

**MỤC LỤC**

**REPORT 0:** Thực hành Linux và Shell

**REPORT 1:** Sử dụng thư viện WiringPi

**REPORT 2:** Điều khiển LED trên kernel của RASPBERRY

**REPORT 3:** Điều khiển LED trên User Space

**REPORT 4:** Điều khiển giao tiếp I2C qua Driver

# BÁO CÁO THỰC TẬP NHÚNG

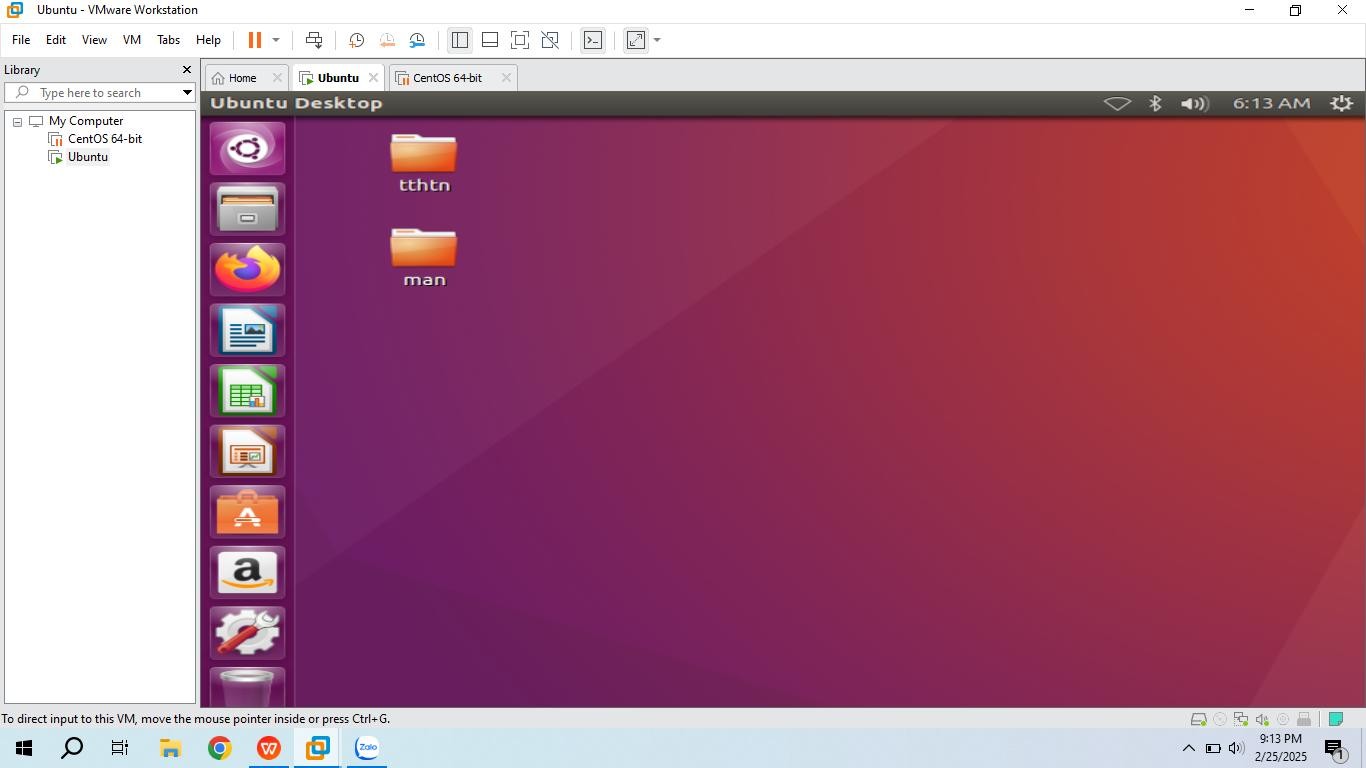
Ngày:12/03/2025

**REPORT 0:** Thực hành Linux và Shell

**LỚP:** 22119CL3B

|  |  |  |
| --- | --- | --- |
| **MSSV** | **HỌ VÀ TÊN** | **GHI CHÚ** |
| 22119137 | Đặng Hưng Thịnh |  |
| 22119095 | Trần Tuấn Kiệt |  |
| 22119130 | Phạm Duy Tân |  |
| 22119112 | Trần Huỳnh Như |  |

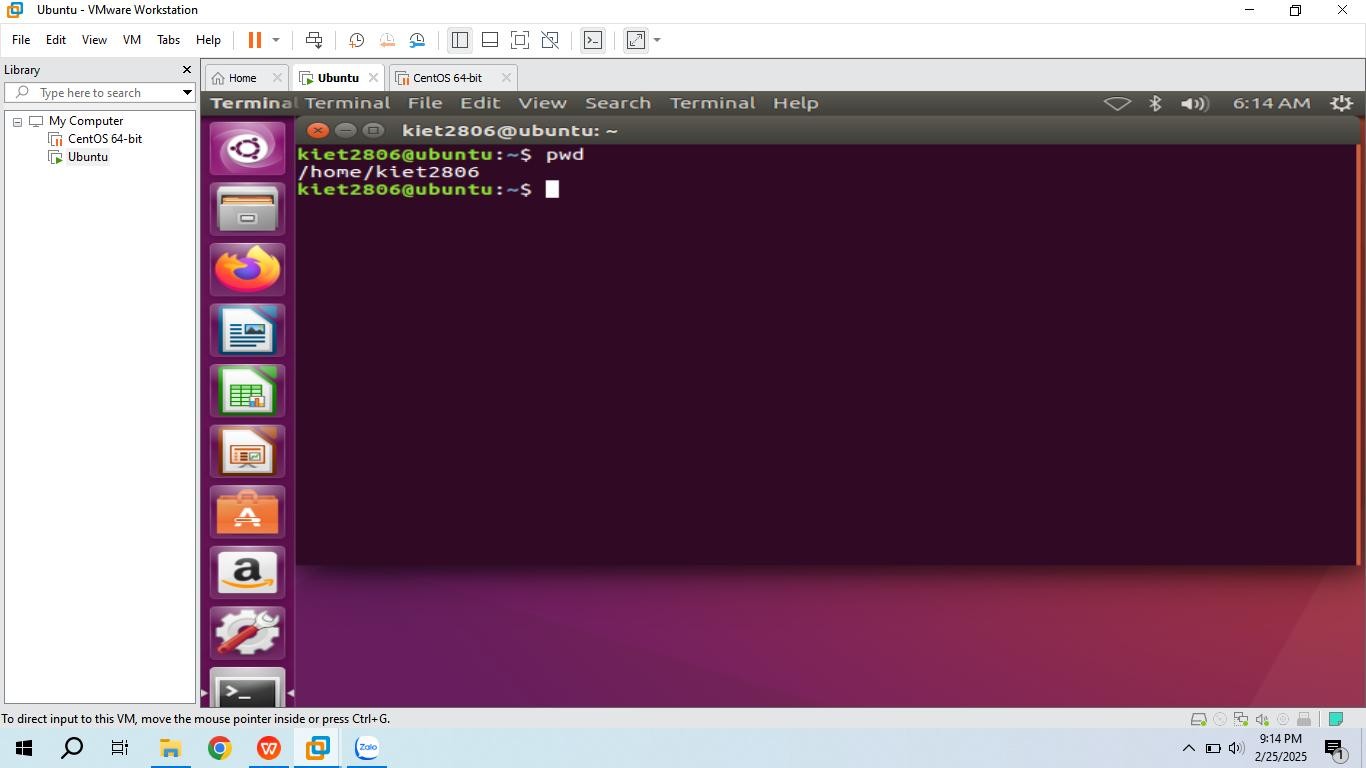
## Câu 1: Chụp hình kết quả cài đặt Vmwave Workstation và Linux



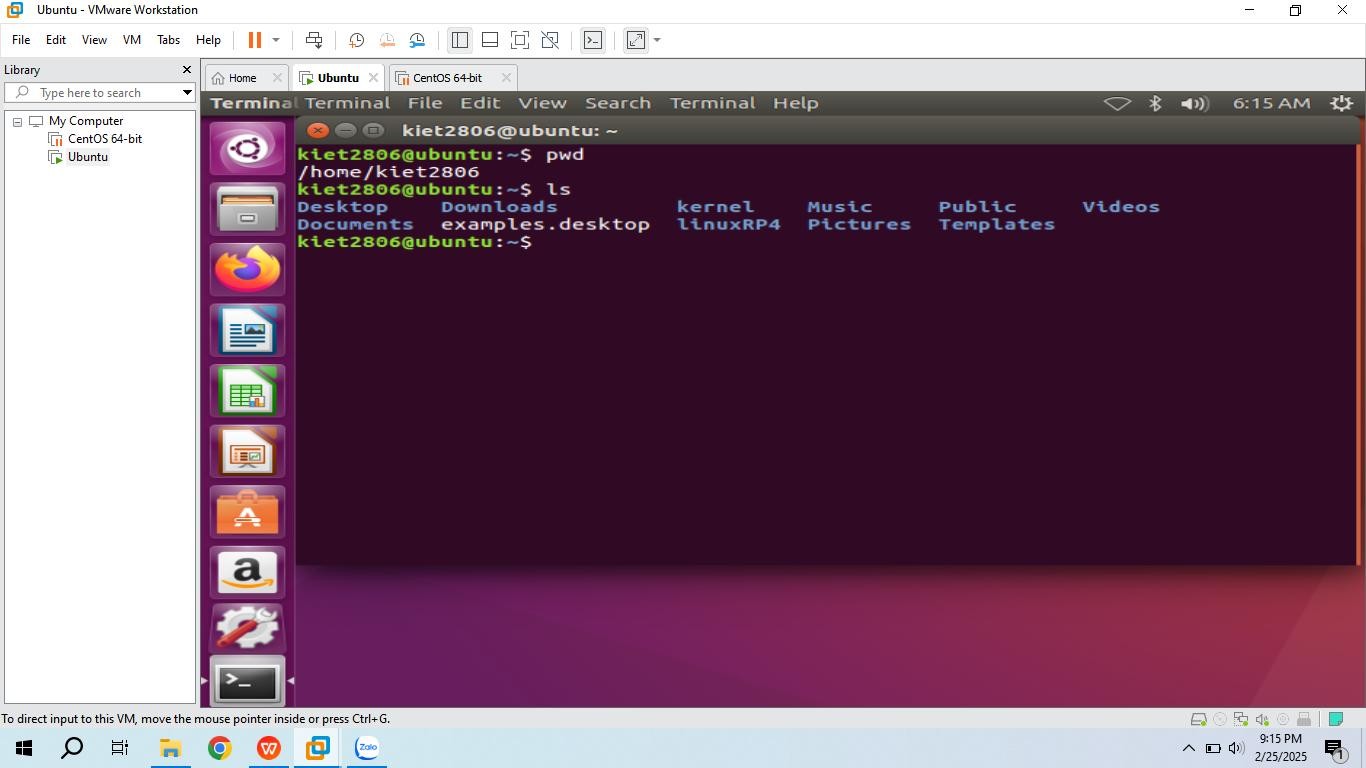
*Hình 1: Kết quả cài đặt Vmwave Workstation và Linux.*

## Câu 2: Thực hiện 1 số lệnh cơ bản về Linux

“pwd”: Hiển thị đường dẫn đầy đủ của thư mục hiện tại.

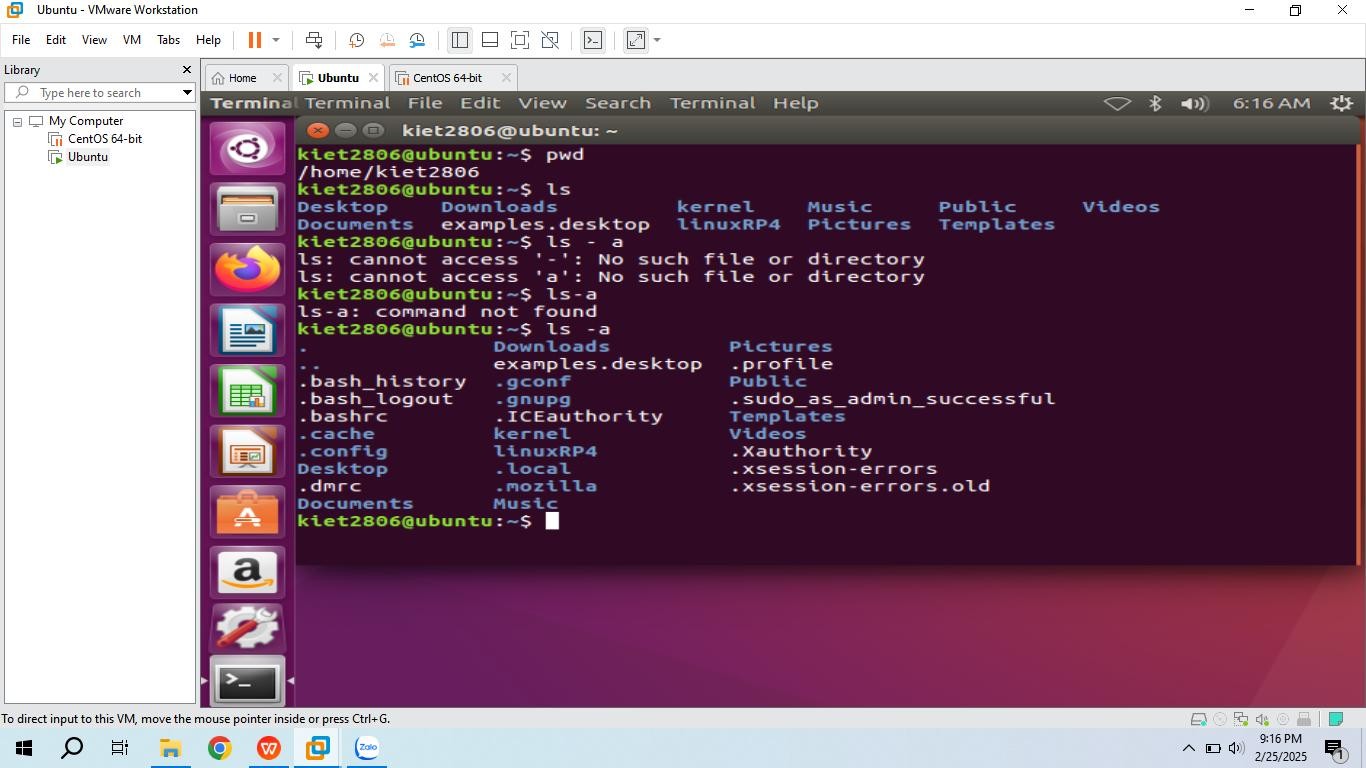


*Hình 2: Kết quả lệnh pwd* “ls”: Liệt kê các tập tin và thư mục trong thư mục hiện tại.



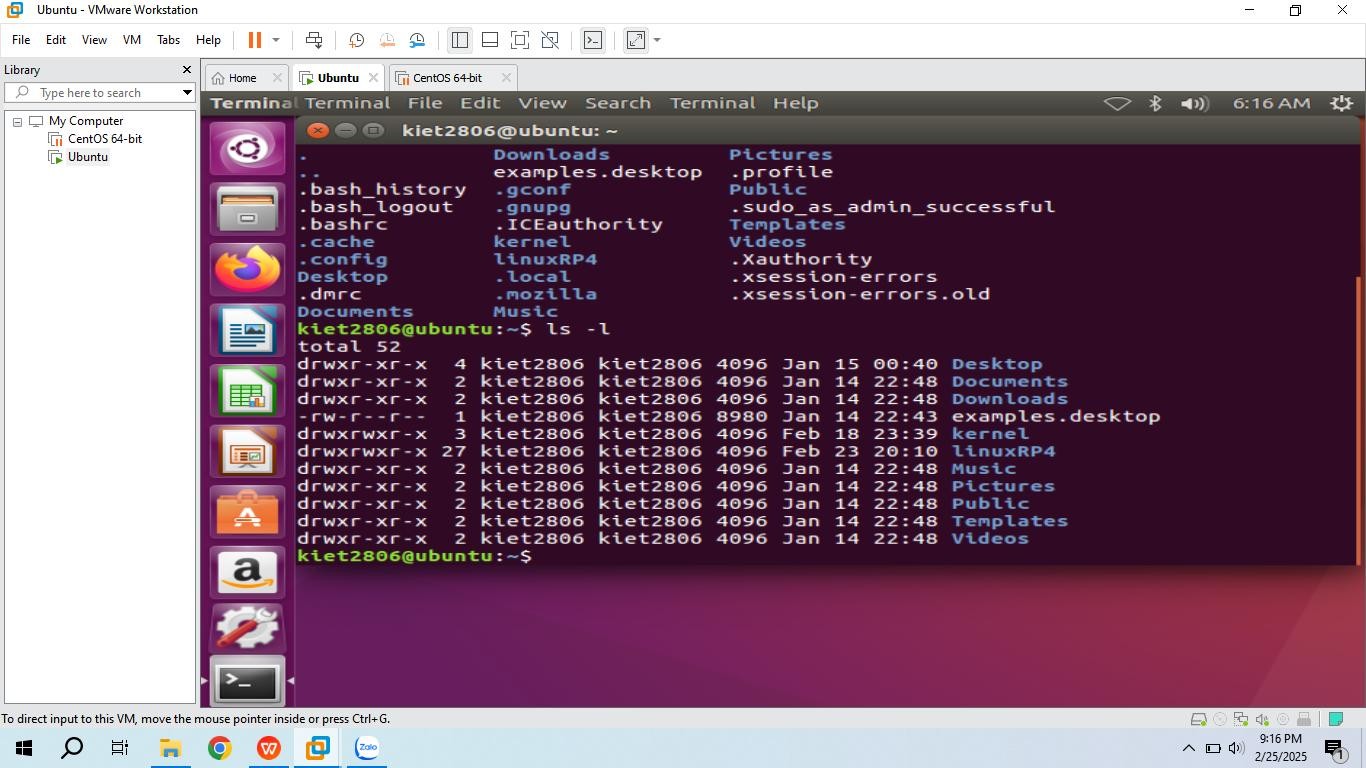
*Hình 3: Kết quả lệnh ls*

“ls -a”: hiển thị tất cả các tập tin, bao gồm cả các tập tin ẩn.



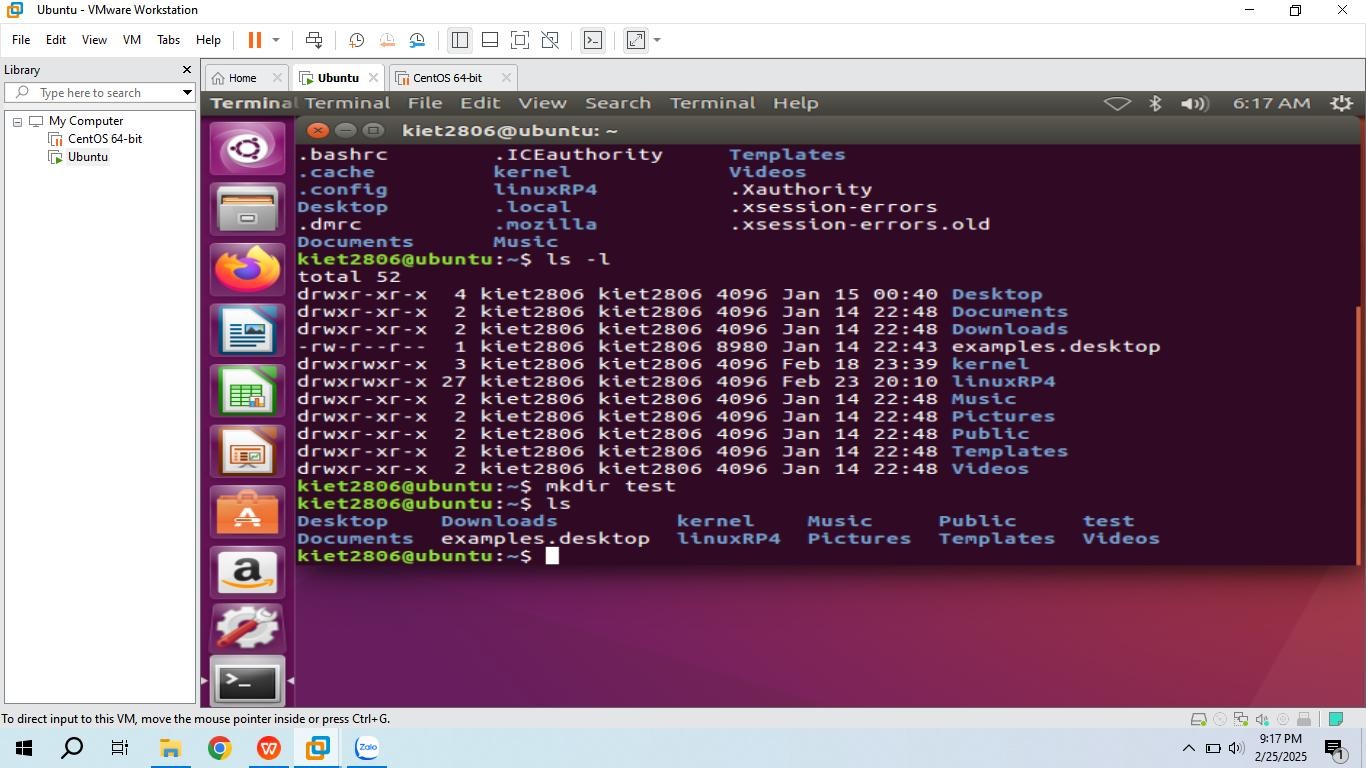
*Hình 4: Kết quả lệnh ls-a*

“ls -l”: hiển thị thông tin chi tiết về các tập tin và thư mục, bao gồm quyền truy cập, chủ sở hữu, nhóm, kích thước, ngày và thời gian tạo và tên tập tin.



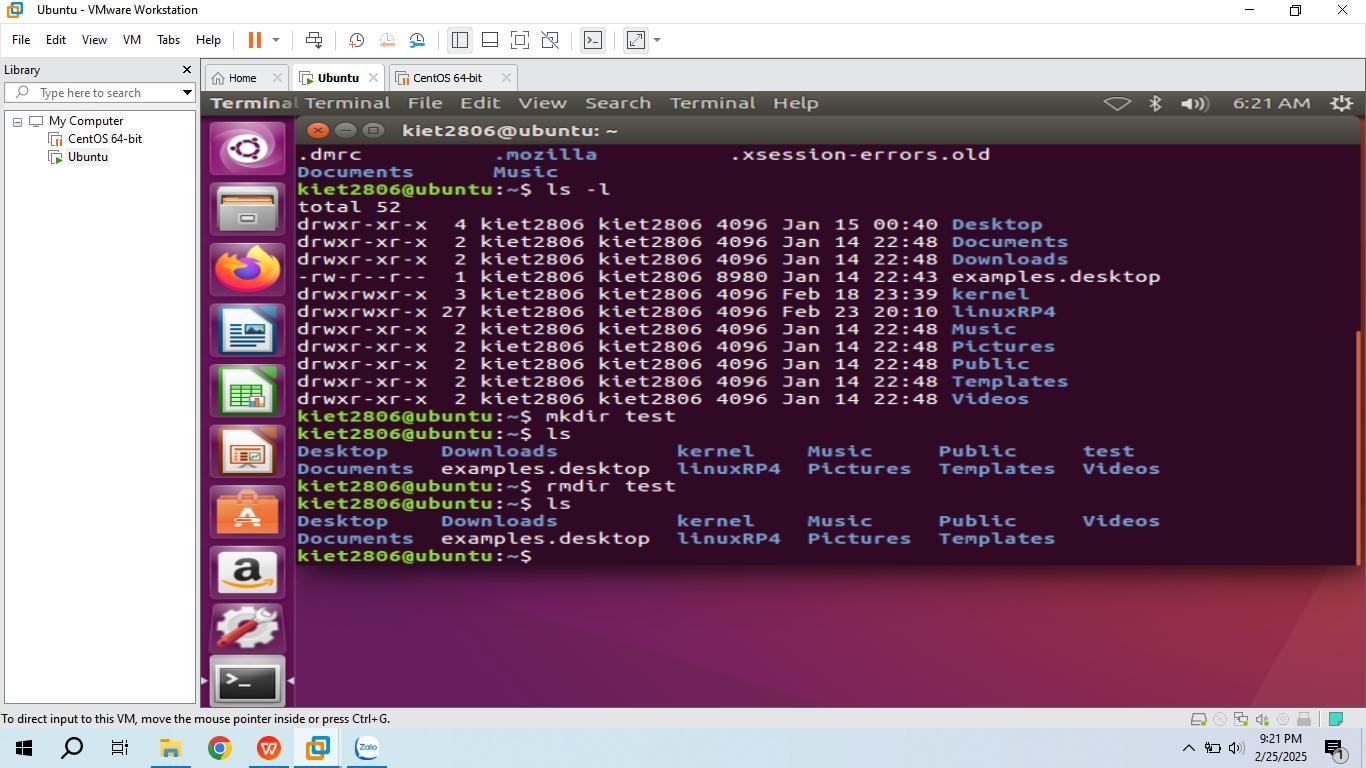
*Hình 5: Kết quả lệnh ls-l*

“mkdir”: để tạo thư mục



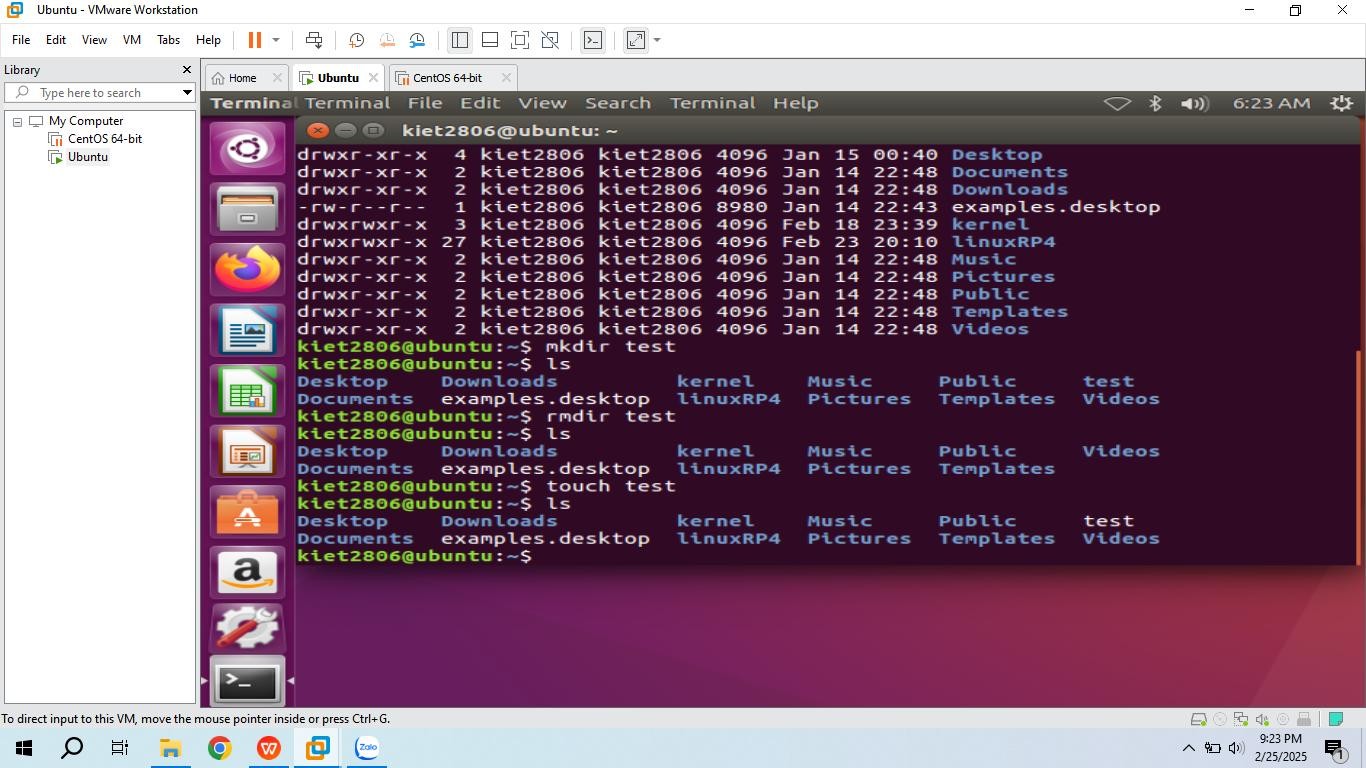
*Hình 6: Kết quả lệnh mkdir*

“rmdir”:xoá thư mục



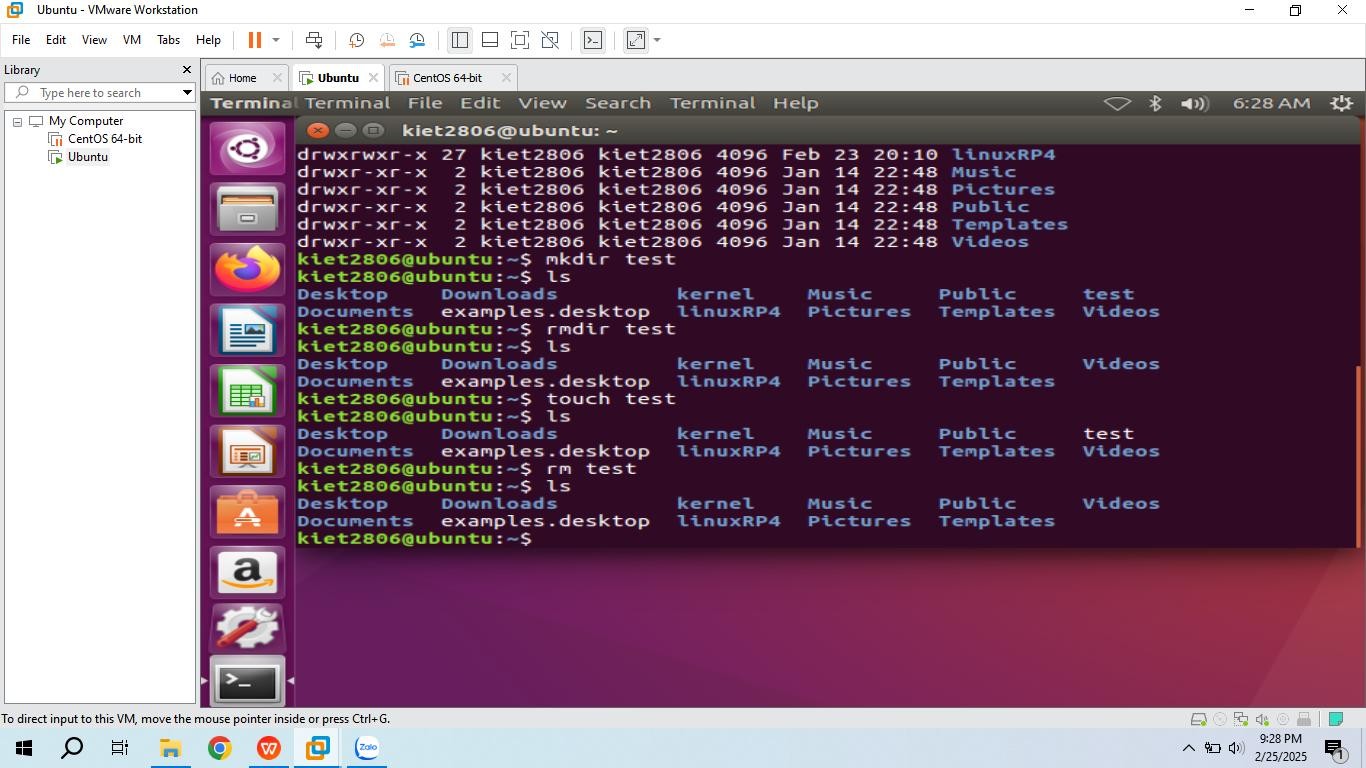
*Hình 7: Kết quả lệnh rmdir*

“touch”: tạo tệp tin



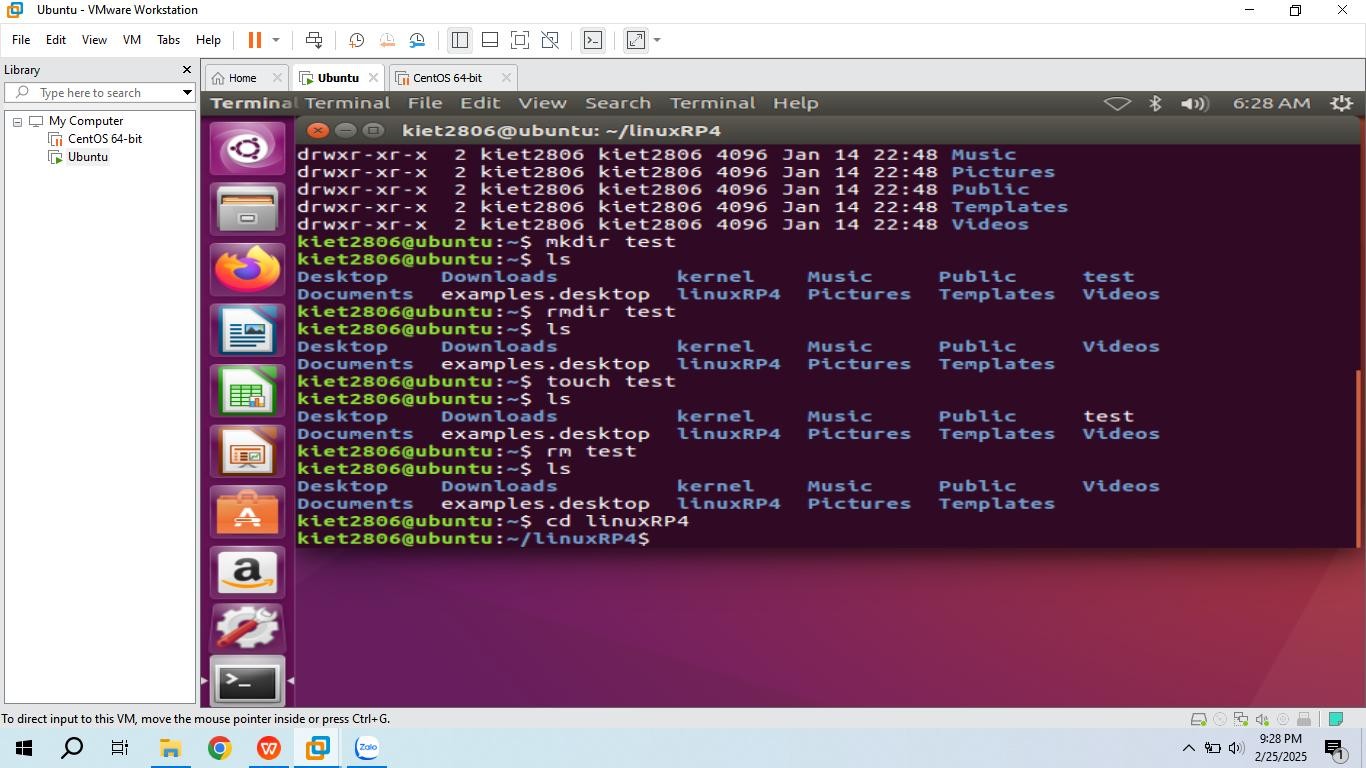
*Hình 8: Kết quả lệnh touch*

“rm”: xoá tệp tin



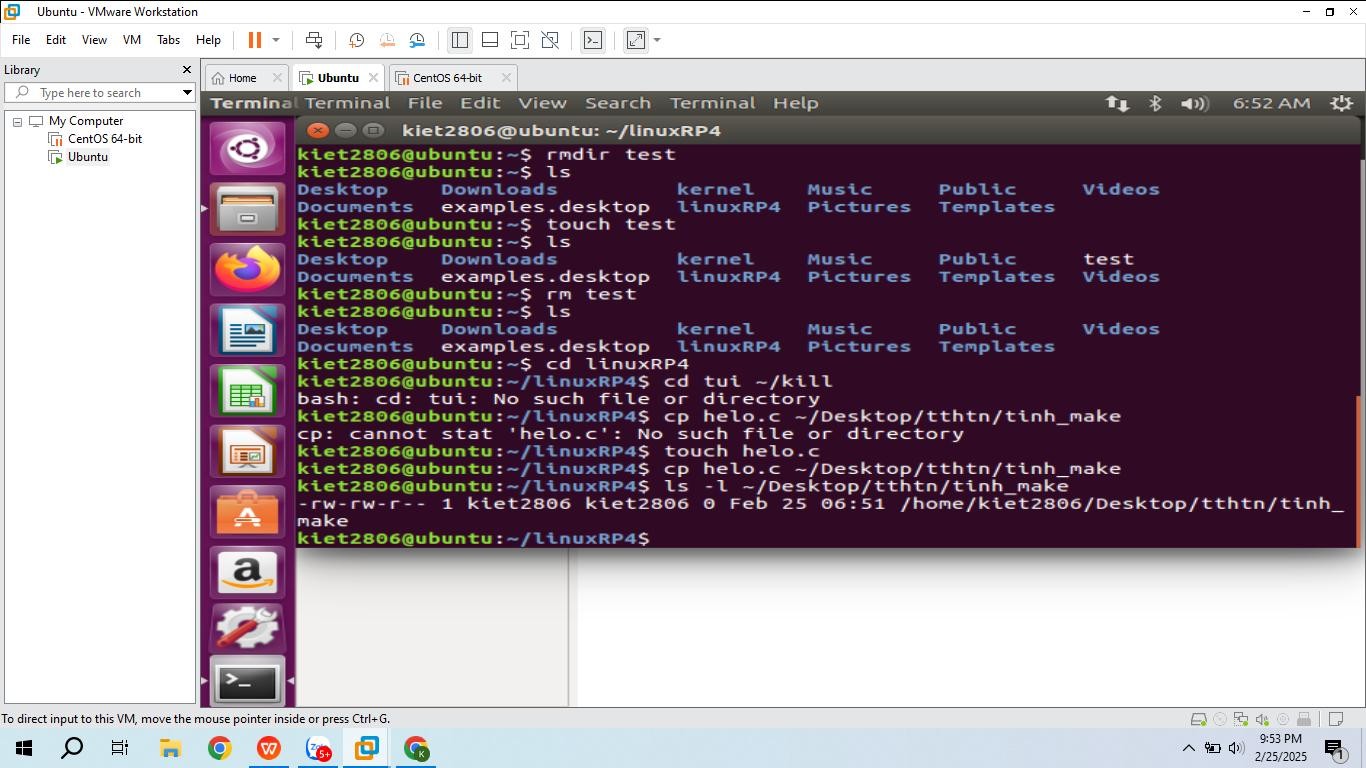
*Hình 9: Kết quả lệnh rm*

“cd”: Di chuyển đến một thư mục khác.



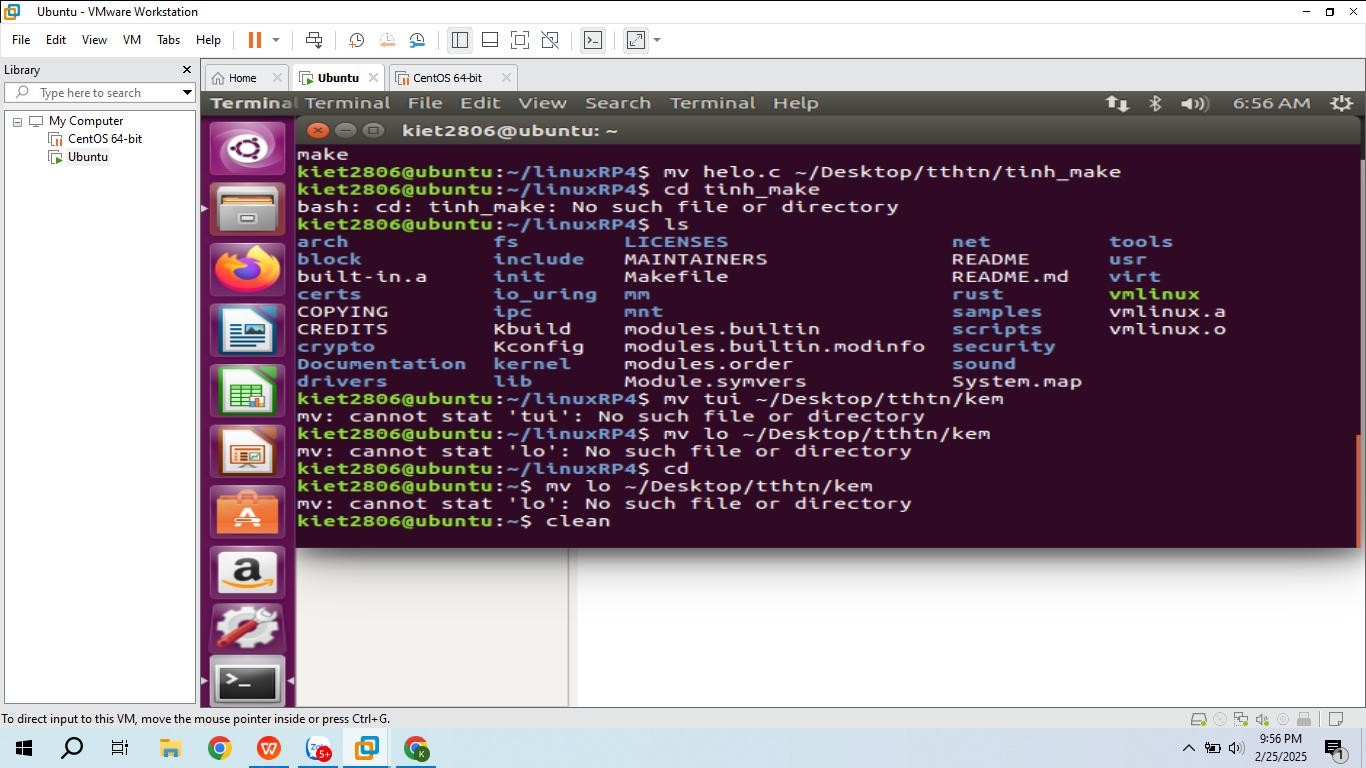
*Hình 10: Kết quả lệnh cd*

“cp”: Sao chép thư mục, tập tin

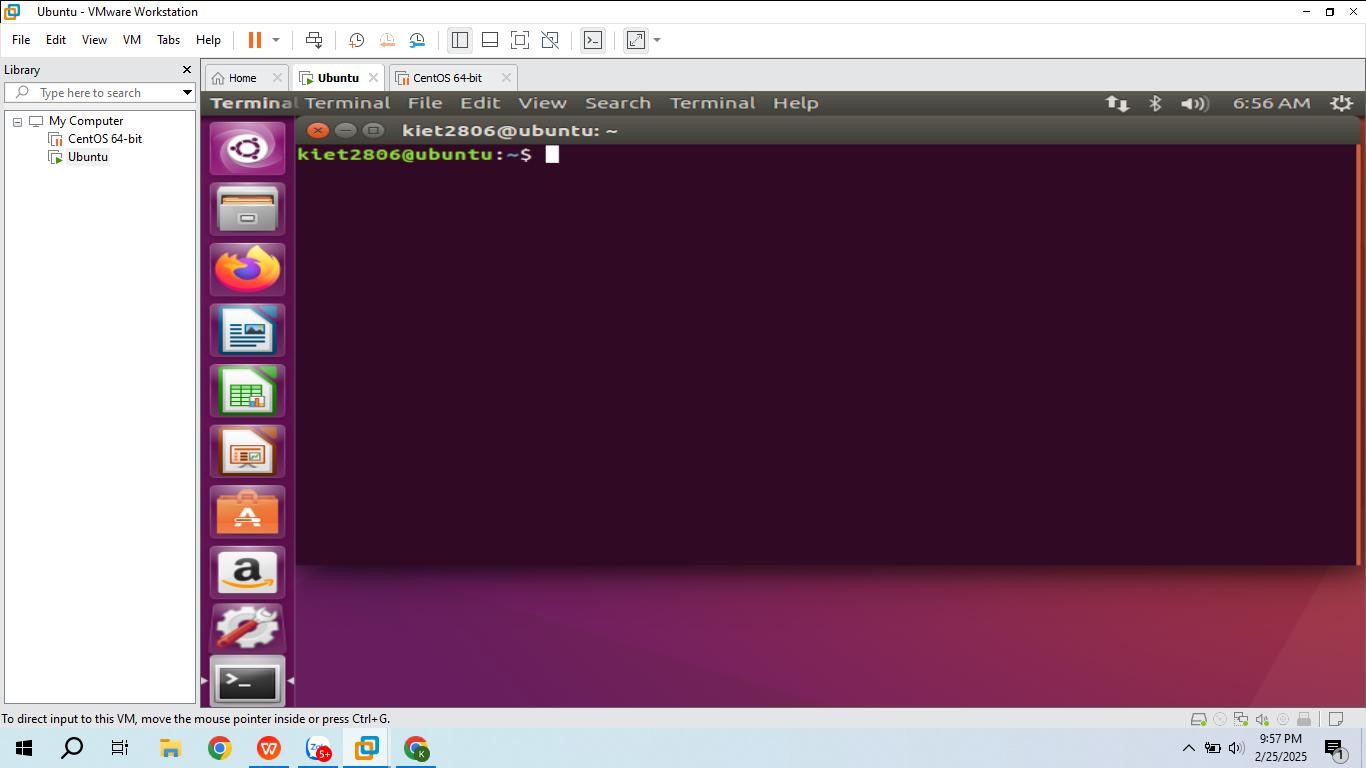


*Hình 11: Kết quả lệnh cp khi copy file helo.c vào thư mục tinh\_make*

“clear”: xoá màn hình Terminal



*Hình 12: Khi gõ lệnh clear*



*Hình 13: Kết quả khi xoá Terminal*

# BÁO CÁO THỰC TẬP NHÚNG

**Ngày:**05/03/2025

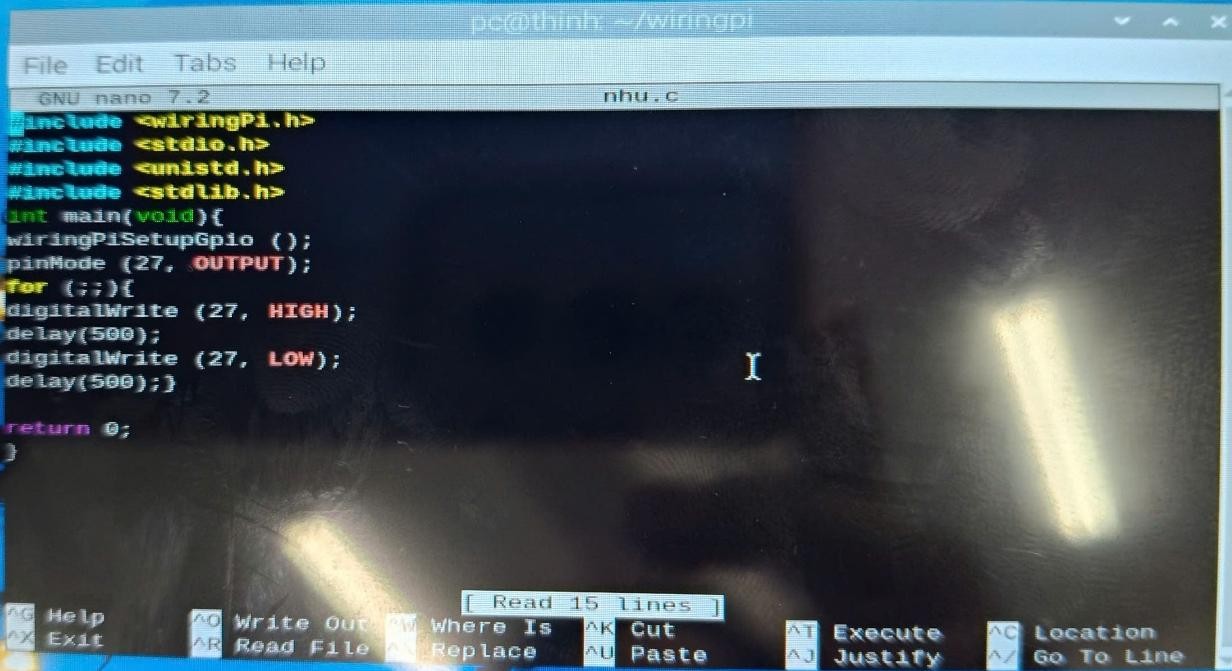
## REPORT 1: Kết nối Raspberry điều khiển LED dùng thư viện WiringPi

**LỚP:** 22119CL3B

|  |  |  |
| --- | --- | --- |
| **MSSV** | **HỌ VÀ TÊN** | **GHI CHÚ** |
| 22119137 | Đặng Hưng Thịnh |  |
| 22119095 | Trần Tuấn Kiệt |  |
| 22119112 | Trần Huỳnh Như |  |
| 22119130 | Phạm Duy Tân |  |

**Thêm thư viện wiringPi:**

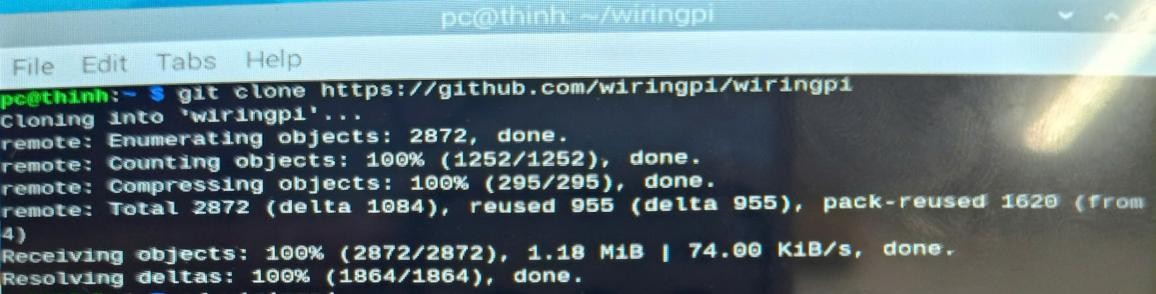
|  |
| --- |
| sudo apt-get update sudo apt-get upgrade  git clone <https://github.com/wiringpi/wiringpi>cd wiringpi  ./build  gpio -v  -Dùng lệnh “ nano nhu.c” để tạo file nhu.c |



*Hình 1: Code nhu.c*

**-** Biên dịch gcc cho file nhu1.c:

Dùng lệnh: “gcc -o nhu1 nhu.c -l wiringPi”



*Hình 2: Thực hiện lệnh gcc tạo ra file nhu.o*

**-** Cấp quyền cho file nhu1

Dùng lệnh: “sudo chmod 777 nhu1”



*Hình 3: Cấp quyền cho file nhu1*

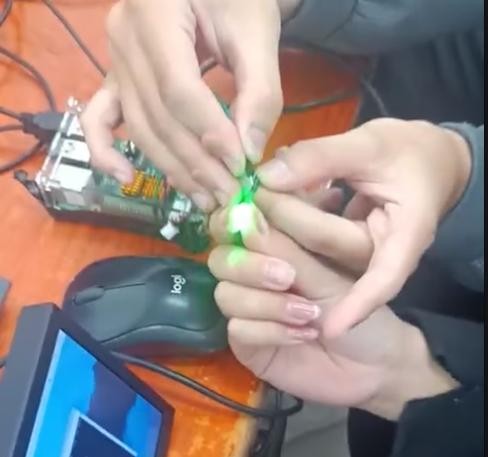
-Chạy nhu1:

Dùng lệnh: “./nhu1”



*Hình 4: Chạy file nhu1*

**-**Kết quả:



*Hình 5: Led sáng*



*Hình 6: Led tắt*

# BÁO CÁO THỰC TẬP NHÚNG

**Ngày:**12/03/2025

## REPORT 2: Điều khiển LED trên kernel của RASPBERRY

**LỚP: 22119CL3B**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **HỌ VÀ TÊN** | **GHI CHÚ** |
| 22119137 | Đặng Hưng Thịnh |  |
| 22119095 | Trần Tuấn Kiệt |  |
| 22119130 | Phạm Duy Tân |  |
| 22119112 | Trần Huỳnh Như |  |

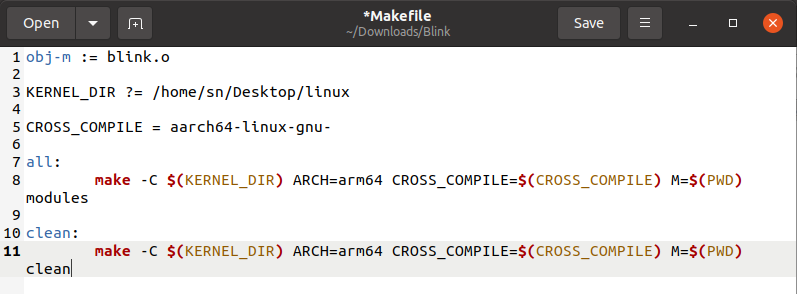
**Câu 1**: Tạo file LED.c và file Makefile:

Bằng cách sử dụng câu lệnh “nano blink.c Makefile”

*Hình 1: Hình tạo file blink.c và file Makefile.*

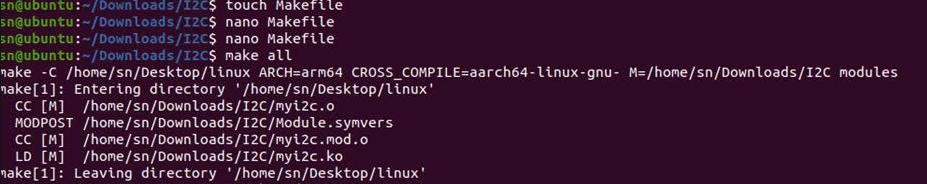
* SOURCE CODE DRIVER:

|  |
| --- |
| #include <linux/fs.h>  #include <linux/cdev.h>  #include <linux/init.h>  #include <linux/device.h>  #include <linux/module.h>  #include <linux/kernel.h>  #include <linux/uaccess.h>  #include <linux/io.h>  #include <linux/kthread.h>  #include <linux/delay.h>  #define DRIVER\_NAME "gpio17\_driver"  #define DRIVER\_CLASS "gpio\_class"  #define BCM2711\_GPIO\_ADDRESS 0xFE200000 // Base address GPIO trên RPi 4  #define GPIO\_PIN 17 // Sử dụng GPIO 17  static dev\_t dev\_no;  static struct cdev dev;  static struct class \*cls;  static void \_\_iomem \*gpio\_base;  static struct task\_struct \*blink\_thread = NULL; // Luồng nhấp nháy LED  static bool blink\_flag = false;  // ========================== GPIO ==========================  static void gpio\_init(void) {  unsigned int \*gpio\_sel\_reg = (unsigned int \*)(gpio\_base + (GPIO\_PIN / 10) \* 4);  unsigned int shift = (GPIO\_PIN % 10) \* 3;  \*gpio\_sel\_reg &= ~(0b111 << shift); // Clear bits  \*gpio\_sel\_reg |= (0b001 << shift); // Set output mode  }  static void gpio\_set(void) {  unsigned int \*gpio\_set\_reg = (unsigned int \*)(gpio\_base + 0x1C);  \*gpio\_set\_reg = (1 << GPIO\_PIN);  }  static void gpio\_clr(void) {  unsigned int \*gpio\_clr\_reg = (unsigned int \*)(gpio\_base + 0x28);  \*gpio\_clr\_reg = (1 << GPIO\_PIN);  }  // ======================== Thread ==========================  static int blink\_function(void \*data) {  while (!kthread\_should\_stop()) {  if (blink\_flag) {  gpio\_set();  msleep(1000); // Bật LED 1 giây  gpio\_clr();  msleep(1000); // Tắt LED 1 giây  } else {  gpio\_clr(); // Đảm bảo LED tắt  break;  }  }  return 0;  }  // ==================== File Operations =====================  static int device\_open(struct inode \*device\_file, struct file \*instance) {  printk("GPIO17 Driver: Opened\n");  return 0;  }  static int device\_close(struct inode \*device\_file, struct file \*instance) {  printk("GPIO17 Driver: Closed\n");  return 0;  }  static ssize\_t device\_write(struct file \*file, const char \*user\_buffer, size\_t size, loff\_t \*offs) {  char user\_data[10];  memset(user\_data, 0, sizeof(user\_data));  if (copy\_from\_user(user\_data, user\_buffer, size)) {  return -EFAULT;  }  printk("GPIO17 Driver: Received command: %s\n", user\_data);  if (strncmp(user\_data, "1", 1) == 0) {  printk("GPIO17 ON (Steady)\n");  blink\_flag = false;  if (blink\_thread) {  kthread\_stop(blink\_thread);  blink\_thread = NULL;  }  gpio\_set();  } else if (strncmp(user\_data, "0", 1) == 0) {  printk("GPIO17 OFF\n");  blink\_flag = false;  if (blink\_thread) {  kthread\_stop(blink\_thread);  blink\_thread = NULL;  }  gpio\_clr();  } else if (strncmp(user\_data, "blink", 5) == 0) {  printk("GPIO17 Blink Mode\n");  blink\_flag = true;  if (!blink\_thread) {  blink\_thread = kthread\_run(blink\_function, NULL, "blink\_thread");  }  }  return size;  }  static struct file\_operations fops = {  .owner = THIS\_MODULE,  .write = device\_write,  .open = device\_open,  .release = device\_close,  };  // ==================== Module Lifecycle ====================  static int \_\_init device\_driver\_init(void) {  if (alloc\_chrdev\_region(&dev\_no, 0, 1, DRIVER\_NAME) < 0) {  printk("GPIO17 Driver: Failed to allocate device number\n");  return -1;  }  printk("GPIO17 Driver: Major - %d, Minor - %d\n", MAJOR(dev\_no), MINOR(dev\_no));  if ((cls = class\_create(DRIVER\_CLASS)) == NULL) {  unregister\_chrdev\_region(dev\_no, 1);  printk("GPIO17 Driver: Failed to create class\n");  return -1;  }  if (device\_create(cls, NULL, dev\_no, NULL, DRIVER\_NAME) == NULL) {  class\_destroy(cls);  unregister\_chrdev\_region(dev\_no, 1);  printk("GPIO17 Driver: Failed to create device file\n");  return -1;  }  cdev\_init(&dev, &fops);  if (cdev\_add(&dev, dev\_no, 1) < 0) {  device\_destroy(cls, dev\_no);  class\_destroy(cls);  unregister\_chrdev\_region(dev\_no, 1);  printk("GPIO17 Driver: Failed to register device\n");  return -1;  }  gpio\_base = ioremap(BCM2711\_GPIO\_ADDRESS, PAGE\_SIZE);  if (!gpio\_base) {  printk("GPIO17 Driver: Failed to map GPIO memory\n");  return -1;  }  gpio\_init();  printk("GPIO17 Driver: Initialized successfully\n");  return 0;  }  static void \_\_exit device\_driver\_exit(void) {  if (blink\_thread) {  kthread\_stop(blink\_thread);  blink\_thread = NULL;  }  iounmap(gpio\_base);  cdev\_del(&dev);  device\_destroy(cls, dev\_no);  class\_destroy(cls);  unregister\_chrdev\_region(dev\_no, 1);  printk("GPIO17 Driver: Unloaded successfully\n");  }  module\_init(device\_driver\_init);  module\_exit(device\_driver\_exit);  MODULE\_LICENSE("GPL");  MODULE\_AUTHOR("Your Name");  MODULE\_DESCRIPTION("GPIO 17 Control Driver for Raspberry Pi 4"); |



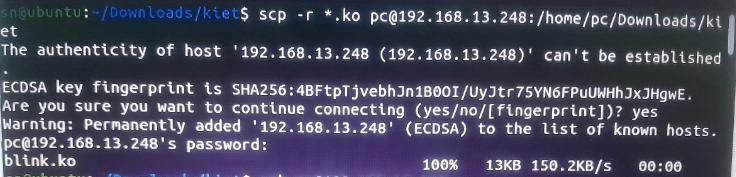
**Câu 4:** Thực hiện make file:

Dùng câu lệnh “make”



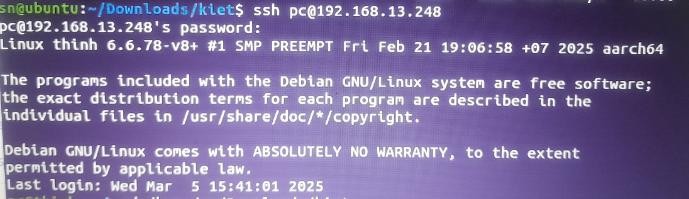
*Hình 3: Hình thực hiện lệnh make.*

**Câu 5:** Sao chép tệp blink.ko an toàn qua ssh tới thư mục kiet của máy đích với user là pc và địa chỉ IP là 192.168.13.248.



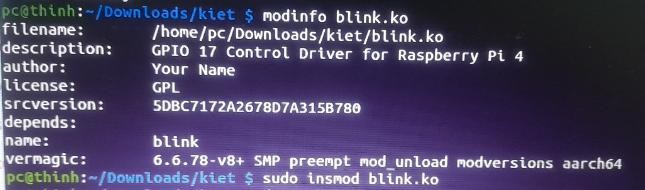
*Hình 4: Hình thực hiện sao chép tệp.*

**Câu 6:** Kết nối từ xa đến raspberry thật mang user là pc và địa chỉ IP là 192.168.13.248 thông qua giao thức ssh.



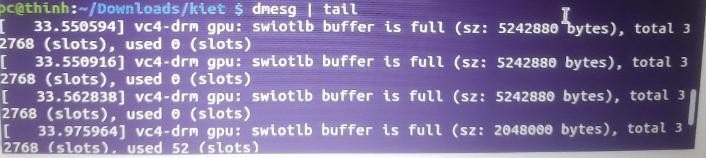
*Hình 5: Hình thực hiện kết nối từ xa đến raspberry thật.*

**Câu 7:** Nạp blink.ko vào hệ thống:



*Hình 6: Hình thực hiện nạp module kernel vào hệ thống.*

**Câu 8:** Xem các thông báo mới nhất của hệ thống, đặc biệt là thông tin liên quan đến kernel: Dùng lệnh: “ dmesg | tail”

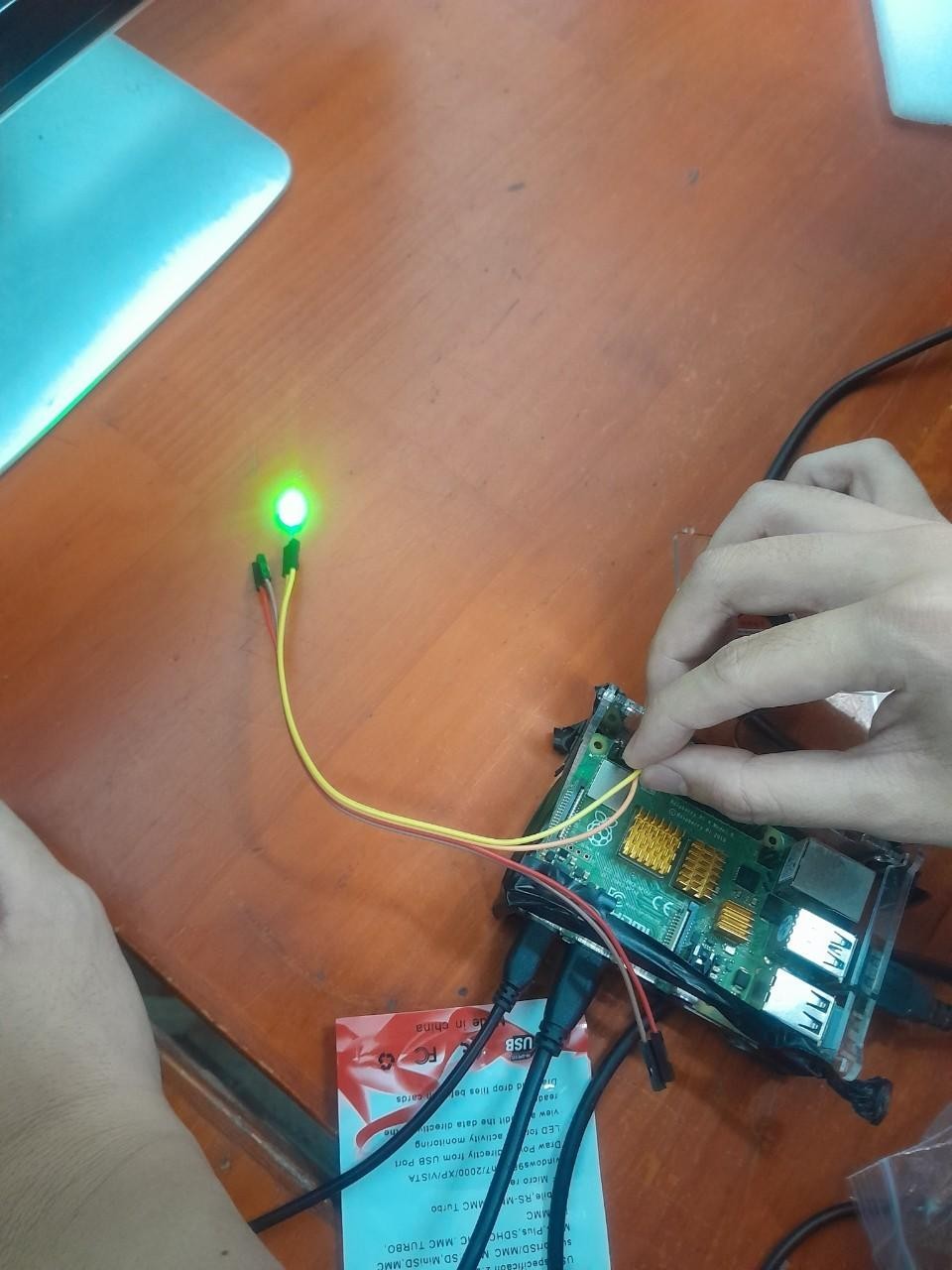


*Hình 7: Kết quả thực hiện lệnh.*

**Câu 9**: Điều khiển đèn led:

* Điều khiển led sáng:





*Hình 8: Kết quả led đã sáng.*

* Điều khiển led tắt:





*Hình 9: Kết quả led đã tắt.*

# BÁO CÁO THỰC TẬP NHÚNG

**Ngày:**12/03/2025

**REPORT 3:** Điều khiển LED trên User Space

**LỚP:** 22119CL3B

|  |  |  |
| --- | --- | --- |
| **MSSV** | **HỌ VÀ TÊN** | **GHI CHÚ** |
| 22119095 | Trần Tuấn Kiệt |  |
| 22119112 | Trần Huỳnh Như |  |
| 22119130 | Phạm Duy Tân |  |
| 22119137 | Đặng Hưng Thịnh |  |

Tạo make file :

-nano blink.c

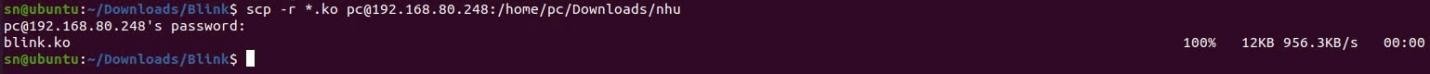
-make all

Tạo file out:

-nano usr\_space.c

- gcc usr\_space\_app.c -o out

Tiến hành kết nối qua rap pi4:



*Hình 1. Kết nối máy qua rap pi4*

- scp -r \*.ko pc@192.168.13.248: /home/pc/Downloas/tan \*tren rasp

-scp /home/sn/Downloads/usr\_space/out pc@192.168.13.248:/home/pc/Downloads/

\*Câu lệnh thực thi trên Command rap pi4:

-Truy xuất vào đường dẫn chứa file blink.ko và file out



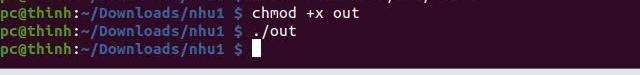
*Hình 2. Kiểm tra phiên bản file thực thi*

-Kiểm tra file blink ko: ls

+cấp quyền cho file blink.ko



*Hình 3. Cấp quyền cho file bink.k0*



*Hình 4. Cấp quyền cho file out*

+Cấp quyền thực thi :sudo chmod 666 /dev/gpio17\_driver

+Cấp quyền thực thi cho out : chmod +x out

-Tiến hành chạy lệnh :./out

# BÁO CÁO THỰC TẬP NHÚNG

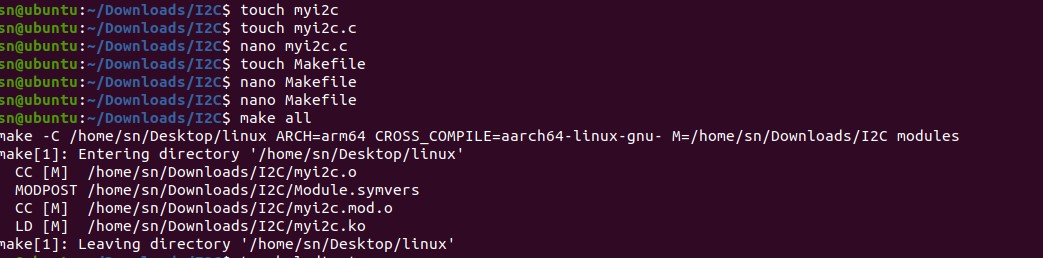
**Ngày:**12/03/2025

**REPORT 4:** Điều khiển giao tiếp I2C qua Driver

**LỚP: Nhóm 0**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **HỌ VÀ TÊN** | **GHI CHÚ** |
| 22119137 | Đặng Hưng Thịnh |  |
| 22119095 | Trần Tuấn Kiệt |  |
| 22119130 | Phạm Duy Tân |  |
| 22119112 | Trần Huỳnh Như |  |

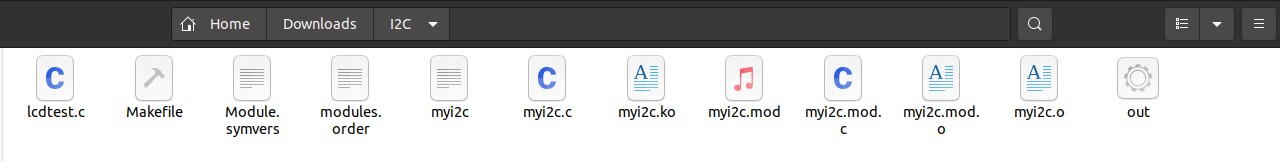
1. Tạo file myi2c và tạo Makefile



*Hình 1. Tạo file myi2c và Makefile* 2. Tạo file lcdtest.c và tạo file out

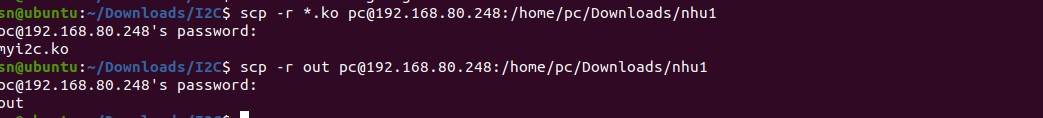


*Hình 2. Tạo file lcdtest.c và tạo file out*

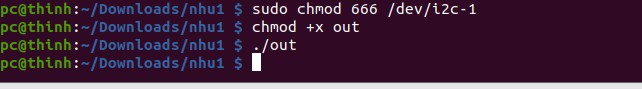
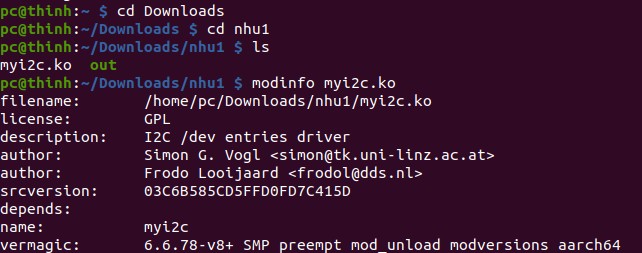


*Hình 3. Các file có được sau khi make all*

3. Kết nối với Rasberry, truyền Makefile và file out cho Rasberry

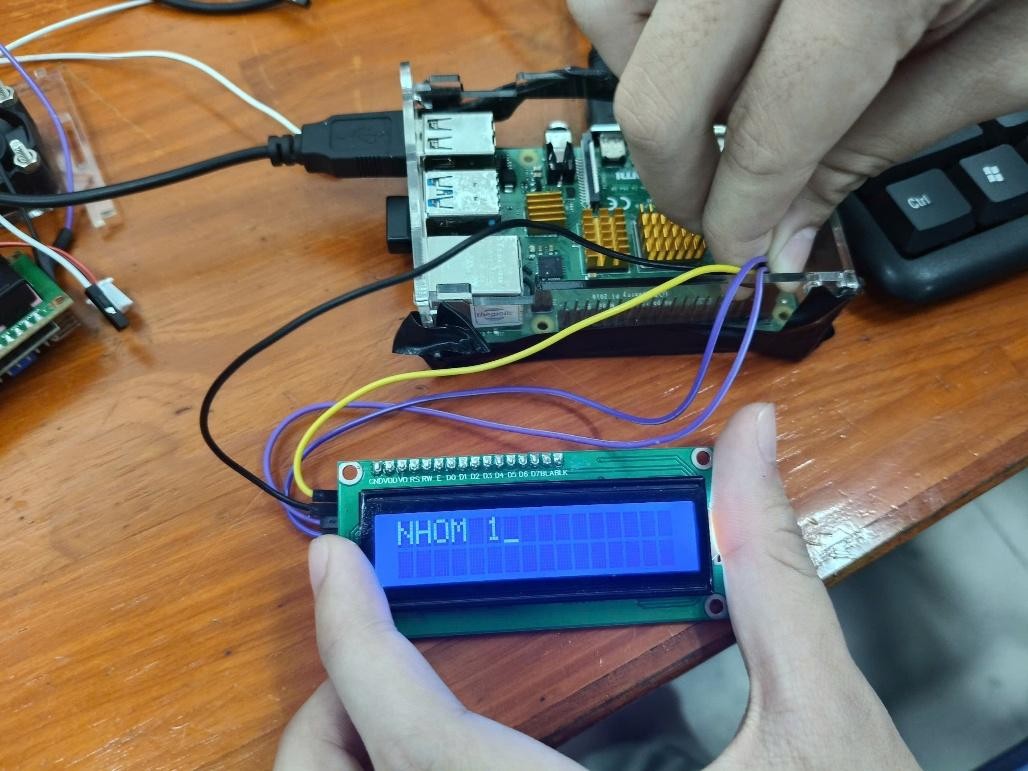


*Hình 4. Các file .ko và out được truyền qua rasPi* 4. Trên terminal của Rasberry:



*Hình 5:Cấp quyền cho các file .ko và file out, rồi thực thi*

5.Kết quả



*Hình 6: Kết quả hiển thị trên LCD*