SIMCom - OpenLinux and FOTA Quickstart [US-EN]

Table of Contents

1. Introduction

Firmware Over-The-Air (FOTA) is a technology present in SIMCom modules so that you can update firmware remotely without requiring physical access to the device.

In a real application, the module can be programmed to periodically connect to the FTP server and update the firmware itself in a fully automated way.

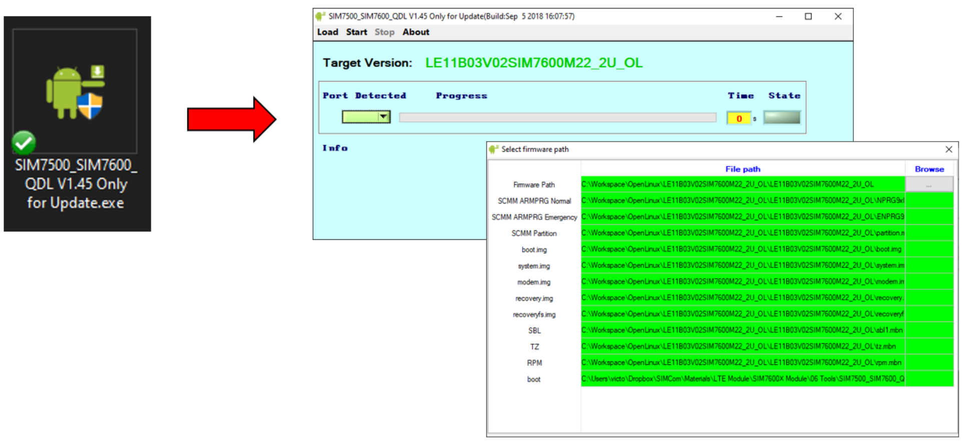
In this guide we will explain how to compile your SIM7600SA-H application with OpenLinux and update the firmware from an FTP server.

To start testing you will need:

- SIM7600SA-H Module
- SIM7600 QDL (Firmware Update Tool)
- SIM7600SA-H OpenLinux Firmware
- ADB (Android Debug Bridge) Installed
- SIMCom OpenLinux SDK
- Linux operating system installed (can be a virtual machine)
- Functional FTP Server

2. Updating Firmware SIM7600SA-H

Unzip the firmware and run the QDL Firmware Update Tool. In the QDL tool click Load, select the folder where firmware was unzipped and click Start. The tool should automatically detect the module and start the update. The module will restart a few times during the process and will be completed after a few minutes.


SIM7500_SIM7600_QDL V1.45 Only for Update.exe
SIM7500_SIM7600_QDL V1.45 Only for Update(Build:Sep 5 2018 16:07:57)
Load
Start
Stop
About
Port Detected
Progress
Time
State
0 s
Info
Select firmware path

Target Version: LE11B03V02SIM7600M22_2U_OL

	File path	Browse
Firmware Path	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL	...
SCMM ARMPRG Normal	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\NPRG9xl	
SCMM ARMPRG Emergency	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\ENPRG9	
SCMM Partition	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\partition.n	
boot.img	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\boot.img	
system.img	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\system.im	
modem.img	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\modem.im	
recovery.img	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\recovery.	
recoveryfs.img	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\recoveryf	
SBL	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\sbl1.mbn	
TZ	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\tz.mbn	
RPM	C:\Workspace\OpenLinux\LE11B03V02SIM7600M22_2U_OL\LE11B03V02SIM7600M22_2U_OL\rpm.mbn	
boot	C:\Users\victo\Dropbox\SIMCom\Materials\LTE Module\SIM7600X Module\06 Tools\SIM7500_SIM7600_Q	

3. Installing OpenLinux SDK

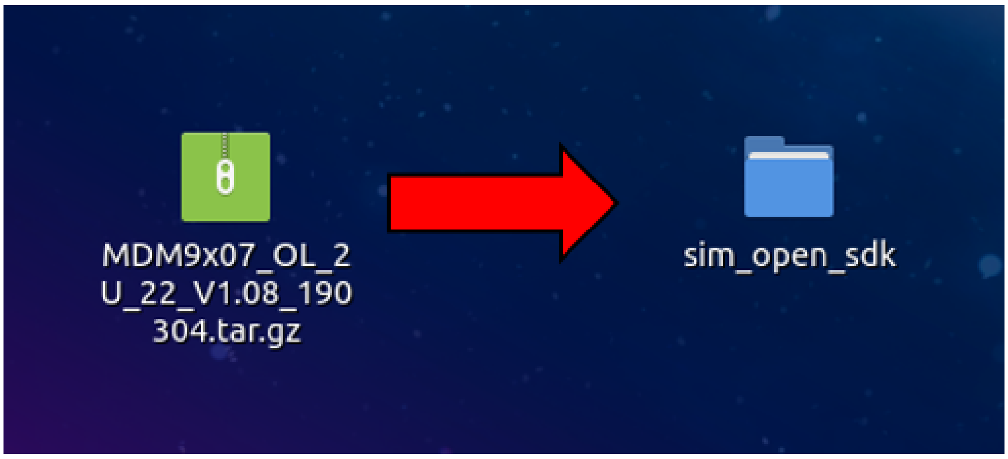
The SDK can be installed and run on a Windows computer, but we recommend using Linux for easier installation. If you prefer, you can use a virtual machine to develop and compile the SDK without any problems.

Install Python and Make with the following commands:

```
sudo apt-get install python
sudo apt-get install make
```

Send the compacted SDK to the Linux system and run the following command to unzip:

```
sudo tar xzf sim_open_sdk.tar.gz
```



The generated folder will be used in the next steps.

4. Compiling Applications

Each folder inside **sim_open_sdk** represents a different part of the OpenLinux system that will be used in the module.

The simcom-demo folder is one that needs to be highlighted, as it contains the demo application that will run on OpenLinux. You can use it as a base to develop your own application.

For this first test with FOTA we will compile all parts of OpenLinux without making any modifications.

First, open the terminal, go to the **sim_open_sdk** folder and set the system variables using the command:

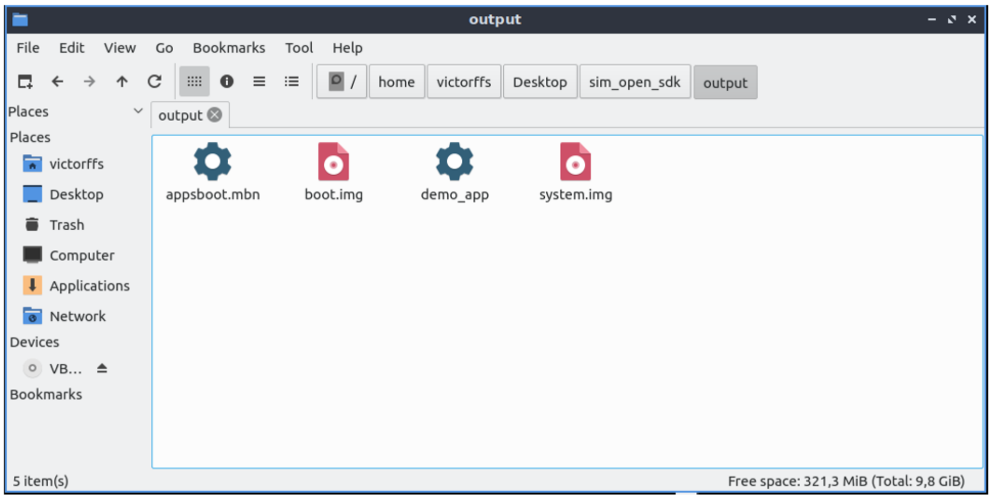
```
source sim_crosscompile/sim-crosscompile-env-init
```

Clean the environment with the **make clean** command and then individually run the **make** command followed by the part to compile as follows:

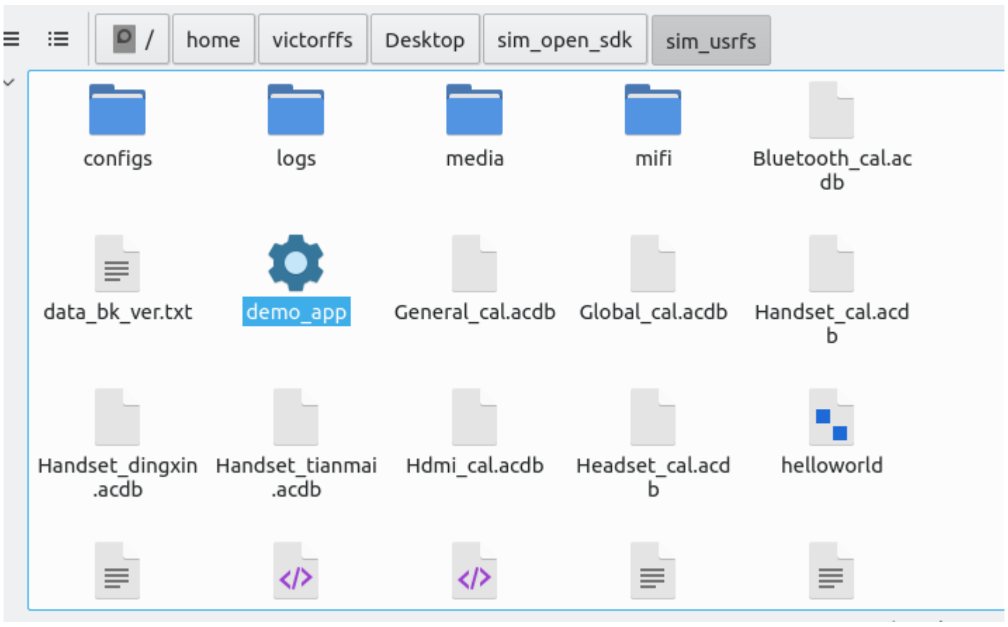
```
make clean

make aboot
make kernel
make rootfs
make demo
```

The 4 compiled files will appear in the output folder:



Move the file **demo_app** to the folder **/sim_open_sdk/sim_usrfs**

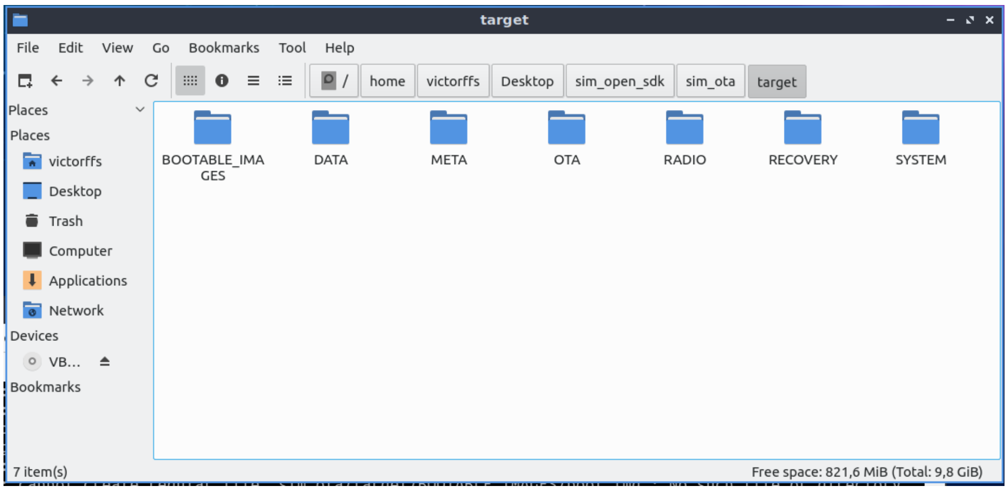


5. Packing FOTA

Run the **make ota** command to read the files that have been compiled and prepare them to be ready to be packaged.

```
make ota
```

The compiled files will be organized inside the folder **sim_ota/target**



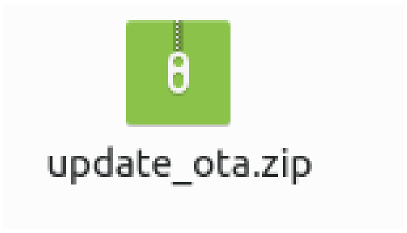
5.1 Full FOTA

Now that the files to be sent to the module have already been organized into the target folder, you need to package them to be read by the module later.

To do this go back to the **sim_open_sdk** folder and run the command:

```
bash ota_full.sh
```

A new file will be generated inside the **output** folder named **update_ota.zip**



This file contains the complete module firmware and could already be sent to the module to be installed.

However, when we look at its size, we will see that it is over 38MB, and this can be very much for a cellular connection, and may even end up with your device's data plan.

So how do we update the firmware without ending the data plan?

The answer is simple... Let's use Delta FOTA!

5.2 Delta FOTA

Delta FOTA consists of analyzing two firmware versions, identifying the differences between them, and sending only code snippets that have been modified to be updated in the module.

This can significantly reduce the size of the file that will be sent to the device.

First let's change the file name **update_ota.zip** to **full_ota.zip**

5.2.1 Creating Firmware Version 2

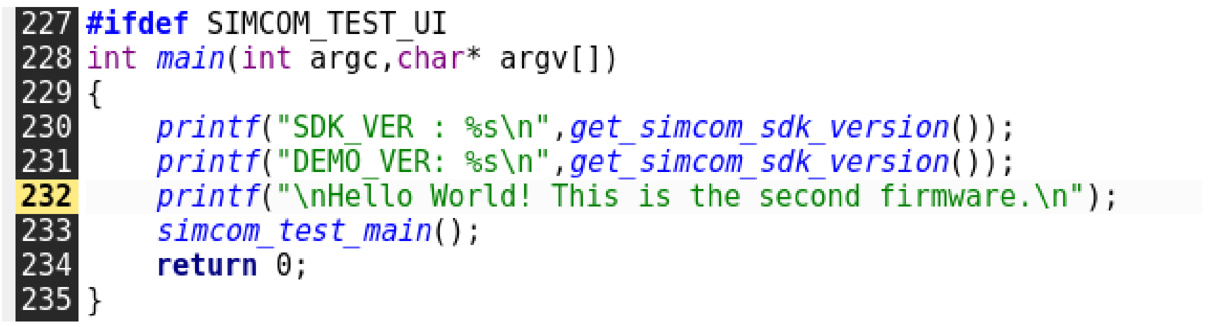
Note: You can skip to step 5.2.2 if you do not want to change or have already changed the original firmware

Let's modify the original firmware so we can have a different firmware version.

Open the folder **/sim_open_sdk/simcom-demo/src/main.c** and edit the file **main.c**

Add a new line after line 231 with the code snippet below and save the file:

```
printf("\nHello World! This is the second firmware.\n");
```

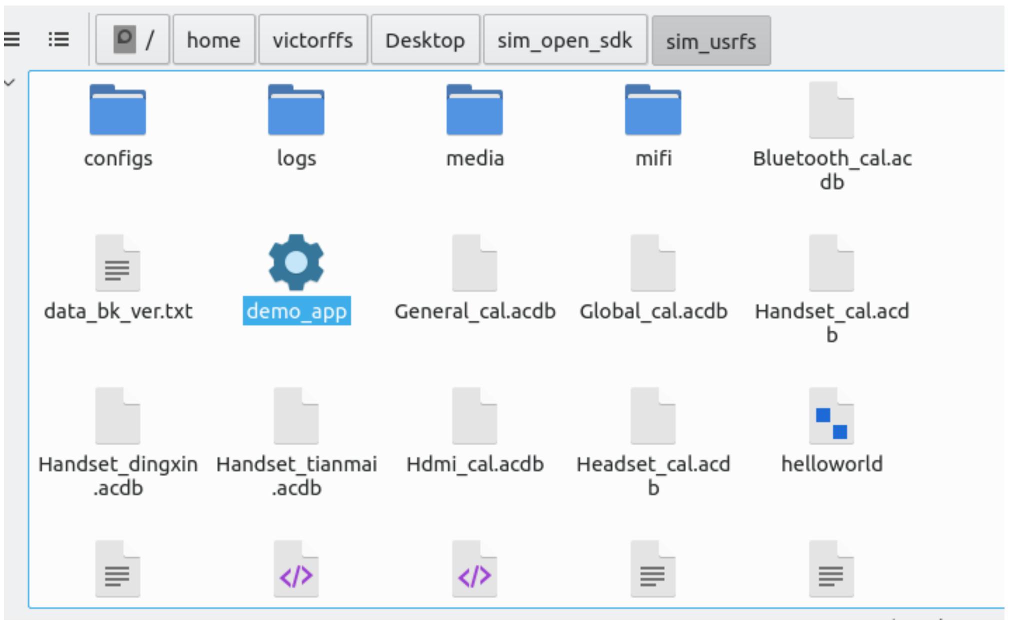


```
#ifdef SIMCOM_TEST_UI
int main(int argc,char* argv[])
{
    printf("SDK_VER : %s\n",get_simcom_sdk_version());
    printf("DEMO_VER: %s\n",get_simcom_sdk_version());
    printf("\nHello World! This is the second firmware.\n");
    simcom_test_main();
    return 0;
}
```

Return to the terminal in the folder **sim_open_sdk**, initialize the variables using the command **source** and compile the demo using the command **make**:

```
source sim_crosscompile/sim-crosscompile-env-init
make demo
```

Then move the new **demo_app** to the folder **/sim_open_sdk/sim_usrfs**



5.2.2 Delta FOTA Source and Packing

Delta FOTA analyzes and compares files from two folders inside **/sim_open_sdk/sim_ota** so you can generate the firmware with only the necessary changes.

- The first one is the **source** folder (not yet existing) which will be the original firmware code
- The second one is the **target** folder which will be the new firmware code.

Access the folder **/sim_open_sdk/sim_ota** through the terminal and copy the folder **target** changing the name to **source** through the command:

```
sudo cp -a target source
```

Now that we have defined our original firwmare, let's define our new firmware.

Return to **/sim_open_sdk** folder through the terminal and compile the OTA for the new firmware inside target folder.

```
make ota
```

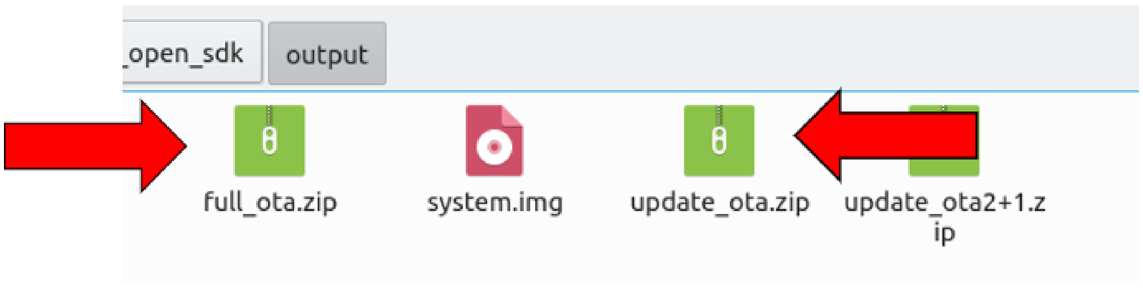
Finally, run the **ota.sh** script to compare firmwares and package only modified code.

```
bash ota.sh
```

You will now see that we have 2 new files inside the output folder, we will use just the **update_ota.zip**.

Note that the update_ota.zip file is about 500KB, much smaller than full_ota.zip

Send the **full_ota.zip** and **update_ota.zip** files to the Windows computer and we will continue the steps.

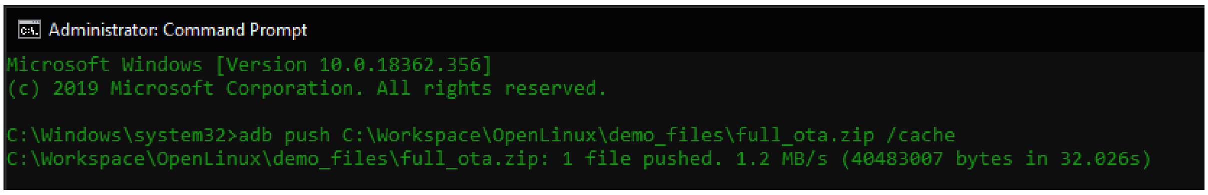


6. Full FOTA via USB

Because of the size of Full FOTA, we will send it to the module via USB.

Open the Windows command terminal, browse to the folder where ADB is installed, and use the command below, (replacing the file location with your Full FOTA file)

```
adb push C:\Workspace\OpenLinux\demo_files\full_ota.zip /cache
```



```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>adb push C:\Workspace\OpenLinux\demo_files\full_ota.zip /cache
C:\Workspace\OpenLinux\demo_files\full_ota.zip: 1 file pushed. 1.2 MB/s (40483007 bytes in 32.026s)
```

Now open the serial terminal and send the command that will update the firmware:

```
AT+CDELTA="/cache/full_ota.zip"
```

The module will restart and enter update mode. No action is required, just wait (about 1min).

When ready, the module will boot and the COM port will be available.

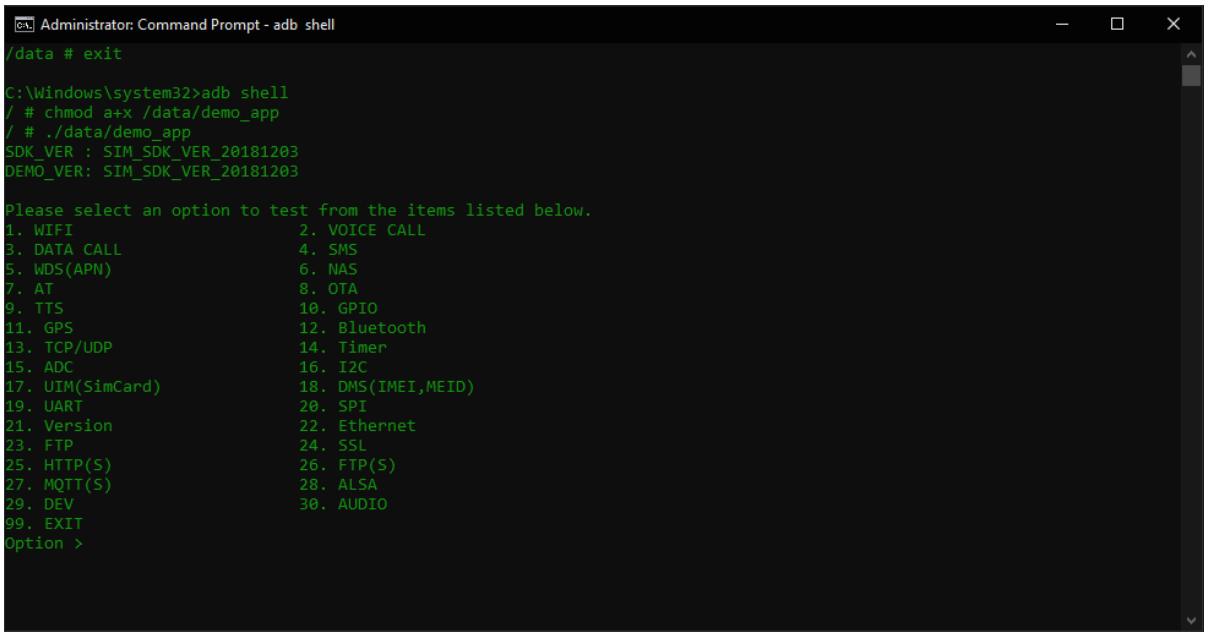
Now let's check if the original application is working.

Open windows terminal and run command

```
adb shell
```

Change the permission of the demo application and run it:

```
chmod a+x /data/demo_app
./data/demo_app
```



```
/data # exit

C:\Windows\system32>adb shell
/ # chmod a+x /data/demo_app
/ # ./data/demo_app
SDK_VER : SIM_SDK_VER_20181203
DEMO_VER: SIM_SDK_VER_20181203

Please select an option to test from the items listed below.
1. WIFI                       2. VOICE CALL
3. DATA CALL                  4. SMS
5. WDS(APN)                   6. NAS
7. AT                         8. OTA
9. TTS                        10. GPIO
11. GPS                       12. Bluetooth
13. TCP/UDP                   14. Timer
15. ADC                       16. I2C
17. UIM(SimCard)              18. DMS(IMEI,MEID)
19. UART                      20. SPI
21. Version                   22. Ethernet
23. FTP                       24. SSL
25. HTTP(S)                   26. FTP(S)
27. MQTT(S)                   28. ALSA
29. DEV                       30. AUDIO
99. EXIT
Option >
```

Finally, browse to the **/cache** folder and delete the **full_ota.zip** file with the command:

```
rm /cache/full_ota.zip
```

7. Delta FOTA via FTP

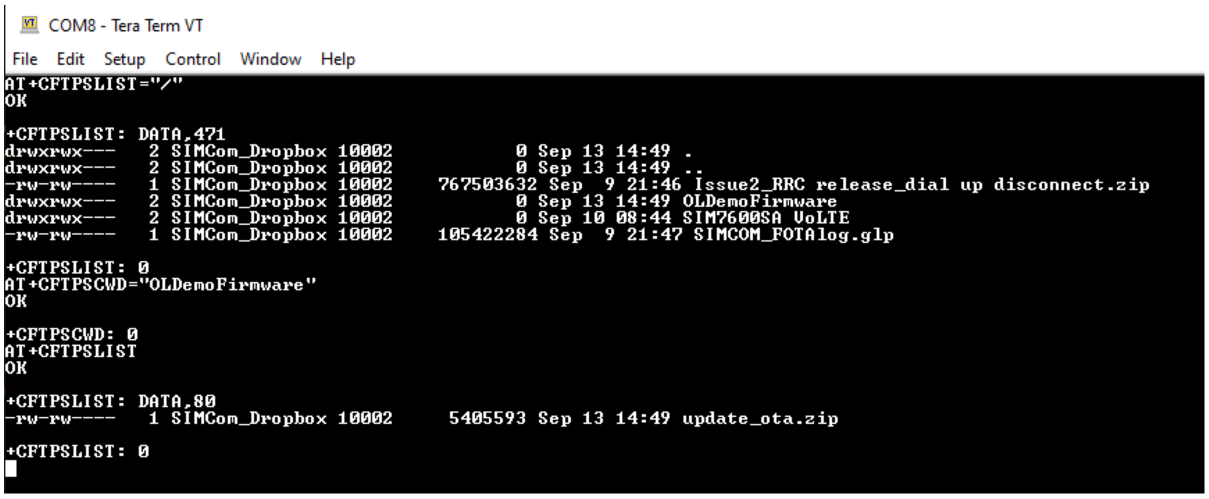
Now we will download Delta FOTA via FTP. First, upload the **update_ota.zip** file to your server.

Then access the server with the module through the serial terminal and the command AT+CFTPSLOGIN

```
AT+NETOPEN
AT+CFTPSSTART
AT+CFTPSSINGLEIP=1
AT+CFTPSLOGIN="YOURIP",PORT,"USER","PASSWORD",0
```

Browse inside the server using the AT+CFTPSLIST and AT+CFTPSCWD commands to list and switch folders respectively.

```
AT+CFTPSLIST="/"
AT+CFTPSLIST
AT+CFTPSCWD="OLDemoFirmware"
```



```
AT+CFTPSLIST="/"
OK

+CFTPSLIST: DATA,471
drwxrwx---   2 SIMCom_Dropbox 10002          0 Sep 13 14:49 .
drwxrwx---   2 SIMCom_Dropbox 10002          0 Sep 13 14:49 ..
-rw-rw----   1 SIMCom_Dropbox 10002  767503632 Sep  9 21:46 Issue2_RRC release_dial up disconnect.zip
drwxrwx---   2 SIMCom_Dropbox 10002          0 Sep 13 14:49 OLDemoFirmware
drwxrwx---   2 SIMCom_Dropbox 10002          0 Sep 10 08:44 SIM7600SA VoLTE
-rw-rw----   1 SIMCom_Dropbox 10002  105422284 Sep  9 21:47 SIMCOM_FOTAlog.glp

+CFTPSLIST: 0
AT+CFTPSCWD="OLDemoFirmware"
OK

+CFTPSCWD: 0
AT+CFTPSLIST
OK

+CFTPSLIST: DATA,80
-rw-rw----   1 SIMCom_Dropbox 10002    5405593 Sep 13 14:49 update_ota.zip

+CFTPSLIST: 0
```

When you find the file, download it to the module cache using the command below.

```
AT+CFTPSGETFILE="update_ota.zip",1
```

If the download is successful, the module should return **+CFTPSGETFILE=0**.

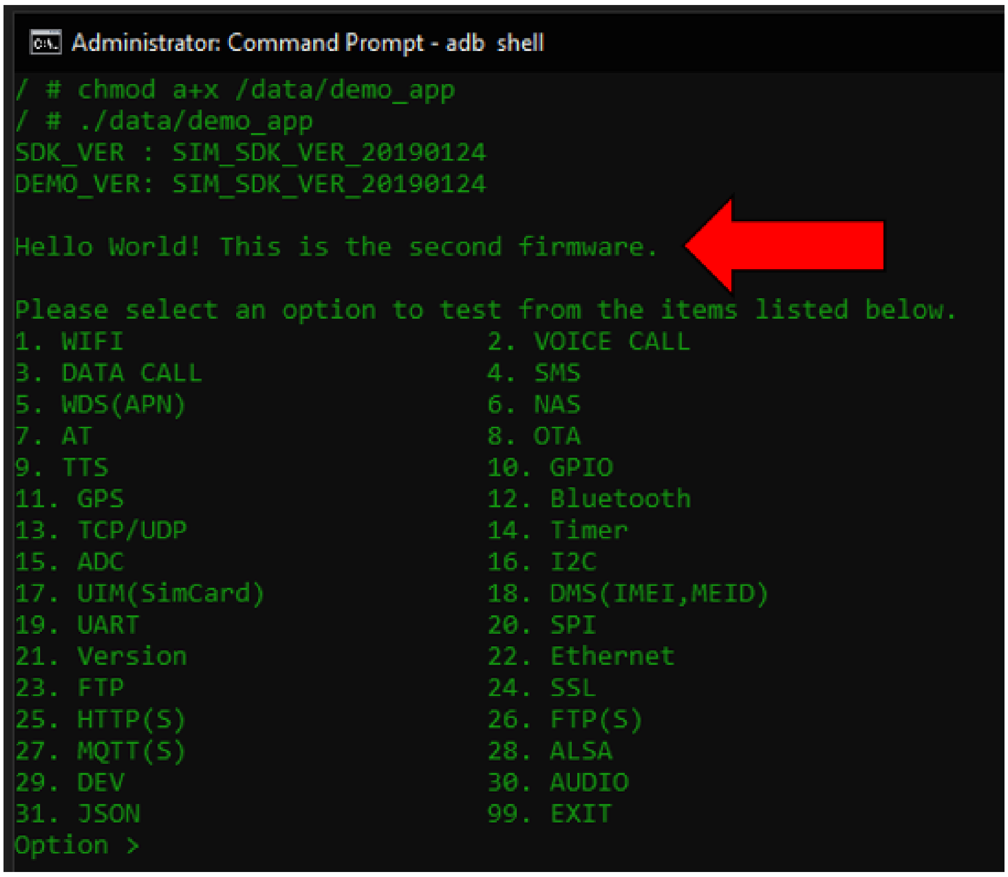
Now let's update the firmware by the new downloaded.

```
AT+CDELTA="/cache/update_ota.zip"
```

The module will restart and initialize when completed as previously.

Open the Windows terminal with adb shell and see the new updated application.

```
chmod a+x /data/demo_app
./data/demo_app
```



```
/ # chmod a+x /data/demo_app
/ # ./data/demo_app
SDK_VER : SIM_SDK_VER_20190124
DEMO_VER: SIM_SDK_VER_20190124

Hello World! This is the second firmware.

Please select an option to test from the items listed below.
1. WIFI                       2. VOICE CALL
3. DATA CALL                  4. SMS
5. WDS(APN)                   6. NAS
7. AT                         8. OTA
9. TTS                        10. GPIO
11. GPS                       12. Bluetooth
13. TCP/UDP                   14. Timer
15. ADC                       16. I2C
17. UIM(SimCard)              18. DMS(IMEI,MEID)
19. UART                      20. SPI
21. Version                   22. Ethernet
23. FTP                       24. SSL
25. HTTP(S)                   26. FTP(S)
27. MQTT(S)                   28. ALSA
29. DEV                       30. AUDIO
31. JSON                      99. EXIT
Option >
```

To know each command in more detail, please check the SIMCom OpenLinux documentation.

If you have additional questions, please send a email to victor.fragoso@simcom.com