TECHNICKÁ UNIVERZITA v LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program:: B2612 - Elektrotechnika a informatika

Studijní obor: Elektrotechnické informační a řídicí systémy

HW generátor náhodných čísel

HW random number generator

Bakalářská práce

Autor: **Michal Třešňák**

Vedoucí práce: Ing. Josef Grosman

V Liberci 16. 5. 2013

Originál zadání práce

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom po-vinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum 16. 5. 2013

_____

Podpis

Abstrakt

Tato práce popisuje problematiku generátorů náhodných čísel a jejich testování. Hlavním záměrem bylo navrhnout a zkonstruovat HW generátor, který by se dal srovnat s generátory běžně dostupnými a ověřit si jeho funkci pomocí série testů. Provedené řešení má dvě hlavní části. První částí je samotný přípravek. Pomocí mikroprocesoru, obsahující A/D převodník, komunikuje s PC přes sériové rozhraní. Druhou částí je uživatelská aplikace, která slouží pro testování a zobrazení výsledků.

Klíčová slova: HW generátor, náhodná čísla, TRNG, PRNG, testování náhodnosti, statistika, vzorkování dat, kryptografie, entropie

Abstract

This bachelor's thesis describes issues about random number generators and testing of them. The main purpose was design and construct of HW generator, that should be equal to commercially available device and check it with sequence of tests. This solution has two main parts. The first one is hardware itself, including microprocessor with A/D converter, comunnicates with PC via serial interface. Second part is a software application for testing and viewing results.

Keywords: HW generator, random number, TRNG, PRNG, testing of randomness, statistics, sampling data, cryptography, entropy

Obsah

Seznam obrázků

1 Úvod

Díky pokročilé době komunikačních technologií je čím dál tím větší problém chránit data před útoky skrze internetovou síť. Nejlepším způsobem ochrany je jejich zašifrování. Nejjednodušším zařízením je například stolní počítač, který pomocí algoritmů udělá ze zdrojových dat pro jiné zařízení nečitelnou informaci. Přesto se dá, při velké snaze, času a výpočetnímu výkonu, rozluštit. Je to způsobeno tím, že se celý proces vykazuje jakousi pravidelnost. Výjimkou jsou delší generované klíče, které svůj průběh zakládají na sekvenci dat, která jsou generována skutečnými generátory. Ty pracují s lehce elektronicky zpracovatelnou fyzikální veličinou. S jejich pomocí lze získat klíč, který bude pro nežádaného čtenáře nerozluštitelný i při použití velkého výpočetního výkonu.

Pro navržení takového generátoru je potřeba absolvovat několik kroků. V první řadě záleží na vybraném zdroji náhody, který se bude zpracovávat. Měl by být skutečně nepředvídatelný, bez zjevné matematické závislosti aktuálního průběhu na předchozích či budoucích stavech. V drtivé většině existujících generátorů se využívá různých druhů šumu. Dále je potřeba signál efektivně navzorkovat a v určitém formátu předat zařízení, které informaci zpracuje. V tuto chvíli už by měla mít data určitý formát, aby se dala interpretovat nebo upravit pomocí korekcí.

Nedílnou součástí při vývoji tohoto zařízení je ověření, zda jsou generovaná čísla opravdu náhodná. K těmto účelům existují skupiny testů, které do jisté míry dokážou rozlišit nenáhodné řady.

2 Druhy generátorů

2.1 Pseudonáhodné

Nazývané také jako deterministické. Jsou nejběžněji používaným typem generátoru, označovány zkratkou PRNG (pseudo random number generator), tedy pseudonáhodné. Jejich princip je založen čistě na matematice, tudíž musí z něčeho vycházet, aby generovaly různé sekvence hodnot. Proto je při začátku algoritmu použito semínko (seed), cože je sekvence náhodných dat, které se dosadí za první hodnotu. To je většinou získáno pomocí skutečného generátoru nebo souboru. Výhodou je velká rychlost a díky ověřeným algoritmům při úměrné délce semínka i velká spolehlivost. Objem dat ke generování je však velmi malý a po vyčerpání ze „zásoby“ se opět opakují. To by byl problém při poskytování zabezpečených stránek, protože, než se data opět naplní, generování se zastaví. V jistých případech, kde je stejná řada vyžadována, je to však velké plus.

Nejčastěji pracují na základě polynomů a rekurze. Jednoduchý vzorec Blum Blum Shub ($x_{n+1} = (x_n)^2 \ mod \ M$) například požívá rekurzivní formuli.

2.1.1 Lineární zpětnovazební registr

Známý pod zkratkou LFSR se též často používá pro získávání pseudonáhodných hodnot. Je to registr, jehož vstupní bit je lineární funkcí předchozího. Nejčastěji používanou funkcí je XOR. Jeho počet hodnot je konečný, lze však použít jinou funkci a získat požadovanou délku sekvence, která se zdá být náhodná. Pomocí Fibonacciho mutace zapojení lze dosáhnout i polynomické funkce.

2.1.2 Mersenne twister

Je pseudonáhodný generátor, který je založen na lineární závislosti matic v binárním poli. Poskytuje rychlejší a kvalitnější řady, než většina ostatních algoritmů. Jeho výstupem jsou dvaatřicetibitová čísla. Problémem je, že se jeho další sekvence dají předem odhadnout.

Je citlivý na počáteční hodnoty a jeho dobu zotavení při „přetečení“ z konečného na počáteční stav.

2.1.3 Fortuna

Patří mezi pro kryptografii bezpečné generátory. Jeho předností je možnost vygenerování nekonečné řady dat. To za předpokladu, že se jeho počáteční hodnoty změní po 1 MiB dat. Jako základ používá sekvenci ze skutečného generátoru. Disponuje odolností proti útoku pomocí změny seedu. Ten se stačí obnovit dříve, než se útočník stačí identifikovat, zda se mu to podařilo, či ne. Podmínkou je dostatek vstupních dat.

2.2 Skutečné

Takzvaný TRNG (true random number generator) je založen na získávání dat z nepředvídatelného zdroje – zdroje entropie. Entropie se dá definovat jako míra neuspořádanosti. Aktuální stav nesmí záležet na předchozích ani budoucích okolnostech. Ideálním je využití přírody, hlavně v oblasti fyziky, nebo i člověka.

2.3 Příklady komerčních generátorů

Výrobci těchto zařízení si velice bedlivě střeží know-how a tomu odpovídá jejich cena. Najdou se i generátory podomácku dělané, levnější, ale jejich doba dodání bývá příliš dlouhá a díky menší rychlosti je jich při větším použití potřeba více. Dodávány jsou se zárukou a s balíkem testů, pomocí nichž si můžete jejich kvalitu ověřit. V úvahu nejsou brány webové online generátory, díky nimž nezískáte krom dobré dostupnosti zhola nic.

2.3.1 ComScire R32MU

Obrázek 1 - ComScire R32MU

- Rychlost – až 32 Mb/s
- Kontinuální hardwarové testování
- Rozhraní USB 2.0
- Spotřeba 135mA max
- Pracuje v rozmezí 0 – 15°C
- Podle výrobce splňuje každý dostupný test
- Cena $ 1,495.00 (cca 30 tis. Kč)

2.3.2 SG100

Obrázek 2 - SG100

- Na trhu od roku 1997
- Odolný vůči rušení
- Rychlost 9,2 kB/s
- Bez slabin
- Prošlo náročnými testy
- Sériové rozhraní
- Cena 302 € (cca 8200,- Kč)
- K dispozici i novější s USB se stejným výkonem za srovnatelnou cenu

2.3.3 Entropy Key

Obrázek 3 - Entropy KEY

- Rozhraní USB
- Využívá hned dvou generátorů šumu
- Při generování spolupráce s PC a internetovým zabezpečením
- Testování používaných klíčů před odesláním na server

- Rychlost neuvedena
- Projde testy
- Cena £36.00 (cca 1200,-Kč)

2.3.4 Zdroje entropie

Spousta zdrojů je však pro účely generování nepoužitelná, jelikož se nedá elektricky a efektivně zpracovat. Mezi ně patří lidský faktor (hod kostkou, mincí nebo ruční zadávání). Přesto se najde možnost ho jistých případech využít. V dnešní době se používá pro vytváření bezpečnostních certifikátů k internetovému bankovnictví. Program zaznamená váš pohyb myší v prázdném okénku a pomocí změny jejích souřadnic vygeneruje žádaný soubor.

V minulosti byla i snaha využít již existující data, jako například soubory s textem. I ty však mají svou strukturu. Přece jen se jistá písmena a slova opakují častěji a jejich použití je skoro až pravidelné.

Jistotou jsou fyzikální jevy. Pro generování se nechají použít různé druhy šumů, které se dají zachytit pomocí elektronických součástek. Jedná se například o šum diody, tranzistoru nebo teplotního čidla. Některá zařízení dokonce fungují tak, že kamerou snímají oblohu, či lávovou lampu a z obrazu a jeho změn získávají požadovaná data.

Základním požadavkem zdroje je dostatečná frekvence změn, která umožní potřebnou rychlost generování.

Nejlepším zdrojem by bylo měření poločasu rozpadu radioaktivního prvku. Toto řešení je však velmi nákladné a náročné na laboratorní vybavení. Nemluvě o dostupnosti radioaktivního materiálu pro veřejnost.

2.4 Testování

Hned na začátek je potřeba zmínit fakt, že žádný z testů nedokáže z dat rozlišit, zda se jedná o generátor skutečný nebo pseudonáhodný. Tedy pokud test předem neví, jakým algoritmem byl vygenerován a jeho semínko, na jehož základě pracoval. Tudíž se všechny testy používají bez ohledu na druh generátoru. V obou případech totiž dochází i k obdobným nedostatkům. Záleží, jak kvalitní zdroj nebo algoritmus byl použit.

Vzhledem k oblasti a důležitosti používání generátorů byl od jejich vzniku vytvořen nespočet statistických testů. Ty se používají po skupinách, které se nazývají *baterie*. Obsahují výpočty na odhalování jednoduchých chyb, jako jsou dlouhé řady stejných čísel, až po složitější statistické – rozbor rozložení jednotlivých datových bloků.

Základem pro testy je dostatek dat. Zdrojem bývá pole čísel, textový řetězec nebo bitový stream. Ten se prožene příslušným algoritmem, který po vykonání vrátí číselnou hodnotu. Aby se testy mezi sebou daly porovnávat a určilo se podle nich, zda je daná sekvence náhodná, musí se používat jednotné měřítko alias p-value. Tato hodnota se pohybuje v intervalu <0,1>. Výsledek záleží na kriteriích daných dokumentací k testu. Minimální hodnota pro splnění náhodnosti je obvykle 0.01. Po provedení více testů se podle ní určí pouze to, zda sekvence dat je či není náhodná. Nelze ve většině případů s jistotou říct, že větší čísla znamenají lepší kvalitu generátoru.

Testy se snaží odhalit pravidelnost, závislost prvků mezi sebou a jejich výsledné rozložení. Ideální je, že všechna čísla stejné hodnoty jsou od sebe přibližně stejně vzdálena, tudíž i jejich počet by měl být stejný. V úvahu nepřichází lineární závislost. Dokonce pokud se budete snažit zdrojová data násobit mezi sebou nebo nějakým koeficientem, výsledek testu se nezmění.

2.4.1 Příklad Maurerova universálního statistického testu

Celý test je podrobně rozebrán v publikaci [1] na straně 42.

Hlavním účelem je zjistit, zda se daná sekvence dat lze zkomprimovat bez ztráty informací. V tomto případě by nebyla náhodná. Jako příklad se dá uvést komprimace pomocí programu winzip. Pokud soubor bude obsahovat bloky shodných a opakujících se dat, program je vyřadí. Proto nelze jeho nekonečným aplikováním na výsledný soubor získat zlomkovou velikost. Při jeho užití na soubor s opravdu náhodnými daty program může i zkolabovat. V lepším případě bude beze změny. Na podobném principu Maurer právě pracuje.

Zdrojová data o délce n se rozdělí po L bitech do pole rozděleného na dvě části. První je určen pro inicializační sekvenci o Q prvcích. Mezivýsledkem je suma $\log_2$ vzdáleností mezi stejnými bloky.

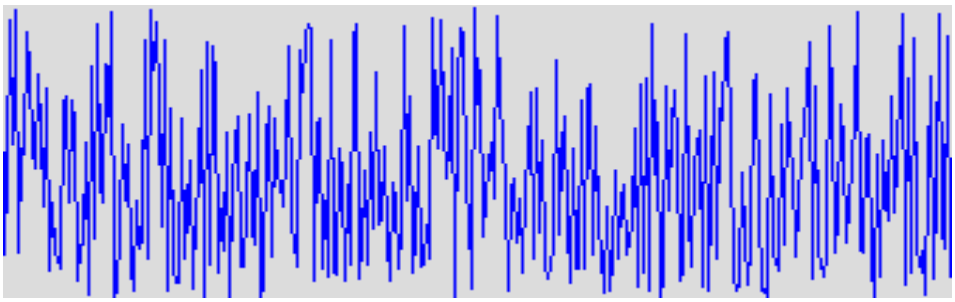
V inicializační sekvenci se první pole naplní indexy posledního výskytu všech prvků. K tomu je potřeba dostatek dat, aby žádná hodnota nebyla nulová. V rámci druhé části pole délky K se spočítá statistika $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K}\log_2(i - T_j)$, kde Tj je položka tabulky, která obsahuje decimální reprezentaci položek v j-tém L-bitovém bloku. Z výsledku, po dosazení tabulkových konstant do vzorce, získáme hodnotu P. Pokud je obdržená hodnota větší než 0.01, dá se zdroj považovat za náhodný.

Tento test je datově náročný. Pro délku bloku L=8 je potřeba přes 2Mb dat. Nicméně je velmi přesný, přísný a spolehlivý.

3 Návrh vlastního HW generátoru

3.1 Zdroj entropie

V počátcích práce jsem se rozhodl pro využití bílého šumu. Ten se charakterizuje jako náhodný signál s rovnoměrnou spektrální hustotou. Má stejný výkon v jakémkoliv pásmu a obsahuje všechny frekvence. V praxi se používá od uspávání dětí až po architektonickou akustiku. Krom toho i ke generování náhodných čísel.



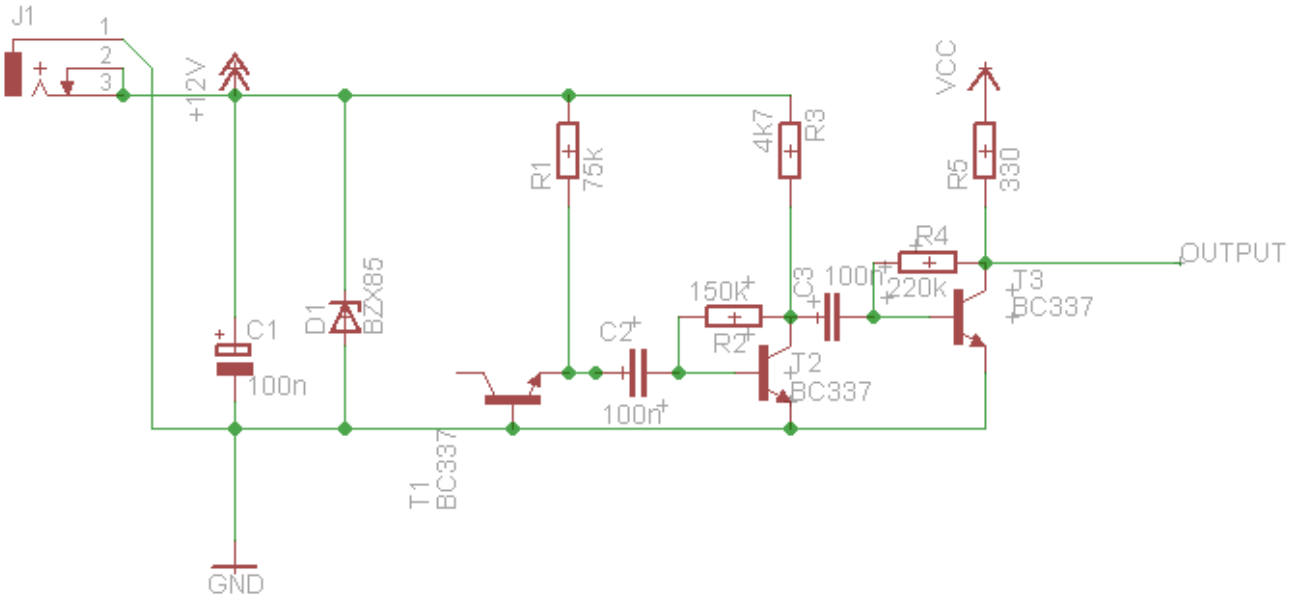
Obrázek 4 - bílý šum

3.2 Zisk signálu

Obvod napájí 12V adaptér, jehož napěťová niance je stabilizována kondenzátorem a Zenerovou diodou.

Šum je odebírán na PN přechodu universálního tranzistoru v zapojení s otevřeným kolektorem. Měl by tak být obohacen o rušení signály z okolního prostředí, jakými jsou například radiové vlny. To vše při napětí na kolektoru kolem 9 V. Slabý signál je potřeba zesílit. To obstarává navazující zesilovač, oddělen kondenzátorem, za účelem neovlivnění pracovního bodu druhého tranzistoru. Přizpůsobení na požadované hodnoty provádí další zesilovací stupeň. Velikosti odporů byly z počátku přibližně určeny pomocí výpočtů a při testování doladěny pomocí trimrů.

Poslední stupeň je připojen, na obvodem 7805, stabilizované napětí 5V, které slouží k napájení integrovaných obvodů na desce.



Obrázek 5 - schéma šumového generátoru

3.3 Digitalizace dat

Pro digitalizaci a zpracování byl původně určen klon mikroprocesoru řady x51 Atmel AT89C51CC01 v pouzdře PCL44. Byly s ním však velké problémy týkající se samotného zprovoznění i návrhu desky. Měl až zbytečně velký počet pinů a vzhledem k pouzdru by musela být oboustranná.

Jako lepší řešení se zdálo použít menší a novější procesor. Vybral jsem si PIC12F675, který taktéž obsahuje A/D převodník, ale v pouzdře DIL8, což umožnilo jednodušší návrh desky. Na druhou stranu bylo potřeba si vypůjčit vývojový kit, pomocí nějž se programuje a to PICKIT 2. Obsluha jeho aplikace je velmi jednoduchá.

Práci usnadnilo i prostředí mikroC for PIC, které obsahuje spoustu knihoven pro obsloužení periferií, jakým je i A/D převodník. Ten má v základu rozlišení 10 b a jeho reference se dá nastavit přímo na napájecí pin.

3.4 Rozhraní

Pro připojení k PC bylo vybráno rozhraní RS-232. Problém však bylo to, že procesor tímto rozhraním nedisponuje. Komunikace se tak provádí pomocí knihoven softwarově. Vzhledem k volbě využití interního oscilátoru 4MHz, je maximální rychlost sběrnice 4800 baudů/s, což znamená, že výsledný užitečný datový tok činí přibližně 3,6 Kb. Komunikace probíhá 8 bitově, bez parity s jedním stop bitem, tudíž nebylo potřeba ani využít celou šířku převodníku.

Celý program v mikroprocesoru je opravdu jednoduchý a účelný.

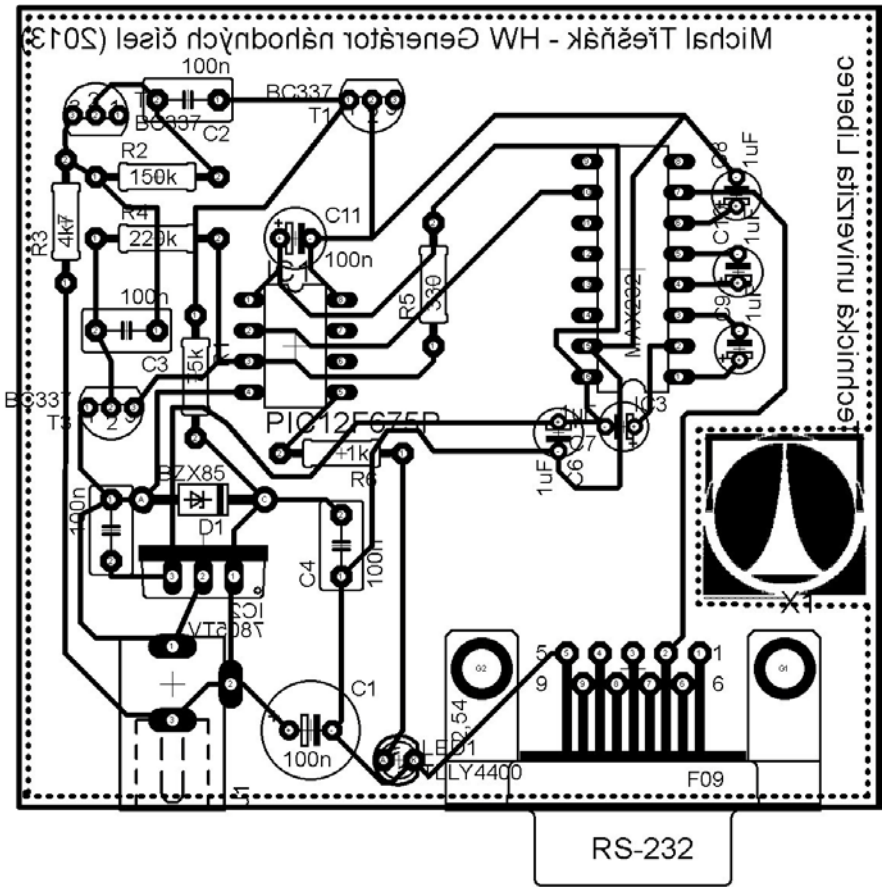
```
#define LED GP2_bit
unsigned adc;
char error;

void main() {

  GPIO = 0x00;
  CMCON = 0x07;
  TRISIO = 0x00;
  TRISIO = 0x10;
  ANSEL = 0x10;

error = Soft_UART_Init(&GPIO, 0, 5, 4800,0);
ADC_Init();

if (error == 0)
{
LED=1;
while(1)
{
        adc = ADC_Get_Sample(3);

        Soft_UART_Write(adc);

}
}else LED = 0;

}
```

Obrázek 6 - program pro procesor v jazyce C

Hardwarová kompatibilita se zajišťuje prostřednictvím obvodu MAX232, který disponuje dvěma kanály, z nichž byl použit pouze jeden a to jednosměrně. Pro připojení kabelu pro propojení s počítačem je použit konektor CANON9.

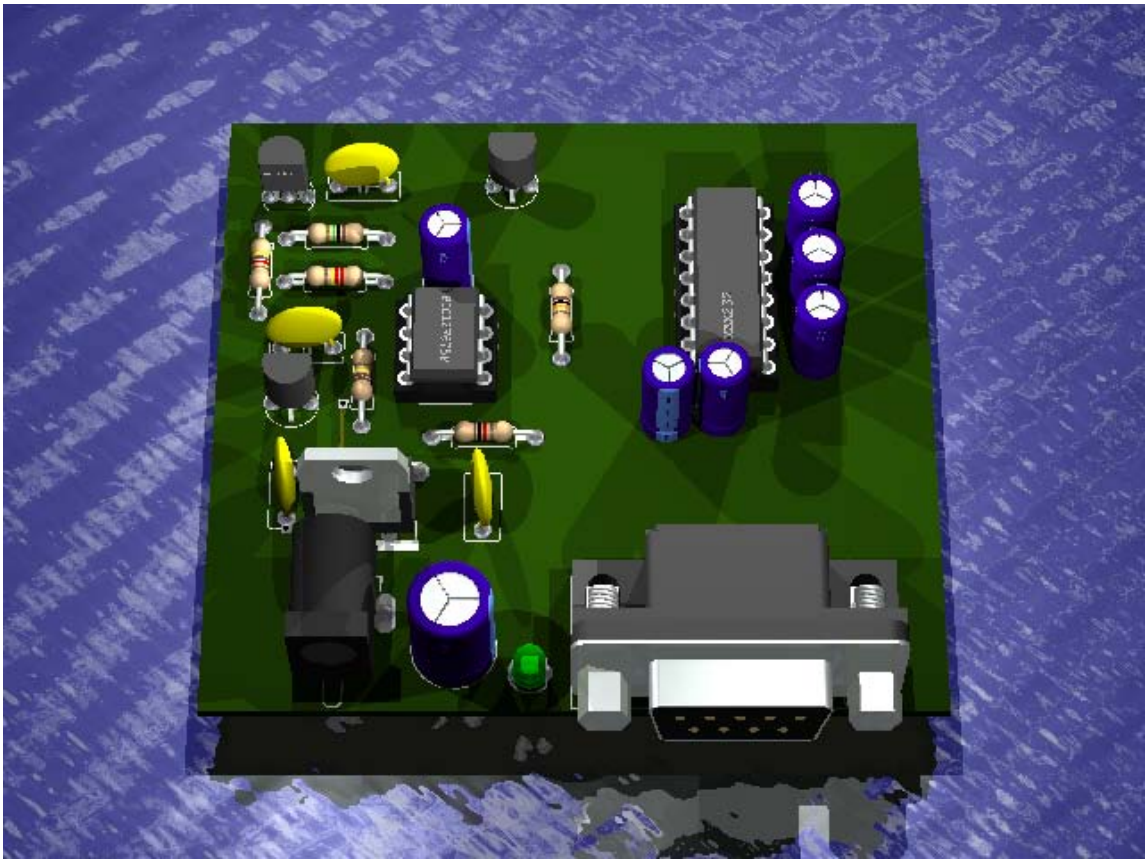
3.5 Deska plošný spojů

Návrh probíhal v programu EAGLE 6.2.0. Nejdříve bylo nutné sestavit elektrické schéma zapojení. Následně rozložit součástky na desku plošných spojů a vhodně zvolit cesty vodičů. Bylo potřeba změnit pájecí plošky v knihovnách, aby se uzpůsobily ručnímu pájení. Na návrhu, který je vidět na obrázku 7 , je pro větší přehlednost vynechána tzv. rozlitá měď, která je jinak propojena se zemí (GND), zajišťující odstínění obvodu.



Obrázek 7 - návrh DPS

Pomocí přidané funkci EAGLE3D, která vygenerovala data pro renderovací software POV-Ray, je možné vytvořit přibližnou podobu výsledného obvodu po osazení.



Obrázek 8 - přibližný vzhled výsledné a osazené DPS

Part	Value	Package	Library	Position (mil)	Orientation
C1	100n	E3,5-8	rcl	(950 250)	R0
C2	100n	C050-035X075	untitled	(550 2300)	R180
C3	100n	C050-035X075	untitled	(350 1550)	R180
C4	100n	C050-035X075	untitled	(1050 850)	R90
C5	100n	C050-035X075	untitled	(300 900)	R270
C6	1uF	E1,8-4	untitled	(1757 1216)	R270
C7	1uF	E1,8-4	untitled	(1969 1237)	R180
C8	1uF	E1,8-4	untitled	(2336 1989)	R90
C9	1uF	E1,8-4	untitled	(2346 1490)	R90
C10	1uF	E1,8-4	untitled	(2343 1740)	R90
C11	100n	E2,5-5	rcl	(900 1850)	R0
D1	BZX85	DO41Z10	untitled	(600 1000)	R180
IC1	PIC12F675P	DIL8	microchip	(900 1500)	R270
IC2	7805TV	TO220V	linear	(600 850)	MR0
IC3	MAX232	DIL16	maxim	(2000 1750)	R90
J1		DCJ0303	con-jack	(500 400)	R0
LED1	TLLY4400	LED3MM	untitled	(1236 153)	R0
R1	75k	0207/10	untitled	(575 1400)	R270
R2	150k	0207/10	untitled	(450 2050)	R0
R3	4k7	0207/10	untitled	(153 1905)	R90
R4	220k	0207/10	untitled	(450 1850)	R0
R5	330	0207/10	untitled	(1350 1700)	R90
R6	1k	0207/10	untitled	(1050 1150)	R180
T1	BC337	TO92	untitled	(1150 2300)	R180
T2	BC337	TO92	untitled	(250 2250)	R0
T3	BC337	TO92	untitled	(300 1300)	R180
X1	RS-232	F09HP	untitled	(2000 450)	R180

Obrázek 9 - Partlist

4 Softwarová část

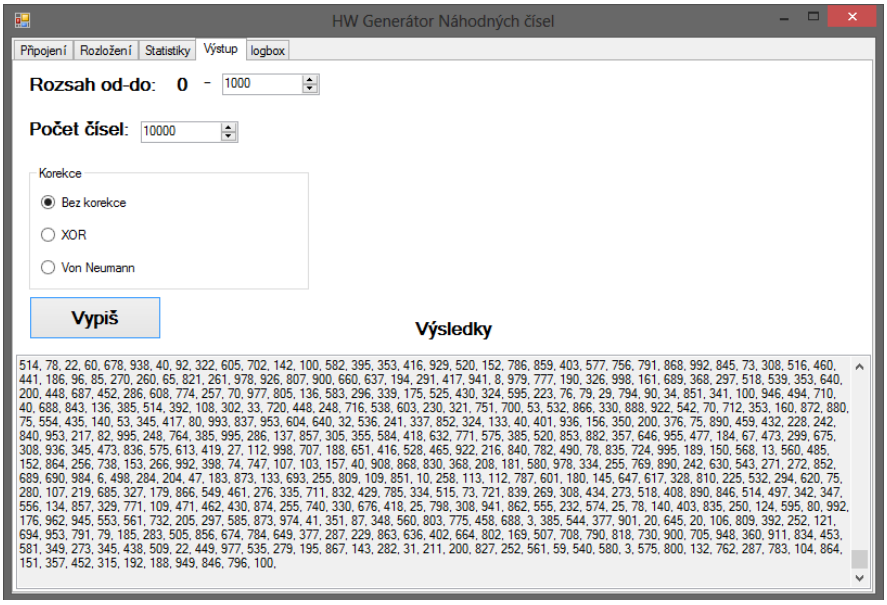
Sám o sobě je generátor k ničemu. Pokud z něj chceme data získat a zpracovat, potřebujeme uživatelskou aplikaci, která umožní připojení k samotnému hardwaru. Pro její tvorbu jsem si zvolil jazyk C# a vývojové prostředí Microsoft Visual Studio 2012.

Aplikace poskytuje uživateli se připojit k příslušnému portu. Pokud takto učiní a změní stav na „Online data“, v rámečku se začnou promítat, po vteřině aktualizované, poslední přijaté vzorky z generátoru a zobrazí se aktuální rychlost generátoru. V tomto okamžiku se začínají sbírat a ukládat přijatá data. K dispozici jsou tři volby pro korekci. Možné je také vygenerovat hodnoty pseudonáhodného generátoru pro porovnání výsledků se skutečným.

V druhé záložce je zobrazeno rozložení čísel jak PRNG, tak i TRNG. Hodnoty se nacházejí ve 256 sloupcích (8 bitová čísla), přičemž každý z nich reprezentuje počet výskytů každého čísla. Po najetí na požadovaný sloupec se jeho hodnota zobrazí. Červený sloupec reprezentuje nejvyšší počet. Z grafu je vidět, jestli některé hodnoty nepřevládají. Zobrazení záleží na zvolené korekci. Graf též přizpůsobí zobrazení podle největšího sloupce, nebo šířce okna aplikace.

Třetí záložka je určena pro výsledky provedených testů, které se provedou po její aktivaci. To může podle množství dat i chvíli trvat. Ve čtyřech sloupcích jsou následující údaje: název testu, výsledek, hodnota p-value a použitá korekce. Testy se aplikují na všechna dostupná data.

Záložku „Výstup“ není nutné moc komentovat. Vypíše požadovaný počet čísel v daném rozsahu s použitím vybrané korekce.



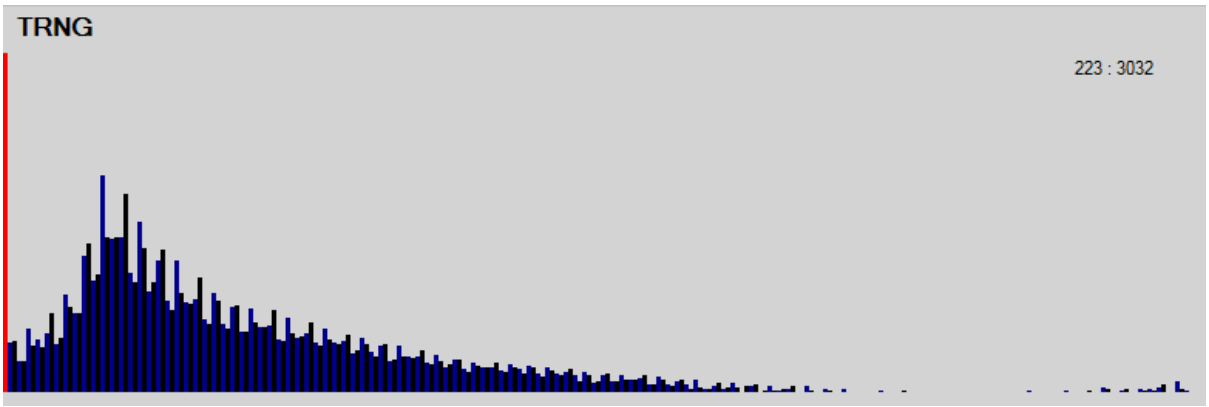
Obrázek 10 - Okno výstupu

4.1 Připojení a ukládání dat

Připojení probíhá skrze komponentu SerialPort. Po startu programu se načte aktuální seznam aktivních portů. Pokud je program připojen online, automaticky reaguje na příchozí data z bufferu a ukládá je do pole typu byte. Odtud jsou přístupná pro další zpracování.

4.2 Vygenerovaná data

Potom, co je program k přípravku úspěšně připojen, lze si data zobrazit jejich interpretaci v kartě „rozložení“.



Obrázek 11 - rozložení surových dat

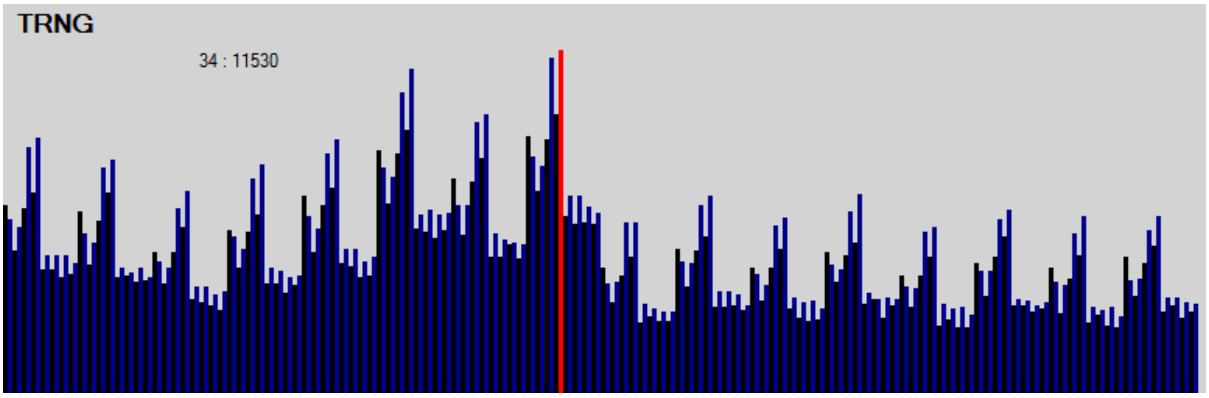
Na obrázku 11 je vidět, které hodnoty jsou zastoupeny nejpočetněji. Nejvyšší sloupec označuje jako položku výskytem číslo 0. Její výskyt je takový, že se menší hodnoty skoro nezobrazují. Tento problém způsobuje kondenzátor před posledním zesilovacím stupněm, od kterého vznikají malé záporné napěťové špičky, kterých se mi nepodařilo se zbavit. Všimnout si můžeme i samotného tvaru, který se k rovnoměrnému rozložení moc nepřibližuje. Proto je potřeba provést korekce.

4.3 Korekce dat

Pokud nejsou přijatá data dostatečně „kvalitní“, je třeba použití bitových korekcí, protože jak již bylo dříve zmíněno – jejich násobení na výsledku nic nezmění. Ty by měly zlepšit poměr jedniček a nul a jejich rozložení v celém rozsahu. Aplikace používá korekce dvě. První je pomocí funkce XOR, která porovnává vždy dvě sousední hodnoty. Pokud jsou stejné, výstupem je 0, pokud rozdílné, tak 1. Po zpracování tak dostaneme pouze poloviční počet dat. Další možností je použít metodu Von Neumanna. Ta souhlasné sousední dva bity eliminuje úplně. Jinak platí 1,0 => 1 a 0,1 => 0. Objem výsledných dat se zmenší na 25% původní velikosti.

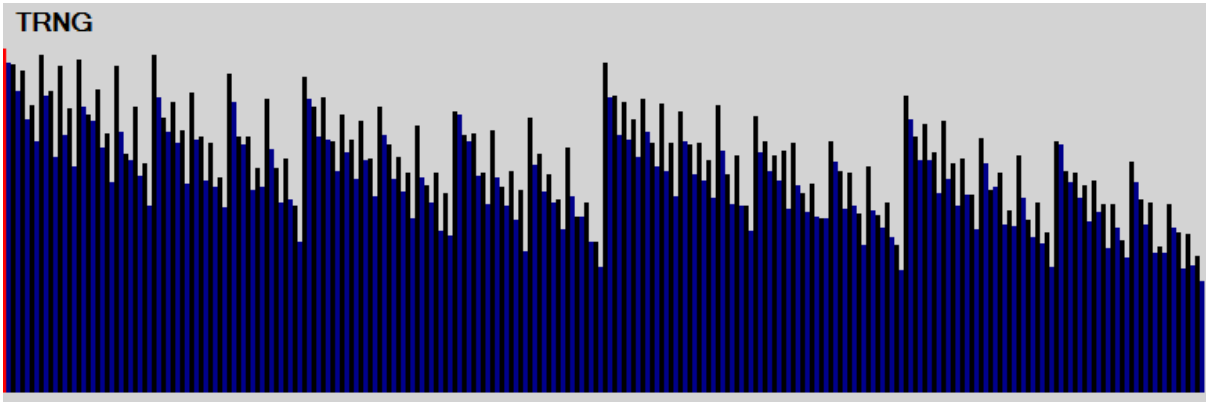
Před provedením korekcí se musí pro jednodušší zpracování bytové pole převést na bitový stream. Po provedení je zkonvertován zpět do původního datového formátu.

Na dalším obrázku můžeme vidět, co s daty provede XOR. Špička, která byla předtím u čísla 0, se i s ostatními daty rozprostřela.



Obrázek 12 - korekce XOR

S pomocí metody Von Neumanna dostaneme následující výsledek. Je třeba pamatovat na to, že jeho účinnost je spjata s eliminací 75% dat.



Obrázek 13 - korekce Von Neumann

4.4 Testování

Pro ověření funkce generátoru, je nutné provést testy náhodnosti. Vybral jsem si čtyři z publikace pro počítačovou bezpečnost [1] a jejich matematické zápisy převedl na testovací algoritmy.

První test (monobit) se zaměřuje na poměr jedniček a nul v celém rozsahu. V případě náhodných dat by tento poměr měl být 1:1. V jiném případě může být výsledek testu negativní. Pro pozitivní výsledek testu je potřeba splnit podmínku, že hodnota p-value je větší než 0.01.

Druhý (Block test) zkoumá stejnou vlastnost, ale pouze v rámci jednotlivých x - bitových bloků.

Dalším testem zjistíme, zda je v blocích konzistentní počet jedniček za sebou takový, jaký by měl v náhodně generované sekvenci být.

Poslední (Maurerův test) byl popsán podrobněji v části 2.4.1.

4.5 Výsledky testů

Pro provedení všech testů na všech typech dat (surová, XOR, Von Neumann) je potřeba vygenerovat více než milion náhodných čísel. Hlavně kvůli kombinaci – Von Neumanovy metody a Maurerova testu, aby byl test proveden korektně dle zadání jeho požadavků. Díky nepříliš velké přenosové rychlosti je celý proces poměrně zdlouhavý.

Po vygenerování přibližně dvou milionů čísel (0-255), byly získány výsledky testů.

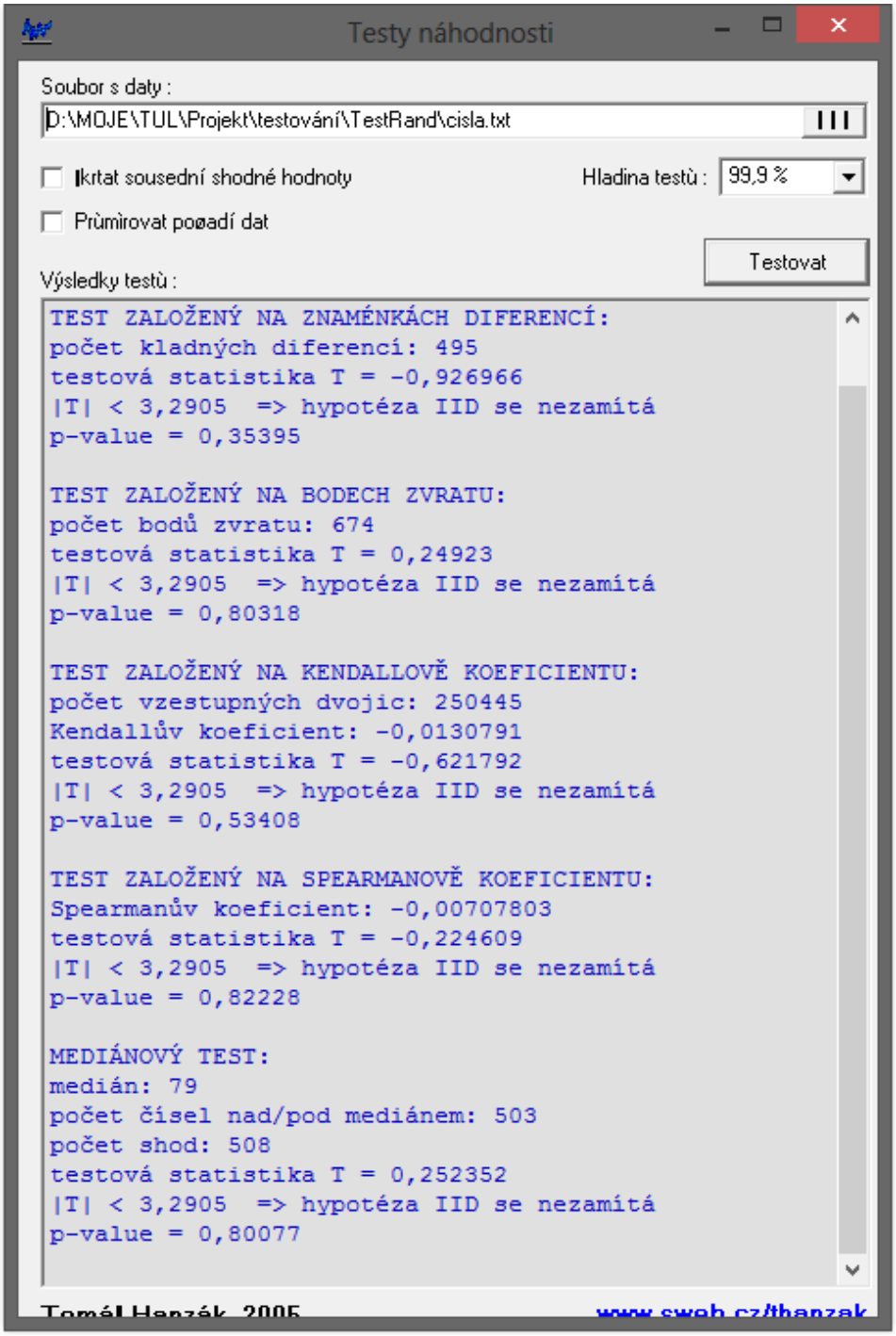
Test	výsledek	hodnota p-value	Provedená korekce
Monobit	NE	0,0000	Bez korekce
	OK	0,5900	XOR
	NE	0,0000	Von Neumann
Blok test	NE	0,0003	Bez korekce
	OK	0,9999	XOR
	OK	0,2370	Von Neumann
Ones in block	OK	0,5500	Bez korekce
	OK	0,6480	XOR
	OK	0,3660	Von neumann
Maurer	NE	0,0000	Bez korekce
	NE	0,0000	XOR
	NE	0,0002	Von Neumann

Obrázek 14 - tabulka výsledků TRNG

Z tabulky můžeme vyčíst, že si generátor vedl docela dobře. V prvním testu se ukázala nejlepší a také jako jediná průchozí data po průchodu korekce XOR. Druhý skončil špatně pouze pro surovou hodnotu. Co se týče testování na obsah jedniček v blocích, obstály všechny metody.

Poslední test byl zcela bez úspěchu. Důvodem tohoto výsledku je pravděpodobně nerovnoměrné rozložení čísel.

Pro neúspěch posledního testu jsem se rozhodl použít ještě jiné externí testy. Použil jsem program TestRand z webu thanzak.sweb.cz/. Do něj jsem vložil přibližně tisíc generovaných hodnot bez korekce.

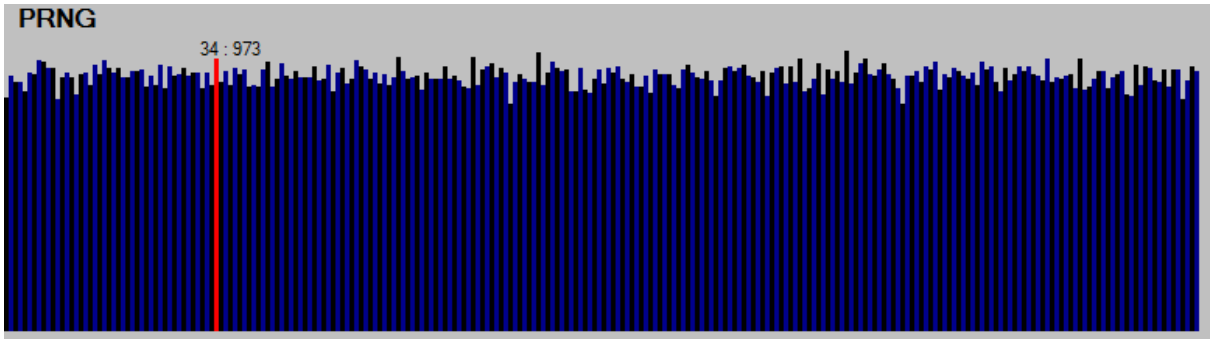


Obrázek 15 - externí testy

Z výsledků je vidět, že testovaná data prošla bez potíží. Nyní můžeme považovat generátor opravdu za náhodný.

4.6 Porovnání s PRNG

Vlastní aplikace mimo jiné umožňuje vygenerovat pseudonáhodná čísla a otestovat je stejně jako ta skutečná. Získání dostatečného množství dat trvalo pouze okamžik.



Obrázek 16 - rozložení PRNG

Vzhledem k rovnoměrnému rozložení nebylo třeba používat korekce a mohlo se přistoupit rovnou k testování.

Test	výsledek	hodnota p-value
Monobit	OK	0,3471
Blok test	OK	0,9038
Ones in block	OK	0,3662
Maurer	OK	0,1955

Obrázek 17 - tabulka výsledků PRNG

Generátor splnil všechna kritéria k tomu, aby mohl být považován za náhodný. Mohlo by se zdát, že výsledek je tedy lepší než generátor skutečný. Problém nastává v případě, že tento algoritmus začne od stejného místa. V tom případě dostaneme úplně stejnou řadu čísel, což je nežádoucí. Tato situace se stává, poněvadž jeho seed není získáván pomocí skutečného generátoru. Mohu potvrdit z vlastních zkušeností s prací s ním.

5 Závěr

Použití jednoho či druhého generátoru záleží na druhu činnosti, ke které bude určen. HW generátory jsou v každém případě lepší pro případy, kdy se požaduje velké zabezpečení dat. Nicméně v mnoha případech mohou tuto možnost ztížit náklady, jež stoupají s rychlostí generátoru. V dnešní době je tak většinou používána mutace obou řešení, kdy je skutečně náhodná sekvence (seed) předána pseudonáhodnému generátoru, jehož rychlost je mnohem větší a záleží tak jen na kvalitě algoritmu. Většina dnes běžně používaných pseudonáhodných generátorů splňuje většinu testů a lze je tedy plně a bez obav užívat.

Při návrhu vlastního HW generátoru jsem se snažil vybrat spolehlivý zdroj entropie. O jeho výběru rozhodovala možnost jednoduchého zpracování signálu. A to pomocí dostupného a levného mikroprocesoru s A/D převodníkem. Lepší rozhodnutí mohlo být, použít externí oscilátor, který by zajistil vyšší komunikační rychlost. Otázkou zůstává, zda by se ve výsledku objevovalo více záporných špiček, kterých se mi nepodařilo se zbavit. Většího výkonu by se určitě dosáhlo i v případě, že by se korekce prováděly přímo na mikročipu. Toto řešení jsem však zamítl z důvodu potřeby surových dat pro interpretaci účinnosti korekcí. Ty se ukázaly jako potřebné, protože počáteční rozložení nebylo zrovna ideální.

Snažil jsem se vybrat jak testy obecné, tak i jeden zcela striktní. Po jejich provedení v aplikaci se jevil generátor jako náhodný. Pro ověření jsem použil externích testů, které tuto skutečnost potvrdily. Zadání tak považuji za splněné.

Bibliografie

[1] J. S. L. V. AndrewRukhin, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," 2010.

[2] M. Oleár, „Mikrokontroléry PIC 1.díl: začáname,“ Elweb, 1999-2013. [Online]. Available: http://www.elweb.cz/clanky.php?clanek=64.

[3] O. Source, „Programujeme jednočipy,“ 17 Listopad 2012. [Online]. Available: http://cs.wikibooks.org/wiki/Programujeme_jedno%C4%8Dipy. [Přístup získán 2012].

[4] R. Davies, „True random number generators,“ [Online]. Available: http://www.robertnz.net/true_rng.html. [Přístup získán 2012].

[5] M. Malý, „Hardwarový generátor náhodných čísel aneb náhoda z atomů,“ root.cz, 2 Prosinec 2010. [Online]. Available: http://www.root.cz/clanky/hardwarovy-generator-nahodnych-cisel-aneb-nahoda-z-atomu/. [Přístup získán 2012].

[6] Nezveřejněný, „A Random Bit Generator for EVPmaker,“ Tonbandstimmen, [Online]. Available: http://www.tonbandstimmen.de/evpmaker/random-bit-generator/index_e.htm. [Přístup získán 2012].

[7] S. C. Magazine, „Circuit Notebook,“ 1996-2013. [Online]. Available: http://archive.siliconchip.com.au/cms/A_103659/article.html. [Přístup získán 2013].

[8] Nezveřejněný, „Test for random number generators,“ VA, [Online]. Available: http://www.cs.hku.hk/cisc/projects/va/index.htm. [Přístup získán 2013].

[9] J. McCaffrey, „Randomness in Testing,“ Microsoft, 2013. [Online]. Available: http://msdn.microsoft.com/en-us/magazine/cc163551.aspx. [Přístup získán 2013].

[10] M. Š. Martin Nejedlík, „Hodnocení kvality generátorů náhodných a pseudonáhodných čísel pro kryptografické aplikace,“ Univerzita Hradec Králové, 2003. [Online]. Available: http://kryptologie.uhk.cz/generatory.htm. [Přístup získán 2013].

[11] V. Klíma, „Testy a zdroje neurčitosti v počítači,“ Decros, [Online]. Available: http://scar.borec.cz/krypto/nahodnost.html. [Přístup získán 2013].

[12] C. Pit-Claudel, „How random is pseudo-random? Testing pseudo-random number generators and measuring randomness,“ 2012. [Online]. Available: http://pit-claudel.fr/clement/blog/how-random-is-pseudo-random-testing-pseudo-random-number-generators-and-measuring-randomness/. [Přístup získán 2013].

[13] J. S. J. N. Andrew Rukhin, „A statistical test suite for random and pseudorandom number generators for cryptographic applications,“ NIST, 2001. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22b.pdf. [Přístup získán 2013].

[14] B. Federmann, „Jak na 3D Eagle,“ Federmann, 2013. [Online]. Available: http://www.federmann.cz/index.php/vyrobni-postupy/64-navrhy/75-jak-na-3d-eagle.html. [Přístup získán 2013].

[15] Calomel.org, „Entropy and Random Number Generators,“ Calomel, 13 Březen 2013. [Online]. Available: https://calomel.org/entropy_random_number_generators.html. [Přístup získán 2013].

16] Microsoft, „SerialPort Class,“ Microsoft, 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx. [Přístup získán 2013].

Přílohy

1) Práce ve formátu docx
2) Práce ve formátu PDF
3) NIST.pdf - J. S. L. V. AndrewRukhin, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," 2010.
4) Projekt pro grafickou aplikaci pro Microsoft Visual Studio 2012
5) Projekt k programu EAGLE