Perbandingan Kecepatan Algoritma *Pseudo Random Number Generator* yang Diimplementasi pada Arduino

Luthfi Fadillah
13515072
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
luthfi.fadillah@outlook.com

Banyak Algoritma PRNG yang dikembangkan, namun pengujian dilakukan pada lingkungan PC, dan jarang dilakukan pada lingkungan yang memiliki performa rendah seperti Arduino. Algoritma yang diujikan adalah LCG, BBS, MT, dan MWC. Pengujian dilakukan untuk mengukur waktu dan memori yang digunakan. Hasil yang didapat adalah semua algoritma memiliki satuan waktu microsecond, sehingga performa yang didapatkan cukup baik

Kata Kunci—PRNG, Arduino, LCG, BBS, MT, MWC

I. PENDAHULUAN

Seiring berjalannya waktu, perkembangan algoritma pembangkit bilangan acak (*random number generator*) sudah semakin pesat. Banyak algoritma-algoritma baru yang dikembangkan dari algoritma yang sudah ada sebelumnya. Beberapa contoh algoritma versi lama yang menjadi dasar untuk perkembangan algoritma selanjutnya adalah *Linear Congruential Generator* (LCG) dan *Blum Blum Shub* (BBS. Beberapa contoh pengembangan dari algoritma yang sudah ada adalah *Mersenne Twister* (MT) dan *Multiply With Carry* (MWC).

Pada umumnya, algoritma-algoritma tersebut dikembangkan pada lingkungan pengembangan berupa *Personal Computer* (PC) atau pada *server* yang memiliki performa yang tinggi, seperti *clockspeed* CPU yang tinggi dan jumlah RAM yang banyak. Sangat jarang dilakukan pengembangan yang menggunakan lingkungan pengembangan dengan performa yang rendah, seperti *microcontroller*.

Arduino adalah *microcontroller* yang digunakan untuk mengembangkan berbagai prototipe sistem perangkat keras, seperti sistem otomasi, robotika, dan sebagainya. Performa yang dimiliki *arduino* terbilang rendah, karena memang penggunaanya ditujukan untuk membuat prototipe sistem perangkat keras. Selain itu, ukuran fisik arduino yang kecil, sehingga tidak mendukung penggunaan komponen yang memiliki performa tinggi.

Melihat penggunaan *arduino* yang fleksibel, maka *arduino* juga dapat digunakan untuk hal-hal yang menggunakan algoritma *random number generator*, seperti membangkitkan bilangan untuk kata sandi (*password*). Namun, karena lingkungan pengembangan algoritma-algoritma RNG yang biasanya berada di PC atau server dan bukan di perangkat-perangkat yang kecil seperti *microcontroller*, maka diperlukan pengujian algoritma-algoritma RNG dan menentukan algoritma mana yang terbaik yang digunakan pada lingkungan *microcontroller*.

II. LANDASAN TEORI

A. Random Number Generator (RNG)

RNG adalah sistem yang dapat membangkitkan bilangan acak, yaitu bilangan yang sulit diprediksi [1]. RNG sendiri dapat dibagi menjadi 2 berdasarkan tingkat kesulitan untuk memprediksi bilangan yang dibangkitkan, yaitu *True Random Number Generator* (TRNG) dan *Pseudo Random Number Generator* (PRNG). TRNG adalah RNG yang hampir mustahil diprediksi oleh manusia, sedangkan PRNG adalah RNG yang masih dapat diprediksi oleh manusia, meskipun membutuhkan sumber daya yang besar. Sistem TRNG dapat ditemui pada lingkungan sekitar, seperti tingkat radiasi kosmik, *atmospheric noise*, *thermal noise*, dan sebagainya. Sistem PRNG terbentuk dari rumus-rumus matematika untuk membangkitkan bilangan acak, seperti algoritma LCG dan BBS. Untuk perbandingan ini, digunakan 4 macam algoritma, yaitu LCG, BBS, MT, dan MWC.

1) Linear Congruential Generator (LCG)

LCG adalah salah satu algoritma PRNG tertua, dimana algoritma LNG menerapkan persamaan *piecewise linear* yang tidak kontinu [2]. Persamaan algoritma LCG dapat didefinisikan sebagai:

$$X_{n+1} = (aX_n + c)\ mod\ m \qquad (1)$$

Dimana X adalah sekuens nilai acak, m adalah modulus yang memenuhi $0 < m$, a adalah faktor pengali yang memenuhi $0 < a < m$, c adalah nilai *increment* yang memenuhi $0 \leq c < m$, dan X_0 adalah nilai awal atau *seed* yang memenuhi $0 \leq X_0 < m$ [3]. Keunggulan LCG adalah mudah untuk dimengerti dan diimplementasikan, serta cepat untuk dijalankan karena hanya membutuhkan sedikit operasi bit [2]. Namun, LCG juga mudah untuk diprediksi nilai yang akan keluar, sehingga tidak cocok digunakan untuk kriptografi [2].

2) Blum Blum Shub (BBS)

BBS adalah algoritma yang berbasis teori bilangan [2]. Algoritma BBS diciptakan oleh Lenore Blum, Manuel Blum dan Michael Shub. Algoritma BBS dapat didefinisikan sebagai berikut [2]:

- Pilih dua bilangan bulat p dan q, yang masing-masing kongruen dengan *3 (mod 4)*.
- Menghitung $pq = n$. Nilai n adalah bilangan bulat Blum.
- Pilih bilangan bulat acak s sebagai *seed* yang memenuhi syarat: (1) $2 \leq s < n$, dan (2) s dan n relatif prima
- Melakukan iterasi berikut
    - Menghitung $x_i = x_{i-1}^2 \ mod \ n$
    - Menghitung barisan acak z, dimana z_i = Least Significant Bit (LSB) dari x_i.

Keunggulan dari BBS adalah algoritmanya yang sederhana dan termasuk algoritma *Cryptographically Secure Pseudo Random Generator* (CSPRNG), yaitu algoritma PRNG yang teruji keamanannya dan dapat digunakan untuk kriptografi [2].

3) Mersenne Twister (MT)

MT adalah algoritma yang banyak diaplikasikan sebagai general-purposed PRNG [5]. Algoritma MT dikembangkan pada tahun 1997 oleh Makoto Matsumoto dan Takuji Nishimura [7]. Nama algoritma MT didasarkan dari panjang periode yang digunakan dipilih dari bilangan prima *Mersenne*, yaitu bilangan prima yang dihasilkan dari dari persamaan:

$$M_n = 2^n - 1 \quad (2)$$

Dimana n adalah bilangan bulat positif [6]. Berikut adalah *pseudocode* dari algoritma MT

```
function extract_number(){
  if index >= n {
    if index > n {
      error "Generator was never seeded"
    }
    twist()
  }
  int y := MT[index]
  y := y xor ((y >> u) and d)
  y := y xor ((y << s) and b)
  y := y xor ((y << t) and c)
  y := y xor (y >> l)
  index := index + 1
  return lowest w bits of (y)
}

function twist(){
  for i from 0 to (n-1) {
    int x := (MT[i] and upper_mask)
    + (MT[(i+1) mod n] and lower_mask)
    int xA := x >> 1
    if (x mod 2) != 0 {
      xA := xA xor a
    }
    MT[i] := MT[(i + m) mod n] xor xA
  }
  index := 0
}
```

Keunggulan dari algoritma MT adalah sudah teruji dari beberapa tes keacakan secara statistik serta memiliki periode yang panjang (2^{19937}-*1*). Kelemahan dari algoritma MT adalah memakan memori yang banyak serta tidak dapat digunakan untuk kriptografi (kecuali varian CryptMT)

4) Multiply With Carry (MWC)

MWC adalah algoritma yang diciptakan oleh George Marsaglia, dimana algoritma MWC adalah pengembangan dari algoritma LCG. Perbedaan algoritma MWC dan LCG terletak pada nilai c, dimana nilai c pada algoritma MWC memiliki nilai yang bervariasi, berbeda dengan LCG yang memiliki nilai yang tetap [8]. Algoritma MWC dapat didefinisikan sebagai berikut:

$$X_{n+1} = (aX_{n-r} + c_{n-1}) \ mod \ m \quad (3)$$

Nilai c_n yang bervariasi didefinisikan sebagai berikut:

$$c_n = \left\lfloor \frac{ax_{n-r}+c_{n-1}}{b} \right\rfloor \quad (4)$$

Dimana a, x, dan c memiliki definisi yang sama dengan LCG, dan b adalah modulus.

B. Arduino

Arduino adalah *microcontroller* yang bersifat *open-source* dan digunakan untuk membuat prototipe sistem digital skala kecil dan menengah [9]. *Microcontroller* sendiri adalah komputer kecil yang berada pada *integrated circuit* (IC) tunggal. Bahasa pemrograman yang digunakan pada Arduino adalah C++ dengan beberapa perubahan, seperti harus mendeklarasikan fungsi *start* dan *loop* untuk setiap program yang dibuat.

Arduino yang paling banyak diproduksi, dan yang akan digunakan untuk melakukan pengujian kecepatan algoritma PRNG, adalah jenis *Arduino Uno*. Tabel berikut menunjukkan spesifikasi dari *Arduino Uno.*

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Arduino Uno memiliki *flash memory* sebesar 32 KB yang digunakan untuk menyimpan kode program, serta memiliki SRAM sebesar 2 KB yang digunakan untuk menyimpan *global variable*. Selain itu, kecepatan CPU yang dimiliki Arduino Uno adalah 16 Mhz.

III. Rancangan Pengujian

A. Kode Program

Kode program antar algoritma yang akan diuji dibuat terpisah. Masing-masing memiliki parameternya tersendiri yang diatur berdasarkan pengujian.

1) LCG

LCG memiliki 4 parameter, yaitu *mod_val*, *a_val*, *c_val*, dan *seed_val*. Parameter *mod_val* adalah nilai modulus dengan nilai 2^32, parameter *a_val* adalan nilai *a* dengan

nilai 1664525, parameter *c_val* adalah nilai *c* dengan nilai 1013904223, dan parameter *seed_val* adalah nilai *seed* dengan nilai 1234567890. Berikut adalah implementasi algoritma LCG.

```
unsigned long lcg(unsigned long modulus,
unsigned long a, unsigned long c, unsigned
long seed){
  return (a * seed + c) % modulus;
}
```

2) BBS

BBS memiliki 3 parameter yaitu *p_val, q_val*, dan *s_val*. Parameter *p_val* adalah nilai *p* dengan nilai 8756687, parameter *q_val* adalah nilai *q* dengan nilai 1548514, dan parameter *s_val* adalah nilai *seed* dengan nilai 1234567890. Berikut adalah implementasi algoritma BBS.

```
class bbs{
  unsigned long p, q, M, seed, actual;

  unsigned long gcd(unsigned long a,
unsigned long b){
    if(b == 0) return a;
      return gcd(b, fmod(a,b));
  }

  public:

  bbs(unsigned long p, unsigned long q,
unsigned long s){
    this->p = p;
    this->q = q;
    this->seed = s;
    M = p*q;
    actual = s;
  }

  unsigned long getrandom(){
    unsigned long r = fmod(actual*actual,M);
    actual = r;
    return r;
  }

  unsigned long getirandom(int i){

    unsigned long g = gcd(p, q);
    unsigned long lcm = p*q/g;
    unsigned long exp = 1;

    for(int j = 1; j <= i; ++j){
      exp = fmod((exp+exp),lcm);
    }

    unsigned long x0 = seed*seed;
    unsigned long r = x0;

    for(int j = 2; j <= exp; ++j){
      r = fmod((r*x0),M);
    }
    return r;
  }
};
```

3) Mersenne Twister

Algoritma MT memiliki banyak parameter, tapi yang perlu diperhatikan adalah nilai seed dan panjang *array* MT, sedangkan nilai parameter lainnya adalah standar dari MT19937. Nilai seed diisi dengan 1234567890, dan nilai panjang array MT diisi dengan 312, atau setengan dari panjang array standar MT19937. Panjang array MT dijadikan setengan dikarenakan panjang array standar MT19937 (624) membutuhkan memori yang lebih banyak dibandingkan yang disediakan oleh SRAM, sehingga program tidak dapat dijalankan. Berikut implementasi algoritma MT.

```
void initialize(unsigned long seed){
  MT[0] = seed;
  for(int i=1;i<312;i++){
    MT[i] = ((1812433253 * MT[i-1]) ^
((MT[i-1] >> 30) + i)) & bitmask_1;
  }
}

void generate_numbers(){
  for(int i = 0;i<312;i++){
    unsigned long y = (MT[i] & bitmask_2) +
(MT[(i + 1 ) % 312] & bitmask_3);
    MT[i] = MT[(i + 397) % 312] ^ (y >> 1);
    if( y % 2 != 0){
      MT[i] = MT[i] ^ 2567483615;
    }
  }
}

unsigned long extract(){
  if (index==0){
    generate_numbers();
  }
  unsigned long y = MT[index];
  y = y ^ (y >> 11);
  y = y ^ ((y << 7) & 2636928640);
  y = y ^ ((y << 15) & 4022730752);
  y = y ^ (y >> 18);

  index = (index + 1) % 312;
  return y;
}
```

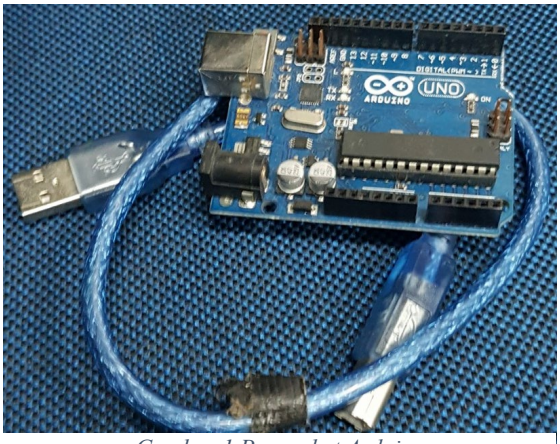
4) MWC

Algoritma MWC memiliki 2 parameter, yaitu *m_w* dan *m_z*. parameter m_z diisi dengan nilai 1 dan parameter m_w diisi dengan nilai 2. Berikut adalah implementasi algoritma MWC

```
unsigned long getRandom()
{
    m_z = 36969L * (m_z & 65535L) + (m_z >>
16);
    m_w = 18000L * (m_w & 65535L) + (m_w >>
16);
    return (m_z << 16) + m_w;
}
```

B. Arduino

Perangkat *Arduino* yang digunakan adalah *Arduino Uno* beserta kabel untuk menghubungkan *Arduino* ke laptop. Pengujian dilakukan tanpa menambahkan *peripheral* apapun. Berikut adalah perangkat yang digunakan untuk pengujian.



Gambar 1 Perangkat Arduino

C. *Pengukuran Pengujian*

Hal yang diukur pada pengujian adalah waktu dan memori yang digunakan. Untuk mengukur waktu, satuan waktu yang digunakan adalah *microsecond* dan dihitung dengan cara mengurangi waktu awal tepat sebelum fungsi dipanggil dengan waktu akhir ketika fungsi selesai dipanggil. Pengujian dilakukan dengan dua kondisi, yaitu saat fungsi dipanggil sekali dan saat fungsi dipanggil 10 kali. Waktu akan ditampilkan pada *serial monitor* yang ada pada aplikasi Arduino IDE dengan *baud rate* 9600. Berikut adalah kode program pengujian

```
void setup() {
  Serial.begin(9600);
  unsigned long time_start = micros();
  for (int i = 0; i < 10; i++){
    // Fungsi PRNG
  }
  unsigned long time_end = micros();
  Serial.print("Time : ");
  Serial.println(time_end-time_start);
}
```

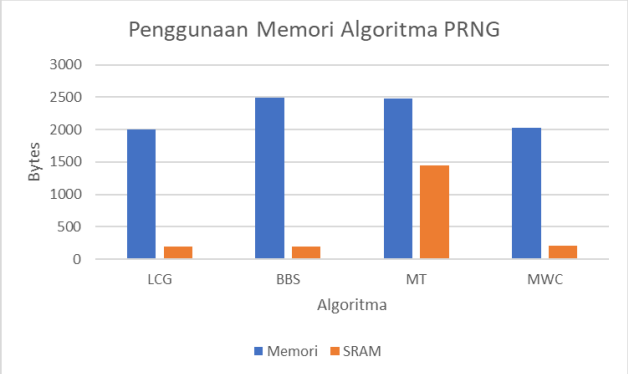
Untuk mengukur memori, satuan yang digunakan adalah *bytes*, dan dihitung ketika program dikompilasi. Memori yang diukur adalah penggunaan *flash memory* (yang kemudian disebut sebagai memori) dan penggunaan SRAM (yang kemudian disebut sebagai RAM). Penggunaan memori bergantung pada panjang kode program yang dibuat. Penggunaan RAM bergantung pada seberapa banyak global variable yang digunakan.

IV. PENGUJIAN DAN ANALISIS

A. *Pengujian Memori*

Berikut adalah hasil pengujian memori untuk setiap algoritma.

Algoritma	Memori (bytes)	RAM (bytes)
LCG	2008	200
BBS	2486	196
MT	2480	1446
MWC	2030	204



Dari tabel dan grafik tersebut, didapat bahwa penggunaan memori hampir sama untuk setiap algoritma, dimana BBS dan MT menggunakan memori lebih banyak dibandingkan LCG dan MWC dikarenakan kode program yang dibuat memiliki baris yang lebih banyak. Untuk penggunaan RAM, algoritma MT menggunakan RAM jauh lebih besar dibandingkan yang lain karena kebutuhan RAM untuk *array* MT yang berukuran 312 dan berjenis *unsigned long*.

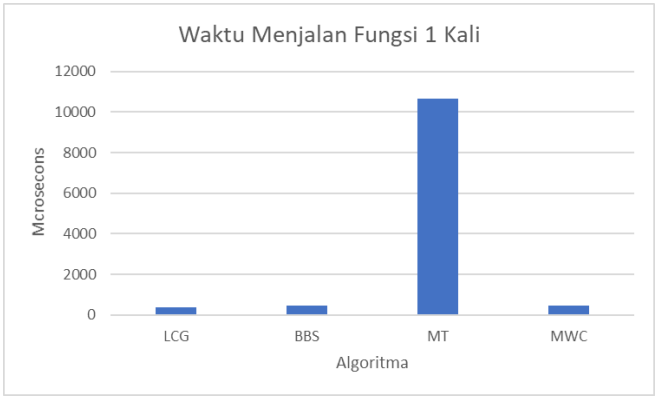
B. *Pengujian Waktu*

Pengujian waktu dibagi menjadi 2 keadaan, yaitu waktu untuk menjalankan fungsi 1 kali dan waktu untuk menjalankan fungsi 10 kali.

1) *Menjalankan Fungsi 1 Kali*

Berikut adalah hasil pengujian waktu untuk menjalankan fungsi 1 kali.

Algoritma	Waktu (microseconds)
LCG	396
BBS	444
MT	10644
MWC	444

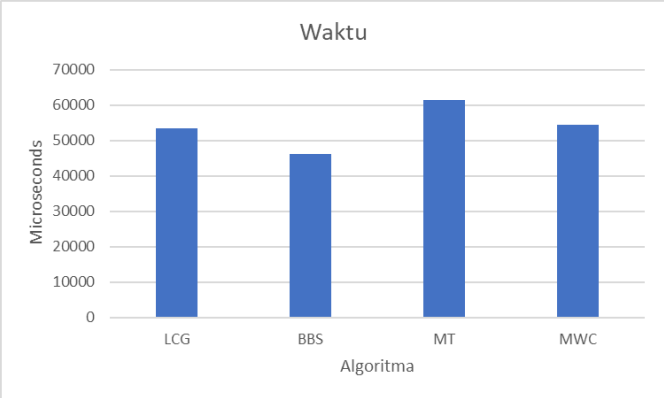


Dari tabel dan grafik tersebut, didapat bahwa algoritma MT membutuhkan waktu yang lebih banyak untuk menjalankan fungsi 1 kali dibandingkan algoritma lainnya. Hal ini dikarenakan algoritma MT yang cukup rumit, ditambah RAM yang tersisa hanya 30% dari total RAM yang dimiliki sehingga proses berjalan lama.

2) *Menjalankan Fungsi 10 Kali*

Berikut adalah hasil pengujian waktu untuk menjalankan fungsi 10 kali

Algoritma	Waktu
LCG	53388
BBS	46160
MT	61540
MWC	54468



Dari tabel dan grafik tersebut, didapat bahwa waktu yang dibutuhkan untuk menjalankan algoritma sebanyak 10 kali hampir sama. Algoritma MT membutuhkan waktu yang lebih banyak dikarenakan algoritmanya yang rumit.

V. KESIMPULAN DAN SARAN

Kesimpulan dari hasil pengujian waktu dan memori adalah hampir semuanya memiliki waktu yang cukup rendah (dalam satuan microsecond), meskipun tidak serendah pengujian pada PC.

Saran dari pengujian ini adalah menggunakan *seed* berupa waktu saat ini,sehingga hasil bilangan acak akan terlihat tanpa memunculkan pola

SAMBUTAN

Puji dan Syukur dipanjatkan kehadiran Allah S.W.T yang telah memberikan rahmat-Nya sehingga saya dapat menyelesaikan makalah ini. Terima kasih kepada kedua orang tua saya yang selalu mendukung saya dalam hal-hal positif, termasuk membuat makalah ini, Terima kasih juga kepada Bapak Rinaldi Munir yang telah membantu saya untuk memahami mata kuliah Kriptografu sehingga saya dapat menuangkan ilmu yang saya dapat melalui makalah ini..

DAFTAR REFERENSI

[1] Munir, R. *Pembangkit Bilangan Acak*. Slide Kuliah IF4020 Kriptografi. 2019.

[2] *Linear Congruential Generators – Wolfram Demonstration Project*. http://demonstrations.wolfram.com/LinearCongruentialGenerators/ diakses tanggal 10 Mei 2019.

[3] Knuth, Donald E. *Art of Computer Programming, Volumes 1-4A Boxed Set*. Addison-Wesley Professional, 2011.

[4] Blum, L., Blum, M., dan Shub, M., *A simple unpredictable pseudo-random number generator*. SIAM Journal on computing 15.2 (1986): 364-383.

[5] Marsland, S. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011.

[6] *GIMPS Discovers, Largest Known Prime Number: $2^{82,589,933}$-1*. https://www.mersenne.org/primes/press/M82589933.html diakses tanggal 10 Mei 2019.

[7] Matsumoto, M. and Nishimura, T., *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*. ACM Transactions on Modeling and Computer Simulation (TOMACS), 8(1), pp.3-30, 1998.

[8] Marsaglia, G. and Zaman, A., *A new class of random number generators*. The Annals of Applied Probability, pp.462-480, 1991.

[9] *Arduino*. https://www.arduino.cc/ diakses tanggal 10 Mei 2019.