

ĐẠI HỌC QUỐC GIA HÀ NỘI – TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

---



**BÁO CÁO MÔN HỌC**

Môn: Phát triển hệ thống nhúng bảo mật

Đề tài:

Xây dựng bộ tạo số ngẫu nhiên

Giảng viên hướng dẫn: PGS.TS Trần Xuân Tú  
TS. Bùi Duy Hiếu

|                      |                 |          |
|----------------------|-----------------|----------|
| Sinh viên thực hiện: | Lê Toàn Thắng   | 21021441 |
|                      | Phạm Thái Dương | 21021407 |
|                      | Phạm Minh Quang | 21020601 |

Hà Nội, ngày 30 tháng 12 năm 2024

## LỜI CẢM ƠN

Lời đầu tiên cho chúng em xin phép được gửi lời cảm ơn chân thành nhất đến PGS.TS Trần Xuân Tú và Tiến sĩ Bùi Duy Hiếu vì đã cho chúng em có thêm kiến thức mới quý giá trong quá trình học cũng như quá trình làm bài tập lớn. Đồng thời chúng em cũng xin cảm ơn sự giúp đỡ của các anh trong lab của thầy Hiếu, đặc biệt là anh Long và anh Doanh đã tận tình giúp đỡ chúng em trong suốt quá trình.

Tuy nhiên, do thời gian có hạn và kiến thức còn nhiều thiếu sót nên chúng em không thể hoàn thành trọn vẹn được bài tập lớn, chúng em mong được thầy và các anh thông cảm và tạo điều kiện ạ.

## MỤC LỤC

|  |    |
|--|----|
| LỜI CẢM ƠN .....   | 2  |
| A. ĐẶT VẤN ĐỀ, GIỚI THIỆU PHƯƠNG PHÁP .....                                | 5  |
| I. Các Môi Đe Dọa An Ninh Mạng .....                                       | 5  |
| II. Các Loại Môi Đe Dọa An Ninh Mạng .....                                 | 5  |
| III. Phương pháp đề xuất.....  | 5  |
| 1. Sử dụng bộ tạo số ngẫu nhiên (True Random Number Generator) .....       | 5  |
| 2. Định nghĩa về TRNG.....   | 5  |
| 3. Các tiêu chuẩn dành cho TRNG .....                                      | 6  |
| 4. Ứng dụng của TRNG.....  | 6  |
| B. THIẾT KẾ MẠCH TẠO SỐ NGẪU NHIÊN DỰA TRÊN NHIỀU AVALANCHE .....          | 7  |
| I. Sơ đồ khối của mạch điện.....   | 7  |
| II. Nguyên lý hoạt động của các khối .....                                 | 7  |
| III. Các linh kiện cần chuẩn bị .....                                      | 7  |
| IV. Thiết kế mạch nguyên lý trên phần mềm Altium.....                      | 8  |
| 1. Mạch đề xuất.....   | 8  |
| 2. Mạch thiết kế lại .....   | 8  |
| 3. Sản phẩm sau khi thiết kế .....   | 9  |
| 4. Chạy demo Arduino với mạch tạo nhiễu: .....                             | 9  |
| V. Cơ chế hoạt động của các khối trong mạch.....                           | 10 |
| VI. Bảng linh kiện được sử dụng .....                                      | 12 |
| C. THỰC HIỆN TRÊN FPGA.....  | 13 |
| I. Giới thiệu .....  | 13 |
| II. Cấu hình phần cứng để kết nối mạch tạo số ngẫu nhiên cho PYNQ-Z2 ..... | 13 |
| III. Dữ liệu đầu ra.....   | 16 |
| 1. Dữ liệu đầu ra trên Arduino .....                                       | 16 |
| 2. Dữ liệu đầu ra trên PYNQ – Z2 .....                                     | 17 |
| IV. Đánh giá kết quả .....   | 17 |
| 1. Định dạng kết quả: .....  | 17 |
| 2. Giới thiệu NIST TEST .....  | 17 |
| 3. Kết quả NIST TEST.....  | 18 |
| V. Tài liệu tham khảo .....  | 23 |

## DANH SÁCH CÁC HÌNH ĐƯỢC SỬ DỤNG

|  |    |
|--|----|
| Hình 1. Sơ đồ khối mạch tạo nhiễu Avalanche .....      | 7  |
| Hình 2. Sơ đồ mạch đề xuất.....                        | 8  |
| Hình 3. Mạch nguyên lý được thiết kế trên Altium ..... | 8  |
| Hình 4. Mạch sản phẩm .....                            | 9  |
| Hình 5. Kết quả chạy demo trên Arduino UNO R3 .....    | 9  |
| Hình 6. Khối tăng áp 5V – 12V .....                    | 10 |
| Hình 7. Khối tạo nhiễu Avalanche trong mạch .....      | 11 |
| Hình 8. Khối tạo xung clock .....                      | 11 |
| Hình 9. Khối D-Flip Flop.....                          | 12 |
| Hình 10. KIT PYNQ-Z2 EVALUATION .....                  | 13 |
| Hình 11. Cấu hình phần cứng của PYNQ-Z2 .....          | 15 |
| Hình 12. Kết quả chạy demo trên Arduino UNO R3 .....   | 16 |
| Hình 13. Kết quả chạy demo trên PYNQ – Z2 .....        | 17 |
| Hình 14. Bước chạy NIST Test .....                     | 19 |
| Hình 15. Bước chạy NIST Test .....                     | 19 |
| Hình 16. Bước chạy NIST Test .....                     | 20 |
| Hình 17. Bước chạy NIST Test .....                     | 20 |

## DANH SÁCH CÁC BẢNG TRONG BÁO CÁO

|  |    |
|--|----|
| Bảng 1. Các linh kiện.....                   | 13 |
| Bảng 2. Bảng các bài kiểm tra của NIST ..... | 18 |
| Bảng 3. Bảng các bài kiểm tra của NIST ..... | 22 |
| Bảng 4. Bảng các bài kiểm tra của NIST ..... | 23 |

## A. ĐẶT VẤN ĐỀ, GIỚI THIỆU PHƯƠNG PHÁP

### I. Các Mối Đe Dọa An Ninh Mạng

Các mối đe dọa an ninh mạng là những hành động của các cá nhân có ý đồ xấu, nhằm đánh cắp dữ liệu, gây thiệt hại hoặc làm gián đoạn các hệ thống máy tính. Các mối đe dọa này có thể đến từ nhiều nguồn khác nhau, bao gồm các quốc gia thù địch, nhóm khủng bố, hacker cá nhân, hoặc thậm chí những người trong tổ chức lạm dụng quyền truy cập của họ.

### II. Các Loại Mối Đe Dọa An Ninh Mạng

Với sự phát triển và cải tiến không ngừng của công nghệ, sự đe dọa tới an ninh mạng là mối nguy luôn luôn tiềm ẩn với bất kỳ tổ chức hoặc cá nhân nào. Các thủ đoạn tấn công ngày càng trở nên đa dạng và xuất hiện dưới nhiều hình thức khác nhau:

1. Tấn công Social Engineering: Lừa đảo người dùng để cung cấp thông tin nhạy cảm hoặc cài đặt phần mềm độc hại.
2. Tấn công Man-in-the-Middle: Can thiệp vào các cuộc giao tiếp giữa hai bên để nghe lén hoặc giả mạo một trong các bên.
3. Tấn công Injection (Chèn mã độc vào hệ thống): Lợi dụng các lỗ hổng bảo mật trong mã nguồn của ứng dụng web để chèn mã độc vào hệ thống.

### III. Phương pháp đề xuất

1. Sử dụng bộ tạo số ngẫu nhiên (True Random Number Generator)

Với sự phát triển mạnh mẽ của các thiết bị kết nối và các cuộc tấn công ngày càng tinh vi, nhu cầu bảo mật trong các sản phẩm và dữ liệu của các tổ chức trở nên quan trọng hơn bao giờ hết. True Random Number Generators, hay các bộ sinh số ngẫu nhiên thực, là yếu tố cốt lõi trong các hệ thống bảo mật, đóng vai trò quan trọng trong việc tạo ra các khóa mã hóa và bảo vệ dữ liệu nhạy cảm. Những số ngẫu nhiên yếu hoặc có thể dự đoán sẽ tạo điều kiện cho các cuộc tấn công, từ đó làm lộ khóa, đánh cắp dữ liệu và xâm nhập vào thiết bị.

2. Định nghĩa về TRNG

True Random Number là các số ngẫu nhiên được tạo ra từ hiện tượng vật lý không thể dự đoán, gọi là nguồn entropy. Chúng không tuân theo bất kỳ quy luật hay thuật toán nào, đảm bảo tính không thể đoán trước, độc lập thống kê và phân phối đồng đều. TRNs đóng vai trò nền tảng trong bảo mật, được sử dụng để tạo và bảo vệ các thông tin nhạy cảm, đảm bảo tính an toàn cho các hệ thống và thiết bị.

### 3. Các tiêu chuẩn dành cho TRNG

Có một số tiêu chuẩn quốc tế xác định các phương pháp kiểm tra và chứng nhận độ ngẫu nhiên của TRNGs, bao gồm:

- NIST SP 800-90A/B/c: Các tiêu chuẩn này xác định các tiêu chí phân tích thống kê mà một TRNG cần phải đáp ứng để được coi là đủ ngẫu nhiên cho các ứng dụng mã hóa.
- AIS 20/31 của BSI: Tiêu chuẩn của Đức cho RNGs.
- Các chứng nhận như FIPS 140-2/140-3, Common Criteria, và OSCCA của Trung Quốc đảm bảo rằng sản phẩm cuối cùng đáp ứng các yêu cầu bảo mật và có thể được chứng nhận cho việc sử dụng trong các ứng dụng chính thức.

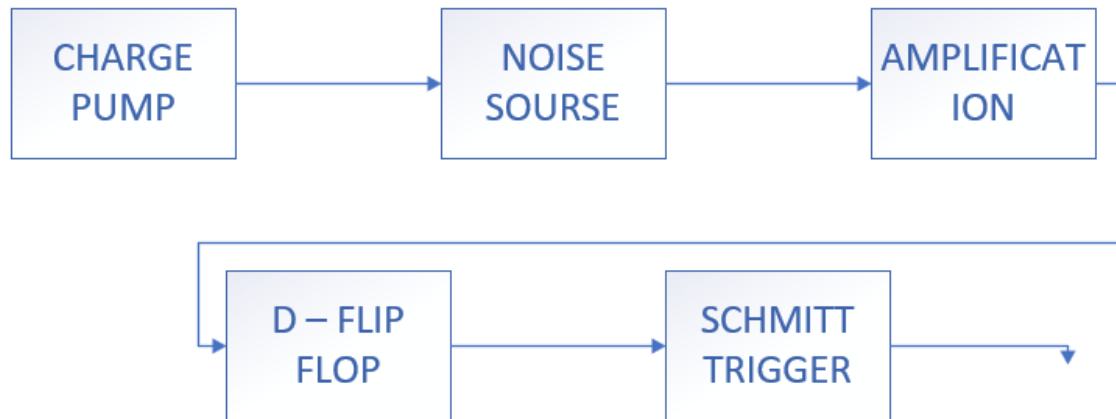
### 4. Ứng dụng của TRNG

True Random Number Generators có nhiều ứng dụng quan trọng trong các hệ thống bảo mật và các lĩnh vực công nghệ khác. Dưới đây là một số ứng dụng phổ biến của TRNG:

- Tạo khóa bảo mật: cho các thuật toán đối xứng, bất đối xứng,...
- Xác thực và chứng thực
- Giá trị khởi tạo: sử dụng trong mã hóa, thuật toán MAC,...

## B. THIẾT KẾ MẠCH TẠO SỐ NGẪU NHIÊN DỰA TRÊN NHIỀU AVALANCHE

### I. Sơ đồ khối của mạch điện



Hình 1. Sơ đồ khối mạch tạo nhiễu Avalanche

### II. Nguyên lý hoạt động của các khối

- Charge pump:** khối này sẽ thực hiện nhiệm vụ tăng điện áp đầu vào từ 5V lên 12V để cung cấp điện áp cho các khối khác trong mạch.
- Noise source:** đây là khối sẽ thực hiện việc tạo nhiễu avalanche để phục vụ quá trình sinh số ngẫu nhiên.
- Amplification:** khi tín hiệu nhiễu ngẫu nhiên ban đầu đi qua tầng này sẽ được khuếch đại đến một ngưỡng nhất định giúp tín hiệu dễ dàng được xử lý.
- Schmitt Trigger:** tín hiệu nhiễu ngẫu nhiên ban đầu sau khi được khuếch đại sẽ được chuyển sang dạng xung ô vuông kỹ thuật số.
- Toggle (T-Flop):** khối này hoạt động như một công tắc chuyển đổi giữa giá trị cao và thấp để chuyển đổi xung sang dạng tín hiệu 1-0 và đưa vào Arduino xử lý.

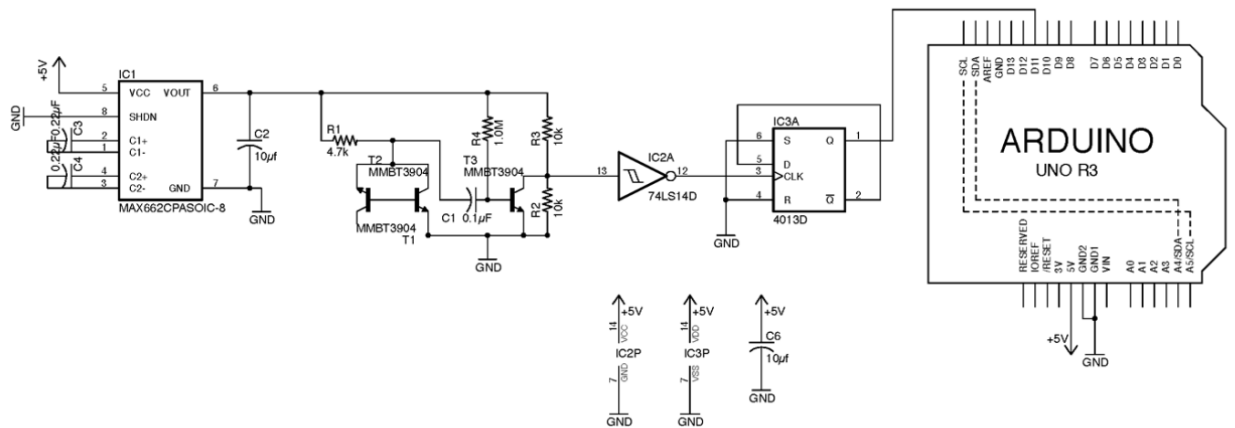
### III. Các linh kiện cần chuẩn bị

| Linh kiện | Miêu tả             | Số lượng |
|-----------|---------------------|----------|
| SN74LS14N | Hex-Schmitt Trigger | 1        |
| CD4013BE  | D - Flip Flop       | 1        |
| 22uF Cap  |                     | 2        |
| 0.1uF Cap |                     | 1        |
| 1uF Cap   |                     | 2        |
| 10uF Cap  |                     | 2        |

|             |                |   |
|-------------|----------------|---|
| 2N3904      | NPN Transistor | 3 |
| 4.7Kohm Res |                | 1 |
| 1M Res      |                | 1 |
| 10K Res     |                | 2 |
| Arduino     |                | 1 |
| Breadboard  |                | 1 |

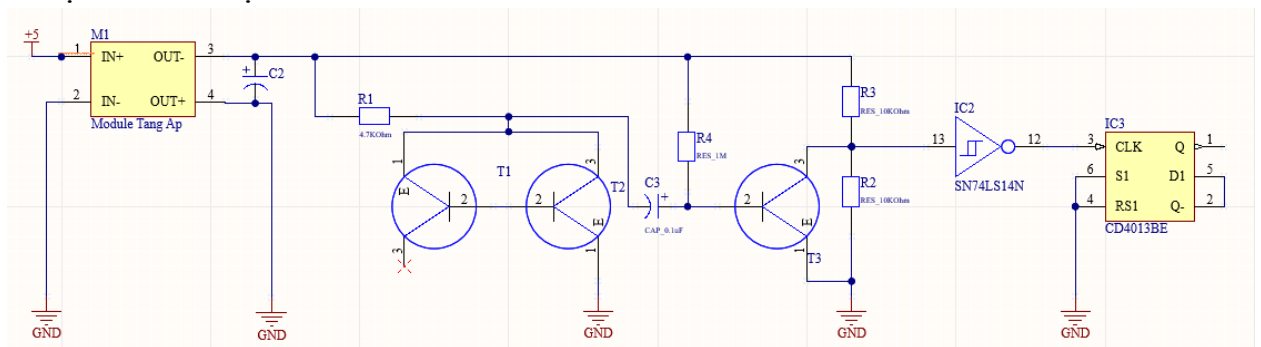
#### IV. Thiết kế mạch nguyên lý trên phần mềm Altium

##### 1. Mạch đề xuất



Hình 2. Sơ đồ mạch đề xuất

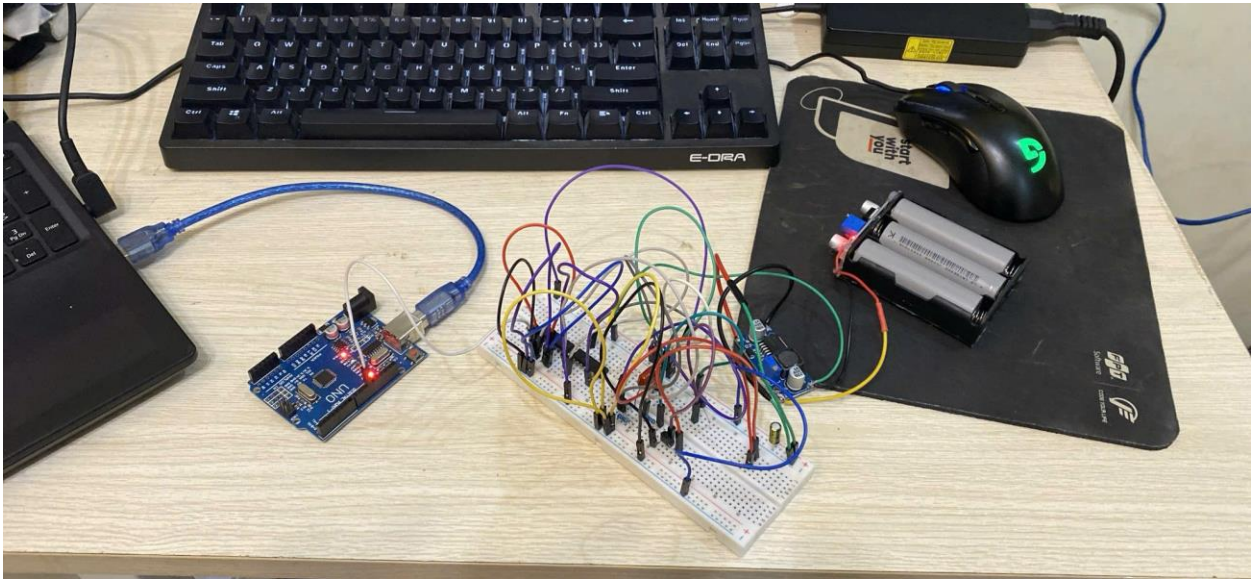
##### 2. Mạch thiết kế lại



Hình 3. Mạch nguyên lý được thiết kế trên Altium



### 3. Sản phẩm sau khi thiết kế



Hình 4. Mạch sản phẩm

### 4. Chạy demo Arduino với mạch tạo nhiễu:

```
1 // Arduino R3 MM4845
2
3 #include <Arduino.h>
4
5 #define GEN_RND_PIN 11 // Digital pin 11 connected to the output of the noise avalanche circuit.
6 #define NUM_BITS 10 // Specifies number of bits to combine.
7
8 unsigned int rnd = 0; // Global placeholder to access changing random number.
9
10 int differRandom() {
11   rnd = 0; // Reset to zero before next conversion.
12   for (int i = 0; i < NUM_BITS; i++) { // Loop to collect and combine bits.
13     rnd = (rnd << 1) | digitalRead(GEN_RND_PIN); // Combine bits using bitwise operations.
14     delay(5); // Small delay to stabilize readings.
15   }
16   return rnd;
17 }
18
19 // Function to convert an integer to a bitstream string.
20 String toBitstream(int num, int numBits) {
21   String bitstream = "";
22   for (int i = numBits - 1; i >= 0; i--) { // Iterate through each bit from MSB to LSB.
23     bitstream += ((num >> i) & 1) ? "1" : "0"; // Extract each bit and append to string.
24   }
25   return bitstream;
26 }
27
28 void setup() {
29   Serial.begin(9600);
30 }
31
32 void loop() {
33   Serial.print(toBitstream(differRandom(), NUM_BITS));
34   Serial.println();
35   delay(1000);
36 }
```

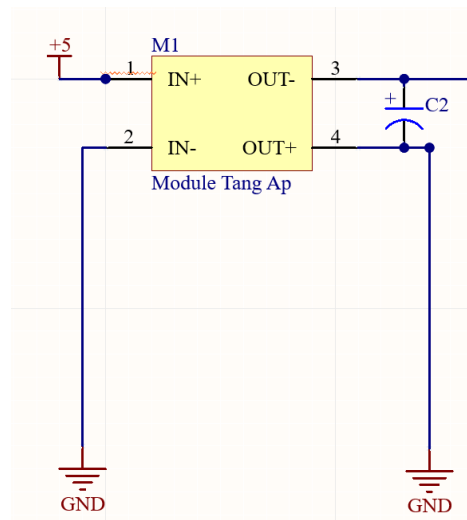
1001100010  
0010011001  
1001100110  
0110011001  
1001100110  
0100011001  
0001000100  
0100011001  
0001000100  
0100010001  
0001000100  
0100110011  
0010001000  
1000100010  
0010001000  
1000100010  
0010001000

Hình 5. Kết quả chạy demo trên Arduino UNO R3

## V. Cơ chế hoạt động của các khối trong mạch

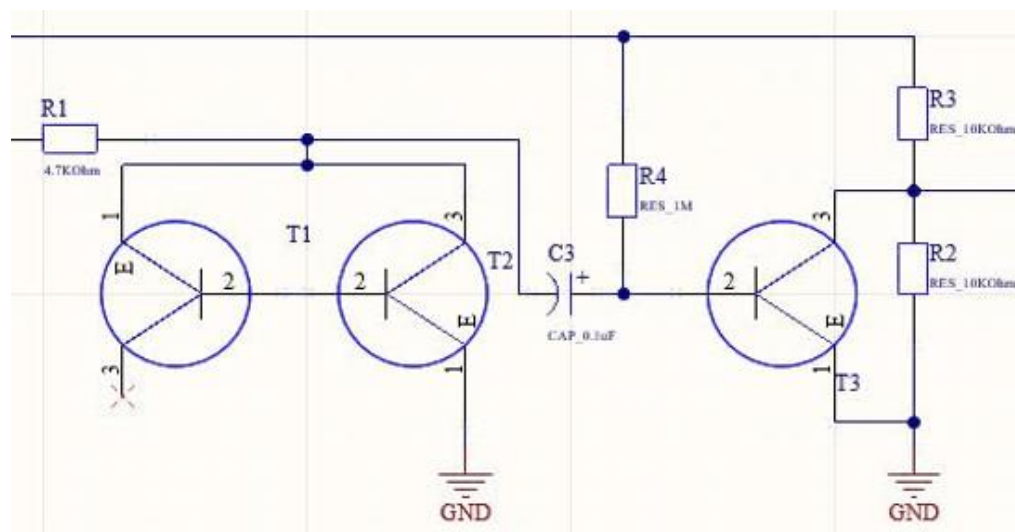
### 1. Khối tăng áp

Module tăng áp-boost converter có chức năng tăng điện áp một chiều ở đầu vào thành điện áp một chiều có giá trị cao hơn ở đầu ra. Tụ điện được đặt ở đầu ra của module có chức năng ổn định điện áp đầu ra ở mức 12V.



Hình 6. Khối tăng áp 5V – 12V

### 2. Khối tạo nhiễu Avalanche



### Hình 7. Khởi tạo nhiễu Avalanche trong mạch

#### a. Giới thiệu về cách hoạt động của mạch

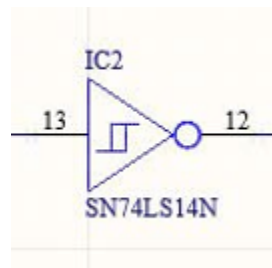
Mạch hoạt động dựa trên nguyên lý nhiễu avalanche tạo bởi transistor T1 và T2. Tín hiệu nhiễu sau khi được tạo ra sẽ được khuếch đại bởi transistor T3.

#### b. Nguyên lý nhiễu avalanche

Khi đặt điện áp ngược vượt quá điện áp lên diode hoặc lớp bán dẫn p-n, dòng điện tăng đột ngột do các điện tử va chạm với nguyên tử, tạo ra thêm cặp điện tử-lỗ trống. Hiện tượng này tạo ra nhiễu avalanche – một loại nhiễu có tính ngẫu nhiên cao.

Trong thực tế, người ta thường sử dụng phân lớp base-emitter của transistor NPN để tạo nhiễu avalanche, vì lớp này có điện áp đánh thủng thấp. Nhiễu này được ứng dụng làm nguồn entropy cho các hệ thống tạo số ngẫu nhiên.

### 3. Khởi tạo xung clock



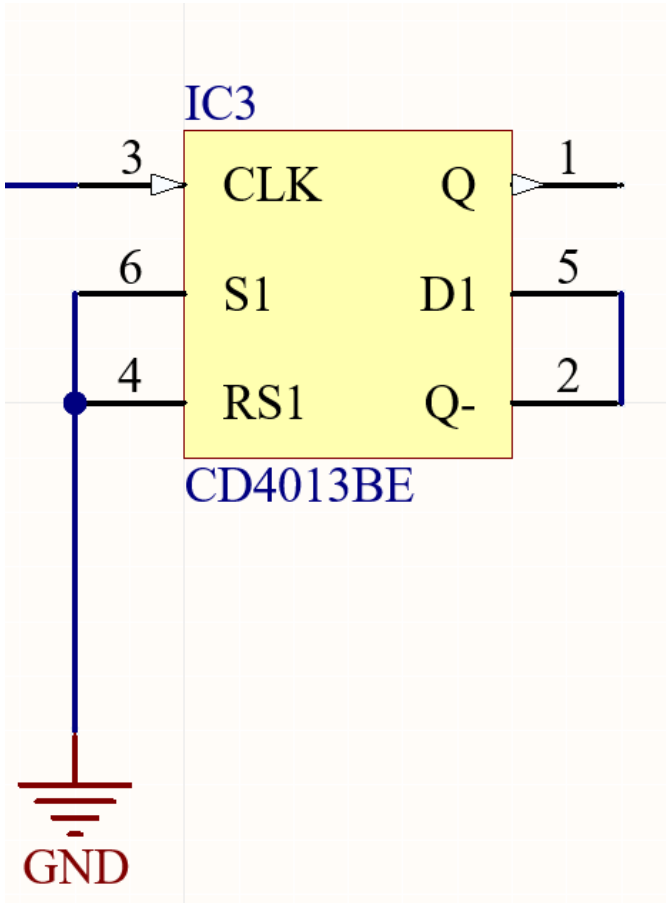
Hình 8. Khởi tạo xung clock

Khởi tạo xung clock sử dụng IC SN74LS14N nhận đầu vào là tín hiệu nhiễu analog có dạng biên độ và dạng sóng không ổn định. IC SN74LS14N hoạt động như một Schmitt trigger, có nhiệm vụ chuyển đổi tín hiệu analog này sang dạng tín hiệu digital. Tín hiệu digital đầu ra là dạng tín hiệu rời rạc, chỉ có hai mức logic cụ thể là 0 và 1, hay còn gọi là xung vuông kỹ thuật số.

### 4. Khởi D-Flip Flop

IC CD4013BE là một Flip-Flop D được sử dụng để lưu trữ và tạo chuỗi tín hiệu số ngẫu nhiên. Trong mạch, nó nhận xung vuông ngẫu nhiên từ IC SN74LS14N tại chân CLK và thay đổi trạng thái đầu ra Q hoặc Q-

dựa trên tín hiệu này. Mỗi xung clock ở chân CLK sẽ làm IC chuyển đổi giữa hai trạng thái logic 0 hoặc 1, tạo ra chuỗi số nhị phân ngẫu nhiên ở đầu ra.



### Hình 9. Khối D-Flip Flop

## VI. Bảng linh kiện được sử dụng

| Tên linh kiện         | Mô tả                        | Số lượng |
|-----------------------|------------------------------|----------|
| Module tăng áp 5V-12V |                              | 1        |
| SN74LS14N             | Hex-Schmitt Trigger          | 1        |
| CD4013BE              | D-Flop                       | 1        |
| 0.1uF Cap             | Tụ 0.1 microFarad            | 1        |
| 2N3904                | Transistor loại NPN          | 3        |
| 10uF Cap              | Tụ 10 microFarad             | 1        |
| 1M resistors          | Điện trở 1M Ohm              | 1        |
| 4.7k resistors        | Điện trở 4.7K Ohm 1/4W<br>1% | 1        |

|                |                  |   |
|----------------|------------------|---|
| 10k resistors  | Điện trở 10k Ohm | 2 |
| Breadboard     | Mạch test        | 1 |
| Dây nối        |                  |   |
| Arduino UNO R3 |                  | 1 |

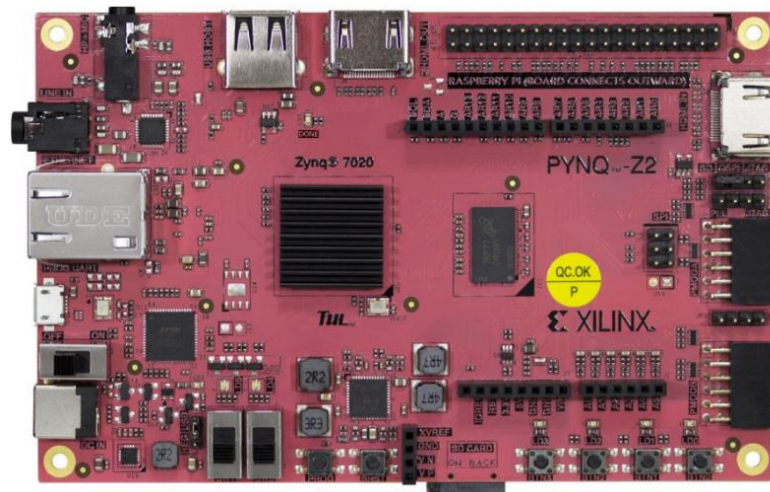
Bảng 1. Các linh kiện

## C. THỰC HIỆN TRÊN FPGA

### I. Giới thiệu

FPGA (Field Programmable Gate Array) là một loại vi mạch tích hợp có thể lập trình để thực hiện các chức năng logic phức tạp. Không giống như các vi điều khiển truyền thống, FPGA cho phép thiết kế phần cứng tùy chỉnh bằng cách cấu hình các khối logic và kết nối nội bộ. FPGA nổi bật với khả năng xử lý song song, hiệu suất cao và độ linh hoạt, phù hợp cho các ứng dụng trong xử lý tín hiệu, viễn thông, và trí tuệ nhân tạo.

Kit PYNQ-Z2 là một nền tảng phát triển dựa trên FPGA Xilinx Zynq-7000, tích hợp cả mạch logic lập trình và bộ xử lý ARM Cortex-A9. Kit hỗ trợ các ứng dụng học sâu, xử lý hình ảnh, và hệ thống nhúng, đồng thời cho phép lập trình dễ dàng bằng ngôn ngữ Python thông qua môi trường PYNQ. Với thiết kế nhỏ gọn và tính năng mạnh mẽ, PYNQ-Z2 được sử dụng rộng rãi trong nghiên cứu, giáo dục và phát triển công nghệ.



Hình 10. KIT PYNQ-Z2 EVALUATION

### II. Cấu hình phần cứng để kết nối mạch tạo số ngẫu nhiên cho PYNQ-Z2

Để thực thi bộ sinh số ngẫu nhiên, trước tiên phải thiết kế và cấu hình cho FPGA. Sau đây là các bước trong quy trình cấu hình cho FPGA để nhận dữ liệu từ mạch, và hiển thị trên màn hình máy tính. Việc thiết kế được thực hiện trên phần mềm Vivado-một công cụ phần mềm của Xilinx được thực hiện theo các bước dưới đây:

a. Tạo Block Design

- Tạo Block Design bằng cách thêm các IP và kết nối chúng.
- Bước này giúp thêm các mô-đun cần thiết để FPGA có thể nhận và truyền dữ liệu.
- Sử dụng công cụ "Validate Design" để kiểm tra thiết kế đáp ứng yêu cầu hệ thống.

b. Tạo tệp Wrapper cho hệ thống

- Tệp Wrapper được dùng để chuyển đổi Block Design sang ngôn ngữ mô tả phần cứng (HDL).
- Chọn "Create HDL Wrapper" để tạo mô hình VHDL cấp cao nhất.

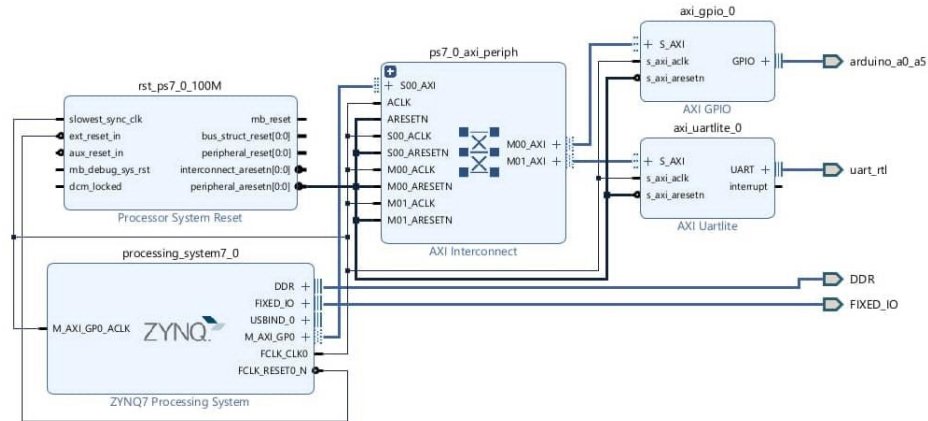
c. Tạo tệp Constraint

- Tệp Constraint ràng buộc các chân đầu ra cho FPGA.
- Để tạo được Constraint cần chạy chế độ Synthesis trước. Đây là quá trình chuyển đổi ngôn ngữ mô tả phần cứng của thiết kế thành netlist (Netlist là một danh sách các cổng và các cấu trúc logic cụ thể trong thiết kế). Bước này còn giúp điều chỉnh và tối ưu hóa thiết kế trước khi thực thi trên FPGA.
- Sau đó ánh xạ các chân cần sử dụng và xuất ra file XDC (constraint).

d. Implementation: Bước này giúp triển khai thiết kế từ mô hình logic thành nguyên mẫu để chạy trên FPGA. Quá trình này để đảm bảo thiết kế được thực thi một cách chính xác trên FPGA.

e. Tạo Bitstream: Đây là một trong những bước quan trọng nhất trong quá trình cấu hình FPGA. Tạo một tệp bitstream để có thể nạp vào FPGA để thực hiện chức năng cụ thể đã được thiết kế.

f. Lập trình trên FPGA: Sau khi tạo tệp bitstream từ bước trước đó, cần tạo code trên phần mềm SDK. Chương trình sẽ được biên dịch và chạy trên FPGA, kết quả sẽ được hiển thị trên phần mềm.



Hình 11. Cấu hình phần cứng của PYNQ-Z2

Sau khi đã cấu hình phần cứng trên PYNQ-Z2, tiếp theo là quá trình kết nối và giao tiếp giữa mạch sinh số ngẫu nhiên và bo mạch PYNQ-Z2.

Dữ liệu từ bộ sinh số ngẫu nhiên sẽ được truyền đến bo mạch PYNQ-Z2 qua chân IO A5. Để dữ liệu có thể truyền, cần thêm các ràng buộc GPIO tại tệp XDC trong quá trình cấu hình. Cần đảm bảo rằng kết nối này được thực hiện đúng cách để dữ liệu có thể truyền đến mạch PYNQ-Z2.

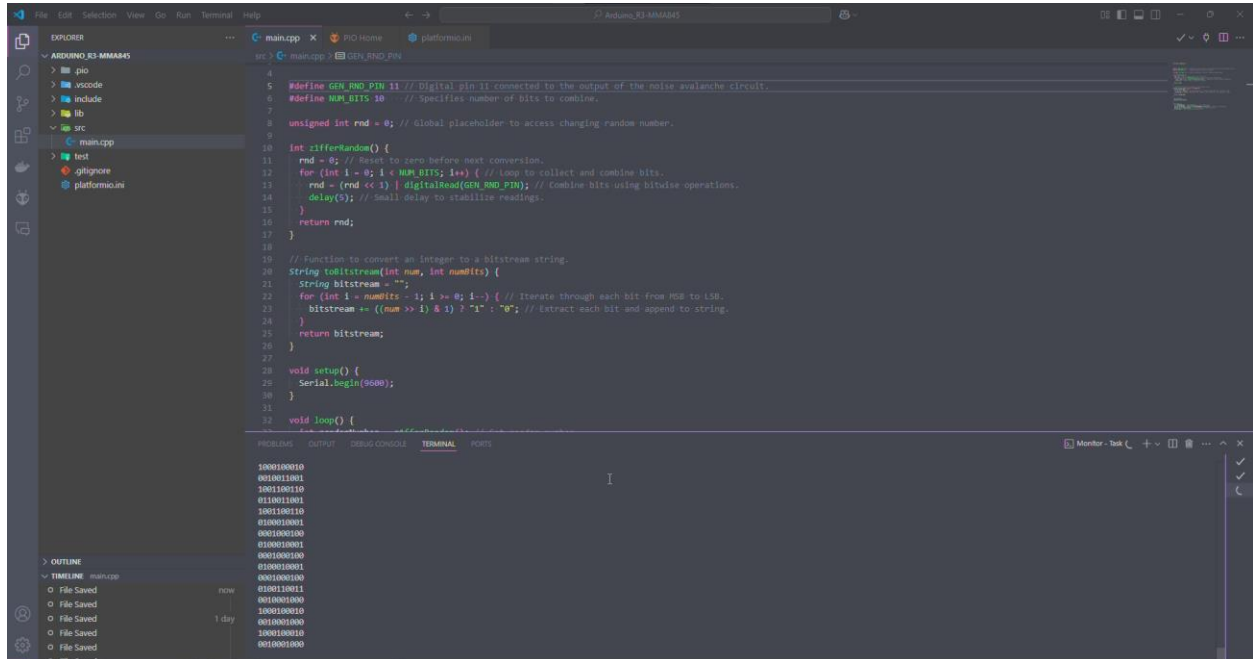
PYNQ-Z2 giao tiếp với máy tính thông qua cổng UART. Vì vậy, cần kết nối các chân UART của PYNQ-Z2 với máy tính và cấu hình AXI-UART trong thiết kế. Cổng UART được sử dụng để gửi và nhận dữ liệu giữa FPGA và máy tính.

Khi PYNQ-Z2 được kết nối với máy tính, dữ liệu đầu ra từ mạch sinh số ngẫu nhiên có thể được truyền và hiển thị trên màn hình máy tính. Dữ liệu được biểu diễn dưới dạng các chuỗi bit 0 và 1, sử dụng phần mềm PUTTY để dữ liệu hiển thị rõ ràng hơn.



### III. Dữ liệu đầu ra

#### 1. Dữ liệu đầu ra trên Arduino



The screenshot shows the Arduino IDE interface with a C++ program in the main editor and the serial monitor at the bottom. The program defines a random pin (GEN\_RND\_PIN = 11) and a number of bits to combine (NUM\_BITS = 10). It includes a function `zifferRandom()` that reads the pin, combines bits, and returns a random number. Another function `convertStream(int num, int numBits)` converts the random number into a binary string. The `setup()` function initializes the serial port at 9600 baud, and the `loop()` function prints the binary string to the serial monitor.

```
1 // Arduino_R3-MMAB45
2
3 #include <Arduino.h>
4
5 #define GEN_RND_PIN 11 // Digital pin 11 connected to the output of the noise avalanche circuit.
6 #define NUM_BITS 10 // Specifies number of bits to combine.
7
8 unsigned int rnd = 0; // Global placeholder to access changing random number.
9
10 int zifferRandom() {
11     rnd = 0; // Reset to zero before next conversion.
12     for (int i = 0; i < NUM_BITS; i++) { // Loop to collect and combine bits.
13         rnd = (rnd << 1) | digitalRead(GEN_RND_PIN); // Combine bits using bitwise operations.
14         delay(5); // Small delay to stabilize readings.
15     }
16     return rnd;
17 }
18
19 // Function to convert an integer to a bitstream string.
20 String convertStream(int num, int numBits) {
21     String bitstream = "";
22     for (int i = numBits - 1; i >= 0; i--) { // Iterate through each bit from MSB to LSB.
23         bitstream += ((num >> i) & 1) ? "1" : "0"; // Extract each bit and append to string.
24     }
25     return bitstream;
26 }
27
28 void setup() {
29     Serial.begin(9600);
30 }
31
32 void loop() {
33     String bitstream = convertStream(zifferRandom(), NUM_BITS);
34     Serial.println(bitstream);
35     delay(1000);
36 }
```

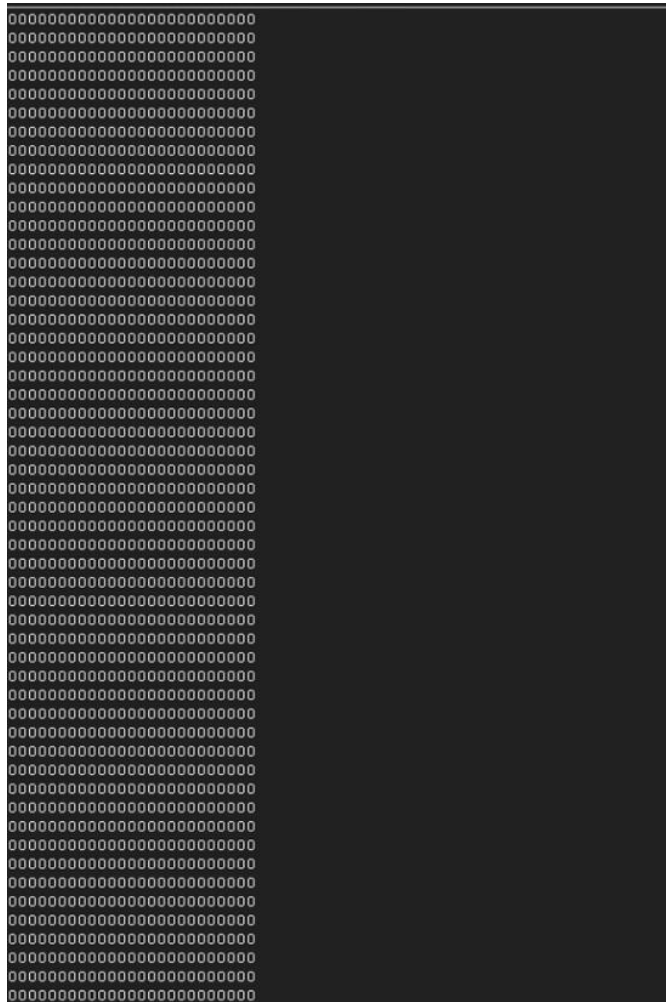
The serial monitor displays the following output:

```
1000100010
0100110001
1001100110
0110011001
1001100110
0100110001
0001000100
0100011001
0010001000
0001000100
0100110011
0010001000
1000100010
0110001000
1000100010
0010001000
```

Hình 12. Kết quả chạy demo trên Arduino UNO R3



## 2. Dữ liệu đầu ra trên PYNQ – Z2



Hình 13. Kết quả chạy demo trên PYNQ – Z2

## IV. Đánh giá kết quả

### 1. Định dạng kết quả:

Kết quả của quá trình sinh số ngẫu nhiên sẽ ghi lại và xuất ra màn hình bởi phần mềm Coolterm. Kết quả đầu ra sẽ được ghi lại vào 1 một file dưới định dạng file text với mỗi dòng là 10 giá trị 0 hoặc 1

### 2. Giới thiệu NIST TEST

Bộ kiểm tra NIST là gói thống kê bao gồm 15 thử nghiệm được phát triển để kiểm tra tính ngẫu nhiên của các chuỗi nhị phân ( dài tùy ý ) được tạo bởi các bộ tạo số ngẫu nhiên hoặc giả ngẫu nhiên bằng mật mã dựa trên phần cứng hoặc phần mềm. Các thử nghiệm này tập trung vào nhiều loại không ngẫu nhiên khác

nhau có thể tồn tại theo một trình tự. Một số bài kiểm tra có thể phân tách thành nhiều bài kiểm tra phụ. Bảng sau đây là 16 bài kiểm tra của bộ NIS TEST.

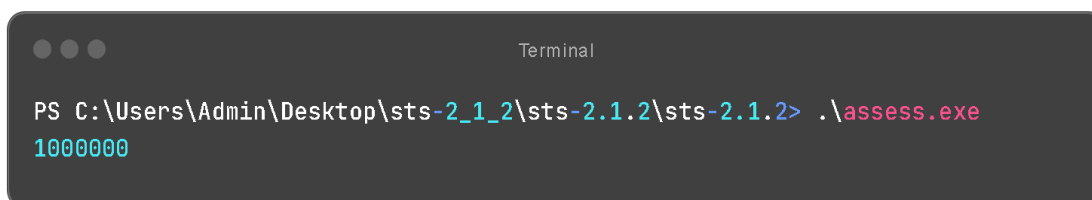
|     |                                    |   |
|-----|------------------------------------|---|
| 1.  | Tần số đơn                         | Kiểm tra tỉ lệ số lượng số 1 so với số 0 trong chuỗi bit, đánh giá xem chuỗi có phân phối đều không.  |
| 2.  | Tần số khối                        | Chia chuỗi thành các khối và kiểm tra sự phân bố của số 1 và số 0 trong từng khối.                    |
| 3.  | Kiểm tra loạt                      | Kiểm tra số lượng chuỗi (runs) liên tục của số 1 hoặc số 0 trong chuỗi bit                            |
| 4.  | Loạt dài nhất của các số 1         | Kiểm tra số lần xuất hiện của chuỗi số 1 dài nhất trong từng khối                                     |
| 5.  | Kiểm tra hạng của ma trận nhị phân | Xác định hạng của ma trận nhị phân được tạo từ chuỗi bit  |
| 6.  | Fourier rời rạc                    | Sử dụng phép biến đổi Fourier rời rạc để kiểm tra tính phổ của chuỗi                                  |
| 7.  | Mẫu không chồng chéo               | Kiểm tra số lần mẫu (template) xuất hiện không chồng lấn trong chuỗi                                  |
| 8.  | So khớp mẫu chồng chéo             | Kiểm tra số lần mẫu xuất hiện có chồng lấn trong chuỗi  |
| 9.  | Thống kê vận năng Maurer           | Sử dụng kỹ thuật thống kê để kiểm tra tính không chu kỳ của chuỗi                                     |
| 10. | Kiểm tra độ phức tạp tuyến tính    | Đo độ phức tạp tuyến tính của chuỗi bit   |
| 11. | Kiểm tra nối tiếp (serial test)    | Kiểm tra sự phụ thuộc giữa các chuỗi bit liên tiếp trong chuỗi  |
| 12. | Entropy gần đúng                   | Đánh giá độ ngẫu nhiên của chuỗi dựa trên độ lệch của thông tin entropy xấp xỉ                        |
| 13. | Tổng tích lũy                      | Kiểm tra xem tổng tích lũy của chuỗi có lớn hay nhỏ so với kỳ vọng của một chuỗi ngẫu nhiên hay không |
| 14. | Du ngoạn ngẫu nhiên                | Đếm số lượng chu kỳ cụ thể trong một cuộc đi bộ ngẫu nhiên của chuỗi                                  |
| 15. | Biến thể du ngoạn ngẫu nhiên       | Đếm số lần một trạng thái cụ thể được thăm trong một cuộc đi bộ ngẫu nhiên của chuỗi                  |

Bảng 2. Bảng các bài kiểm tra của NIST

### 3. Kết quả NIST TEST

#### a. Các bước thực hiện

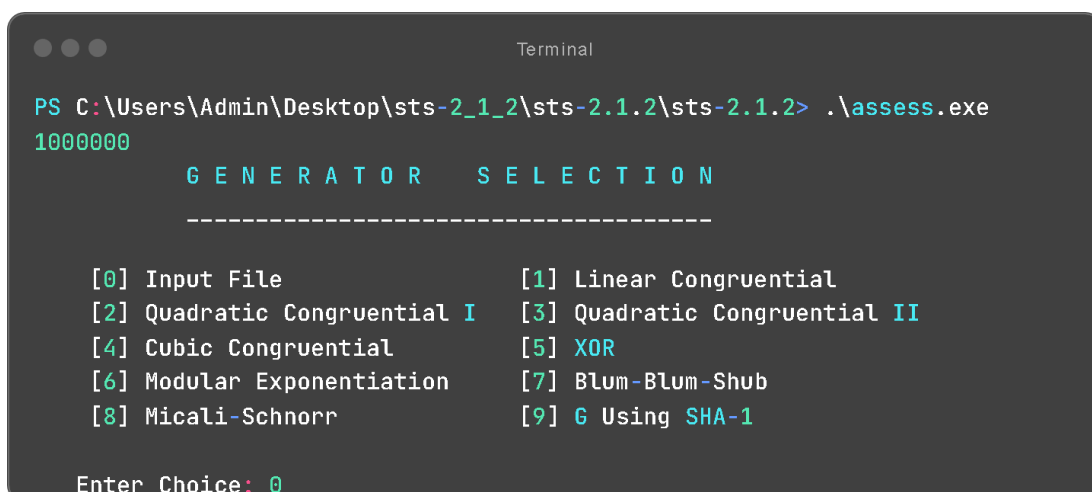
Đầu tiên mở file assess.exe trong Terminal sau đó chạy bằng cách nhập lệnh



```
Terminal
PS C:\Users\Admin\Desktop\sts-2_1_2\sts-2.1.2\sts-2.1.2> .\assess.exe
10000000
```

Hình 14. Bước chạy NIST Test

Hình 12. đầu vào. Sau khi enter, terminal sẽ xuất hiện bảng lựa chọn:



```
Terminal
PS C:\Users\Admin\Desktop\sts-2_1_2\sts-2.1.2\sts-2.1.2> .\assess.exe
10000000

  G E N E R A T O R   S E L E C T I O N
-----

[0] Input File                [1] Linear Congruential
[2] Quadratic Congruential I  [3] Quadratic Congruential II
[4] Cubic Congruential        [5] XOR
[6] Modular Exponentiation    [7] Blum-Blum-Shub
[8] Micali-Schnorr            [9] G Using SHA-1

Enter Choice: 0
```

Hình 15. Bước chạy NIST Test

Thay vì chọn các input có sẵn, ta sẽ chọn file tự tạo được xuất ra từ arduino. Sau đó, một bảng lựa chọn các bài kiểm tra hiện ra, có thể chọn test từng bài hoặc là test tất cả các bài kiểm tra cùng một lúc:

```
Terminal

S T A T I S T I C A L   T E S T S
-----

[01] Frequency                      [02] Block Frequency
[03] Cumulative Sums                [04] Runs
[05] Longest Run of Ones            [06] Rank
[07] Discrete Fourier Transform      [08] Nonperiodic Template

Matchings
[09] Overlapping Template Matchings [10] Universal Statistical
[11] Approximate Entropy             [12] Random Excursions
[13] Random Excursions Variant       [14] Serial
[15] Linear Complexity

INSTRUCTIONS
Enter 0 if you DO NOT want to apply all of the
statistical tests to each sequence and 1 if you DO.

Enter Choice: 1
```

Hình 16. Bước chạy NIST Test

Tiếp theo, bảng điều chỉnh thông số trước khi thực hiện các bài test:

```
Terminal

P a r a m e t e r   A d j u s t m e n t s
-----

[1] Block Frequency Test - block length(M):      128
[2] NonOverlapping Template Test - block length(m): 9
[3] Overlapping Template Test - block length(m):  9
[4] Approximate Entropy Test - block length(m):   10
[5] Serial Test - block length(m):                16
[6] Linear Complexity Test - block length(M):     500

Select Test (0 to continue):
```

Hình 17. Bước chạy NIST Test

b. Sau đây sẽ là kết quả của NIST test với định dạng đầu vào bộ số ngẫu nhiên gồm 5.000.000 mẫu thu thập được từ Arduino:

| Type of Test                                    | P-Value  | Conclusion |
|---|----------|------------|
| 01. Frequency (Monobit) Test                    | 0.442833 | Random     |
| 02. Frequency Test within a Block               | 0.094070 | Random     |
| 03. Runs Test                                   | 0.055463 | Random     |
| 04. Test for the Longest Run of Ones in a Block | 0.167146 | Random     |
| 05. Binary Matrix Rank Test                     | 0.714770 | Random     |
| 06. Discrete Fourier Transform (Spectral) Test  | 0.435542 | Random     |
| 07. Non-overlapping Template Matching Test      | 0.360178 | Random     |
| 08. Overlapping Template Matching Test          | 0.867884 | Random     |
| 09. Maurer's "Universal Statistical" Test       | 0.048087 | Random     |
| 10. Linear Complexity Test                      | 0.997637 | Random     |
| 11. Serial Test:                                | 0.941821 | Random     |
|   | 0.601823 | Random     |
| 12. Approximate Entropy Test                    | 0.049711 | Random     |
| 13. Cumulative Sums Test (Forward)              | 0.701151 | Random     |
| 14. Cumulative Sums Test (Backward)             | 0.745239 | Random     |

| 15. Random Excursions Test |             |          |            |
|----------------------------|-------------|----------|------------|
| State                      | Chi Squared | P-Value  | Conclusion |
| -4                         | 1.657111    | 0.894258 | Random     |
| -3                         | 1.877718    | 0.865793 | Random     |
| -2                         | 7.854219    | 0.164455 | Random     |
| -1                         | 7.168276    | 0.208423 | Random     |

|   |          |          |        |
|---|----------|----------|--------|
| 1 | 1.256552 | 0.939338 | Random |
| 2 | 2.159608 | 0.826648 | Random |
| 3 | 2.425977 | 0.787599 | Random |
| 4 | 2.220518 | 0.817867 | Random |

Bảng 3. Bảng các bài kiểm tra của NIST

| 16. Random Excursions Variant |        |          |            |
|-------------------------------|--------|----------|------------|
| State                         | COUNTS | P-Value  | Conclusion |
| -9                            | 589    | 0.386367 | Random     |
| -8                            | 596    | 0.381736 | Random     |
| -7                            | 617    | 0.431501 | Random     |
| -6                            | 634    | 0.471190 | Random     |
| -5                            | 645    | 0.483739 | Random     |
| -4                            | 670    | 0.585120 | Random     |
| -3                            | 684    | 0.630147 | Random     |
| -2                            | 708    | 0.796597 | Random     |
| -1                            | 749    | 0.528517 | Random     |
| 1                             | 706    | 0.617804 | Random     |
| 2                             | 691    | 0.606199 | Random     |
| 3                             | 692    | 0.698338 | Random     |
| 4                             | 745    | 0.842641 | Random     |
| 5                             | 777    | 0.648969 | Random     |
| 6                             | 751    | 0.836892 | Random     |
| 7                             | 688    | 0.787551 | Random     |
| 8                             | 704    | 0.886769 | Random     |

|   |     |          |        |
|---|-----|----------|--------|
| 9 | 747 | 0.888562 | Random |
|---|-----|----------|--------|

Bảng 4. Bảng các bài kiểm tra của NIST

#### V. Tài liệu tham khảo

1. <https://www.imperva.com/learn/application-security/cyber-security-threats/>
2. <https://www.synopsys.com/designware-ip/technical-bulletin/true-random-number-generator-security-2019q3.html>
3. <https://github.com/jongrover/true-random?tab=readme-ov-file>
4. <https://how2electronics.com/boost-converter-basics-working-design-application/>
5. <https://www.openrandom.org/Resources>