NRC7292 Application Note
(Firmware OTA)
Ultra-low power & Long-range Wi-Fi

Ver 1.3
Sep. 8, 2021

NEWRACOM, Inc.

NRC7292 Application Note (Firmware OTA)
Ultra-low power & Long-range Wi-Fi

Office

Newracom, Inc.
25361 Commercentre Drive, Lake Forest, CA 92630 USA
http://www.newracom.com

Contents

1 Overview 5

2 HTTP(S) Server 6

3 FOTA Sample Procedure 8

3.1 FOTA server location 9

3.2 FOTA sample application file configuration 9

3.3 FOTA server operation 10

3.4 FOTA Sample SDK operation (sample_fota) 11

4 Revision history 14

List of Figures

Figure 1.1 FOTA procedure 5
Figure 2.1 Directory listing via a web browser 6
Figure 2.2 An example version file 7
Figure 2.3 CRC32 Python script (crc.py) 7
Figure 3.1 FOTA procedure in sample application 8
Figure 3.2 FOTA network diagram 9
Figure 3.3 configuration in sample_fota.c 9
Figure 3.4 Three files on HTTP_Server 10
Figure 3.5 Run crc.py 10
Figure 3.6 Edit ‘version’ file 10
Figure 3.7 Starting the FOTA server operation (HTTP) 10
Figure 3.8 Starting the FOTA server operation (HTTPS) 11
Figure 3.9 Sample FOTA application log (pre-update binary) 13
Figure 3.10 Sample FOTA application log (post-update binary) 13

1 Overview

This short user guide is a supplementary document for the sample FOTA (Firmware Over-the-air) update application. The reference application will demonstrate the FOTA functionality using a simple Python-based HTTP server. But any other custom HTTP server that can communicate using the FOTA protocol could also be utilized instead.

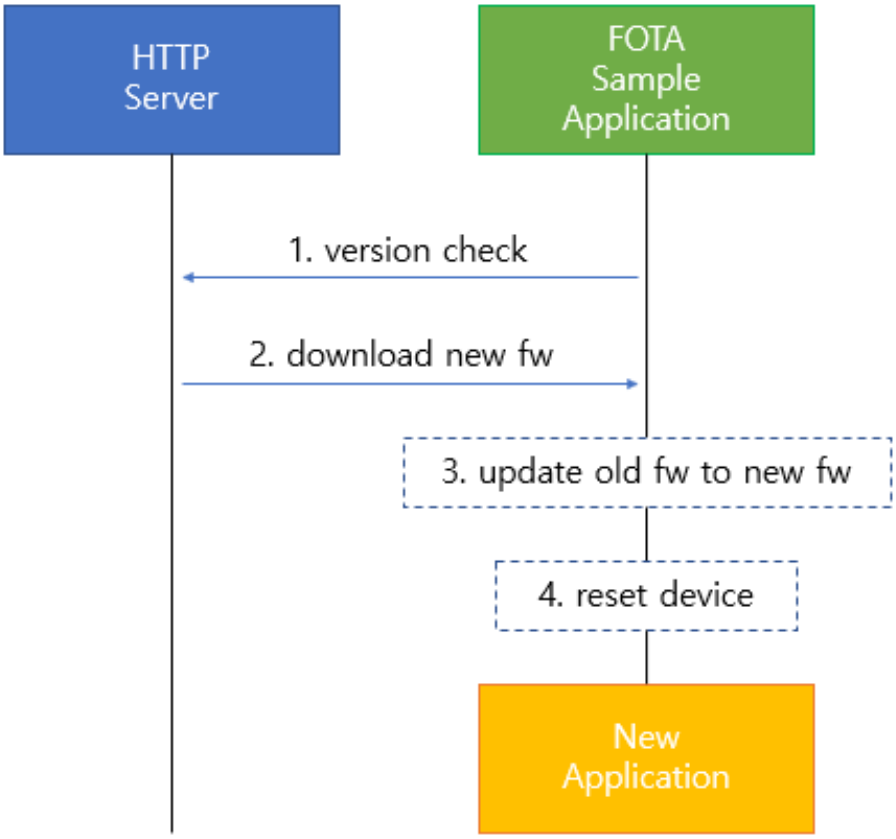

Figure 1.1 FOTA procedure

In the example application, the module periodically fetches the latest firmware version identifier every 10 seconds from the HTTP server. If a new version is available on the server, the new firmware binary is downloaded and updated. More detailed information about the procedure is outlined in later chapters.

2 HTTP(S) Server

The Python-based Simple HTTP Server will be used for the example application. The server will act as both a web server and a file server. The terminal command to start the server is a simple one-liner:

$python3 -m http.server <port number>

Sample FOTA also supports HTTPS environment with MbedTLS applied. Refer to the separate script file for the execution of the HTTPS Server. After running the script above, the server can be accessed remotely using a browser by typing the IP and the port number corresponding to the machine hosting the server. For example, if the IP of the host running the server is 192.168.1.11 and the port is set to 12345, typing "192.168.1.11:12345" in the search bar will redirect the browser to the index page. Upon accessing the server using a browser, the first page will display the list of files in the directory from which the server was started. More detailed information about setting up the server will be covered in a later chapter.

Figure 2.1 Directory listing via a web browser

The directory contains three types of files:

***.bin:**

The files ending with the '.bin' extension are firmware binaries.

crc.py:

This Python script is used by the server to compute the CRC bits of firmware binaries for integrity check.

version:

The JSON formatted version file consists following fields:

1. “version” : the version identifier of the latest firmware,
2. “crc” : the computed CRC bit sequence corresponding to the latest firmware binary and
3. “fw_name” : the file name of the latest firmware binary.

```
{
    "version" : "106",
    "crc" : "97cb8611",
    "fw_name" : "new_FW.bin"
}
```

Figure 2.2 An example version file

```python
#!/usr/bin/python
import sys
import zlib

def unsigned32(n):
    return n & 0xFFFFFFFF

def getCRC32(filename):
    try:
        f = open(filename, 'rb')
        crc = zlib.crc32(f.read())
        return unsigned32(crc)
    except IOError:
        print("Error: Cannot fine such file.")
        #return -1
        exit(1)

if len(sys.argv) == 1:
    print("Error: There's no file name")
    print("Usage:")
    print("    $python crc.py <file name>")
    exit(1)

print("%x" % getCRC32(sys.argv[1]))
```

Figure 2.3 CRC32 Python script (crc.py)

3 FOTA Sample Procedure

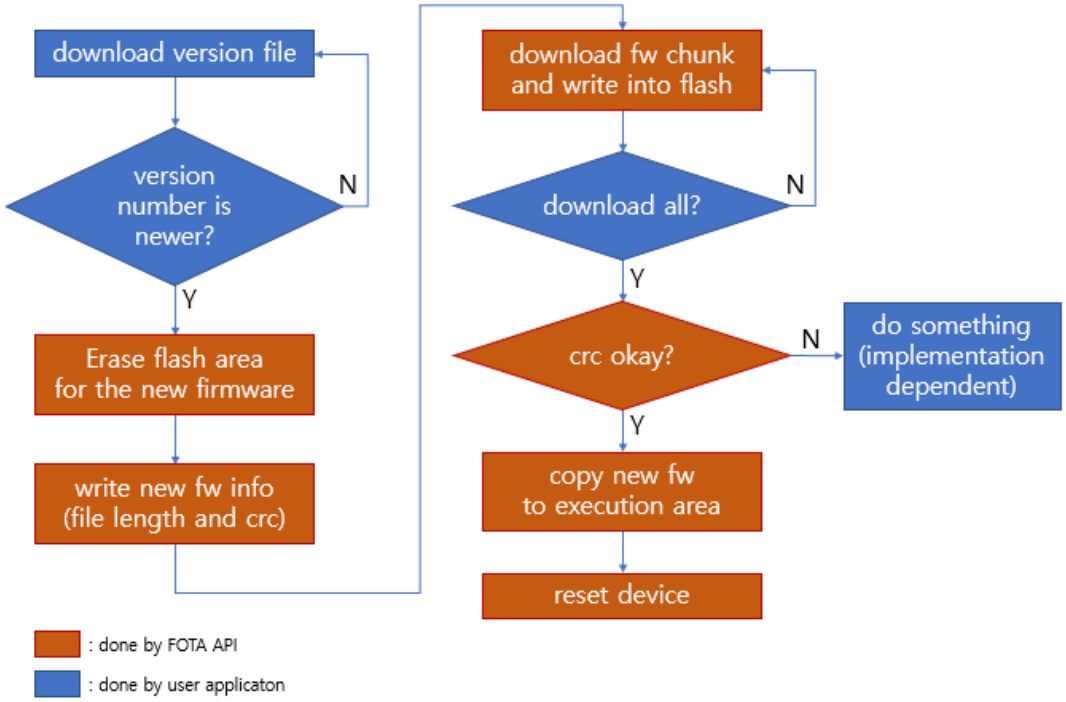

Figure 3.1 FOTA procedure in sample application

The diagram above shows the flowchart of the sample FOTA procedure. The module starts out by periodically fetching the version file from the HTTP server every 10 seconds. The version file contains information including the latest firmware version identifier, the CRC bits corresponding to the firmware binary file and the file name of the latest firmware binary. If the version available on the server is newer than the existing version on the module, the module proceeds with the update procedure.

The update procedure consists of multiple steps. First, the module erases the flash address block designated for the metainformation (file length and expected CRC bits) and the firmware binary. Then, the metainformation derived from the version file and the HTTP header is written to the flash and the download begins. Once the download is complete, the CRC bits corresponding to the downloaded firmware binary is calculated once again on the module and compared with the expected CRC bits stored in the received metainformation. If the two CRC bit sequences match, the new firmware binary is copied to the XIP memory block and the module will reset on its own to run the new firmware binary. A mismatch between the two CRC bit sequences indicates the corruption of the downloaded binary file and its handling is implementation-dependent, although a reasonable approach would be to reattempt the download procedure.

The FOTA API is required for implementing the FOTA procedure. The SDK API document contains more detailed information about the FOTA API usage.

3.1 FOTA server location

The FOTA HTTP server can either run on the AP or elsewhere on another external host.

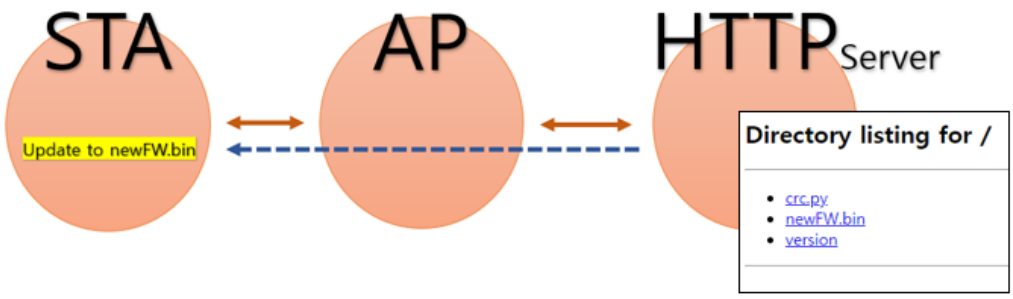

Figure 3.2 FOTA network diagram

3.2 FOTA sample application file configuration

```
33 #define RUN_HTTPS
34 #if defined( SUPPORT_MBEDTLS ) && defined( RUN_HTTPS )
35 #define SERVER_URL "https://192.168.10.199:4443/"
36 #else
37 #define SERVER_URL "http://192.168.10.199:8080/"
38 #endif
39
40 #define CURRENT_FW_VER 105
41 #define CHECK_VER SERVER_URL "version"
42 #define CHUNK_SIZE 2048
```

Figure 3.3 configuration in sample_fota.c

The following parameters in the sample source file "**sample_fota.c**" are relevant for FOTA operation:

- **[LINE 33-38] SERVER_URL.**
  The FOTA HTTP/HTTPS Server URL.
- **[LINE 40] CURRENT_FW_VER:**
  The current firmware version.
- **[LINE 41] CHECK_VER**
  EX.) 'http://192.168.10.199/version'
- **[LINE 42] CHUNK_SIZE**
  Chunk Size Definition.

3.3 FOTA server operation

The HTTP server requires three files: ‘crc.py’, ‘newFW.bin’, and ‘version’.

```
root@damonoh-H81N:~/FOTA# ls -al
total 644
drwxr-xr-x  2 root root   4096 11월  15 11:35 .
drwx------ 32 root root   4096 11월  15 14:01 ..
-rwxr-xr-x  1 root root    501 10월  28 15:24 crc.py
-rwxr-xr-x  1 root root 642328 10월  29 17:50 newFW.bin
-rwxr-xr-x  1 root root     23 11월  15 11:03 version
```

Figure 3.4 Three files on HTTP_Server

To update the version file, calculate the CRC value of newFW.bin using 'crc.py’.

- Run crc.py ($ python crc.py newFW.bin)
- Update the generated CRC value to the ‘version’ file

```
root@damonoh-H81N:~/FOTA# python crc.py newFW.bin
97cb8611
```

Figure 3.5 Run crc.py

The firmware version, the CRC value of the new firmware and the name of the new firmware binary must be updated in the JSON version file.

```
{
“version” : “110”,    -> Update firmware if the version is higher than current version installed.
“crc” : “97cb8611”,              -> Created CRC value
“fw_name” : “new_FW.bin” -> New firmware file name (defined by user)
}
```

Figure 3.6 Edit ‘version’ file

HTTP Server execute depending on whether Python2 or Python3 is used, one of the following terminal commands can be used in the directory containing the three files to start the FOTA server operation.

python version	command
python2	python -m SimpleHTTPServer 8080
python3	python3 -m http.server 8080

```
root@damonoh-H81N:~/FOTA# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

Figure 3.7 Starting the FOTA server operation (HTTP)

HTTPS Server execute depending on whether Python2 or Python3 is used (**Use 'Https_Server.py'**)

```
root@damonoh-H81N:~/FOTA# python3 Https_Server.py
Serving HTTP on 0.0.0.0 port 4443..

root@damonoh-H81N:~/FOTA# python2 Https_Server.py
Serving HTTP on 0.0.0.0 port 4443..
```

Figure 3.8 Starting the FOTA server operation (HTTPS)

3.4 FOTA Sample SDK operation (sample_fota)

- In the sample_fota application, the module periodically fetches the latest firmware version identifier **every 10 seconds** from the HTTP server. If a new version is available (over 105), new firmware binary is downloaded, and the existing application is updated.

- **Step 1) Check the firmware version in the server.**
- **Step 2) Format the flash memory block for downloading OTA firmware.**
- **Step 3) Download the new firmware binary and Update**
- **Step 4) Replace the old binary and Reboot(Execution new firmware)**

```
========================================
STEP 1. Check firmware version in server.
========================================
[httpc_parse_url] scheme:http, host:10.198.1.214, port:8080, uri:/version
s1g: 0 update beacon interval(1000)
s1g: 0 listen_interval: 100 beacon_interval: 1000
s1g: 0 update short beacon interval(intv: 100)
s1g: 0 listen_interval: 100 short beacon_interval: 100
[httpc_connect] result:0, handle:0x00000000
--- Request Header ---
GET /version HTTP/1.1
Host: 10.198.1.214:8080

[httpc_recv] size=200

[httpc_recv] size=80

[httpc_recv] size=0
[httpc_close] httpc_close()
[HTTP Response Length] 280
-----Recvd Data-----
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.8.10
Date: Thu, 09 Sep 2021 17:27:16 GMT
Content-type: application/octet-stream
Content-Length: 80
Last-Modified: Wed, 08 Sep 2021 20:59:08 GMT

{
    "version" : "106",
    "crc" : "f8cbb153",
    "fw_name" : "new_FW.bin"
}

[parse_response_version] data length : 280, body length : 80
[parse_response_version] version data : {
    "version" : "106",
    "crc" : "f8cbb153",
    "fw_name" : "new_FW.bin"
}

[parse_response_version] version : 106
[parse_response_version] crc : f8cbb153
[parse_response_version] URL : new_FW.bin
[parse_response_version] version: 106,  crc: f8cbb153  fw_url: http://10.198.1.214:8080/new_FW.bin
========================================================
STEP 2. Erase flash area for OTA firmware to be downloaded.
========================================================
Erasing......Done.

========================================
STEP 3. Download new firmware and update.
========================================

[httpc_parse_url] scheme:http, host:10.198.1.214, port:8080, uri:/new_FW.bin
[httpc_connect] result:0, handle:0x00000000
--- Request Header ---
GET /new_FW.bin HTTP/1.1
Host: 10.198.1.214:8080

[httpc_recv] size=204
-- head content length: 204

[HTTP Response Length] 272866032
-----Recvd Data-----
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.8.10
Date: Thu, 09 Sep 2021 17:27:19 GMT
Content-type: application/octet-stream
Content-Length: 828224
Last-Modified: Wed, 08 Sep 2021 20:57:56 GMT


[parse_response_content] fw_len = 828224

FW Size: 828224
FW CRC: 0xF8CBB153
```

Figure 3.9 Sample FOTA application log (pre-update binary)

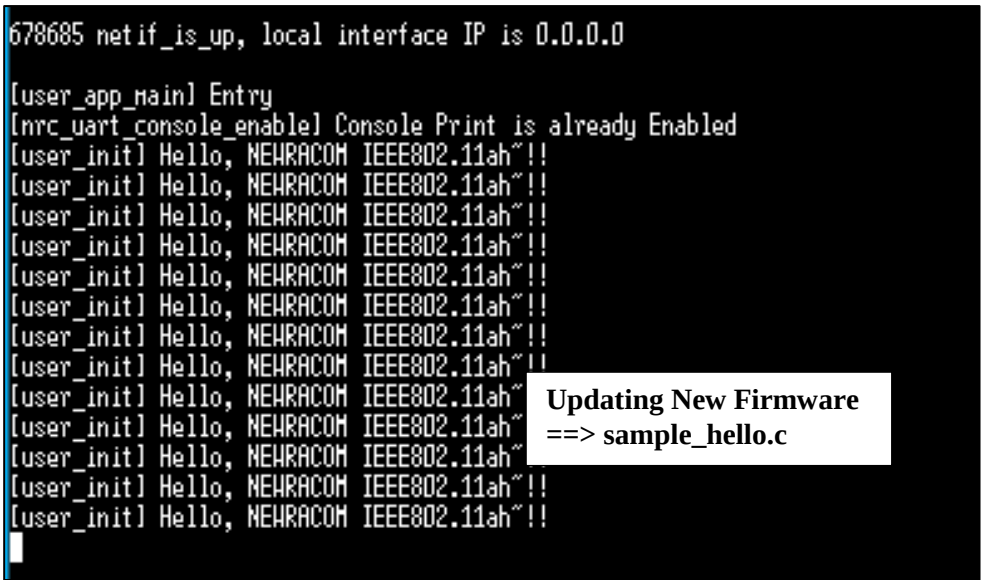

Figure 3.10 Sample FOTA application log (post-update binary)

4 Revision history

Revision No	Date	Comments
Ver 1.0	03/25/2020	Initial version
Ver 1.1	05/25/2020	Add Support MbedTLS (HTTPS)
Ver 1.2	04/13/2021	Update figure about version file
Ver 1.3	09/08/2021	Update version file to JSON format