W60X MicroPython User Guide

V0.3

Beijing Winner Microelectronics Co., Ltd. (Winner Micro)

Address: 18th Floor, Yindu building, No.67 Fucheng Road, Haidian District, Beijing, China

Tel: +86-10-62161900

Website: www.winnermicro.com

Document History

Version	Completion Date	Revision Record	Author	Auditor
V0.1	2018-11-13	Initial release	Ray	LiLM
V0.2	2019-09-12	Add application demos	Ray	LiLM
V0.3	2019-11-08	Add samples of ADC & SSL	Ray	LiLM

Content

1 Introduction

1.1 Purpose

This document introduces how to use W60X under MicroPython.

1.2 Readers

The developers with W60X.

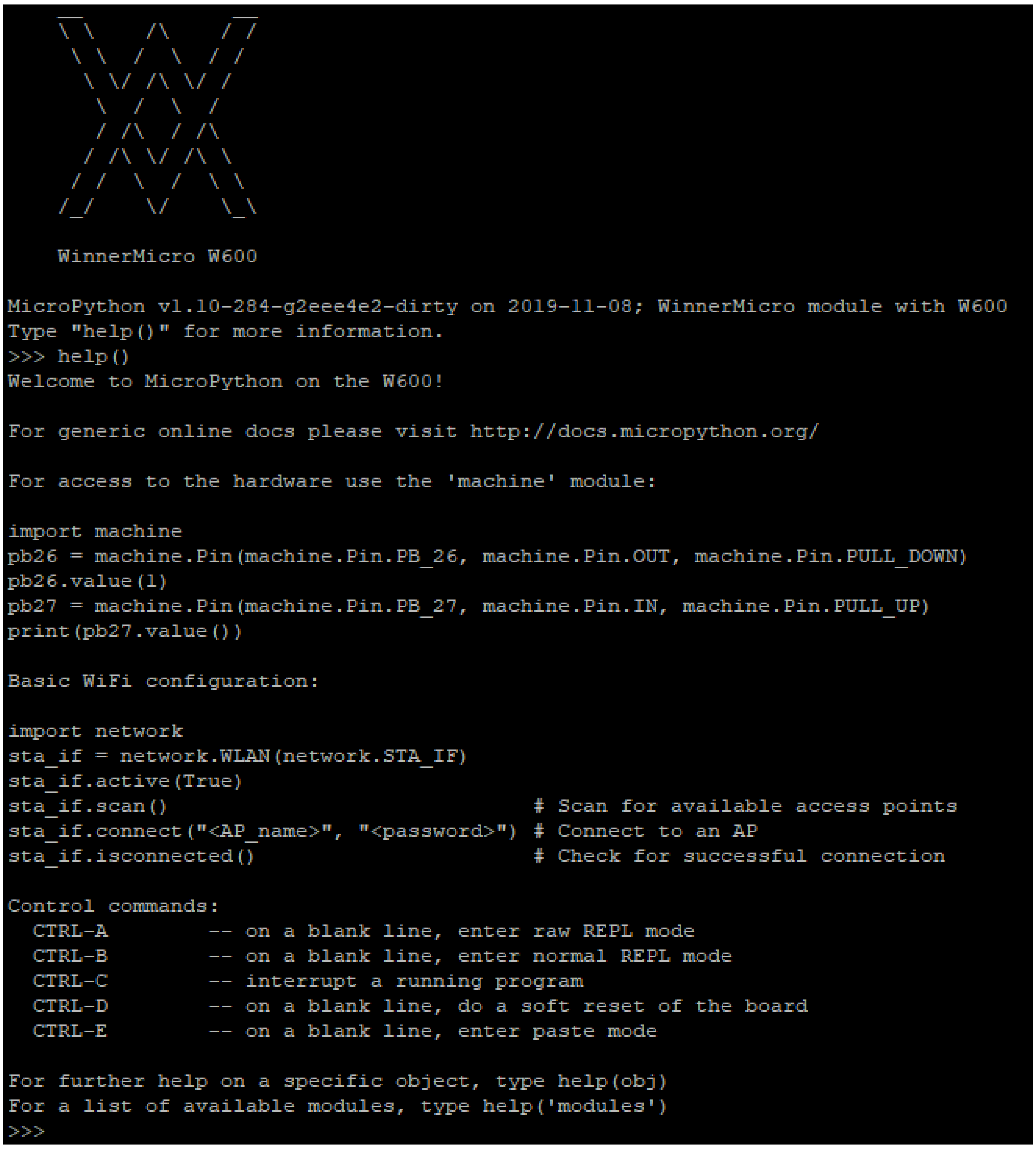
2 Introduction of MicroPython

MicroPython is a streamlined and effective realization with Python 3 language. It includes a small part of Python standard library and it has optimized the micro processer and limited environment.

Features of MicroPython on W60X:

- Support REPL (Python prompt) over UART0
- Support 16KB task stack and 40KB heap for running MicroPython
- Support most features of MicroPython and inter library (Unicode, high precision integer, single precision floating point number, complex number)
- Support hardware interface such as GPIO、UART, SPI, I2C, PWM, WDT, ADC, RTC, Timer.
- Support Wi-Fi network feature (include OneShot Config)
- Support SSL using hardware encryption and decryption (only supported by 2M Flash devices)
- Support internal flash filesystem (32KB available)
- Support built-in FTP server transfer script files

The startup interface is in the following figure:



```
    __             __
    \ \    /\    / /
     \ \  /  \  / /
      \ \/ /\ \/ /
       \  /  \  /
       / /\  / /\
      / /\ \/ /\ \
     / /  \  /  \ \
    /_/    \/    \_\

    WinnerMicro W600

MicroPython v1.10-284-g2eee4e2-dirty on 2019-11-08; WinnerMicro module with W600
Type "help()" for more information.
>>> help()
Welcome to MicroPython on the W600!

For generic online docs please visit http://docs.micropython.org/

For access to the hardware use the 'machine' module:

import machine
pb26 = machine.Pin(machine.Pin.PB_26, machine.Pin.OUT, machine.Pin.PULL_DOWN)
pb26.value(1)
pb27 = machine.Pin(machine.Pin.PB_27, machine.Pin.IN, machine.Pin.PULL_UP)
print(pb27.value())

Basic WiFi configuration:

import network
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.scan()                             # Scan for available access points
sta_if.connect("<AP_name>", "<password>") # Connect to an AP
sta_if.isconnected()                      # Check for successful connection

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL-D        -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
>>>
```

The usage method supported by MicroPython can be found in the docs directory.

Function modules can be downloaded from https://github.com/micropython/micropython-lib.

Users can download it and update to W60X chipset, then import it into the script.

3 Quick Start MicroPython

W60X has been transplanted successfully with MicroPython 1.10. It supports the source code package for users to rebuilt and also supports compiled firmware for update directly.

3.1 Firmware Download

The compiled firmware has been added on the website: http://www.winnermicro.com/en. Users can just download the firmware and update it to W60X.

3.2 Rebuild MicroPython

This operation is based on GCC building and can be used in shell directly. With Windows system, users should install Cygwin. The W60X_IDE includes Cygwin environment and is suggested to be used. Users can download the W60X_IDE from WinnerMicro's website.

3.2.1 Download Cross Compiling Tools

The GCC cross compiling tools used by W60X is arm-none-eabi-gcc, and the download address is: https://launchpad.net/gcc-arm-embedded/4.9/4.9-2014-q4-major.

After decompression, the tools's path should be put into the environment variables. For example, put it into the directory /opt:

```
export PATH=$PATH:/opt/tools/arm-none-eabi-gcc/bin
```

This configuration can be write to ".bashrc" to make the change permanent.

Note: The WM_SDK only supports the GCC version of V4.X, so users should deal with the compiling errors by themselves when using other version compiler.

3.2.2 Download WM_SDK Package

The SDK package can be downloaded from http://www.winnermicro.com .

Note: MicroPython is supported from version G3.01.00 with W60X SDK.

After decompressed, the environment variable "*WMSDK_PATH*" should be setted with the directory os WM_SDK. For example：

```
export WMSDK_PATH=/home/w60x/WM_SDK
```

This configuration can be write to ".bashrc" to make the change permanent.

There are some components in WM_SDK which is useless for MicroPython. These components can be cut off before compiling to reduce this code size. Users can open the WM_SDK/Include/wm_config.h and modify the macro of such useless component from "CFG_ON" to "CFG_OFF".
The suggested useless components are:

```
#define TLS_CONFIG_HOSTIF          CFG_OFF
#define TLS_CONFIG_RMMS            CFG_OFF
#define TLS_CONFIG_HTTP_CLIENT     CFG_OFF
#define TLS_CONFIG_NTP             CFG_OFF
```

Note: During building, if it prompts the code size exceeding the ROM limit, the cutting off should be operated.

3.2.3 Download MicroPython

Please download the source code package from http://www.winnermicro.com/en and decompress it.

3.2.4 Compiling

Entry the folder of ports/w60x of MicroPython project with shell command line. If using 2M Flash W600, users can modify "MICROPY_USE_2M_FLASH" in Makefile file, and operate the compiling command:

```
make V=s
```

After the compiling completed, the firmware will be generated under the folder of ports/w60x/build.

Note: If W600 is 2M Flash type, users should use 2M Flash firmware.

3.3 Burning MicroPython

If it is the first time to update MicroPython firmware into W60X, the *.fls file should be used. Following is the commands:

```
make flash V=s
```

After burning the firmware, you can compile the *_gz.img file for burning, and the programming speed is faster. The command is as follows:

```
make image V=s
```

The UART port number should be set before burning firmware. Users can base on the tips of shell or open the w60x/tools/download_*.sh to modify the “SERIAL_NAME” to the actual UART port number, then start to deal with updating commands.

4 Examples for Command Lines

MicroPython supports interactive command line named REPL, the commands can be entered into the command line.

4.1 Basic Command of Printing

```
print('hello world')
print(b'bytes 1234\x01')
print(123456789)
for i in range(4):
    print(i)
```

4.2 How to Use Wi-Fi

```
import network

sta_if = network.WLAN(network.STA_IF)
```

```
sta_if.active(True)
sta_if.scan()
sta_if.connect("WM2G","87654321")
sta_if.isconnected()
```

4.3 How to Use OneShot Config

OneShot Config need the SmartPhone's APP.
OneShot requires mobile app support, which can be installed on the official website or app store. You can also use the WeChat AirKiss to follow the public number or install the AirKiss test app.

```
import network

sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.oneshot(1)

#Start OneShot APP

sta_if.isconnected()
```

4.4 How to Use Socket

```
import socket

s = socket.socket()
addr = ('www.qq.com', 80)
s.connect(addr)

s.send("hello world!")
s.recv(64)
```

```
s.close()
```

Note: Socket operation should start after Wi-Fi connected to router.

4.5 How to Use SSL

```
import socket
import ussl

s = socket.socket()
addr = ('www.baidu.com', 443)
s.connect(addr)

sec = ussl.wrap_socket(s)
sec.write("GET / HTTP/1.1\r\n\r\n")
sec.read(1024)

sec.close()
s.close()
```

Note: SSL operation should start after Wi-Fi connected to router.

4.6 How to Use PIN

```
from machine import Pin

led = Pin(Pin.PB_16, Pin.OUT, Pin.PULL_FLOATING)
led.value(1)
led.value(0)
```

4.7 How to Use I^2C

MicroPython not only supports hardware I^{2}C interface, but also supports software emulated I^{2}C. When the product ID is -1, the software I^{2}C function can be used. When such ID is not -1, the hardware I^{2}C can be used. W60X supports I^{2}C function, following is the demo with temperature and humidity sensor SHT30.

```
from machine import Pin, I2C
import time

i2c = I2C(0, scl=Pin(Pin.PB_13), sda=Pin(Pin.PB_14), freq=100000)

buf = bytearray(2)
buf[0] = 0x30
buf[1] = 0xA2
i2c.writeto(0x44, buf)
time.sleep_ms(1000)

buf2 = bytearray(6)
buf[0] = 0x2c
buf[1] = 0x06
i2c.writeto(0x44, buf)
buf2 = i2c.readfrom(0x44, 6)

temp_raw = (buf2[0] << 8) + (buf2[1])
humi_raw = (buf2[3] << 8) + (buf2[4])
temp = 175 * temp_raw / 65535 - 45
humi = 100 * humi_raw / 65535
print("temp = {:.2f}, humi = {:.2f}".format(temp, humi))
```

Note: Following is the W600's PINs which support I^{2}C function:

I^{2}C Fucntion	IO PINs
scl	PA_06，PA_08，PB_11，PB_13，PB_21
sda	PA_07，PA_15，PB_12，PB_14，PB_22

4.8 How to Use RTC

```
from machine import RTC


rtc = RTC()
rtc.init((2019, 9, 12, 3, 13, 0, 0, 0))
print(rtc.now())
```

Note: the week day's value is from 0 to 6, 0 means Monday.

4.9 How to Use SPI

W60X supports hi-speed SPI (max 50MHz) and low-speed SPI (Max 20MHz) functions. W60X can be only used with slave device with hi-speed SPI, and can be used with master device with low-speed SPI.
In MicroPython, when the ID is -1, the software SPI function can be used, and when ID is 0, the hardware SPI can be used. Following is the demo for low-speed SPI:

```
from machine import Pin, SPI


spi = SPI(0, baudrate=200000, polarity=1, phase=0, sck=Pin(Pin.PB_16), mosi=Pin(Pin.PB_18), miso=Pin(Pin.PB_17), cs=Pin(Pin.PB_15))
spi.read(10)
spi.read(10, 0xff)


buf = bytearray(50)
spi.readinto(buf)
spi.readinto(buf, 0xff)


spi.write(b'12345')


buf2 = bytearray(4)
spi.write_readinto(b'1234', buf2)
spi.write_readinto(buf2, buf2)
```

Note: following is the IO which can be used for hardware SPI:

SPI Function	IO PINs
sck	PA_01，PA_11，PB_16，PB_27
mosi	PA_04，PA_09，PA_10，PB_02，PB_18
miso	PA_03，PA_05，PA_10，PB_01，PB_17
cs	PA_02，PA_12，PB_00，PB_07，PB_15

4.10 How to Use PWM

W60X supports 5 channel hardware PWM from 0 to 4, with the frequency from 1 to 156250, and the duty cycle is from 0 to 255.

```
from machine import Pin, PWM

pwm1 = PWM(Pin(Pin.PB_16), channel=2, freq=100, duty=0)
pwm1 = PWM(Pin(Pin.PB_16), channel=2, freq=100, duty=255)
pwm1.deinit()

pwm2 = PWM(Pin(Pin.PB_18))
pwm2.freq()
pwm2.freq(100)
pwm2.duty()
pwm2.duty(250)
```

Note: Following is the IO pins for PWM:

PWM Channel	IO PINs
Channel 0	PA_00，PA_05，PB_05，PB_18，PB_19，PB_30
Channel 1	PA_01，PA_07，PB_04，PB_13，PB_17，PB_20
Channel 2	PA_02，PA_08，PB_04，PB_03，PB_16，PB_21
Channel 3	PA_03，PA_09，PB_02，PB_06，PB_15，PB_22
Channel 4	PA_04，PA_10，PB_01，PB_08，PB_14，PB_23

4.11How to Use Timer

W60X supports 6 sets timers (timer0 has beed used by WM_SDK, so users can use timer1 to timer5). When the ID is -1, the software timer can be used. When the ID is 1 to 5, the hardware timer can be used.

```
from machine import Timer


timer1 = Timer(-1)
timer1.init(period=5000, mode=Timer.ONE_SHOT, callback=lambda t:print(1))


timer3 = Timer(3)
timer3.init(period=2000, mode=Timer.PERIODIC, callback=lambda t:print(2))
```

4.12 How to Use ADC

W60X supports 12 Channels of ADC, the Channel number is from 0~11.

```
from machine import ADC


adc = ADC(0)
vcc = adc.read()
print("vcc = {:.3f}".format((vcc - 8192.0) / 8192 * 2.25 / 1.2 + 1.584))
```

Note: Following is the IOs to ADC channel:

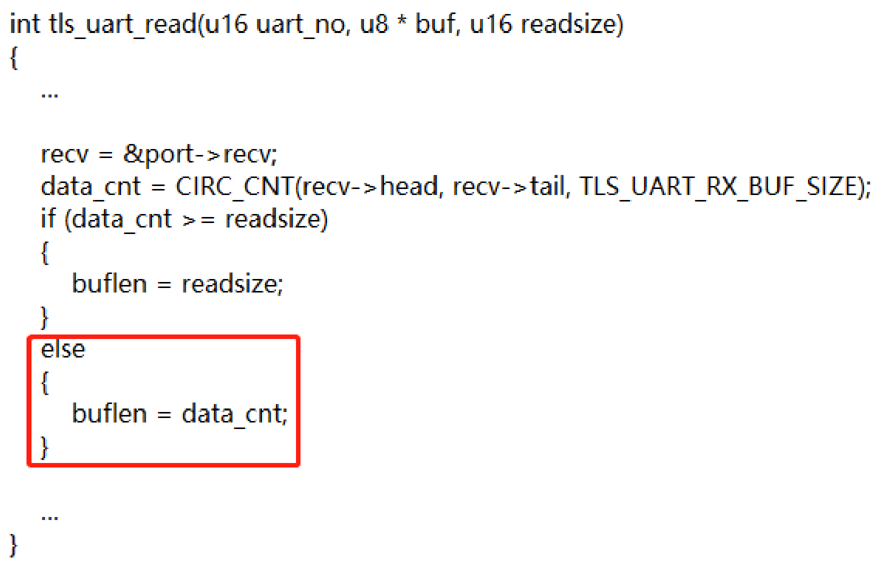
ADC Channel No.	IO
0	PB_19
1	PB_20
2	PB_21
3	PB_22
5	PB_23
6	PB_24

7	PB_25
8	PB_19 and PB_20
9	PB_2 and PB_22
10	PB_2 and PB_24
11	PB_2 and PB_26

4.13How to Use UART

```
from machine import UART

uart = UART(1, 115200)
uart.init(115200, bits=8, parity=None, stop=1)

uart.write('hello world')

uart.readline()
print(uart.read(5))

buf = bytearray(6)
uart.readinto(buf)
print(buf)
```

Note: When the number of reading bytes less than the number of actual received bytes, no data can be got. The reason is the WM_SDK has such limit, so users can modify the Platform\Drivers\uart\wm_uart.c file to avoid this limit. Following is how to modify the file:



```
int tls_uart_read(u16 uart_no, u8 * buf, u16 readsize)
{
    ...

    recv = &port->recv;
    data_cnt = CIRC_CNT(recv->head, recv->tail, TLS_UART_RX_BUF_SIZE);
    if (data_cnt >= readsize)
    {
        buflen = readsize;
    }
    else
    {
        buflen = data_cnt;
    }

    ...
}
```

4.14 How to Use WDT

```
from machine import WDT

wdt = WDT(0,5000000)
wdt.feed()
```

Note: After enabling WDT, it can run normally without feeding the dog in the MicroPython script, because WM_SDK has automatically added the dog feed function to the underlying lowest priority task.

5 How to Use Script File

5.1 Convert Script File to Bytecode and Compile to Firmware

MicroPython supports compiling script file to firmware directly. After W60X is power on, the firmware will execute the _boot.by in the path ports/w60x/scrips defaultly. Users can put the Python script code into the file and let it can be executed automatically after power on.

All the files which are in the path ports/w60x/scrips will be compiled into the firmware. The command “pyexec_frozen_module” can be used to execute specified script file. For example:

```
pyexec_frozen_module("_boot.py");
```

Other script file can be called in _boot.py, both methods are OK.

easyw600.py is a script that is programmed into the firmware in this way. It integrates some commonly used functions and can be used by the user. The module provides the following methods:

```
import easyw600
easyw600.scan()#Scanning Peripheral WiFi Network
easyw600.oneshot()#Start the Oneshot until the IP address is printed after networking
easyw600.connect(ssid="myssid", password=None) # connect to WiFi, print IP after
networking
easyw600.disconnect()#disconnect from the network
easyw600.createap(ssid="w60x_softap", password=None) #create a soft ap
easyw600.closeap()#close the soft ap
easyw600.ftpserver()#start the built-in FTP server, port number is 21, user name root,
password root
```

Compling script file into firmware will not occupy the area of file system, but will increase the size of image file. So users can choose the suitable method by the real situation.

5.2 Update Script File to W60X’s Flash

5.2.1 How to Update Script File

W60X’s internal Flash provides limited size of file system. Users can save the script file into the file system. For convenience, the W60X MicroPython has integrated the feature of FTP server. After the W60X has joined the network, users can copy script file to W60X by FTP client on PC.

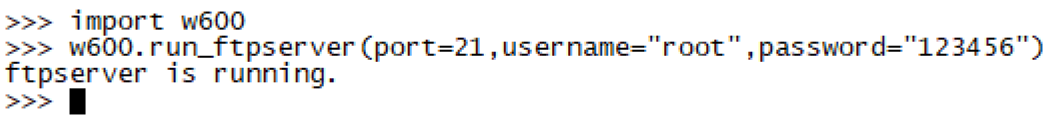
After W60X joined the network, following operation can be used to startup FTP server:

```
import w600
w600.run_ftpserver(port=21,username=None,password=None)
```

All the parameters have default values: default port number is 21; default user name and password are anonymous. User name and password should be string format with double quotation marks during setting.

When logging in with anonymous user name, users can only view and download files and can not upload, modify and delete files. If prompted for unauthorized operation, please set the user name and password and try again.

Following information will display when W60X FTP server starts up.

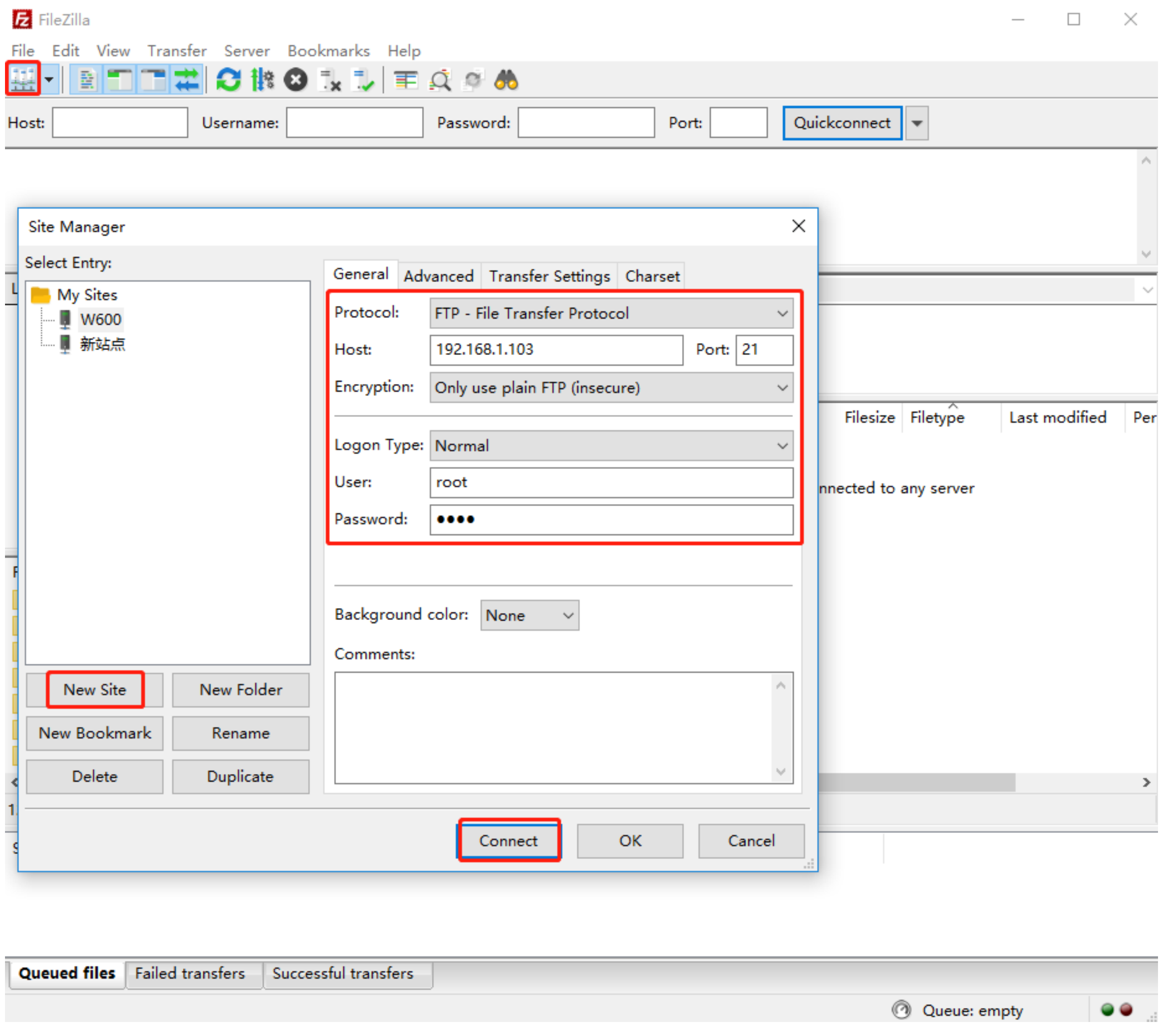


```
>>> import w600
>>> w600.run_ftpserver(port=21,username="root",password="123456")
ftpserver is running.
>>> █
```

Limited functions are supported by embedded FTP server and there are many kind of FTP clients, so there may be compatibility issues. If users can not operate, they can try to modify FTP connecting mode (active/passive).

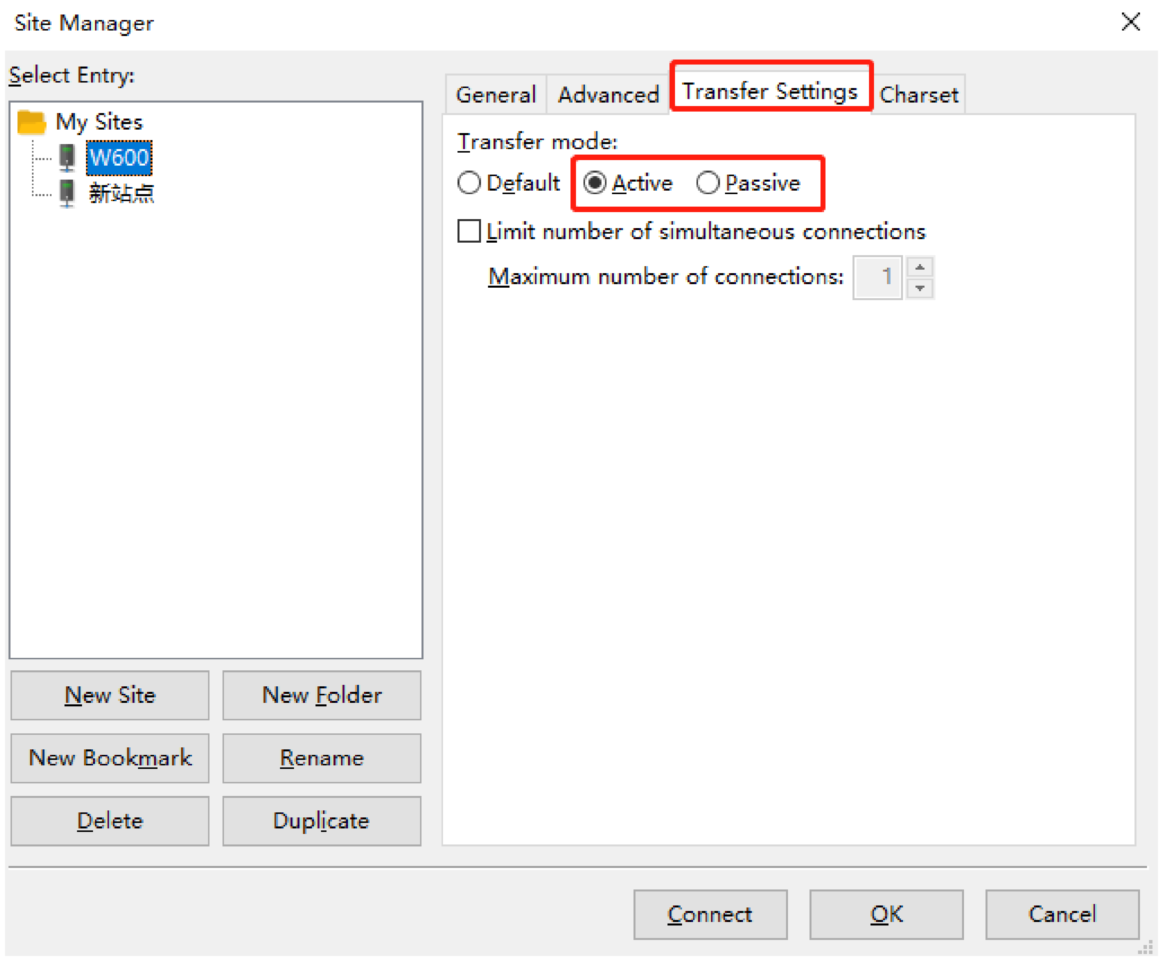
The firewall (iptables) should be disable in Linux system, otherwise, some host PCs will not connect to FTP.

Following is the example with FileZilla:



Protocol: FTP - File Transfer Protocol
Host: 192.168.1.103 Port: 21
Encryption: Only use plain FTP (insecure)
Logon Type: Normal
User: root
Password: ●●●●
Background color: None
Comments:

Following is the operation for setting active or passive mode:



Transfer mode:

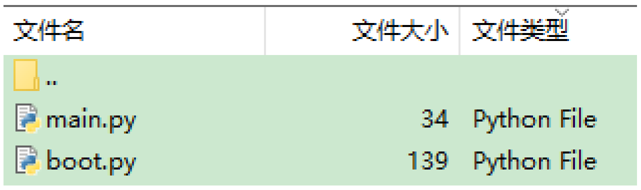
○ Default ● Active ○ Passive

☐ Limit number of simultaneous connections

Maximum number of connections: 1

5.2.2 Flash File Structure

There are following files in W60X's default file system:



文件名	文件大小	文件类型
..		
main.py	34	Python File
boot.py	139	Python File

These files are created defaultly by system, users can modify them directly.

When W60X powers on, the boot.py script will be executed firstly. And then the main.py will be executed. If some user code will be executed, such code can be writen into these two script files. The initial code can be writen into boot.py and function code can be writen into main.py. Users can add new script files according to their own requirement.

6 Version Specification

Current W60X's MicroPython version number is W60X_MicroPython_1.10_B1.5.

W600 provids 2 versions of firmwares for 1MByte or 2MByte integrated Flash. Only 2Mbyte Flash version firmware supports SSL function.

When using internal Flash Filesystem, FatFS or LittleFS can be used for formatting area. The LittleFS is default used.

When FatFS is used, a part of OTA area will be occupied for Filesystem. So the size of *_gz.img should be less than 352KByte with 1M Flash verison, and less than 736KByte with 2M Flash version.