BÁO CÁO THỰC TẬP

LINUX PROGRAMMING ASSIGMENT

Nhóm thực tập Nhúng – dự án gNodeB 5G VHT

Sinh viên: Đinh Tiến Dương
MSSV: 18020394

Hà Nội, 19/7/2022

MỤC LỤC

I. PHẦN GIỚI THIỆU

1. Đặt vấn đề

Tìm hiểu luồng trong C (pthread), xây dựng các luồng để lấy được thời gian hiện tại của hệ thống (đơn vị nanoseconds) , lưu trữ kết quả ra file txt cũng như tính toán thời gian chênh lệch giữa hai lần lấy mẫu. Thời gian lấy mẫu được lấy từ file txt.

2. Lý thuyết liên quan

Trong phần này, để giải quyết được các vấn đề đặt ra, ta sẽ sử dụng công cụ trong lập trình C Linux như Multi Thread ,Thread synchronization, Clock_nanosleep(), shell script.

2.1 Multi thread

2.1.1 Khởi tạo

```
#include <pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start)(void *), void *arg);
                                    Returns 0 on success, or a positive error number on error
```

2.1.2 Thread_join

```
include <pthread.h>

int pthread_join(pthread_t thread, void **retval);
                                    Returns 0 on success, or a positive error number on error
```

2.1.3 Kết thúc

```
include <pthread.h>

void pthread_exit(void *retval);
```

2.2 Thread synchronization

Để đồng bộ hóa các Thread, tránh xung đột khi các thread cùng truy cập đến condition variable tại một thời điểm, ta sử dụng khóa Mutex

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

Both return 0 on success, or a positive error number on error

---

2.3 Clock_nanosleep()

Để có thể tạo các task sau mỗi khoảng thời gian nhất định (nanosecond), ta cần sử dụng hàm Clock_nanosleep().
Khai báo cho Clock_nanosleep()

```
#define _XOPEN_SOURCE 600
#include <time.h>

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

Returns 0 on successfully completed sleep,
or a positive error number on error or interrupted sleep

Trong đó hàm Clock_nanosleep() có ưu điểm hơn so với hàm nanosleep() đó chính là trong lúc hàm Clock_nanosleep() tình toán thời gian, hệ thống vẫn có khả năng hoạt động với các luồng tiếp theo, trong khi nanosleep() sẽ làm hệ thống bị treo trong khoảng thời gian đếm thời gian.

2.4 Shell script

Với vấn đề đặt ra như trên, Shell script giúp ta có thể thao tác với file freq.txt (file chứa thời gian lấy mẫu của chương trình) mà không phải thay đổi giá trị thủ công bằng cách truy cập vào file đó.

II. TRIỂN KHAI VÀ PHÂN TÍCH KẾT QUẢ

1. Chạy 3 luồng (INPUT, SAMPLE, LOGGING)

1.1 Thread SAMPLE

Souce code :

```
void *getTime(void *args)
{
   clock_gettime(CLOCK_REALTIME, &request1);
   while (check_loop == 1)
   {
      clock_gettime(CLOCK_REALTIME, &tp);
         request1.tv_nsec += freq;
         if(request1.tv_nsec > 1000000000)
         {
          request1.tv_sec +=1;
```

```
            request1.tv_nsec -= 1000000000;
          }
          if (clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &request1,
    NULL) != 0)
          {
             check_loop = 0;
          }
          else
          {
             check_loop = 1;
          }

       }
    }
```

1.2 Thread INPUT

Souce code

```
void *getFreq(void *args)
{
   while (check_loop)
   {
      pthread_mutex_lock(&mutex);
      FILE *fp;
      fp = fopen("freq.txt", "r");
      unsigned long new_freq = 0;
      fscanf(fp, "%lu", &new_freq);
      fclose(fp);
      if (new_freq == freq)
      {
         pthread_mutex_unlock(&mutex);
      }
      else
      {
         freq = new_freq;
         pthread_mutex_unlock(&mutex);
      }
   }
}
```

1.3 Thread LOGGING

Souce code

```
void *save_time(void *args)
{
   // save time
   while (1)
   {
      FILE *file;
      file = fopen("freq_100.txt", "a+");
      long diff_sec = tp.tv_sec - tmp.tv_sec;
```

```
            long diff_nsec;
            if (tmp.tv_nsec != tp.tv_nsec || tmp.tv_sec != tp.tv_sec)
            {
               if (tp.tv_nsec > tmp.tv_nsec)
               {
                  diff_nsec = tp.tv_nsec - tmp.tv_nsec;
               }
               else
               {
                  diff_nsec = 1000000000 + tp.tv_nsec - tmp.tv_nsec;
                  diff_sec = diff_sec - 1;
               }
               fprintf(file, "\n%ld.%09ld", diff_sec, diff_nsec);
               tmp.tv_nsec = tp.tv_nsec;
               tmp.tv_sec = tp.tv_sec;
            }
            fclose(file);
         }
      }
```

2. Chạy Shell script và file C

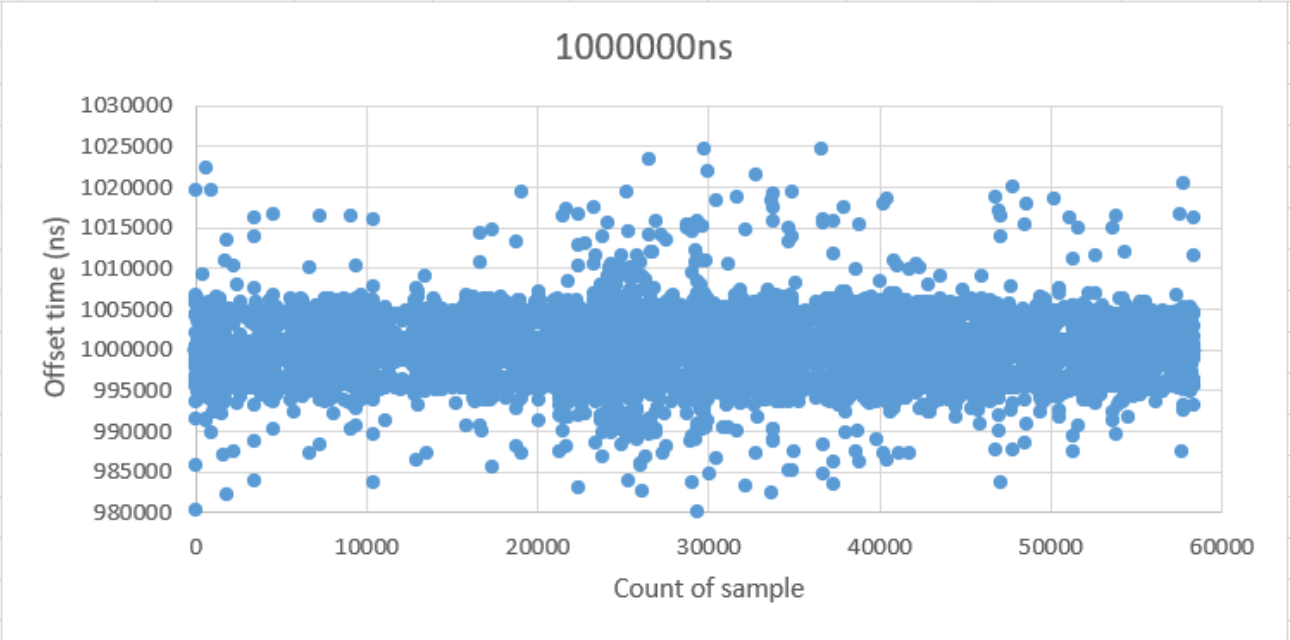
Chương trình được chạy trong vòng 5 phút, trong đó ứng với mỗi chu kì lấy mẫu 1000000ns đến 100ns, mỗi chu kì sẽ lấy mẫu trong vòng 1 phút. Kết quả trả về được lưu vào trong file time_and_interval.txt

Cấu trúc chương trình Shell script :

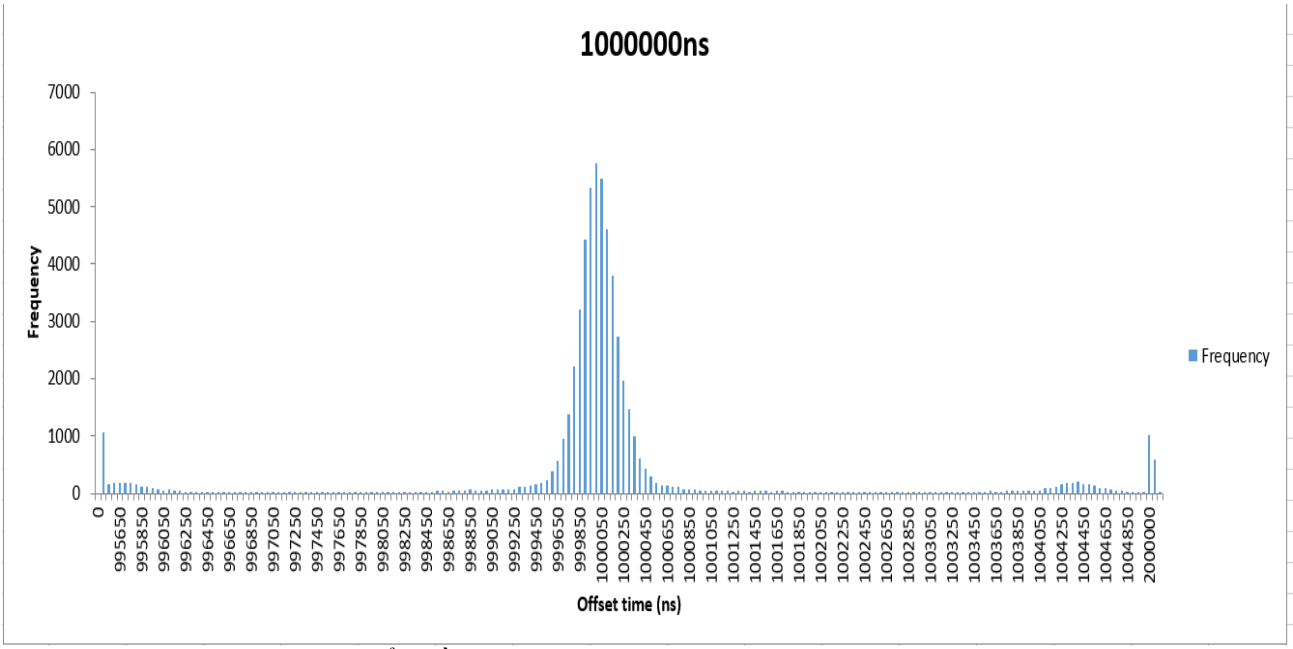
```
#!/bin/sh

echo "1000000">freq.txt
timeout 60s ./b1
echo "100000">freq.txt
timeout 60s ./b1
echo "10000">freq.txt
timeout 60s ./b1
echo "1000">freq.txt
timeout 60s ./b1
echo "100">freq.txt
timeout 60s ./b1
```

Trong đó echo “freq” > freq.txt : thay đổi chu kỳ lấy mẫu trong file freq.txt
Timeout 60s ./b1: Chạy file C trong vòng 60s.

2.1 Với chu kỳ X = 1000000ns

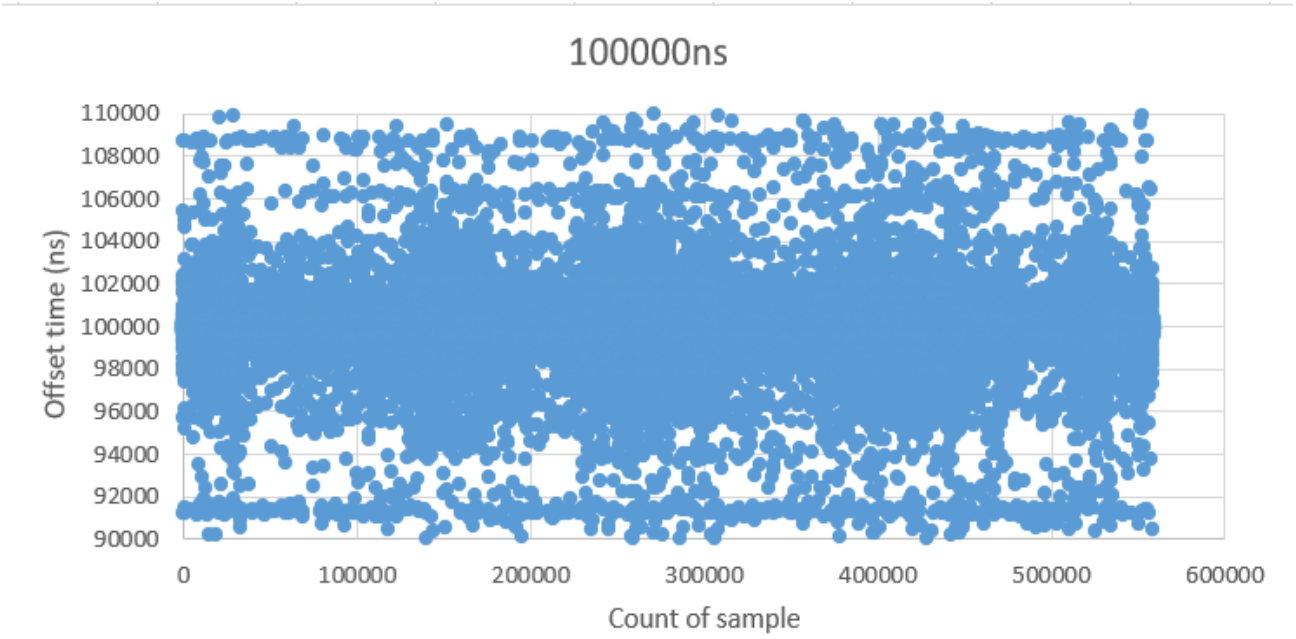


Đồ thị giá trị interval khi chu kỳ X = 1000000ns

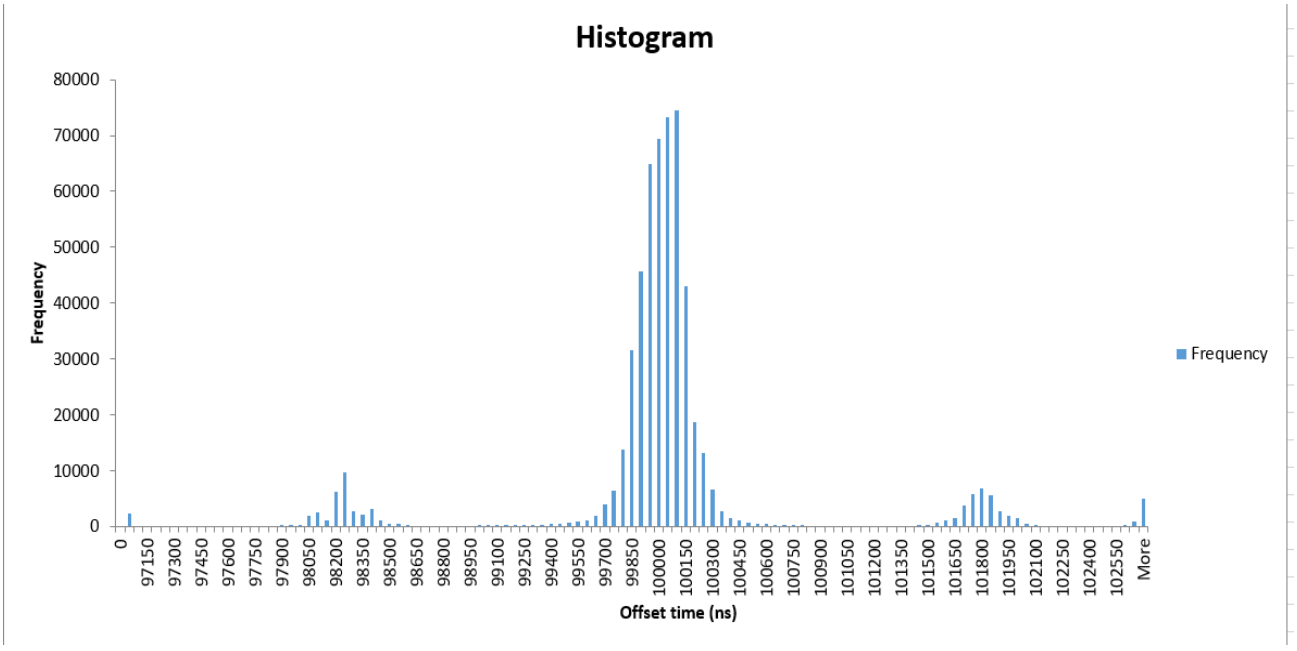


Biểu đồ histogram khi chu kỳ X = 1000000ns

2.2 Với chu kỳ X = 100000ns

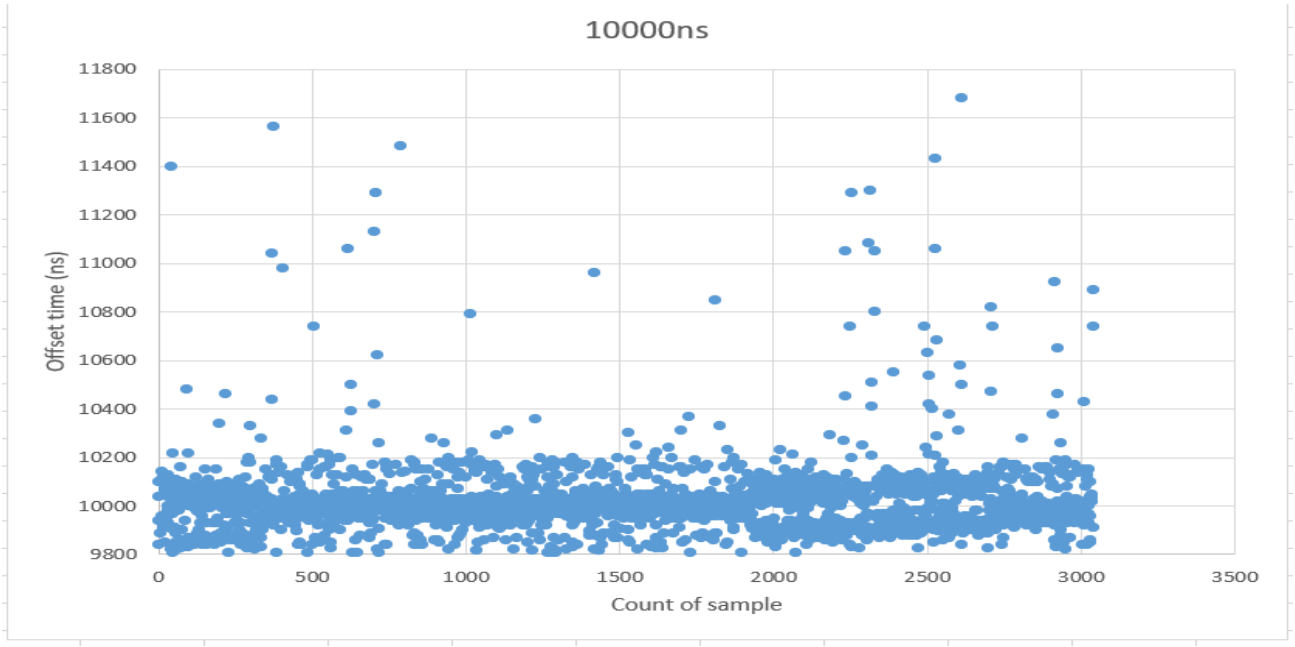


Đồ thị giá trị interval khi chu kỳ X = 100000ns

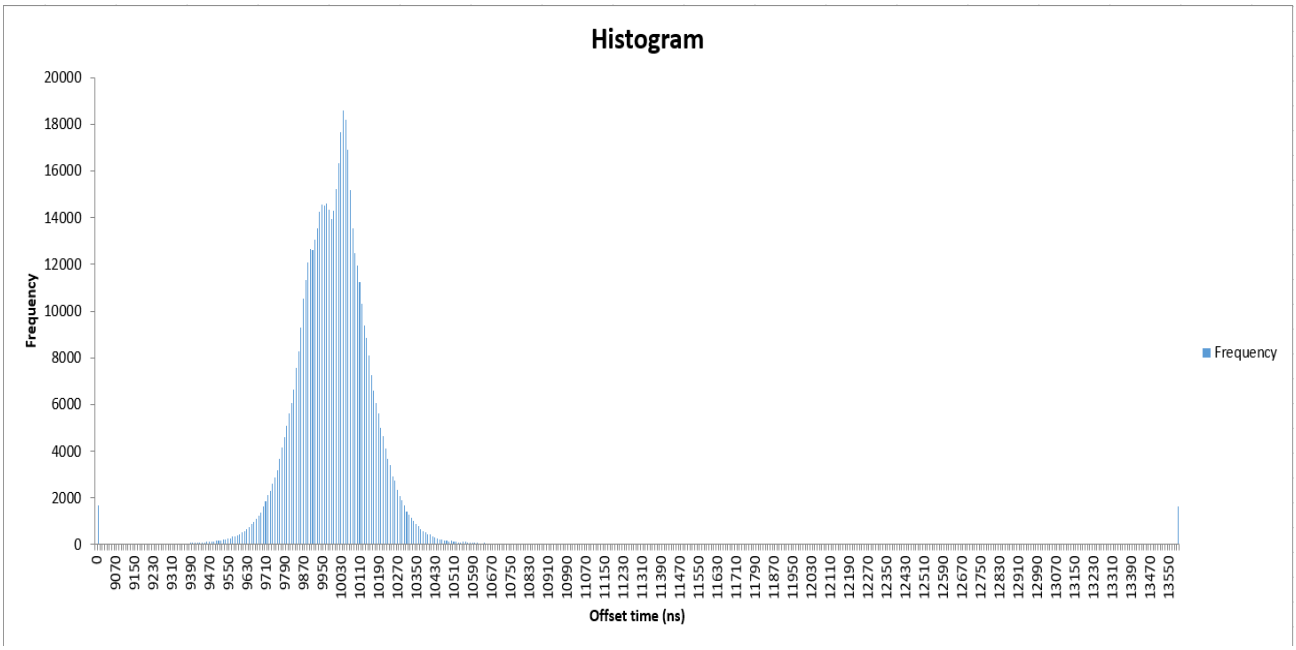


Biểu đồ histogram khi chu kỳ X = 100000ns

2.3 Với chu kỳ X = 10000ns

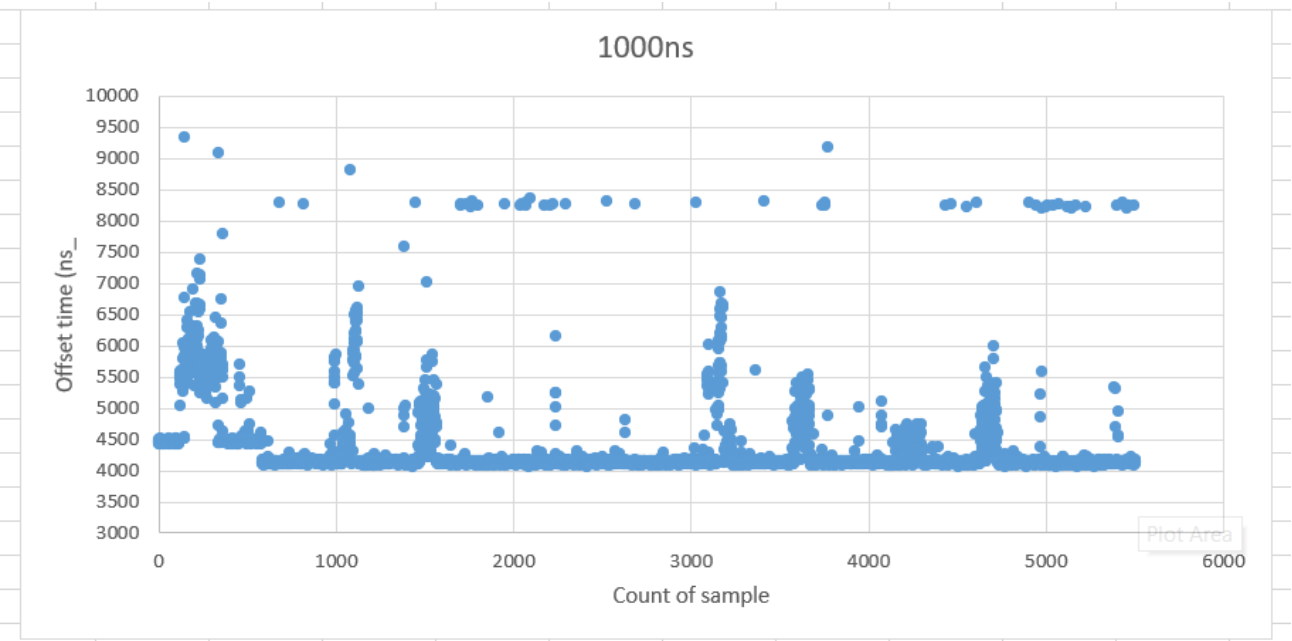


Đồ thị giá trị interval khi chu kỳ X = 10000ns

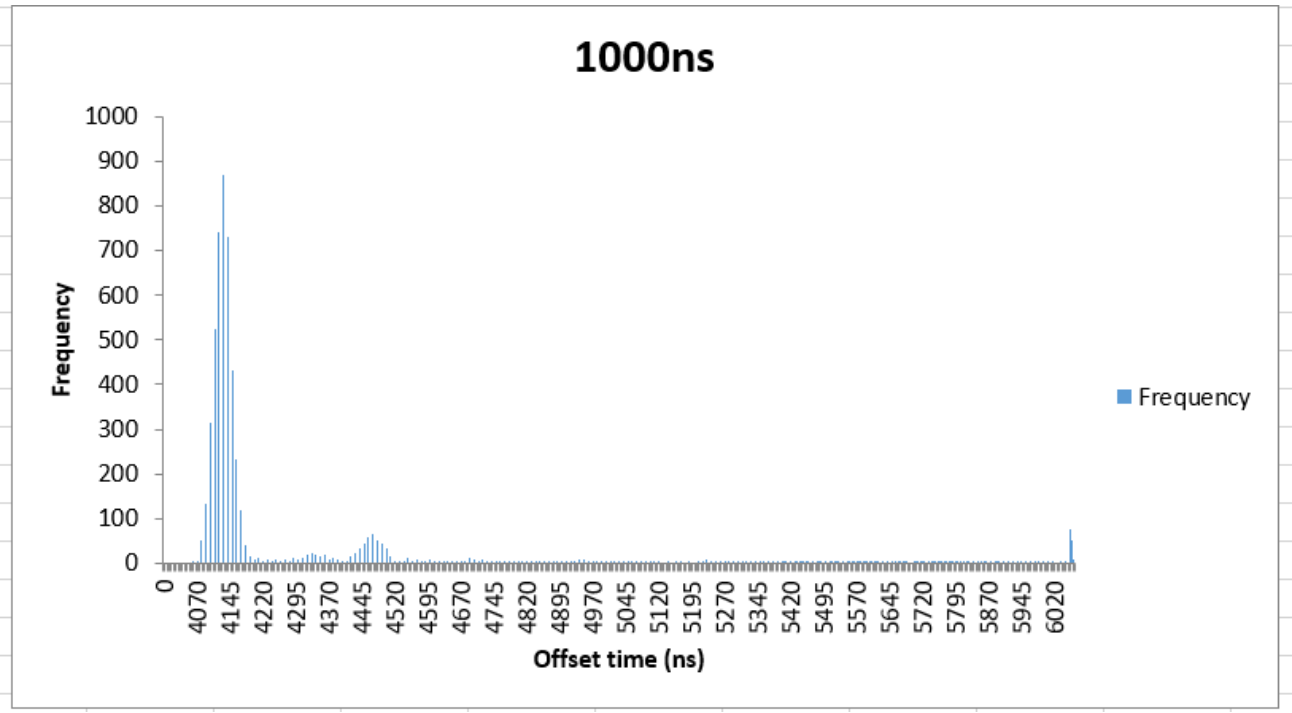


Biểu đồ histogram khi chu kỳ X = 10000ns

2.4 Với chu kỳ X = 1000ns

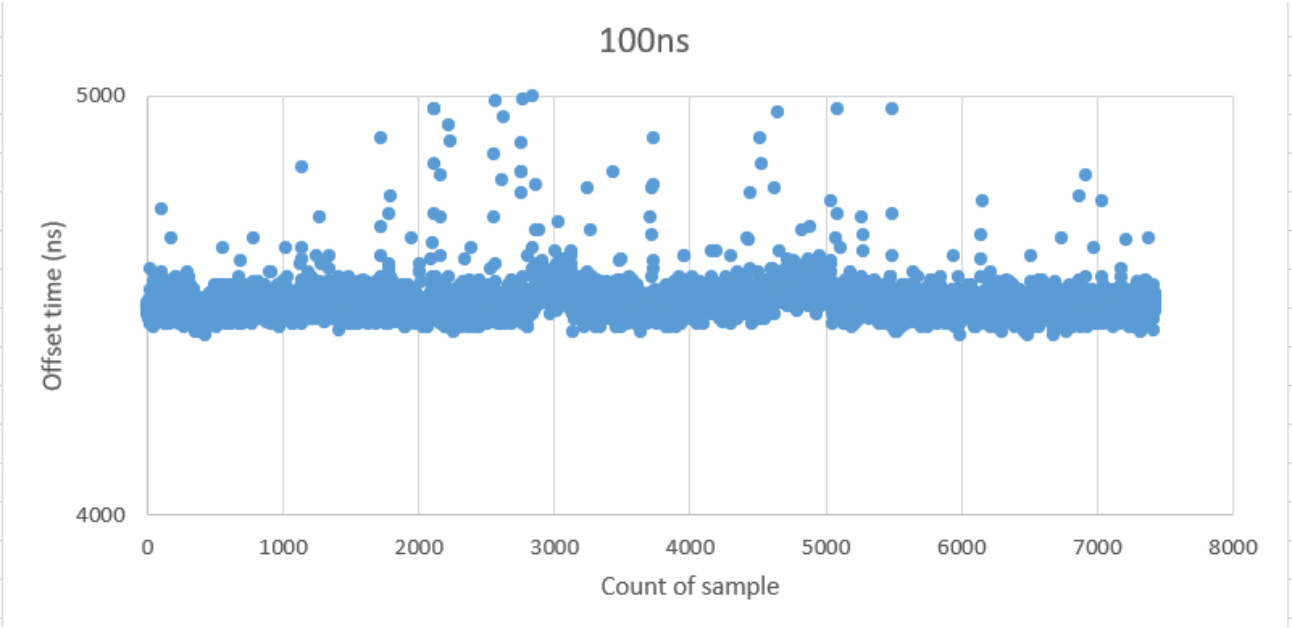


Đồ thị giá trị interval khi chu kỳ X = 1000ns

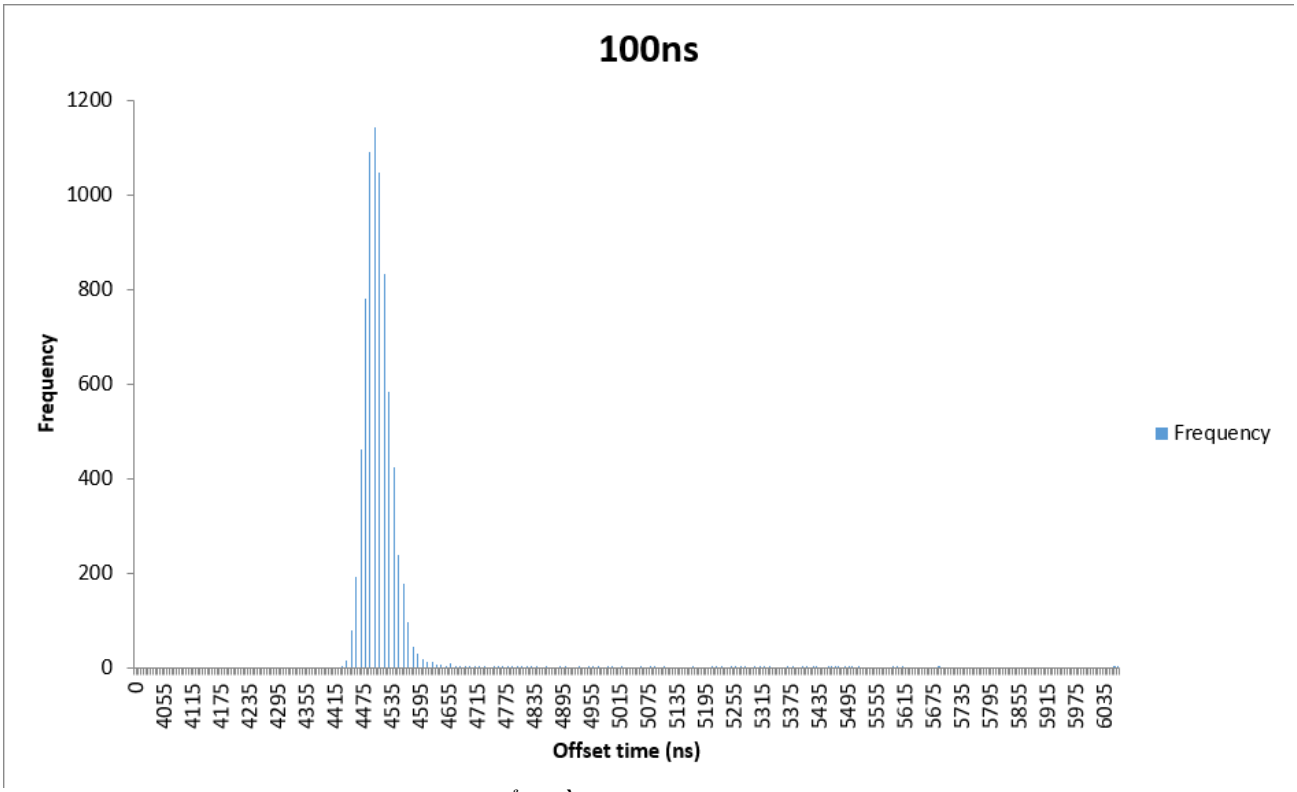


Biểu đồ histogram khi chu kỳ X = 1000ns

2.5 Với chu kỳ X = 100ns



Đồ thị giá trị interval khi chu kỳ X = 100ns



Hình 4.5 2 Biểu đồ histogram khi chu kỳ X = 100ns

2.6 Đánh giá kết quả

Thời gian trung bình của các lần lấy mẫu :

- Với 1000000ns: 1027980 ns.
- Với 100000ns: 107305 ns.
- Với 10000ns: 10535 ns
- Với 1000ns: 4670 ns
- Với 100ns: 4503 ns

Đối với các chu kỳ lấy mẫu 1000000ns, 100000ns, 10000ns, thời gian sai lệch giữa hai lần lấy mẫu tương đối chính xác. Đối với thời gian lấy mẫu 1000ns và 100ns, thời gian offset đã không còn chính xác nữa. Lúc này thời gian sẽ rơi vào khoảng 4000ns (không đạt với yêu cầu đề ra).

III. KẾT LUẬN

- **Khó khăn:**
  - Chưa quen thuộc với Linux Programming, shell script.
  - Kết quả thu được chưa đạt được yêu cầu mong muốn (xuất hiện mẫu có sai lệch lên đến 0.003s mặc dùng chu kì lấy mẫu là 1000000ns).
  - Thuật toán chưa được tối ưu hoàn toàn.
- **Các lỗi đã gặp phải**:
  - Chưa tối ưu thuật toán, dẫn đến sai số thời gian lớn (Khởi tạo liên tục 3 thread trong mỗi vòng lặp), sử dụng hàm nanosleep() để tạo chu kì gấy ra tốn kém tài nguyên của CPU, làm delay chương trình.
- **Khắc phục:**
  - Sử dụng khóa Mutex để thực hiện các thread một cách hiệu quả, thay thế hàm nanosleep() bằng hàm clock_nanosleep() (hàm clock_nanosleep() không làm delay hệ thống, trong lúc thực hiện hàm này, Thread khác có thể sử dụng tài nguyên hệ thống để tính toán offset giữa 2 thời điểm lấy mẫu khác nhau).