

TÀI LIỆU THÍ NGHIỆM IT4425
PHÁT TRIỂN PHẦN MỀM NHÚNG THÔNG MINH
2022

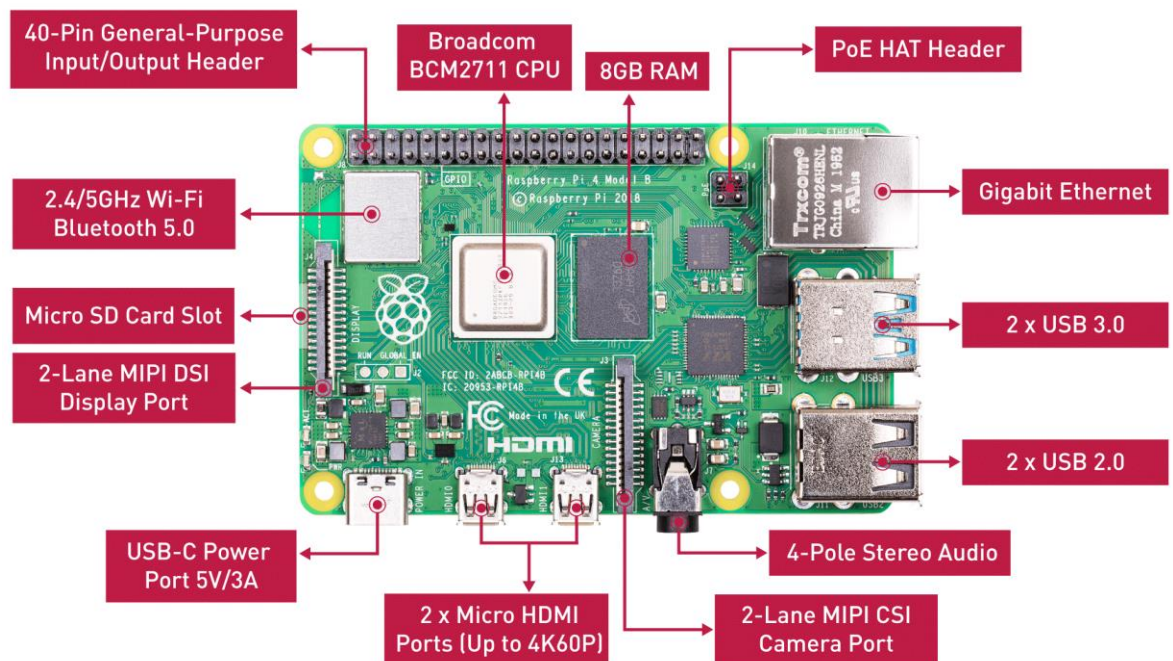
Bài mở đầu

CÀI ĐẶT VÀ SỬ DỤNG THIẾT BỊ

1. Cài đặt các công cụ

Các bài thí nghiệm phát triển phần mềm nhúng được thực hiện trên thiết bị nhúng Raspberry Pi 4 (nền tảng ARM Linux). Cần cài đặt chuẩn bị trước trên máy tính (Windows hoặc Ubuntu) các công cụ sau để phục vụ thực hiện các bài thí nghiệm:

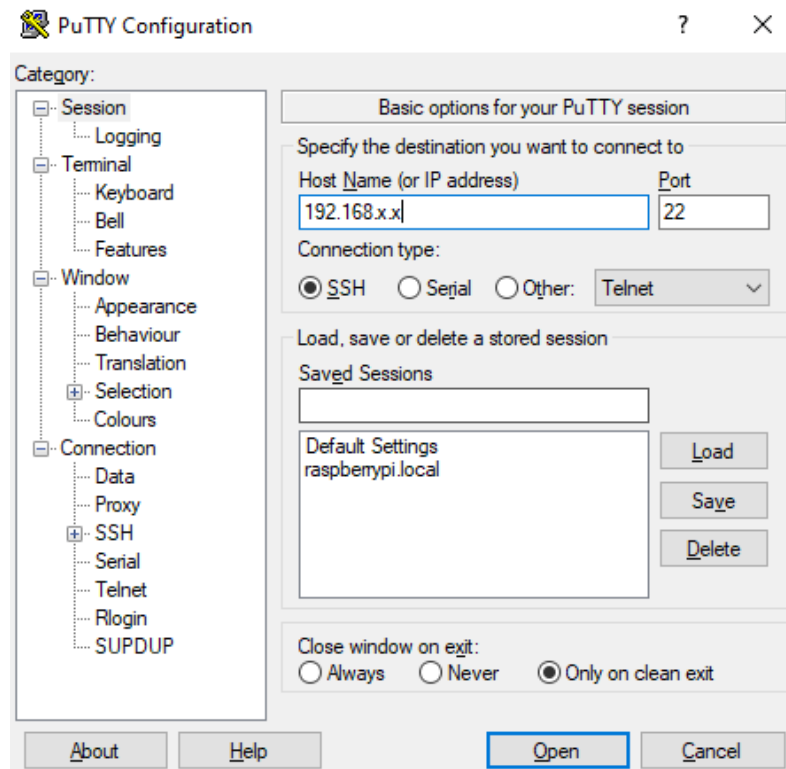
- Phần mềm PuTTY hoặc OpenSSH (Windows) hoặc openssh-server (Linux) – truy cập remote tới Pi
- VNC Viewer – sử dụng giao diện đồ họa trên Pi (không bắt buộc)
- Visual Studio Code – trình soạn thảo code với các extension cần thiết như là C/C++, Remote – SSH



2. Hướng dẫn sử dụng

2.1. Truy cập remote tới Raspberry Pi

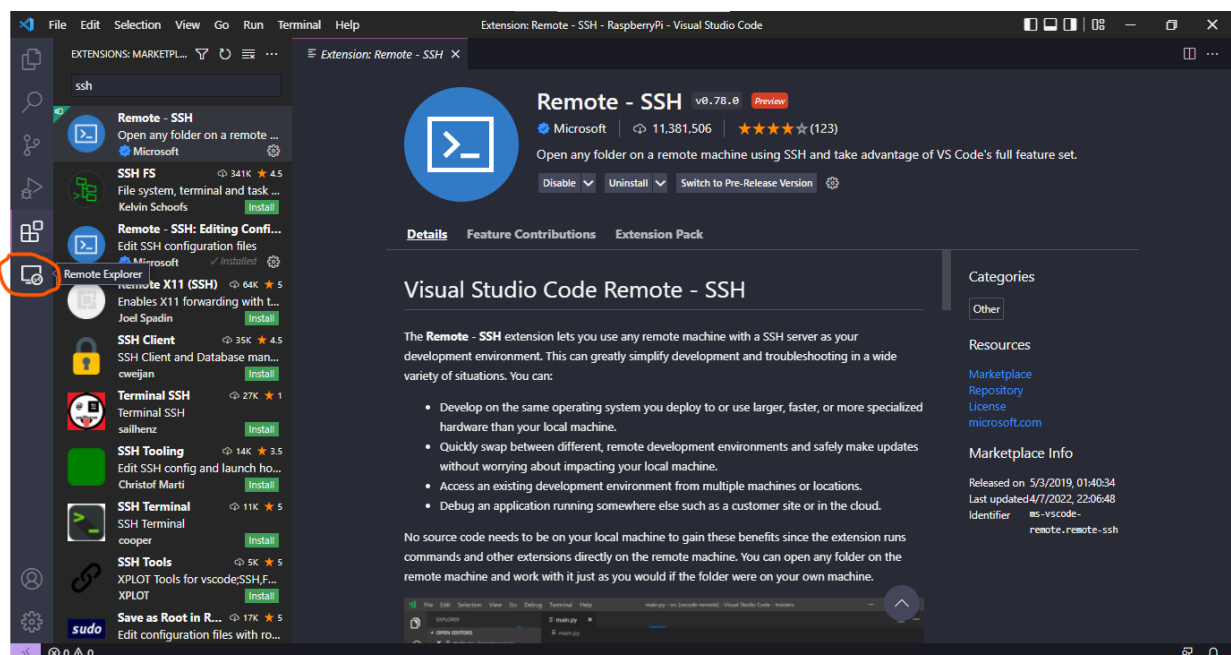
- Kết nối tới Pi bằng SSH với địa chỉ tĩnh như sau
 - o ssh pi@<địa chỉ IP>
 - o Login : pi
 - o Password : 1
- Sử dụng PuTTY cũng với cách tương tự, nhập địa chỉ IP và kết nối theo mặc định (SSH ở port 22) và chọn Open



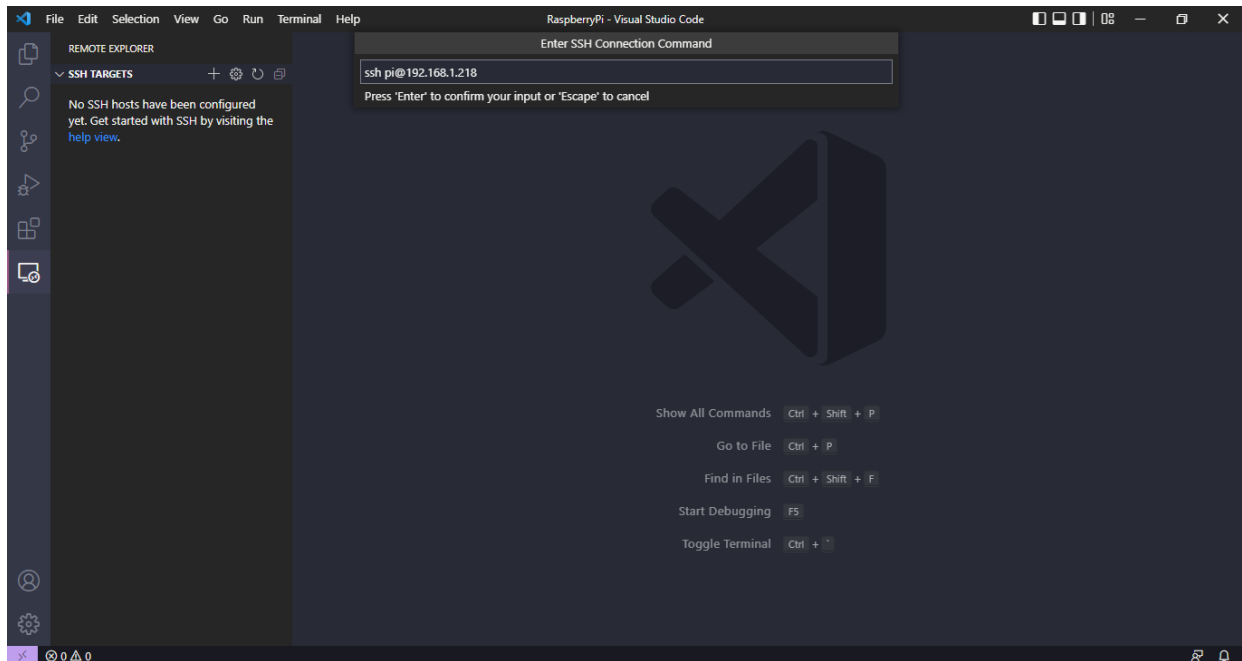
- Trong trường hợp có kết nối mạng LAN (giữa Pi và máy tính) và không cần quan tâm tới địa chỉ IP, nhập Host Name như sau : raspberrypi.local

2.2. Sử dụng Visual Studio Code để viết code, nạp, debug chương trình trên Pi

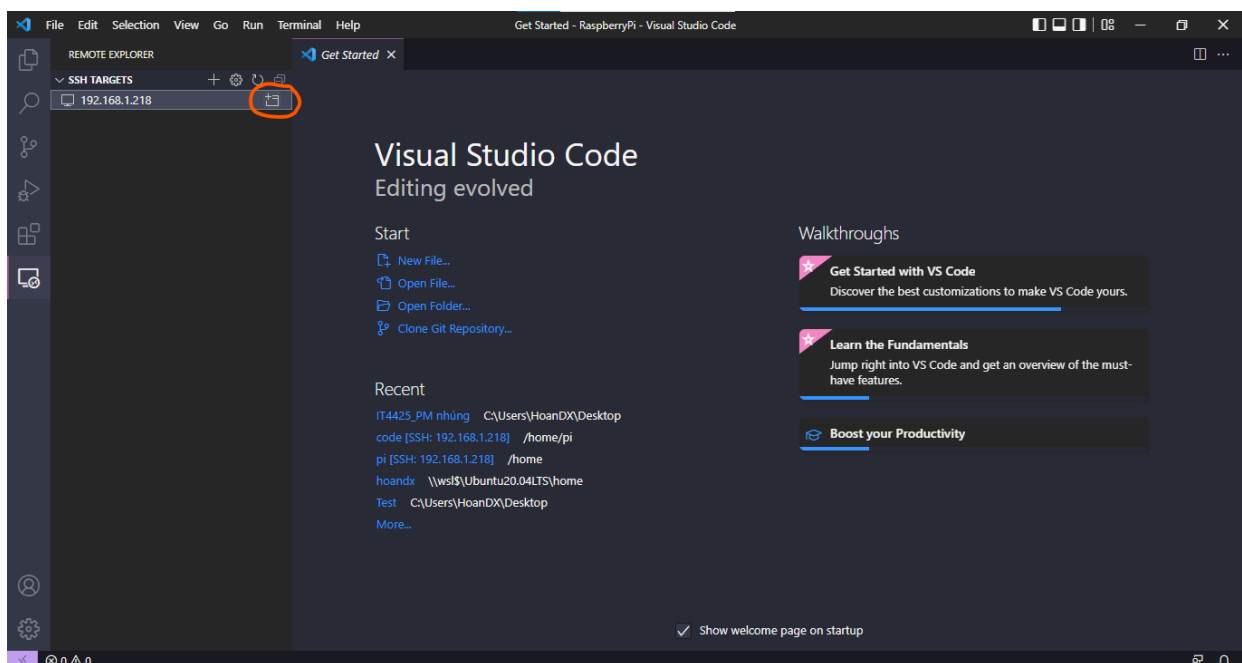
- Sau khi cài đặt extension Remote – SSH trên VS Code xong, tab Remote Explorer sẽ hiển thị ở thanh công cụ



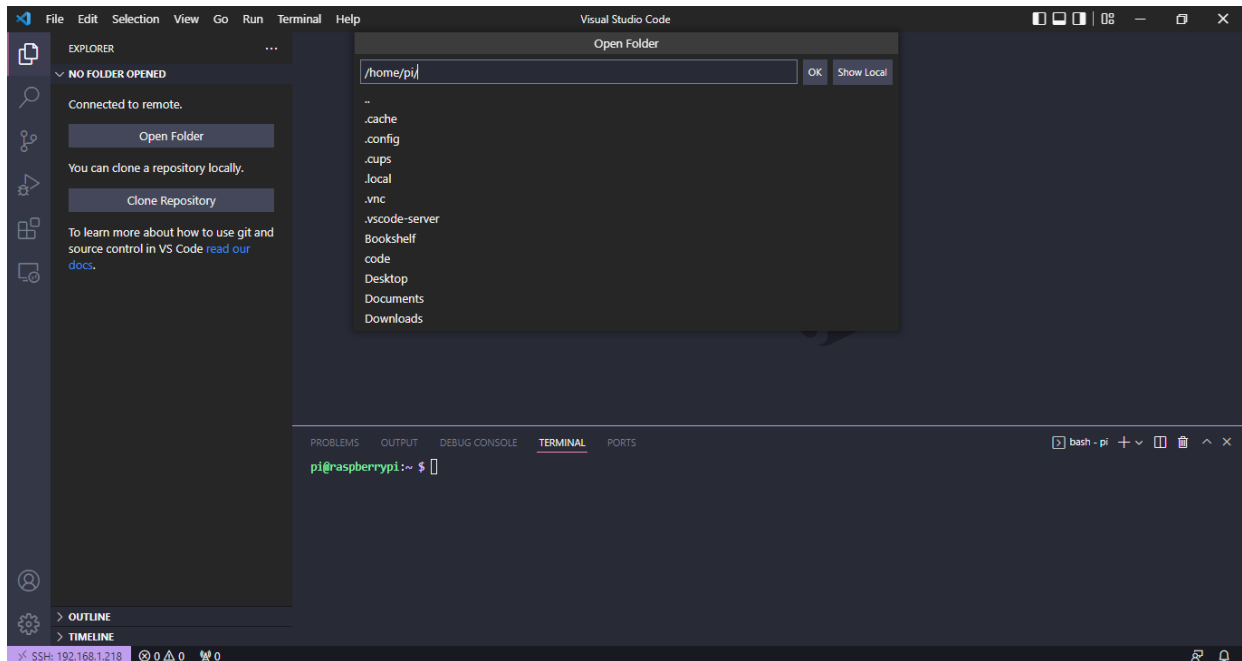
- Chọn dấu + để thêm kết nối SSH mới và nhập câu lệnh tương tự như kết nối SSH thông thường



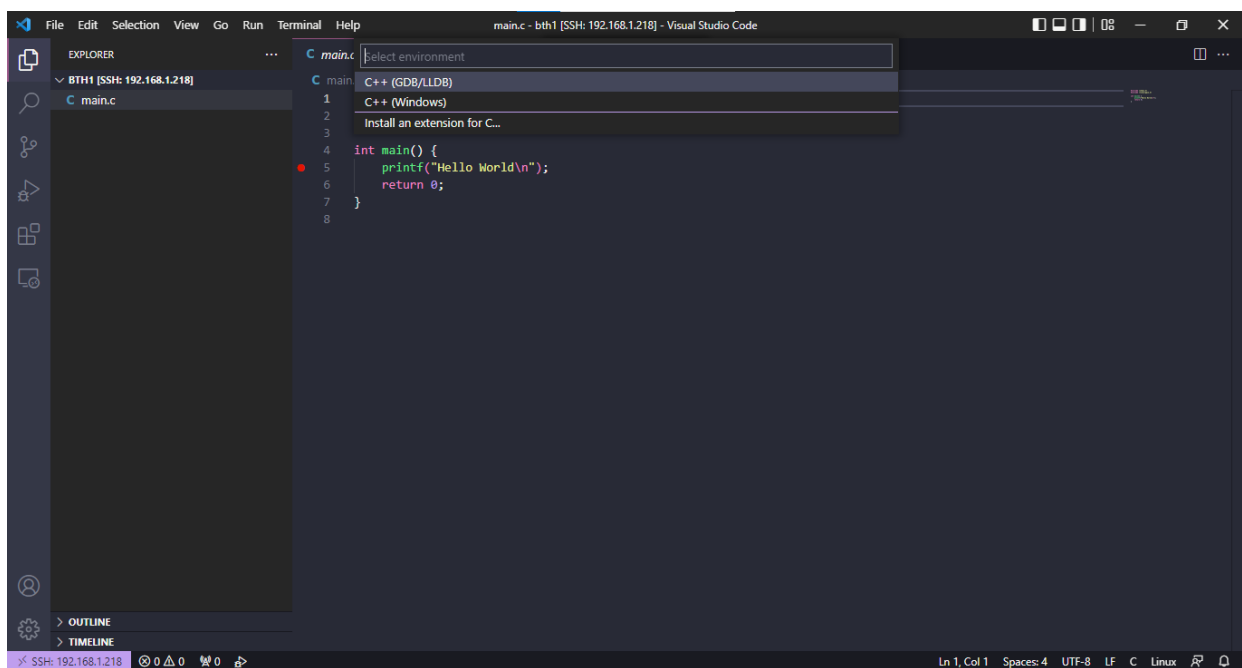
- Sau khi nhập câu lệnh thành công, kết nối sẽ được khởi tạo như trong hình. Để bắt đầu phiên làm việc ở một cửa sổ mới, chọn Connect to Host in New Window.



- Giờ có thể sử dụng VS Code để remote tới Pi và sử dụng như bình thường. Để mở thư mục hoặc mở file trong Pi, chọn Open Folder hoặc Ctrl + K Ctrl + O

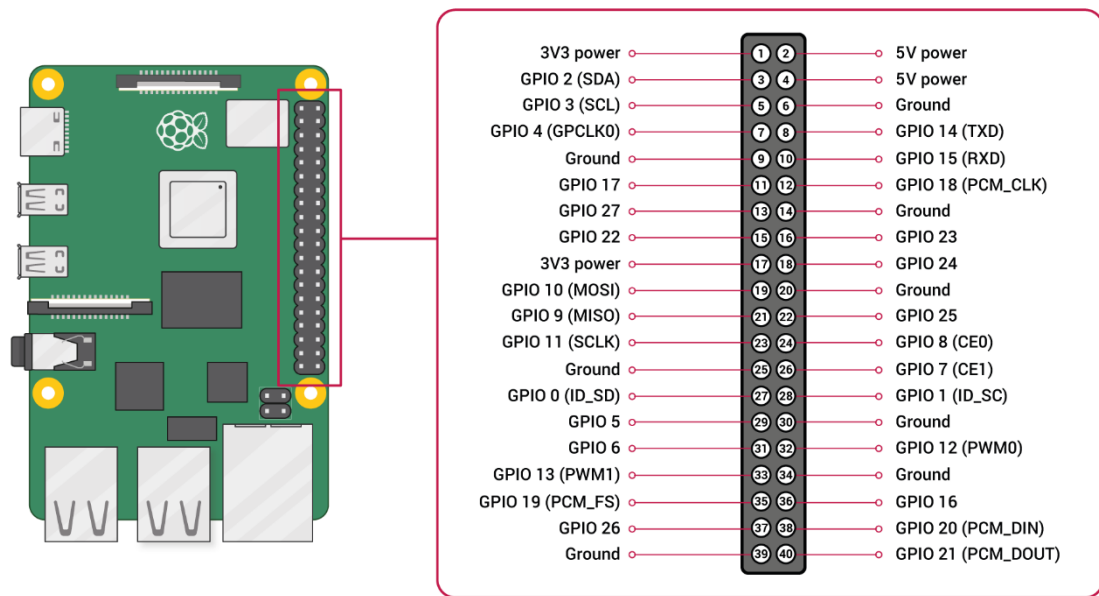


- Trong trường hợp muốn debug trên VS Code, nhấn F5 và chọn trình gỡ lỗi là GDB và phiên bản mới nhất của GCC cho C/C++



2.3. Cấu hình GPIO của Pi

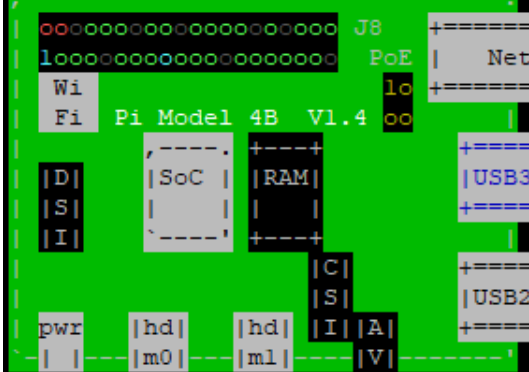
- Các chân GPIO của Raspberry Pi 4 được mô tả bằng hình bên dưới



- Hoặc có thể tra cứu trực tiếp bằng cách chạy câu lệnh sau trên Raspberry Pi 4 để hiển thị sơ đồ GPIO

pi@raspberrypi: ~ \$ pinout

```
pi@raspberrypi:~ $ pinout
```



```
Revision           : c03114
SoC                : BCM2711
RAM               : 4GB
Storage           : MicroSD
USB ports         : 4 (of which 2 USB3)
Ethernet ports    : 1 (1000Mbps max. speed)
Wi-fi             : True
Bluetooth         : True
Camera ports (CSI) : 1
Display ports (DSI) : 1
```

J8:

3V3	(1)	(2)	5V
GPIO2	(3)	(4)	5V
GPIO3	(5)	(6)	GND
GPIO4	(7)	(8)	GPIO14
GND	(9)	(10)	GPIO15
GPIO17	(11)	(12)	GPIO18
GPIO27	(13)	(14)	GND
GPIO22	(15)	(16)	GPIO23
3V3	(17)	(18)	GPIO24
GPIO10	(19)	(20)	GND
GPIO9	(21)	(22)	GPIO25
GPIO11	(23)	(24)	GPIO8
GND	(25)	(26)	GPIO7
GPIO0	(27)	(28)	GPIO1
GPIO5	(29)	(30)	GND
GPIO6	(31)	(32)	GPIO12

GPIO13	(33)	(34)	GND
GPIO19	(35)	(36)	GPIO16
GPIO26	(37)	(38)	GPIO20
GND	(39)	(40)	GPIO21

POE :

TR01	(1)	(2)	TR00
TR03	(3)	(4)	TR02

For further information, please refer to <https://pinout.xyz/>

```
pi@raspberrypi:~ $
```

Bài thí nghiệm 1

GIAO TIẾP GPIO TRONG PHẦN MỀM NHÚNG NỀN TẢNG ARM LINUX

1. Mục đích

- Nắm được kỹ năng lập trình, phát triển phần mềm trên hệ nhúng nền tảng ARM Linux, thực hiện các giao tiếp vào ra cơ bản GPIO, thực hiện trên Raspberry Pi 4.

2. Chuẩn bị

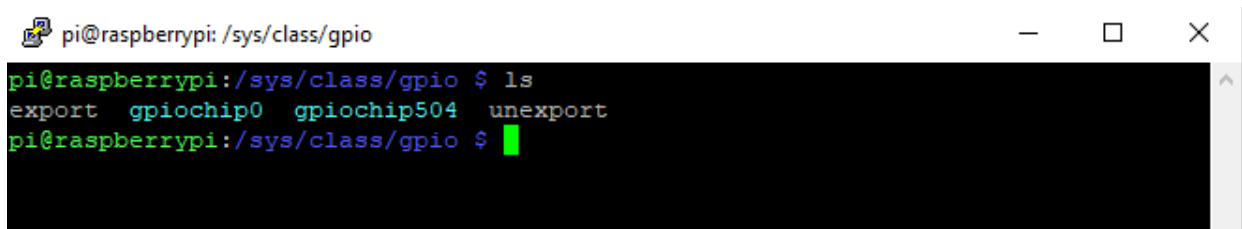
- Máy tính Windows hoặc Ubuntu với các công cụ cần thiết (trong bài mở đầu)
- KIT phát triển nền tảng ARM Linux: Raspberry Pi 4

3. Tiến hành

Để thực hiện giao tiếp vào ra GPIO trên phần mềm nhúng, thường sử dụng 2 cơ chế sau:

- **Cách 1.** Sử dụng gpio driver (có sẵn trên hệ điều hành, hoặc cần xây dựng gpio driver tương ứng), chương trình trên tầng ứng dụng sẽ giao tiếp GPIO thông qua driver này.
- **Cách 2.** Nền tảng ARM Linux cung cấp giao diện gpio sysfs (gpiolib) cho phép theo tác với các chân vào ra trực tiếp từ không gian người dùng (user space). Chương trình trên tầng ứng dụng sẽ giao tiếp gpio thông qua giao diện gpiolib này.

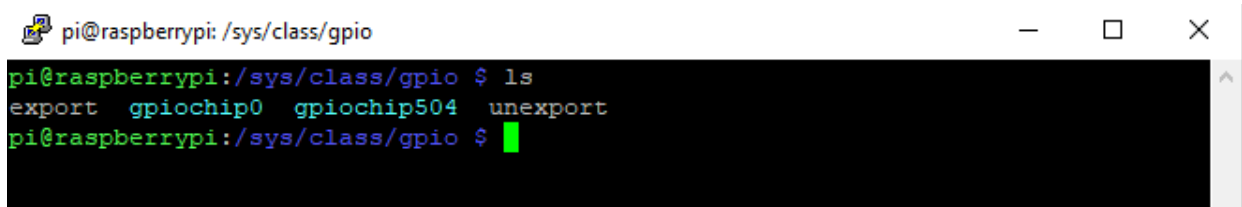
Giao diện điều khiển gpio sysfs nằm trong thư mục /sys/class/gpio. Kiểm tra bằng lệnh `ls /sys/class/gpio`



```
pi@raspberrypi: /sys/class/gpio
pi@raspberrypi:/sys/class/gpio $ ls
export  gpiochip0  gpiochip504  unexport
pi@raspberrypi:/sys/class/gpio $
```

Trong trường hợp này (với linux kernel 5.15.34-v71+ trên Raspberry Pi 4), gpio sysfs interface gồm các files:

- export
- unexport
- gpiochip0, gpiochip504
- Các gpiochip này tương ứng với các GPIO port của CPU, mỗi file gpiochip quản lý 32 pin GPIOs (32 chân/1 port)
- (GPIOA ↔ gpiochip0, GPIOB ↔ gpiochip32, GPIOF ↔ gpiochip160, ...)



```
pi@raspberrypi: /sys/class/gpio
pi@raspberrypi:/sys/class/gpio $ ls
export  gpiochip0  gpiochip504  unexport
pi@raspberrypi:/sys/class/gpio $
```

Việc giao tiếp điều khiển input/output với mỗi chân gpio sẽ trở nên dễ dàng với các thao tác đọc/ghi file tương ứng do giao diện lập trình cung cấp. Có thể sử dụng lập trình c/c++ (hàm read/write), có thể sử dụng lập trình shell linux (echo, cat, ...)

Bài 3.1. Sử dụng gpiolib bằng các lệnh của Linux

Yêu cầu: Remote vào Raspberry Pi, thực hiện các bước sau bằng dòng lệnh

Bước 1. Exporting a GPIO (đăng ký để sử dụng chân gpio từ user space)

Trước khi có thể sử dụng 1 chân gpio, cần export ra không gian người dùng

Để export 1 chân gpio, ghi số hiệu (ID number) của nó vào file /sys/class/gpio/export

Ví dụ: **Để export chân GPIO2: (base on gpiochip0)**

```
echo 2 > /sys/class/gpio/export
```

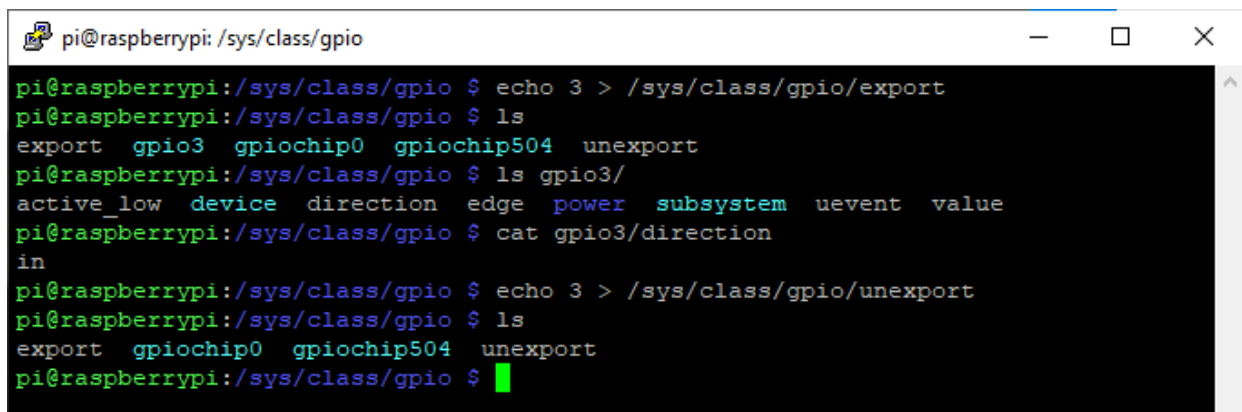
Để export chân GPIO3:

```
echo 3 > /sys/class/gpio/export
```

Sau khi export một chân gpio, nó sẵn sàng sử dụng qua file /sys/class/gpio/gpio[ID] (ví dụ /sys/class/gpio/gpio3).

Nếu không cần sử dụng nữa, có thể giải phóng bằng cách ghi số hiệu ID của nó vào file /sys/class/gpio/unexport

```
echo 37 > /sys/class/gpio/unexport
```



```
pi@raspberrypi: /sys/class/gpio
pi@raspberrypi:/sys/class/gpio $ echo 3 > /sys/class/gpio/export
pi@raspberrypi:/sys/class/gpio $ ls
export  gpio3  gpiochip0  gpiochip504  unexport
pi@raspberrypi:/sys/class/gpio $ ls gpio3/
active_low  device  direction  edge  power  subsystem  uevent  value
pi@raspberrypi:/sys/class/gpio $ cat gpio3/direction
in
pi@raspberrypi:/sys/class/gpio $ echo 3 > /sys/class/gpio/unexport
pi@raspberrypi:/sys/class/gpio $ ls
export  gpiochip0  gpiochip504  unexport
pi@raspberrypi:/sys/class/gpio $
```

Bước 2. Cấu hình chế độ vào ra (input/output) cho gpio pin

Cấu hình chân gpio là input/output bằng cách ghi giá trị in/out đến file direction của chân tương ứng

```
/sys/class/gpio/gpio[ID]/direction
```

Ví dụ: Thiết lập gpio3 là chân output

```
$ echo "out" > /sys/class/gpio/gpio3/direction
```

hoặc là chân input

```
$ echo "in" > /sys/class/gpio/gpio3/direction
```

Hoặc sử dụng giá trị:

"high" để cấu hình là chân output với giá trị khởi tạo là 1

"low" để cấu hình là chân output với giá trị khởi tạo là 0

Bước 3. Truy cập giá trị của chân gpio

Giá trị của một chân gpio có thể đọc/ghi bằng cách đọc/ghi file `/sys/class/gpio/gpio[ID]/value`.

Ví dụ: Cấu hình chân GPIO3 output, và xuất giá trị 0 ra chân này

```
echo 3 > /sys/class/gpio/export  
echo "out" > /sys/class/gpio/gpio3/direction  
echo 0 > /sys/class/gpio/gpio3/value
```

Bài 3.2. Xây dựng thư viện c gồm các hàm giao tiếp gpio

Yêu cầu: Sử dụng mã nguồn tham khảo sau, xây dựng một thư viện cung cấp các hàm để truy cập gpio sử dụng giao diện /sys/class/gpio

File mã nguồn gpio.h

```
#ifndef GPIO_H  
#define GPIO_H  
int gpio_export(unsigned gpio); //Hàm export pin ra user space  
int gpio_unexport(unsigned gpio); //Hàm giải phóng pin khi không còn sử dụng  
int gpio_dir_out(unsigned gpio); //Cấu hình pin là output  
int gpio_dir_in(unsigned gpio); //Cấu hình pin là input
```

```
int gpio_value(unsigned gpio, unsigned value); //Đọc/ghi giá trị của
pin gpio
#endif
```

File mã nguồn gpio.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#define GPIO_DIR_IN    0
#define GPIO_DIR_OUT   1
/*****
Hàm đăng ký (export) chân gpio muốn sử dụng ra không gian người dùng
*****/
int gpio_export(unsigned gpio)
{
    int fd, len;
    char buf[11];

    fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("gpio/export");
        return fd;
    }

    len = snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, len); //Ghi số hiệu (ID) pin muốn sử dụng vào
file /sys/class/gpio/export khi đăng ký sử dụng
    close(fd);
    return 0;
}
/*****
Hàm giải phóng (unexport) chân gpio khi không còn sử dụng
*****/
int gpio_unexport(unsigned gpio)
{
    int fd, len;
    char buf[11];
    fd = open("/sys/class/gpio/unexport", O_WRONLY);
    if (fd < 0) {
        perror("gpio/export");
        return fd;
    }
    len = snprintf(buf, sizeof(buf), "%d", gpio);
    write(fd, buf, len); //Ghi số hiệu (ID) pin muốn sử dụng vào
file /sys/class/gpio/unexport khi giải phóng
    close(fd);
    return 0;
}
```

```

}
/*****
Hàm cấu hình chân gpio là in hay out (dir)
*****/

int gpio_dir(unsigned gpio, unsigned dir)
{
    int fd, len;
    char buf[60];
    len = snprintf(buf, sizeof(buf),
"/sys/class/gpio/gpio%d/direction", gpio);
    fd = open(buf, O_WRONLY);
    if (fd < 0) {
        perror("gpio/direction");
        return fd;
    }
    //Cấu hình pin là input/output bằng cách ghi giá trị (ASCII) in,
out vào file /sys/class/gpio/gpio[ID]/direction
    if (dir == GPIO_DIR_OUT)
        write(fd, "out", 4);
    else
        write(fd, "in", 3);
    close(fd);
    return 0;
}
/*****
Hàm thiết lập chân gpio out
*****/
int gpio_dir_out(unsigned gpio)
{
    return gpio_dir(gpio, GPIO_DIR_OUT); //trường hợp là output
}
/*****
Hàm thiết lập chân gpio in
*****/

int gpio_dir_in(unsigned gpio)
{
    return gpio_dir(gpio, GPIO_DIR_IN); //trường hợp là input
}
/*****
Hàm ghi ra trị ra chân gpio out (tương ứng xuất 0 , 1)
*****/

int gpio_value(unsigned gpio, unsigned value)
{
    int fd, len;
    char buf[60];
    len = snprintf(buf, sizeof(buf), "/sys/class/gpio/gpio%d/value",
gpio);
    fd = open(buf, O_WRONLY);

```

```

    if (fd < 0) {
        perror("gpio/value");
        return fd;
    }
    //Xuất giá trị 1, 0 bằng cách ghi ra file value tương ứng với
pin đã cấu hình
    if (value)
        write(fd, "1", 2);
    else
        write(fd, "0", 2);
    close(fd);
    return 0;
}
/*****
Chương trình chính để test các hàm này
*****/
#ifdef 1
int main(int argc, char *argv[])
{
    int i = 20;
    int pin_no = 2; //Sử dụng chân GPIO2
    gpio_export(pin_no);
    gpio_dir_out(pin_no);
    while (i-- > 0) {
        gpio_value(pin_no, i & 1); //toggle on GPIO2
        sleep(1);
    }
    gpio_unexport(pin_no);
}
#endif

```

- Tạo 1 Makefile để hỗ trợ biên dịch chương trình

Lưu File Makefile.txt đặt cùng thư mục với file mã nguồn ở trên

```

CC := gcc
CFLAGS := -Wall -c
SRCS := gpio.c
OBJS := $(SRCS:.c=.o)
TARGET := gpio

all: $(TARGET)
$(TARGET): $(OBJS)
    ${CC} gpio.c -o gpio
gpio.o: gpio.c
    ${CC} ${CFLAGS} gpio.c
.PHONY: clean
clean:
    rm -f $(OBJS)

```

- Để biên dịch chương trình, chạy câu lệnh sau trong cùng thư mục với các tệp mã nguồn ở trên:

```
pi@raspberrypi: ~/code/bth1 $ make
pi@raspberrypi: ~/code/bth1 $ ./gpio
```

- Câu lệnh **make** sẽ chạy tệp Makefile chứa các câu lệnh biên dịch chương trình và build ra tệp thực thi gpio. Mặc định, make sẽ chạy rule đầu tiên là all để biên dịch các tệp mã nguồn thành tệp object và từ tệp object sẽ biên dịch ra chương trình. Nếu muốn chạy bất kỳ một rule nào, chỉ cần để tên rule đó phía sau câu lệnh make giống như cách đặt tham số và Makefile sẽ chỉ thực thi một rule chỉ định như yêu cầu. Ví dụ: make gpio.o hoặc make clean

- Trong tệp Makefile ở trên, biến SRCS là danh sách chứa các tệp mã nguồn .c của chương trình, biến OBJS chính là các tệp object (.o) tương ứng mong muốn được tạo ra và biến TARGET chính là chương trình thực thi cuối cùng sau khi đã liên kết (Linker) các tệp .o ở trên lại.

- **clean** là một rule khá phổ biến trong một tệp Makefile. Nó được sinh ra với mục đích đơn giản là để dọn dẹp các tệp được ra ra trong quá trình biên dịch mà không còn sử dụng đến nữa.

Bài 3.3. Sử dụng Java để giao tiếp với GPIO sử dụng giao diện /sys/class/gpio

Thực hiện viết chương trình bật/tắt đèn đơn giản bằng Java chạy trên Raspberry Pi cũng bằng phương thức tương tự như trên.

GpioLed.java

```
import java.io.File;
import java.io.FileWriter;

import static java.lang.System.*;

public class GpioLed {
    /* Hàm main để khởi chạy chương trình */
    public static void main(String[] args) {
        int value, gpioNo;
        if (args.length != 2) {
            System.out.println("error");
            exit(1);
        }
        try {
            gpioNo = Integer.parseInt(args[0]);
            value = Integer.parseInt(args[1]);
            if (gpioNo < 0 || gpioNo > 3 || value < 0 || value > 1){
                System.out.println("Invalid Argument Value");
                exit(1);
            }
        }
    }
}
```

```

    }
    /* Khởi tạo một đối tượng GPIO và truyền vào các tham số
dòng lệnh từ bàn phím */
    Gpio gpio = new Gpio(gpioNo, "out", value);
    GpioControl gpioControl = new GpioControl(gpio);
    gpioControl.writeValue(value);
} catch (NumberFormatException ex) {
    ex.printStackTrace();
}
}
}

class Gpio {
    private int gpioNo;
    private String direction;
    private int value;

    public Gpio(int gpioNo, String direction, int value) {
        this.gpioNo = gpioNo;
        this.direction = direction;
        this.value = value;
    }
}

class GpioControl {
    private Gpio gpio;

    public GpioControl(Gpio gpio) {
        this.gpio = gpio;
    }
    /* Phương thức hỗ trợ ghi giá trị 0/1 vào chân GPIO */
    public void writeValue(int value) {
        try {
            File file = new File("/sys/class/gpio/gpio" +
this.gpio.getGpioNo() + "/value");
            FileWriter fw = new FileWriter(file);
            fw.write(Integer.toString(value));
            fw.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

- Để biên dịch mã nguồn trên bằng trình biên dịch Java – **javac**, sử dụng câu lệnh sau:

pi@raspberrypi: ~/code/bth1 \$ javac GpioLed.java

pi@raspberrypi: ~/code/bth1 \$ java GpioLed <gpioNo> <value>

- Với gpioNo và value là tham số truyền vào cho chương trình, lần lượt là số hiệu chân gpio và giá trị 0/1. Ví dụ, để bật led ở gpio số 2 :

pi@raspberrypi: ~/code/bth1 \$ java GpioLed 2 1

Bài 3.4 (Bài tập bổ sung) Viết chương trình điều khiển dãy led đơn với các hiệu ứng khác nhau:

- Flash (nhấp nháy tắt cả led)
- Running (tuần tự từng led từ phải qua trái hoặc ngược lại)
- Spot bumper (đối xứng từ ngoài vào trong)

Tham khảo mã nguồn sau:

```
/* Nhấp nháy led */
int gpio_flash(unsigned int gpio[], int loop)
{
    unsigned char value = 0;
    while (loop--)
    {
        for (int i = 0; i < 6; i++)
        {
            gpio_value(gpio[i], value);
        }
        value = ~value;
        sleep(1);
    }
}

/* Led chạy đối xứng từ ngoài vào trong */
int gpio_spot_bumper(unsigned int gpio[], int loop) {
    unsigned char value = 0;
    while (loop--)
    {
        for (int i = 0; i < 3; i++)
        {
            gpio_value(gpio[i], 1);
            gpio_value(gpio[5 - i], 1);
            if (i == 0) {
                gpio_value(gpio[2], 0);
                gpio_value(gpio[3], 0);
            } else {
                gpio_value(gpio[i - 1], 0);
                gpio_value(gpio[6 - i], 0);
            }
        }
        sleep(1);
    }
}
```



```
/* Led đuổi từ trái sang phải */
int gpio_running(unsigned int gpio[], int loop) {
    unsigned char value = 0;
    while (loop--)
    {
        for (int i = 0; i < 6; i++)
        {
            gpio_value(gpio[i], 1);
            if (i == 0) {
                gpio_value(gpio[5], 0);
            } else {
                gpio_value(gpio[i - 1], 0);
            }
            sleep(1);
        }
    }
}

/*****
Chương trình chính để test các hàm này
*****/
#if 1
int main(int argc, char *argv[])
{
    int loop = 5;
    unsigned int gpio[6] = {2, 3, 17, 22, 27, 23}; /* Khai báo các
số hiệu chân sử dụng */
    for (int i = 0; i < 6; i++)
    {
        gpio_export(gpio[i]);
        gpio_dir_out(gpio[i]);
    }
    gpio_spot_bumber(gpio, loop);
    for (int i = 0; i < 6; i++)
    {
        gpio_unexport(gpio[i]);
    }
}
#endif
```

- Việc biên dịch chương trình trên cũng có thể thông qua tệp Makefile đã được tạo ra trước đó, chỉ cần thay đổi tên tệp .c, .h và .o

4. Báo cáo và đánh giá thực hiện thí nghiệm

- Sinh viên viết báo cáo thí nghiệm (file .docx) mô tả: Nội dung các bài tập đã thực hiện (gồm 4 bài), kèm mã nguồn, chụp ảnh màn hình kết quả thực hiện
- Kiểm tra kết quả thực hiện bài thí nghiệm trên KIT tại buổi thực hành