XXII Seminarium

ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2012
Oddział Gdański PTETiS
Referat nr 3

GENERATOR LICZB LOSOWYCH WYKORZYSTUJĄCY MIKROKONTROLER ATMEGA16

Dominik AMBROZIAK[1], Mariusz DĄBKOWSKI[2], Grzegorz REDLARSKI[3]

1. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-233 Gdańsk
   tel.: 058 347 19 94 fax: +48 (58)347 21 36 e-mail: dambroziak@ely.pg.gda.pl
2. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-233 Gdańsk
   tel.: 058 347 24 73 fax: +48 (58)347 21 36 e-mail: m.dabkowski@ely.pg.gda.pl
3. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-233 Gdańsk
   tel.: 058 347 23 17 fax: +48 (58)347 21 36 e-mail: g.redlarski@ely.pg.gda.pl

Streszczenie: Istotę artykułu stanowi opis rozwiązania problemu dotyczącego syntezy generatorów liczb losowych: fizycznych oraz komputerowych – programowych, bazujących na układach scalonych wykorzystujących moduł mikrokontrolera serii Atmega16. Z tego względu we wstępie przedstawiono genezę, zastosowanie oraz rodzaje generatorów liczb losowych, natomiast w dalszej części artykułu opisano przykładowe generatory liczb quasi-losowych. Następnie zaprezentowano istotę podjętego problemu oraz propozycję jego rozwiązania. Na zakończenie przedstawiono krótkie podsumowanie oraz podano najważniejsze wnioski.

Słowa kluczowe: generator liczb losowych, mikrokontroler Atmega16, generatory liczb losowych.

1. WPROWADZENIE

1.1. Historia generatorów liczb losowych

Pierwsze opisywane w literaturze przedmiotowej [1, 2, 4] generatory liczb losowych, w praktyce stanowiły tablice z ciągiem losowo ułożonych liczb, z zakresu od 0 do 9. Chronologicznie, pierwsza tego typu tablica została opracowana przez brytyjskiego fizyka i statystę *L. H. Tippett'a* w roku 1927 [1]. Uczony ten uzyskał ciąg 41600 cyfr (z zakresu od 0 do 9) na podstawie szeregu liczb wyrażających powierzchnię brytyjskich parafii. Zważając następnie na fakt, że liczba wyrażająca wielkość parafii nie jest cyfrą, *Tippett* każdą z liczb poddał odpowiedniej obróbce wynikającej z opracowanego przez siebie algorytmu. Propozycja odnośnego algorytmu polegała na usunięciu dwóch pierwszych oraz dwóch ostatnich cyfr każdej liczby, a następnie na utworzeniu odpowiedniego wpisu do tablicy – pozostałej, cyfry środkowej [1]. Na przestrzeni kolejnych dekad rozmiary tablic nieustannie rosły, a metody ich otrzymywania stawały się coraz bardziej skomplikowane. Obecnie natomiast tablicowe generatory liczb zostały niemal całkowicie wyparte, a ich miejsce zastąpiły tzw. generatory komputerowe. Aktualnie, za pierwszy z takich generatorów uznawany jest *algorytm kwadratowy von Neumana* [2].

1.2. Rodzaje i sposoby wykorzystywania generatorów

Wynik rzutu kostką do gry, wybór pojedynczego losu z urny, czy wynik gry w ruletkę jest nazywany zdarzeniem losowym. Analizując przykładowo rzut kostką do gry, wykonuje się serię niezależnych zdarzeń losowych, a następnie – zapisując zaobserwowane wyniki – tworzy się na odpowiednim zborze ciąg liczb losowych {1; 2; 3; 4; 5; 6}. Zakładając, że próby te wykonane są z należytą starannością (przy pomocy kości w przypadku, której istnieje jednakowe prawdopodobieństwo uzyskania każdego z wyników) można stwierdzić, że taka zmienna losowa ma rozkład równomierny dla całego zbioru liczb {1; 2; 3; 4; 5; 6}. Wówczas opisywana kostka do gry, pozwalająca na uzyskanie pewnego ciągu liczb, może być nazwana generatorem liczb losowych o rozkładzie równomiernym.

W ogólnym przypadku ciągiem liczb losowych zwykło się nazywać taki ciąg, którego nie można zapisać za pomocą algorytmu, w postaci krótszej od niego samego. W związku z tym na podstawie znajomości poprzednich wyrazów takiego ciągu nie jest możliwe wygenerowanie kolejnych jego wartości, ani też (na podstawie znajomość wszystkich jego wyrazów) nie jest możliwe opracowanie reguł algorytmu, które pozwalałyby odtworzyć ten ciąg. Stąd też generatory liczb, których ciągi są tworzone na podstawie pewnych reguł są nazywane ciągami liczb quasi-losowych. Inny podział obowiązuje w przypadku metod generowania tych ciągów. Przykładowo wynik rzutu kotką do gry będzie nazywany fizycznym generatorem liczb losowych, a program komputerowy bazujący na odpowiednim algorytmie będzie nazywany programowym (lub komputerowym) generatorem liczb quasi-losowych [1].

W praktyce inżynierskiej, w pełni efektywne opracowanie i wykonanie generatorów fizycznych nie jest z reguły możliwe. Jest to zazwyczaj związane ze słabą ich stabilnością, w przypadku której nawet niewielkie zmiany warunków otoczenia (lub własności fizycznych) mogą istotnie wpływać na zmiany własności probabilistycznych generatora. Rozwiązanie tego problemu wiąże się zatem z koniecznością implementacji w układzie fizycznym szeregu

Recenzent: Prof. dr hab. inż. Krzysztof Karwowski – Wydział Elektrotechniki i Automatyki Politechnika Gdańska

dodatkowych elementów (testujących oraz korekcyjnych), których działanie w efekcie znacznie komplikuje całą konstrukcję generatora. Z tego względu większość obecnie uzyskiwanych liczb losowych (lub quasi-losowych) jest otrzymywana z pomocą generatorów komputerowych, zwanych także programowymi. Ciągi generowanych wartości są wówczas zdeterminowane przez pewne reguły (algorytmy), których realizacja prowadzi do uzyskania szeregu liczb quasi-losowych. Okazuje się ponadto, iż wskazana wada generatorów programowych w większości przypadków może zostać pominięta, bowiem otrzymane liczby quasi-losowe charakteryzują się na tyle wysoką losowością, by w praktyce na ich podstawie możliwe było np. prowadzenie odpowiednich badań naukowych, czy obliczeń [2].

Znajomość zagadnień związanych z generatorami liczb losowych pozwala na określenie głównych obszarów ich stosowania. Obecnie, w odnośnym zakresie, można wyróżnić cztery takie dziedziny. Pierwszą z nich stanowią wszelkie badania statystyczne, w których istotnym elementem rozpatrywanego procesu są dane o charakterze losowym analizowanych próbek. Tego typu zagadnienia są z reguły wykorzystywane w licznych zagadnieniach ekonomicznych, marketingowych i społecznych. Drugą dziedziną, w której opisywane generatory znalazły zastosowanie, są wszelakie badania symulacyjne. W wyniku ich stosowania możliwe staje się uwzględnienie wpływu szeregu czynników o charakterze losowym (np. zakłóceń), poprzez ich wprowadzenie do procesu. Istotnym zagadnieniem w odnośnym zakresie jest np. opracowanie złudnego zjawiska realizmu w grach komputerowych, czy w inteligentnych automatach do gier zręcznościowych. Również znana od lat i powszechnie stosowana metoda *Monte Carlo* bazuje na generatorach liczb losowych [3]. Z jej pomocą można poszukiwać rozwiązań szeregu zagadnień wielowymiarowych, w tym również równań całkowych oraz równań różniczkowych i algebraicznych, a także szeregu procesów optymalizacji. Metody te są także powszechnie wykorzystywane w przypadkach, gdy klasyczne metody rozwiązania nie mogą zostać zastosowane np. z uwagi na ograniczenia obliczeniowe. Ostatnią dziedziną, w której generatory liczb losowych znajdują powszechne zastosowania jest ochrona informacji, często rozumiana, jako kryptografia. Podczas związanego z nimi szyfrowania strumieniowego generatory służą, jako klucze. Tego rodzaju zabiegi szyfrowania można spotkać szczególnie często w sieciach telefonii komórkowych oraz w sieciach komputerowych.

2. ALGORYTMY WYBRANYCH GENERATORÓW LICZB

2.1. Generator liniowy

Generator liniowy jest jednym z podstawowych oraz powszechnie stosowanych generatorów liczb quasi-losowych [4]. Wartości kolejnych liczb są wyznaczane z zależności

$$X_{n+1} = (a \cdot X_n) \bmod M \tag{1}$$

gdzie: X_{n+1} – wartość kolejnej liczby ciągu; X_n – wartość poprzedniej liczby ciągu; a, M – parametry o wartościach naturalnych [4]

W ciągu tym wartość kolejnej liczby jest równa iloczynowi liczby poprzedniej oraz parametru *a*, następnie podzielonych modularnie przez parametr *M*. Wartość *zerowej* liczby ciągu (tzw. ziarna – X_0 [4]) musi być znana przed uruchomieniem generatora i przyjmować wartości nie większe niż parametr *M*. Współczynnik *a*, przez który jest wymnażana wartość poprzedniej liczby także nie może przyjmować wartości równych lub większych niż wartość parametru *M*. Zbiór liczb, jaki generator może wygenerować mieści się zatem w przedziale od zera do *M*. Okres generatora, czyli powtarzalność ciągu, jest – jak wynika ze wzoru (1), zależny od wartości stałych *a* oraz *M*. W przypadku właściwie dobranych współczynników, maksymalny okres ciągu wynosi *M*-1.

2.2. Generator afiniczny

Rozszerzeniem generatora liniowego jest generator afiniczny [1], wyrażający się wzorem rekurencyjnym w postaci

$$X_{n+1} = (a \cdot X_n + b) \bmod M \tag{2}$$

gdzie: X_{n+1} – wartość kolejnej liczby ciągu; X_n – wartość poprzedniej liczby ciągu; a, b, M – parametry o wartościach naturalnych [1]

Wartość kolejnej liczby tego ciągu jest obliczana z iloczynu liczby poprzedniej oraz parametru *a*, do którego dodawana jest pewna stała wartość *b*. Następnie całość wyrażenia jest dzielona modularnie przez parametr *M*.

Jak wcześniej stwierdzono generator afiniczny jest rozszerzeniem generatora linowego. Jedyną różnicę pomiędzy tymi generatorami stanowi parametr *b*, którego wartość jest dodawana przed wykonaniem operacji dzielenia modularnego (2). Wartości parametrów *a* oraz *b* nie mogą przekraczać wartości stałej *M*, a okres tego generatora, podobnie jak w przypadku generatora liniowego, jest zależny od wartości parametrów *a*, *b* i *M*, a jego maksymalna długość wynosi *M*-1.

2.3. Generator reszt potęgowy

Przedstawione w poprzednich punktach generatory są tzw. generatorami liniowymi, natomiast generator reszt potęgowych stanowi generator nieliniowy [2]. Jego istota wynika z występowania w poniższym wzorze rekurencyjnym, elementu nieliniowego (kwadratu wartości)

$$X_{n+1} = \left(X_n^2\right) \bmod M \tag{3}$$

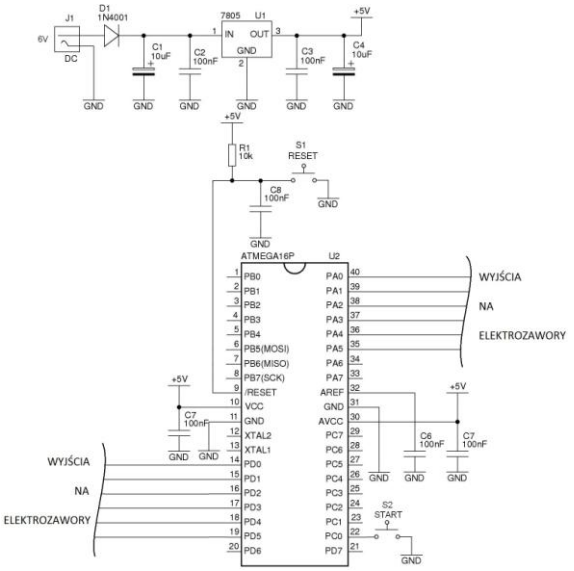
gdzie: X_{n+1} – wartość kolejnej liczby ciągu; X_n – wartość poprzedniej liczby ciągu; $M = pq$ – parametr naturalny [2]

Idea tego generatora polega na podnoszeniu do kwadratu poprzedniego wyrazu ciągu, a następnie na dzieleniu wyniku modularnie. Jednakże algorytm ten nie pozwala na uzyskiwanie wszystkich liczb naturalnych z przedziału od 0 do 9, dlatego też za kolejny element ciągu uznaje się tylko i wyłącznie ostatnią cyfrę liczby otrzymanej przez ten algorytm. W celu wydłużenia okresu do maksimum, wartość parametru *M* jest dzielona na dwie składowe *p* i *q*. Zmienne te powinny być liczbami pierwszymi, takimi że reszta każdej z nich (z dzielenia przez cztery) powinna stanowić wynik równy trójce (przykładowo: 7 i 19) [1].

3. ISTOTA PODJĘTEGO PROBLEMU ORAZ JEGO ROZWIĄZANIE

Podjęty w ramach artykułu problem polega na opracowaniu algorytmu, dzięki któremu obiekt fizyczny, jakim jest robot bokser [5], będzie mógł wyprowadzać uderzenia losową ręką oraz w losowych odstępach czasu. Robot jest wyposażony w dwa ramiona, przy czym każde z nich jest zasilane trzema siłownikami pneumatycznymi. Za dostarczanie medium do siłowników odpowiedzialne są elektrozawory 5/2 (po jednym dla każdego z siłowników). Propozycję rozwiązania tego zagadnienia stanowi synteza dwóch generatorów liczb losowych: fizycznego oraz komputerowego.

Przedstawione zagadnienie należy rozpatrywać w dwóch płaszczyznach: programowej oraz układowej, stąd też część programowa jest odpowiedzialna za sterowanie robotem oraz generację liczb losowych, natomiast druga – układowa, odpowiada za fizyczną realizację tego zadania (rys. 1).



Rys. 1. Schemat ideowy układu odpowiedzialnego za fizyczne zastosowanie mikrokontrolera Atmega16

Opracowanie części sprzętowej układu, której istotę działania stanowi mikrokontroler Atmega16 (rys. 1), było podyktowane następującymi przesłankami:

- niską ceną,
- łatwości programowania, szczególnie w językach strukturalnych – zastosowano język C,
- odpowiednią wielkością pamięci *flash*, która była w stanie pomieścić cały program sterujący, o objętości 16 kB.

W przedstawionym na rys. 1 rozwiązaniu można wyodrębnić dwa podzespoły: pierwszy – odpowiedzialny za zasilanie układu oraz drugi – odpowiedzialny za transmisję sygnałów przychodzących i wychodzących z mikrokontrolera. Część odpowiedzialna za zasilanie składa się z zasilacza napięcia stałego o wartości 6 *V*, diody prostowniczej, stabilizatora napięcia oraz kondensatorów. Mikrokontroler Atmega16 wykorzystuje dwanaście portów cyfrowych (od PD0 do PD5 oraz od PA0 do PA5) do zadawania sygnałów sterujących na cewki elektrozaworów. Mikrokontroler ten posiada również możliwość odczytywania wszystkich sygnałów za pośrednictwem dwóch przycisków sterujących S1 oraz S2. Przycisk S1 jest podłączony do portu RESET i odpowiada za awaryjne przerywanie programu w wypadku ewentualnego, błędnego działania robota. Drugi z przycisków – S2 (oznaczony jako START), jest podłączony do portu PC0, a jego zadanie sprowadza się do inicjalizacji programu w chwili jego wciśnięcia.

Część programowa proponowanego rozwiązania (rys. 2) jest odpowiedzialna za syntezę generatora fizycznego z generatorem programowym oraz za wystawianie odpowiednich sygnałów sterujących na wejścia elektrozaworów.

```
void prawa(unsigned char t)
{
_delay_ms(t);
PORTA = 0x02;
_delay_ms(t);
PORTA = 0x08;
_delay_ms(t);
PORTA = 0x20;
_delay_ms(t);
PORTA = 0x15;
_delay_ms(50);
PORTA = 0x00;
}
```

Rys. 2. Fragment kodu programu odpowiedzialnego za sterowanie prawym elektrozaworem ramienia robota [5]

Fragment kodu źródłowego przedstawionego na rys. 2 pozwala na wyprowadzanie uderzeń prawym ramieniem robota boksera. Przedstawione oprogramowanie zostało opracowane przy pomocy funkcji *void*. Zastosowane w programie opóźnienia czasowe (*_delay_ms*) odpowiadają za właściwą kolejność zadziałania siłowników w prawym ramieniu robota. Odpowiedni dobór czasu zadziałania poszczególnych siłowników w tym przypadku jest szczególnie ważny, gdyż umożliwia on uzyskiwanie płynnych i naturalnych uderzeń wyprowadzanych kolejno przez robota.

Opisany problem syntezy obydwu generatorów należy rozpatrywać w dwóch kategoriach. Pierwsza z nich jest odpowiedzialna za wykonanie generatora fizycznego i programowego, a dopiero druga stanowi o ich syntezie. Uwzględniając z kolei dodatkowo, że generatorem fizycznym można nazwać rzut monetą, grę w ruletkę, czy wybór oznaczonego losu z urny, należy zadać pytanie, w jaki sposób zaimplementować odpowiedni typ generatora w opracowanym programie komputerowym (rys. 2).

W celu właściwego rozwiązania powyższego zadania wykorzystano dużą szybkość działania mikrokontrolera Atmega16, który po podłączeniu do źródła zasilania wykonuje program wynikający z zaproponowanego algorytmu. Algorytm ten przypisuje kolejnej chwili czasu wartości z następującego zbioru liczb {0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15}. Przy czym proces przypisywania kolejnych chwil powtarza się cyklicznie, aż do momentu, w którym użytkownik zatrzyma go, poprzez wciśnięcie przycisku sterującego S2 (oznaczającego START). Idea generatora fizycznego polega więc na tym, iż chwila, w której zostanie naciśnięty przycisk START jest całkowicie losowa. Dzięki takiemu rozwiązaniu otrzymuje się prosty i unikalny generator liczb losowych

wykorzystujący mikrokontroler Atmega16. Z tego względu dalsze prace w tym zakresie mogą skupiać się wyłącznie na wykonaniu komputerowego generatora liczb quasi-losowych. Tego rodzaju opis został szczegółowo przedstawiony w pracy [3], gdzie została zastosowana historycznie najstarsza metoda generacji liczb losowych [1]. Polegała ona na opracowaniu kilkunastu tablic spośród których, za pomocą generatora fizycznego, jest losowana jedna z nich. Takim sposobem została dokonana synteza obu wspomnianych generatorów. Z kolei algorytm syntezy obydwu generatorów jest następujący: po podłączeniu zasilania do zmiennej losowej dodawana jest stała wartość jednostkową a następnie jej wynik jest dzielony modularnie przez 16 (takim sposobem podczas każdego cyklu mikrokontrolera, w kolejnej chwili czasu jest przypisywana inna wartość liczbowa ze zbioru {0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15} i w ten sposób powstaje fizyczny generator liczb losowych). Powyższy stan cyklicznie powtarza się, aż do chwili, w której użytkownik nie wciśnie przycisku S2 (oznaczającego START). Z kolei w chwili wciśnięcia przycisku START, na podstawie wartości przypisanej odpowiedniej chwili (w której został naciśnięty przycisk) jest wybierana jedna z szesnastu tablic zawierających ciąg liczb quasi-losowych. W taki sposób zostaje wyeliminowany okres ciągu liczb quasi-losowych, niemniej jednak pojawia się problem powtarzalności „segmentów" tych liczb w generowanych ciągach. Do rozwiązania tego problemu możliwe jest zastosowanie dwóch metod. Pierwsza z nich bazuje na zwiększaniu liczby *M* podczas dzielenia modularnego, w taki sposób, aby móc zwiększyć zakres przedziału (od 0 do *M*-1), z którego jest wybierana wartość będąca następnie przypisaną do danej chwili czasu – co spowoduje zmniejszenie prawdopodobieństwa wystąpienia jednakowych „segmentów" kilka razy z rzędu. Z kolei prawdopodobieństwo powtórzenia się „segmentów" liczb quasi-losowych można wyznaczyć ze wzoru:

$$P(A) = \frac{1}{M^n} \tag{4}$$

gdzie: *A* – zdarzenie polegające na powtórzeniu się togo samego „segmentu" liczb quasi-losowych *n*-razy; *M* – wartość, przez którą dzieli się modularnie; *n* – parametr określający liczbę powtórzeń tego samego „segmentu"

Ze wzoru (4) wynika, iż prawdopodobieństwo powtórzenia się dwóch „segmentów" dla *M* = 10 jest rzędu 1%. Zatem rozwiązanie polegające na zwiększaniu liczby *M*, nieuchronnie wiąże się z ręcznym wpisywaniem do kodu tablic z długimi ciągami liczb quasi-losowych. W praktyce stanowi to długi, żmudny i monotonny proces, który dodatkowo zwiększa zużycie pamięci mikrokontrolera. Z tego względu lepszym rozwiązaniem wydaje się zastąpienie prymitywnych tablic (wszystkich bądź tylko pewnej ich części) prostymi generatorami liczb quasi-losowych, przykładowo: linowym, afinicznym, czy nawet reszt potęgowych. Dzięki temu zmniejsza się zużycie pamięci mikrokontrolera oraz ograniczana jest monotonia związana z wprowadzaniem do kodu programu długich ciągów liczb quasi-losowych.

4. WNIOSKI KOŃCOWE

Ze względu na konieczność wprowadzania do licznych układów fizycznych szeregu elementów testujących oraz korekcyjnych fizyczne generatory liczb losowych są w praktyce wyjątkowo rzadko stosowane. Z kolei stosowane powszechnie generatory komputerowe (zwane również programowymi), które takich elementów nie wymagają, generują liczb typu quasi-losowego. Zaproponowane rozwiązanie, polegające na syntezie obydwu tych generatorów nie wymaga koniecznego stosowania elementów testujących bądź korekcyjnych, a uzyskane dzięki temu połączeniu ciągi liczbowe, można nazywać losowymi. Należy zatem uznać, że proponowane rozwiązanie pozwala na zwiększenie efektywności wykorzystania generatorów fizycznych w czasie generowania ciągów liczb losowych, w wielu zastosowaniach praktycznych.

5. BIBLIOGRAFIA

1. Kotulski W.: Generatory liczb losowych: algorytmy, testowanie, zastosowania, Matematyka Stosowana: matematyka dla społeczeństwa Nr 2 (43), Warszawa 2001 s. 32-66, ISSN: 1730-2668.
2. Wieczorkowski R., Zieliński R.: Komputerowe generatory liczb losowych, Warszawa 1997, ISBN: 83-204-2160-8.
3. Niederreiter H.: Random Number Generation and Quasi-Monte Carlo Methods, Philadelphia 1992, ISBN: 0-89871-295-5.
4. L'Ecuyer P.: Random Number Generation, Handbook of Simulation, rozdział 4, New York 1998, ISBN: 0-471-13403-1.
5. Ambroziak D., Beskow M.: Robot bokser, Gdańsk 2011, praca dyplomowa magisterska (opiekun pracy: M. Dąbkowski).

RANDOM NUMBER GENERATOR LEVERAGE A MICROCONTROLLER ATMEGA16

Key-words: random number generator, microcontroller Atmega16, randomness.

The purpose of this publication is to provide a solution to the problem of synthesis of physical and computer random number generators based on a simple integrated circuit leveraging Atmega16 microcontroller. Article in the introduction presents the origins, application and types of random number generators. Later in this paper are presented examples of pseudorandom numbers generators. Another part of this publication is accurate to look at the problem and its solution method. Article ends with a summary presenting the final conclusions of this work.