
InGateway Documentation

Release 0.0.1

zhangning

May 05, 2023

1	InGateway Documentation Site Navigation	1
1.1	Quick Start for MobiusPi Python Development	1
1.2	MobiusPi API Manual	44

InGateway Documentation Site Navigation

The InGateway series of Beijing InHand Networks Technology Co., Ltd. (InHand) consists of InGateway902 and InGateway501.

MobiusPi is a secondary development platform for the InGateway series.

1.1 Quick Start for MobiusPi Python Development

The InGateway series of Beijing InHand Networks Technology Co., Ltd. (InHand) consists of InGateway902 (referred to as **IG902** hereinafter), InGateway502 (referred to as **IG502** hereinafter) and InGateway501 (referred to as **IG501** hereinafter). MobiusPi is a secondary development platform for the InGateway series. This document describes how to perform Python-based secondary development on MobiusPi.

- *1. Build a MobiusPi development environment*
 - *1.1 Prepare the hardware and network environment*
 - * *1.1.1 Connect the power supply and use a network cable to connect the platform to the PC*
 - * *1.1.2 Access the MobiusPi*
 - * *1.1.3 Connect MobiusPi to the Internet*
 - * *1.1.4 Update the software version*
 - * *1.1.5 Enable the debugging mode for MobiusPi*
 - *1.2 Prepare the PC environment*

- * 1.2.1 *Install the Python interpreter*
- * 1.2.2 *Install Visual Studio Code*
- * 1.2.3 *Install OpenSSH*
- 1.3 *Prepare the VS Code development environment*
 - * 1.3.1 *Install the VS Code extension*
 - * 1.3.2 *Configure the Python interpreter version*
 - * 1.3.3 *Configure the project template*
 - 1.3.3.1 *Apply the InHand standard project template*
 - 1.3.3.2 *Custom project template*
- 2. *Compile the MobiusPi App: Hello World*
 - 2.1 *Use a template to create a project*
 - 2.2 *Encoding*
 - 2.3 *Debugging*
 - * 2.3.1 *Create an SFTP connection*
 - * 2.3.2 *Debug the code*
 - 2.4 *Build an App release package*
 - 2.5 *Deploy the App on the MobiusPi web page*
 - 2.6 *View the App running status*
 - 2.7 *Update the App configuration file*
 - 2.8 *Annex*
 - * 2.8.1 *Install a third-party dependency library for the specified App*
 - * 2.8.2 *Install the third-party dependency library to SDK*
 - * 2.8.3 *Enable automatic code completion*
- *FAQ*
 - *During SFTP connection building, it prompts “REMOTE HOST IDENTIFICATION HAS CHANGED and Host key verification failed”*
 - *When synchronizing the code to the remote server, it prompts “All configured authentication methods failed”*
 - *How to call the serial port and network port of MobiusPi*
 - *When creating an SFTP connection with MobiusPi, it prompts “SSH error”*

1.1.1 1. Build a MobiusPi development environment

Before starting development, ensure that you get the following items ready:

- InGateway
 - InGateway501
 - * Firmware version: V2.0.0.r12351 or later
 - * SDK version: py2sdk-V1.3.4 or later
 - * The network is connected and the debugging mode is enabled.
 - InGateway502
 - * Firmware version: V2.0.0.r13771 or later
 - * SDK version: py3sdk-V1.3.5 or later
 - * The network is connected and the debugging mode is enabled.
 - InGateway902
 - * Firmware version: V2.0.0.r12537 or later
 - * SDK version: py3sdk-V1.3.5 or later
 - * The network is connected and the debugging mode is enabled.
- PC
 - Python 2.7.X/3.7.X interpreter
 - Visual Studio Code software
 - * Python plug-in
 - * Project Templates plug-in
 - * SFTP plug-in
 - OpenSSH

If your MobiusPi and PC have met all the above items, skip this section. Otherwise, prepare a development environment as follows.

- *1.1 Prepare the hardware and network environment*
- *1.2 Prepare the PC environment*
- *1.3 Prepare the VS Code development environment*

1.1 Prepare the hardware and network environment

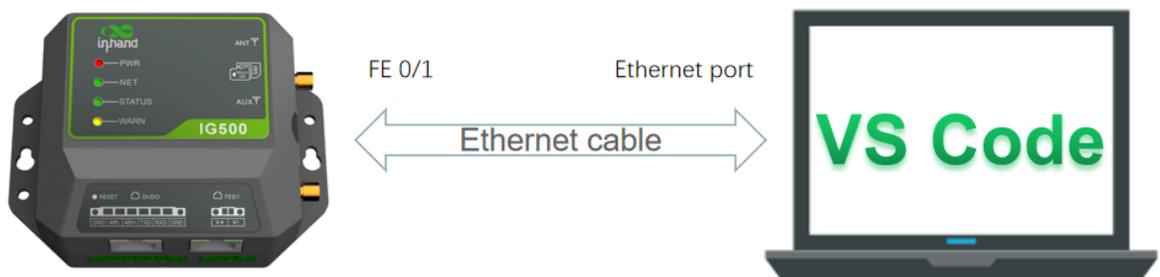
- *1.1.1 Connect the power supply and use a network cable to connect the platform to the PC*

- *1.1.2 Access the MobiusPi*
- *1.1.3 Connect MobiusPi to the Internet*
- *1.1.4 Update the software version*
- *1.1.5 Enable the debugging mode for MobiusPi*

1.1.1 Connect the power supply and use a network cable to connect the platform to the PC

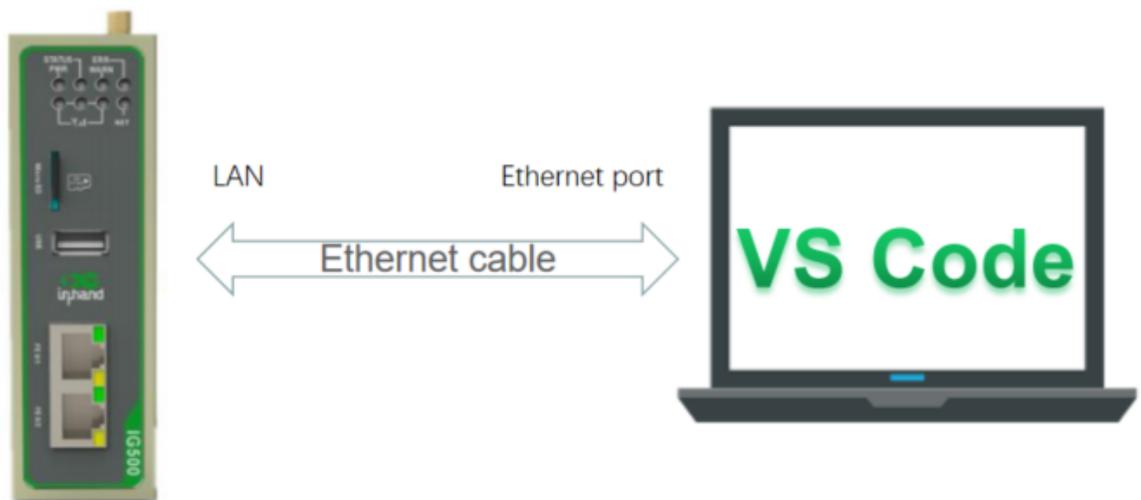
- Prepare the IG501 hardware

Power on IG501 and connect the PC and IG501 through an Ethernet cable according to the topology.



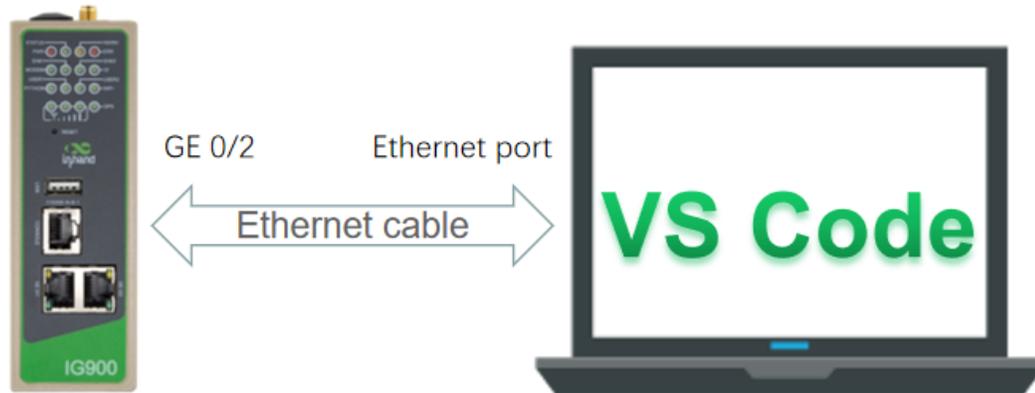
- Prepare the IG502 hardware

Power on IG502 and connect the PC and IG502 through an Ethernet cable according to the topology.



- Prepare the IG902 hardware

Power on IG902 and connect the PC and IG902 through an Ethernet cable according to the topology.



1.1.2 Access the MobiusPi

- Access the IG501 by referring to [Access the IG501](#).
- Access the IG902 by referring to [Access the IG902](#).

1.1.3 Connect MobiusPi to the Internet

- Connect IG501 to the Internet by referring to [Connect IG501 to the Internet](#).
- Connect IG502 to the Internet by referring to [Connect IG502 to the Internet](#).
- Connect IG902 to the Internet by referring to [Connect IG902 to the Internet](#).

1.1.4 Update the software version

If you want to get the latest MobiusPi and its functional characteristics, please visit the [Resource](#). To update the software version, see the following links:

- [Update the IG501 software version](#)
- [Update the IG502 software version](#)
- [Update the IG902 software version](#)

1.1.5 Enable the debugging mode for MobiusPi

To run and debug Python code on MobiusPi during development, you need to enable the debugging mode for MobiusPi.

- [Enable the IG501 debugging mode](#)
- [Enable the IG502 debugging mode](#)

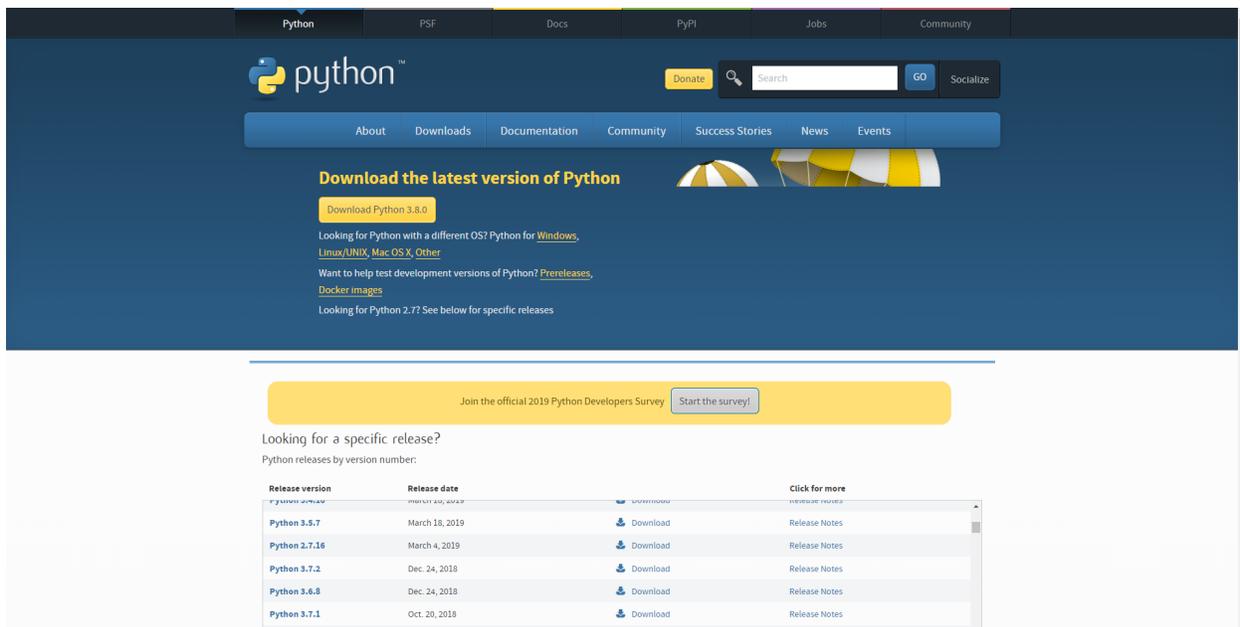
- Enable the IG902 debugging mode

1.2 Prepare the PC environment

- *1.2.1 Install the Python interpreter*
- *1.2.2 Install Visual Studio Code*
- *1.2.3 Install OpenSSH*

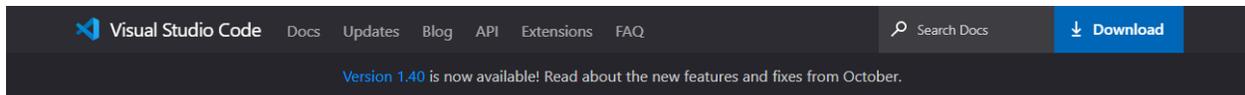
1.2.1 Install the Python interpreter

The PC shall be installed with the Python2.7.X or 3.7.X interpreter (the 3.7.X interpreter is recommended). You can download the install package from <https://www.python.org/downloads/> and install it to the PC.



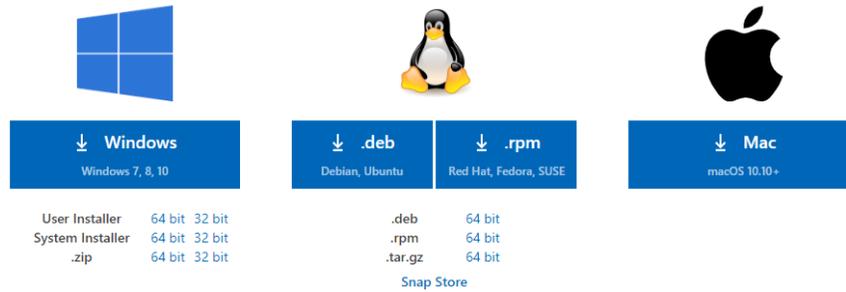
1.2.2 Install Visual Studio Code

You can download Visual Studio Code (VS Code) from <https://code.visualstudio.com/Download>.



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

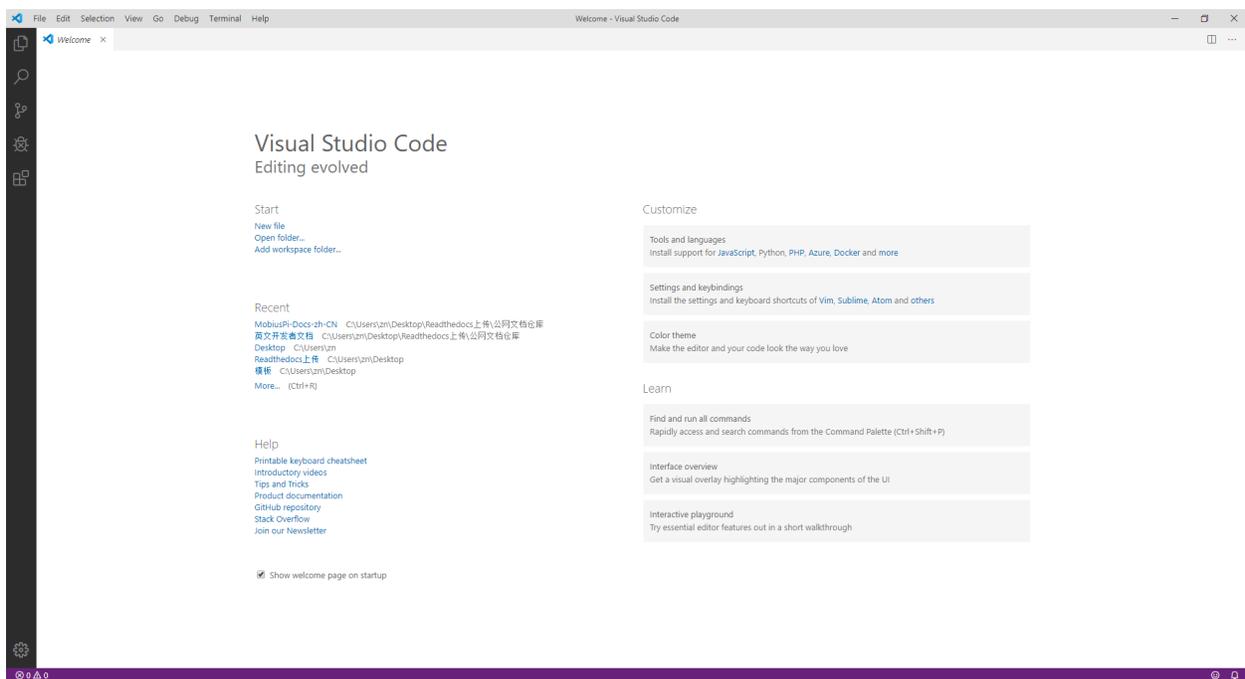


By downloading and using Visual Studio Code, you agree to the [license terms](#) and [privacy statement](#).

Want new features sooner?

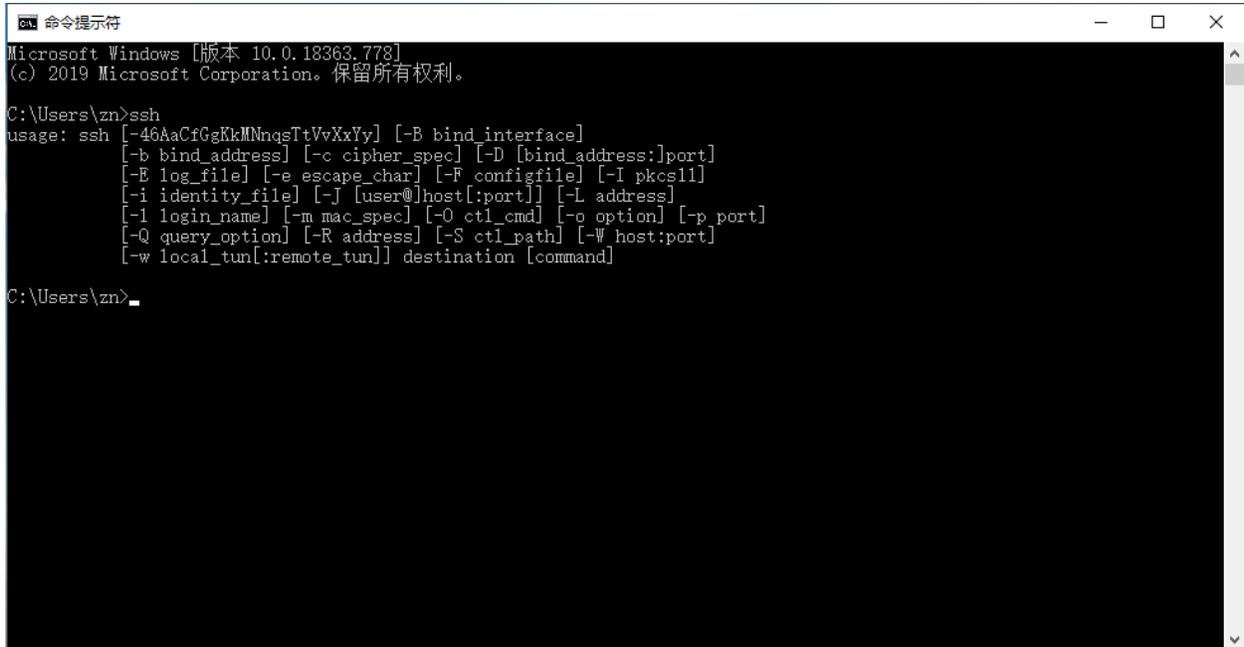
Get the [Insiders build](#) instead.

After VS Code is downloaded and installed, run the software, and its page is as follows.



1.2.3 Install OpenSSH

You can enter the `ssh` command in Command Prompt to check whether the PC supports the SSH protocol. If the PC supports the SSH protocol, the following figure is displayed:

A screenshot of a Windows Command Prompt window titled "命令提示符". The window shows the output of the 'ssh' command, which displays the usage and options for the ssh command. The text in the window is as follows:

```
Microsoft Windows [版本 10.0.18363.778]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\zn>ssh
usage: ssh [-46AaCfGgKkMnqsTtVvXxYy] [-B bind_interface]
          [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
          [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
          [-i identity_file] [-J [user@]host[:port]] [-L address]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
          [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
          [-w local_tun[:remote_tun]] destination [command]

C:\Users\zn>_
```

If the PC does not support the SSH protocol, download OpenSSH from <https://www.openssh.com> and install it to the PC.

1.3 Prepare the VS Code development environment

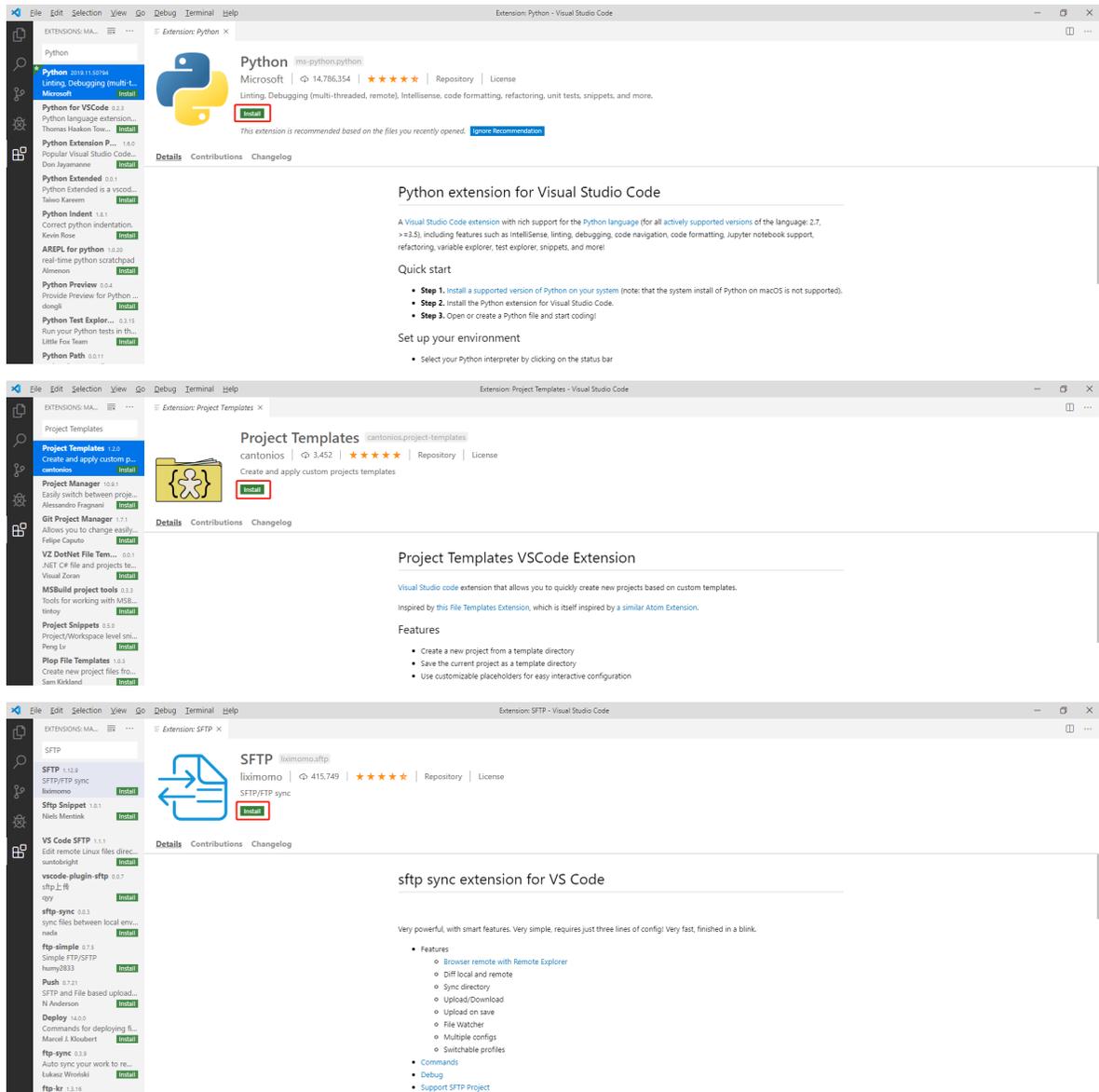
- *1.3.1 Install the VS Code extension*
- *1.3.2 Configure the Python interpreter version*
- *1.3.3 Configure the project template*
 - *1.3.3.1 Apply the InHand standard project template*
 - *1.3.3.2 Custom project template*

1.3.1 Install the VS Code extension

To develop and debug the Python code on MobiusPi, you must install the following extensions in Extensions of VS Code IDE:

- Python is a VS Code Python extension. It provides rich features, including IntelliSense, Linting, debugging, code navigation and formatting, support to Jupyter notebook, restructuring, variable resource manager, test resource manager, and code snippets. For more information, see its [official website](#).

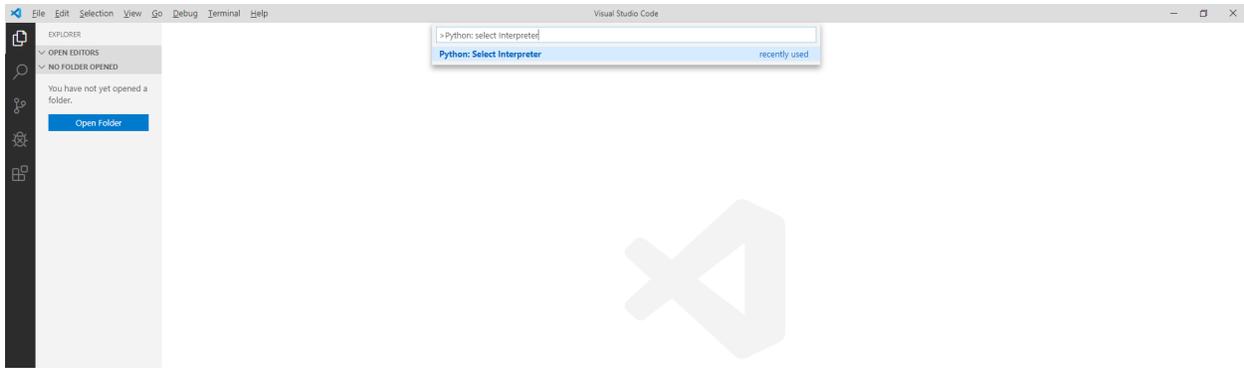
- **Project Templates** is a VS Code extension that allows you to quickly create new projects based on a custom template. InHand will release multiple Python App templates. You can use Project Templates to import a template and initialize your project quickly.
- **SFTP** allows you to use the SFTP Sync plug-in to upload the code to MobiusPi.



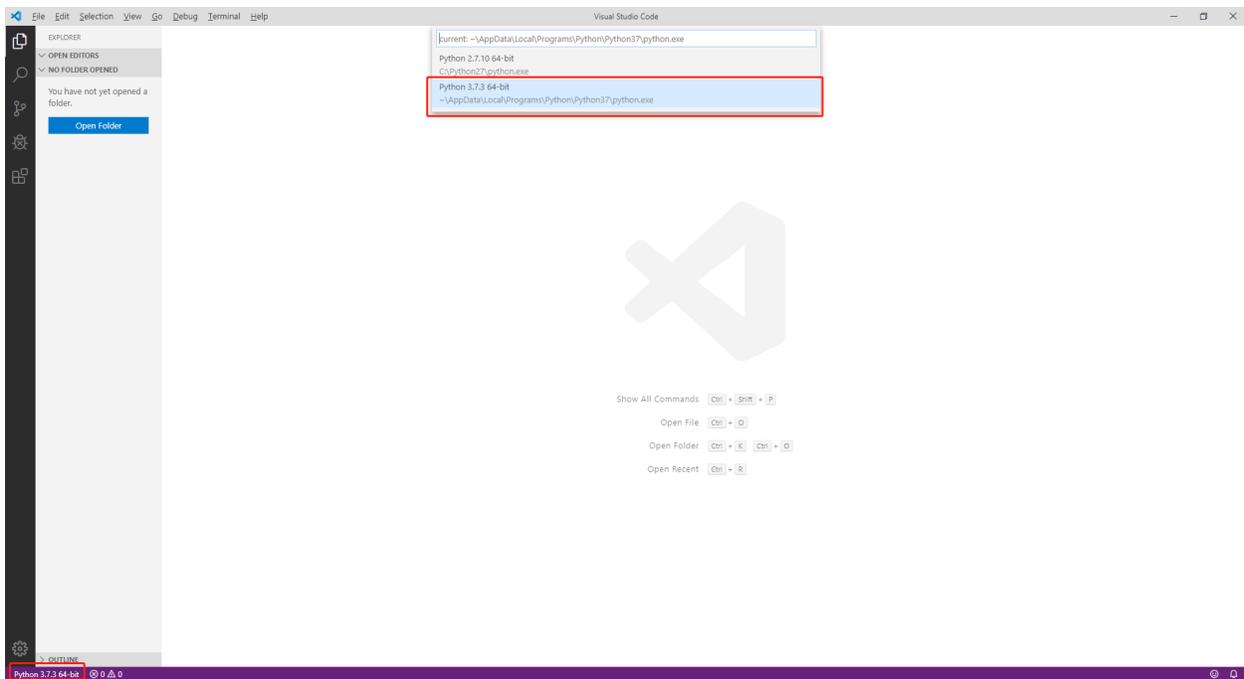
Now, all plug-ins required by MobiusPi have been installed. For more information about VS Code plug-ins, see the [Visual Studio Code official website](#).

1.3.2 Configure the Python interpreter version

Press **Ctrl+Shift+P**. On the displayed Command Palette, enter **>Python: select Interpreter**.



Select a required Python interpreter. In this document, the Python 3.7.X interpreter is used. The selected 3.7.X version shall be consistent with that on the **Edge Computing > Python Edge Computing** page. Then, the selected Python interpreter version is displayed in the lower-left corner of VS Code.

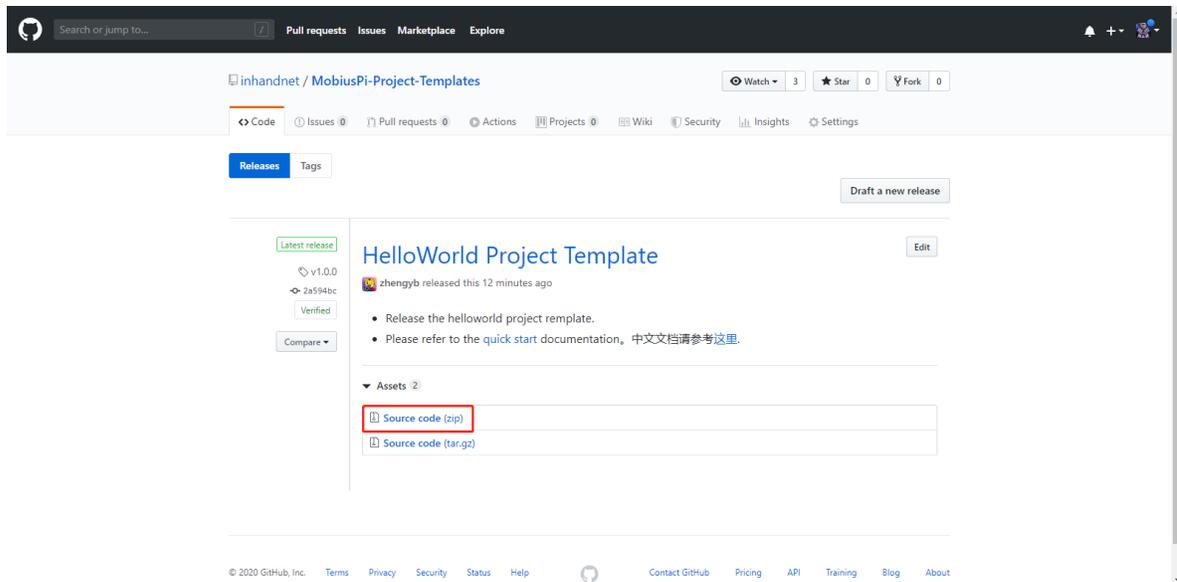


1.3.3 Configure the project template

1.3.3.1 Apply the InHand standard project template

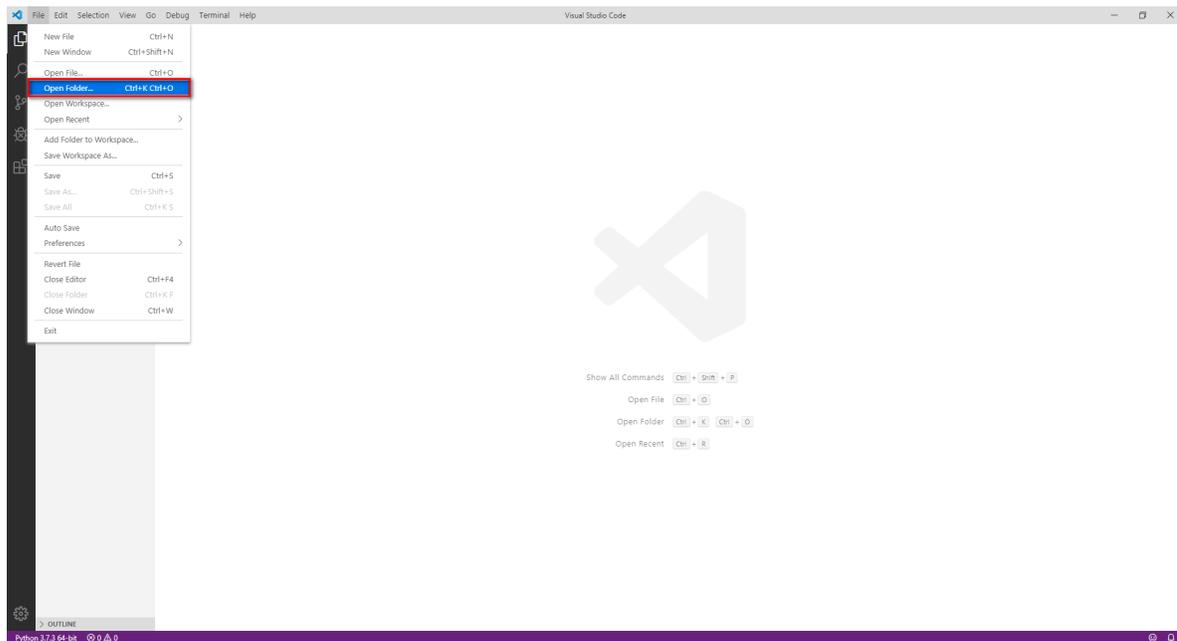
- Step 1: Click [Here](#) to download a MobiusPi project template.

MobiusPi provides multiple project templates for you to quickly initialize your project directory. For more information about each project template, see [README.md](#). In this document, the standard project template **helloworld-template** is used as an example.



- Step 2: Open the project template.

Decompress the downloaded project template package, use VS Code to open the helloworld-template folder in the decompressed package, choose **Files > Open Folder**, and select the decompressed **helloworld-template** folder.



The following figure shows the opened project template folder **helloworld-template**. The project template contains the following information:

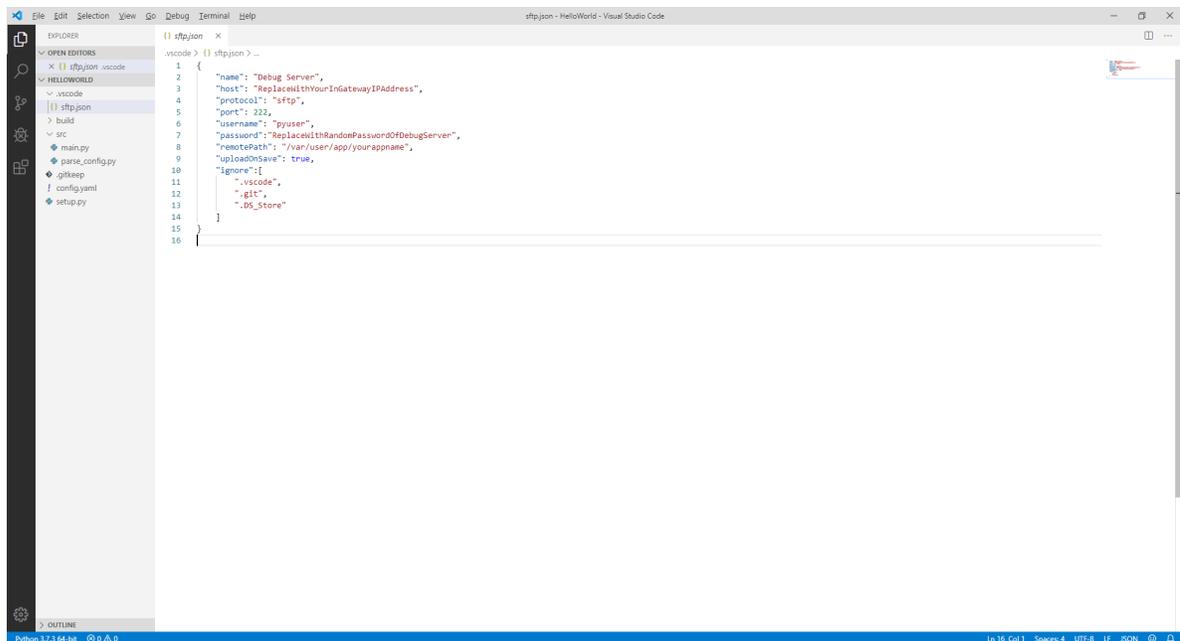
```
.vscode
  sftp.json
build
```

(continues on next page)

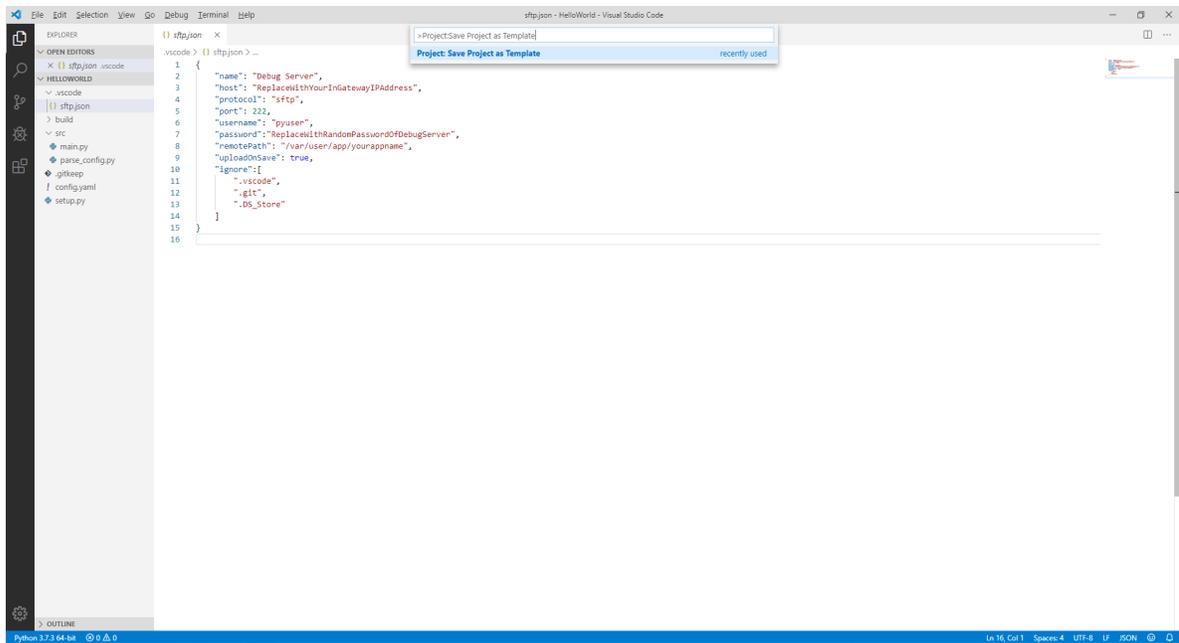
(continued from previous page)

```
lib
src
  main.py
  parse_config.py
config.yaml
setup.py
```

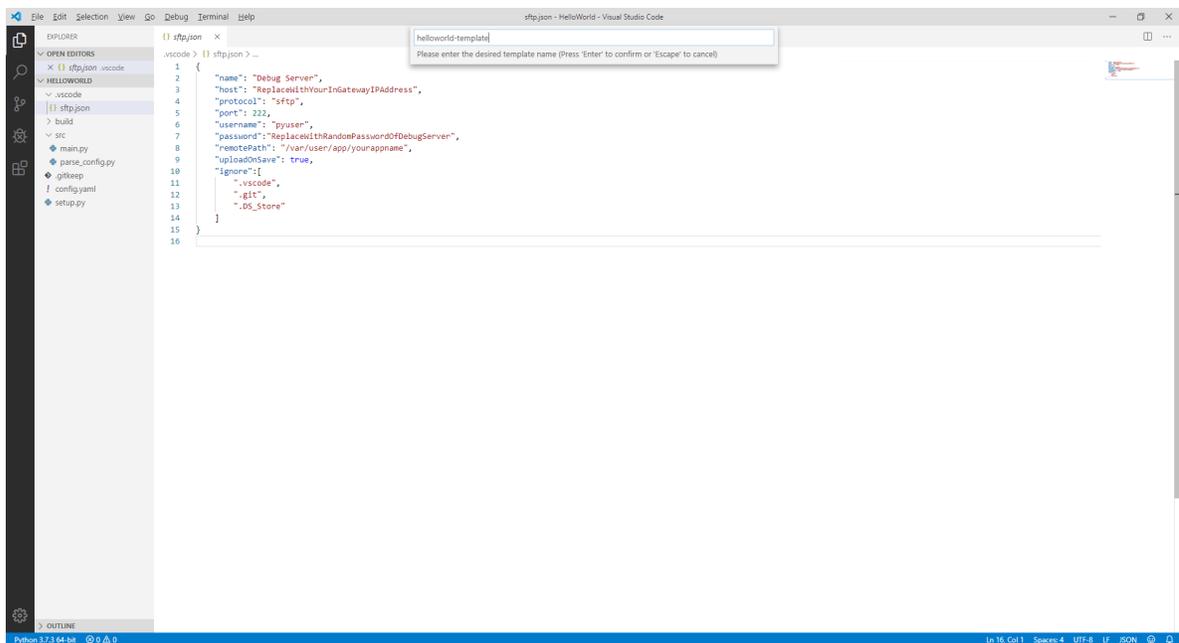
- `.vscode`: The VS Code configuration folder.
 - * `sftp.json`: The configuration file of the SFTP plug-in, which is used to connect to MobiusPi over SFTP.
- `build`: The App release package folder.
- `lib`: The folder of the App third-party dependency library.
- `src`: The App source code folder.
 - * `main.py`: The App entry.
 - * `parse_config.py`: The App parsing configuration file.
- `config.yaml`: The App configuration file.
- `setup.py`: The App version, SDK version, and other information.



- Step 3: In the Command Palette, enter the `>Project:Save Project as Template` command to save the current project file as a template.



Enter a name for your template, such as helloworld-template.



1.3.3.2 Custom project template

- Step 1: Create a project template folder which must contain the following information. You can add other files as needed.

```
.vscode
  sftp.json
```

(continues on next page)

(continued from previous page)

```
build
src
  main.py
  setup.py
```

- `.vscode`: The VS Code configuration file. You can enter the `>SFTP:Config` command in the VS Code Command Palette to quickly create the `.vscode` folder and the `sftp.json` file.
 - * `sftp.json`: The configuration file of the SFTP plug-in, which is used to connect to MobiusPi over SFTP.
 - `build`: The App release package folder.
 - `src`: The App source code folder.
 - * `main.py`: The App entry.
 - `setup.py`: The App version, SDK version, and other information. We recommend that you define the information according to the standard template.
- Step 2: Use VS Code to open the custom project template folder, choose **Files > Open Folder**, and select the custom project template folder.
 - Step 3: In the Command Palette, enter the `>Project:Save Project as Template` command to save the current project file as a template.

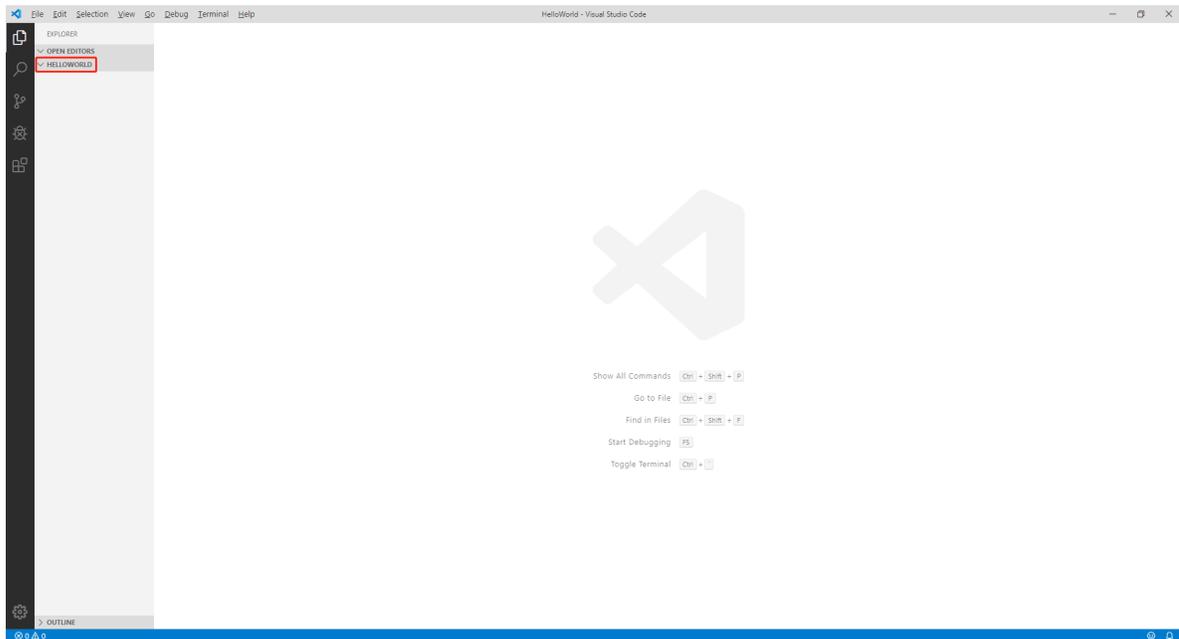
1.1.2 2. Compile the MobiusPi App: Hello World

This document takes the **HelloWorld** App as an example to describe how to develop MobiusPi Python Apps in VS Code. This App can print a “hello world!” log every 10s in MobiusPi and allows you to import the configuration file and modify the log.

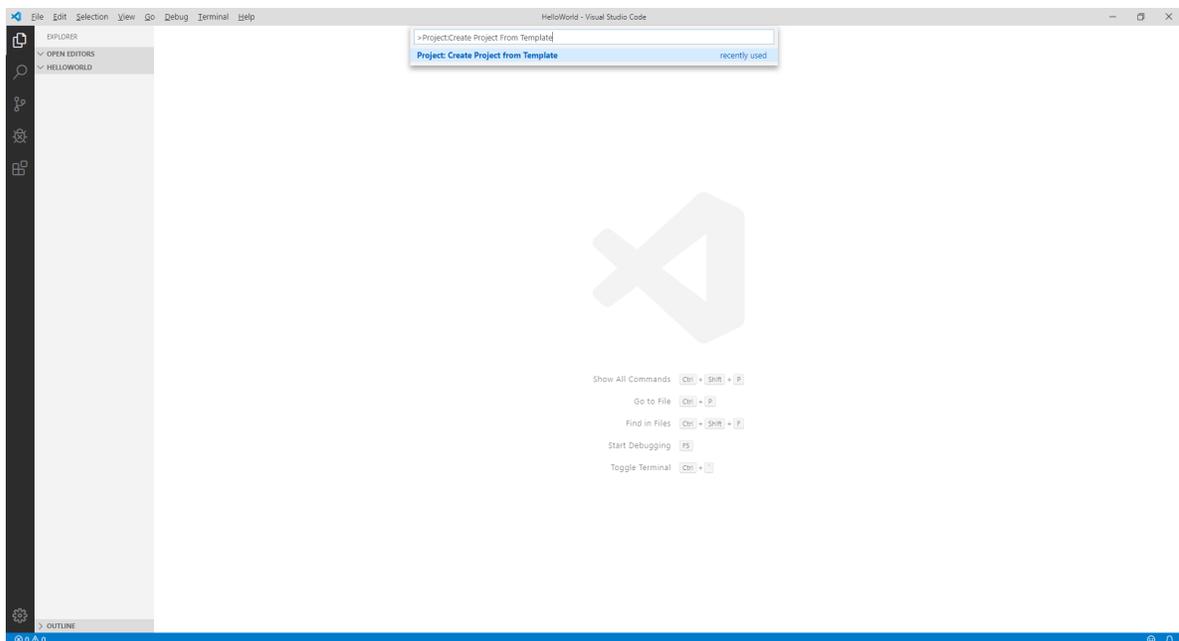
- *2.1 Use a template to create a project*
- *2.2 Encoding*
- *2.3 Debugging*
- *2.4 Build an App release package*
- *2.5 Deploy the App on the MobiusPi web page*
- *2.6 View the App running status*
- *2.7 Update the App configuration file*
- *2.8 Annex*

2.1 Use a template to create a project

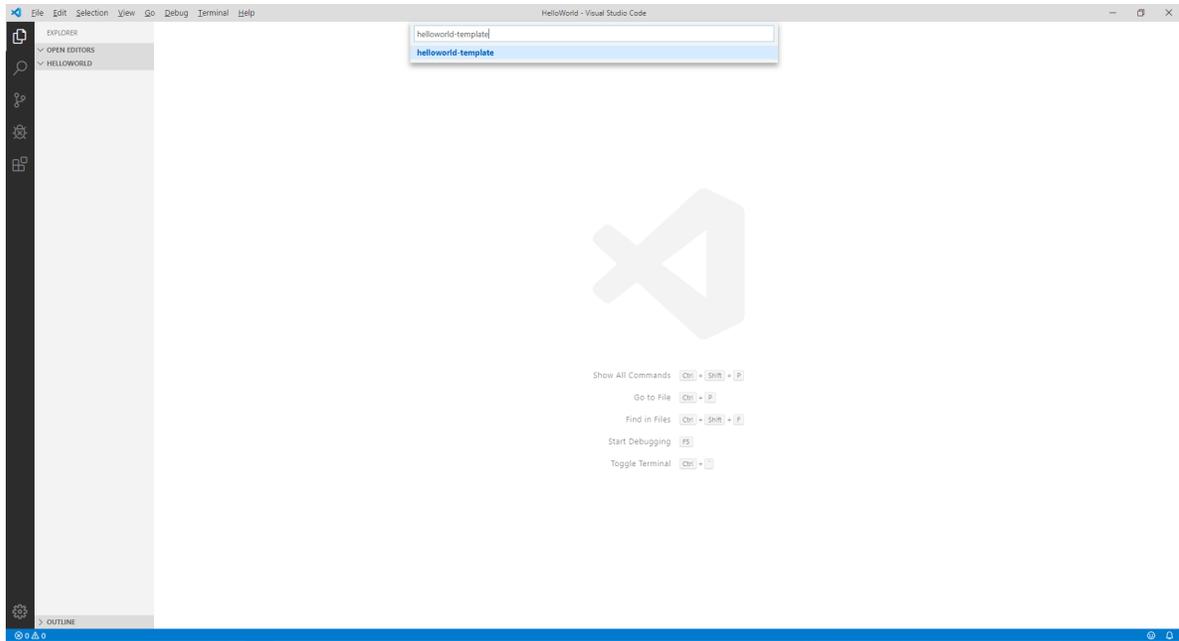
- Step 1: Use VS Code to open the project folder of the Python App. The following figure is displayed:



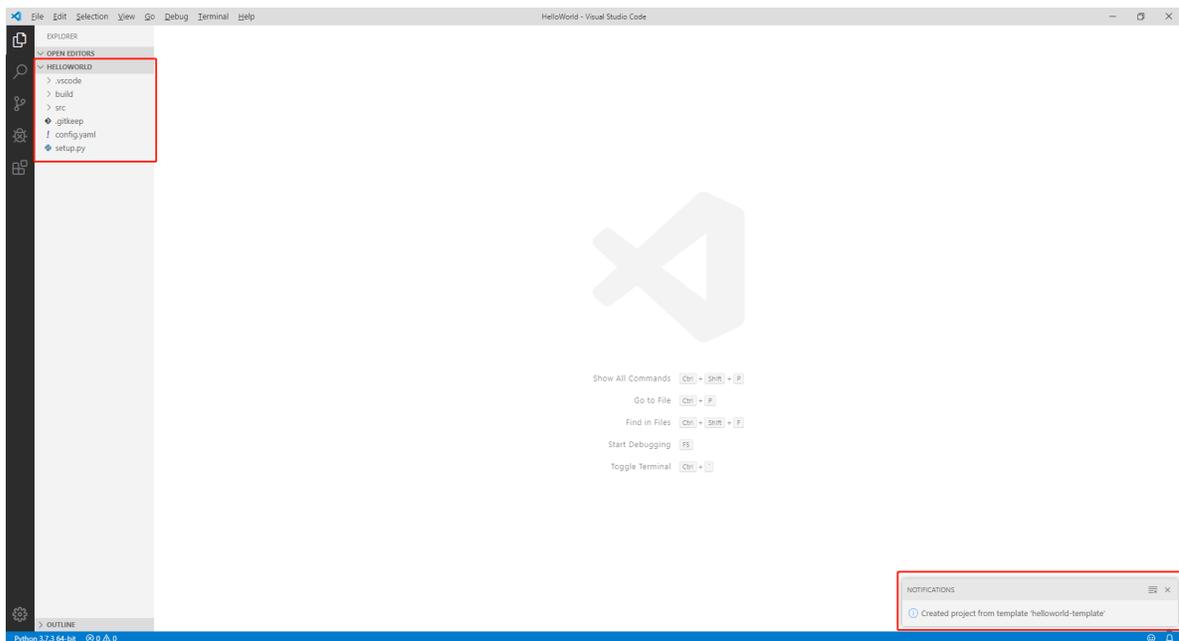
- Step 2: In the Command Palette, enter the `>Project:Create Project From Template` command to quickly create a project directory based on the existing template.



- Step 3: Enter the name of the helloworld-template, and press Enter.



After the template is selected, VS Code automatically adds the file that you add in the template to the folder of the current project.



2.2 Encoding

The standard project template **helloworld-template** can print a “hello world!” log every 10s in MobiusPi and allows you to import the configuration file and modify the log. To modify the App name, modify the code in `main.py` and `setup.py` by referring to the following figure. Note: The Python App name cannot contain any space.

The first screenshot shows the `main.py` file in Visual Studio Code. The code defines a `main` function that logs a message and then reads a configuration file. A red box highlights the string `"HelloWorld"` in the `APPConfig` constructor, and a red arrow points to it with the text: "Replace appname with the actual App name. HelloWorld is used in this document."

```

1 import os
2 import sys
3 import logging
4 import time
5 from parse_config import YamlConfig
6 from mobiuspi_lib.config import Config as APPConfig
7
8 debug_format = '[%(asctime)s] [%(levelname)s] [%(filename)s %(lineno)d]: %(message)s'
9 logging.basicConfig(format=debug_format, level=logging.INFO)
10
11 def main(argv=sys.argv):
12     logging.info("Hello, world! Welcome to the Inhand!")
13     print("Hello, world! Welcome to the Inhand!")
14
15     """get config file of user app
16     APPConfig(app_name="appname")
17     'appname' is the name of user app
18     it should the same as the name in setup.py
19     app_config.get_app_cfg_file()
20     get user app configuration file
21     """
22     app = APPConfig(app_name="HelloWorld")
23     app_config_file = app.get_app_cfg_file()
24     if not app_config_file:
25         logging.warn("Do not find config file, please import first!")
26         return
27
28     #print("app config file:%s" % app_config_file)
29     config = YamlConfig(app_config_file)
30     config_others = config.get_option_config('config', 'others')
31     while True:
32         logging.info("decription:%s" % (config.get_option_config('config', 'description')))
33         print("decription:%s" % (config.get_option_config('config', 'description')))
34
35         logging.info("debug:%s" % (config_others['LOG']['debug']))
36         print("debug:%s" % (config_others['LOG']['debug']))
37

```

The second screenshot shows the `setup.py` file in Visual Studio Code. The code uses `setuptools` to define the application's metadata. Three red boxes highlight the values `'HelloWorld'`, `'0.0.1'`, and `'1.3.5'`, with red arrows pointing to them and labels: "App name", "App version number", and "App SDK version number" respectively.

```

9     """
10     from setuptools import setup, find_packages
11     setup(
12         name='HelloWorld',
13         version='0.0.1',
14         sdk_version='1.3.5',
15         author='Inhand',
16         author_email='',
17         description='',
18         license='PRIVATE',
19         packages=find_packages('src'),
20         package_dir={'': 'src'},
21         zip_safe=False,
22         install_requires=[],
23         entry_points="""
24         [console_scripts]
25         linkedge = Application:main
26         """)

```

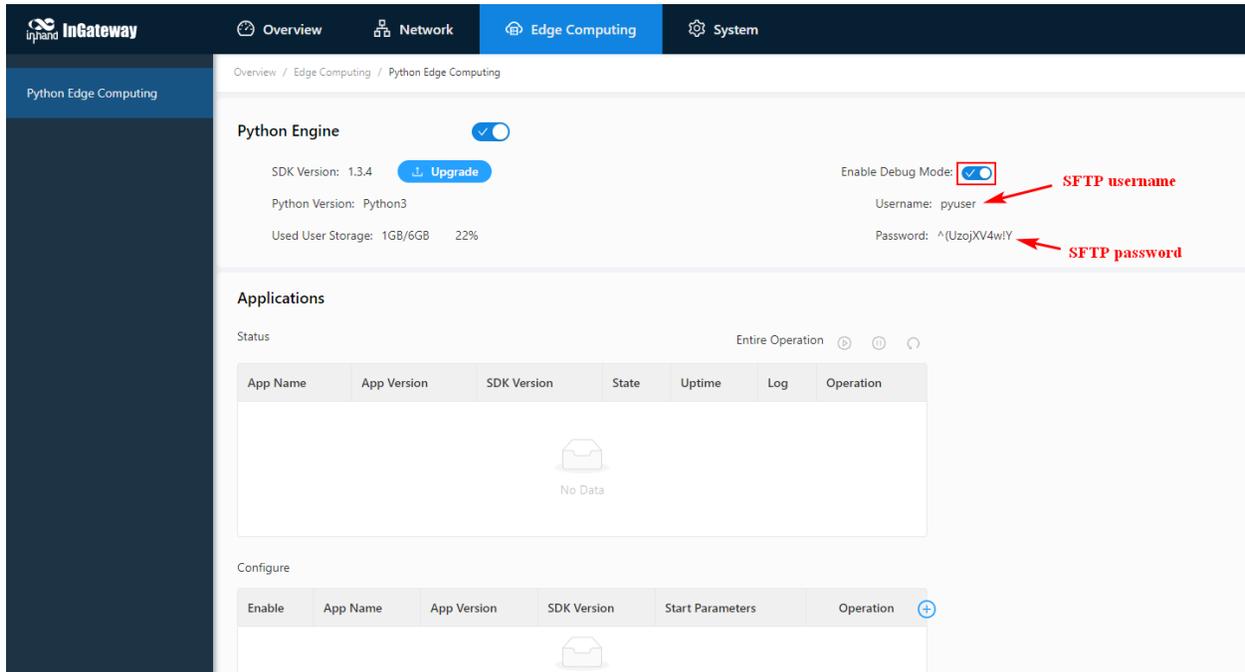
2.3 Debugging

- *2.3.1 Create an SFTP connection*

- 2.3.2 Debug the code

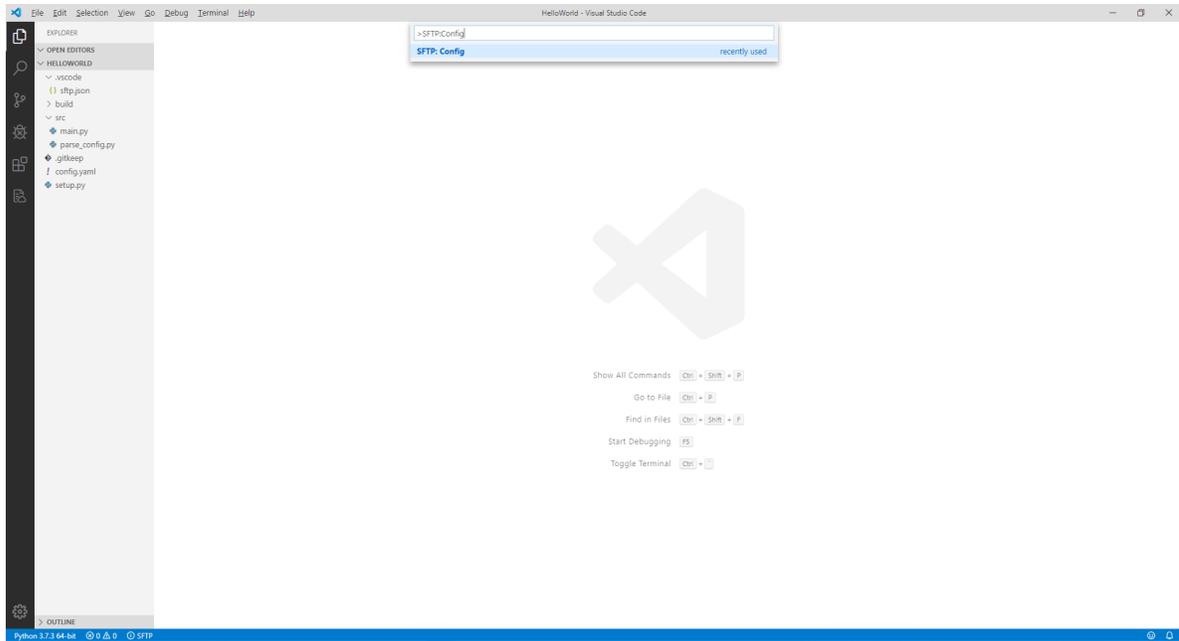
2.3.1 Create an SFTP connection

To debug the code remotely, upload the local code to the remote server (MobiusPi). Before uploading the local code, ensure that the debugging mode has been enabled for MobiusPi. After the debugging mode is enabled, the following figure is displayed:



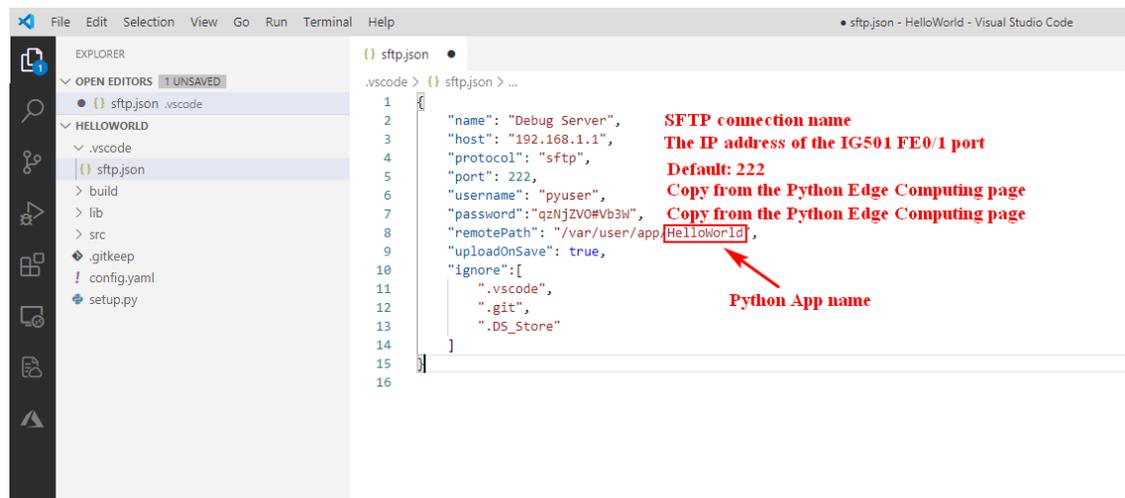
- Step 1: Open the `sftp.json` file.

In the Command Palette, enter the `>SFTP:Config` command and then open the `sftp.json` file.



- Step 2: Configure the SFTP connection.
 - Configure the SFTP connection for IG501

In the `sftp.json` file, configure the SFTP connection according to the connection parameters on the **Edge Computing > Python Edge Computing** page. Note: The Python App name must be same to that in `mian.py`.



- Configure the SFTP connection for IG502

In the `sftp.json` file, configure the SFTP connection according to the connection parameters on the **Edge Computing > Python Edge Computing** page. Note: The Python App name must be same to that in `mian.py`.

```

1 {
2   "name": "Hello World Debug Server",
3   "host": "192.168.2.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "$maVz0nfJEG&",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

SFTP connection name
The IP address of the IG502 LAN port
Default: 222
Copy from the Python Edge Computing page
Copy from the Python Edge Computing page
Python App name

- Configure the SFTP connection for IG902

In the `sftp.json` file, configure the SFTP connection according to the connection parameters on the **Edge Computing > Python Edge Computing** page. Note: The Python App name must be same to that in `mian.py`.

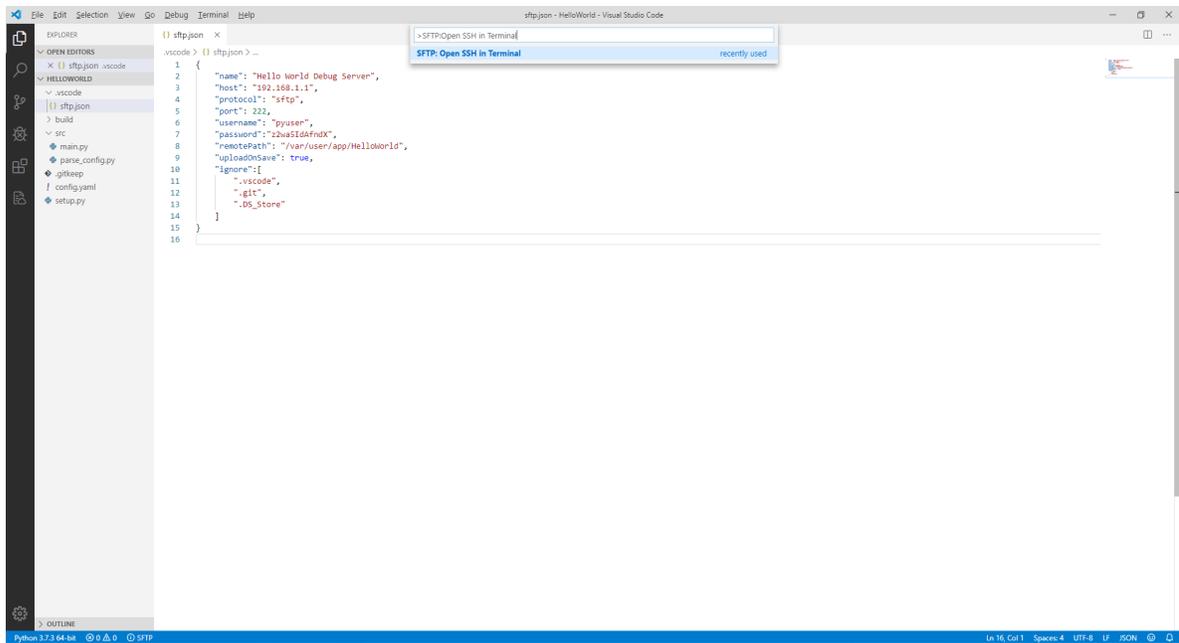
```

1 {
2   "name": "Debug Server",
3   "host": "192.168.2.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "qzNjZVO#Vb3W",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

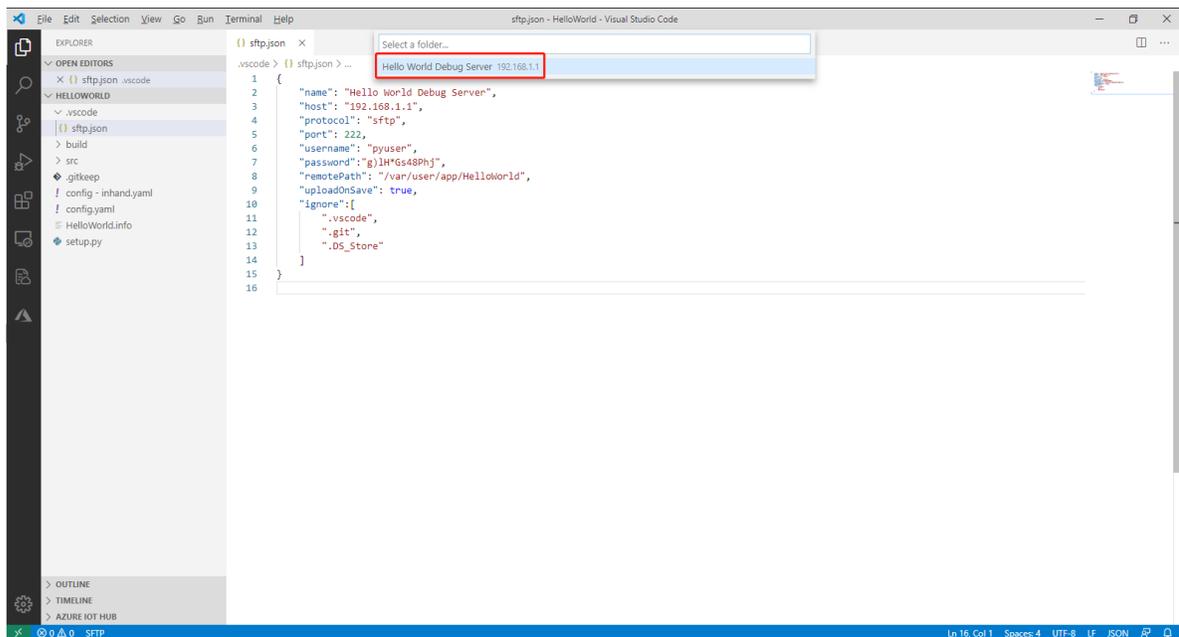
```

SFTP connection name
The IP address of the IG902 GE0/2 port
Default: 222
Copy from the Python Edge Computing page
Copy from the Python Edge Computing page
Python App name

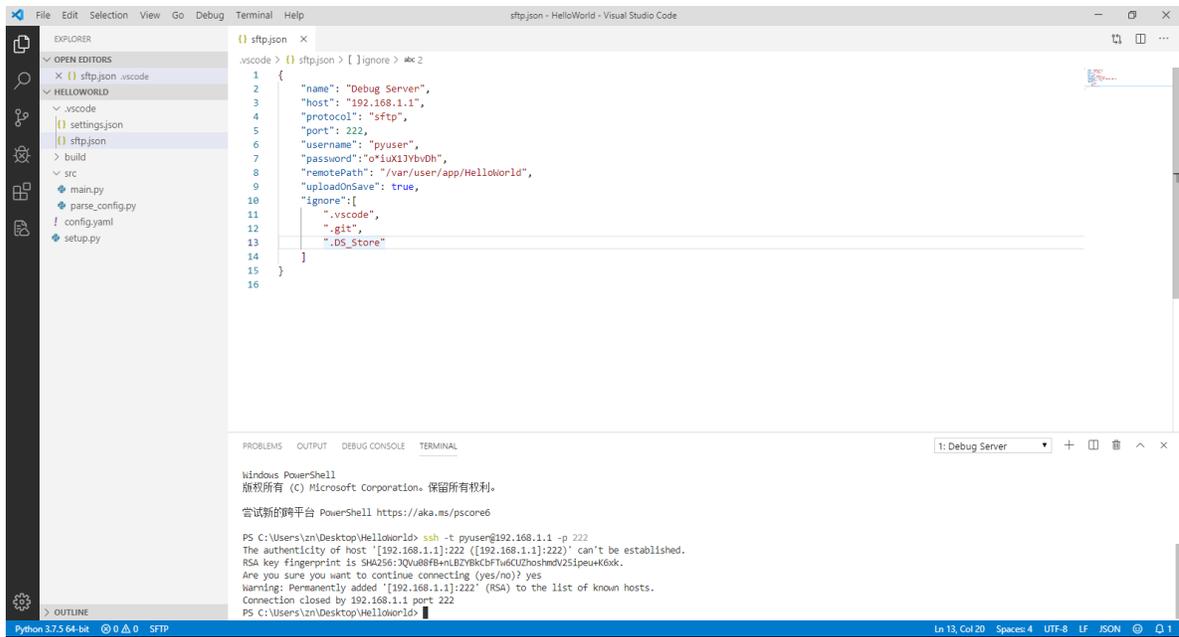
- Step 3: After the configuration is completed and saved, enter the `>SFTP:Open SSH` in Terminal command in the Command Palette to connect to the remote server.



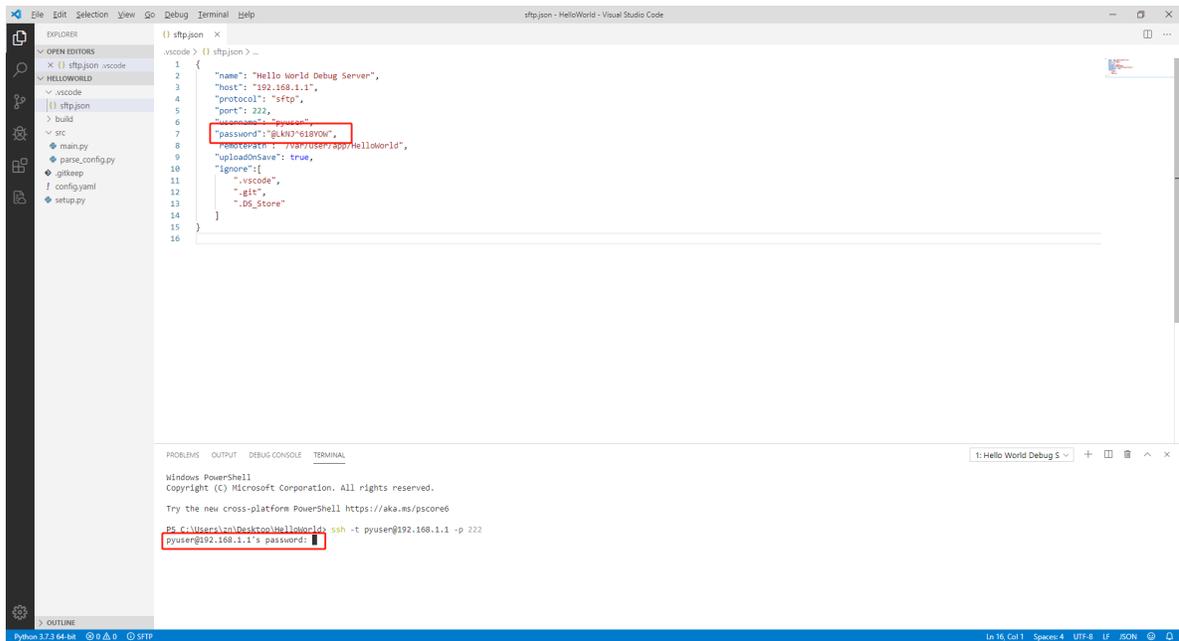
- Step 4: In the Command Palette, you are requested to select the SFTP server to be connected. Select the SFTP server in `sftp.json` and press Enter.



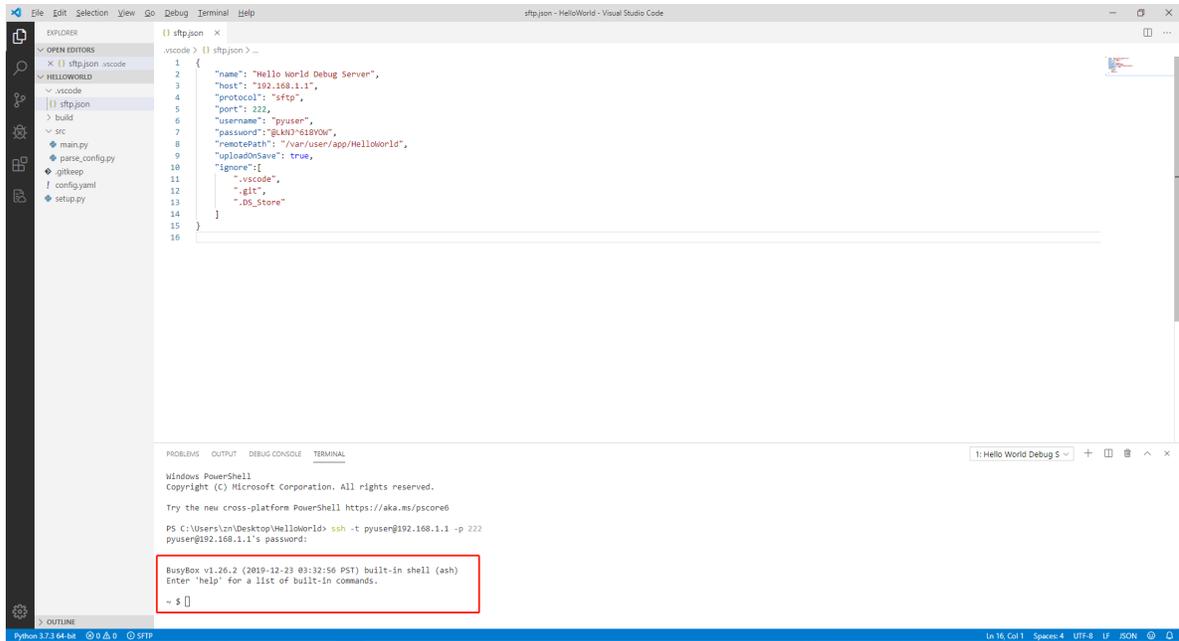
- Step 5: If this is the first time you connect to the server, the **TERMINAL** window prompts you to confirm whether you want to connect to the server. Enter **Yes** and press Enter. Then, enter the `>SFTP:Open SSH in Terminal` command and the IP address of the SFTP server in the Command Palette again.



- Step 6: The **TERMINAL** window prompts you to enter the password. You only need to copy the password in the `sftp.json` file, and paste it to the required place.



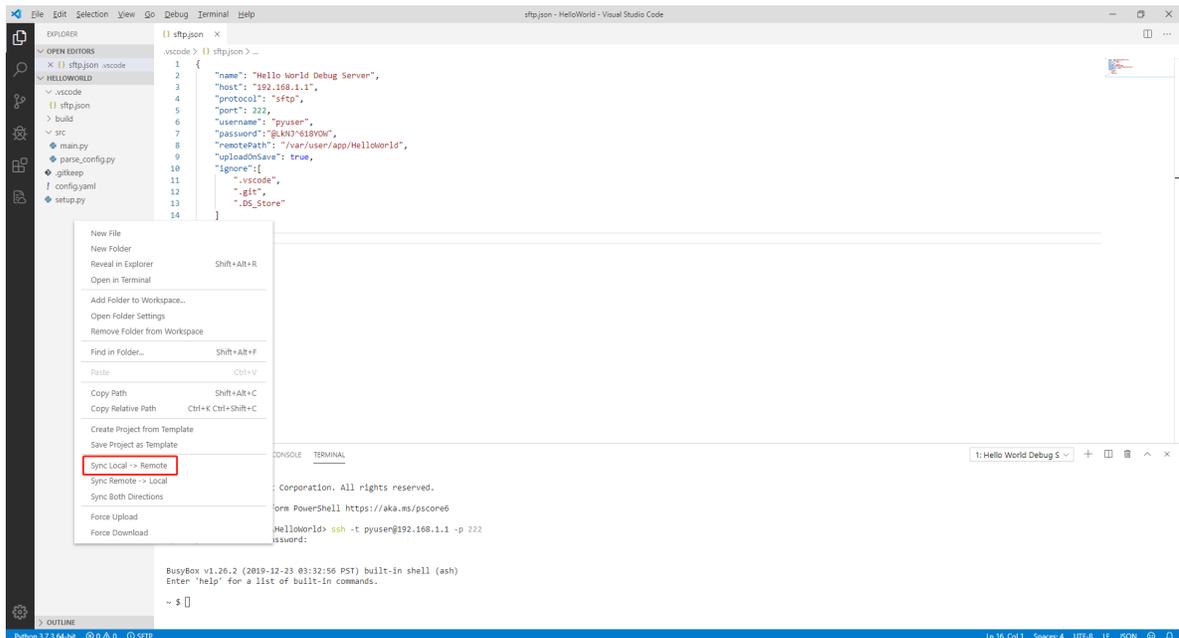
After an SFTP connection with MobiusPi is created, the page is shown as follows:



2.3.2 Debug the code

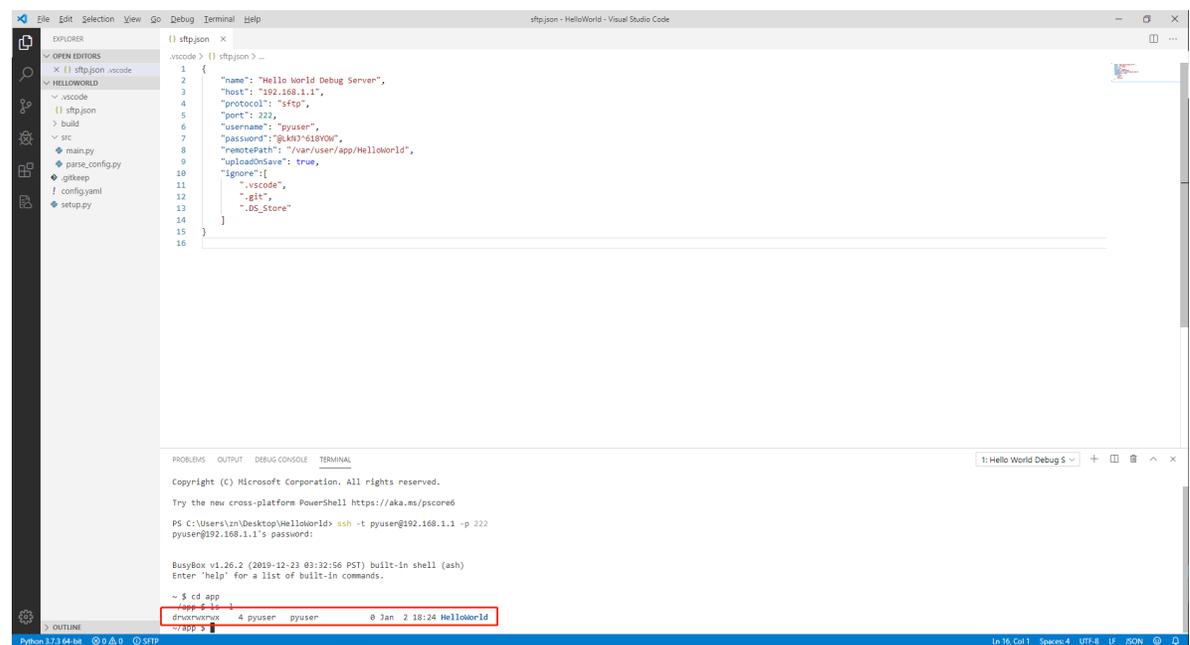
- Step 1: Synchronize the code.

After the SFTP connection is created, right-click the blank area on the left, and choose **Sync Local > Remote** to synchronize the code to the remote server. Later, if you modify or delete code on the local PC, the code changes will be automatically synchronized to the remote server.



You can check whether the remote server has received the App code in the TERMINAL window. In the TERMINAL window, enter the following command to view uploaded App folder information:

```
cd app
ls -l
```



```

1 {
2   "name": "Hello World Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "@L4K7-618Y0W",
8   "remotePath": "C:/Users/pyuser/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16 }

```

```

PS C:\Users\pyuser\Desktop\HelloWorld> ssh -t pyuser@192.168.1.1 -p 222
pyuser@192.168.1.1's password:
BusyBox v1.26.2 (2019-12-23 03:32:56 PST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~$ cd app
~/app $ ls -l
drwxrwxr-x 4 pyuser pyuser 0 Jan 2 18:24 HelloWorld

```

- Step 2: Debug the script in the **TERMINAL** window.
 - Method 1: Use `ptvsd` to debug the script

After the code is synchronized, in the **TERMINAL** window, enter the following command to execute the script on MobiusPi immediately (take IG501 as an example). After the script is executed, you can view the execution result in the **TERMINAL** window: the log “hello world!” is printed. Note: MobiusPi’s Python development environment does not provide `ptvsd` dependent libraries by default, you can refer to [2.8.2 Install the third-party dependency library to SDK](#) to install it yourself

```
python -m ptvsd --host 192.168.1.1 --port 3000 HelloWorld/src/main.py
```

- * 192.168.1.1 is the IP address of the **host** configured in `sftp.json`.
- * 3000 is the recommended debugging port number.
- * `HelloWorld/src/main.py` is the execution path of `mian.py`, which can be adjusted as needed.

For more information about the `ptvsd` plug-in, see [ptvsd user manual](#).

```

1 {
2   "name": "Hello World Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sfsp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "G!K!N3*6!EY0M",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".vs_store"
14  ]
15 }
16

```

```

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\ian\Desktop\HelloWorld> ssh -t pyuser@192.168.1.1 -p 222
pyuser@192.168.1.1's password:

BusyBox v1.26.2 (2019-12-23 03:32:56 PST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

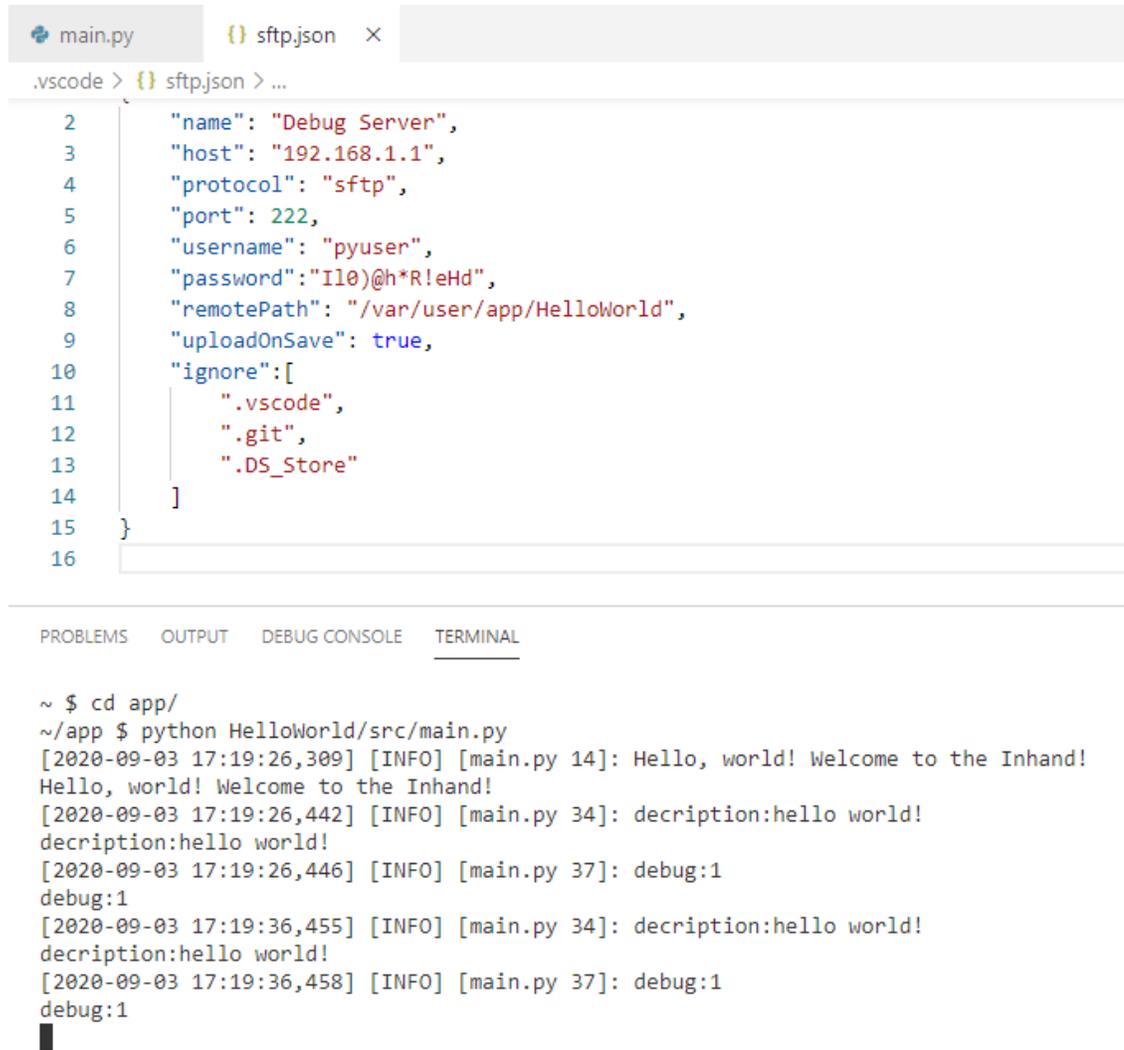
~$ cd app
~/app $ ls -l
drwxr-xr-x 4 pyuser pyuser 0 Jan 2 18:24 HelloWorld
~/app $ python -m ptvsd --host 192.168.1.1 --port 3808 HelloWorld/src/main.py
pydev debugger: CRITICAL WARNING: This version of python seems to be incorrectly compiled (internal generated filenames are not absolute)
pydev debugger: The debugger may still function, but it will work slower and may miss breakpoints.
pydev debugger: Related bug: http://bugs.python.org/issue16688?
-----
pydev debugger: Unable to find real location for: /var/pycom/11b/site-python
pydev debugger: Unable to find real location for: /var/user/11b/python2.7/site-packages
pydev debugger: Unable to find real location for: /root/INOS/109/system/install/PySDK-109/11b/python2_std/posixpath.py
pydev debugger: Unable to find real location for: /root/INOS/109/system/install/PySDK-109/11b/python2_std/genericpath.py

```

- Method 2: Run the script directly

After the code is synchronized, in the **TERMINAL** window, enter the following command to execute the script on IG501 immediately. After the script is executed, you can view the execution result in the **TERMINAL** window: the log “hello world!” is printed. (HelloWorld/src/main.py is the execution path of main.py, which can be adjusted as needed.)

```
python HelloWorld/src/main.py
```



```
main.py sftpjson x
.vscode > sftpjson > ...
2     "name": "Debug Server",
3     "host": "192.168.1.1",
4     "protocol": "sftp",
5     "port": 222,
6     "username": "pyuser",
7     "password": "I!0)@h*R!eHd",
8     "remotePath": "/var/user/app/HelloWorld",
9     "uploadOnSave": true,
10    "ignore":[
11        ".vscode",
12        ".git",
13        ".DS_Store"
14    ]
15 }
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

~ $ cd app/
~/app $ python HelloWorld/src/main.py
[2020-09-03 17:19:26,309] [INFO] [main.py 14]: Hello, world! Welcome to the Inhand!
Hello, world! Welcome to the Inhand!
[2020-09-03 17:19:26,442] [INFO] [main.py 34]: decription:hello world!
decription:hello world!
[2020-09-03 17:19:26,446] [INFO] [main.py 37]: debug:1
debug:1
[2020-09-03 17:19:36,455] [INFO] [main.py 34]: decription:hello world!
decription:hello world!
[2020-09-03 17:19:36,458] [INFO] [main.py 37]: debug:1
debug:1
█
```

- Step 3: After debugging, press `Ctrl + C` on the terminal to terminate the debugging process.

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the project structure for 'HelloWorld', including files like `main.py`, `parse_config.py`, `config.yaml`, and `setup.py`. The Editor pane shows the `sftp.json` file with the following content:

```

1 {
2   "name": "Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "dypmPEA6h$1t",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

The Terminal pane shows the following output:

```

[2020-05-26 10:46:39,502] [INFO] [main.py 35]: debug:1
debug:1
[2020-05-26 10:46:49,509] [INFO] [main.py 32]: decription:hello world!
decription:hello world!
[2020-05-26 10:46:49,514] [INFO] [main.py 35]: debug:1
debug:1
[2020-05-26 10:46:59,533] [INFO] [main.py 32]: decription:hello world!
decription:hello world!
[2020-05-26 10:46:59,542] [INFO] [main.py 35]: debug:1
debug:1
^CTraceback (most recent call last):
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 193, in _run_module_as_main
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 85, in _run_code
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 446, in <module>
    main(sys.argv)
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 432, in main
    run()
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 316, in run_file
    runpy.run_path(target, run_name='__main__')
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 263, in run_path
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 96, in _run_module_code
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 85, in _run_code
  File "app/HelloWorld/src/main.py", line 41, in <module>
    main()
  File "app/HelloWorld/src/main.py", line 38, in main
    time.sleep(10)
KeyboardInterrupt
~ $

```

A red box highlights the traceback section, and a red arrow points to the prompt `~ $` with the text "Terminate the debugging".

2.4 Build an App release package

After debugging, you can build an App release package to quickly deploy the App to other MobiusPi devices.

- Step 1: Build an App release package.

In the **TERMINAL** window, run the `build_py_app.sh HelloWorld` command to build an App release package. (That is, the `build_py_app.sh` Python App name.)

```

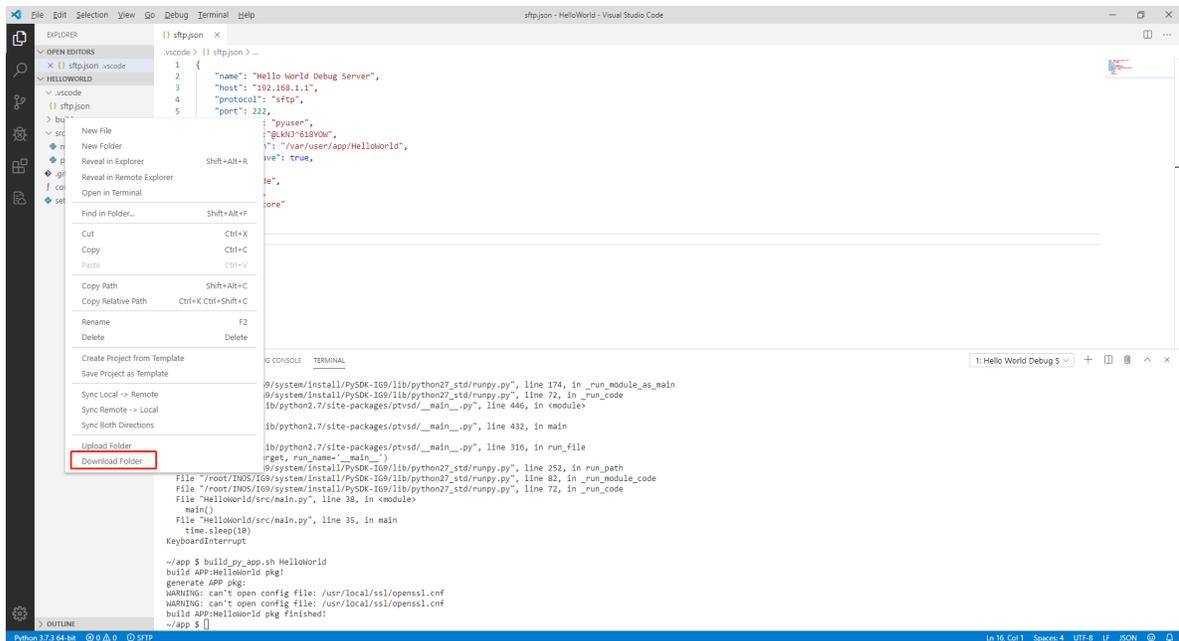
sftp.json
{
  "name": "Debug Server",
  "host": "192.168.1.1",
  "protocol": "sftp",
  "port": 222,
  "username": "pyuser",
  "password": "dypmPEA6h$1t",
  "remotePath": "/var/user/app/HelloWorld",
  "uploadOnSave": true,
  "ignore": [
    ".vscode",
    ".git",
    ".DS_Store"
  ]
}

[2020-05-26 10:46:59,533] [INFO] [main.py 32]: decription:hello world!
decription:hello world!
[2020-05-26 10:46:59,542] [INFO] [main.py 35]: debug:1
debug:1
^CTraceback (most recent call last):
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 193, in _run_module_as_main
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 85, in _run_code
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 446, in <module>
    main(sys.argv)
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 432, in main
    run()
  File "/var/pycore/lib/python3.7/site-packages/ptvsd/__main__.py", line 316, in run_file
    runpy.run_path(target, run_name='__main__')
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 263, in run_path
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 96, in _run_module_code
  File "/root/INOS/IG9/system/install/PySDK-IG9/lib/python37_std/runpy.py", line 85, in _run_code
  File "app/HelloWorld/src/main.py", line 41, in <module>
    main()
  File "app/HelloWorld/src/main.py", line 38, in main
    time.sleep(10)
KeyboardInterrupt

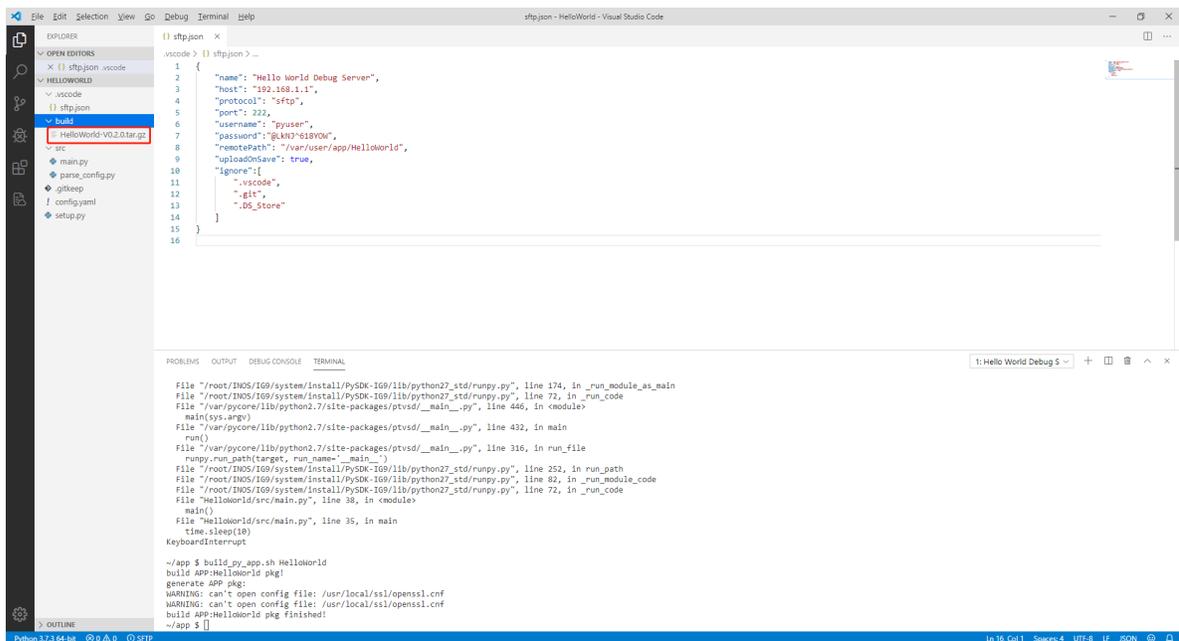
~ $ build py_app.sh HelloWorld
build APP:HelloWorld pkg!
generate APP pkg:
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
build APP:HelloWorld pkg finished!
    
```

- Step 2: Download the App release package.

After the command is executed, the App release package is automatically generated in the build directory on the remote server. Right-click the **build** folder and select **Download Folder** to download the built App release package to the local PC for subsequent deployment.



After the package is downloaded, you can view the HelloWorld App release package in the build directory.



2.5 Deploy the App on the MobiusPi web page

Run the `main.py` script. Or, after the App release package is built, the App is automatically generated on the connected MobiusPi, but this App cannot be started. Refer to the following links to deploy the App to MobiusPi:

- [Deploy an App on IG501](#)

- Deploy an App on IG502
- Deploy an App on IG902

2.6 View the App running status

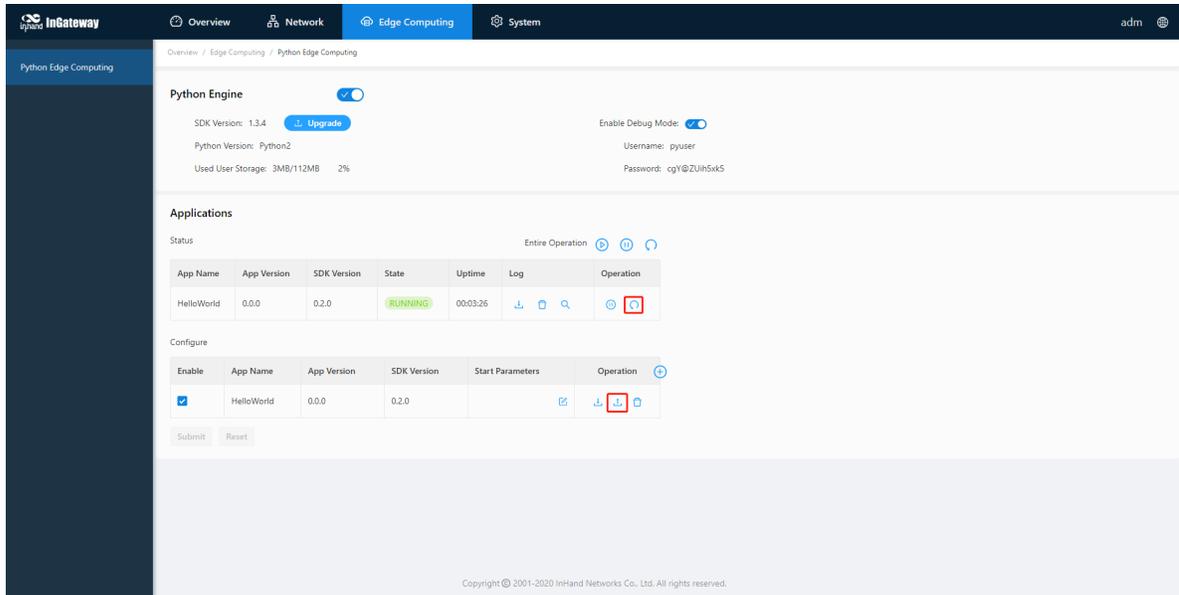
You can view the App running status on the **Edge Computing > Python Edge Computing** page in MobiusPi.

The screenshot shows the InGateway interface for Python Edge Computing. The top navigation bar includes Overview, Network, Edge Computing, and System. The main content area is titled 'Python Edge Computing' and contains the following sections:

- Python Engine:** SDK Version: 1.3.4 (Upgrade), Python Version: Python2, Used User Storage: 3MB/112MB (2%), Enable Debug Mode: checked, Username: pyuser, Password: cgY@ZUih5sk5.
- Applications:** A table with columns: App Name, App Version, SDK Version, State, Uptime, Log, and Operation. The 'HelloWorld' application is highlighted with a red box, showing a 'RUNNING' state and '00:00:06' uptime.
- Configure:** A table with columns: Enable, App Name, App Version, SDK Version, Start Parameters, and Operation. The 'HelloWorld' application is checked as enabled.

Click **View Logs** to view the App running logs.

This screenshot is identical to the previous one, but the 'View Logs' icon (represented by a magnifying glass) in the 'Log' column of the 'HelloWorld' application row is highlighted with a red box.



After restart, the HelloWorld App runs with the updated configuration file and prints a log “hello inhand!” every 10s.

```

DEBUG:root:Hello, world! Welcome to the Inhand!
INFO:root:decription:hello world!
INFO:root:debug:1
INFO:root:decription:hello world!
INFO:root:debug:1
INFO:root:decription:hello world!
INFO:root:debug:1
INFO:root:decription:hello world!
DEBUG:root:Hello, world! Welcome to the Inhand!
INFO:root:decription:hello inhand!
INFO:root:debug:1
INFO:root:decription:hello inhand!
INFO:root:debug:1
INFO:root:decription:hello inhand!
INFO:root:debug:1
INFO:root:decription:hello inhand!
INFO:root:debug:1
INFO:root:decription:hello inhand!

```

2.8 Annex

- *2.8.1 Install a third-party dependency library for the specified App*
- *2.8.2 Install the third-party dependency library to SDK*
- *2.8.3 Enable automatic code completion*

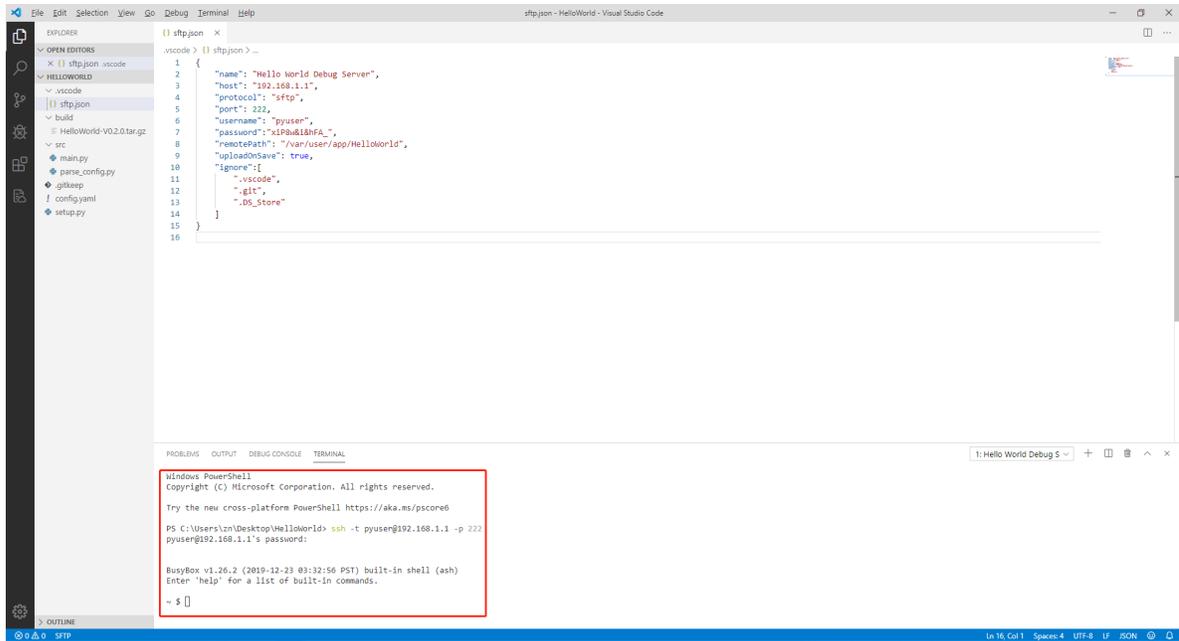
2.8.1 Install a third-party dependency library for the specified App

To install a third-party dependency library for the specified App, you need to enable the debugging mode for MobiusPi and connect to the Internet. The following takes installing the `x1rd` dependency library to the HelloWorld App as an example to describe how to install a third-party dependency library to a specified App.

The screenshot shows the InGateway interface for Python Edge Computing. The 'Python Engine' section is active, showing SDK Version 1.3.4, Python Version Python2, and User Storage 3MB/112MB. The 'Enable Debug Mode' toggle is checked and highlighted with a red box. Below this, the 'Applications' section shows a table with columns for App Name, App Version, SDK Version, State, Uptime, Log, and Operation, but it is currently empty. The 'Configure' section also shows a table with columns for Enable, App Name, App Version, SDK Version, Start Parameters, and Operation, which is also empty.

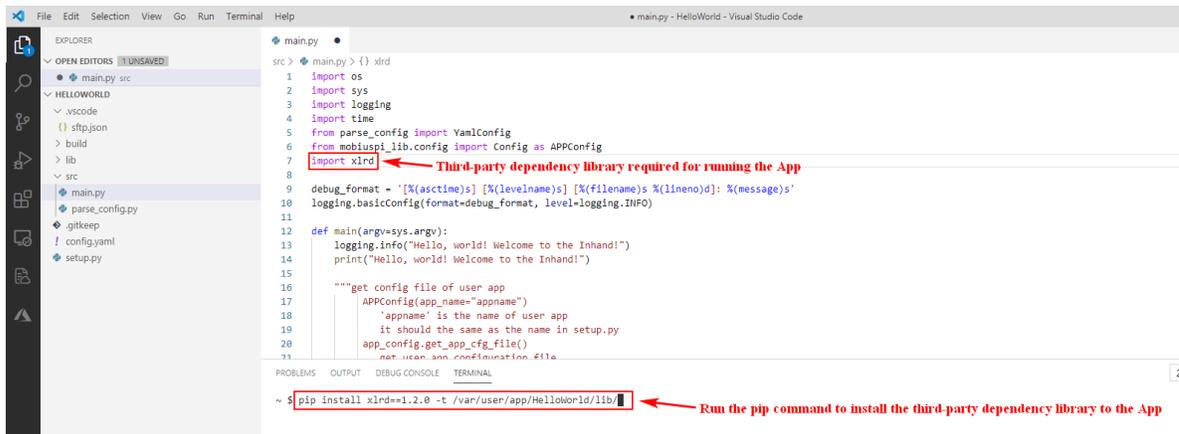
The screenshot shows the InGateway interface for Network Interfaces Cellular. The 'Status' section shows 'Connected' highlighted with a red box. Below this, the 'Enable Cellular' toggle is checked. The 'Profile' section shows a table with columns for Index, Network Type, APN, Access Number, Auth Method, Username, Password, and Operation. The table contains one row with Index 1, Network Type GSM, APN 3gnet, Access Number *99**1#, Auth Method Auto, Username gprs, and Password *****.

- Step 1: Use VS Code to connect to MobiusPi over SFTP. For more information, see [Create an SFTP connection](#).



- Step 2: Run `pip install + dependency library name + ==version number + -t + App's lib folder path` and press Enter to install the dependency library to the specified App. (If the version number is not added, after the pip command is run, the latest dependency library is automatically installed.)

```
pip install xlrd==1.2.0 -t /var/user/app/HelloWorld/lib/
```



- Step 3: Then, the dependency library is automatically downloaded and installed. After the library is installed, the following figure is displayed:

```

src > main.py > ...
1 import os
2 import sys
3 import logging
4 import time
5 from parse_config import YamlConfig
6 from mobiuspi_lib.config import Config as APPConfig
7 import xlrD
8
9 debug_format = '%(asctime)s %(levelname)s %(filename)s %(lineno)d: %(message)s'
10 logging.basicConfig(format=debug_format, level=logging.INFO)
11
12 def main(argv=sys.argv):
13     logging.info("Hello, world! Welcome to the Inhand!")
14     print("Hello, world! Welcome to the Inhand!")
15
16     """get config file of user app
17     APPConfig(app_name="appname")
18     'appname' is the name of user app
19     it should be the same as the name in setup.py
20     app_config.get_app_cfg_file()
21     not user app confirmation file
"""
~ $ pip install xlrD==1.2.0 -t /var/user/app/HelloWorld/lib/
Collecting xlrD==1.2.0
Using cached https://files.pythonhosted.org/packages/b0/16/63576a1a081752e34bf8ea62e367997530dc553b689356b9879339cf45a4/xlrD-1.2.0-py3-none-any.whl
Installing collected packages: xlrD
Successfully installed xlrD-1.2.0
~ $

```

Installed successfully

- Step 4: Run the `export` command to set an environment variable for the App. In the TERMINAL window, run the following command (replace **HelloWorld** with the actual App name).

```

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/var/user/app/HelloWorld/lib/
export PYTHONPATH=$PYTHONPATH:/var/user/app/HelloWorld/lib/

```

```

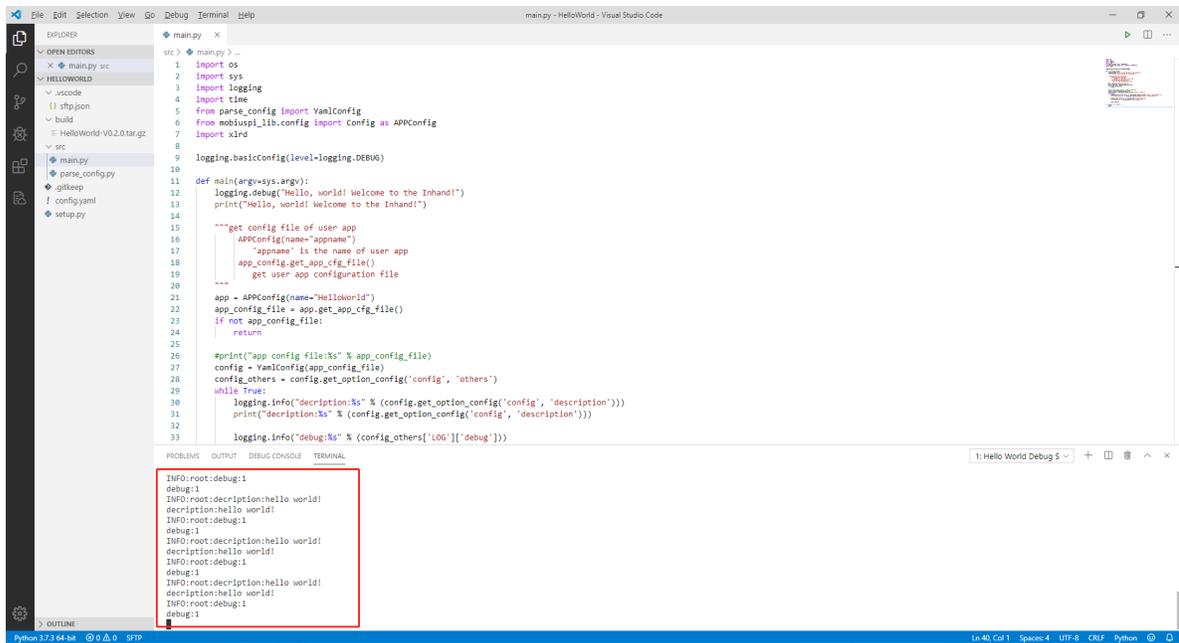
src > main.py > ...
1 import os
2 import sys
3 import logging
4 import time
5 from parse_config import YamlConfig
6 from mobiuspi_lib.config import Config as APPConfig
7 import xlrD
8
9 debug_format = '%(asctime)s %(levelname)s %(filename)s %(lineno)d: %(message)s'
10 logging.basicConfig(format=debug_format, level=logging.INFO)
11
12 def main(argv=sys.argv):
13     logging.info("Hello, world! Welcome to the Inhand!")
14     print("Hello, world! Welcome to the Inhand!")
15
16     """get config file of user app
17     APPConfig(app_name="appname")
18     'appname' is the name of user app
19     it should be the same as the name in setup.py
20     app_config.get_app_cfg_file()
21     not user app confirmation file
"""
~ $ pip install xlrD==1.2.0 -t /var/user/app/HelloWorld/lib/
Collecting xlrD==1.2.0
Using cached https://files.pythonhosted.org/packages/b0/16/63576a1a081752e34bf8ea62e367997530dc553b689356b9879339cf45a4/xlrD-1.2.0-py2.py3-none-any.whl
Installing collected packages: xlrD
Successfully installed xlrD-1.2.0
~ $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/var/user/app/HelloWorld/lib/
~ $ export PYTHONPATH=$PYTHONPATH:/var/user/app/HelloWorld/lib/
~ $

```

Configure the environment variable for the App

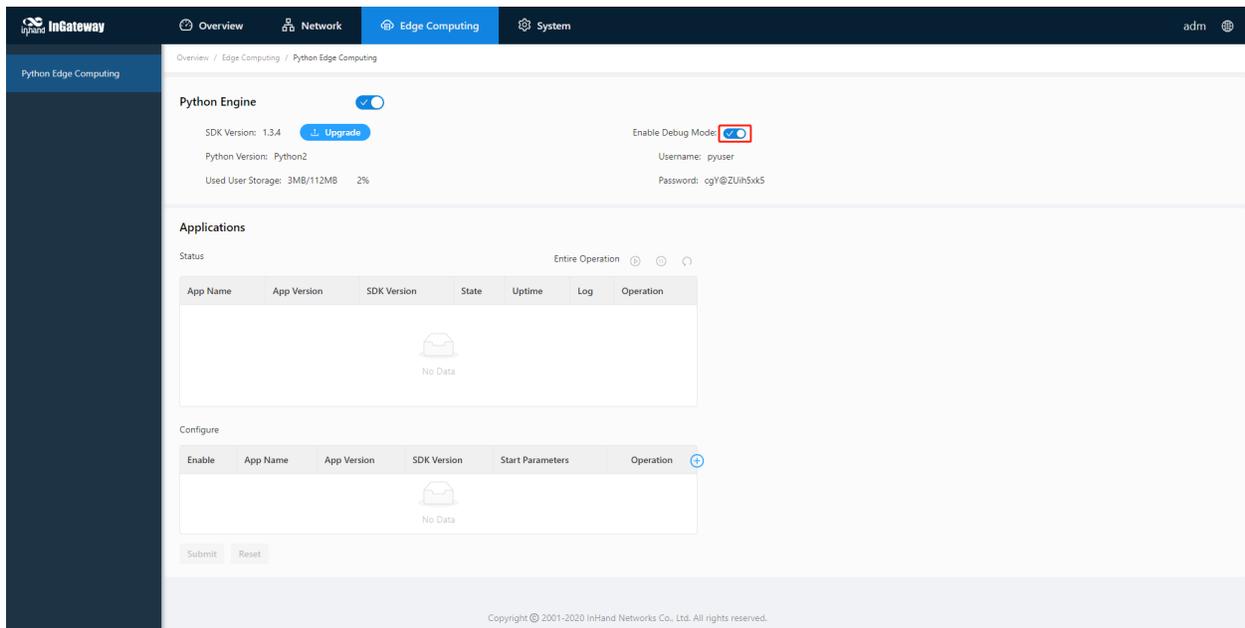
After the third-party dependency library is installed for the specified App, you must configure a corresponding environment variable before debugging this App; otherwise, this App cannot run during debugging. When multiple third-party dependency libraries are installed, you just need to add the environment variable once. After an SFTP connection is created again, you need to configure the environment variable again. When the App is started in MobiusPi, the environment variable of the third-party dependency library is automatically added to the App's lib folder. (This method is mainly used for third-party dependent libraries where the App needs to install a special version)

- Step 5: Run the code to check whether the App runs normally.



2.8.2 Install the third-party dependency library to SDK

To install a third-party dependency library to SDK, you need to enable the debugging mode for MobiusPi and connect to the Internet. The following takes installing the `x1rd` dependency library to SDK as an example to describe how to install a third-party dependency library to SDK.



The screenshot shows the InGateway Network configuration page for Cellular. The status is 'Connected'. The modem information includes IMEI Code: 860588047910875, IMSI Code: 460110923689639, and ICCID Code. The network information includes IP Address: 10.139.6.60, Netmask: 255.255.255.255, and MTU: 1500. The 'Enable Cellular' toggle is turned on. The profile configuration table is as follows:

Index	Network Type	APN	Access Number	Auth Method	Username	Password	Operation
1	GSM	3gnet	*99**1#	Auto	gprs	*****	[Edit] [Delete]

Additional configuration options include Network Type (Auto), Profile (Auto), Roaming (checked), PIN code (****), Static IP (unchecked), Connection Mode (Always Online), and Redial Interval (10 s).

- Step 1: Use VS Code to connect to MobiusPi over SFTP. For more information, see [Create an SFTP connection](#).

The screenshot shows Visual Studio Code with an SFTP connection configuration in `sftp.json`. The configuration is as follows:

```

1 {
2   "name": "Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "asjuc5QW4+",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

The terminal window shows the following output:

```

Microsoft PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

PS C:\Users\lan\Desktop\HelloWorld> ssh -t pyuser@192.168.1.1 -p 222
pyuser@192.168.1.1's password:

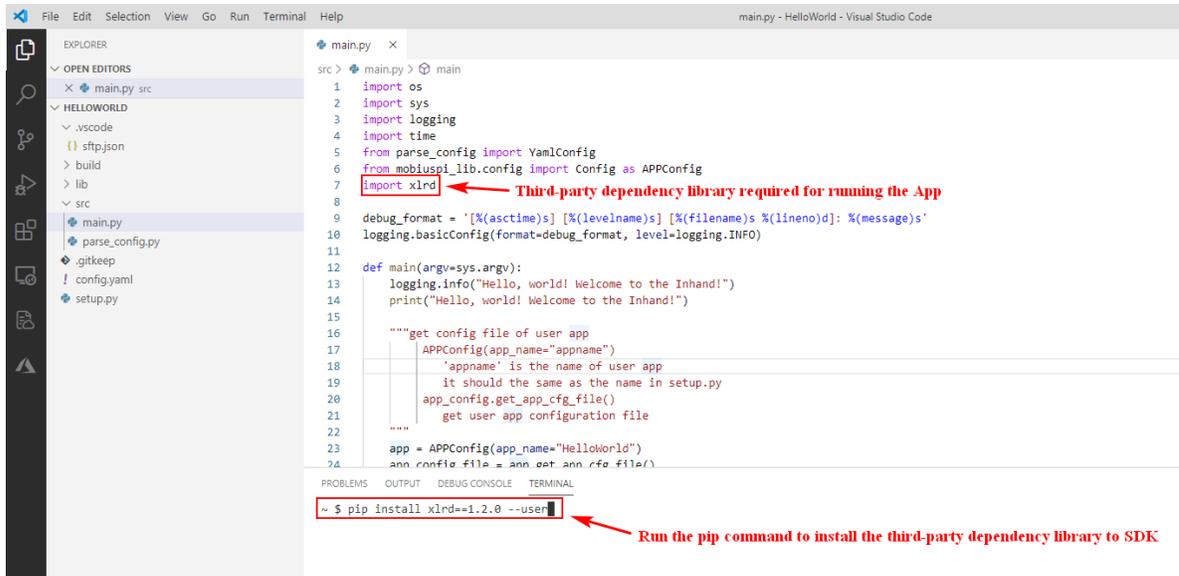
BusyBox v1.26.2 (2019-11-11 16:48:18 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ $

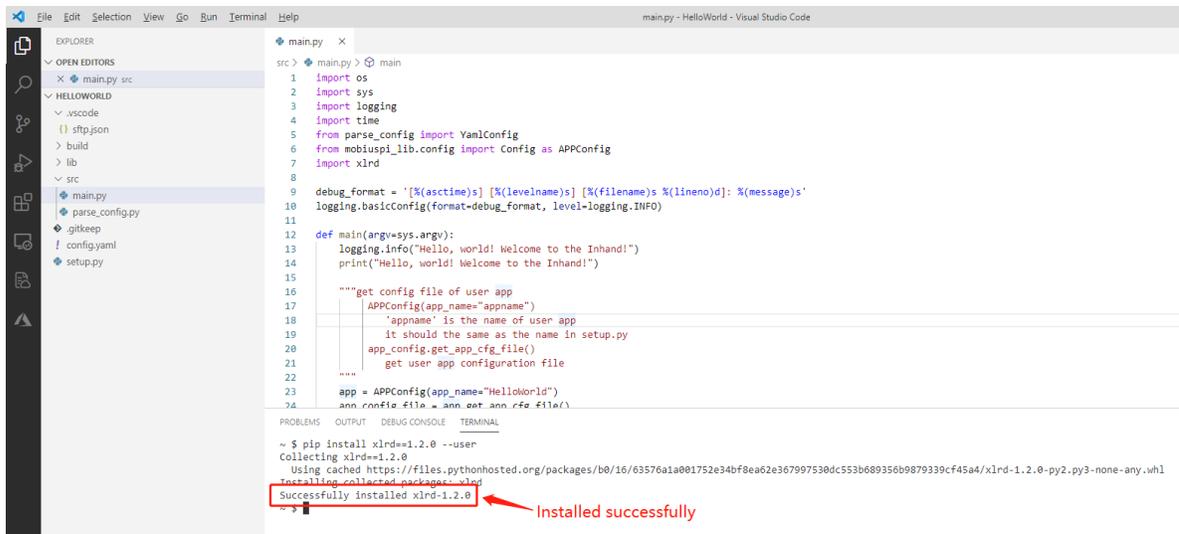
```

- Step 2: Run `pip install + dependency library name + ==version number + --user` and press Enter to install the dependency library to SDK. (If the version number is not added, after the pip command is run, the latest dependency library is automatically installed.)

```
pip install xlrd==1.2.0 --user
```



- Step 3: Then, the dependency library is automatically downloaded and installed. After the library is installed, the following figure is displayed:



- Step 4: Run the code to check whether the App runs normally.

The screenshot shows the Visual Studio Code interface with a Python file named `main.py` open. The code in the editor includes imports for `os`, `sys`, `logging`, and `time`, along with configuration for `APPConfig`. The terminal output shows the execution of `pip install xlrd==1.2.0` and `python app/Helloworld/src/main.py`, resulting in a series of log messages and debug statements.

```

src > main.py > main
1 import os
2 import sys
3 import logging
4 import time
5 from parse_config import YamlConfig
6 from mobiuspi_lib.config import Config as APPConfig
7 import xlrd
8
9 debug_format = '[%(asctime)s] [%(levelname)s] [%(filename)s %(lineno)d]: %(message)s'
10 logging.basicConfig(format=debug_format, level=logging.INFO)
11
12 def main(argv=sys.argv):
13     logging.info("Hello, world! Welcome to the Inhand!")
14     print("Hello, world! Welcome to the Inhand!")
15
16     """get config file of user app
17     APPConfig(app_name="appname")
18     'appname' is the name of user app
19     it should be the same as the name in setup.py
20     app_config.get_app_cfg_file()
21     get user app configuration file
22     """
23     app = APPConfig(app_name="Helloworld")
24     app.config_file = app.get_app_cfg_file()

```

```

~ $ pip install xlrd==1.2.0 --user
Collecting xlrd==1.2.0
Using cached https://files.pythonhosted.org/packages/b0/16/635761a0e1752e34bf8ea2e367997530dc553b689356b9879339cf45a4/xlrd-1.2.0-py2.py3-none-any.whl
Installing collected packages: xlrd
Successfully installed xlrd-1.2.0
~ $ python app/Helloworld/src/main.py
[2020-05-26 19:39:09,221] [INFO] [main.py 12]: Hello, world! Welcome to the Inhand!
Hello, world! Welcome to the Inhand!
[2020-05-26 19:39:09,355] [INFO] [main.py 32]: decription:hello world!
decription:hello world!
[2020-05-26 19:39:09,358] [INFO] [main.py 35]: debug:1
debug:1
[2020-05-26 19:39:19,364] [INFO] [main.py 32]: decription:hello world!
decription:hello world!
[2020-05-26 19:39:19,367] [INFO] [main.py 35]: debug:1
debug:1

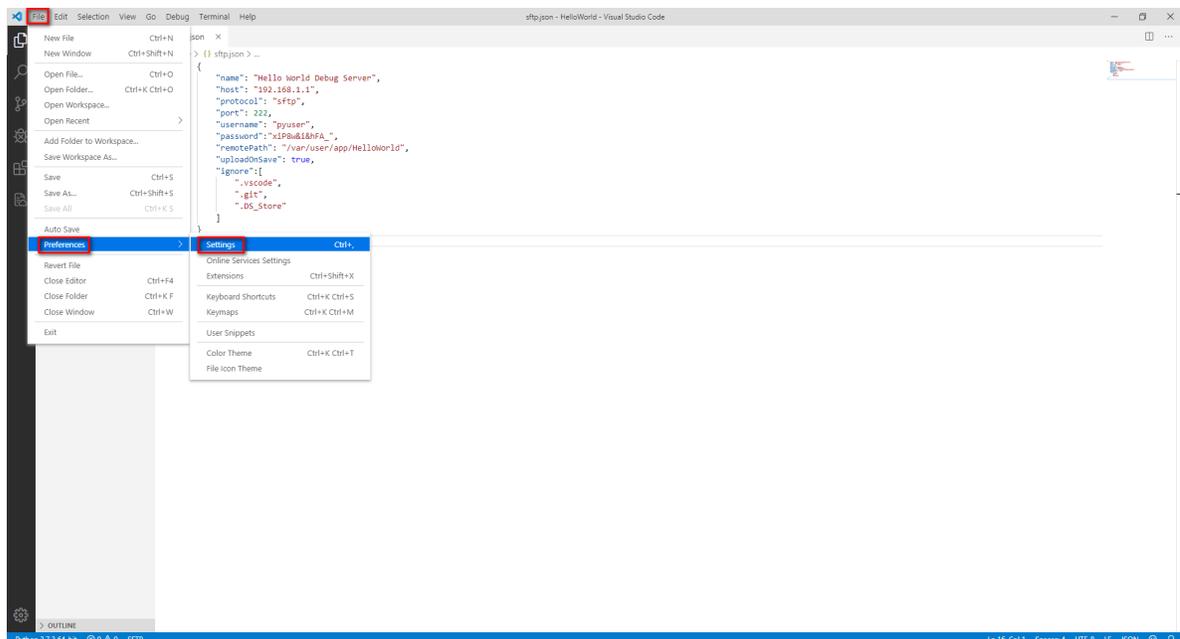
```

Note: When using this method to install a third-party dependency library, the packaged App release package will be automatically installed into the MobiusPi when it is deployed on other MobiusPi.

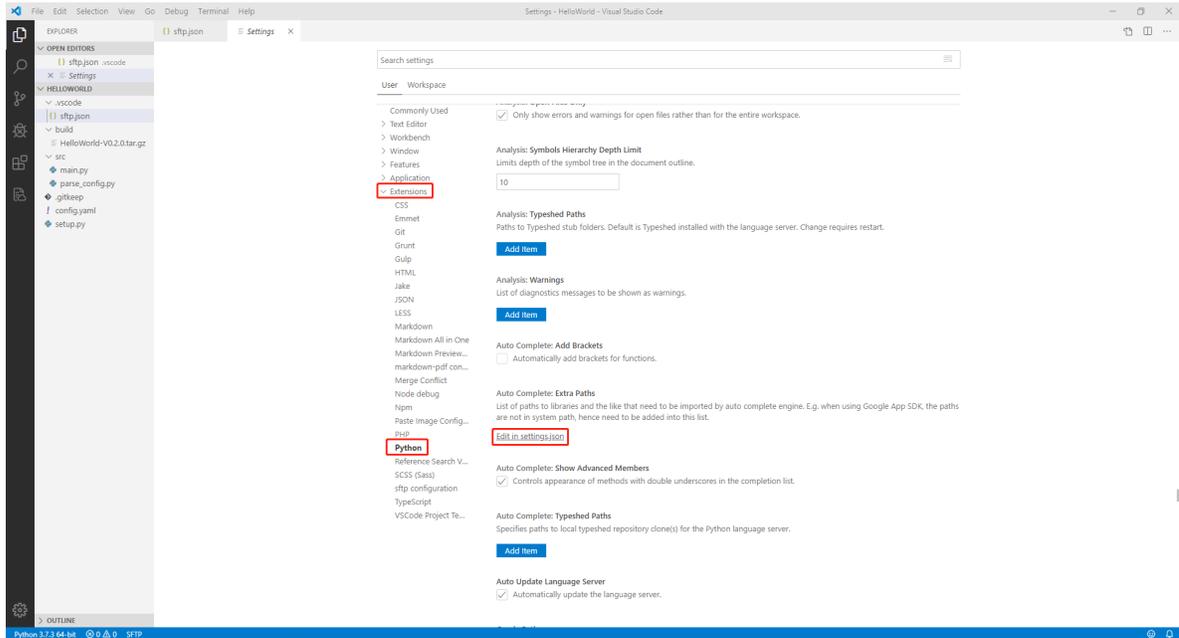
2.8.3 Enable automatic code completion

To improve the encoding efficiency, you can implement automatic code completion by using Python extensions.

- Step 1: Choose **Files > Preferences > Settings** to enter the Settings page.

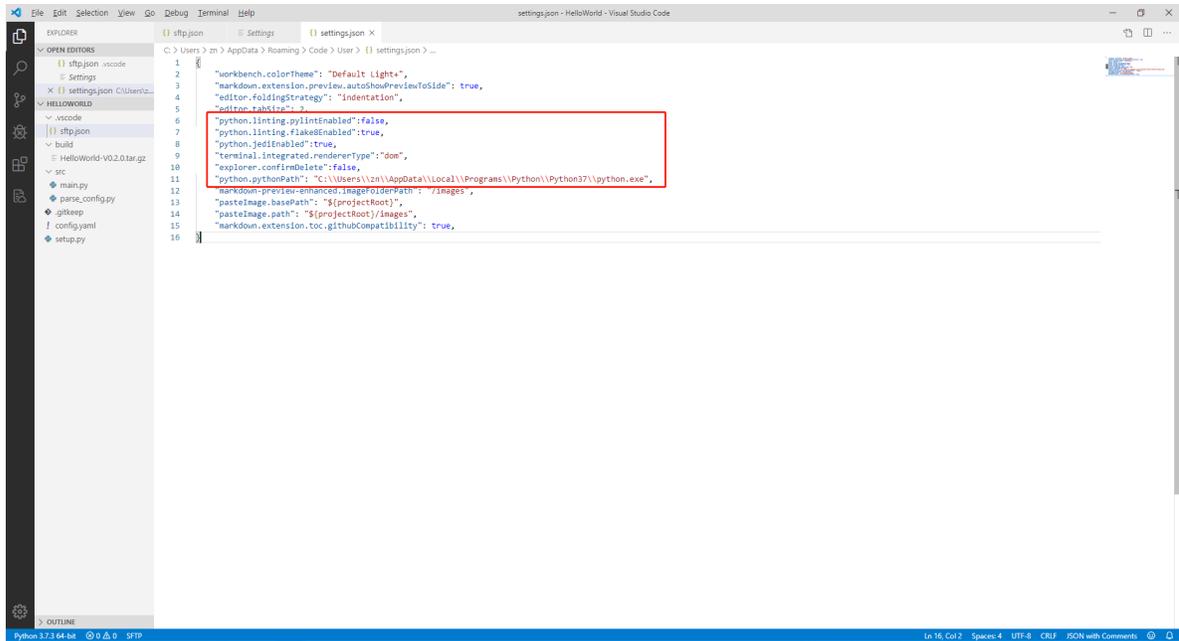


- Step 2: On the Settings page, choose **Extensions > Python**, locate **Auto Complete: Extra Paths**, and then click **Edit in settings.json**.



Add the following configuration items to **settings.json** and save the configuration (python.pythonPath is the installation path of the Python interpreter).

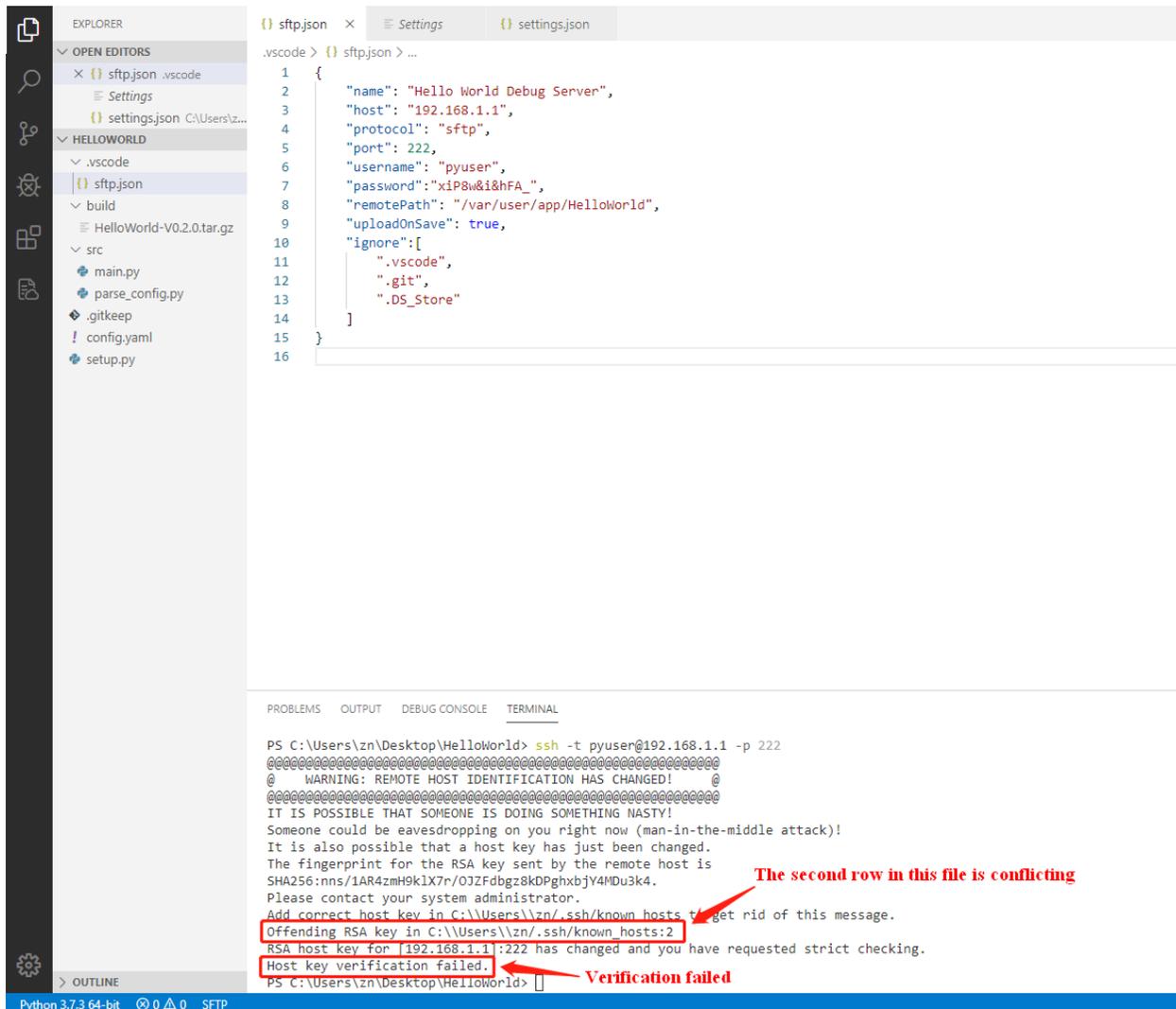
```
"python.linting.pylintEnabled":false,
"python.linting.flake8Enabled":true,
"python.jediEnabled":true,
"terminal.integrated.rendererType":"dom",
"explorer.confirmDelete":false,
"python.pythonPath":"C:/Users/zn/AppData/Local/Programs/Python/Python37",
```



1.1.3 FAQ

- During SFTP connection building, it prompts “REMOTE HOST IDENTIFICATION HAS CHANGED and Host key verification failed”
- When synchronizing the code to the remote server, it prompts “All configured authentication methods failed”
- How to call the serial port and network port of MobiusPi
- When creating an SFTP connection with MobiusPi, it prompts “SSH error”

Q1: During SFTP connection building, it prompts “REMOTE HOST IDENTIFICATION HAS CHANGED and Host key verification failed” . What should I do?



A1: The reason is that the MobiusPi key has been updated but the PC key does not, resulting in authentication failure. To solve the problem, just delete the row with key conflict from the key file. Press Ctrl and click the conflicting item to quickly access the link.



After deletion, run the >SFTP:Open SSH in Terminal command. Then, an SFTP connection is created.

```

1 {
2   "name": "Hello World Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "xIP8&18hFA_",
8   "remotePath": "/var/user/app/Helloworld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

```

Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:nms/iAR4zmHkLX7r/0Z7Fbqz8KDPghubjY4NDu3k4.
Please contact your system administrator.
Add correct host key in C:\Users\lzn/.ssh/known_hosts to get rid of this message.
Offending RSA key in C:\Users\lzn/.ssh/known_hosts:2
RSA host key for [192.168.1.1]:222 has changed and you have requested strict checking.
Host key verification failed.
PS C:\Users\lzn\Desktop\HelloWorld> ssh -t pyuser@192.168.1.1 -p 222
The authenticity of host '[192.168.1.1]:222 ([192.168.1.1]:222)' can't be established.
RSA key fingerprint is SHA256:nms/iAR4zmHkLX7r/0Z7Fbqz8KDPghubjY4NDu3k4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.1.1]:222' (RSA) to the list of known hosts.
pyuser@192.168.1.1's password:

```

Q2: After the SFTP connection is created, right-click on a blank area and choose Sync Local > Remote to synchronize the code to the remote server. It prompts “All configured authentication methods failed” . What should I do?

```

1 {
2   "name": "Hello World Debug Server",
3   "host": "192.168.1.1",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "xIP8&18hFA_",
8   "remotePath": "/var/user/app/Helloworld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

```

PS C:\Users\lzn\Desktop\HelloWorld> ssh -t pyuser@192.168.1.1 -p 222
pyuser@192.168.1.1's password:

BusyBox v1.26.2 (2019-12-23 03:32:56 PST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ $

```

[192.168.1.1]: All configured authentication methods failed
Source: SFTP (Extension) [Detail](#)

A2: Ensure that the password in the `sftp.json` file is same to that on MobiusPi. After the passwords are the same, synchronize the code again.

Q3: How to call the serial port and network port of MobiusPi?

A3: The RS485 serial port name of IG902 is `/dev/tty03` and the RS232 serial port name is `/dev/tty01`; The RS485 serial port name of IG502 is `/dev/tty03` and the RS232 serial port name is `/dev/tty01`; The RS485 serial port name of IG501 is `/dev/tty01` and the RS232 serial port name is `/dev/tty05`. The serial port and network port can be called with the standard Python serial port/network port usage. For example, use the `pyserial` library to call the serial port.

Q4: When creating an SFTP connection with MobiusPi, it prompts “SSH error”, as shown in the following figure. What should I do?

The screenshot shows a Visual Studio Code editor with a file named `sftp.json` open. The file contains the following JSON configuration:

```

1 {
2   "name": "Hello World Debug Server",
3   "host": "10.5.16.79",
4   "protocol": "sftp",
5   "port": 222,
6   "username": "pyuser",
7   "password": "X10*b0n*D",
8   "remotePath": "/var/user/app/HelloWorld",
9   "uploadOnSave": true,
10  "ignore": [
11    ".vscode",
12    ".git",
13    ".DS_Store"
14  ]
15 }
16

```

Below the editor, the terminal window shows the following output:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\zn\Desktop\HelloWorld> ssh -t pyuser@10.5.16.79 -p 222
ssh: The term 'ssh' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ ssh -t pyuser@10.5.16.79 -p 222
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (ssh:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\zn\Desktop\HelloWorld>

```

A4: Install OpenSSH to support the SSH protocol. Download OpenSSH from <https://www.openssh.com>.

1.2 MobiusPi API Manual

- *MobiusPi API Manual*
 - *Overview*
 - *Installation SDK*
 - *Python requirements*
 - *1. Basic*
 - * *Getting started*
 - * *Instructions for use*

- * *Method description*
 - *reboot()*
 - *Description*
 - *Request parameters*
 - *Returns*
- 2. *Cellular*
 - * *Getting started*
 - * *Instructions for use*
 - * *Method description*
 - *get_modem()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *Exception*
 - *get_network()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *Exception*
- 3. *Config*
 - * *Getting started*
 - * *Instructions for use*
 - * *Method description*
 - *get_app_path()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *get_app_log_path()*
 - *Description*
 - *Request parameters*

- *Returns*
- *get_app_cfg_path()*
- *Description*
- *Request parameters*
- *Returns*
- *get_app_cfg_file()*
- *Description*
- *Request parameters*
- *Returns*
- *get_default_app_cfg_file()*
- *Description*
- *Request parameters*
- *Returns*
- *get_app_db_base_path()*
- *Description*
- *Request parameters*
- *Returns*
- *get_app_db_path()*
- *Description*
- *Request parameters*
- *Returns*

– 4. *GPS*

- * *Getting started*
- * *Instructions for use*
- * *Method description*
 - *get_position_status()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *Exception*

– 5. I/O

* *Getting started*

* *Instructions for use*

* *Method description*

· *get_io_list()*

· *Description*

· *Request parameters*

· *Returns*

· *Exception*

· *get_io_info(io_name)*

· *Description*

· *Request parameters*

· *Returns*

· *Exception*

· *get_all_io_info()*

· *Description*

· *Request parameters*

· *Returns*

· *Exception*

· *setup_digital_io(io_name, mode)*

· *Description*

· *Request parameters*

· *Returns*

· *Exception*

· *setup_analog_io(io_name, mode)*

· *Description*

· *Request parameters*

· *Returns*

· *Exception*

· *read_io(io_name)*

- *Description*
- *Request parameters*
- *Returns*
- *Exception*
- *write_io(io_name)*
- *Description*
- *Request parameters*
- *Returns*
- *Exception*
- 6. *Serial*
 - * *Getting started*
 - * *Instructions for use*
 - * *Method description*
 - *get_serial232_path()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *Exception*
 - *get_serial485_path()*
 - *Description*
 - *Request parameters*
 - *Returns*
 - *Exception*
- 7. *SystemInfo*
 - * *Getting started*
 - * *Instructions for use*
 - * *Method description*
 - *get_system_info()*
 - *Description*
 - *Request parameters*

- *Returns*
- *Exception*

1.2.1 Overview

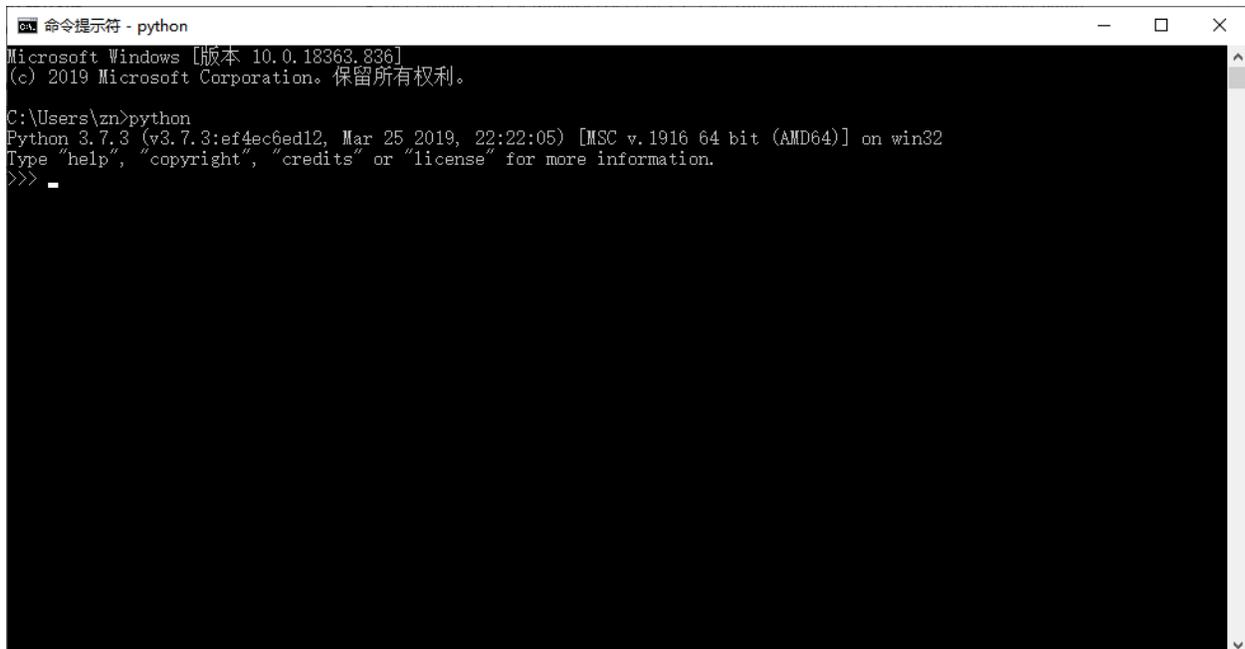
MobiusPi is the secondary development platform for the InGateway series product. This document explains how to call the APIs of the `mobiuspi_lib` library. This library is used to retrieve MobiusPi's runtime status and call MobiusPi's physical interfaces.

1.2.2 Installation SDK

InHand Networks provides the software development kit (SDK) that includes the `mobiuspi_lib` library. Please contact our customer services if you want to obtain MobiusPi's SDK and its feature information. For more information about how to install and upgrade the SDK, see [IG902 Updated Software Versions](#).

1.2.3 Python requirements

The MobiusPi Python SDK is applicable to Python 3.7 and 3.8. If you use other Python versions, code execution may be abnormal. You can access the command prompt or start python IDE and run the `python` command to view your Python version. The content of this document is based on Python 3.7 and 3.8.



```
命令提示符 - python
Microsoft Windows [版本 10.0.18363.836]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\zn>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

1.2.4 1. Basic

Getting started

The following example shows how to restart MobiusPi.

```
# Import the Basic class
from mobiuspi_lib.basic import Basic

# Create a basic instance
basic = Basic()

# Restart MobiusPi
print("will reboot ...")
r = basic.reboot()
print("reboot : %s" % r)
```

Instructions for use

You can use the Basic class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create a `basic` instance
- Use `reboot()` to restart MobiusPi

Method description

`reboot()`

Description

Use this method to restart MobiusPi.

Request parameters

None

Returns

- Return type
str
- Return value

```
ok
```

- Return value description
 - `ok`: MobiusPi is restarted successfully.
 - `failed`: MobiusPi cannot be restarted.

1.2.5 2. Cellular

Getting started

The following example shows how to retrieve the MobiusPi cellular information:

```
# Import the Cellular class
from mobiuspi_lib.cellular import Cellular

# Create a cellular instance
cellular = Cellular()

# Retrieve modem information
modem = cellular.get_modem()
print("get modem: %s" % modem)
```

Output result of the example:

```
get modem: {
  "active_sim": "SIM 1",
  "imei_code": "811622048741556",
  "imsi_code": "411220441893359",
  "iccid_code": "84463317227780999882",
  "phone_number": "+8611162203133",
  "signal_level": 0,
  "dbm": 113,
  "rerp": 0,
  "rerq": 0,
  "register_status": 0,
  "operator": "CHN-CT",
  "apns": "",
  "network_type": "4G",
  "lac": "BB00",
```

(continues on next page)

(continued from previous page)

```
"cell_id": "DD788B81"
}
```

Instructions for use

You can use the Cellular class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create a `cellular` instance
- Use `get_modem()` to retrieve the modem status
- Use `get_network()` to retrieve network connection information

Method description

`get_modem()`

Description

Use this method to retrieve the modem status.

Request parameters

None

Returns

- Return type
dict
- Return value

```
{
  "active_sim": "SIM 1",
  "imei_code": "811622048741556",
  "imsi_code": "411220441893359",
  "iccid_code": "84463317227780999882",
  "phone_number": "+8611162203133",
  "signal_level": 0,
  "dbm": 113,
```

(continues on next page)

(continued from previous page)

```
"rerp": 0,  
"rerq": 0,  
"register_status": 0,  
"operator": "CHN-CT",  
"apns": "",  
"network_type": "4G",  
"lac": "BB00",  
"cell_id": "DD788B81"  
}
```

- Return value description
 - `active_sim(string)`: current SIM card.
 - `imei_code(string)`: international mobile equipment identity (IMEI). A `null string` is returned if the IMEI cannot be retrieved.
 - `imsi_code(string)`: international mobile subscriber identity (IMSI). A `null string` is returned if the IMSI cannot be retrieved.
 - `iccid_code(string)`: integrated circuit card identifier (ICCID). A `null string` is returned if the ICCID cannot be retrieved.
 - `phone_number(string)`: phone number. A `null string` is returned if the phone number cannot be retrieved.
 - `signal_level(int)`: signal value.
 - `dbm(int)`: value, in the unit of dBm.
 - `rerp(int)`: reference signal received power (RSRP), reserved.
 - `rerq(int)`: reference signal received quality (RSRQ), reserved.
 - `register_status(int)`: registration status.
 - * 0: Network registration is in progress.
 - * 1: Network registration is successful.
 - * 5: Network registration is successful, and the user is in the roaming state.
 - * 6: Network registration is not completed.
 - * 7: Unregistered.
 - `operator(string)`: Operator. A `null string` is returned if the operator name cannot be retrieved.
 - `apns(string)`: access point name (APN), reserved.

- `network_type(string)`: network type. A `null string` is returned if the network type cannot be retrieved.
- `lac(string)`: location area code (LAC). A `null string` is returned if the LAC cannot be retrieved.
- `cell_id(string)`: cell ID. A `null string` is returned if the cell ID cannot be retrieved.

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

`get_network()`

Description

Use this method to retrieve the network connection information.

Request parameters

None

Returns

- Return type
list
- Return value

```
[
{
  'status': 0,
  'ip_addr': '0.0.0.0',
  'netmask': '0.0.0.0',
  'gateway': '0.0.0.0',
  'dns': '0.0.0.0',
  'mtu': 1200,
  'connect_time': 0
}]
```

- Return value description

- `status(int)`: network status
 - * 0: network not connected
 - * 1: network connected
- `ip_addr(string)`: IP address
- `netmask(string)`: subnet mask
- `gateway(string)`: gateway
- `dns(string)`: domain name server (DNS)
- `mtu(int)`: maximum transmission unit (MTU)
- `connect_time(int)`: connection time, in the unit of seconds

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

1.2.6 3. Config

Getting started

The following example shows how to retrieve the app path and its configuration file path:

```
# Import the Config class
from mobiuspi_lib.config import Config

# Create a config instance, which corresponds to the app name HelloWorld. If no app_
↳ exists in the /var/user/app/ path when the config class is instantiated, the error_
↳ message "FileExistsError: Invalid app_name, do not find app HelloWorld" is returned.
config = Config(app_name="HelloWorld")

# Retrieve the app path
get_app_path = config.get_app_path()
print("get_app_path: %s" % get_app_path)

# Retrieve the path of the app configuration folder
get_app_cfg_path = config.get_app_cfg_path()
print("get_app_cfg_path: %s" % get_app_cfg_path)
```

Output result of the example:

```
get_app_path: /var/user/app/HelloWorld
get_app_cfg_path: /var/user/cfg/HelloWorld
```

Instructions for use

You can use the Config class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create a config instance
- Use `get_app_path()` to retrieve the app path
- Use `get_app_cfg_path()` to retrieve the path of the app configuration folder

Method description

`get_app_path()`

Description

Use this method to retrieve the app path.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/app/HelloWorld
```

`get_app_log_path()`

Description

Use this method to retrieve the app log file path.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/log/HelloWorld
```

get_app_cfg_path()**Description**

Use this method to retrieve the path of the app configuration folder.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/cfg/HelloWorld
```

get_app_cfg_file()**Description**

Use this method to retrieve the path of the current app configuration file.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/app/HelloWorld/config.yaml
```

get_default_app_cfg_file()

Description

Use this method to retrieve the path of the default app configuration file.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/app/HelloWorld/config.yaml
```

get_app_db_base_path()

Description

Use this method to retrieve the home path of the database.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/data/dbhome
```

get_app_db_path()**Description**

Use this method to retrieve the app database path.

Request parameters

None

Returns

- Return type
str
- Return value

```
/var/user/data/dbhome/HelloWorld
```

1.2.7 4. GPS**Getting started**

The following example shows how to retrieve the GPS information:

```
# Import the GPS class
from mobiuspi_lib.gps import GPS

# Create a gps instance
gps = GPS()

# Retrieve GPS information
position_status = gps.get_position_status()
print("position_status: ")
print(position_status)
```

Output result of the example:

```
position_status: {
  'gps_enable': 1,
  'gps_time': '2020-06-10 09:31:25',
  'latitude': "30° 35.276870' N",
  'longitude': "104° 3.251330' E",
  'speed': '0.3500 Knots (1knot = 1.852km/h)'
}
```

Instructions for use

You can use the GPS class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create a `gps` instance
- Use `get_position_status()` to retrieve GPS information

Method description

`get_position_status()`

Description

Use this method to retrieve MobiusPi's GPS information.

Request parameters

None

Returns

- Return type

dict

- Return value

```
{
  'gps_enable': 1,
  'gps_time': '2020-06-10 09:31:25',
  'latitude': "30° 35.276870' N",
  'longitude': "104° 3.251330' E",
  'speed': '0.3500 Knots (1knot = 1.852km/h)'
}
```

- Return value description

- `gps_enable`: specifies whether GPS is enabled.
 - * 0: GPS is not enabled.
 - * 1: GPS is enabled.
- `gps_time`: positioning time.
- `latitude`: latitude. A null string is returned if the latitude cannot be retrieved.
- `longitude`: longitude. A null string is returned if the longitude cannot be retrieved.
- `speed`: speed.

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

1.2.8 5. I/O

Getting started

The following example shows how to retrieve the I/O name, specify the input I/O mode, read the I/O status, and modify the output I/O:

```
# Import the I/O class and variables used by I/O-related methods
from mobiuspi_lib.io import IO, DIGITAL_DRY_CONTACT_MODE, DIGITAL_WET_CONTACT_MODE, DRY_
↪CONTACT_HIGH_VALUE, SHUT_DOWN_MODE, DRY_CONTACT_LOW_VALUE, ANALOG_LOW_A_MODE, ANALOG_
↪HIGH_A_MODE, ANALOG_LOW_V_MODE, and ANALOG_HIGH_V_MODE

# Create an io instance
io = IO()

# Retrieve all I/O names
io_list = io.get_io_list()
print("io_list: %s " % io_list)

# Set the digital input I/O mode
sdi = io.setup_digital_io(io_name="di0", mode=DIGITAL_DRY_CONTACT_MODE)
print("sdi: %s" % sdi)

# Read the I/O status
ri0 = io.read_io(io_name="di0")
print("ri0: %s" % ri0)

# Modify the output I/O
io.write_io(io_name="do0", value=DRY_CONTACT_HIGH_VALUE)
ro0 = io.read_io(io_name="do0")
print("ro0: %s" % ro0)
```

Output result of the example:

```
io_list: ['di0', 'di1', 'di2', 'di3', 'do0', 'do1', 'ai0', 'ai1']
sdi: {'index': 0, 'name': 'di0', 'type': 'digital input', 'mode': 'drycontact'}
ri0: LOW
ro0: ON
```

Note: Only devices with model IG902 and IO support AI support this API. For other models, please refer to the [I/O Module](#) for information on obtaining IO status.

Instructions for use

You can use the IO class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create an io instance
- Use `get_io_list()` to retrieve all I/O names

- Use `setup_digital_io()` to set the digital input I/O mode
- Use `setup_analog_io()` to set the analog input I/O mode
- Use `read_io(io_name="")` to retrieve the I/O status
- Use `write_io()` to modify the digital output I/O status

Method description

`get_io_list()`

Description

Use this method to retrieve all I/O names.

Request parameters

None

Returns

- Return type
list
- Return value

```
['di0', 'di1', 'di2', 'di3', 'do0', 'do1', 'ai0', 'ai1']
```

- Return value description di0 to di3 specify digital inputs DIO to DI3. do0 and do1 specify digital outputs DO0 and DO1. ai0 and ai1 specify analog inputs AI0 and AI1.

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

`get_io_info(io_name)`

Description

Use this method to retrieve the type and mode of a specific I/O.

Request parameters

- `io_name`: I/O name

Returns

- Return type
dict
- Return value

```
{
  'index': 0,
  'name': 'di0',
  'type': 'digital input',
  'mode': 'drycontact'
}
```

- Return value description
 - `index`: index
 - `name`: I/O name
 - `type`: I/O type
 - * `digital input`: digital input
 - * `digital output`: digital output
 - * `analog input`: analog input
 - `mode`: I/O mode
 - * Digital input I/O
 - `wetcontact`: wet contact
 - `drycontact`: dry contact
 - `shutdown`: shutdown
 - * Digital output I/O
 - `connect`: connected
 - `break`: disconnected
 - * Analog input I/O
 - `0_20mA`: 0 mA to 20 mA

- 4_20mA: 4 mA to 20 mA
- 0_5V: 0 V to 5 V
- 0_10V: 0 V to 10 V
- shutdown: shutdown

Exception

- When `io_name` is set to an incorrect I/O name, such as `dd1`, the following error message is returned:

```
KeyError: 'Invalid io_name'
```

- When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

get_all_io_info()

Description

Use this method to retrieve the types and modes of all I/Os.

Request parameters

None

Returns

- Return type
list
- Return value

```
[{  
    'index': 0,  
    'name': 'di0',  
    'type': 'digital input',  
    'mode': 'drycontact'  
}, {  
    'index': 1,
```

(continues on next page)

(continued from previous page)

```
'name': 'di1',
'type': 'digital input',
'mode': 'wetcontact'
}, {
  'index': 2,
  'name': 'di2',
  'type': 'digital input',
  'mode': 'shutdown'
}, {
  'index': 3,
  'name': 'di3',
  'type': 'digital input',
  'mode': 'drycontact'
}, {
  'index': 0,
  'name': 'do0',
  'type': 'digital output',
  'mode': 'connect'
}, {
  'index': 1,
  'name': 'do1',
  'type': 'digital output',
  'mode': 'break'
}, {
  'index': 0,
  'name': 'ai0',
  'type': 'analog input',
  'mode': '4_20mA'
}, {
  'index': 1,
  'name': 'ai1',
  'type': 'analog input',
  'mode': '0_5V'
}]
```

- Return value description
 - index: index
 - name: I/O name
 - type: I/O type

- * `digital input`: digital input
 - * `digital output`: digital output
 - * `analog input`: analog input
- `mode`: I/O mode
- * Digital input I/O
 - `wetcontact`: wet contact
 - `drycontact`: dry contact
 - `shutdown`: shutdown
 - * Digital output I/O
 - `connect`: connected
 - `break`: disconnected
 - * Analog input I/O
 - `0_20mA`: 0 mA to 20 mA
 - `4_20mA`: 4 mA to 20 mA
 - `0_5V`: 0 V to 5 V
 - `0_10V`: 0 V to 10 V
 - `shutdown`: shutdown

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

`setup_digital_io(io_name, mode)`

Description

Use this method to set the digital input I/O mode.

Request parameters

- `io_name`: I/O name (only digital input I/O is supported)
- `mode`: digital input I/O mode

- DIGITAL_DRY_CONTACT_MODE: dry contact mode
- DIGITAL_WET_CONTACT_MODE: wet contact mode
- SHUT_DOWN_MODE: shutdown

Returns

- Return type

dict

- Return value

```
{
  'index': 0,
  'name': 'di0',
  'type': 'digital input',
  'mode': 'drycontact'
}
```

- Return value description

- index: index
- name: I/O name
- type: I/O type
 - * digital input: digital input
- mode: I/O mode
 - * wetcontact: wet contact
 - * drycontact: dry contact
 - * shutdown: shutdown

Exception

- When io_name is set to an incorrect I/O name, such as dd1, the following error message is returned:

```
KeyError: 'Invalid io_name'
```

- When the name of a digital output I/O or analog input I/O is entered, such as do0, the following error message is returned:

```
KeyError: 'Parameter Conflict'
```

- When an incorrect mode is entered, such as 1234, the following error message is returned:

```
KeyError: 'Invalid mode'
```

- When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

- When MobiusPi is busy, the following error message is returned:

```
KeyError: 'Device Busy'
```

setup_analog_io(io_name, mode)

Description

Use this method to set the analog input I/O mode.

Request parameters

- `io_name`: I/O name (only analog input I/O is supported)
- `mode`: analog input I/O mode
 - `ANALOG_LOW_A_MODE`: 0-20 mA mode
 - `ANALOG_HIGH_A_MODE`: 4-20 mA mode
 - `ANALOG_LOW_V_MODE`: 0-5 V mode
 - `ANALOG_HIGH_V_MODE`: 0-10 V mode
 - `SHUT_DOWN_MODE`: shutdown

Returns

- Return type
dict
- Return value

```
{
  'index': 0,
  'name': 'ai0',
  'type': 'analog input',
  'mode': '4_20mA'
}
```

- Return value description
 - `index`: index
 - `name`: I/O name
 - `type`: I/O type
 - * `analog input`: analog input
 - `mode`: I/O mode
 - * `0_20mA`: 0 mA to 20 mA
 - * `4_20mA`: 4 mA to 20 mA
 - * `0_5V`: 0 V to 5 V
 - * `0_10V`: 0 V to 10 V
 - * `shutdown`: shutdown

Exception

- When `io_name` is set to an incorrect I/O name, such as `dd1`, the following error message is returned:

```
KeyError: 'Invalid io_name'
```

- When the name of a digital input or output I/O is entered, such as `do0`, the following error message is returned:

```
KeyError: 'Parameter Conflict'
```

- When an incorrect mode is entered, such as `1234`, the following error message is returned:

```
KeyError: 'Invalid mode'
```

- When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

- When `MobiusPi` is busy, the following error message is returned:

```
KeyError: 'Device Busy'
```

read_io(io_name)

Description

Use this method to read the I/O status.

Request parameters

- `io_name`: I/O name

Returns

- Return type
str
- Return value

```
LOW
```

- Return value description
 - ON
 - * ON is returned when the digital input I/O mode is wet contact and the input voltage ranges from 10 V to 30 V.
 - * ON is returned when the digital output I/O is connected.
 - OFF
 - * OFF is returned when the digital input I/O mode is wet contact and the input voltage ranges from 0 V to 3 V.
 - * OFF is returned when the digital output I/O is disconnected.
 - LOW: LOW is returned when the digital input I/O mode is dry contact and disconnected.
 - HIGH: HIGH is returned when the digital input I/O mode is dry contact and connected.
 - Current or voltage of analog input

Exception

- When `io_name` is set to an incorrect I/O name, such as `dd1`, the following error message is returned:

```
KeyError: 'Invalid io_name'
```

- When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

`write_io(io_name)`

Description

Use this method to modify the digital output I/O status.

Request parameters

- `io_name`: name of a digital output I/O.
- `value`: value of the digital output I/O.
 - `DRY_CONTACT_LOW_VALUE`: Set the digital output I/O to the disconnected state.
 - `DRY_CONTACT_HIGH_VALUE`: Set the digital output I/O to the connected state.

Returns

- Return type
str
- Return value

```
TRUE
```

- Return value description
 - `TRUE`: Settings are successfully delivered

Exception

- When `io_name` is set to an incorrect I/O name, such as `dd1`, the following error message is returned:

```
KeyError: 'Invalid io_name'
```

- When the name of a digital input I/O or analog input I/O is entered, such as do0, the following error message is returned:

```
KeyError: 'Invalid Parameter'
```

- When an incorrect value is entered, such as 1234, the following error message is returned:

```
KeyError: 'Invalid value'
```

- When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

- When MobiusPi is busy, the following error message is returned:

```
KeyError: 'Device Busy'
```

1.2.9 6. Serial

Getting started

The following example shows how to retrieve the 232 or 485 serial port path:

```
# Import the Serial class
from mobiuspi_lib.serial import Serial

# Create a Serial instance
serial = Serial()

# Retrieve the 232 serial port path
path_232 = serial.get_serial232_path()
print("232 path: %s" % path_232)

# Retrieve the 485 serial port path
path_485 = serial.get_serial485_path()
print("485 path: %s" % path_485)
```

Output result of the example:

```
232 path: /dev/tty01
485 path: /dev/tty03
```

Instructions for use

You can instantiate the `Serial` class or its subclass. Procedure:

- Create an `serial` instance
- Use `get_serial232_path()` to retrieve the 232 serial port path.
- Use `get_serial485_path()` to retrieve the 485 serial port path.

Method description

`get_serial232_path()`

Description

Use this method to retrieve the 232 serial port path.

Request parameters

None

Returns

- Return type
str
- Return value

```
/dev/tty01
```

- Return value description
 - `/dev/tty05`: This value is returned when IG501 is used.
 - `/dev/tty01`: This value is returned when IG902 is used.

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

get_serial485_path()

Description

Use this method to retrieve the 485 serial port path.

Request parameters

None

Returns

- Return type
str
- Return value

```
/dev/tty03
```

- Return structure
 - /dev/tty01: This value is returned when IG501 is used.
 - /dev/tty03: This value is returned when IG902 is used.

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```

1.2.10 7. SystemInfo

Getting started

The following example shows how to retrieve the MobiusPi system information:

```
# Import the SystemInfo class
from mobiuspi_lib.systeminfo import SystemInfo

# Create a sysinfo instance
sysinfo = SystemInfo()

# Retrieve the MobiusPi system information
get_system_info = sysinfo.get_system_info()
print("get system info: %s" % get_system_info)
```

Output result of the example:

```
get system info: {
  'language': 'Chinese',
  'hostname': 'InGateway',
  'model_name': 'IG902H',
  'oem_name': 'inhand',
  'serial_number': 'GT902XXXXXXXXXX',
  'mac_addr1': '00:XX:XX:XX:XX:XX',
  'mac_addr2': '00:XX:XX:XX:XX:XX',
  'firmware_version': '2.0.0.r12644',
  'bootloader_version': '2017.01.r10517',
  'product_number': 'TH09-W-RE',
  'description': 'www.inhand.com.cn',
  'auto_save': 1,
  'encrypt_passwd': 1
}
```

Instructions for use

You can use the SystemInfo class as an instance, within a class or by subclassing. The general usage flow is as follows:

- Create an `sysinfo` instance
- Use `get_system_info()` to retrieve the MobiusPi system information

Method description

`get_system_info()`

Description

Use this method to retrieve the MobiusPi system information.

Request parameters

None

Returns

- Return type
dict
- Return value

```
{
  'language': 'Chinese',
  'hostname': 'InGateway',
  'model_name': 'IG902H',
  'oem_name': 'inhand',
  'serial_number': 'GT902XXXXXXXXXX',
  'mac_addr1': '00:XX:XX:XX:XX:XX',
  'mac_addr2': '00:XX:XX:XX:XX:XX',
  'firmware_version': '2.0.0.r12644',
  'bootloader_version': '2017.01.r10517',
  'product_number': 'TH09-W-RE',
  'description': 'www.inhand.com.cn',
  'auto_save': 1,
  'encrypt_passwd': 1
}
```

- Return value description
 - language: language
 - * Chinese: Chinese
 - * English: English
 - hostname: MobiusPi name
 - model_name: MobiusPi model
 - oem_name: OEM name
 - serial_number: MobiusPi serial number

- `mac_addr1`: MAC address 1 of MobiusPi
- `mac_addr2`: MAC address 2 of MobiusPi
- `firmware_version`: firmware version
- `bootloader_version`: bootloader version
- `product_number`: product number
- `description`: product description
- `auto_save`: whether to automatically save modified configurations
 - * 0: not automatically save modified configurations
 - * 1: automatically save modified configurations
- `encrypt_passwd`: whether to encrypt plaintext passwords
 - * 0: not encrypt plaintext passwords
 - * 1: encrypt plaintext passwords

Exception

When the request times out, the following error message is returned:

```
KeyError: 'Connection Timeout'
```