



NRC7394 Application Note

(Firmware OTA)

Ultra-low power & Long-range Wi-Fi

Ver 1.0
Jul. 17, 2023

NEWRACOM, Inc.

NRC7394 Application Note (Firmware OTA)

Ultra-low power & Long-range Wi-Fi

© 2023 NEWRACOM, Inc.

All right reserved. No part of this document may be reproduced in any form without written permission from Newracom.

Newracom reserves the right to change in its products or product specification to improve function or design at any time without notice.

Office

Newracom, Inc.

505 Technology Drive, Irvine, CA 92618 USA

<http://www.newracom.com>

Contents

- 1 Overview..... 5**
 - 1.1 Unicast FOTA.....5
 - 1.2 Broadcast FOTA.....6
- 2 HTTP(S) Server..... 7**
- 3 FOTA Sample Procedure..... 9**
 - 3.1 FOTA server location.....10
 - 3.2 FOTA sample application file configuration.....10
 - 3.3 FOTA server operation11
 - 3.4 FOTA Sample SDK operation (sample_fota)12
- 4 Revision history..... 15**

List of Figures

Figure 1.1	FOTA procedure	5
Figure 1.2	Broadcast FOTA procedure	6
Figure 2.1	Directory listing via a web browser	7
Figure 2.2	An example version file.....	8
Figure 2.3	CRC32 Python script (crc.py).....	8
Figure 3.1	FOTA procedure in sample application.....	9
Figure 3.2	FOTA network diagram	10
Figure 3.3	configuration in sample_fota.c.....	10
Figure 3.4	Three files on HTTP_Server.....	11
Figure 3.5	Run crc.py.....	11
Figure 3.6	Edit 'version' file.....	11
Figure 3.7	Starting the FOTA server operation (HTTP)	11
Figure 3.8	Starting the FOTA server operation (HTTPS)	12
Figure 3.9	Sample FOTA application log (pre-update binary)	13
Figure 3.10	Sample FOTA application log (post-update binary).....	14

1 Overview

The current implementation of the application includes support for two firmware update procedures. The first procedure involves periodic polling of the server to check for available firmware updates. The second procedure involves listening for incoming firmware updates through the beacon frame. This user guide serves as a supplementary document for the sample FOTA (Firmware Over-the-Air) update application and the broadcast FOTA functionality.

1.1 Unicast FOTA

The reference 'sample_fota' application aims to showcase the functionality of Firmware Over-The-Air (FOTA) using a basic Python-based HTTP server. However, it is also possible to employ any other custom HTTP server capable of communicating through the FOTA protocol as an alternative.

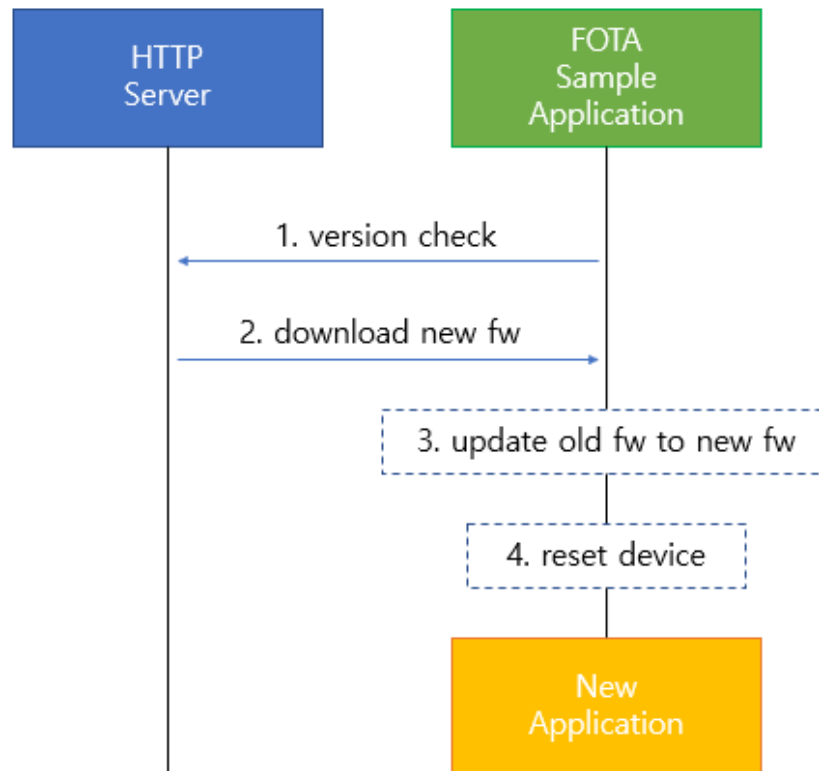


Figure 1.1 FOTA procedure

In the example application "sample_fota," the module follows a periodic fetching approach where it retrieves the new firmware version identifier every 10 seconds from the HTTP server. If a newer version is found on the server, the module proceeds to download and update the firmware binary accordingly. For a more comprehensive understanding of this procedure, you can refer to Chapter 2 and Chapter 3, which provide detailed information about the update process.

1.2 Broadcast FOTA

In contrast to the periodic polling approach used in "sample_fota," the broadcast FOTA is designed to command devices to update their firmware. This feature enables the simultaneous upgrade of a large number of devices by sending small chunks of firmware data in beacon frames from an access point. When the access point initiates the broadcast FOTA, all devices paired with the access point prepare to listen for incoming beacons containing firmware data chunks. The process to initiate the broadcast FOTA is detailed in Chapter 4, providing a step-by-step illustration of the procedure.

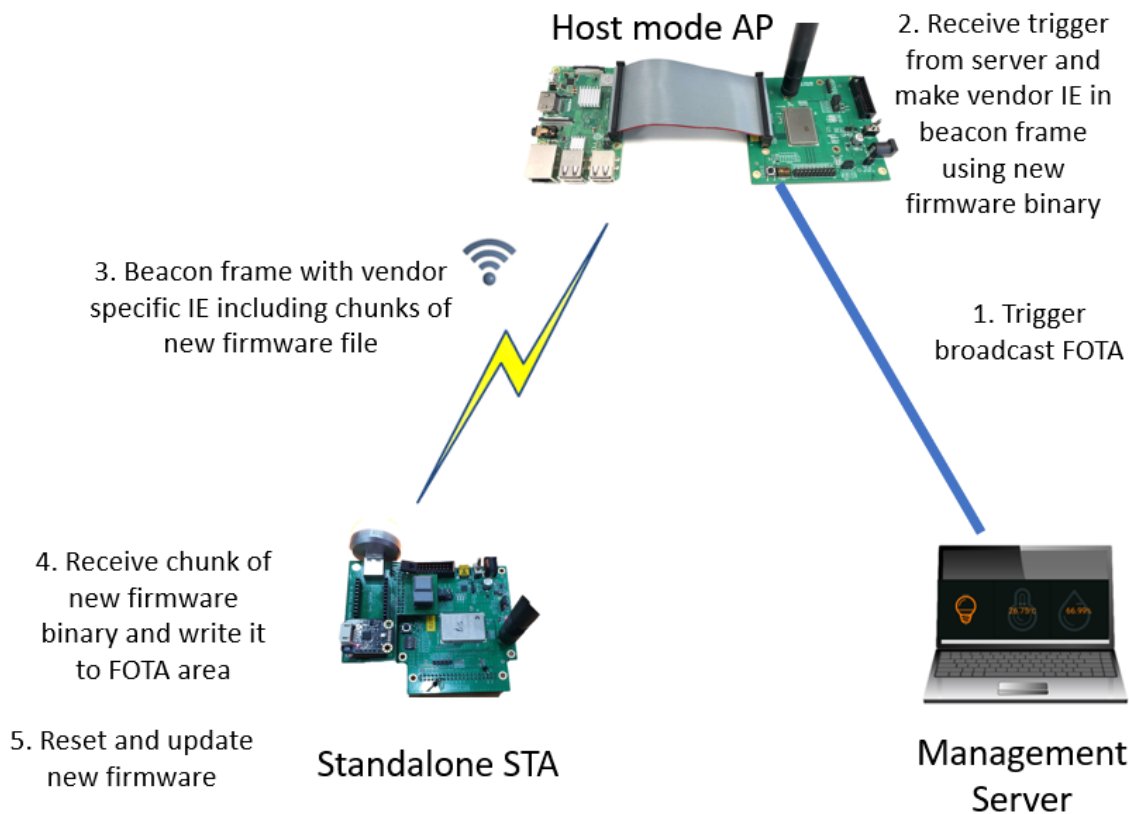


Figure 1.2 Broadcast FOTA procedure

2 HTTP(S) Server

For the example application, a Python-based Simple HTTP Server will be utilized, which functions as both a web server and a file server. To start the server, a simple one-liner terminal command can be used:

```
$python3 -m http.server <port number>
```

In addition, the sample FOTA application also supports an HTTPS environment with MbedTLS implementation. For executing the HTTPS Server, a separate script file is provided. Once the server is running, it can be accessed remotely using a browser by entering the IP address and port number of the machine hosting the server. For instance, if the server's host IP is 192.168.1.11 and the port is set to 12345, typing "192.168.1.11:12345" in the browser's search bar will redirect to the index page of the server. The initial page displayed in the browser will show the list of files in the directory from which the server was started. Further instructions on setting up the server will be covered in a later chapter, providing more detailed information.

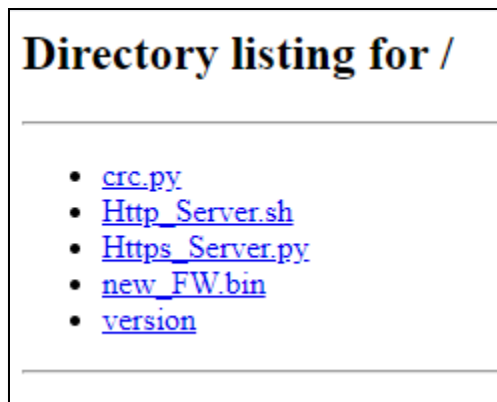


Figure 2.1 Directory listing via a web browser

The directory contains three types of files:

file name	Description
*.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.
4. "force" : the force update flag : 1(true) or 0 (false)

```
{  
    "version" : "1.1.0",  
    "crc" : "3cad437a",  
    "force" : "0",  
    "fw_name" : "nrc7394_standalone_xip_hello_world.bin"  
}
```

Figure 2.2 An example version file

```
1  #!/usr/bin/python  
2  import sys  
3  import zlib  
4  
5  def unsigned32(n):  
6  |... return n & 0xFFFFFFFF  
7  
8  def getCRC32(filename):  
9  |... try:  
10 |... |... f = open(filename, 'rb')  
11 |... |... crc = zlib.crc32(f.read())  
12 |... |... return unsigned32(crc)  
13 |... except IOError:  
14 |... |... print("Error: Cannot find such file.")  
15 |... |... #return -1  
16 |... |... exit(1)  
17  
18 if len(sys.argv) == 1:  
19 |... print("Error: There's no file name")  
20 |... print("Usage:")  
21 |... print("... $python crc.py <file name>")  
22 |... exit(1)  
23  
24 print("%x" % getCRC32(sys.argv[1]))  
25
```

Figure 2.3 CRC32 Python script (crc.py)

3 FOTA Sample Procedure

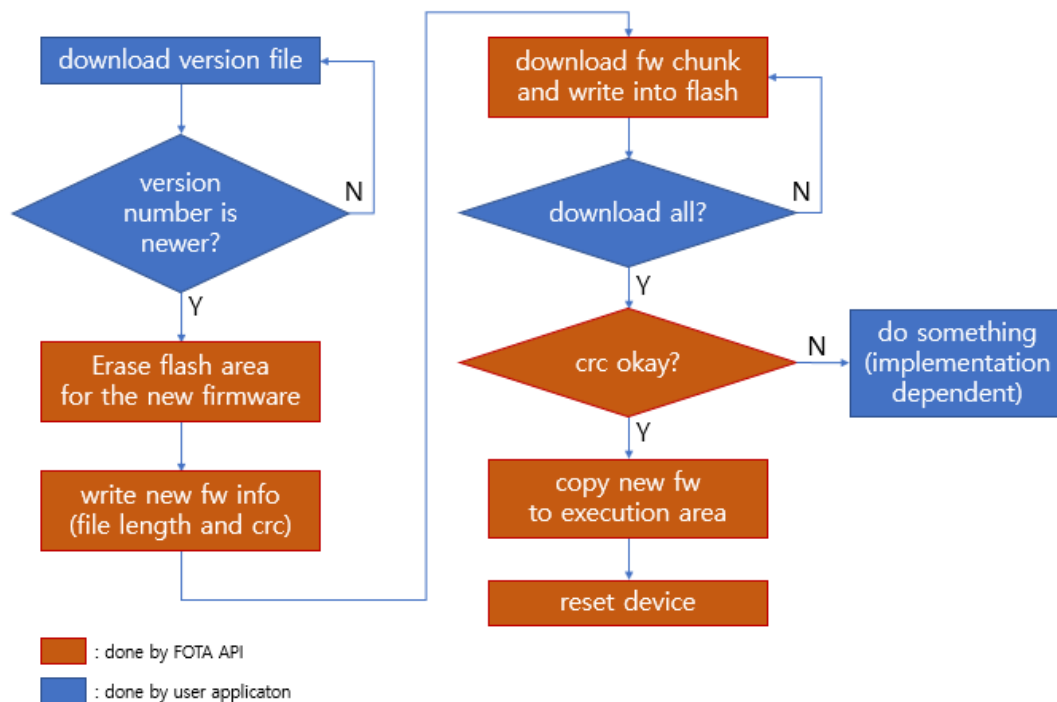


Figure 3.1 FOTA procedure in sample application

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 meta-information (file length and expected CRC bits) and the firmware binary. Then, the meta-information 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 meta-information. 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.

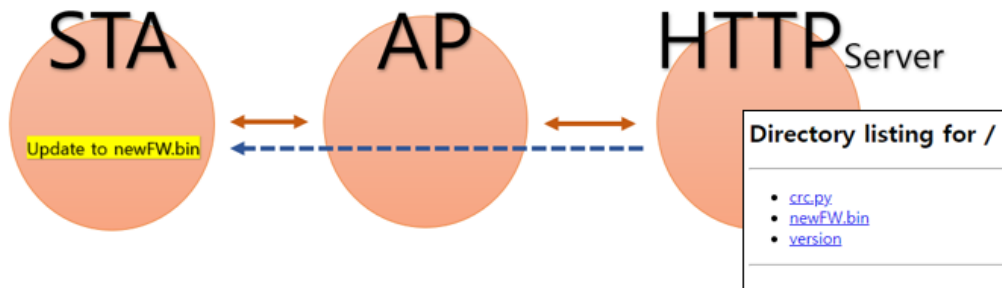


Figure 3.2 FOTA network diagram

3.2 FOTA sample application file configuration

```
#if defined( SUPPORT_MBEDTLS ) && defined( RUN_HTTPS )
#define SERVER_URL "https://192.168.200.1:4443/"
#else
#define SERVER_URL "http://192.168.200.1:8080/"
#endif
#define CHECK_VER_URL SERVER_URL "version"
#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:

- **SERVER_URL.**
The FOTA HTTP/HTTPS Server URL.
- **CHECK_VER_URL**
EX.) ‘http://192.168.10.199/version’
- **CHUNK_SIZE**
Chunk Size Definition.

※ SAMPLE version is existed in sample_fota_version.h

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 11:15 .
drwx----- 32 root root   4096 11:15 ..
-rwxr-xr-x  1 root root    501 10:28 crc.py
-rwxr-xr-x  1 root root 642328 10:29 newFW.bin
-rwxr-xr-x  1 root root     23 11:15 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_rcv] size=200

[httpc_rcv] size=80

[httpc_rcv] size=0
[httpc_close] httpc_close()
[HTTP Response Length] 280
-----Recv 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_rcv] size=204
-- head content length: 204

[HTTP Response Length] 272866032
-----Recv 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)

```
678685 netif_is_up, local interface IP is 0.0.0.0
[user_app_main] Entry
[nrc_uart_console_enable] Console Print is already Enabled
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~
[Updating New Firmware ==> sample_hello.c]
[user_init] Hello, NEWRACOM IEEE802.11ah~
[user_init] Hello, NEWRACOM IEEE802.11ah~
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
[user_init] Hello, NEWRACOM IEEE802.11ah~!!
```

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

4 Revision history

Revision No	Date	Comments
Ver 1.0	7/17/2023	Initial version