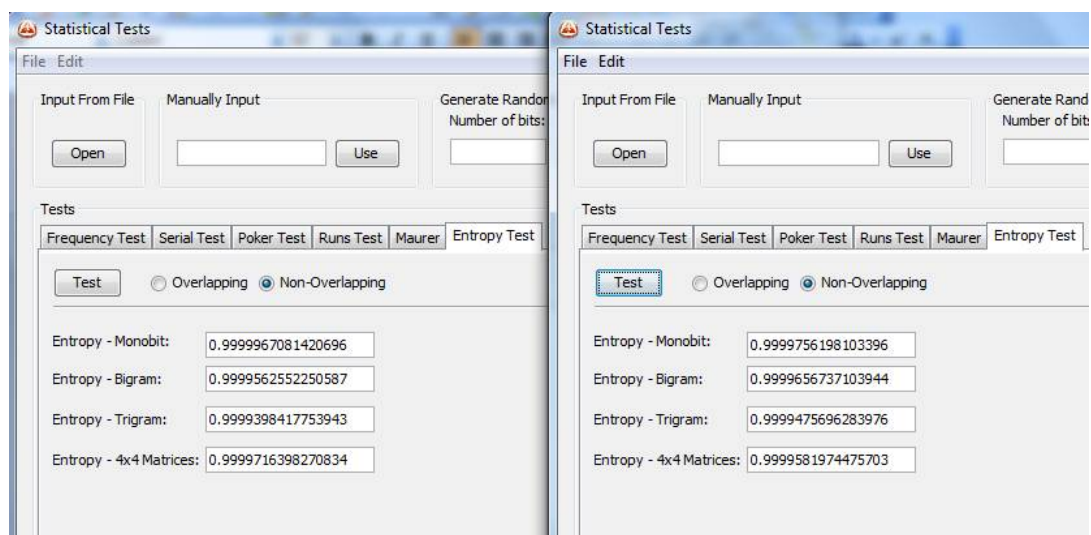
Slika 4.4-8 Rezultati testa uzastopnih ponavljanja istih vrednosti – uzorak random.org

4.4.5 Entropijski test (sa i bez preklapanja)



Slika 4.4-9 Rezultati entropijskog testa sa preklapanjem

(levo TRNG, desno random.org)



Slika 4.4-10 Rezultati entropijskog testa bez preklapanjem

(levo TRNG, desno random.org)

Na osnovu prikazanih rezultata statističkih testova, može se zaključiti da vrednosti koje je naš izvor slučajnosti generisao u potpunosti zadovoljavaju sve kriterijume koji su predefinisani i odnose se na proveru kvaliteta slučajno generisanih vrednosti. Poređenjem sa rezultatima dobijenim testiranjem vrednosti dobijenih sa sajta „random.org“, možemo zaključiti da je naš generator postigao značajno bolje rezultate. Obzirom da postoji ograničenje u veličini uzorka dobijenog sa sajta „random.org“ sa kojim smo vršili poređenje, 110MB podataka dobijenih od našeg generatora, testirano je pomoću „ENT“ i „Diehard“ testova, a rezultati testiranja se mogu videti na Internet adresi <http://www.cacert.at/cgi-bin/rngresults> (pretragu dobijene tabele izvršiti prema kriterijumu „Igor-XR232USB“).

4.5 *Oblast primene predloženog rešenja*

Prikazani generator slučajnosti može imati višestruku primenu. Osnovna namena ovog sklopa je obezbeđivanje statistički proverenih sekvenci slučajnih vrednosti za upotrebu u kriptologiji. Generisane vrednosti se mogu koristiti kao ključevi ili inicijalni vektori u algoritmima za šifrovanje, kao početne vrednosti za pseudo-slučajne izvore itd. Najznačajnija karakteristika ovog uređaja je integrisana zaštita od spoljnih uticaja na u vidu elektromagnetnog zračenja kao i zaštita od nepravilnog rada usled otkaza dela sklopa za generisanje šuma. U slučaju da izvor iz nekog razloga počne da generiše samo jednu od dve moguće binarne vrednosti, u relativno kratkom vremenskom periodu ($t > 20ms$), kontrolna procedura implementirana u kodu mikroprocesora će aktivirati zaštitu tako što će izvor prestati da generiše nove vrednosti, serijska komunikacija će biti zaustavljena i nakon toga će se uređaj resetovati. Prilikom procedure reseta, pored inicijalizacije registrara i ostalih potrebnih parametara, vrši se i reinicijalizacija SRAM memorije mikroprocesora tako sa ciljem uklanjanja ranije generisanih vrednosti.

5 Zaključak

5.1 *Sumarni ciljevi rada i ostvareni rezultati*

Nakon opisa nekoliko komercijalno dostupnih uređaja koji generišu slučajne vrednosti, možemo uočiti da se većina proizvođača odlučuje da, za izvor slučajnosti i način naknadne obrade generisanih vrednosti, koristi vrlo slične metode. Iako brzine generisanja kod prikazanih uređaja nisu velike kada ih poredimo sa nekim mnogo skupljim rešenjima u ovoj oblasti primene, upravo je balans između cene i brzine generisanja ove grupe uređaja ono što ih čini interesantnim. Kako bismo u praksi potvrdili neke od prikazanih karakteristika, pristupili smo izgradnji sopstvenog uređaja po uzoru na rešenje koje predlaže J.Thomas [1] što smo i dokumentovali u vidu električne šeme i izgleda štampanog kola uređaja. Koristeći namensku aplikaciju za komunikaciju sa uređajem generisali smo prve datoteke sa slučajnim vrednostima koje smo podvrgli rigoroznoj teorijsko-informacionoj analizi upotrebom odgovarajuće aplikacije koja implementira bateriju statističkih testova. Dobijeni rezultati su više nego zadovoljavajući kada ih uporedimo sa rezultatima istih testova kojima smo podvrgli referentne uzorke dobijene sa sajta "random.org". Sve prethodno navedeno apsolutno potvrđuje činjenicu da se ovaj tip uređaja, uz minimalna novčana ulaganja i uz minimalna poboljšanja koja su opisana u sledećem poglavlju, može koristiti u kriptološke svrhe, tj. može imati praktičnu primenu u zaštiti informacionih sistema.

5.2 *Predlog za budući rad*

Imajući u vidu činjenicu da je, za potrebe ovog rada, generator slučajnosti realizovan u relativno kratkom vremenskom roku upotrebom trenutno dostupnih elektronskih komponenti, nije bilo moguće realizovati proizvodnju adekvatnog metalnog oklopa koji bi u mnogome smanjio uticaj spoljnog elektro-magnetnog i visoko-frekventog radio zračenja na generator slučajnosti. Takođe, pored spoljašnjeg oklopa, treba osmisliti i implementirati elektronske zaštite od elektro-magnetnih smetnji.

Razmotriti mogućnosti upotrebe nekog naprednijeg modela mikrokontrolera u cilju povećanja brzine generisanja slučajnosti, povećanja brzine komunikacije, povećanja kapaciteta interne memorije čime bi se obezbedio veći skladišteni prostor kako za kod mikroprocesora tako i za veću količinu slučajnih vrednosti koje su spremne za isporuku.

Obzirom da je postojeći programski kod koji se izvršava u mikroprocesoru uređaja napisan u Assembler jeziku, trebalo bi razmotriti upotrebu nekog od viših programskih jezika u cilju bolje preglednosti i lakših izmena koda. U slučaju korišćenja jezika višeg nivoa sigurno će se produžiti vreme izvršavanja nekih operacija, ali ukoliko se opredelimo za moderniji mikroprocesor ovo ne bi trebalo da predstavlja značajno ograničenje, a za programske jezike višeg nivoa postoji i Integrirano razvojno okruženje (engl. IDE – Integrated Development Environment), konkretno Atmel Studio. Upotreba IDE-a omogućava lakše testiranje i kontrolu izvršavanja koda i to u grafičkom okruženju.

6 Literatura

- [1] Julien Thomas, XR232USB, [Internet] Dostupno na: http://www.itxp.org/tech/xr232usb_en.htm [pristupljeno 18.04.2015.]
- [2] Julien Thomas, XR232NT.exe, ...
- [3] Univerzitet Singidunum, Statistical Tests, ...
- [4] Araneus Information Systems Oy, Alea II, [Internet] Dostupno na: <http://www.araneus.fi/products/alea2/en/> [pristupljeno 09.06.2015.]
- [5] Tectrolabs LLC, TL200, [Internet] Dostupno na: <http://tectrolabs.com/tl200/> [pristupljeno 08.06.2015.]
- [6] Ubld.it, TrueRNG, [Internet] Dostupno na: <http://ubld.it/products/truerng-hardware-random-number-generator/> [pristupljeno 08.06.2015.]
- [7] Moonbase Otago, OneRNG, [Internet] Dostupno na: <http://onerng.info/> [pristupljeno 08.06.2015.]
- [8] Milosavljević M., Adamović S., (2013). Osnove teorije informacija i kodovanja. Beograd: Univerzitet Singidunum
- [9] Hartley R.V.L., (1928). Transmission of Information. Lucent Technologies
- [10] C.E. Shannon, (1948). A Mathematical Theory of Information
- [11] J. B. Johnson, (1928). Thermal Agitation of Electricity in Conductors, Physical Review vol 32, p97
- [12] H. Nyquist, (1928). Thermal Agitation of Electric Charge in Conductors, Physical Review vol 32, p110
- [13] NIST, (2010). Special Publication 800-22 Revision 1a
- [14] U.Maurer, (1992). A Universal Statistical Test for Random Bit Generators, Journal of Cryptology vol 5.
- [15] Randomness and Integrity Services Ltd, [Internet] Dostupno na: <https://www.random.org/bytes/> [pristupljeno 25.06.2015.]

PRILOG 1 Listing izvornog programskog koda mikrokontrolera

```

*****
;
; X R 2 3 2 U S B
; AVR Firmware for the XR232-USB-Stick Appliance HW Version 1.1
; *****
;
; Program Version:
; FULL SRAM FOR RANDOM POOLING, IMPROVED BITSAMPLING METHOD
; *****
;
;
; -----
; LICENSING INFORMATION
; -----
;
; Written in 2010-2014 by Julien Thomas (Germany) released under GPL
;
; This program is free software; you can redistribute it and/or
; modify it under the terms of the GNU General Public License
; as published by the Free Software Foundation; either version 3
; of the License, or (at your option) any later version.
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
; GNU General Public License for more details.
; You should have received a copy of the GNU General Public License
; along with this program; if not, write to the Free Software
; Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
; For more information on the GPL, please go to:
; http://www.gnu.org/copyleft/gpl.html
;
;
; -----
; PROJECT DESCRIPTION
; -----
;
; - See accompanying documentation. -
;
; -----
; PORTS I/O
; -----
;
; Microcontroller: AVR ATtiny85
;
; PB0 <- (ISP: MOSI) Sense RTS handshake line
; PB1 -> (ISP: MISO) TD = Transmit Databit output (RXD on FT232)
; PB2 -> (ISP: SCK) output activity-LED and CTS-return
; PB3 <- RD = Received Databit input (TXD coming from FT232)
; PB4 <- Digital Noise input coming from random source
; PB5 <- /RESET (triggered by falling edge of DTR)
;
; USB-Interface: FT232RL (inverted RS232-TTL-logic levels)
;
; Entropy Source: LM393 fast comparators in a chaotic oscillator setup
; with buffer to feed microcontroller input with 'digital noise'
;
; -----
; FUSEBITS
; -----
;
;

```

```

; Standard Fuses
; -----
; Ext : FF
; High : DF
; Low : E1
; Uses internal PLL for 16 MHz system clock.
; These fuses are safe under normal conditions and stable Vcc.
;
;
; Supporting Bootloader
; -----
; Ext : FE
; High : DC
; Low : E1
; Uses internal PLL for 16 MHz system clock.
; SELFPGEN enabled for Bootloader and BODLEVEL to mitigate risk of
; flash corruption under uncertain Reset/Vcc conditions.
;
;
; -----
; ASSEMBLER DIRECTIVES
; -----
.nolist
.include "tn85def.inc"
;
; VARIABLES AND CONSTANTS
; -----
.def ccnt1 = r10 ; cycle counter low
.def ccnt2 = r11 ; cycle counter high (measure baudrate)
.def txbyte = r12 ; random data byte for delivery
.def tcntv = r3 ; fast digital noise pulsewidth counter
.def rnbyte = r4 ; random new byte
.def robyte = r5 ; random old byte
.def temp = r16 ; global temporary register
.def dnnew = r17 ; new byte buffer DN-probe from PINB4
.def dnold = r18 ; old byte buffer DN-probe from PINB4 ;
.def rnds = r19 ; # of rounds for a new random byte
.def rpsl = r22 ; lowbyte random pool start
.def rpsh = r23 ; highbyte random pool start
.def rpel = r24 ; lowbyte random pool end
.def rpeh = r25 ; highbyte random pool end
;
; MACROS
; -----
.macro skipRTS0
    sbic PINB, 0
.endmacro
;
.macro skipRTS1
    sbis PINB, 0
.endmacro
;
.macro TXD0
    cbi PORTB, 1
.endmacro
;
.macro TXD1
    sbi PORTB, 1
.endmacro
;
.macro CTS0
    cbi PORTB, 2

```

```

.endmacro
;-----
.macro CTS1
    sbi PORTB, 2
.endmacro
;-----
.macro skipRXD0
    sbic PINB, 3
.endmacro
;-----
.macro skipRXD1
    sbis PINB, 3
.endmacro
;-----
.macro skipRDN0
    sbic PINB, 4
.endmacro
;-----
.macro skipRDN1
    sbis PINB, 4
.endmacro
;-----
; Fill Random Pool with noisy bits, sampling, balancing
;-----
.macro MaintainRandomPool
xfrp:
    inc tcntv                ; fast counter provides random bits
    in dnnew, PINB           ; load current state of PINB
    andi dnnew, 0b00010000   ; mask port bit 4
    cp dnnew, dnold          ; compare to previous state
    breq xfrpx               ; no changes -> leave
    mov dnold, dnnew         ; remember portbit has changed
    lsr tcntv                ; half tcntv and shift out lsb to carry
    rol rnbyte                ; shift that carry bit into rnbyte
    dec rnds                  ; round count down for a new random byte
    brne xfrpx
    ldi rnds, 8               ; re-set round counter to 8
    ld robyte, x              ; load older random byte from current X
    eor rnbyte, robyte        ; xor old+new random byte
    st x+, rnbyte             ; store new byte in X and increment X
    cp xl, rpel               ; check for last address in RP reached
    cpc xh, rpeh
    brcs xfrpx                ; if not overrun, leave
    movw xl, rpsl             ; else point back to RP start address
xfrpx:
.endmacro
;-----
; Collect randomness whilst received data bit = 0
;-----
.macro WhileRD0
    WhRD0:
        MaintainRandomPool
        skipRXD1
        rjmp WhRD0
.endmacro
;-----
; Collect randomness whilst received data bit = 0 (and measure startbit)
;-----
.macro WhileRD0Start

```

```

        clr ccnt1                ; clear startbit-length-counters
        clr ccnt2
WhRD0St:
        skipRXD0
        rjmp WhRD0Stx
        TXD0                    ; instantly reply with a startbit
        inc ccnt1                ; measure duration of host's startbit
        brne WhRD0St
        inc ccnt2
        brne WhRD0St
        rjmp RESET              ; overflow/timeout = RESET
WhRD0Stx:
.endmacro
;-----
; Collect randomness whilst received data bit = 1 (regular databit)
;-----
.macro WhileRD1
WhRD1:
        MaintainRandomPool
        skipRTS0                ; if RTS is deactivated
        CTS1                    ; immediately deactivate CTS too
        skipRXD0
        rjmp WhRD1

.endmacro
;-----
; Collect randomness whilst received data bit = 1 (stopbit)
;-----
.macro WhileRD1Stopp
        sbrcl tcntv, 7           ; tcntv >= 128 --- error!
        rjmp WRD1Serror
WRD1WaitStopbit:
WRD1WS:
        skipRXD1                ; continue while stopbit there is
        rjmp WRD1WSx            ; else return (startbit detected!)
        MaintainRandomPool      ; continue random pooling
        skipRTS0                ; if RTS became high (inactive)
        rjmp RESET              ; reset the whole stuff and restart
        rjmp WRD1WS             ; else loop on
        rjmp WRD1WSx
WRD1Serror:                     ; dn-too-long unchanged -- error
        CTS1                    ; deactivate CTS (and switch off LED)
WRD1Sewait:
        ser temp                ; set temp = 255 (# of randomitions)
WRD1Sew0:
        skipRDN1                ; if random noise is 1 then go WRD1Sew1
        rjmp WRD1Sew0           ; else loop on
WRD1Sew1:
        skipRDN0                ; if random noise is 0 then go WRD1Sew0
        rjmp WRD1Sew1           ; else loop on
        dec temp                ; count down rounds
        brne WRD1Sew0           ; loop on
        rjmp RESET              ; and RESET
WRD1WSx:                        ; normal leave
.endmacro
;-----
; Send txbyte (register being left untouched)
;-----
; While RS232 character is being sent out,
; random noise sampling and processing continues.

```

```

,
;
;-----
; MAIN PROGRAM START
;-----
.cseg
.org $0000
;-----
; JUMPTABLE
;-----
    rjmp RESET                ; one and only irq allowable
    reti                     ; (jumptable may be obsolete)
    reti
    reti
    reti
    reti
    reti
    reti

```

```

    reti
    reti
    reti
    reti
    reti
    reti
;-----
; FIRMWARE IDENTIFIER
;-----
IDENT:
; 0123456789012345678901234567890123456789012345678901234567890
.db "XR232USB build 20140717"
.db 0, 0
;-----
; RESET / RE-INIT
;-----
RESET:
    cli                    ; interrupts - no, thanks
    CTS1                  ; inhibit RS232 activity for now
; Set stackpointer to end of SRAM (actually we won't use the stack)
    ldi temp, low (RAMEND) ; ramend low stackpointer
    out SPL, temp
    ldi temp, high(RAMEND) ; ramend high stackpointer
    out SPH, temp
; Init portbits
    ldi temp, 0b00000110   ; set data directions on PORTB
    out DDRB, temp
    ldi temp, 0b00111111   ; set pullups and initial states
    out PORTB, temp
; Init some working registers
    clr ccnt1              ; reset cycle counter low
    clr ccnt2              ; reset cycle counter high
    clr dnnew              ; portbyte new
    clr dnold              ; portbyte old
    ldi rnds, 8            ; round counter
; Init pointers and register constants for Random Pool
    ldi xh, high(SRAM_START) ; pointer for sram writes (RP)
    ldi xl, low (SRAM_START)
    ldi zh, high(SRAM_START) ; pointer for sram reads (RS232)
    ldi zl, low (SRAM_START)
    ldi rpsl, low (SRAM_START) ; fill registers with constants
    ldi rpsh, high(SRAM_START) ; for fastest calculation
    ldi rpel, low (RAMEND)      ; define full SRAM for RP
    ldi rpeh, high(RAMEND)     ; (512 Bytes)
; Refresh and recycle Random Pool (RP) data after Reset / in Idle state
InitRPLoop:
    MaintainRandomPool
    dec ccnt1
    brne InitRPLoop
    dec ccnt2
    brne InitRPLoop        ; do at least 65535 rounds
    skipRTS0              ; if RTS is still not active
    rjmp InitRPLoop        ; simply loop on
;-----
; MAINLOOP
;-----
Mainloop:
    CTS0                  ; CTS + LED on
Mloop:
Mlo1:

```

```

    TXD1                                ; stopbit/idle
    WhileRD1Stopp                       ; maintaining RP
    TXD0                                ; startbit
    WhileRD0Start                       ; measured (ccnt)
    ldi temp, 4                         ; test baudrate
    cp ccnt2, temp                      ; if higher baudrate been detected
    brcs Mlo2                           ; go on with random service
    rjmp SendIdentifier                 ; else send identifier
Mlo2:
    ld txbyte, z+                       ; load value from RP
    Sendbyte                           ; transmit random byte
    cp zl, rpl                          ; check for end of RP
    cpc zh, rpeh
    brcs Mlo3
    movw zl, rpsl                       ; pointer Z back to start of RP
Mlo3:
    rjmp Mloop
;-----
; Send identifier
;-----
SendIdentifier:
    ldi zl, low (IDENT*2)               ; load identifier address start
    ldi zh, high(IDENT*2)               ; low and highbyte
    rjmp Sid1
Sid0:
    TXD0                                ; send start bit
    WhileRD0Start                       ; check some conditions
Sid1:
    lpm txbyte, z+                      ; load data byte from flash
    Sendbyte
    TXD1
    WhileRD1Stopp
    cpi zl, low (IDENT*2)+32             ; cycle same string of 32 chars
    brne Sid2
    ldi zl, low (IDENT*2)               ; load identifier address start
    ldi zh, high(IDENT*2)               ; low and highbyte
Sid2:
    rjmp Sid0
;-----
.exit
;*****
;
; END OF XR232USB-FW
;*****
;

```

PRILOG 2 Izvorni kod XR232NT aplikacije

```
*****
' XR232NT - NEW COMMANDLINE TOOL FOR XR232 AND XR232USB
*****
' Program Version: See identifier below
' Compile with: Freebasic v0.90 +
*****
'
'-----
' LICENSING INFORMATION
'-----
' Written in 2010-2014 by Julien Thomas <joytec@gmx.de> under GPL
'
' This program is free software. You can redistribute it and/or
' modify it under the terms of the GNU General Public License
' as published by the Free Software Foundation, either version 3
' of the License, or (at your option) any later version.
' This program is distributed in the hope that it will be useful,
' but WITHOUT ANY WARRANTY' without even the implied warranty of
' MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
' GNU General Public License for more details.
' You should have received a copy of the GNU General Public License
' along with this program. If not, write to the Free Software
' Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
' For more information on the GPL, please go to:
' http://www.gnu.org/copyleft/gpl.html
'-----
' GLOBAL DEFINITIONS
'-----
#lang "fb"
#include once "xr232lib.bas"      ' standalone
#include once "xr232lib.bi"      ' dll-dependent
Const Versionnumber = "20140720"
Const CR = chr$(13)
Dim shared Comport as String
Dim shared RXDA (255) as UByte
Dim shared Astr as String
Dim shared Bstr as String
Dim shared Filename as String
Dim shared as UInteger i,j,a,b,c,t,z
Declare Function Bitmap()      as Byte
Declare Function Hexdump()    as Byte
Declare Function DigitalSumValue() as Byte
Declare Function ByteFrequency() as Byte
Declare Function MakeRAWFile(byref Ftarget as Uinteger) as Byte
Declare Function MakeXORFile(byref Ftarget as Uinteger) as Byte
Declare Function MakeRNDFile(byref Ftarget as Uinteger) as Byte
Declare Sub   XRAutoList ()
'-----
On Error GoTo Xerror
'-----
Function Bitmap() as Byte
dim x as Uinteger
dim y as Uinteger
dim i as Ushort
if XR_Open(Comport) <> 0 then return (1)
WindowTitle "XR232 @ "+ Comport + " - BITMAP" 'Astr
```

```

Screen 16                                'same as ScreenRes 512, 384
x = 0 : y = 0
while inkey$ = ""
    if XR_GetBlock(Comport, RXDA()) <> 0 then return (1)
    for i = 0 To 255
        c = RXDA(i)
        pset (x, y), abs (bit (c, 7) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 6) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 5) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 4) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 3) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 2) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 1) = 0) * 15 : x = x + 1
        pset (x, y), abs (bit (c, 0) = 0) * 15 : x = x + 1
    if x > 511 then x = 0 : y = y + 1
    next i
if y > 383 then y = 0
Wend
return (0)
End Function
'-----

Function Hexdump() as Byte
if XR_Open(Comport) <> 0 then return (1)
do
    if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
    For i = 0 To 255
        Print whex(RXDA(i),2);
    Next i
loop until inkey$ <> ""
return (0)
End Function
'-----

Function DigitalSumValue() as Byte
dim i as UShort
dim a as Ubyte
dim bcnt (0 To 255) as UByte 'precalculated table of bitcounts per byte
for i = 0 to 255
    a = 0
    if (i and 128) then a = a + 1
    if (i and 64) then a = a + 1
    if (i and 32) then a = a + 1
    if (i and 16) then a = a + 1
    if (i and 8) then a = a + 1
    if (i and 4) then a = a + 1
    if (i and 2) then a = a + 1
    if (i and 1) then a = a + 1
    bcnt(i) = a
next i
dim countallbits as ULong = 0
dim countallones as ULong = 0
dim roundonebits as ULong = 0
dim numberblocks as UInteger = 0
print
if XR_Open(Comport) <> 0 then return (1)
While numberblocks < 51200 and inkey$=""
    for j = 0 to 3
        if XR_GetBlock(Comport, RXDA()) <> 0 then return (1)
        numberblocks = numberblocks + 1
        roundonebits = 0
        for i = 0 to 255

```

```

        roundonebits = roundonebits + bcnt( RXDA(i) )
    next i
    countallones = countallones + roundonebits
next j
countallbits = 8 * 256 * numberblocks
print CR; "DSV-BLOCK :"; : print using "##.#####"; roundonebits / (8*256);
print ,
print "KB TOTAL : "; numberblocks\4;
print " ";
If numberblocks\512 = numberblocks/512 then
    print
    Print "DSV-TOTAL :"; : print using "##.#####"; countallones / countallbits
    print "ONES (1) : "; countallones
    print "ZEROES (0) : "; countallbits - countallones
    print "TOTAL BITS : "; countallbits
    print "RATIO      : "; : print using "##.#####"; 100 * (countallones / countallbits)
    print "BIAS       : "; : print using "##.#####"; 100 * (countallones / countallbits) - 50
    print
end if
Wend
return (0)
end Function
'-----
Function ByteFrequency() as Byte
dim bfrq (255) as Ubyte
dim k as Ushort
dim x as Uinteger
dim y as Uinteger
dim xp as Uinteger = 60
dim yp as Uinteger = 60
if XR_Open(Comport) <> 0 then return(1)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
Astr = "XR232 @ " + Comport + " - BYTE FREQUENCY"
WindowTitle Astr
Screen 11
Locate 2,8
Print "Frequency of byte values (cumulative)"
Line (0,yp)-(xp+512+xp, yp),1
Line (0,yp+128)-(xp+512+xp, yp+128),1
Line (0,yp+256)-(xp+512+xp,yp+256),1
Locate 5,2 : Print "+128"
Locate 13,1 : Print " 0 "
Locate 21,2 : Print "-127"
Locate 22,8 : Print "0   31   63   95   127   159   191   223   255"
For i = 0 To 07: Line (xp+i*64,yp+256)-(xp+i*64,yp+272), 1 : Next i
For i = 0 To 15: Line (xp+i*32,yp+256)-(xp+i*32,yp+268), 1 : Next i
For i = 0 To 31: Line (xp+i*16,yp+256)-(xp+i*16,yp+264), 1 : Next i
Line (xp+511,yp+256)-(xp+511,yp+272), 1
Locate 27,8
Print "Scanning up to 10 MBit - Press any key to stop."
Locate 25,9
Print "0 MBit tested."; CR;
k = 0
While k < 5120 And InKey$=""
    if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
    'Block2Array
    k = k + 1
    b = 0
    For i = 0 To 255
        a = RXDA(i)

```

```

        bfrq(a) = bfrq(a) + 1
    Next i

    For i = 0 To 255
        y = bfrq(i) - k
        y = 255 - (y + 127) And 255
        y = y + yp
        x = i*2 + xp
        Line (x,y)-(x,yp+255),1
        Line (x,yp)-(x, y),0
    Next i
    If k\512 = k/512 then
        Print "      "; k\512;" MBit tested."; CR;
    EndIf
Wend
Sleep
return (0)
end Function
'-----
Function ByteFrequencyRND() as Byte
dim bfrq(255) as Ubyte
dim k as Ushort
dim x as Uinteger
dim y as Uinteger
dim xp as Uinteger = 60
dim yp as Uinteger = 60
if XR_Open(Comport) <> 0 then return(1)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
Astr = "BYTE FREQUENCY OF RND()"
WindowTitle Astr
Screen 11
Locate 2,8
Print "Frequency of byte values in pseudorandom function (cumulative)"
Line (0,yp)-(xp+512+xp, yp),1
Line (0,yp+128)-(xp+512+xp, yp+128),1
Line (0,yp+256)-(xp+512+xp,yp+256),1
Locate 5,2 : Print "+128"
Locate 13,1 : Print " 0 "
Locate 21,2 : Print "-127"
Locate 22,8 : Print "0   32   64   96   128   160   192   224   255"
For i = 0 To 07: Line (xp+i*64,yp+256)-(xp+i*64,yp+272), 1 : Next i
For i = 0 To 15: Line (xp+i*32,yp+256)-(xp+i*32,yp+268), 1 : Next i
For i = 0 To 31: Line (xp+i*16,yp+256)-(xp+i*16,yp+264), 1 : Next i
Line (xp+511,yp+256)-(xp+511,yp+272), 1
Locate 27,8
Print "Scanning up to 10 MBit - Press any key to stop."
Locate 25,9
Print "0 MBit tested."; CR;
k = 0
Randomize Timer,4
While k < 5120 And InKey$=""
    For i = 0 To 255
        RXDA(i) = int(rnd*256)
    Next
    k = k + 1
    b = 0
    For i = 0 To 255
        a = RXDA(i)
        bfrq(a) = bfrq(a) + 1
    Next

```

```

        For i = 0 To 255
            y = bfrq(i) - k
            y = 255 - (y + 127) and 255
            y = y + yp
            x = i*2 + xp
            Line (x,y)-(x,yp+255),1
            Line (x,yp)-(x, y),0
        Next i
    If k\512 = k/512 then
        Print "      "; k\512;" MBit tested."; CR;
    EndIf
Wend
sleep
return (0)
end function
'-----
Function MakeRAWFile (byref Ftarget as Uinteger) as Byte
dim i as Ushort
dim ff as Ubyte : ff = freefile
dim Fsize as Uinteger = 0
if XR_Open(Comport) <> 0 then return(1)
if open (Filename for output as #ff) <> 0 then return(1)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1) ' get 1st block to 'run warm'
print
' send full blocks of 256 bytes each
do until (Fsize \ 256) = (Ftarget \ 256)
    '~ ' get block of randomness from TRNG (256 bytes)
    '~ if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
    '~ ' send random data block (directly from RXD-string)
    '~ print #ff, RXD;
    if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
    for i = 0 to 255 : print #ff, chr$(RXDA(i)); : next i
    Fsize = Fsize + 256
    ' show progress every now and then
    if Fsize \ 1024 = Fsize / 1024 then
        Print CR; "Write XR232 Random File (RAW): ";
        Print Fsize ; " of "; Ftarget; " Bytes ";
        if inkey$ <> "" then return (1)
    endif
endif
loop
' send remainder of less than 256 bytes
if (Ftarget-Fsize) = 0 then return(0)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
' send remainder < 256
for i = 1 to (Ftarget-Fsize) : print #ff, chr$(RXDA(i-1)); : next i
Fsize = Ftarget
Print CR; "Write XR232 Random File (RAW): ";
Print Fsize ; " of "; Ftarget; " Bytes ";
close ff
return(0)
end function
'-----
Function MakeXORFile (byref Ftarget as UInteger) as Byte
dim ff as Ubyte : ff = freefile
dim RXDX (255) as Ubyte 'internal array be XOR'ed
dim Fsize as Uinteger = 0
if XR_Open(Comport) <> 0 then return(1)
if open (Filename for output as #ff) <> 0 then return(1)

```

```

if XR_GetBlock(Comport, RXDA()) <> 0 then return(1) ' get 1st block
for i = 0 to 255 : RXDX(i) = RXDA(i) : next i ' copy to RXDX(x)
print
' send full blocks of 256 bytes each
do until (Fsize \ 256) = (Ftarget \ 256)
    'get another block of randomness from TRNG (256 bytes)
    if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
    ' XOR block with current data
    for i = 0 to 255 : RXDX(i) = RXDX(i) xor RXDA(i) : next i
    for i = 0 To 255 : print #ff, chr$(RXDX(i)); : next i
    Fsize = Fsize + 256
    ' show progress
    if Fsize \ 1024 = Fsize / 1024 then
        Print CR; "Write XR232 Random File (XOR): ";
        Print Fsize ; " of "; Ftarget; " Bytes ";
    if inkey$ <> "" then close ff : return(1)
    endif
loop
' send remainder of less than 256 bytes
if (Ftarget - Fsize) = 0 then return(0)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
for i = 0 to 255 : RXDX (i) = RXDX(i) xor RXDA(i) : next i
for i = 1 To (Ftarget-Fsize) : print #ff, chr$(RXDX(i-1)); : next i
Fsize = Ftarget
Print CR; "Write XR232 Random File (XOR): ";
Print Fsize ; " of "; Ftarget; " Bytes ";
close #ff
return(0)
end Function
'-----
Function MakeRNDFile (byref Ftarget as Uinteger) as Byte
dim ff as Ubyte : ff = freefile
if XR_Open(Comport) <> 0 then return(1)
if open (Filename for output as #ff) <> 0 then return(1)
if XR_GetBlock(Comport, RXDA()) <> 0 then return(1) ' discard 1st block to 'run warm'
dim Fsize as Uinteger = 0
print
j = 1
' send full blocks of 256 bytes each
do until (Fsize \ 256) = (Ftarget \ 256)
    ' RND function is mersenne-twister seeded with 32 bits from TRNG
    randomize (RXDA (j+0) + 2^8 * RXDA(j+1) + 2^16 * RXDA(j+2) + 2^24 * RXDA(j+3)), 3
    ' send 256 bytes from PRNG freshly seeded with real entropy
    for i = 0 to 255 : Print #ff, chr$(int(rnd*256)); : next i
    Fsize = Fsize + 256
    j = j + 4
    ' reload real randomness from XR232 every 16K (64*256) Bytes
    if j > 256 then
        if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
        j = 1
    endif
    ' show progress indicator all 1024 bytes
    if Fsize \ 1024 = Fsize / 1024 then
        Print CR; "Write XR232 Random File (RND): ";
        Print Fsize ; " of "; Ftarget; " Bytes ";
    if inkey$ <> "" then return (1)
    endif
loop
' send remainder of less than 256 bytes
if (Ftarget - Fsize) = 0 then return (0)

```

```

if XR_GetBlock(Comport, RXDA()) <> 0 then return(1)
randomize (RXDA (1) + 2^8 * RXDA(2) + 2^16 * RXDA(3) + 2^24 * RXDA(4)), 3
for i = 1 to (Ftarget-Fsize)
    print #ff, chr$(int(rnd*256));
next i
Fsize = Ftarget
Print CR; "Write XR232 Random File (RND): ";
Print Fsize ; " of "; Ftarget; " Bytes ";
close ff
return(0)
End Function
'-----
Function CheckXR232Identifier (byref Comport as string) as Byte
dim astr as string
astr = XR_GetIdent(Comport)
if astr = "" then return(1)
astr = Comport + " : " + astr
print astr
return(0)
End Function
Sub AutoList ()
dim Comport as String
dim c as UShort
#ifdef __FB_WIN32__ ' specific code for Win32
c = 1
do while inkey$ = "" and c < 256
    Comport = "COM" + str$(c)
    if XR_Probe (Comport) = 0 then
        print "XR232 detected at "; Comport
    endif
c = c + 1
loop
#endif
#ifdef __FB_UNIX__ 'specific code for Linux
c = 0
do while inkey$ = "" and c < 256
    Comport = "/dev/ttyS" + str$(c)
    if XR_Probe (Comport) = 0 then print "XR232 detected at "; Comport
c = c + 1
loop
c = 0
do while inkey$ = "" and c < 256
    Comport = "/dev/ttyUSB" + str$(c)
    if XR_Probe (Comport) = 0 then print "XR232USB detected at "; Comport
c = c + 1
loop
#endif
end sub
'-----
' Parse commandline
'-----
CommandlineParse:
print
select case lcase$(right$(command$(1), 1))
    case "" : goto Helpscreen
    case "h" : goto Helpscreen
    case "l" : goto Licensenote
end select
Comport = command$(1)

```

```

if left$(lcase$(Comport), 4) = "auto" then
    Comport = XR_AutoDetect()
    if Comport = "" then
        goto Xerror
    else
        Comport = Comport + right$(command$(1),len(command$(1))-4)
    endif
endif
' upper-case COM-port name (windows)
if left$(Comport, 3) = "com" then Comport = ucase$(Comport)
' no further arguments, only COM port specified -> check for identifier
if command$(2) = "" then
    if CheckXR232Identifier(Comport) = 0 then
        goto Finished
    else
        goto Xerror
    endif
endif
' set default baudrate if not specified but mode switch
i = 0
Comport = trim (Comport, ":")
do : i = i + 1 : loop until (i = len(Comport)) or (mid$(Comport,i,1) = ":")
if i = (len(Comport)) then Comport = Comport + ":19200"
print Comport
if command$(3) = "" then      'Options with screen output
    select case lcase$(right$(command$(2), 1))
        case "" : CheckXR232Identifier(Comport)
        case "h" : if Hexdump() then goto Xerror
        case "d" : if DigitalSumValue() then goto Xerror
        case "b" : if Bitmap() then goto Xerror
        case "f" : if ByteFrequency() then goto Xerror
        case "r" : if ByteFrequencyRND() then goto Xerror
        case else: goto Xerror
    end select
    goto Finished
endif
Filename = command$(2)      ' Options with file output
b = val(command$(3))        ' Next argument should be # of bytes
if b = 0 then goto Xerror
if lcase$(right$(command$(3),1)) = "k" then b = b * 1024
if lcase$(right$(command$(3),1)) = "m" then b = b * 1024 * 1024
if lcase$(right$(command$(3),1)) = "g" then b = b * 1024 * 1024 * 1024
select case lcase$(right$(command$(4), 1))
    case "" : if MakeRAWFile(b) then goto Xerror
    case "x" : if MakeXORFile(b) then goto Xerror
    case "r" : if MakeRNDFile(b) then goto Xerror
    case else : goto Xerror
end select
goto Finished
'-----
Xerror:
print
print "ERROR!"
print
Beep
End (err)
'-----
Finished:
print
XR_Close()

```

```

End (0)
'-----
' HELPSCREEN
'-----
Helpscreen:
'-----
#ifdef __FB_WIN32__ ' specific code for Win32
'-----
Print
Print "-----"
Print "XR232 New Commandline Tool for Win32, version "; Versionnumber; " (jt) joytec@gmx.de"
Print "-----"
Print ""
Print "Testing:"
Print ""
Print "xr232nt [-h|-?]      Helpscreen + Autodetection"
Print "xr232nt -l           Show license note and lib info"
Print "xr232nt COMx          Check identifier"
Print "xr232nt COMx:Baud -H   Hex-Dump on screen"
Print "xr232nt COMx:Baud -D   Digital Sum Value"
Print "xr232nt COMx:Baud -B   Bitmap"
Print "xr232nt COMx:Baud -F   Frequency of bytes"
Print ""
Print ""
Print "Operation:"
Print ""
Print "xr232nt COMx:Baud Filename N[K|M] [X|R]"
print ""
Print "Write N Bytes of binary random data to the specified [Path]Filename."
Print "With suffix 'K' it's Kilobytes; suffix 'M' is Megabytes."
Print "Additional balancing options:"
Print "      -X      XOR successive blocks"
Print "      -R      Seeded RND (very fast)"
Print ""
Print ""
Print "Replace 'COMx' by 'AUTO'    Uses first autodetected XR232/XR232USB"
Print ""
print "Scanning for XR232/XR232USB - Press any key to abort."
print ""
AutoList()
goto Finished
#endif
'-----
#ifdef __FB_UNIX__ 'specific code for Unix/Linux
'-----
Print
print "-----"
print "XR232 New Commandline Tool for Linux, version ";Versionnumber; " (jt) joytec@gmx.de"
print "-----"
print ""
print "Testing:"
print ""
print "xr232nt [-h|-?]      Helpscreen + Autodetection"
print "xr232nt /dev/ttySx    Check identifier"
print "xr232nt /dev/ttySx:Baud -H   Hex-Dump on screen"
print "xr232nt /dev/ttySx:Baud -D   Digital Sum Value"
print "xr232nt /dev/ttySx:Baud -B   Bitmap"
print "xr232nt /dev/ttySx:Baud -F   Frequency of bytes"
print ""
print "NOTE: USB serial ports (XR232USB) are being mapped to /dev/ttyUSBx."

```

```

print ""
print "Operation:"
print ""
print "xr232nt COMx:Baud Filename N[K|M] [X|R]"
print ""
print "Write N Bytes of binary random data to the specified [Path]Filename"
print "With suffix 'K' it is interpreted as Kilobytes; suffix 'M' is Megabytes"
print "Additional balancing options:"
print "        -X      XOR successive blocks"
print "        -R      Seeded RND (very fast)"
Print ""
Print "Replace '/dev/...' by 'auto' Use first autodetected XR232/XR232USB"
Print ""
print "Scanning for XR232/XR232USB - Press any key to abort."
print
AutoList()
goto Finished
#endif
'-----

Licensenote:
print "XR232NT - Console Tool for XR232/XR232USB True Random Number Generators"
print "Copyright (C) Julien Thomas <joytec@gmx.de>"
print ""
print "This program is free software; you can redistribute it and/or modify it"
print "under the terms of the GNU General Public License as published"
print "by the Free Software Foundation; either version 3 of the License"
print "or (at your option) any later version."
print "This program is distributed in the hope that it will be useful,"
print "but WITHOUT ANY WARRANTY; without even the implied warranty"
print "of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE."
print "See the GNU General Public License for more details."
print "You should have received a copy of the GNU General Public License"
print "along with this program; if not, see <http://www.gnu.org/licenses/>."
print ""
print "Compile date: " & __DATE_ISO__ " with " & __FB_SIGNATURE__
print
print "Using xr232lib version: "; XR_LibVersion()
print
end
'-----

'Notes on serial port parameters:
'CS      CTS-timeout
'DS      DSR-timeout
'CD0     DCD not used
'OP      timeout for disconnected port
'TB      transceive buffer (number of bytes)
'RB      receive buffer (number of bytes)
'RS      disable RTS/CTS-handshake (NOT recommended with XR232USB!)
'FE      ignore some character errors
'ME      ignore all errors (danger of hangup)
'BIN     activate binary transmission mode
'":19200,N,8,1,CS1000,DS0,CD0,OP0,TB0,RB0,ME,BIN"
'":19200,N,8,1,CS1000,DS0,CD0,OP1000,TB0,RB0,BIN"
'":19200,N,8,1,CS1000,DS0,CD0,OP0,TB0,RB0,FE,BIN"
'":1200,N,8,1,CS1000,DS0,CD0,OP1000,TB0,RB0,ME,BIN"
'-----

'-- END OF PROGRAM SOURCE "XR232NT" COMMANDLINE TOOL -----
'-----

```