

ĐẠI HỌC CÔNG NGHỆ - ĐHQGHN
KHOA ĐIỆN TỬ VIỄN THÔNG



Thực tập thiết kế hệ thống nhúng
Nhóm thực tập Nhúng- Dự án gNode 5G VHT

Báo cáo thực tập

Sinh viên thực hiện:
Đỗ Thái Vũ - 19021540

Hà Nội, năm 2022

CHƯƠNG 1: LINUX PROGRAMMING ASSIGNMENT.....	4
1. Viết chương trình C trên Linux chạy 3 thread SAMPLE, LOGGING, INPUT.....	4
1.1 Thread SAMPLE	4
1.2. Thread INPUT	6
1.3. Thread LOGGING	7
2. Shell script để thay đổi giá trị chu kỳ X	8
3. Thực hiện chạy shell script + chương trình C	8
4. Khảo sát giá trị interval	9
4.1. Với chu kỳ X = 1000000ns	9
Hình 4.1a Biểu đồ histogram khi chu kỳ X = 1000000ns.....	9
Hình 4.1b Biểu đồ value với chu kỳ X =1000000 ns	9
4.2. Với chu kỳ X = 100000ns	10
Hình 4.2a Biểu đồ histogram khi chu kỳ X = 100000ns.....	10
Hình 4.2b Biểu đồ value khi chu kỳ X = 100000ns.....	10
4.3. Với chu kỳ X = 10000ns	11
Hình 4.3a Biểu đồ histogram khi chu kỳ X = 10000ns.....	11
Hình 4.3b Biểu đồ value khi chu kỳ X =10000ns.....	11
4.4. Với chu kỳ X = 1000ns	12
Hình 4.4a Biểu đồ histogram khi chu kỳ X = 1000ns.....	12
Hình 4.4b Biểu đồ value khi chu kỳ X=1000ns.....	12
4.5. Với chu kỳ X = 100ns	13
Hình 4.5a Biểu đồ histogram khi chu kỳ X = 100ns.....	13
Hình 4.5b Biểu đồ value khi chu kỳ X=100ns.....	13
5. Kết luận	14
6. Lý thuyết bổ sung	14
a. Multi Threading	14
b. Mutex	15
c. Timer and sleep	16
CHƯƠNG 2 BẮT BẦU VỚI KIT ORANGE PI.....	18
1. Tìm hiểu về Orange Pi.....	18
Hình 1: Orange Pi 3 Datasheets	18
2. Tiến hành cài File Os vào thẻ nhớ.....	19
Hình 2: Tải Image tương thích cho KIT	19

Hình 3: Sử dụng Etcher để ghi vào thẻ nhớ	20
3. Tiến hành Debug bằng cổng TTL UART	20
Hình 4: Kết nối USB2UART với KIT	20
Hình 5: Thiết lập bằng đường Serial	21
Hình 6: Giao diện CLI của KIT	22
Hình 7: Thiết lập Wifi cho KIT	22
Hình 8: SSH control bằng Ip address	23
4. Tiến hành Remote Desktop	23
Hình 10: Đăng nhập VNC server	24
Hình 11 Yêu cầu đăng nhập từ VNC Viewer	25
Hình 12 Giao diện Armbian được remote desktop của KIT	25
Hình 13: Chạy code và tạo file freq.txt	26
5. Kết luận	27
CHƯƠNG 3: BUILD FIRMWARE CHO PI SỬ DỤNG YOCTO	28
1. Tiến hành cài đặt	28
Hình 1: Build Image	30
2. Tìm file image và lệnh tải lên sdcard	30
3. Lỗi phiên bản Python và cách khắc phục	31
Hình 2: Lỗi phiên bản python	31
Hình 3: Config default python (trên hình là phiên bản python 3.9)	32
4. Nhận xét và đánh giá build Image cho Orange pi bằng Yocto	32
Hình 4: Chọn cấu hình cho KIT	33
5. Kết luận	34

CHƯƠNG 1: LINUX PROGRAMMING ASSIGMENT

1. Viết chương trình C trên Linux chạy 3 thread SAMPLE, LOGGING, INPUT.

1.1 Thread SAMPLE

- Thread **SAMPLE** thực hiện vô hạn lần nhiệm vụ sau với chu kì **X** ns.
Nhiệm vụ là đọc thời gian hệ thống hiện tại (chính xác đến đơn vị ns) vào biến **T**.

```
void *getTime(void *args )
{
    clock_gettime(CLOCK_REALTIME,&request1);
    while(check_loop == 1)
    {
        clock_gettime(CLOCK_REALTIME,&tp);
        long temp;
        if(request1.tv_nsec + freq > 1000000000)
        {
            temp = request1.tv_nsec;
            request1.tv_nsec = temp + freq - 1000000000;;
            request1.tv_sec +=1;
            if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME, &request1,NULL) != 0)
            {
                check_loop = 0;
            }
            else
            {
                check_loop = 1;
            }
        }
    }
}
```

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

Lưu ý :

- Khi tín hiệu được phân phối ở tốc độ cao thì nanosleep có thể dẫn đến sự thiếu chính xác lớn trong cả 1 quy trình (do thời gian sleep), để giải quyết vấn đề, lần call đầu tiên dùng hàm clock_gettime() để truy xuất thời gian, sau đó thêm thời lượng mong muốn vào. Cuối cùng gọi clock_nanosleep() với TIMER_ABSTIME flag.

1.2. Thread INPUT

- Thread **INPUT** kiểm tra file “freq.txt” để xác định chu kỳ **X** (của thread **SAMPLE**) có bị thay đổi không?, nếu có thay đổi thì cập nhật lại chu kỳ X. Người dùng có thể echo giá trị chu kỳ X mong muốn vào file “freq.txt” để thread **INPUT** cập nhật lại **X**.

```
void *getFreq(void *args)
{
    while(1)
    {
        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

- Thread **LOGGING** chờ khi biến **T** được cập nhật mới, thì ghi giá trị biến **T** và giá trị **interval** (offset giữa biến **T** hiện tại và biến **T** của lần ghi trước) ra file có tên "time_and_interval.txt".

```
void *save_time(void *args)
{
    while(1)
    {
        FILE *file;
        file = fopen("interval.txt","a+");
        long diff_sec = ((long) 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. Shell script để thay đổi giá trị chu kỳ X

Trong phần này, ta sẽ viết shell script để thay đổi lại giá trị chu kỳ X trong file “freq.txt” sau mỗi 1 phút. Và các giá trị X lần lượt được ghi như sau: 1000000 ns, 100000 ns, 10000 ns, 1000 ns, 100ns

File script.sh

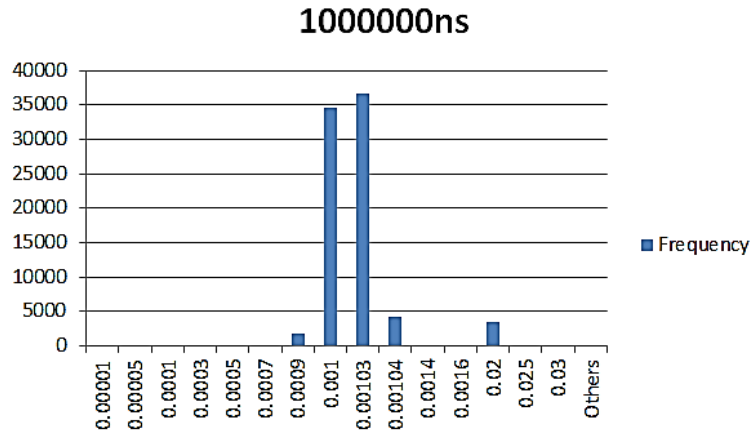
```
#!/bin/sh
echo "1000000">freq.txt
timeout 60s ./b1
echo "100000">freq.txt
timeout 60s ./b1
echo "10000">freq.txt
timeout 60s ./b1
echo "1000">freq.txt
timeout 60s ./b1
echo "100">freq.txt
timeout 60s ./b1
```

3. Thực hiện chạy shell script + chương trình C

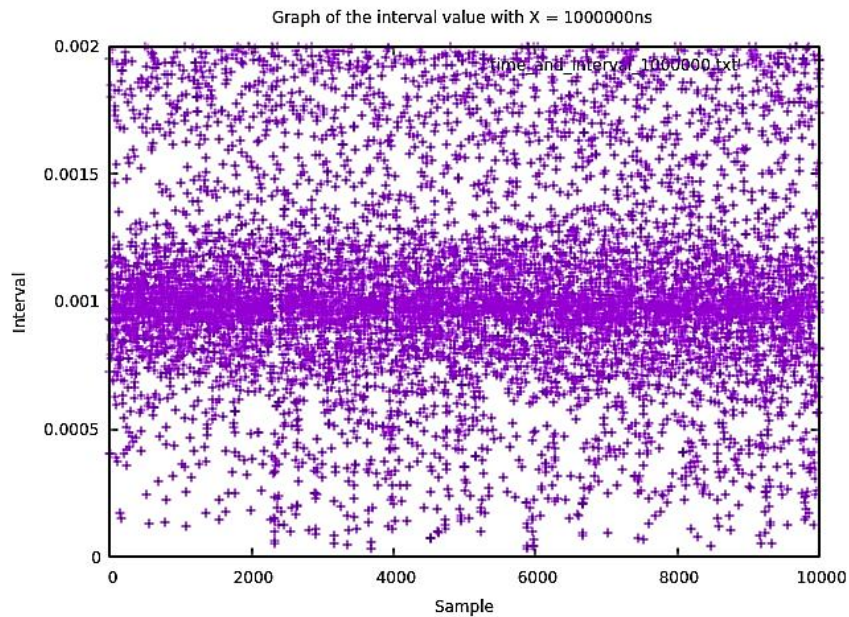
Trong phần này, ta sẽ cho chạy chương trình C và shell script trong vòng 5 phút, và sau đó dừng chương trình C (theo lệnh trên thì chương trình mỗi freq chỉ chạy trong 60 s).

4. Khảo sát giá trị interval

4.1. Với chu kỳ $X = 1000000\text{ns}$



Hình 4.1a Biểu đồ histogram khi chu kỳ $X = 1000000\text{ns}$

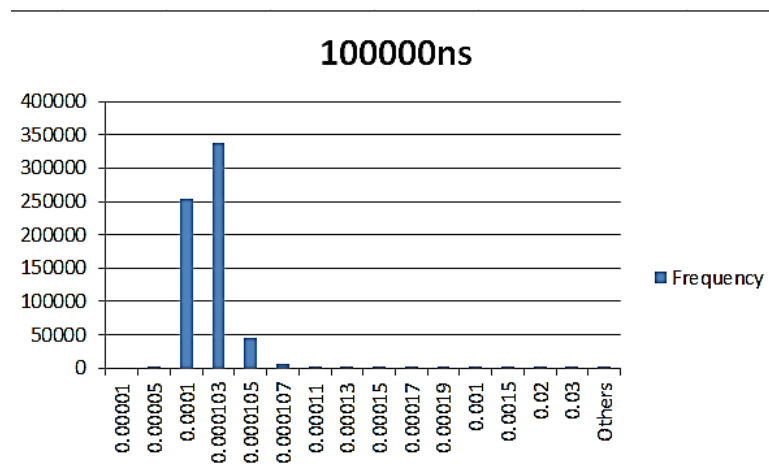


Hình 4.1b Biểu đồ value với chu kỳ $X = 1000000\text{ ns}$

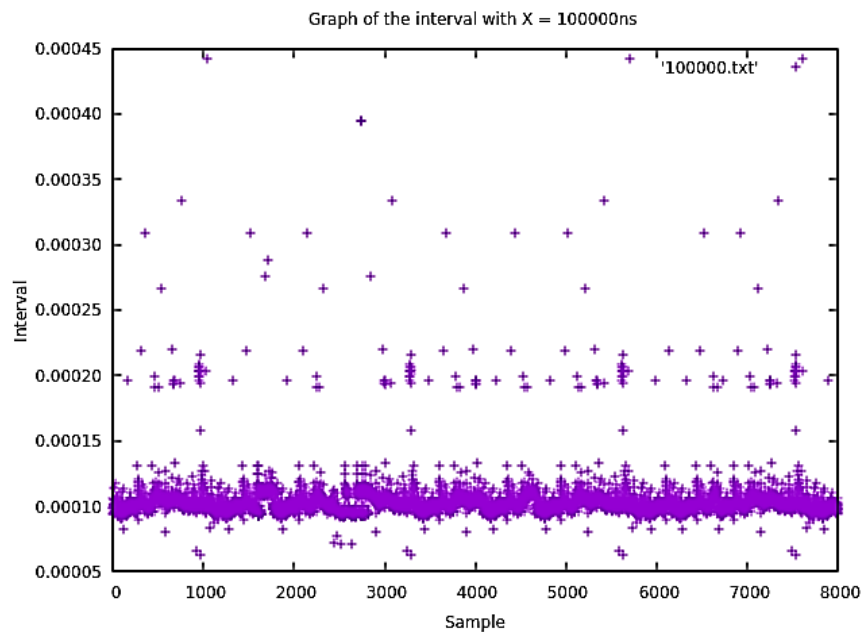
Nhận xét : ta có thể thấy các interval chủ yếu ở 0.001s và 0.00101s, chủ yếu trong khoảng $[0.001-0.002]$.

4.2. Với chu kỳ $X = 100000\text{ns}$

4.2. Với chu kỳ $X = 100000\text{ns}$



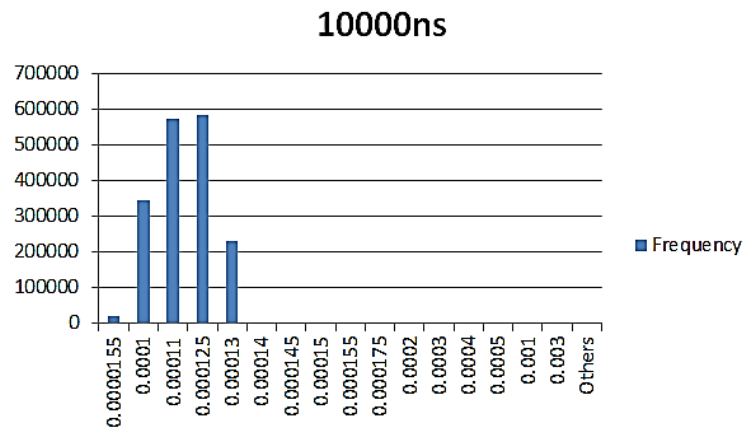
Hình 4.2a Biểu đồ histogram khi chu kỳ $X = 100000\text{ns}$



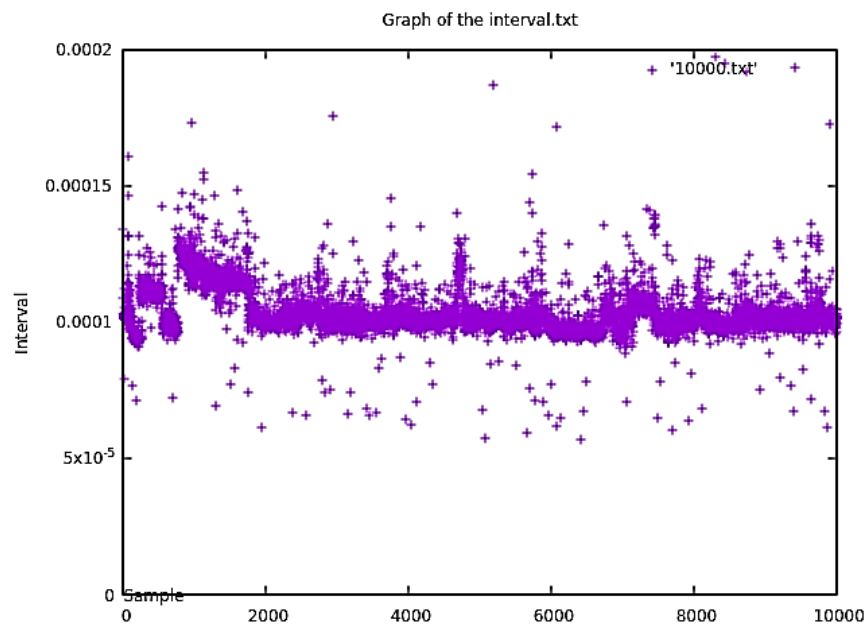
Hình 4.2b Biểu đồ value khi chu kỳ $X = 100000\text{ns}$

Nhận xét : Các interval xuất hiện nhiều ở offset 0.0001s và 0.000101s , chủ yếu nằm trong khoảng $[0.0001-0.00012]$

4.3. Với chu kỳ $X = 10000\text{ns}$



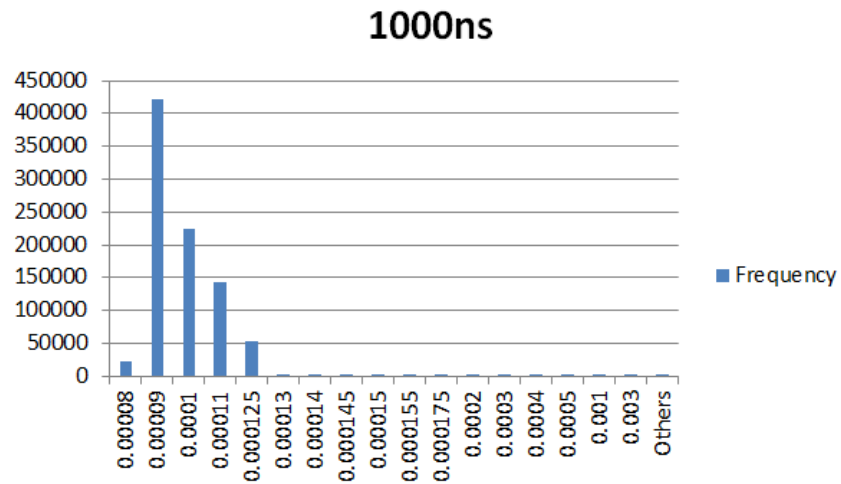
Hình 4.3a Biểu đồ histogram khi chu kỳ $X = 10000\text{ns}$



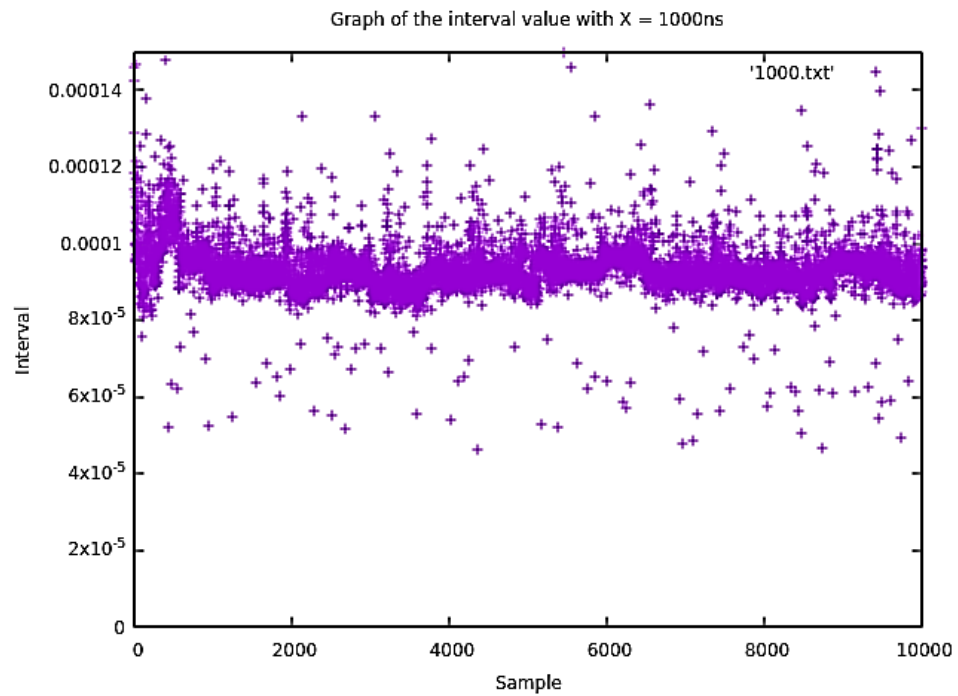
Hình 4.3b Biểu đồ value khi chu kỳ $X = 10000\text{ns}$

Nhận xét: Các Interval xuất hiện nhiều ở offset 0.0000125s, 0.000012, 0.00001s

4.4. Với chu kỳ $X = 1000\text{ns}$

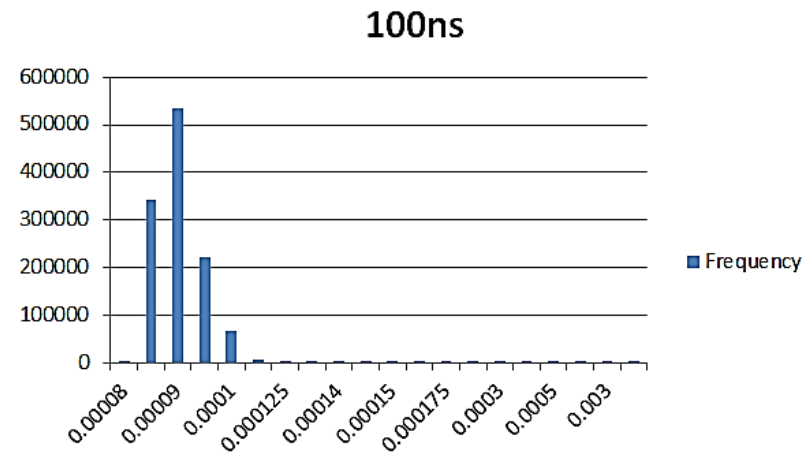


Hình 4.4a Biểu đồ histogram khi chu kỳ $X = 1000\text{ns}$

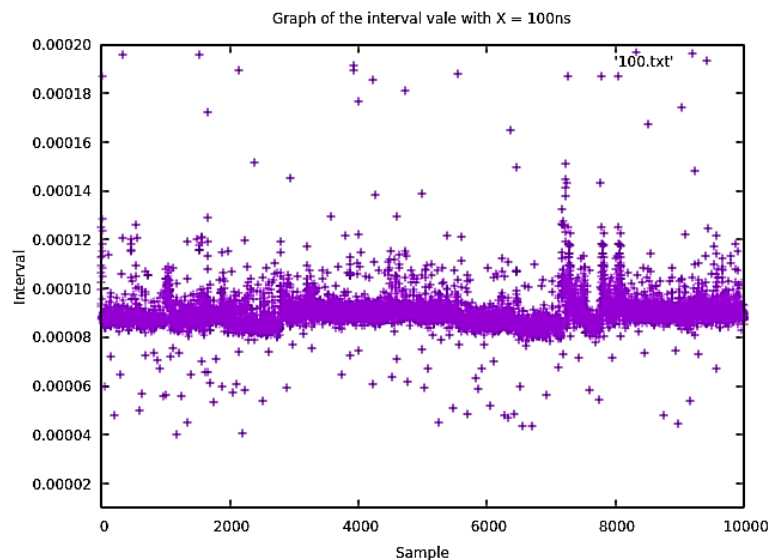


Hình 4.4b Biểu đồ value khi chu kỳ $X=1000\text{ns}$

4.5. Với chu kỳ $X = 100\text{ns}$



Hình 4.5a Biểu đồ histogram khi chu kỳ $X = 100\text{ns}$



Hình 4.5b Biểu đồ value khi chu kỳ $X=100\text{ns}$

Nhận xét chung :

- Độ chính xác của thuật toán chỉ đúng với chu kỳ lớn (100000ns) còn với chu kỳ nhỏ cần tối ưu thuật toán thêm.
- Khi chu kỳ càng lớn thì giá trị interval giữa T hiện tại và T của lần ghi trước càng lớn.
- Trong các lần lấy mẫu đều xuất hiện thời gian lớn (0.003s)

5. Kết luận

- Tiến độ công việc: Đã hoàn thành trước deadline (15/7/2022)
- Khó khăn gặp phải :
 - + Shell Script trên Hệ ĐH Linux
 - + Kiến thức về C như Threading , Set time
 - + Chưa tối ưu được thuật toán một cách tốt nhất: Giá trị interval thu được chưa đạt như mong muốn;

6. Lý thuyết bổ sung

a. Multi Threading

Giống như các quy trình, các luồng là một cơ chế cho phép ứng dụng thực hiện nhiều nhiệm vụ đồng thời. Một quy trình duy nhất có thể chứa nhiều luồng. Tất cả các luồng này đều thực thi độc lập giống nhau và tất cả chúng đều chia sẻ cùng một bộ nhớ chung, bao gồm cả dữ liệu được khởi tạo, dữ liệu chưa khởi tạo và phân đoạn đóng. (Quy trình UNIX truyền thống chỉ đơn giản là một quy trình đặc biệt trường hợp của một quy trình đa luồng; nó là một quá trình chỉ chứa một chuỗi.)

- Chia tiến trình dùng hàm `fork()` nhưng có nhược điểm:
 - + Khó để chia sẻ thông tin giữa các processes. Bởi vì giữa parent and child ko share memory
 - + Quá trình tạo `fork()` tương đối tốn kém
- Threads có thể giải quyết dc vấn đề đó:
 - + Có thể chia sẻ thông tin giữa các thread (luồng) nhanh và dễ.
 - + Thread creation nhanh hơn process creation-typically 10 lần or better.
- Bên cạnh global memory (bộ nhớ chung), thread còn share 1 số thuộc tính chung cho 1 process, thay vì cụ thể cho 1 thread).
Các thuộc tính này bao gồm (663 pdf)
- Background Details of the Pthreads API

Data type	Description
pthread_t	Thread identifier
pthread_mutex_t	Mutex
pthread_mutexattr_t	Mutex attributes object
pthread_cond_t	Condition variable
pthread_condattr_t	Condition variable attributes object
pthread_key_t	Key for thread-specific data
pthread_once_t	One-time initialization control context
pthread_attr_t	Thread attributes object

- Các lệnh thường được dùng là :

```
+ pthread_create(&thread, NULL, func, &arg)
+ pthread_exit(void *retval)
+ pthread_cancel()
+ pthread_self(void)
+ pthread_join()
```

b. Mutex

+ Mutex được hiểu là đối tượng (hoặc cờ hiệu) mang 2 trạng thái là đang được sử dụng và chưa được sử dụng (trạng thái sẵn sàng)

+ Khai báo mutex cấp tốc :

```
pthread_mutex_t a_mutex = PTHREAD_MUTEX_INITIALIZER;
```

Lưu ý : Đối tượng mutex này không thể bị khóa 2 lần bởi cùng 1 luồng.

Trong luồng nếu bạn đã gọi hàm khóa mutex này và thực hiện khóa mutex 1 lần nữa, bạn sẽ rơi vào trạng thái khóa chết (deadlock).

Có thể dùng

```
pthread_mutex_init(pthread_mutex_t * mutex, const pthread_mutexattr_t *
mutexattr);
```

+ Khóa và tháo khóa

```
int rc = pthread_mutex_lock( & a_mutex);
```

rc = 1 thì có lỗi ; rc = 0 thì khóa lại thành công.

Mở khóa để trả lại tài nguyên cho luồng khác

```
int rc = pthread_mutex_unlock( & a_mutex);
```

rc = 1 thì có lỗi ; rc = 0 thì mở khóa lại thành công.

+ Hủy mutex (nên hủy sau khi dùng xong)

```
rc = pthread_mutex_destroy (& a_mutex)
```

c. Timer and sleep

- Các lệnh gọi hệ thống chính trong API đồng hồ POSIX là:

+ clock_gettime () : lấy giá trị hiện tại của đồng hồ;

+ clock_getres () : trả về độ phân giải của đồng hồ;

+ clock_settime () : cập nhật đồng hồ.

- Hàm nanosleep() để tạo 1 thời gian “ngủ” cho chương trình. Cần sử dụng thư viện unistd.h

- Khi tín hiệu được phân phối ở tốc độ cao thì nanosleep có thể dẫn đến sự thiếu chính xác lớn trong cả 1 quy trình (do thời gian sleep), để giải quyết vấn đề, lần call đầu tiên dùng hàm clock_gettime() để truy xuất thời gian, sau đó thêm thời lượng mong muốn vào. Cuối cùng gọi clock_nanosleep() với TIMER_ABSTIME flag.

- Một số struct quan trọng :

```
struct timespec {  
    time_t tv_sec; /* Seconds ('time_t' is an integer type) */  
    long tv_nsec; /* Nanoseconds */  
};
```



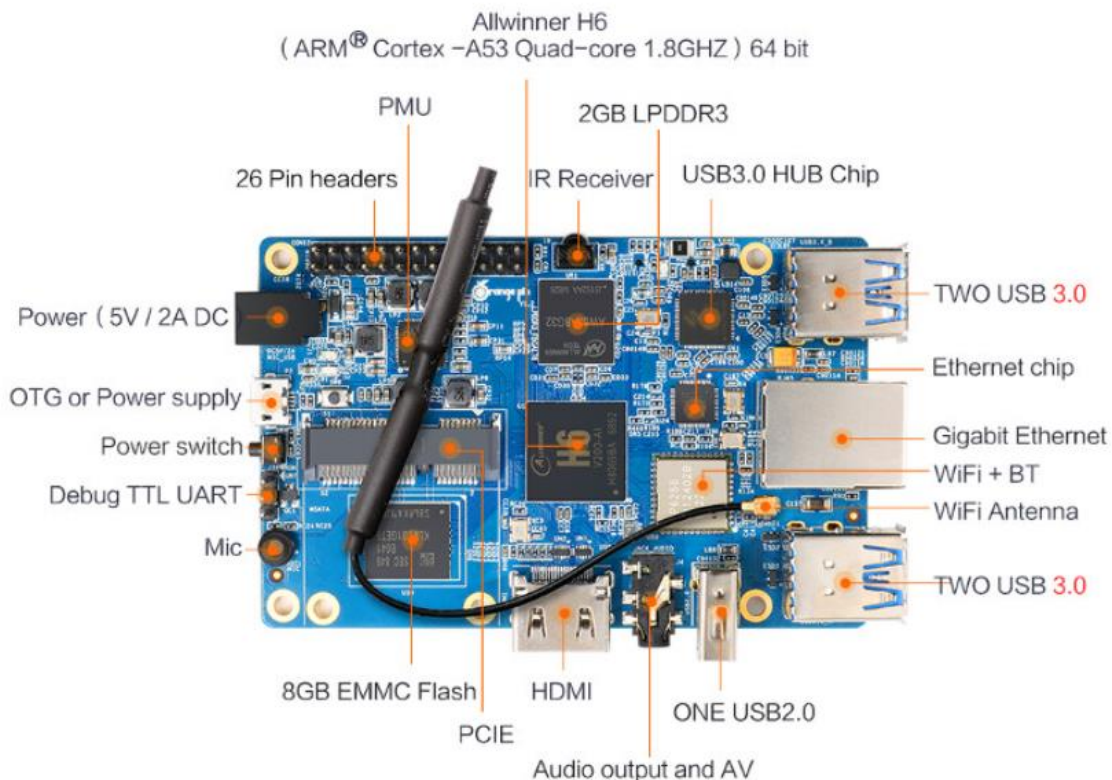
```
struct timespec request;
/* Retrieve current value of CLOCK_REALTIME clock Page 538 */
if (clock_gettime(CLOCK_REALTIME, &request) == -1)
    errExit("clock_gettime");
request.tv_sec += 20; /* Sleep for 20 seconds from now */
s = clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &request,
NULL);
if (s != 0) {
    if (s == EINTR)
        printf("Interrupted by signal handler\n");
    else
        errExitEN(s, "clock_nanosleep");
}
```

CHƯƠNG 2 BẮT BẦU VỚI KIT ORANGE PI

1. Tìm hiểu về Orange Pi

- Orange pi là một loại máy tính nhúng , có nhiều phiên bản khác nhau đáp ứng các loại cấu hình khác nhau của người dùng.
- Có họ hàng với các loại pi phổ biến khác như raspberry pi, pine pi, ...
- Ở đây, em dùng orange pi 3 để làm việc.
- Orange pi 3 datasheet

WiFi 5, có tới tận 4 cổng USB 3.0 ports, HDMI 2.0a, và đặc biệt là 1 khe cắm mPCIe.

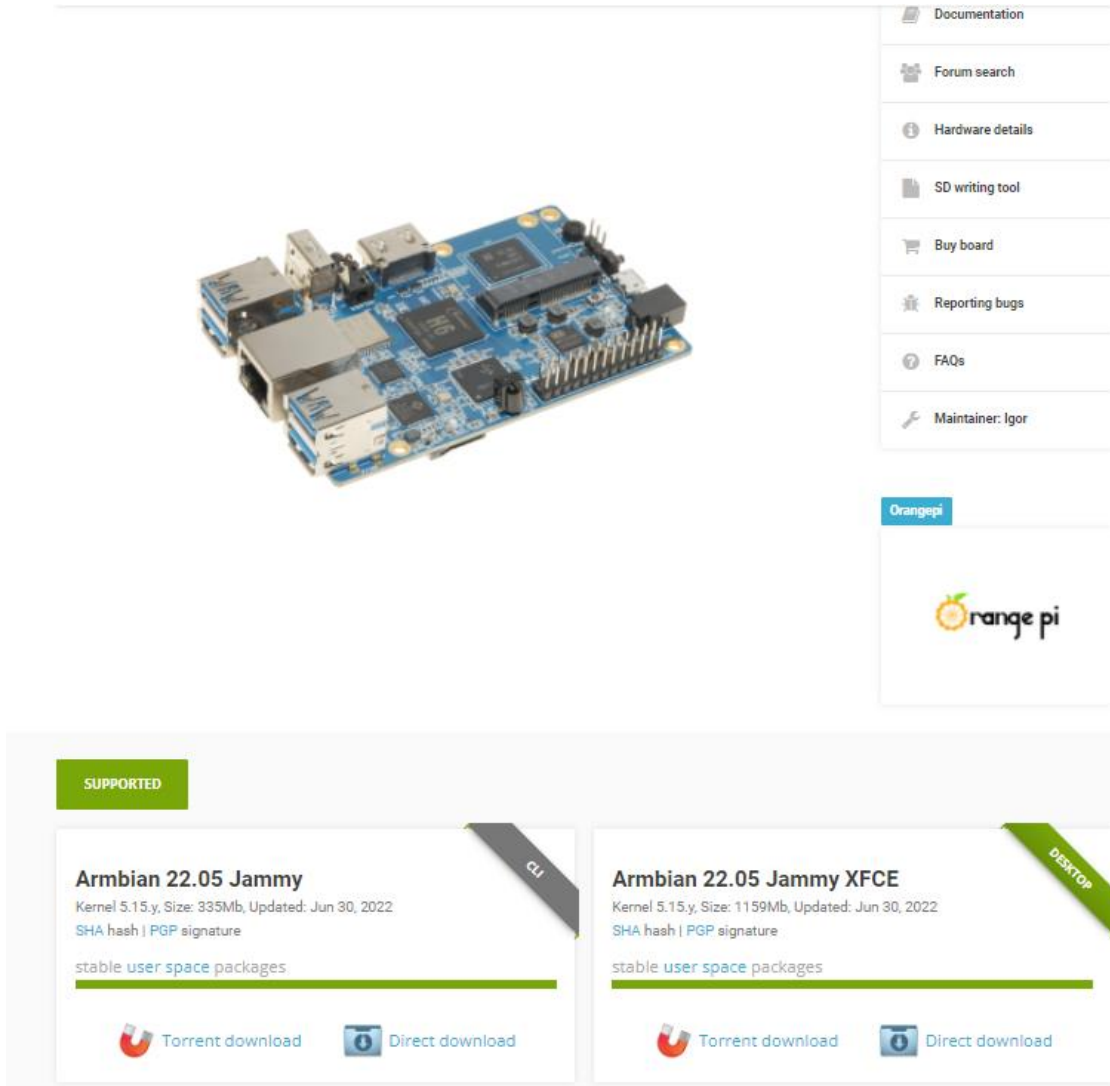


Hình 1: Orange Pi 3 Datasheets

- Đáng chú ý :
- + Nguồn 5V-3A nếu vượt quá thì kit sẽ cháy.
- + Chip H6 arm-64 bit thuộc RISC
- + Ram 2GB.
- + Bộ nhớ trong 8GB EMMC Flash.

2. Tiến hành cài File Os vào thẻ nhớ

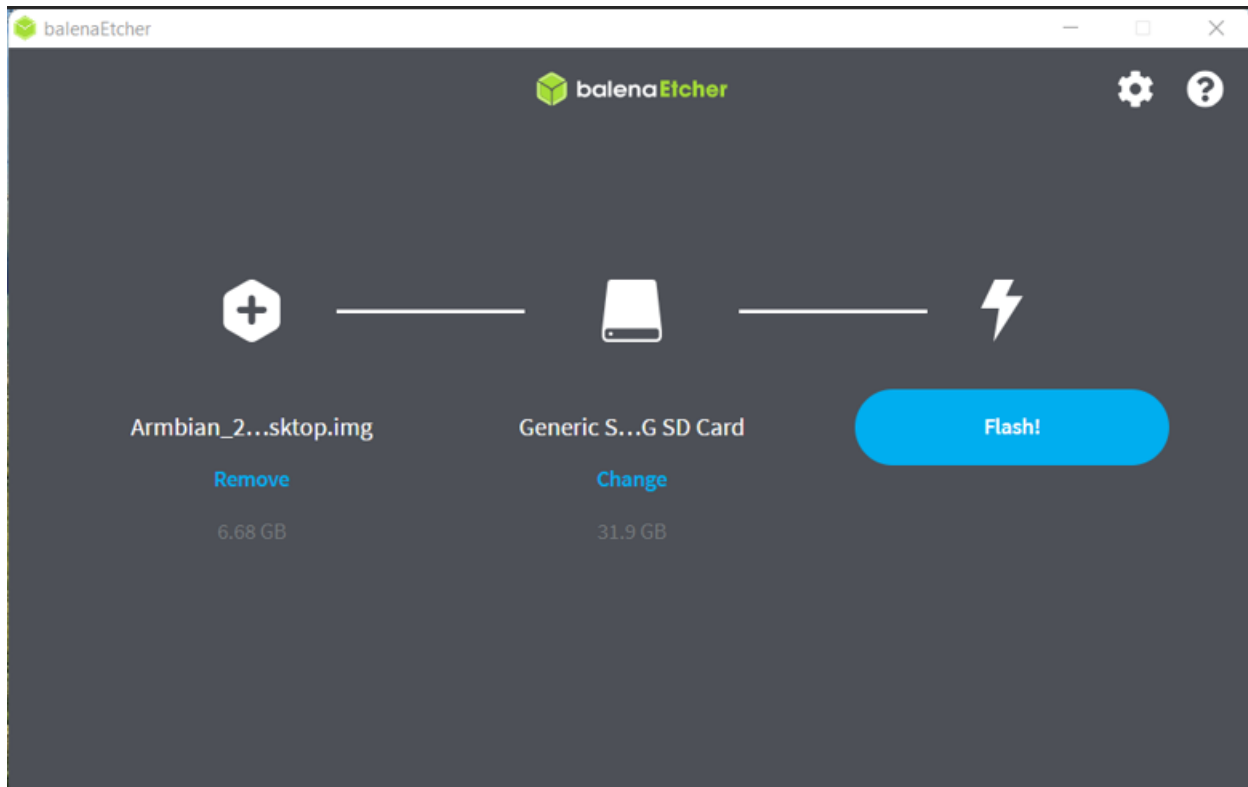
- Vào trang chủ của armbian tải file img tương thích về



Hình 2: Tải Image tương thích cho KIT

- Ở đây, chọn bản Desktop (khuyến nghị)

-Sau khi giải nén File Image ta tiến hành Flash vào Thẻ nhớ

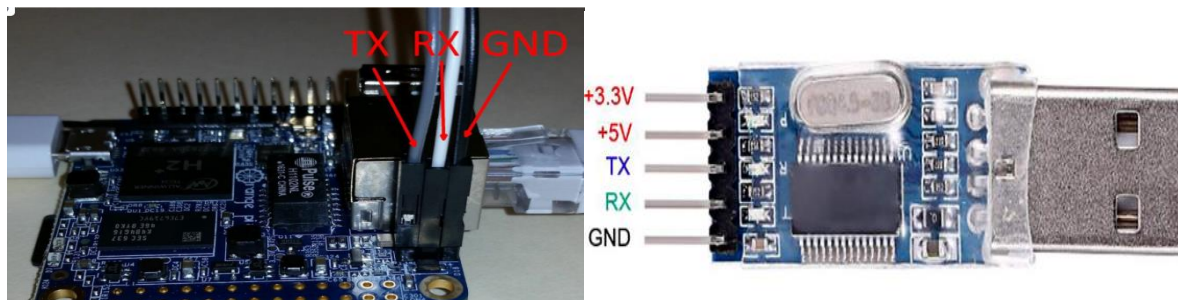


Hình 3: Sử dụng Etcher để ghi vào thẻ nhớ

- Sau khi Flash xong thì rút thẻ nhớ ra khỏi máy tính, cắm thẻ nhớ vào KIT, ngoài ra sử dụng USB2UART để tiến hành Debug TTL UART.

3.Tiến hành Debug bằng cổng TTL UART

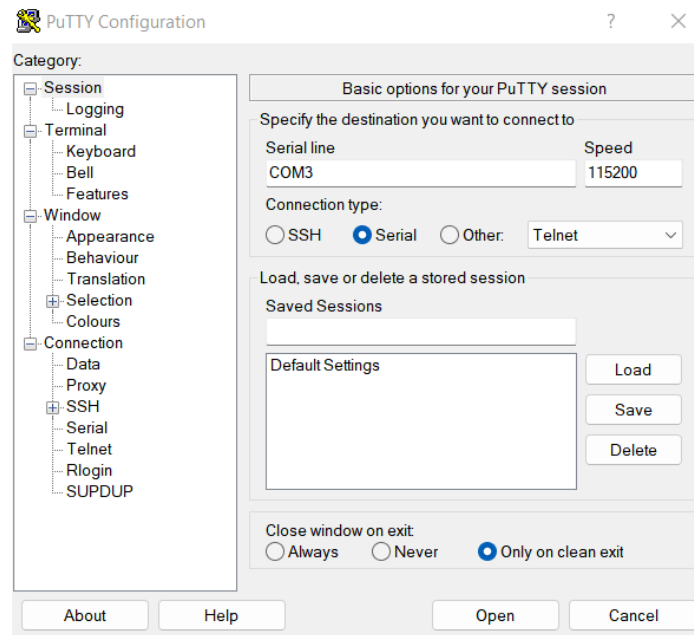
Tiến hành nối USB2UART với cổng TTL UART của bo mạch



Hình 4: Kết nối USB2UART với KIT

- +2 Chân GND nối với nhau
- +RX của chân UART nối với TX của KIT
- +TX của UART nối với RX của KIT

-Tiến hành debug sử dụng phần mềm PuTTY



Hình 5: Thiết lập bằng đường Serial

- Sau đó cài đặt tài khoản và mật khẩu mới ta được:

```
HD 0M3 - PuTTY
Armbian 22.05.3 Jammy ttys0
orangeip3 login: nmtui-connect
Password:

Login incorrect
orangeip3 login: dothaivu
Password:

  0013
Welcome to Armbian 22.05.3 Jammy with Linux 5.15.48-sunxi64

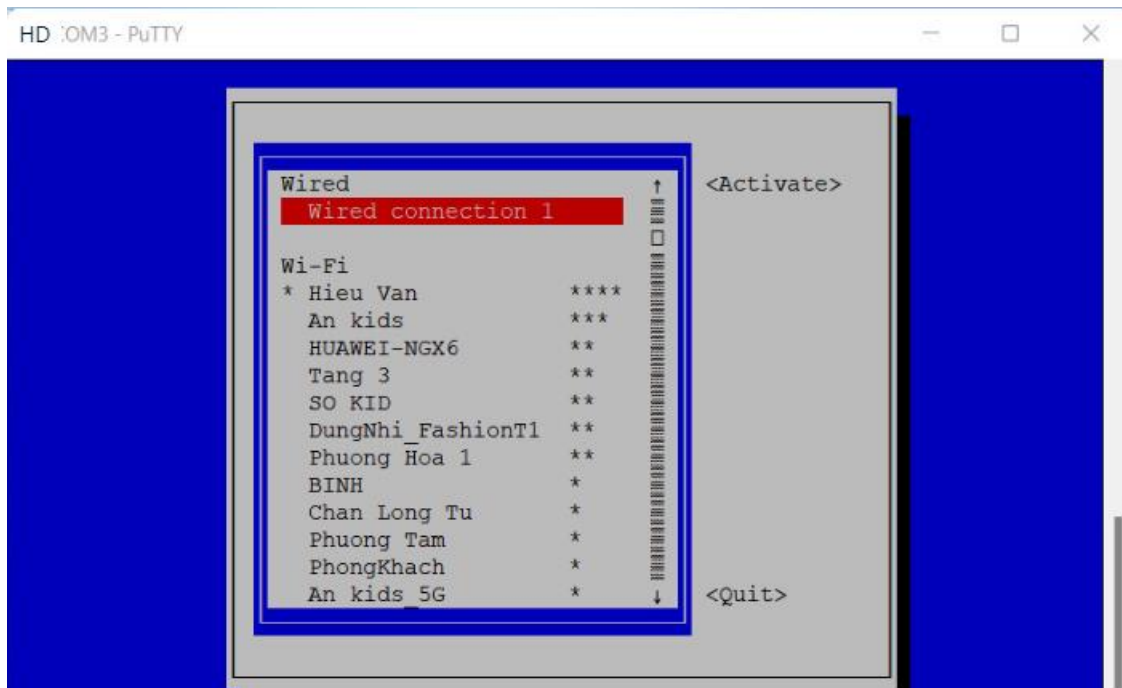
System load:  7%          Up time:      1 min
Memory usage: 7% of 1.94G IP:          192.168.1.135
CPU temp:    44°C        Usage of /:  5% of 29G

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Hình 6: Giao diện CLI của KIT

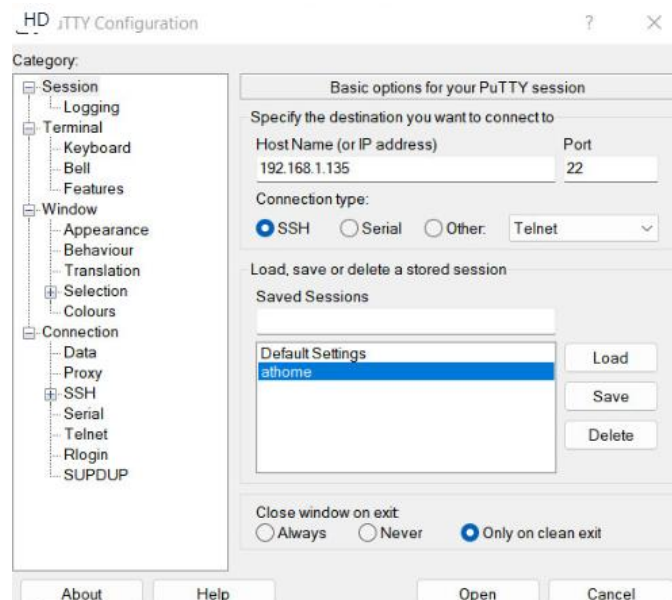
- Dùng lệnh để thiết lập Wifi :

nmtui-connect

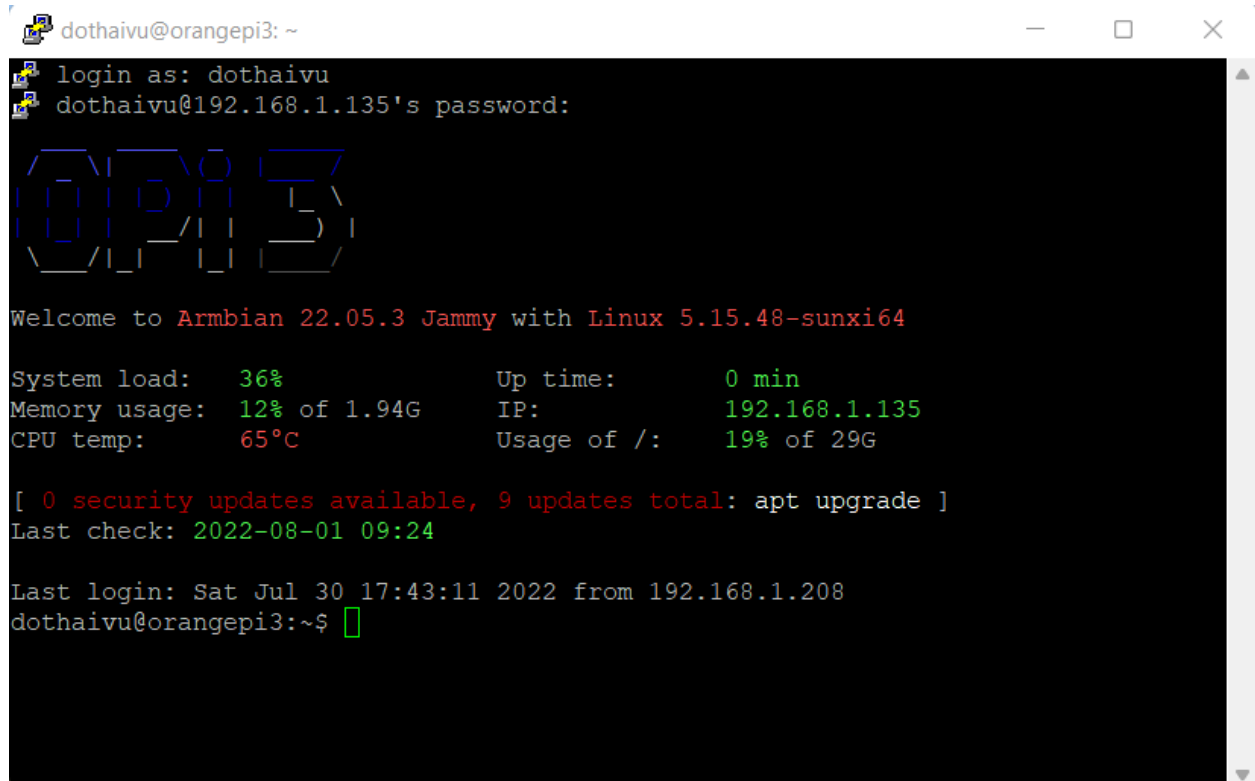


Hình 7: Thiết lập Wifi cho KIT

- Sử dụng lệnh **ifconfig** để xem địa chỉ Ip của KIT, việc này giúp chúng ta có điều khiển thông qua ssh.



Hình 8: SSH control bằng Ip address



```
dothaivu@orangepi3: ~  
login as: dothaivu  
dothaivu@192.168.1.135's password:  
  
Welcome to Armbian 22.05.3 Jammy with Linux 5.15.48-sunxi64  
  
System load: 36%      Up time: 0 min  
Memory usage: 12% of 1.94G  IP: 192.168.1.135  
CPU temp: 65°C      Usage of /: 19% of 29G  
  
[ 0 security updates available, 9 updates total: apt upgrade ]  
Last check: 2022-08-01 09:24  
  
Last login: Sat Jul 30 17:43:11 2022 from 192.168.1.208  
dothaivu@orangepi3:~$
```

Hình 9: SSH thành công.

4. Tiến hành Remote Desktop

- Sử dụng lệnh để cài đặt x11vnc :

```
sudo apt-get install xorg x11-common xfce4 x11vnc xfce4-goodies  
gnome-icon-theme -y
```

- Sử dụng lệnh để thay đổi mật khẩu của vnc-server và lưu địa chỉ lưu trữ của password đó.

```
x11vnc - -storepasswd
```

- Sử dụng lệnh để tiến hành config lại vnc server

```
sudo dpkg-reconfigure x11-common
```

- Sử dụng lệnh để chỉnh sửa file vnc.sh

```
sudo nano vnc.sh
```

+ Nội dung

```
#!/bin/bash
```

```
/bin/su dothaivu -c "/usr/bin/screen -dmS xfce4 startxfce4"
```

```
sleep 5
```

```
x11vnc -dontdisconnect -display :0 -auth /var/run/lightdm/root/:0 -  
nottruecolor -noxfixes -shared -forever -rfbport 5900 -bg -o  
/var/log/x11vnc.log -rfbauth /home/dothaivu/.vnc/passwd
```

+ Khởi tạo bằng lệnh mỗi khi cần sử dụng :

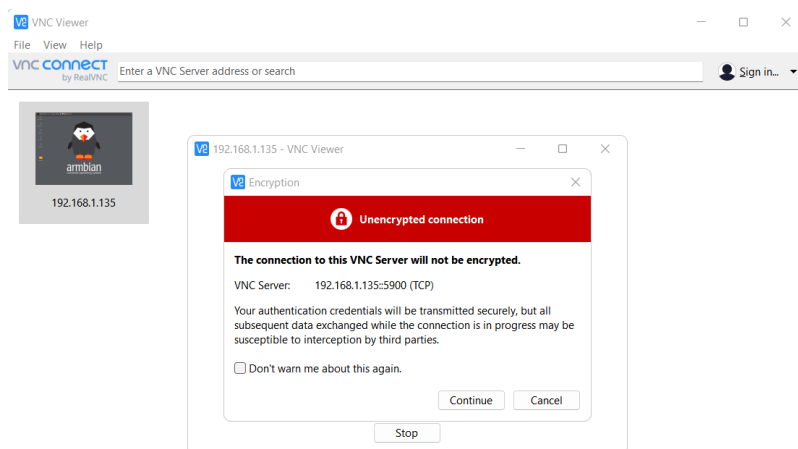
```
sudo bash vnc.sh
```

```
Last login: Sat Jul 30 17:43:11 2022 from 192.168.1.208  
dothaivu@orangeip3:~$ sudo bash vnc.sh  
[sudo] password for dothaivu:  
PORT=5900  
dothaivu@orangeip3:~$
```

Hình 10: Đăng nhập VNC server

Thành công sẽ hiện PORT=5900.

- Để remote desktop, sử dụng phần mềm VNC Viewer



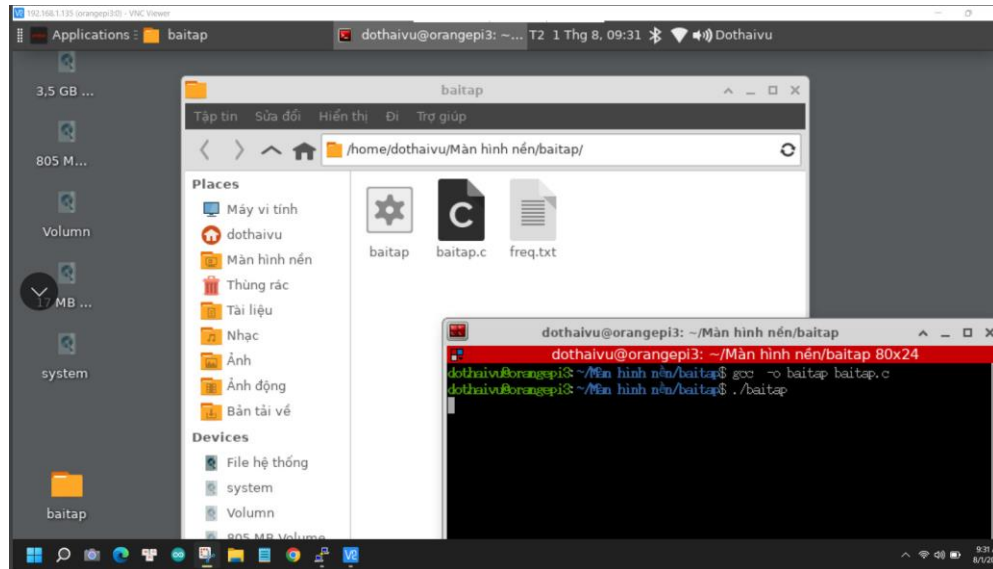
Hình 11 Yêu cầu đăng nhập từ VNC Viewer

Kết quả:



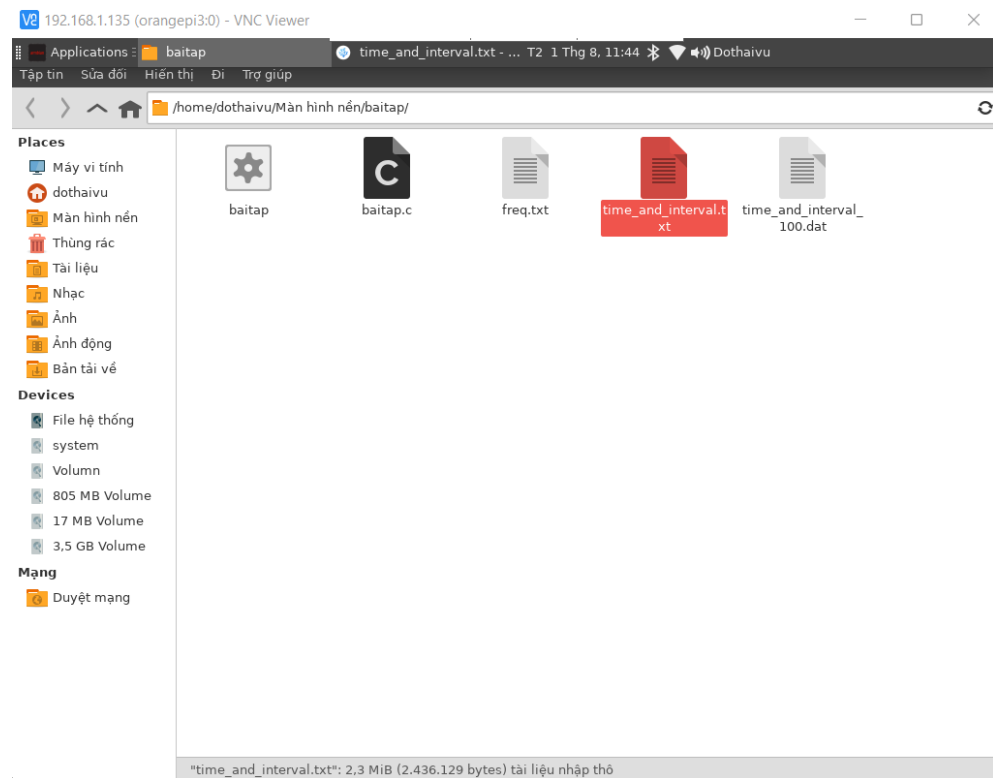
Hình 12 Giao diện Armbian được remote desktop của KIT

- Chạy bài code tuần 1:



Hình 13: Chạy code và tạo file freq.txt

-Kết quả thu được:



```
home > dothaivu > Màn hình nền > baitap > time_and_interval.txt
1 |
2 | 1659321054.338643718
3 | 0.004165988
4 | 0.001180527
5 | 0.000865362
6 | 0.000180665
7 | 0.001548022
8 | 0.001017445
9 | 0.000430495
10 | 0.000330037
11 | 0.000161748
12 | 0.000227622
13 | 0.000167040
14 | 0.000165081
15 | 0.000083541
16 | 0.000721324
17 | 0.000182540
18 | 0.000174498
19 | 0.000193123
20 | 0.000162915
21 | 0.000054499
22 | 0.000172248
23 | 0.000101296
```

-Nhận xét :

+ Kết quả thu được khá giống tương đối với kết quả làm trên Hệ điều hành Ubuntu trong tuần trước.

+ Chính xác ở mức 100000 ns trở lên, nếu ở mức 10000ns thì cần tối ưu thuật toán.

5. Kết luận

- Đã hoàn thành công việc.

+ Tải bản release OS lên kit thành công.

+ Chạy thử code chương 1 thành công.

- Cài đặt thành công các app liên quan đến làm việc KIT như : Etcher, PuTTY, VNC Viewer, ..

- Khó khăn:

+ Tài liệu tìm kiếm liên quan đến Orange Pi ít

+ Ngoài ra do cấu hình Chip, RAM, khác nhau của từng loại KIT Orange nên một số dữ liệu cần chính xác và đôi khi có lỗi phiên bản hệ điều hành hoặc lỗi phiên bản KIT.

CHƯƠNG 3: BUILD FIRMWARE CHO PI SỬ DỤNG YOCTO

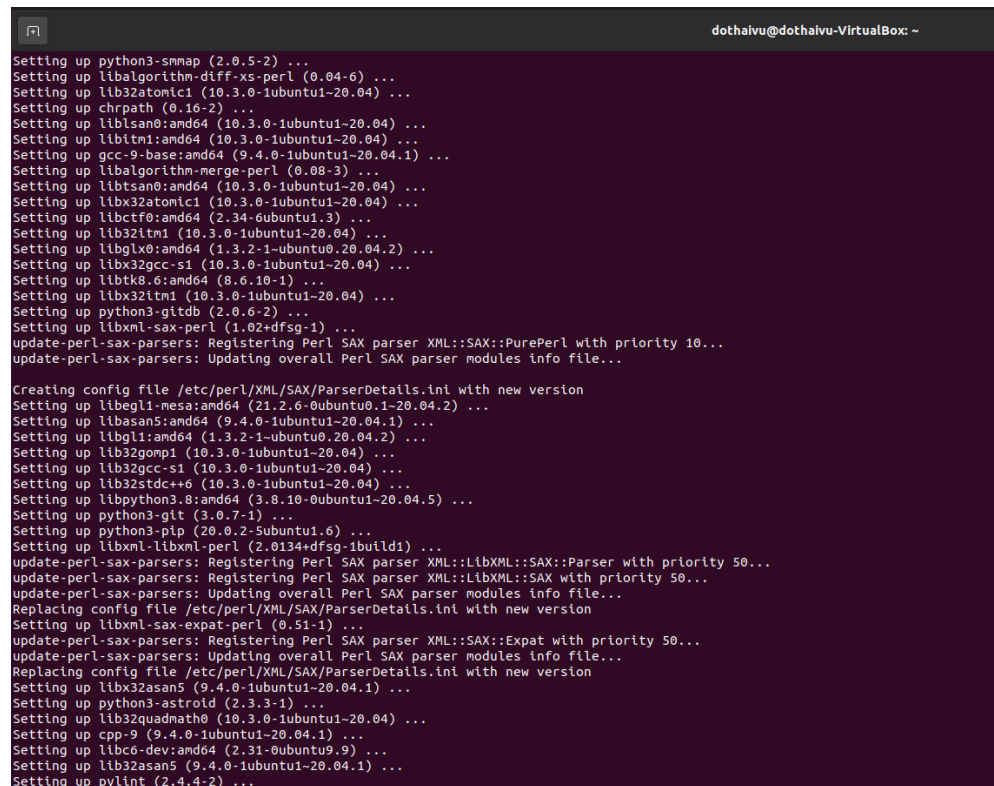
Lưu ý : Do Orange pi chưa được Yocto Project hỗ trợ nên phần Git Clone được lấy từ Git Hub (ko rõ nguồn). Họ lưu ý là chỉ chạy được với các loại Kit Orange Pi Zero, Orange Pi PC, Orange Pi PC Plus.

Do vậy, em sẽ tiến hành Build cho Orange Pi Zero.

1. Tiến hành cài đặt

- Để build Image cho Orange Pi sử dụng Yocto, cần cài các app mà Yocto phụ thuộc.

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
  build-essential chrpath socat cpio python3 python3-pip python3-pexpect \
  xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa \
  libssl1.2-dev pylint3 xterm
```



```
dothaivu@dothaivu-VirtualBox: ~
Setting up python3-smmap (2.0.5-2) ...
Setting up libalgorithm-diff-xs-perl (0.04-6) ...
Setting up lib32atomic1 (10.3.0-1ubuntu1-20.04) ...
Setting up chrpath (0.16-2) ...
Setting up liblsan0:amd64 (10.3.0-1ubuntu1-20.04) ...
Setting up libitm1:amd64 (10.3.0-1ubuntu1-20.04) ...
Setting up gcc-9-base:amd64 (9.4.0-1ubuntu1-20.04.1) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up libtsan0:amd64 (10.3.0-1ubuntu1-20.04) ...
Setting up libx32atomic1 (10.3.0-1ubuntu1-20.04) ...
Setting up libctf0:amd64 (2.34-6ubuntu1.3) ...
Setting up lib32itm1 (10.3.0-1ubuntu1-20.04) ...
Setting up libglx0:amd64 (1.3.2-1-ubuntu0.20.04.2) ...
Setting up libx32gcc-s1 (10.3.0-1ubuntu1-20.04) ...
Setting up libtk8.6:amd64 (8.6.10-1) ...
Setting up libx32itm1 (10.3.0-1ubuntu1-20.04) ...
Setting up python3-github (2.0.6-2) ...
Setting up libxml-sax-perl (1.02+dfsg-1) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::SAX::PurePerl with priority 10...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...

Creating config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Setting up libegl1-mesa:amd64 (21.2.6-0ubuntu0.1-20.04.2) ...
Setting up libasan5:amd64 (9.4.0-1ubuntu1-20.04.1) ...
Setting up libgl1:amd64 (1.3.2-1-ubuntu0.20.04.2) ...
Setting up lib32gomp1 (10.3.0-1ubuntu1-20.04) ...
Setting up lib32gcc-s1 (10.3.0-1ubuntu1-20.04) ...
Setting up lib32stdc++6 (10.3.0-1ubuntu1-20.04) ...
Setting up libpython3.8:amd64 (3.8.10-0ubuntu1-20.04.5) ...
Setting up python3-git (3.0.7-1) ...
Setting up python3-pip (20.0.2-5ubuntu1.6) ...
Setting up libxml-libxml-perl (2.0134+dfsg-1build1) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::LibXML::SAX::Parser with priority 50...
update-perl-sax-parsers: Registering Perl SAX parser XML::LibXML::SAX with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Setting up libxml-sax-expat-perl (0.51-1) ...
update-perl-sax-parsers: Registering Perl SAX parser XML::SAX::Expat with priority 50...
update-perl-sax-parsers: Updating overall Perl SAX parser modules info file...
Replacing config file /etc/perl/XML/SAX/ParserDetails.ini with new version
Setting up libx32asan5 (9.4.0-1ubuntu1-20.04.1) ...
Setting up python3-astroid (2.3.3-1) ...
Setting up lib32quadmath0 (10.3.0-1ubuntu1-20.04) ...
Setting up cpp-9 (9.4.0-1ubuntu1-20.04.1) ...
Setting up libc6-dev:amd64 (2.31-0ubuntu9.9) ...
Setting up lib32asan5 (9.4.0-1ubuntu1-20.04.1) ...
Setting up pylint (2.4.4-2) ...
```

- Lấy phần Source code xuống (Đây ko phải source code mà Yocto hỗ trợ và chưa được kiểm chứng)

```
git clone https://github.com/Halolo/orange-pi-distro
```

-Sao chép các module bên ngoài và update bằng lệnh:

```
git submodule update --init
```

```
dothaivu@dothaivu-VirtualBox:~/orange-pi-distro$ git submodule update --init
Submodule 'meta-openembedded' (git://git.openembedded.org/meta-openembedded) registered for path 'meta-openembedded'
Submodule 'meta-qt5' (https://github.com/meta-qt5/meta-qt5.git) registered for path 'meta-qt5'
Submodule 'poky' (git://git.yoctoproject.org/poky) registered for path 'poky'
Cloning into '/home/dothaivu/orange-pi-distro/meta-openembedded'...
Cloning into '/home/dothaivu/orange-pi-distro/meta-qt5'...
Cloning into '/home/dothaivu/orange-pi-distro/poky'...
Submodule path 'meta-openembedded': checked out 'a15d7f6ebcb0ed76c83c28f854d55e3f9d5b3677'
Submodule path 'meta-qt5': checked out 'f22750291b224a3ee68456f0319ba18d428e197a'
Submodule path 'poky': checked out '33e5b9e5adfc941ce65da8161c5cdf81fd1ed6c1'
```

- Vào source-me và chọn Machine:

```
. source-me <machine>
```

Ở đây có 3 option là "orange-pi-zero", "orange-pi-pc" or "orange-pi-pc-plus", do làm việc trên orange pi zero nên câu lệnh sẽ viết lại là

```
. source-me orange-pi-zero
```

```
dothaivu@dothaivu-VirtualBox:~/orange-pi-distro$ . source-me orange-pi-zero
Traceback (most recent call last):
  File "/usr/lib/command-not-found", line 28, in <module>
    from CommandNotFound import CommandNotFound
  File "/usr/lib/python3/dist-packages/CommandNotFound/CommandNotFound.py", line 19, in <module>
    from CommandNotFound.db.db import SQLiteDatabase
  File "/usr/lib/python3/dist-packages/CommandNotFound/db/db.py", line 5, in <module>
    import apt_pkg
ModuleNotFoundError: No module named 'apt_pkg'
OpenEmbedded requires 'python' to be python v2 (>= 2.7.3), not python v3.
Please upgrade your python v2.
Run 'bitbake <target>'
Images:
opipcplus-minimal
opiz-minimal
opipc-minimal
opipc-qt5
```

- Chúng ta cần upgrade python2 nên sử dụng sau :

```
sudo apt install python2
```

```
dothaivu@dothaivu-VirtualBox:~/orange-pi-distro/build$ sudo apt install python2
[sudo] password for dothaivu:
```

```

dothaivug@dothaivu-VirtualBox:~/orange-pi-distro/build$ bitbake opiz-minimal
Parsing recipes: 100% |#####|
Parsing of 1544 .bb files complete (0 cached, 1544 parsed). 2194 targets, 134 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.40.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-20.04"
TARGET_SYS           = "arm-oe-linux-gnueabi"
MACHINE              = "orange-pi-zero"
DISTRO               = "opi"
DISTRO_VERSION        = "nodistro.0"
TUNE_FEATURES         = "arm armv7ve vfp neon vfpv4 callconvention-hard cortexa7"
TARGET_FPU           = "hard"
meta
meta-poky
meta-yocto-bsp        = "HEAD:33e5b9e5adfc941ce65da8161c5cdf81fd1ed6c1"
meta-oe               = "HEAD:a15d7f6ebcb0ed76c83c28f854d55e3f9d5b3677"
meta-qt5              = "HEAD:f22750291b224a3ee68456f0319ba18d428e197a"
meta-opi              = "master:0b7412f9ad645c6cfa72004892ffd00cad921504"

Initialising tasks: 100% |#####|
Sstate summary: Wanted 1082 Found 0 Missed 1082 Current 0 (0% match, 0% complete)
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
Currently 1 running tasks (145 of 3113) 4% |#####|
0: gcc-cross-initial-arm-8.2.0-r0 do_compile - 33s (pid 144022)

```

Hình 1: Build Image

Chạy lệnh

`bitbake<target>`

Có 4 option như hình trên, ở đây ta chọn

bitbake opiz-minimal

2. Tìm file image và lệnh tải lên sdcard

- Tập lưu trữ root, bootloader, kernel image tất cả phân vùng cần thiết phải được tạo ra trong :

`build/tmp-glibc/deploy/images/<machine>/`

+ Ở đây machine là orange pi zero

`scp build/tmp-glibc/deploy/images/orange-pi-zero/u-boot-sunxi-with-spl.bin root@<ip>:`

-Có thể sử dụng lệnh dd để flash nó lên sdcard(đồng thời ngắt kết nối tất cả các phân vùng trước đó.

`dd if=build/tmp-glibc/deploy/images/<machine>/<image>-<machine>.sunxi-sdimg of=/dev/??? bs=1M`

+ Các bước tiếp theo làm giống chương 2.

3. Lỗi phiên bản Python và cách khắc phục

-Nếu hiện lỗi này, đây có thể là lỗi do các phiên bản python đã qua cũ, hãy thử vào source file của nó để sửa.

```
ImportError: cannot import name 'MutableMapping' from 'collections' (/usr/lib/python3.10/collections/__init__.py)
```

Hình 2: Lỗi phiên bản python

- Để sửa lỗi trên, ta phải chọn phiên bản phù hợp là python 3.6 :

```
sudo apt update  
sudo apt install software-properties-common  
sudo add-apt-repository ppa:deadsnakes/ppa  
sudo apt install python3.6
```

- Vào thư mục này để kiểm tra có cài đặt được python3.6 không :

```
ls /usr/bin/python*
```

- Chọn python3.6 làm trình biên dịch mặc định

```
sudo update-alternatives --install /usr/bin/python python  
/usr/bin/python3.6 1  
sudo update-alternatives --install /usr/bin/python3 python3  
/usr/bin/python3.6 1  
sudo update-alternatives --config python ( lệnh này hiện ra  
bảng config của nó )  
sudo update-alternatives --config python3 ( lệnh này hiện ra  
bảng config của nó )
```

```
There are 3 choices for the alternative python (providing /usr/bin/python).

  Selection    Path                Priority    Status
  -----
* 0            /usr/bin/python3.8    3          auto mode
  1            /usr/bin/python3.10    1          manual mode
  2            /usr/bin/python3.8     3          manual mode
  3            /usr/bin/python3.9     2          manual mode

Press <enter> to keep the current choice[*], or type selection number: 3
update-alternatives: using /usr/bin/python3.9 to provide /usr/bin/python (python)
ual mode
```

Hình 3: Config default python (trên hình là phiên bản python 3.9)

4. Nhận xét và đánh giá build Image cho Orange pi bằng Yocto

- Git Clone cho Orange Pi không được Yocto Project hỗ trợ, còn Git Clone trên không đảm bảo, lại phụ thuộc vào các phiên bản phần mềm đi cùng (như Python) và chỉ hỗ trợ được cho một số KIT Orange Pi như Orange Pi Zero, PC, PC Plus nên vì thế công việc cần tuy chỉ có các bước trên nhưng mất nhiều thời gian để sửa các lỗi.

- Vì thế, em tìm được một cách build image cho orange pi

git clone <https://github.com/orangepi-xunlong/orangepi-build.git>

đây là git clone của trang chủ orangepi-xunlong – Nhà sản xuất KIT Orange pi.

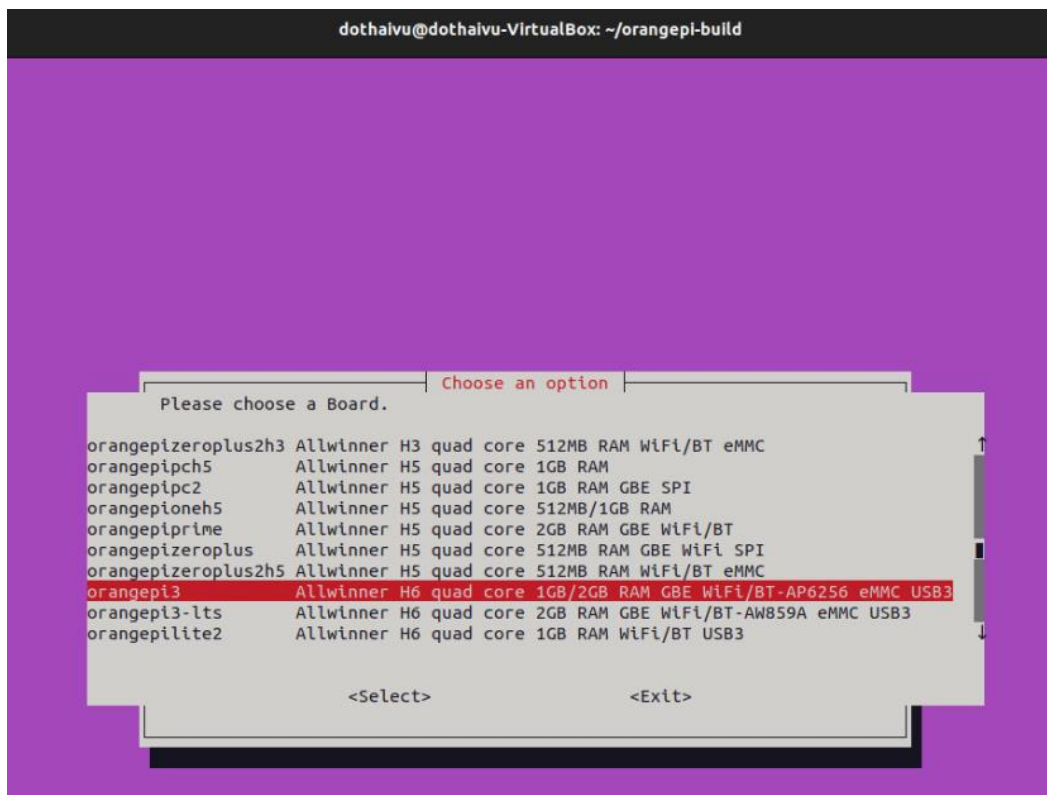
- Sử dụng lệnh để bắt đầu:

cd orangepi-build

./build.sh

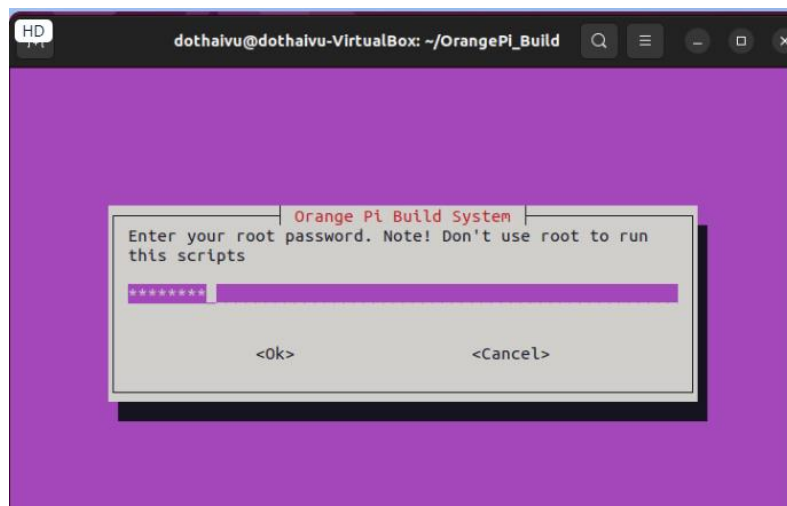


- Ta sẽ lựa chọn loại KIT, loại chip,....



Hình 4: Chọn cấu hình cho KIT

- Nhập mật khẩu root:



- Sau đó, sẽ tự động build Image cho KIT.

5. Kết luận

- Đã hoàn thành công việc (Build Firmware orangepi zero bằng yocto)
- Khó khăn:
 - + Tài liệu viết về Yocto cho Orangepi khá ít và không phổ biến.
 - + Ubuntu 22.04 không còn hỗ trợ python3.6 nên để sử dụng được thì phải tải Ubuntu 20.04