File reading and writing

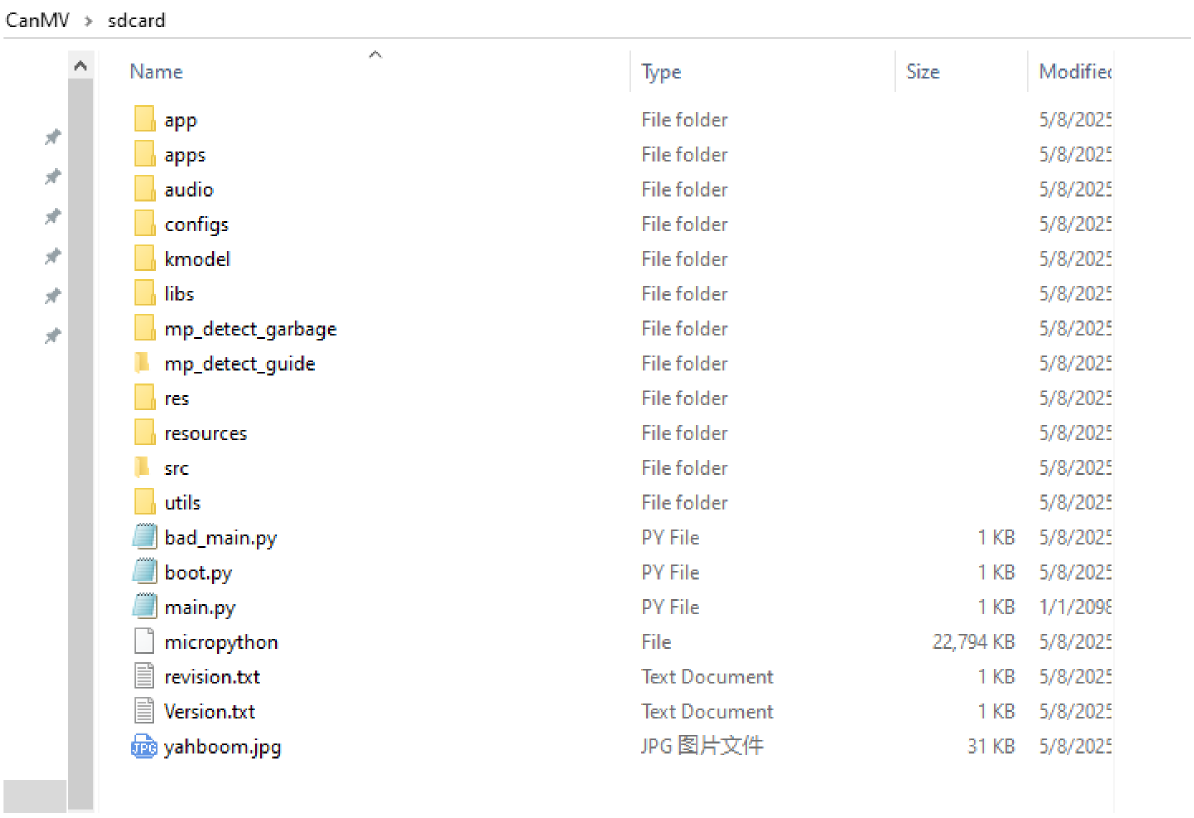
Introduction to the results of routine experiments

> We often encounter situations where we need to save data after power failure, such as storing the data obtained by the sensor in the form of a file.
>
> This facilitates further operations on the data. In fact, whether it is importing a module using import in the code, or taking a photo to save a picture or reading a model file, which we will learn later,
>
> The underlying layer of these behaviors all rely on the read and write operations of the file system. In this tutorial, we will introduce a more commonly used method of file read and write operations.

The example code of this section is located in: [Source code/02.Basic/14.file.py]

We use CanMV IDE to open the sample code and connect K230 to the computer with a USB cable.

In the system's file manager, open the directory corresponding to CanMV. We enter the directory CanMV/sdcard

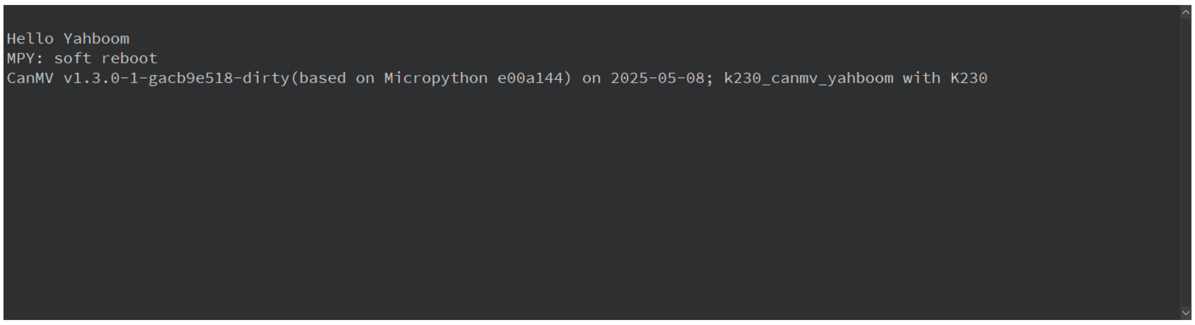


CanMV › sdcard

Name	Type	Size	Modifie
app	File folder		5/8/2025
apps	File folder		5/8/2025
audio	File folder		5/8/2025
configs	File folder		5/8/2025
kmodel	File folder		5/8/2025
libs	File folder		5/8/2025
mp_detect_garbage	File folder		5/8/2025
mp_detect_guide	File folder		5/8/2025
res	File folder		5/8/2025
resources	File folder		5/8/2025
src	File folder		5/8/2025
utils	File folder		5/8/2025
bad_main.py	PY File	1 KB	5/8/2025
boot.py	PY File	1 KB	5/8/2025
main.py	PY File	1 KB	1/1/2098
micropython	File	22,794 KB	5/8/2025
revision.txt	Text Document	1 KB	5/8/2025
Version.txt	Text Document	1 KB	5/8/2025
yahboom.jpg	JPG 图片文件	31 KB	5/8/2025

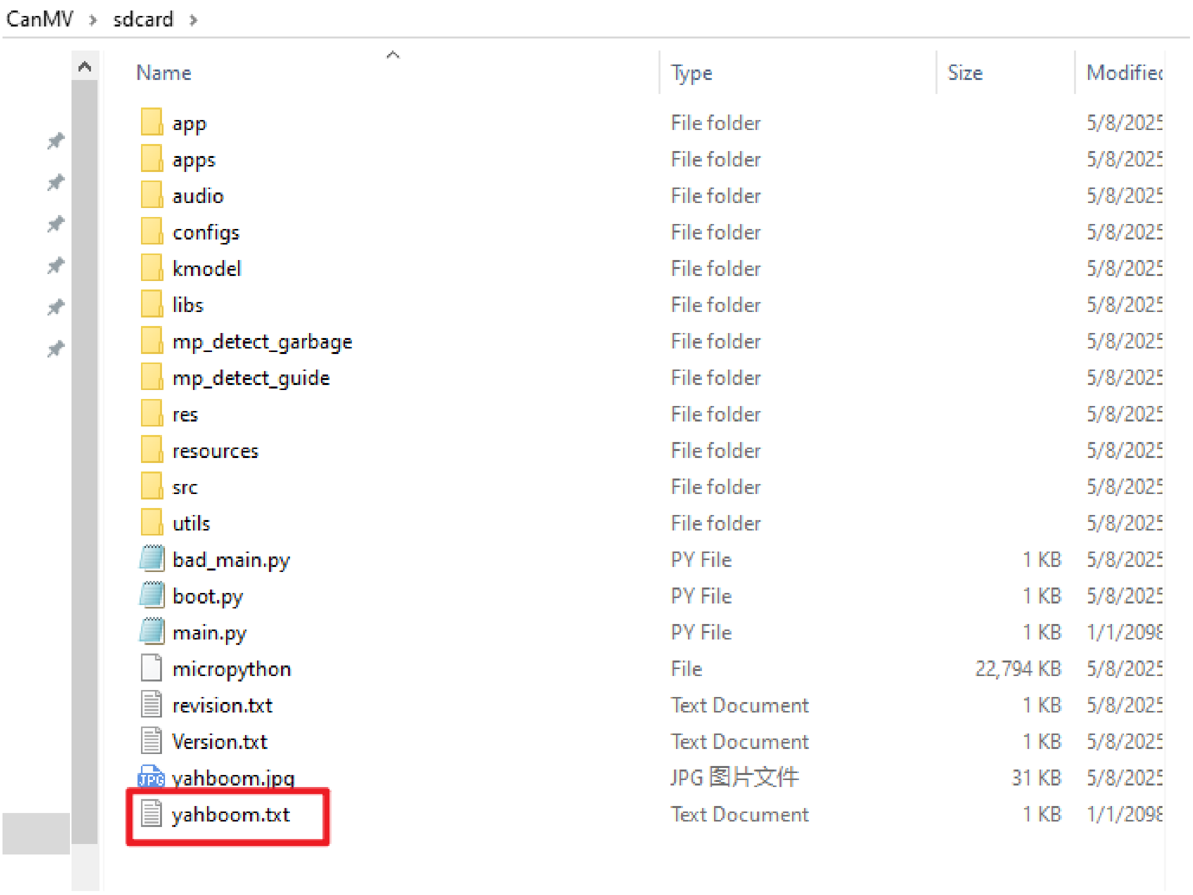
> The CanMV/sdcard/ directory corresponds to /sdcard/ in the code.

Click the Run button in the lower left corner of CanMV IDE and open the serial terminal below. You can see that Hello Yahboom is output here.



```
Hello Yahboom
MPY: soft reboot
CanMV v1.3.0-1-gacb9e518-dirty(based on Micropython e00a144) on 2025-05-08; k230_canmv_yahboom with K230
```

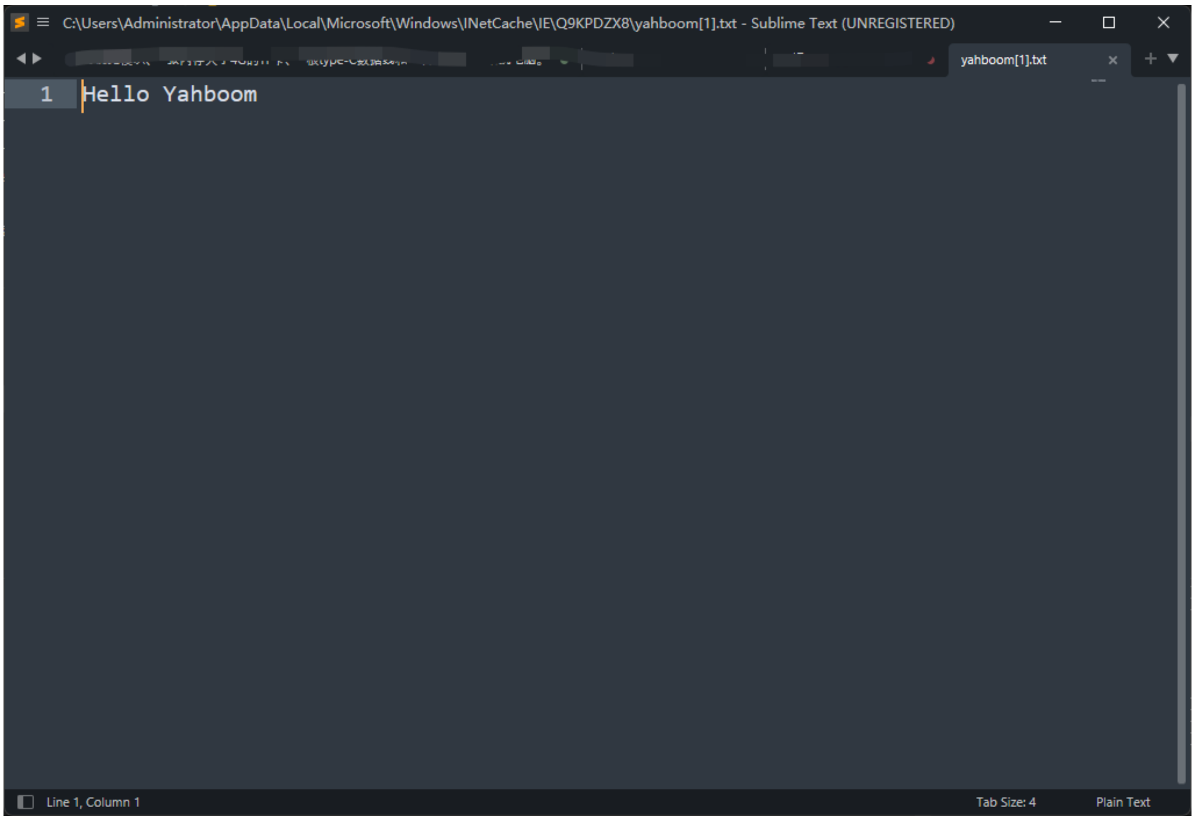
After waiting for the program to finish running, we return to the file manager and can see that there is an additional yahboom.txt file.



CanMV › sdcard ›

Name	Type	Size	Modified
app	File folder		5/8/2025
apps	File folder		5/8/2025
audio	File folder		5/8/2025
configs	File folder		5/8/2025
kmodel	File folder		5/8/2025
libs	File folder		5/8/2025
mp_detect_garbage	File folder		5/8/2025
mp_detect_guide	File folder		5/8/2025
res	File folder		5/8/2025
resources	File folder		5/8/2025
src	File folder		5/8/2025
utils	File folder		5/8/2025
bad_main.py	PY File	1 KB	5/8/2025
boot.py	PY File	1 KB	5/8/2025
main.py	PY File	1 KB	1/1/2098
micropython	File	22,794 KB	5/8/2025
revision.txt	Text Document	1 KB	5/8/2025
Version.txt	Text Document	1 KB	5/8/2025
yahboom.jpg	JPG 图片文件	31 KB	5/8/2025
yahboom.txt	Text Document	1 KB	1/1/2098

We double-click to open this file, and the content inside is "Hello Yahboom"


C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\IE\Q9KPDZX8\yahboom[1].txt - Sublime Text (UNREGISTERED)
yahboom[1].txt
Line 1, Column 1
Tab Size: 4
Plain Text

```
Hello Yahboom
```

Code Explanation

```
# Write to file
with  open ( '/sdcard/yahboom.txt' , 'w' ) as  f :
    f . write ( "Hello Yahboom" )

# File reading
with  open ( '/sdcard/yahboom.txt' , 'r' ) as  f :
    print ( f . read ())
```

We use the with open as syntax to open files in the file system

The open() method has two parameters, the first parameter is the file path, and the second parameter is the opening mode.

Commonly used ones are r (read, read mode) and w (write, write mode)

> with open as .... is a more recommended way to operate on files
>
> 1. `with open as` The file will be closed automatically, regardless of whether an exception occurs in the code. When exiting the with code block, Python will automatically call the file object's `close()` method.
> 2. Direct use `open()` requires manual call `close()` to close the file. If you forget to close it or the code throws an exception, it may cause resource leakage.
>
> Let's take a look at the difference between using and not using

```
# with open as (recommended)
with  open ( 'file.txt' , 'r' ) as  f :
    content = f . read ()
# File automatically closed

# Use open directly
f = open ( 'file.txt' , 'r' )
content = f . read ()
f . close ()   # Need to be closed manually
```

About file reading and writing operations

File operations in MicroPython are similar to those in standard Python, but the functions are relatively simplified.

> Limitations to note:
>
> 1. Memory limit: avoid reading too large files at once
> 2. File system: Some development boards may only support limited file system functions
> 3. Path operation: It is recommended to use a simple relative path

The following are the main file operation methods:

1. Opening a file

```
# Open in read mode
f = open('test.txt', 'r')

# Open in write mode
f = open('test.txt', 'w')

# Open in append mode
f = open('test.txt', 'a')

# Open in binary mode
f = open('test.txt', 'rb')
```

2. Basic read and write operations

```
# Read Operation
f = open ( 'test.txt' , 'r' )
content = f . read ()       # Read all content
line = f . readline ()      # read a line
lines = f . readlines ()    # read all lines into a list
f . close ()

# Write Operation
f = open ( 'test.txt' , 'w' )
f . write ( 'hello\n' )       # Write string
f . writelines ([ 'line1\n' , 'line2\n' ])   # Write multiple lines
f . close ()
```

3. It is recommended to use the with statement

```
with  open ( 'test.txt' , 'r' ) as  f :
    content = f . read ()
```

4. File system operations (supplement, not necessarily fully compatible in Micropython)

```
import  os

# List the files in the current directory
os . listdir ()

# Check if the file exists
os . stat ( 'test.txt' )

# Delete files
os . remove ( 'test.txt' )

# Rename the file
os .rename ( 'old.txt' , ' new.txt' )
```