

Hello Raspberry! Experiment

Introduction

After a programmer retired, he liked calligraphy. One day, he suddenly had an aesthetic mood after a meal, so he prepared the "Four Treasures of the Study", Mao, Ink, Paper, and Research, with Wang Xizhi's demeanor and Yan Zhenqing's attitude. After a moment of contemplation, write down carefully: "Hello world!". Why are programmers so keen on these words? The birth of "Hello world" can be traced back to 1972. The famous researcher Brian Kernighan of Bell Laboratory first used it (program) when writing "Tutorial the Introduction to the Language B". This is the earliest "Hello" and "word" are used together in computer work. Then, in 1978, he used this sentence pattern "hello, world" again in the C language bible "C programming language", co-authored with Dennis Ritchie, as the first program of the opening ceremony. In this program, the output of "hello, world" is all lowercase, without an exclamation mark, and a comma followed by a space. Since then, "Hello, World" has become a tradition for the world to welcome the outside world. "Hello RaspberryPi!", without exception, became the first program in the tutorial

connection

Use the power cord to directly power the Raspberry Pi

C++ program

```
#include <stdio.h>
#include <wiringPi.h>
using namespace std;

int main ()
{
    printf("Hello RaspberryPi !\n");
    while(1);
}
```

Python program

```
print("Hello RaspberryPi !")
```

Java program

```
public class HelloRaspberryPi{
    public static void main(String[] args){
        System.out.println("HelloRaspberryPi ! ");
    }
}
```

Run the program and observe the experimental results

1. Open the VNC connection on the computer;
2. Open the VNC software and click "File—New connection, as shown in Figure 3.1.1

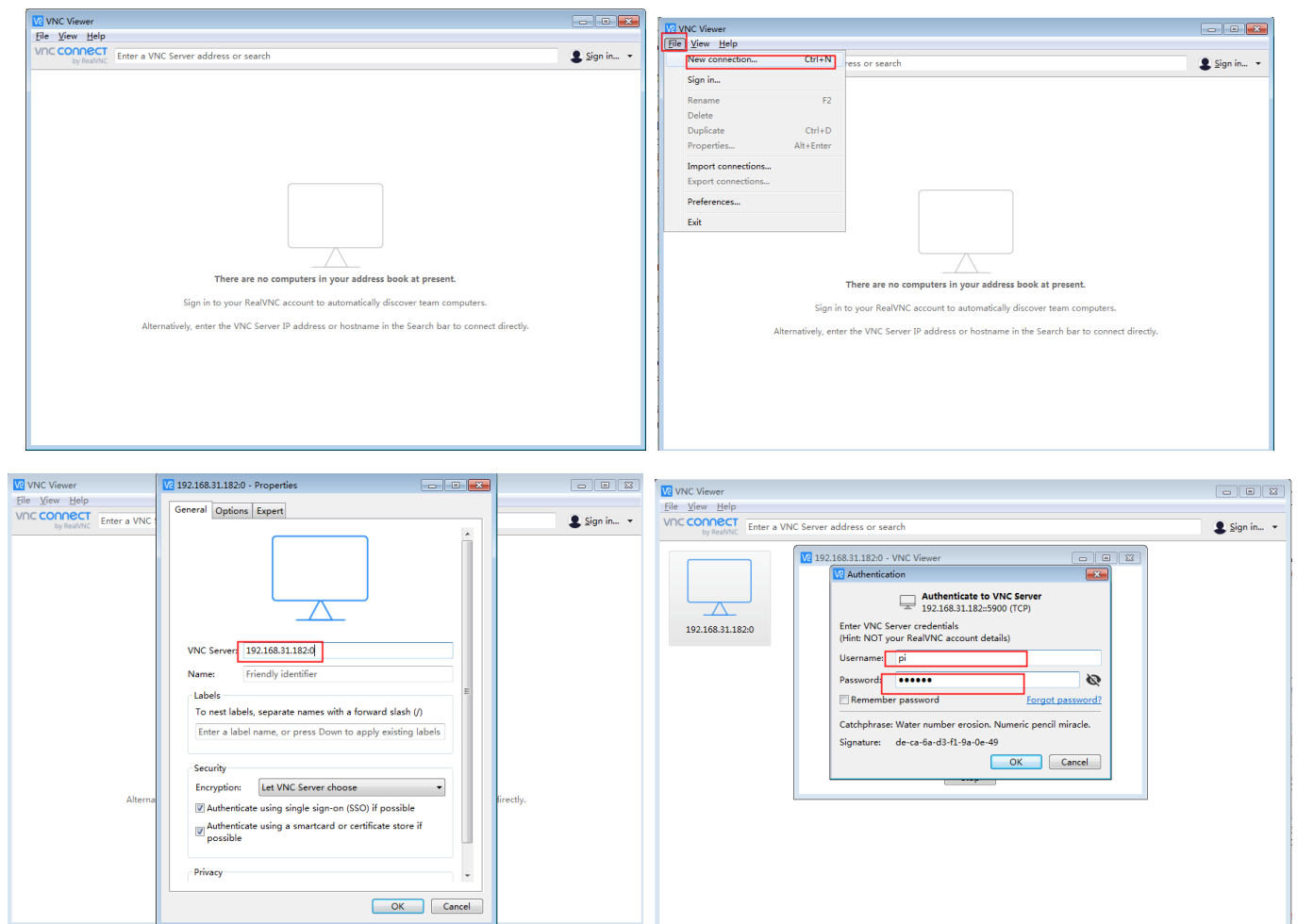


Figure 3.1.1 New VNC connection diagram

- 1) Open the Geany software, as shown in Figure 3.1.1

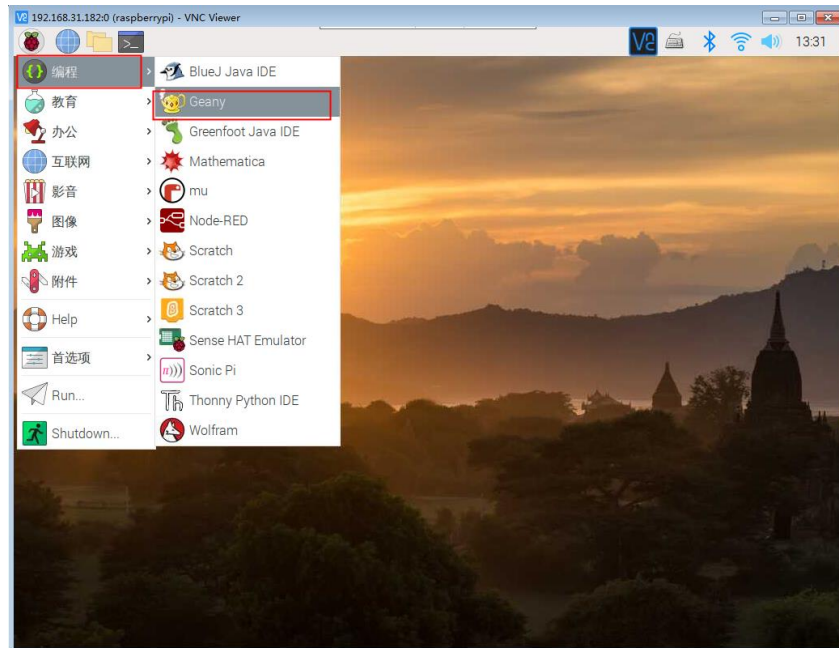


Figure 3.1.2 Geany software

2) Click "File-Open"

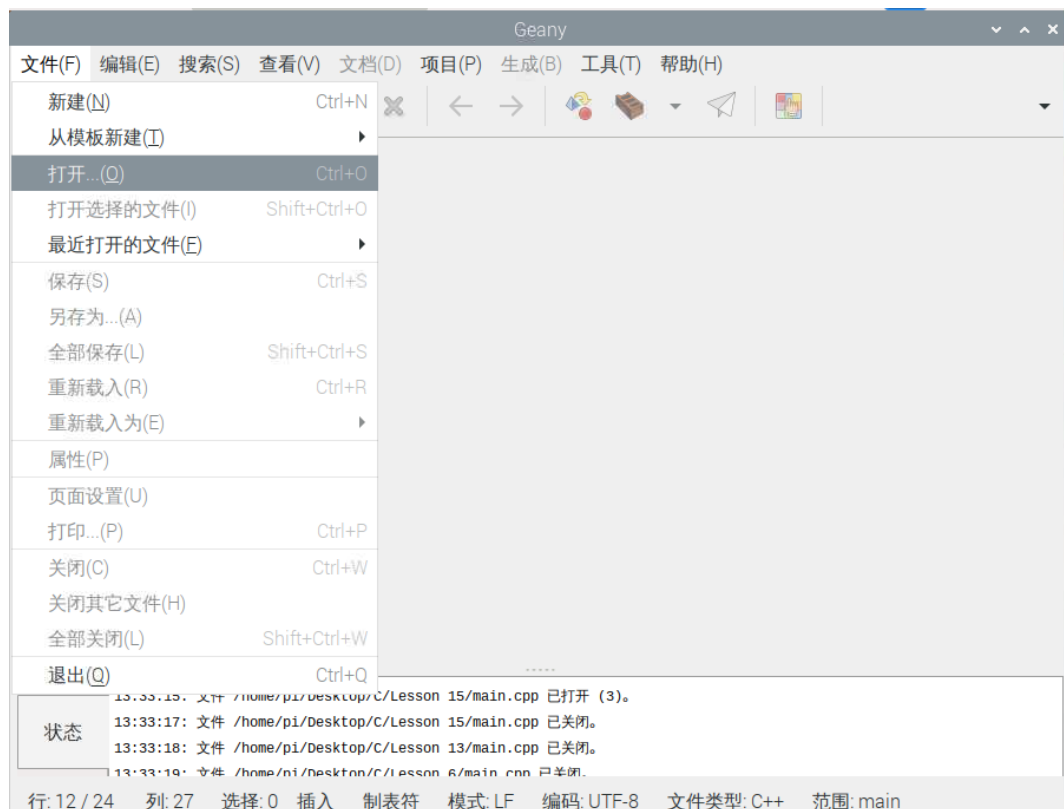


Figure 3.1.3 Programmer selection

3) Select the CPP file and open the program

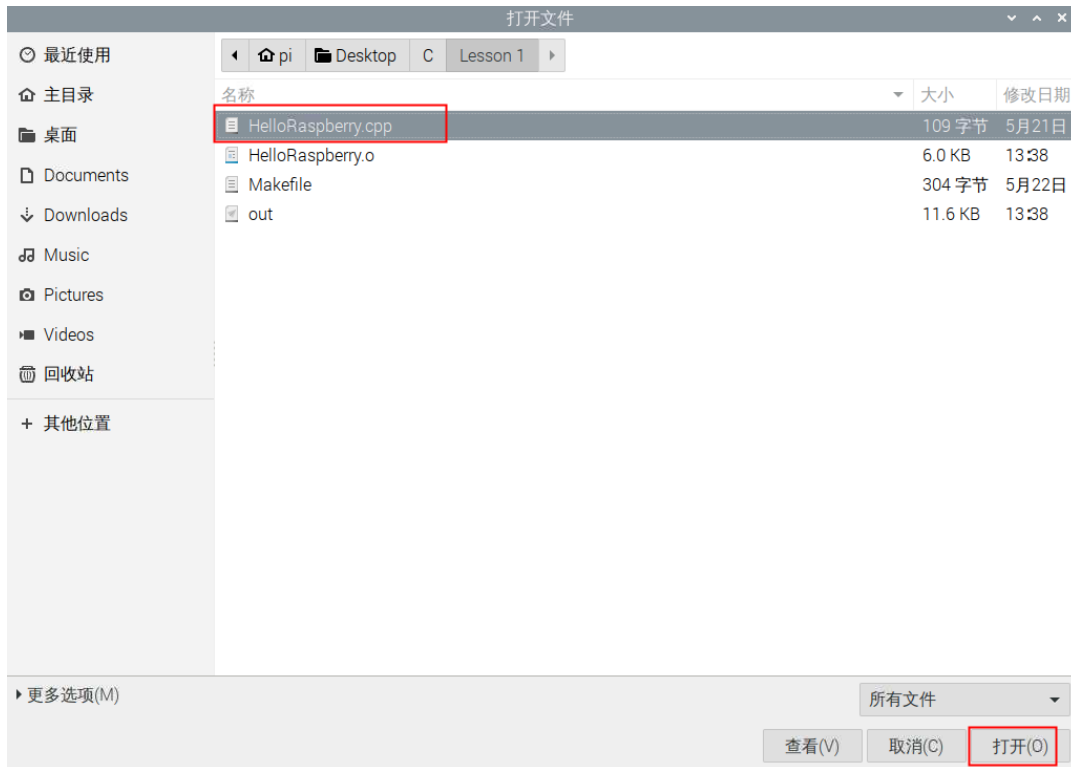


Figure 3.1.4 Open the program

- 4) Select build (B)-build (M) to show that the compilation is completed successfully. As shown in Figure 3.1.6

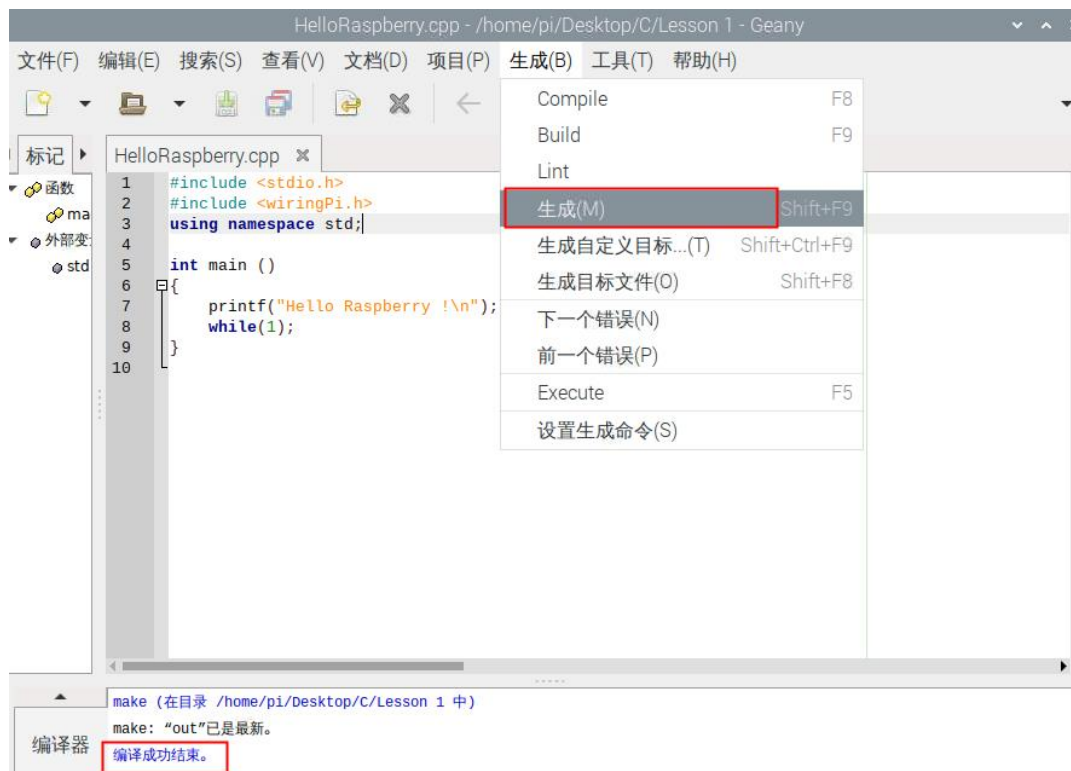


Figure 3.1.5 Compile to generate executable file

- 5) Find the out file in the file, double-click to run it, select Execute in the terminal emulator, and you will see the serial port continuously printing "Hello RaspberryPi!"

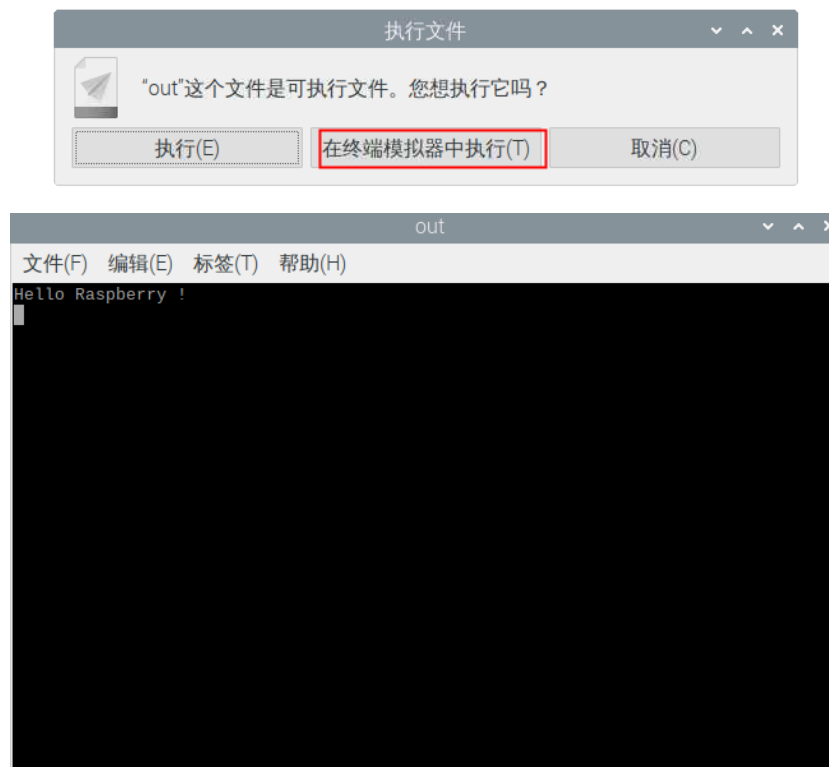


Figure 3.1.6 Terminal display

- 6) Open the Thonny Python IDE software, as shown in Figure 3.1.7

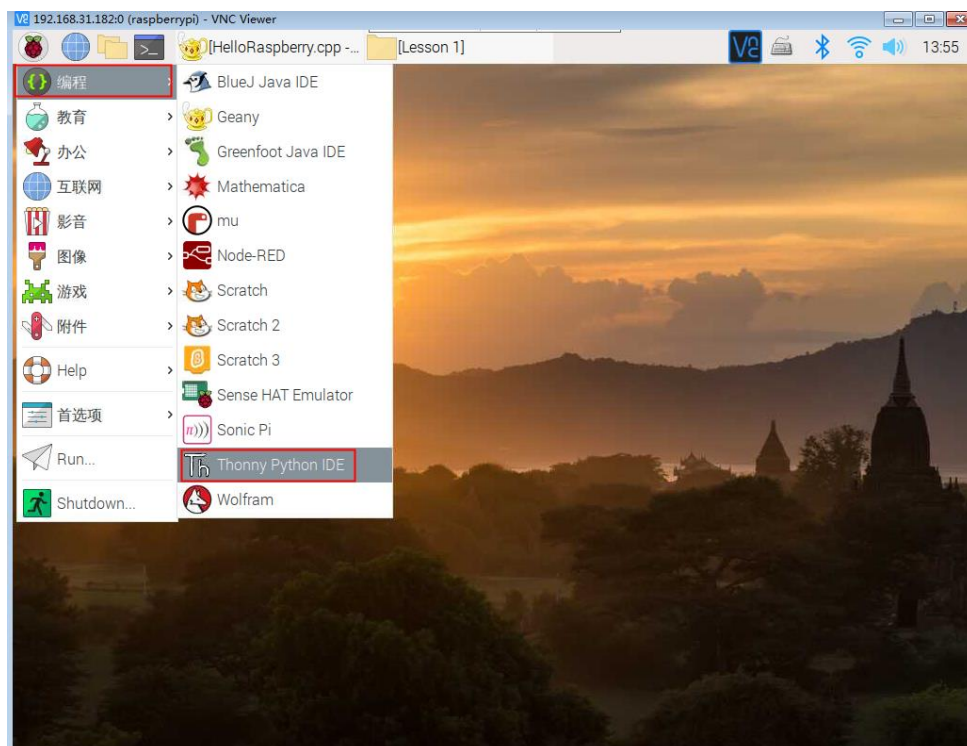


Figure 3.1.7 Thonny Python IDE software

7) Click "Load" to load the Python file

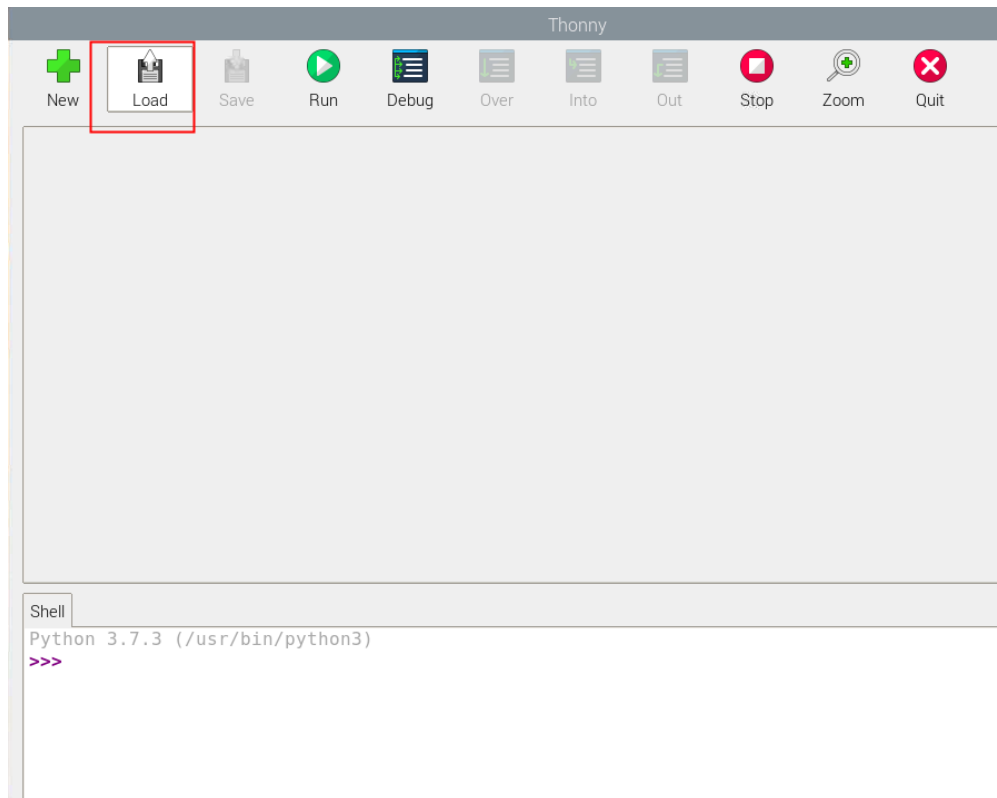


Figure 3.1.8 Programmer selection

8) Select the ".py" file and open the program

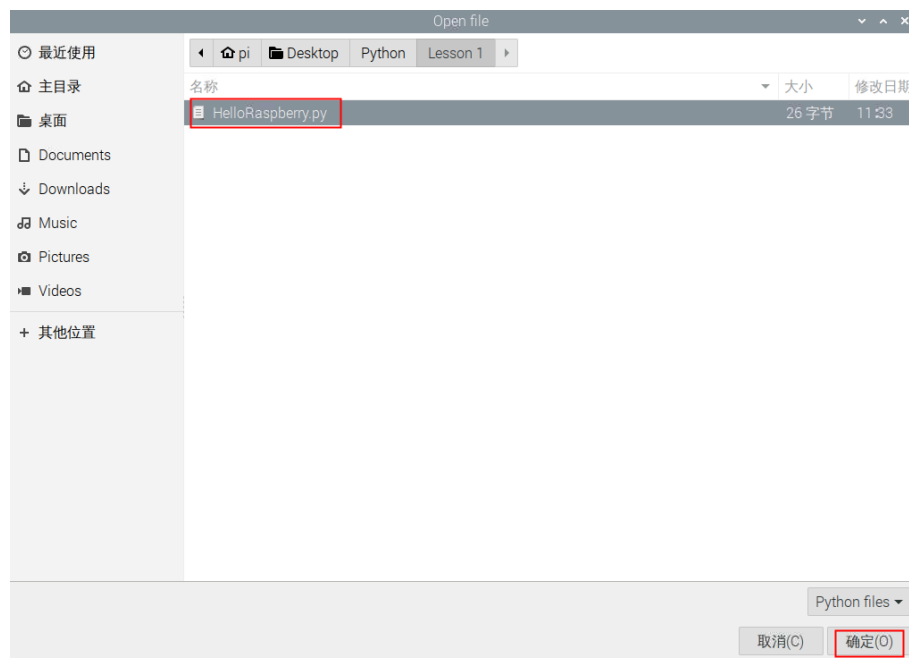


Figure 3.1.9 Open the program

9) Select Run, and see the running result in the Shell below. As shown in Figure 3.1.6

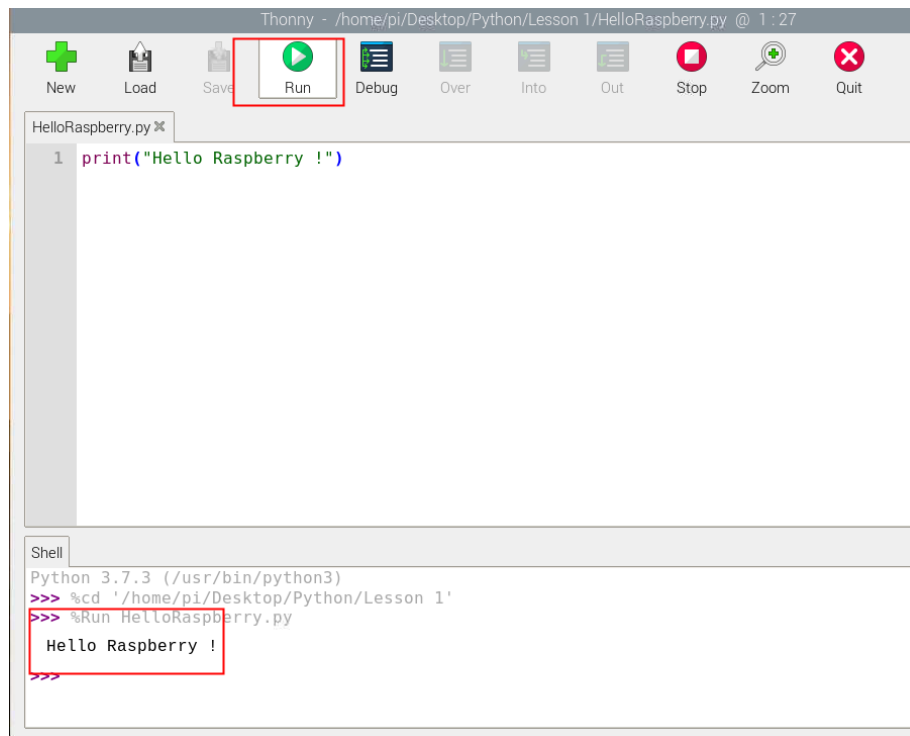
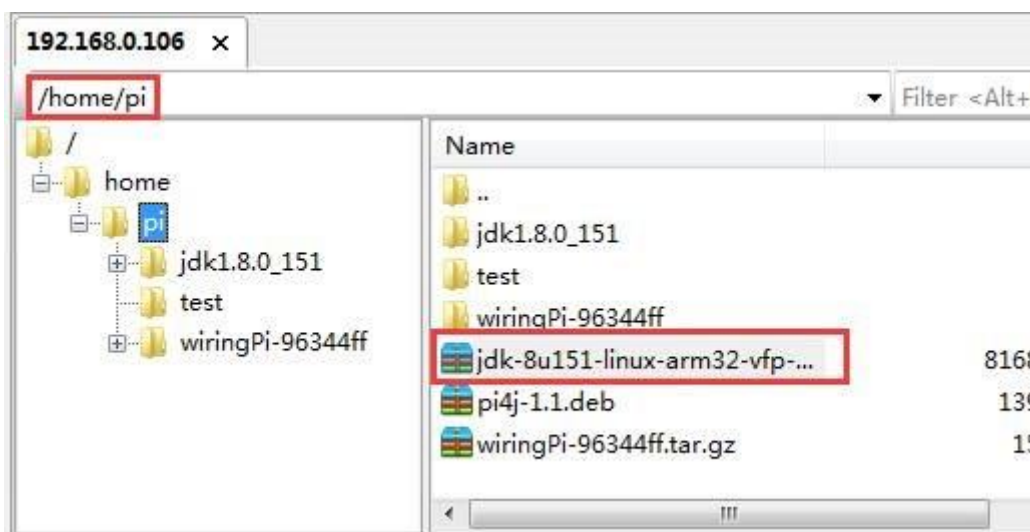


Figure 3.1.10 Run and display the result

10) Java environment construction:

Install JDK



Enter on the command line (decompress the file to the current directory):

tar -zxvf jdk-8u151-linux-arm32-vfp-hflt.tar.gz

Java environment variable configuration

Modify the configuration file:

Command line input: **sudo nano /etc/profile**

Add at the end of the file:

JAVA_HOME=/home/pi/jdk1.8.0_151

CLASSPATH=.:\$JAVA_HOME/jre/lib/rt.jar:\$JAVA_HOME/lib/dt.jar:\$JAVA_HOME/lib/tools.jar\$

PATH=\$JAVA_HOME/bin:\$PATH

export JAVA_HOME CLASSPATH PATH

Keyboard input: Ctrl+O, (Enter, Save) Ctrl+X, (Exit)

Command line input: source /etc/profile

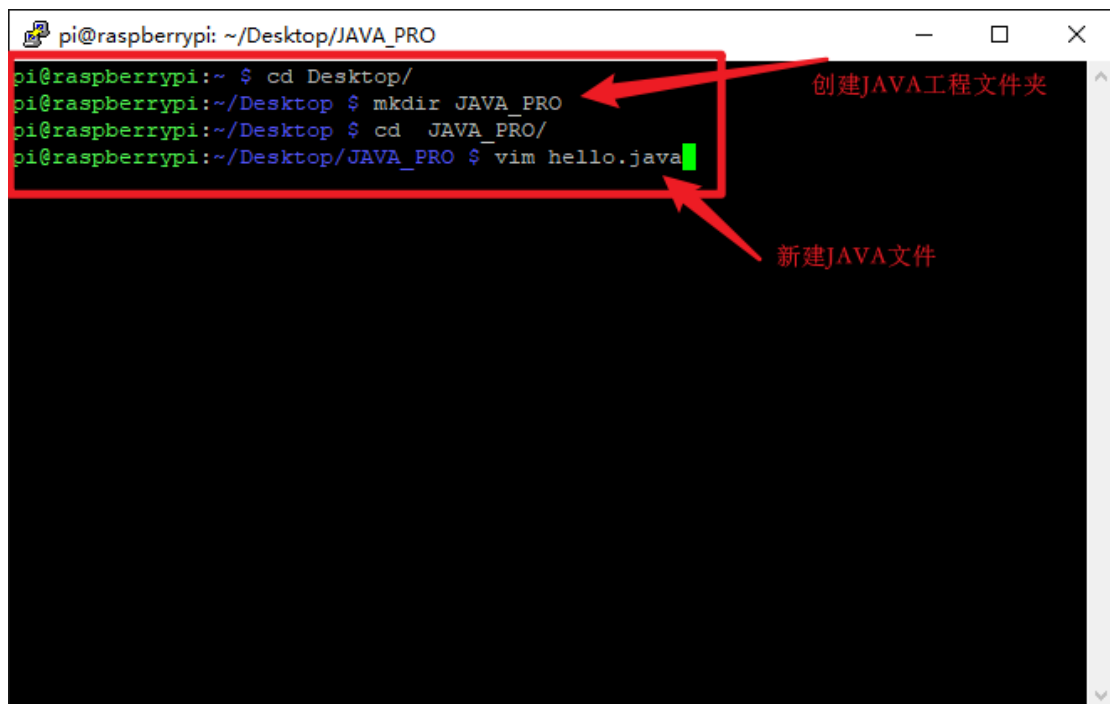
a) Install pi4j

Terminal online installation:

Command: curl -s get.pi4j.com | sudo bash

b) Install wiringPi

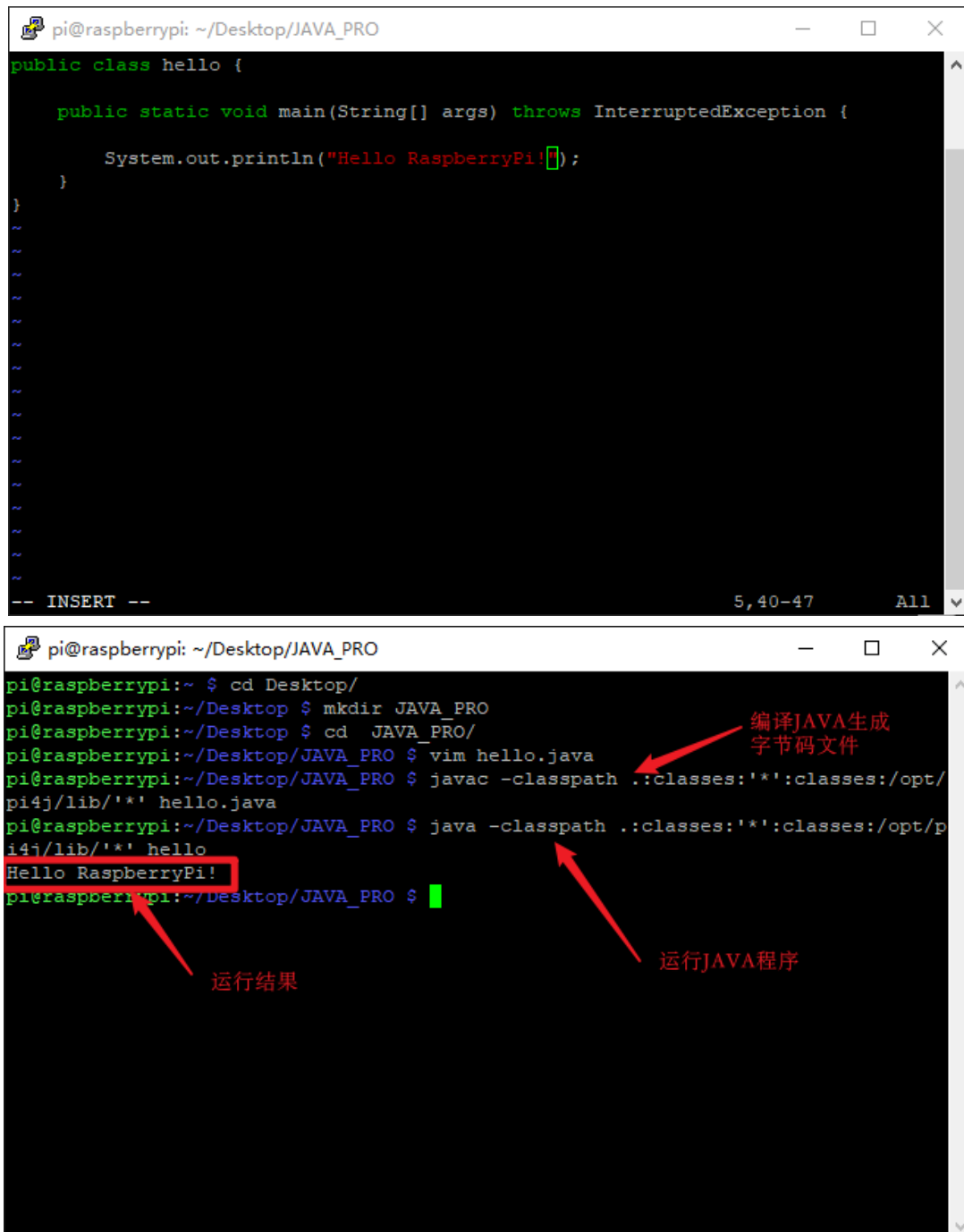
Command: sudo apt-get purge wiringpi



The image shows a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~/Desktop/JAVA_PRO'. The terminal output shows the following commands and their results:

```
pi@raspberrypi:~ $ cd Desktop/  
pi@raspberrypi:~/Desktop $ mkdir JAVA_PRO  
pi@raspberrypi:~/Desktop $ cd JAVA_PRO/  
pi@raspberrypi:~/Desktop/JAVA_PRO $ vim hello.java
```

Two red arrows point from Chinese text labels to the terminal output. The first arrow points to the 'mkdir JAVA_PRO' command, with the label '创建JAVA工程文件夹' (Create JAVA project folder). The second arrow points to the 'vim hello.java' command, with the label '新建JAVA文件' (Create new JAVA file).



The image shows two terminal windows from a Raspberry Pi. The top window displays a Java class named 'hello' with a 'main' method that prints 'Hello RaspberryPi!'. The bottom window shows the execution of this program. It includes commands to create the directory, compile the Java file, and run the program. Red arrows and text labels point to specific parts of the terminal output: '编译JAVA生成字节码文件' (Compile Java to generate byte code file) points to the 'javac' command; '运行JAVA程序' (Run Java program) points to the 'java' command; and '运行结果' (Run result) points to the output 'Hello RaspberryPi!'.

```
pi@raspberrypi: ~/Desktop/JAVA_PRO
public class hello {
    public static void main(String[] args) throws InterruptedException {
        System.out.println("Hello RaspberryPi!");
    }
}
-- INSERT -- 5,40-47 All
```

```
pi@raspberrypi: ~/Desktop/JAVA_PRO
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ mkdir JAVA_PRO
pi@raspberrypi:~/Desktop $ cd JAVA_PRO/
pi@raspberrypi:~/Desktop/JAVA_PRO $ vim hello.java
pi@raspberrypi:~/Desktop/JAVA_PRO $ javac -classpath .:classes:~/.opt/p
pi4j/lib/* hello.java
pi@raspberrypi:~/Desktop/JAVA_PRO $ java -classpath .:classes:~/.opt/p
i4j/lib/* hello
Hello RaspberryPi!
pi@raspberrypi:~/Desktop/JAVA_PRO $
```

编译JAVA生成字节码文件

运行JAVA程序

运行结果

Compile and generate class file command:

```
javac -classpath .:classes:~/.opt/pi4j/lib/* XXX.java
```

Execute file command:

```
java -classpath .:classes:~/.opt/pi4j/lib/* XXX
```

Experimental results:

Use C++ to output Hello RaspberryPi through VNC programming software!

Use Python to output Hello RaspberryPi through Thonny Python IDE programming software!