TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

KHOA ĐIỆN TỬ VIỄN THÔNG

BÁO CÁO THỰC TẬP

LINUX PROGRAMMING ASSIGMENT

Sinh viên: Nguyễn Quang Trường 19020647

Nhóm: Nhóm thực tập Nhúng – dự án gNodeB 5G VHT

Hà Nội - 2022

MỤC LỤC

I. GIỚI THIỆU VỀ BÀI TOÁN

1. Bài toán

Linux là một hệ điều hành máy tính mã nguồn mở, cho phép người dùng có thể thực hiện tương tác chủ động và khai thác trực tiếp trên đó. Với ưu thế sử dụng command line trên terminal thực hiện các thao tác và nhiều ưu điểm khác, hệ điều hành Linux ngày càng phổ biến trong nhiều chủ đề công nghệ, trong đó có lập trình nhúng.

Programming Linux cho phép người sử dụng thực hiện lập trình tương tác với hệ điều hành Linux với nhiều ngôn ngữ lập trình khác nhau. Ở đây, lập trình C với bài toán: Lập trình tương tác với Luồng (Thread) thực hiện lấy mẫu thời gian thực trên hệ thống Linux theo chu kỳ sẽ được triển khai trong báo cáo này.

2. Cơ sở lý thuyết

2.1 Luồng (Thread)

Luồng là một phần của tiến trình (process), một nhánh hoạt động độc lập và có thể tương tác với nhau hoặc với luồng của tiến trình khác. Các thao tác thực hiện với luồng sẽ nhanh hơn và đơn giản hơn so với khi thực hiện với tiến trình. Một tiến trình có thể có đơn luồng (sigle thread) hoặc đa luồng (multi thread). Đơn luồng sẽ hoạt động nhanh, còn đa luồng sẽ thực hiện được nhiều tác vụ trong một tiến trình. Lập trình đa luồng đòi hỏi sự cẩn thận bởi sự phức tạp của các tương tác giữa các luồng với nhau.

Lập trình đa luồng trên hệ điều hành linux cần sử dụng thư viện `pthread.h` và khi thực hiện chạy chương trình bằng command line cần có `-pthread`.

Tạo Thread:

```
int pthread_create(pthread_t *thread, const pthread_attr_t
*attr, void*(*start)(void *), void *arg;
```

Kết thúc Thread:

```
int pthread_join(pthread_t *thread, void **retval);
```

Hoặc có thể thực hiện thoát một Thread như sau:

```
Int pthread_exit(void *retval);
```

Ngoài ra, Thread còn được đồng bộ bằng mutex. Mutex có thể cho phép một thread trong một thời điểm nhất định được hoạt động và ngăn chặn các tương tác giữa các thread khác. Dưới đây là các mã code thực hiện với mutex:

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

2.2 Thao tác với thời gian trên hệ thống

Thời gian thực trên hệ thống được thực hiện với thư viện `time.h`. Thư viện sẽ cung cấp các hàm thực hiện lấy mẫu thời gian thực trên hệ thống. Các hàm được sử dụng như:

Hàm clock_gettime() thực hiện đọc thời gian thực và trả về giá trị thời gian được khai báo trong hàm.

```
Int clock_gettime(clockid_t clockid, struct timespec *tp);
```

Hàm nanosleeps() đưa chương trình vào chế độ "ngủ" trong một khoảng thời gian được đưa vào trong hàm.

```
Int nanosleep(const struct timespec *request, struct
timespec *remain);
```

Hàm clock_nanosleeps() cũng sẽ thực hiện cộng dồn thời gian "ngủ" được đưa vào với mốc thời gian được lấy trước đó giúp tăng hiệu quả lấy mẫu của chương trình.

```
Int clock_nanosleep(clockid_t clockid, int flags, const
struct timespec *request, struct timespec *remain);
```

2.3 Lập trình Shell Scripts

Shell scripts là một chương trinhg thông dịch lệnh của hệ điều hành, cung cấp khả năng tương tác đồng thời của các dòng lệnh với hệ thống thay vì thực hiện nhập từng dòng lệnh lên terminal. Shell script thường được bắt đầu bởi `#!/bin/bash` và đuôi file có dạng `.sh` . Việc thực hiện viết một shell scripts giúp giảm các thao tác chạy các chương trình phức tạp, các lệnh được lưu tương tự với các lệnh nhập vào terminal. Nhưng bên cạnh đó, shell scripts chạy khác

chậm và phức tạp với các chương trình lớn; một lỗi trong chương trình shell scripts có thể gây ảnh hướng lớn tới chương trình.

II. TRIỂN KHAI BÀI TOÁN VÀ ĐÁNH GIÁ KẾT QUẢ

1. Triển khai bài toán

1.1 Phân tích

Bài toán Lập trình C với các Thread thực hiện lấy mẫu thời gian thực của hệ thống theo chu kỳ được mô tả như sau:

Thread SAMPLE thực hiện vô hạn lần nhiệm vụ đọc thời gian hiện tại của hệ thống với đơn vị ns và lưu vào biến T theo chu kỳ lấy mẫu X

Thread INPUT thực hiện kiểm tra file "freq.txt" để xác định chu kỳ lấy mẫu X. Người dùng có thể đưa giá trị chu kỳ X vào file "freq.txt" để thread INPUT cập nhật lại.

Thread LOGGING thực hiện in giá trị biến T và interval, khoảng thời gian giữa hai mốc lấy mẫu vào file "time_and_interval.txt"

Chương trình được thực hiện lấy mẫu với các chu kỳ khác nhau: 1000000ns, 100000ns, 10000ns, 1000ns và 100ns. Viết một chương trình Shell scripts thực hiện chay đồng thời chương trình C và đưa lần lượt các giá trị chu kỳ lấy mẫu vào file "freq.txt".

Thực hiện khảo sát các file với giá trị interval của các chu kỳ lấy mẫu và đánh giá chương trình C.

1.2 Code chương trình

Chương trình chính Thread.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/time.h>
#include <string.h>
#include <signal.h>
#include <math.h>


static int X;                // Chu ky lay mau X
static char  T[50];           // Bien luu thoi gian
static long int interval;    // Bien luu khoang thoi gian giua
hai moc


static struct timespec ts, request;
static long int time_sec_prev;
static long int time_nsec_prev;
```

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

FILE *file_print;
FILE *file_freq;

void *Sampling(void * SAMPLE){

    clock_gettime(CLOCK_REALTIME, &request);

    while (1){
     pthread_mutex_lock(&mutex);

        clock_gettime(CLOCK_REALTIME, &ts);
        // printf("Current Time: %ld.%09ld\n", ts.tv_sec,
ts.tv_nsec);

     // Lap lai voi chu ky X (nanoseconds)
     request.tv_nsec +=X;
     if (request.tv_nsec > 1000000000){
         request.tv_nsec -= 1000000000;
         request.tv_sec ++;
     }

     clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &request,
NULL);

        // Luu thoi gian vao bien T
        char buff[50];
        strftime(buff, sizeof buff, "%s", gmtime(&ts.tv_sec));
        snprintf(T, sizeof T, "%s.%09ld\n", buff, ts.tv_nsec);
     pthread_mutex_unlock(&mutex);
    }
```

```
}

void *Logging_Print(void *LOGGING){

    file_print = fopen("freq_1000000ns_test.txt","a");

    while(1){

     interval = (1e9*ts.tv_sec+ts.tv_nsec) -
(1e9*time_sec_prev+time_nsec_prev);

     time_sec_prev = ts.tv_sec;
     time_nsec_prev = ts.tv_nsec;

     // In gia tri thoi gian trong bien T vaof file
tiime_and_interval.txt

     if (file_print && interval != 0)
            fprintf(file_print,"%ld\n",interval);

    }

    fclose(file_print);
}

void *Reading_Input(void *INPUT){

    while(1){

     // Doc gia tri chu ky lay mau tu file freq.txt
     file_freq = fopen("freq.txt","r");
     fscanf (file_freq, "%d", &X);
     //printf("Frequency = %d\n",X);
```

```
     fclose(file_freq);


    }
}


int main(int argc, char** argv) {
    pthread_t SAMPLE, LOGGING, INPUT;


    pthread_create( &INPUT, NULL, Reading_Input, (void*)
&INPUT);
    pthread_create( &SAMPLE, NULL, Sampling, (void *)&SAMPLE);
    pthread_create( &LOGGING, NULL, Logging_Print, (void*)
&LOGGING);




    pthread_join( INPUT, NULL);
    pthread_join( SAMPLE, NULL);
    pthread_join( LOGGING, NULL);


    exit(0);
}
```

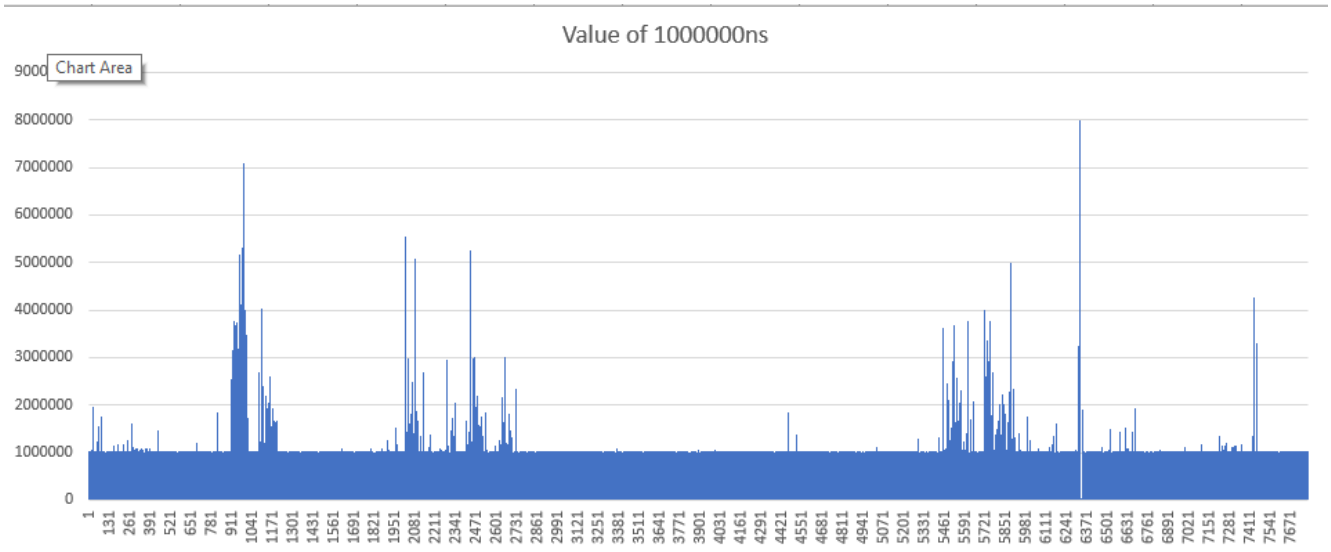
Chương trình Shell Scripts Thread.sh:

```
#!/bin/bash


gcc -pthread -o Thread Thread.c


for period in 1000000 100000 10000 1000 100
do
    echo "$period" > freq.txt
    timeout 30 ./Thread
done
```

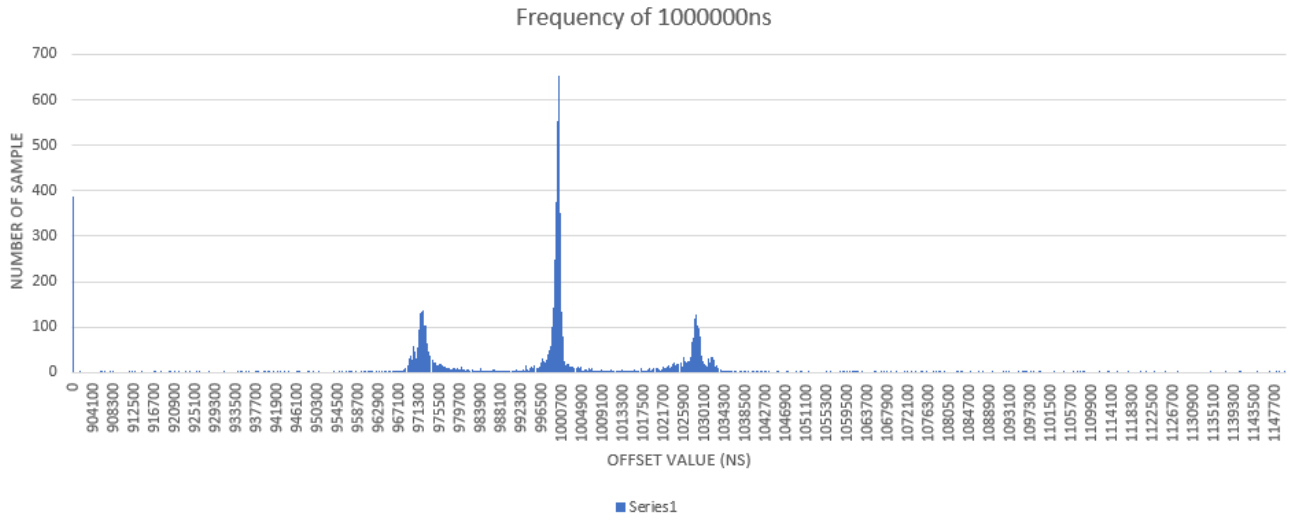
2. Đánh giá kết quả

2.1 Với chu kỳ 1000000ns

Đồ thị giá trị thời gian và đồ thị histogram của chu kỳ 1000000ns được mô tả trong hình 1 và hình 2 như sau:



Hình 1. Giá trị offset của giữa hai lần lấy mẫu của chu kỳ 1000000ns

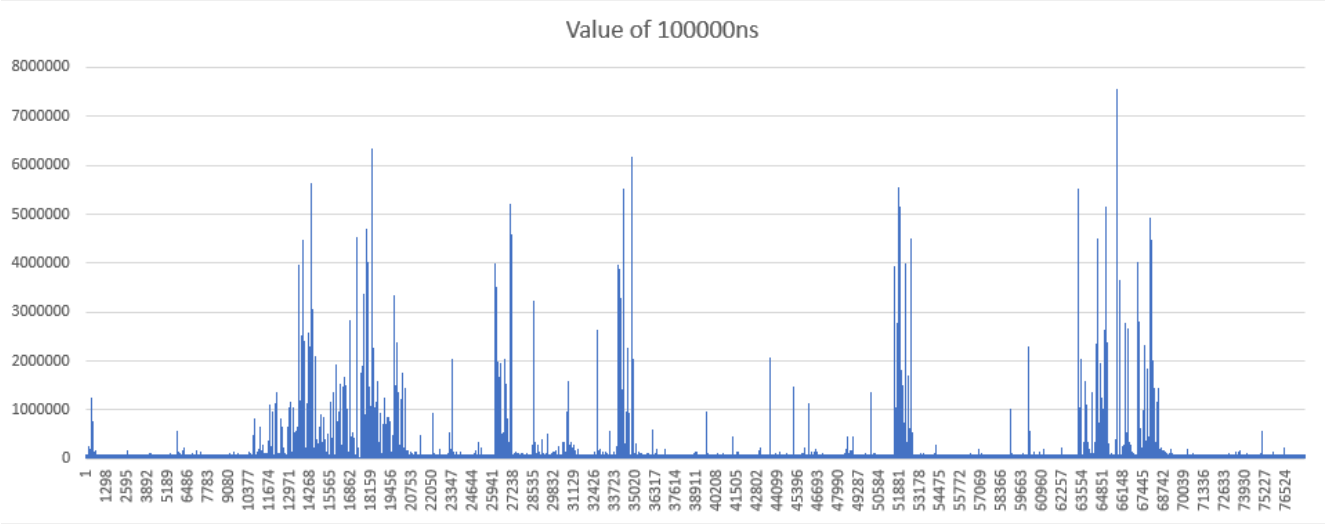


Hình 2. Phần bố giá trị offset của chu kỳ 1000000ns

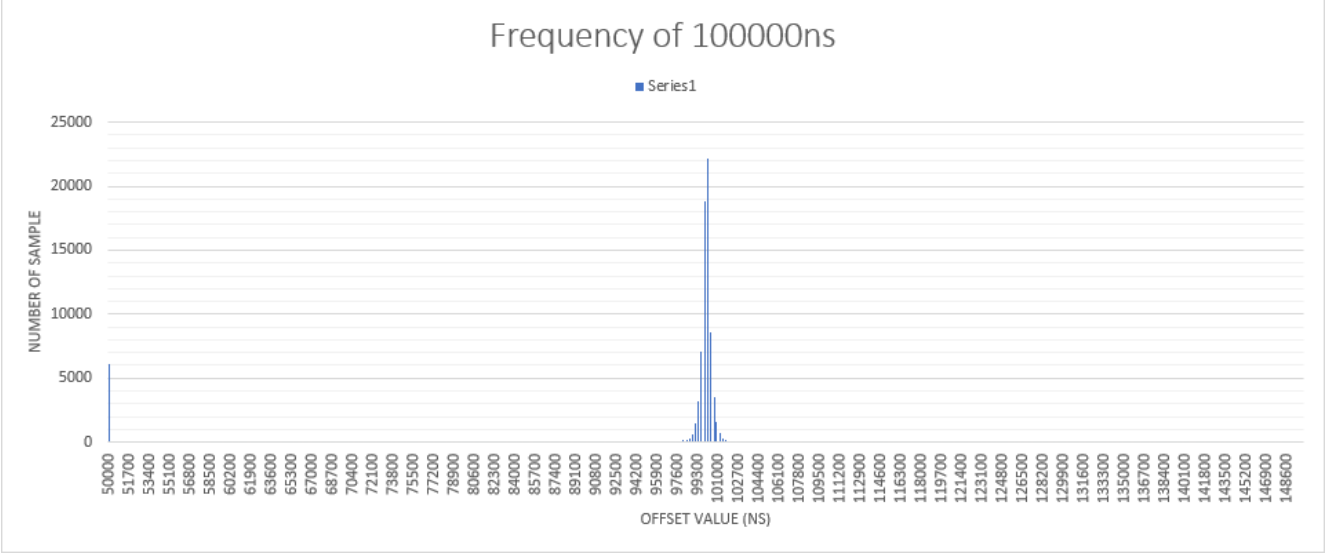
Từ đồ thị, ta thấy giá trị offset giữa hai lần lấy mẫu thời gian được phân bố gần với giá trị của chu kỳ lấy mẫu là 1000000ns với sai số nhỏ. Phần lớn số lượng mẫu thu được có sai số là khoảng [700ns – 2500ns] nhưng vẫn có số lượng offset có sai số khoảng [25000ns-30000ns].

2.2 Với chu kỳ 100000ns

Đồ thị giá trị thời gian và đồ thị histogram của chu kỳ 100000ns được mô tả trong hình 3 và hình 4 như sau:



Hình 3. Giá trị offset giữa hai lần lấy mẫu của chu kỳ 100000ns

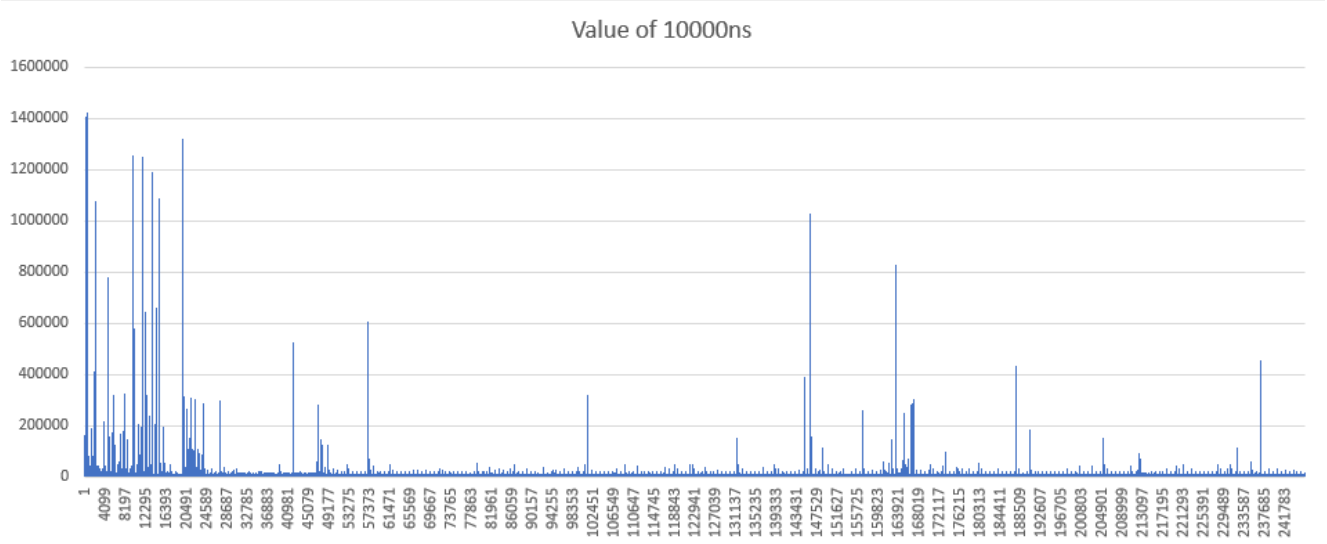


Hình 4. Phân bố giá trị offset của chu kỳ 100000ns

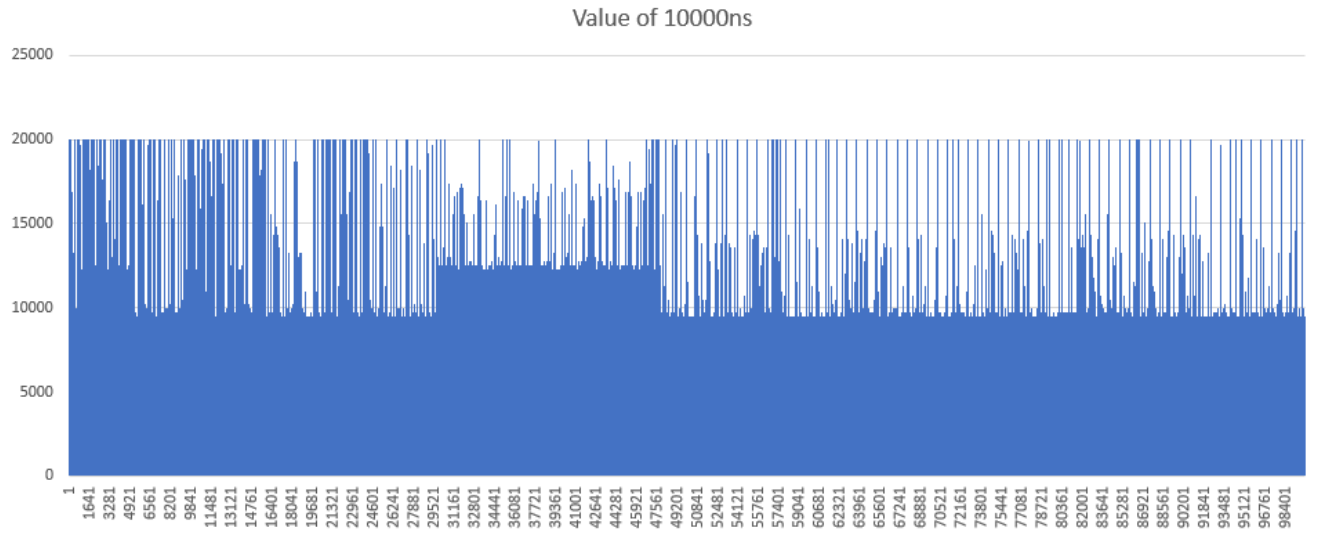
Trong chu kỳ lấy mẫu 100000ns, số lượng giá trị offset gần với giá trị chu kỳ lấy mẫu rất lớn với sai số trong khoảng [700ns-1000ns]. Nhưng trong hình 3, xuất hiện nhiều giá trị offset có độ lớn 7ms.

2.3 Với chu kỳ 10000ns

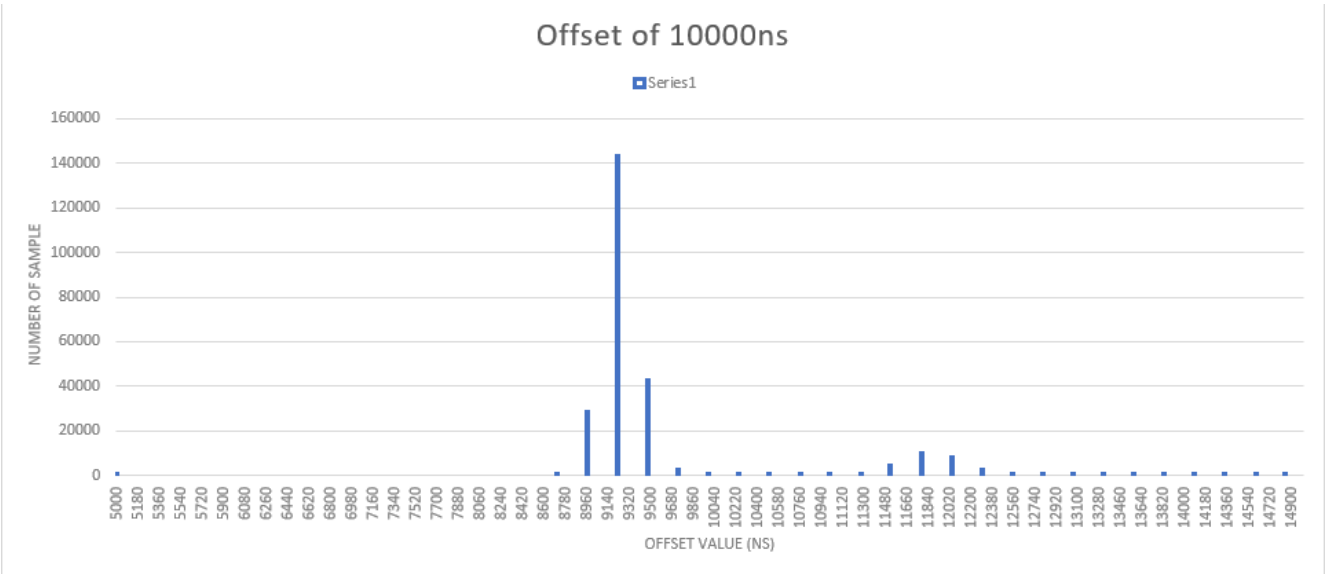
Đồ thị giá trị thời gian và đồ thị histogram của chu kỳ 10000ns được mô tả trong hình 5, hình 6 và hình 7 như sau:



Hình 5. Giá trị offset giữa hai lần lấy mẫu của chu kỳ 10000ns



Hình 6. Giá trị offset của chu kỳ 10000ns giới hạn trong mức nhỏ hơn 20000ns

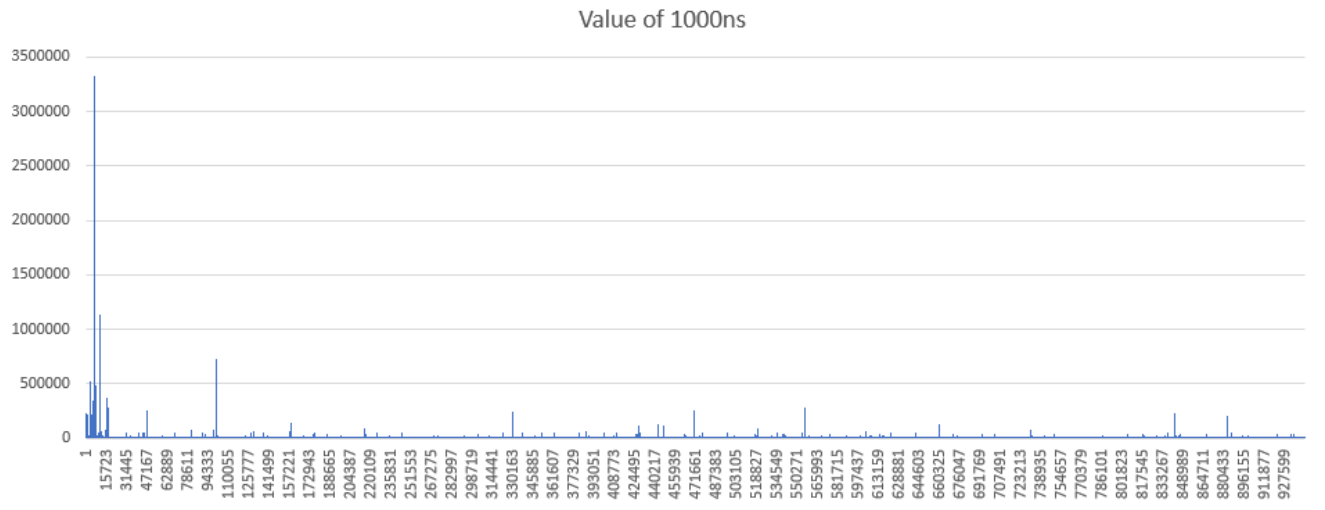


Hình 7. Phân bố giá trị offset của chu kỳ 10000ns

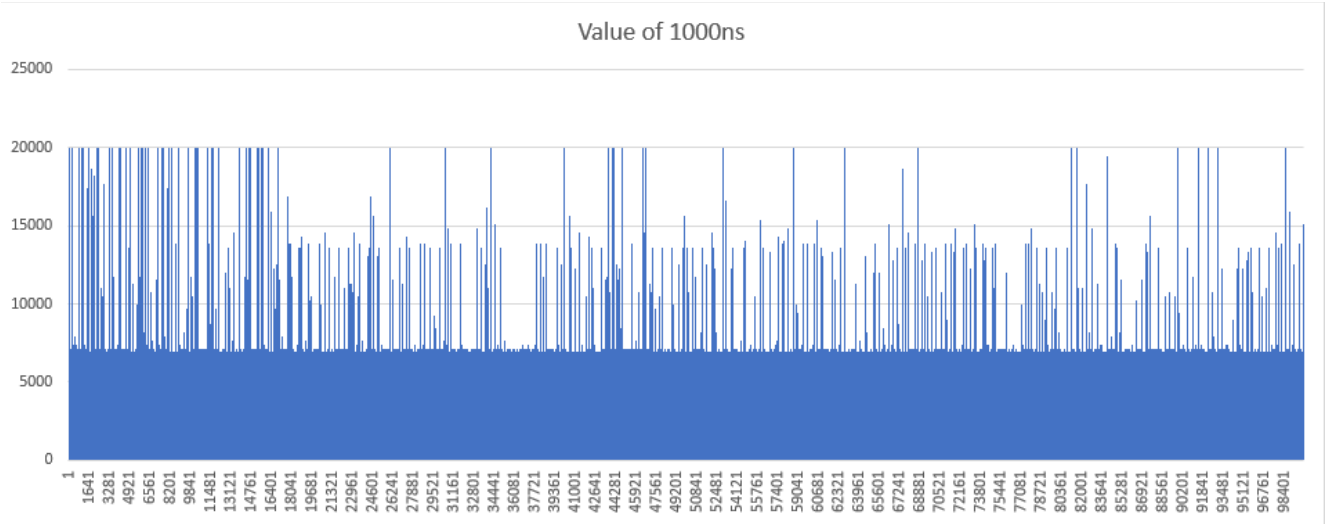
Với chu kỳ 10000ns, phần lớn số lượng offset phân bố tại khoảng giá trị [8900ns -9600ns] và một số tại [11400ns -12300ns] . Đây là khoảng sai số gần với chu kỳ 10000ns, sai số khoảng [1000ns -2000ns]. Ở hình 3 các giá trị vọt lố khoảng 1,2ms -1,4ms.

2.4 Với chu kỳ 1000ns

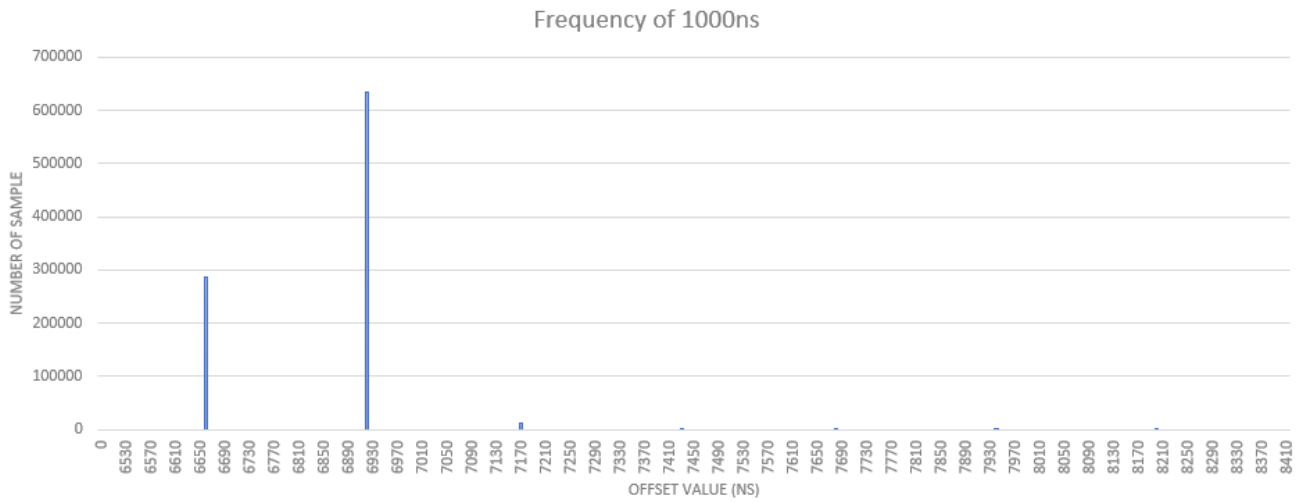
Đồ thị giá trị thời gian và đồ thị histogram của chu kỳ 1000ns được mô tả trong hình 8, hình 9 và hình 10 như sau:



Hình 8. Giá trị offset giữa hai lần lấy mẫu của chu kỳ 1000ns



Hình 9. Giá trị offset của chu kỳ 1000ns giới hạn trong mức nhỏ hơn 20000ns

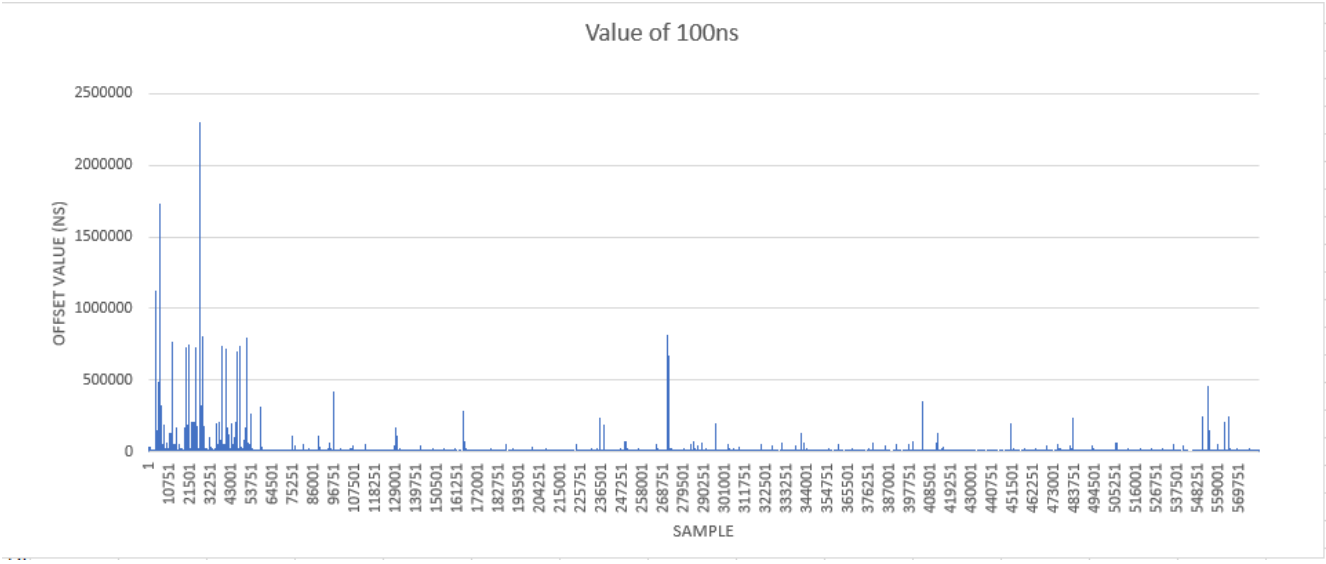


Hình 10. Phân bố giá trị offset trong chu kỳ 1000ns

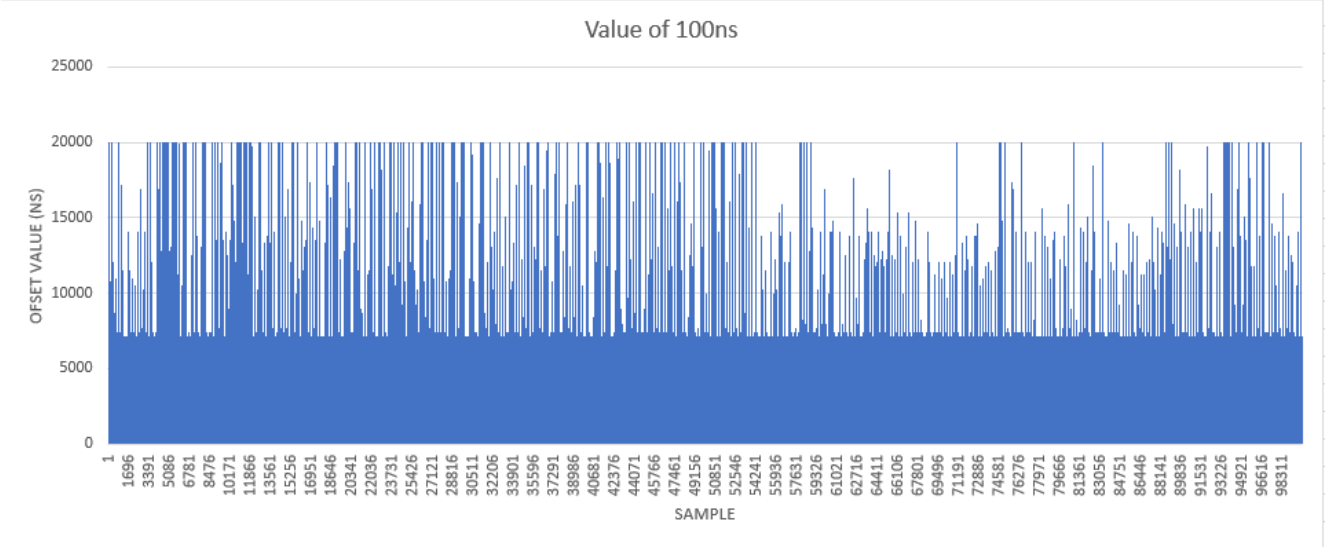
Trong chu kỳ 1000ns, đa phần các giá trị offset có giá trị khoảng 6650ns và 6900ns. Đây là mức sai số khá lớn so với giá trị của chu kỳ lấy mẫu là 1000ns. Trong hình 8, các giá trị offset vọt lố ít hơn nhưng vẫn rất cao khoảng 3ms.

2.5 Với chu kỳ 100ns

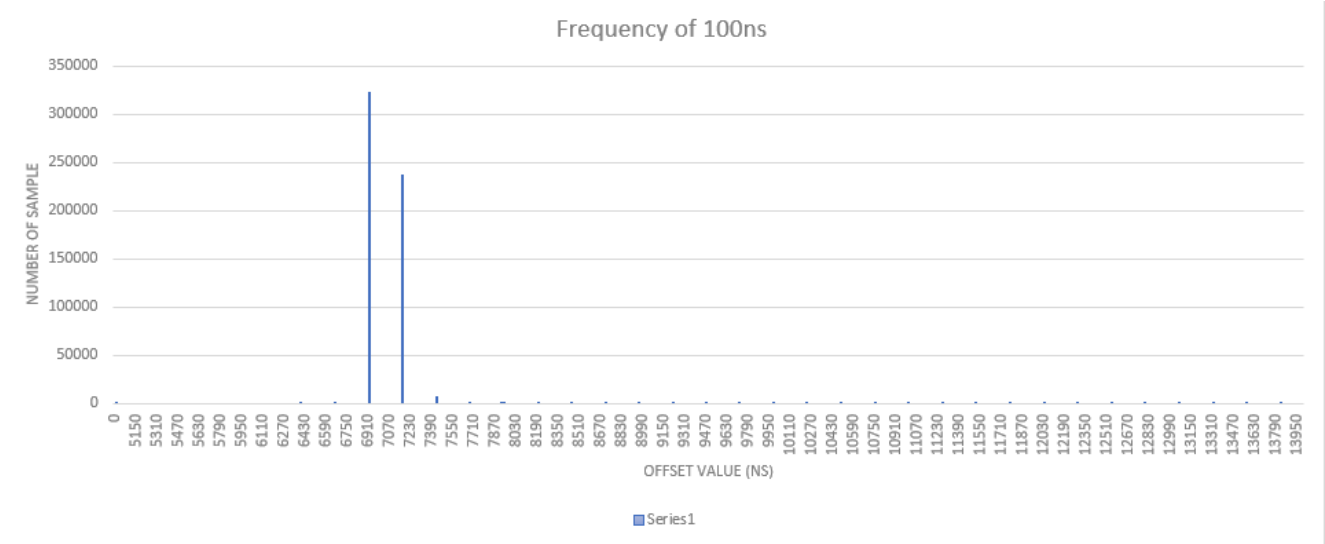
Đồ thị giá trị thời gian và đồ thị histogram của chu kỳ 100ns được mô tả trong hình 11, hình 12 và hình 13 như sau:



Hình 11. Giá trị offset giữa hai lần lấy mẫu của chu kỳ 100ns



Hình 12. Giá trị offset của chu kỳ 100ns với mức giới hạn nhỏ hơn 20000ns trong 100000 mẫu



Hình 13. Phân bố giá trị offset của chu kỳ 100ns

Từ các đồ thị trên, với chu kỳ 100ns rất nhỏ, các giá trị offset thuộc khoảng [6900ns – 7200ns]. Đây là một khoảng sai số khá lớn, thể hiện chương trình chưa thể lấy mẫu chính xác ở mức 100ns. Hình 11 thể hiện các giá trị vọt lố của offset lên đến 3ms như các chu kỳ lấy mẫu 1000ns và 10000ns.

III. KẾT LUẬN

1. Khó khăn gặp phải

- Hạn chế trong kiến thức về Thread và Time

- Chưa làm quen với tạo chương trình trên shell scripts.

- Chưa tối ưu được chương trình chính.

2. Kết luận

Chương trình hoàn thành việc lấy mẫu thời gian thực của hệ thống, tính toán được giá trị thời gian chênh lệch giữa hai lần lấy mẫu. Từ đó có cách đánh giá về chương trình. Bên cạnh đó, chương trình vẫn chưa được tối ưu tốt nhất do có sai số lớn ở chu kỳ 100ns, 1000ns và xuất hiện những giá trị offset lớn lên đến 3ms.