Over the Air Programming with 802.15.4 and ZigBee
Laying the groundwork
## INDEX

1. **Overview** ............................................................................................................................................... 4

2. **Benefits** .................................................................................................................................................. 4

3. **Concepts** .................................................................................................................................................. 5
   3.1. Binary File Generation ............................................................................................................................ 6
   3.2. OTA with 802.15.4/ZigBee modules ......................................................................................................... 7
   3.3. OTA with 3G/GPRS/WiFi modules via FTP ............................................................................................. 8

4. **Checking the SD card** ............................................................................................................................... 9
   4.1. Linux ..................................................................................................................................................... 9
   4.2. Windows .............................................................................................................................................. 10
   4.3. Mac OS ............................................................................................................................................. 11

5. **OTA with 802.15.4/ZigBee** ................................................................................................................... 12
   5.1. OTA Step by Step .................................................................................................................................. 12
   5.2. Memory and Storage System in Waspmote ......................................................................................... 15
   5.3. Topologies .......................................................................................................................................... 17
   5.4. OTA Shell .......................................................................................................................................... 26
      5.4.1. Acronyms ...................................................................................................................................... 26
      5.4.2. OTA Commands ............................................................................................................................ 26
      5.4.3. Configuring the 802.15.4/ZigBee radio ......................................................................................... 26
      5.4.4. Searching the node to upgrade ...................................................................................................... 27
      5.4.5. Getting the bootlist ........................................................................................................................ 27
      5.4.6. Preparing the Delivery ................................................................................................................... 28
      5.4.7. Sending the program ...................................................................................................................... 29
      5.4.8. Starting a new program ................................................................................................................. 31
      5.4.9. Delete a Program .......................................................................................................................... 32
      5.4.10. Responses .................................................................................................................................... 33
   5.5. Setting the environment ....................................................................................................................... 34
      5.5.1. Meshlium ....................................................................................................................................... 34
      5.5.2. Linux ......................................................................................................................................... 35
      5.5.3. Windows ..................................................................................................................................... 36
      5.5.4. Mac OSX ..................................................................................................................................... 38
   5.6. Hardware Setup .................................................................................................................................... 40
      5.6.1. Important dates and versions ......................................................................................................... 40
      5.6.2. Waspmote Setup .......................................................................................................................... 40
      5.6.3. Waspmote Gateway Setup ............................................................................................................ 42
6. OTA with 3G/GPRS/WiFi via FTP ................................................................. 44
   6.1. Procedure .............................................................................................. 44
   6.2. Setting the FTP server configuration .................................................. 45
       6.2.1. Extern FTP server setup .............................................................. 45
       6.2.2. Meshlium FTP server setup ....................................................... 46
   6.3. Wasmote basic sketch ........................................................................ 48
   6.4. OTA via GPRS and GPRS+GPS module ............................................. 49
   6.5. OTA via 3G module ............................................................................. 52
   6.6. OTA via WiFi module .......................................................................... 53

7. Documentation changelog .......................................................................... 55
1. Overview

The concept of Wireless Programming or commonly know as Programming Over the Air (OTA) has been used in the past years overall for the reprogramming of mobile devices such as cell phones. However, with the new concepts of Wireless Sensor Networks and the Internet of Things where the networks consist of hundreds or thousands of nodes (“motes”) OTA is taken to a new direction, and for the first time it is applied using unlicensed frequency bands (2.4GHz, 868MHz, 900MHz) and with low consumption and low data rate transmission using protocols such as 802.15.4 and ZigBee.

Besides, Libelium provides a new OTA method based on FTP transmissions to be used with GPRS, 3G and WiFi modules.

Note that the concept of OTA may have some other names such as:

- Over the air -> OTA
- Over the air Programming -> OTAP
- Firmware over the air -> FOTA
- Programming Over the air-> POTA
- Over the air service provisioning -> OTASP
- Over the air provisioning -> OTAP
- Over the air parameter administration -> OTAPA
- Over the air upgrade -> OTAU
- Over the air update -> OTAUR
- Over the air Download -> OAD
- Over the air flashing -> OTAF
- Multihop Over the air programming (MOTAP)

**Important:** The OTA process has been defined to work with the Waspmote platform. For more info go to:

http://www.libelium.com/development/waspmote

2. Benefits

Libelium OTA Benefits:

- OTA with 802.15.4/ZigBee:
  - Enables the upgrade or change of firmware versions without physical access.
  - Discover nodes in the area just sending a broadcast discovery query.
  - Upload new firmware in few minutes.
  - No interferences: OTA is performed using a change of channel between the programmer and the desired node so no interferences are generated to the rest of the nodes.

- OTA with 3G/GPRS/WiFi:
  - Enables the upgrade or change of firmware versions without physical access.
  - Upgrades the new firmware by querying an FTP server which helps to keep battery life.
  - Upload new firmware in few minutes.
3. Concepts

There are two different OTA methodologies:

- OTA with 802.15.4/ZigBee modules
- OTA with 3G/GPRS/WiFi modules via FTP

Hardware:

Libelium OTA has been designed to work exclusively over the Waspmote platform.

Waspmote is a sensor device specially oriented to developers. It works with different protocols such as: 802.15.4, ZigBee, WiFi, Bluetooth, 3G/GPRS, being capable of getting links up to 12km.

It counts with an hibernate mode of 0.07\(\mu\)A which allows to save battery when it is not transmitting. More than 60 sensors already available and a complete open source IDE (API libraries + compiler) make it really easy to start working with the platform.

Over the Air Programming (OTA) capabilities and Encryption libraries to ensure maximum security in communications are also available making Waspmote the most versatile sensor platform in the market.

**Note:** to run programs which make use of the hibernate mode, it is needed to set the hibernate switch off, so this feature cannot be used with OTA. However, sleep or deepsleep modes can be performed.
3.1. Binary File Generation

The Waspmote IDE v03 or higher automatically generates the binary file to be used in OTA programmings. When a program is compiled or uploaded to Waspmote, two files are generated inside /sketchbook/OTA-FILES folder: a hexadecimal and a binary file. The first one is valid for OTA with XBee modules and the second one is valid for OTA via FTP with 3G/GPRS/WiFi modules.

**STEP 1:** Click Verify button so as to compile the code

![Waspmote IDE generates binary files](image1)

**STEP 2:** Search the binary files in the correct folder:

![OTA-FILES folder](image2)
For instance, we are managing a program called OTA_06_GPRS.pde. When the program is compiled, two files are created inside the sketchbook sub-folder called OTA-FILES:

- OTA_06_.hex is the hexadecimal file used for conventional OTA programming.
- OTA_06_ (seven characters) is the binary file needed for OTA via 3G or GPRS.

The Wasp mote IDE gives these filenames truncating to seven characters the name of the Wasp mote program.

**Note:** It is important to remark that the filename given by the Wasp mote IDE to the binary files **CANNOT be changed.** Otherwise, OTA will not work.

### 3.2. OTA with 802.15.4/ZigBee modules

The idea is simple. When the programmer (normally the Gateway) sends a new program, the nodes store it in the SD card. A second command “start_new_program” is sent to the nodes in order to make the new program start. Then, the nodes copy the program from the SD card to the Flash memory and the new program starts.

**Steps:**

- Locate the node to upgrade.
- Check current software version.
- Send the new program.
- Reboot and start with the new program.
- Restore the previous program if the process fails.

**OTA modes:**

- Unicast: Reprogram an specific node.
- Multicast: Reprogram several nodes at the same time sending the program just once.
- Broadcast: Reprogram the entire network sending the program just once.

**Topologies:**

- Direct access: when the nodes are accessed in just one hop (no forwarding of the packets is needed).
- Multihop: when the nodes are accessed in two or more hops. In this mode some nodes have to forward the packets sent by the Gateway in order to reach the destination.

**Protocols supported:**

- 802.15.4 - 2.4GHz (Worldwide).
- ZigBee - 2.4GHz (Worldwide). **Important:** OTA operations only available from the Gateway, not from Meshlium.
- DigiMesh - 2.4GHz (Worldwide).
- RF - 868MHz (Europe).
- RF - 900MHz (US, Canada, Australia).

**Storage System:**

Once we have sent the program to Wasp mote it will store it in the internal memory, a 2GB SD card.

If we have into account that the maximum size for a program is 128KB, this means we can store thousands different firmware versions inside each node.

There are many SD card models. Any of them has defective blocks, which are ignored when using the Wasp mote’s SD library. However, when using OTA, those SD blocks cannot be avoided, so that the execution could crash. Libelium implements a special process to ensure the SD cards we provide will work fine with OTA. The only SD cards that Libelium can assure that work correctly with Wasp mote are the SD cards we distribute officially.
Encryption and Authentication:

All the data which is sent in the OTA process can be secured by activating the encryption algorithm AES 128b which works in the link layer. As well as this, a second pass key is needed to be known by the OTA programmer (the Gateway) in order to be authenticated and validated by each node before starting with the OTA action requested.

OTA-Shell:

The OTA-Shell application can be used in Windows, Linux and MacOS. It allows to control in a quick and powerful way all the options available in OTA. If you are using Meshlium as the Gateway of the network, the OTA-Shell environment comes already preinstalled and ready to use. This is the recommended way when deploying a real scenario.

3.3. OTA with 3G/GPRS/WiFi modules via FTP

The reprogramming process in this type of OTA is initiated by Wasp mote and it is supported by an FTP server.

Steps:

- Wasp mote queries the FTP server for a new program version
- Check if program name, path and version are correct
- Download the new program
- Reboot and start with the new program

Topologies:

- Protocols which support FTP transmissions are directly connected to the Network Access Point

Protocols supported:

- 3G - Tri-Band (2100/1900/900 MHz), Quad-Band (850/900/1800/1900 MHz)
- GPRS - 850/900/1800/1900 MHz
- WiFi - 2.4GHz (Worldwide).

Storage System:

Once the program is downloaded to Wasp mote it is stored it in the 2GB SD card.

Meshlium OTA-FTP plug-in

Meshlium provides a FTP server and Manager System plug-in which permits to configure the server automatically by attaching the program binary file to be used.
4. Checking the SD card

As explained before, not all the SD cards are compatible with OTA. Given an SD card, depending on the capacity of it, the user is able to know if it is compatible with OTA or not. To check the capacity of a SD card, please follow the next steps according to your OS:

4.1. Linux

It necessary to install a program like ‘gparted’ or similar. When checking the SD card you can see two possibilities. If the capacity is 1.87 GB, the SD card is correct:

On the other hand, if the size is 1.84 GB, that means there are bad sectors in the SD card, so that this card will not be able to implement the Wasmote OTA feature:
4.2. Windows

You can check the size of the SD card by clicking on the properties of the storage device. If the size of the device is 1.86 GB, the SD card is correct:

![Image of SD card properties in Windows]

Otherwise, if the size is 1.83 GB, there are some defective sectors and the OTA operations will fail:

![Image of SD card properties in Windows with defective sectors]
4.3. Mac OS

You can check the size of the SD card by clicking on the properties of the storage device. If the size of the device is 2 GB, the SD card is correct:

Otherwise, if the size is 1.98 GB, there are some defective sectors and the OTA operations will fail:
5. OTA with 802.15.4/ZigBee

5.1. OTA Step by Step

- Locate the node or nodes to upgrade

Using the ‘scan_nodes’ function we can search for a specific node or send a global query looking for any node which is ready to be reprogrammed with the OTA process.

The nodes which are ready at this moment will answer with a "Ready to OTA" frame.
- **Send the new program**

We can use the ‘send’ command with the unicast, multicast or broadcast option depending on how many nodes we want to reprogram at the same time.

Each node which receives the program sends a message to the gateway to inform of the success of the process.
- Reboot and start with the new program

In order to make the nodes start executing the new program, the gateway needs to send the 'start_new_program' command.

Each node which receives this packet will copy the program from the SD to the Flash memory and will start running the new binary.
5.2. Memory and Storage System in Waspmote

There are four different memory systems in Waspmote:

- **RAM (4KB)**: Volatile memory which saves the variables and instructions of the program being executed.
- **EEPROM (8KB)**: Memory used to store variables and certain flags used in the programs and which need to be kept after rebooting.
- **FLASH (128KB)**: Memory used to store the binary program which is currently running.
- **SD Card (2GB)**: High load memory system which uses FAT-16 file system. It allows to manage files which are controlled from an inode table. This is the place where all the new programs are stored.

There a special file called “boot.txt” which contains the information of all the programs available in Waspmote. Each time we add a new program a reference line is added. The content of this file can be accessed using the “get_boot_list” function.

**Important:** The file “boot.txt” can not be deleted as this could make future OTA processes fail.

In the diagram below we can see how the program received is stored in the SD card.
When the gateway sends the 'start_new_program' command the new binary is copied from the SD card to the Flash memory and then automatically started.
5.3. Topologies

There are 9 different topologies depending on the number of hops from the Gateway the data has to make in order to get the destination and how the subnetwork is created when reprogramming several nodes.

OTA modes:
- Unicast: Reprogram an specific node
- Multicast: Reprogram several nodes at the same time sending the program just once
- Broadcast: Reprogram the entire network sending the program just once

Topologies:
- Direct access: when the nodes are accessed in just one hop (no forwarding of the packets is needed). The protocols which work with this mode are: 802.15.4-2.4GHz, RF-868MHz and RF-900MHz.
- Multihop: when the nodes are accessed in two or more hops. In this mode some nodes have to forward the packets sent by the Gateway in order to reach the destination. The protocols which work with this mode are: ZigBee-2.4GHz and DigiMesh-2.4GHz.

Direct Access - UNICAST - 802.15.4 (2.4GHz)

Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified the channel to be used in the firmware transmission along with the Auth Key will be used by the node to validate if the OTA programmer is a trusted source. Both Gateway and node change the channel to the one specified; this way the OTA process do not cause interferences to the rest of the nodes in the network. In the case one or more packets get lost the GW will stop sending the firmware and will return the control to the user.

Information to be sent by the Gateway to the node to be reprogrammed:
- New channel
- Auth key
- New firmware
Direct Access - MULTICAST - 802.15.4 (2.4GHz)

In order to send the firmware in MULTICAST mode what we make is to create a subnetwork formed by the nodes to be reprogrammed and then we send the new firmware using the BROADCAST transmission in order to make each packet reach all nodes at the same time. In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

In order to create the subnetwork we will **change the channel** of the desired nodes before starting with the OTA process. For this reason we will send the New Channel to use along with the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. Both Gateway and nodes will change the channel to the one specified; this way the OTA process will **not to cause interferences to the rest of the nodes in the network**.

Information to be sent by the Gateway to the nodes to be reprogrammed:

- New channel
- Auth key
- New firmware
Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of channel as all the nodes in range are supposed to be reprogrammed.

In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

Information to be sent by the Gateway to the nodes to be reprogrammed:

- Auth key
- New firmware
Direct Access - UNICAST - 868MHz/900MHz

Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified the Auth Key which will be used by the node to validate if the OTA programmer is a trusted source. In this case the channel is not changed as the 868 and 900MHz modules do not support channel variation. In the case one or more packets get lost the GW will stop sending the firmware and will return the control to the user.

Information to be sent by the Gateway to the node to be reprogrammed:

- Auth key
- New firmware
Direct Access - MULTICAST - 868MHz/900MHz

In order to send the firmware in MULTICAST mode what we make is to create a subnetwork formed by the nodes to be reprogrammed and then we send the new firmware using the BROADCAST transmission in order to make each packet to reach all nodes at the same time. In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

In the 868 and 900MHz models we can not change the frequency band, so what we will make is to change the Encryption Key used in order to created a subnetwork with the nodes to be reprogrammed. This way the nodes which are not involved in the OTA process do not have to be discarding the packets sent in BROADCAST mode.

Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified a New Encryption Key along with the Old Encryption Key and the Auth Key to be used by the node to validate if the OTA programmer is a trusted source. Both Gateway and nodes change the Encryption Key to the new specified; this way the OTA process do not cause extra load to the rest of the nodes in the network.

The Old Encryption Key is sent in order the nodes can set it again after the OTA process finishes as can not be read by themselves from the radio module (it is a protected parameter which can not be accessed).

Information to be sent by the Gateway to the nodes to be reprogrammed:

- New Encryption Key
- Old Encryption Key
- Auth key
- New firmware
Direct Access - BROADCAST - 868MHz/900MHz

Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of Encryption Key as all the nodes in range are supposed to be reprogrammed.

In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

Information to be sent by the Gateway to the nodes to be reprogrammed:

- Auth key
- New firmware
Multihop Access - UNICAST - ZigBee & DigiMesh (2.4GHz)

Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified the Auth Key which will be used by the node to validate if the OTA programmer is a trusted source. In a Multihop network neither the channel nor the Encryption key can be changed as the information could not be delivered to the destination.

Information to be sent by the Gateway to the node to be reprogrammed:

- Auth key
- New firmware
Multihop Access - MULTICAST - ZigBee & DigiMesh (2.4GHz)

In order to send the firmware in MULTICAST mode what we make is to create a subnetwork formed by the nodes to be reprogrammed and then we send the new firmware using the BROADCAST transmission in order to make each packet to reach all nodes at the same time. In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

In a Multihop network neither the channel nor the Encryption key can be changed as the information could not be delivered to the destination. For this reason we change the Auth Key (used in the application layer) in order to create a subnetwork. This way the nodes which are not involved in the OTA process will not process the packets sent in BROADCAST mode.

Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified a New Auth Key along with the Old Auth Key to be used by the node to validate if the OTA programmer is a trusted source. Both Gateway and nodes change the Auth Key to the new specified; this way the OTA process do not cause extra process work to the rest of the nodes in the network.

Once the OTA process finishes the Old Auth Key is set again.

Information to be sent by the Gateway to the nodes to be reprogrammed:

- New Auth Key
- Old Auth Key (current)
- New firmware
Multihop Access - BROADCAST - ZigBee & DigiMesh (2.4GHz)

Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of the Auth Key as all nodes in the network are supposed to be reprogrammed.

In this mode no ACK packets are expected so we will have to wait to the end in order to see if all the nodes received the new firmware successfully.

Information to be sent by the Gateway to the nodes to be reprogrammed:

- Auth key
- New firmware

Multihop Access - BROADCAST - ZigBee & DigiMesh (2.4GHz)
5.4. OTA Shell

5.4.1. Acronyms

- **ID** = Node ID (16B), it is the identification name of each node. It is stored in the EEPROM memory.
- **PID** = Process ID (7B), The name and version of the program.
- **MAC** = MAC (8B) address of the node. It is the IEEE 802.15.4.
- **DATE** = Creation date of the current program.
- **ENC_KEY** = It is the key (16B) used by the AES 128b algorithm to encrypt the frames in the Link Layer (IEEE 802.15.4).
- **AUTH_KEY** = It is the key (8B) used in the Application level in order to authenticate the OTA programmer.

5.4.2. OTA Commands

In the “Setting the Environment” section you will find how to download and install the OTA application. Commands and available options are covered here.

When we run the main executable several options are displayed on the screen.

```
# ./otap
You have to specify one command
usage: otap
   -delete_program deletes a specified program
   -get_boot_list get the boot list of nodes
   -info_program shows info about hex file
   -scan_nodes scan for active nodes
   -send send firmware to nodes
   -start_new_program start specified program
   -version show version of the program
```

In the case a 64-bit computer is used, OTA commands need to be called as follows:

- ./otap64

5.4.3. Configuring the 802.15.4/ZigBee radio

Before we start working with the OTA functions we must provide the necessary parameters in order to make the communication possible from the Gateway to the nodes. All these parameters are located in an external file called “xbee.conf”

- **port**: Where the Waspmote Gateway is connected: /dev/ttyUSB0, /dev/ttyUSB1, COM1, COM2, etc.
- **auth key**: It is the key (8B) used in the Application level in order to authenticate the OTA programmer. By default is “LIBELIUM”.
- **panID**: It is the ID of the current network (2B). Only in the case of ZigBee protocol the panID is a 8-Byte identifier.
- **xbeeModel**: possible values are [802.15.4, ZB, DM, 868, 900].
- **channel**: It specifies the channel used by the current network (1Byte).
  - 802.15.4 = [0x0C - 0x17].
  - DigiMesh = [0x0C - 0x17].
  - ZB= not controlled. Chosen by the coordinator.
  - 868/900 = not changeable.
- **encryption**: [on, off] It specifies if the Encryption is activated or not.
- **encryptionKey**: It is the key (16B) used by the AES 128b algorithm to encrypt the frames in the Link Layer (IEEE 802.15.4).
- **discardedDataFile**: the name of the file where the non-OTA frames received by the Gateway will be stored. If not specified these frames will be directly deleted. The information is stored in ascii-hex format (one frame per line).
- **WaspmoteVersion**: indicates the Waspmote version. The possible values are: [11, 12].
5.4.4. Searching the node to upgrade

Prior to upgrade a node we have to check it is ready to receive the new program. To do so we have implemented the “Ready to OTA” state inside the main routine of Wasp mote. If the node is in this state when we send the “scan_nodes” query it will answer the next information: MAC address, Mote ID and actual Program ID.

```
#./otap -scan_nodes
--mac <000000000000000>     mac address, 16 hex characters, separated with commas
--mode <UNICAST|BROADCAST>   scanning mode
--time <seconds>             timeout seconds (optional, by default 10)
```

Let’s see some examples:

**Scan for any node and wait for 3 seconds after sending the request:**

```
#./otap -scan_nodes --mode BROADCAST --time 3
```

```
Total Nodes: 2 - Time elapsed 6s
0 - Node 0013a2004061097c - waspmote001 - prog001 - READY
1 - Node 0013a20041615623 - waspmote002 - prog001 - READY
```

**Search for a specific node:**

```
#./otap -scan_nodes --mode UNICAST --mac 0013a2004061097c --time 2
```

```
Total Nodes: 1 - Time elapsed 2s
0 - Node 0013a2004061097c - waspmote001 - prog001 - READY
```

**Note:** the node ID previously set by the Wasp mote program must set a 16-Byte name. If not, the ‘scan_nodes’ command will not work for that node.

**Note:** If two different nodes have the same node ID, when the ‘scan_nodes’ command is called only one of them will be shown in the screen.

5.4.5. Getting the bootlist

There is a special file called “BOOT.TXT” which contains the information of all the programs available in Wasp mote. Each time we add a new program a reference line is added. In order to get this information the “get_boot_list” option has been created. It returns a list with the PID and date of each of the programs available in the SD.

```
#./otap -get_boot_list
The command needs required options
usage: otap -scan_nodes
--mac <000000000000000>     mac address, 16 hex characters
--mode <UNICAST>             target mode
```

-27-
Examples of usage:

**Ask for the bootlist to a specific node:**

```
# ./otap -get_boot_list --mode UNICAST --mac 0013a2004061097c2
```

Bootlist Node 0013a2004061097c - length: 5

0 – PID: prog001 – Date:2011/03/31 14:06
1 – PID: prog002 – Date:2011/03/31 14:06
2 – PID: prog003 – Date:2011/04/04 10:33
3 – PID: prog004 – Date:2011/04/04 10:33
4 – PID: prog005 – Date:2011/04/04 10:33

**Ask for the bootlist of two different nodes:**

```
# ./otap -get_boot_list -mode UNICAST --mac 0013a2004061097c,0013a20041615623
```

Bootlist Node 0013a2004061097c - length: 5

0 – PID: prog001 – Date:2011/03/31 14:06
1 – PID: prog002 – Date:2011/03/31 14:06
2 – PID: prog003 – Date:2011/04/04 10:33
3 – PID: prog004 – Date:2011/04/04 10:33
4 – PID: prog005 – Date:2011/04/04 10:33

Bootlist Node 0013a20041615623 - length: 5

0 – PID: progr11 – Date:2011/03/31 14:06
1 – PID: progr12 – Date:2011/03/31 14:06
2 – PID: progr13 – Date:2011/04/04 10:33
3 – PID: progr14 – Date:2011/04/04 10:33
4 – PID: progr15 – Date:2011/04/04 10:33

**Note:** If several MAC’s are provided the command is executed sequentially in the UNICAST mode in order to make sure that packets are not lost.

### 5.4.6. Preparing the Delivery

Before sending the program we can extract some information in order to see how many packets and how much time (estimated) it will take to make the upload of the program. To do so we can execute the ‘info_program’ option:

```
# ./otap -info_program --file PROG001.hex
```

Name: PROG001.hex
Date: Wed Apr 27 13:24:18 CEST 2011
Size: 2100 bytes
Packets to send: 17
Estimated time: 3.77 seconds
5.4.7. Sending the program

Once we have selected the node (or nodes) to upgrade, we will send the desired firmware.

We can specify how many times each packet will be sent with the 'delivery' option. By default, in the UNICAST mode the value is 1; in the MULTICAST and BROADCAST modes the value is 2, so each packet will be sent twice.

The total delivery time increases by the same factor that the number of packets and deliveries increases:

\[ \text{Time} = \text{time\_per\_packet} \times \text{number\_packets} \times \text{deliveries} \]

Increasing the deliveries will make the total time increase but will also make the process more robust and it will decrease the probability that a node fails when receiving the packets.

By default the data is sent is BINARY format as the total size is lower (half of the ASCII format). Do not use the ASCII option unless your motes have it specifically activated.

The total time depends on the kind of radio used (802.15.4, ZigBee, DigiMesh, etc) and if we activate the encryption or not. The payload per packet may vary from 62 Bytes (DigiMesh + encryption) to 92 Bytes (802.15.4 + no encryption). In average, we can send 4.5 packets per second so the bandwidth will be in the range 279Bytes/s - 414Bytes/s. This means a simple program (1KB = ~17 packets) will be sent from 2.5 to 3.5s and the most complex one (62KB = ~1000 packets) from 149 to 222s depending on the configuration used.

```
#./otap -send
The command needs required options
usage: otap -send
   --deliveries <times>   number of times each packet is sent
   --file <firmware_file.hex>   file containing the firmware
   --mac <0000000000000000>   mac address, 16 hex characters, separated by commas
   --mode <UNICAST|MULTICAST|BROADCAST>   transmission mode
   --new_authkey <LIBELIUM>   8 ascii characters
   --new_channel <00|0x00>   channel in decimal or hexadecimal format
   --new_enckey <1234567890123456>   1 ascii characters
   --pid <PROGRAM_1>    program identifier, 7 characters in length
```

Examples:

**Direct Access UNICAST - 802.15.4 + Change of channel**

```
# ./otap -send --file PROG001.hex --mac 0013a2004061097c --mode UNICAST --new_channel 0x0C --deliveries 1 --pid prog001
```

---
Name: PROG001.hex
Date: Tue Apr 05 13:22:41 CEST 2011
Size: 1050 bytes
Packets to send: 17
Estimated time: 3.77 seconds
---

Node 0013a2004061097c
[Sending] - 100% - 4 seconds elapsed
Finished. Number of packets sent: 17
Node response: PROGRAM RECEIVED OK
---
OTA with 802.15.4/ZigBee

**Direct Access MULTICAST - 802.15.4 + Change of channel**

```bash
./otap -send --file PROG001.hex --mac 0013a2004061097c,0013a20041615623 --mode MULTICAST --channel 0x0C --deliveries 2 --pid prog001
```

Name: PROG001.hex
Date: Tue Apr 05 13:22:41 CEST 2011
Size: 1050 bytes
Packets to send: 17
Estimated time: 3.77 seconds

Node 0013a2004061097c
[Sending] - 100% - 4 seconds elapsed
Finished. Number of packets sent: 17
Node response: PROGRAM RECEIVED OK

Node 0013a20041615623
[Sending] - 100% - 4 seconds elapsed
Finished. Number of packets sent: 17
Node response: PROGRAM RECEIVED OK

**Direct Access BROADCAST - 802.15.4**

```bash
./otap -send --file PROG001.hex --mode BROADCAST --deliveries 2 --pid prog001
```

Name: PROG001.hex
Date: Tue Apr 05 13:22:41 CEST 2011
Size: 1050 bytes
Packets to send: 17
Estimated time: 3.77 seconds

Node: all
[Sending] - 100% - 4 seconds elapsed
Finished. Number of packets sent: 17
Node response: PROGRAM RECEIVED OK

Node 0013a2004061097c: PROGRAM RECEIVED OK
Node 0013a20041615623: PROGRAM RECEIVED OK
Node 0013a20041744564: PROGRAM RECEIVED OK

**Direct Access UNICAST - 868/900MHz + Change of Encryption Key**

```bash
./otap -send --file PROG001.hex --mac 0013a2004061097c --mode UNICAST --new_enckey secretPassw12345 --deliveries 1 --pid prog001
```

**Direct Access MULTICAST - 868/900MHz + Change of Encryption Key**

```bash
./otap -send --file PROG001.hex --mac 0013a2004061097c,0013a20041615623 --mode MULTICAST --new_enckey secretPassw12345 --deliveries 2 --pid prog001
```

**Direct Access BROADCAST - 868/900MHz**

```bash
./otap -send --file PROG001.hex --mode BROADCAST --deliveries 2 --pid prog001
```

**Multihop UNICAST - ZigBee & DigiMesh (2.4GHz) + Change of Auth Key**

```bash
./otap -send --file PROG001.hex --mac 0013a2004061097c --mode UNICAST --new_authkey secret00 --deliveries 1 --pid prog001
```
Multihop MULTICAST - ZigBee & DigiMesh (2.4GHz) + Change of Auth Key

```bash
./otap -send --file PROG001.hex --mac 0013a2004061097c,0013a20041615623 --mode MULTICAST
--new_authkey secret00 --deliveries 2 --pid prog001
```

Multihop BROADCAST - ZigBee & DigiMesh (2.4GHz)

```bash
./otap -send --file PROG001.hex --mode BROADCAST --deliveries 2 --pid prog001
```

5.4.8. Starting a new program

After sending a new program it does not start automatically. We have to send the "start_new_program" command in order to store it in the Flash memory and start executing it.

If we try to start the same program which is currently running it will make Waspmote to reboot and no change will be made.

```bash
./otap -start_new_program
usage: otap -start_new_program
    --mac <000000000000000> mac address, 16 hex characters
    --macs_file <list.txt> file containing list of macs (if MULTICAST enabled)
    --mode <UNICAST|MULTICAST|BROADCAST> star mode
    --pid <program_name> the program identifier (7 characters length max)
```

Examples:

**Start the program “prog001” in a specific node:**

```bash
./otap -start_new_program --mac 0013a2004061097c --mode UNICAST --pid prog001
```

Waiting confirmation...

Node 0013a2004061097c  - STARTING NEW FIRMWARE
Node 0013a2004061097c  - NEW PROGRAM RUNNING

**Start the program “prog003” in all the nodes of the network:**

```bash
./otap -start_new_program --mode BROADCAST --pid prog003
```

Waiting confirmation...

Node 0013a2004061097c  - STARTING NEW FIRMWARE
Node 0013a20041615623  - STARTING NEW FIRMWARE
Node 0013a20041744564  - STARTING NEW FIRMWARE
Node 0013a2004061097c  - prog003 - NEW PROGRAM RUNNING
Node 0013a20041615623  - prog003 - NEW PROGRAM RUNNING
Node 0013a20041744564  - prog003 - NEW PROGRAM RUNNING
5.4.9. Delete a Program

Sometimes it can be interesting to delete a program if we do not want to use it any more or if we want to send another one with the same name (although it is recommendable to use always different names for each program version (prog001, prog002, etc)).

# ./otap -delete_program

The command needs required options

usage: otap -delete_program

  --mac <0000000000000000>  mac address, 16 hex characters
  --macs_file <list.txt>  file containing list of macs
   (if MULTICAST enabled)

  --mode <UNICAST|MULTICAST|BROADCAST>   start mode

  --pid <program_name>  the program identifier
   (7 characters)

Usage examples:

Delete the program ‘prog002’ from a specific node.

# ./otap -delete_program --mode UNICAST --mac 0013a2004061097c --pid prog002

Delete the program ‘prog003’ from all the nodes in the network.

# ./otap -delete_program --mode BROADCAST --pid prog003
5.4.10. Responses

After executing the OTA commands some information messages are sent by the nodes in order to inform how the process has gone.

**Success Responses:**

**READY**

It is sent by the nodes which are ready to be reprogrammed when we call the “scan_nodes” command. If a node is sleeping or it is not in the OTA slot time no message will be received.

**PROGRAM RECEIVED OK**

It is sent by the nodes when we call the “send” command and the program has been sent successfully received.

**START WITH FIRMWARE OK**

It is sent by the nodes before rebooting when we call the “start_new_program” command.

**NEW PROGRAM RUNNING**

It is sent by the nodes the first time the new program starts. It is generated after calling the “send” command.

**RESTARTING**

It is sent by the nodes before rebooting when OTA is not performed.

**Failure Responses:**

**PROGRAM RECEIVED ERROR**

It is sent by the nodes when we call the “send” command and one or more packets are lost.

**PREVIOUS PROGRAM RUNNING**

It is sent by a node when tries to start a new program the first time and fails. It is produced after executing the “start_new_program” command. The previous stable program is restored and this message sent the OTA programmer.

**START WITH FIRMWARE ERROR**

It is sent by the nodes when we call the “start_new_program” command with an invalid PID.
5.5. Setting the environment

5.5.1. Meshlium

Meshlium works as the Gateway of the Waspmote Sensor Networks. It reads the sensor frames coming from the nodes and store them in its internal data base and in external cloud systems located on the Internet. The frames coming from Waspmote are normally received by the 802.15.4/ZigBee radio and sent to the Internet using Ethernet, WiFi and 3G interfaces.

Meshlium allows us even to detect iPhone and Android devices and in general any device which works with WiFi or Bluetooth interfaces. The idea is to be able to measure the amount of people and cars which are present in a certain point at a specific time, allowing the study of the evolution of the traffic congestion of pedestrians and vehicles.

![Meshlium Router](image)

The environment needed to execute OTA Shell comes already preinstalled in Meshlium. You just need to download the last version of the OTA Shell from the Development section:

http://www.libelium.com/development/waspmote/sdk_applications

First of all stop the default 802.15.4 / ZigBee daemon:

```
#sensorParserD.sh stop
```

Or, for Meshlium versions earlier than Meshlium Xtreme 3.0:

```
#/etc/init.d/ZigbeeScanD.sh stop
```

**Important:** When executing the OTA-Shell application in Meshlium the ZigBee radio will not receive the sensor frames from the network. If you want to perform both processes at the same time two gateways must be used.
Extract all the files of the otap.tar.gz file executing the following in the directory where you have the compressed file:

```
# tar -xzf otap.tar.gz
```

Modify the xbee.conf file to put the previously seen path for the serial port in the right line:

```
port = /dev/ttyS0
```

Now you can run the otap program simply executing ./otap

### 5.5.2. Linux

You need to have java installed in order to execute the program. If you don't java installed install it from the repository using the following command:

```
# aptitude install sun-java6-jre
```

(or you can also download it from java.com webpage and install it)

You may have to run the command `update-alternatives` in order to select the right Java packet (sun-java6):

```
# update-alternatives --config java
```

Plug in the Wasp mote Gateway in a USB port. You need to know what path has assigned the system, so execute the following in a command line terminal:

```
# ls -l /dev/ttyUSB*
```

```
crw-rw---- 1 root dialout 188, 0 may 4 12:17 /dev/ttyUSB0
```

Extract all the files of the otap.tar.gz file executing the following in the directory where you have the compressed file:

```
# tar -xzf otap.tar.gz
```

Modify the xbee.conf file to put the previously seen path for the serial port in the right line:

```
port = /dev/ttyUSB0
```

Now you can run the otap program simply executing ./otap (or ./otap64 if you've a 64 bits OS):
5.5.3. Windows

- Plug in the pc the usb-gateway. If you already have the ftdi drivers installed the system will automatically recognize the device and it will be ready to use. If a window appears asking you for the drivers you can download them from the ftdi webpage.
- Once the device is well installed you need to know what port name has assigned Windows to it. Make right click on you pc, select the Hardware tab and then in the Device Manager button.

- The Waspmote Gateway will appear as a USB Serial port. Take note of the name Windows has assigned to it (in the image example it is COM3).
• Make sure you have java installed or download and install it from java.com webpage.
• Extract all the files from the otap.tar.gz archive using a decompress program (like 7zip)
• Edit the xbee.conf to change the port configuration with the port name previously seen:

```
port = COM5
```

• Now open a command line interpreter going to Start > Run > cmd.exe

• Navigate to the folder where you have uncompressed the otap.tar.gz file and you’re ready to use the otap program typing otap.bat (or otap64.bat if you’ve a 64 bits system):
5.5.4. Mac OSX

- Mac comes with java jre installed by default so you don’t have to install anything extra.
- First you have to plug-in the Waspmote Gateway in your mac machine. Then a screen showing the following can appear:

```
A new network interface has been detected.
The "FT232R USB UART" network interface has not been set up. To set up this interface, use Network Preferences.

Cancel        Network Preferences…
```

Press “Cancel” in order to ignore it.

- Now open a Terminal window: Finder > Go > Utilities > Terminal:

```
Last login: Wed May 4 16:57:33 on ttys008
libelium@dev~$ $
```

- Navigate to the folder where you have your otap.tar.gz archive and uncompress it:

```
$ tar -zxf otap.tar.gz
```

- Enter in the otap folder:

```
~$ cd otap
~/otap$
```
Before start using the program you’ve to modify the xbee.conf file with the appropriate serial port parameter, so execute the following command to see what usb-to-serial adapters are connected to the system:

```bash
~/otap$ ls /dev/tty.usbserial*
/dev/tty.usbserial-A8003LP0
```

Put the showed path in the ‘xbee.conf’ file in the following line:

```bash
port = /dev/tty.usbserial-A8003LP0
```

And now you’re ready to use the otap program executing ./otap (or ./otap64 if you have a 64 bits OS):

```
libelium@dev:/otap$ cd otap
libelium@dev:/otap$ ./otap
You have to specify one command
usage: otap
       -delete_program deletes a specified program
       -get_boot_list get the boot list of nodes
       -info_program shows info about hex file
       -reset reset waspmote
       -scan_nodes scan for active nodes
       -send send firmware to nodes
       -start_new_program start specified program
       -version show version of the program
libelium@dev:/otap$
```
5.6. Hardware Setup

5.6.1. Important dates and versions

Wasp mote v12 is ready to perform OTA operations.

Wasp motes v11 bought after the 17/01/2011 come with the Bootloader ready to run OTA. In order to have it new feature working, make sure you have the API version 0.18 or older installed. If you bought your Wasp mote kits before that date and you want them to run OTA please contact our Sales Department at: [http://www.libelium.com/company/contact](http://www.libelium.com/company/contact)

5.6.2. Wasp mote Setup

For Wasp mote v11, make sure that you are executing the API version 0.18 or older. Any API version will work for Wasp mote v12. Download the last version of the API at:


As OTA is based on XBee, your program must turn it on and wait for packets somewhere in the code.

OTA requires an SD card to work properly, so an SD card must be placed in the SD socket before trying to re-program a Wasp mote.

The ‘Auth Key’ must be set in each Wasp mote to support OTA. By default, this 8-byte key is set as ‘LIBELIUM’ and it is stored in the EEPROM. Utilities library provides the function to set it:

```java
Utils.setAuthKey("LIBELIUM");
```

The ‘Mote ID’ is an identifier to distinguish every Wasp mote in a unique way. It is stored in the EEPROM and by default it is set as ‘WASPMOTE00000001’. Utilities library provides the function to set it:

```java
Utils.setID("WASPMOTE00000001");
```

There are 2 important programming restrictions using OTA (with API 0.18 or older):

1. The OTA operation requires complex communications between the Gateway and each one of the Wasp motes in the network. This process is done through special XBee frames which are identified with special frame types. The OTA communication is transparent for the user, but it is mandatory to avoid creating frames with one of the following frame types using the WaspFrame class (please check the Wasp mote Frame Programming Guide):

2. OTA uses space in the EEPROM of each Wasp mote for storing flags, IDs, MACs or code. If it is planned to operate OTA, the user needs to avoid to write in the EEPROM addresses from 0 to 162. As a result, EEPROM[163] is the first available byte to write.

An example of a program that sets the ‘Auth Key’, the ‘Mote ID’ and that is prepared to be re-programmed OTA is shown below:

```java
#include <WaspXBee802.h>
#define key_access "LIBELIUM"
#define id_mote "WASPMOTE00000001"

void setup()
{
    // Write Authentication Key to EEPROM memory
    Utils.setAuthKey(key_access);

    // Write Mote ID to EEPROM memory
    Utils.setID(id_mote);

    // Initialize XBee module
    xbee802.ON();
}
```
OTA with 802.15.4/ZigBee

// CheckNewProgram is mandatory in every OTA program
xbee802.checkNewProgram();
}

void loop()
{
    // Check if new data is available
    if( xbee802.available() )
    {
        xbee802.treatData();
        // Keep inside this loop while a new program is being received
        while( xbee802.programming_ON && !xbee802.checkOtapTimeout() )
        {
            if( xbee802.available() )
            {
                xbee802.treatData();
            }
        }
    }
}
5.6.3. Waspmote Gateway Setup

The Waspmote GW is going to be the node that sends the new firmware to Waspmote. The XBee placed on this GW must be configured at 38400 (BD=5) and API mode 1 (AP=1).

To configure the XBee module you can use X-CTU (manufacturer's software to change XBee parameters).

First of all, you have to connect your Waspmote GW to the computer and select the port where it has been connected to.

Then you select the baudrate and API mode at which XBee is configured.

Go to 'Modem Configuration' and press the 'Read' button. Once you have read the parameters you have to change the parameters BD (BD=5) and AP (AP=1).
Press the 'Write' button and you will have your XBee module configured to be used in OTA.
6. OTA with 3G/GPRS/WiFi via FTP

It is possible to update the Wasp mote's program using Over The Air Programming and the following modules: 3G, GPRS or WiFi module.

6.1. Procedure

The Wasp mote reprogramming is done using an FTP server and an FTP client which is Wasp mote itself. The FTP server can be configured by Meshlium. Otherwise, the user will have to setup an FTP server with the settings described in this guide.

There are two basic steps involved in OTA procedure:

- **Step 1**: Wasp mote requests a special text file which gives information about the program to update: program name, version, size, etc.
- **Step 2**: If the information given is correct, Wasp mote queries the FTP server for a new program binary file and it updates its flash memory in order to run the new program.
6.2. Setting the FTP server configuration

The FTP server that Waspmote connects to needs a specific configuration so as to OTA work properly. There are two ways to set up the FTP server:

- Extern user’s FTP server
- Meshlium FTP server

6.2.1. Extern FTP server setup

There is a special text file called UPGRADE.TXT which defines several fields. This file must be located in the FTP Server root directory. Each field is defined by a specific label as shown below. Each line of the text file must end with the corresponding line end character: ‘\n’.

There are labels defined for each parameter:

- **FILE**: identifies the name of the binary to be downloaded (it is a 7-character name)
- **PATH**: identifies the path inside the FTP server where the file is found.
- **SIZE**: identifies the size of the binary file. It must be written as the number of bytes.
- **VERSION**: identifies the program version. It must be defined as a 1-unsigned-byte number (range: from 0 to 255). This label must be used by the user so as to validate new program versions when the file name specified in FILE does not change. Thus, Waspmote will know if the same program has changed its version and if it is necessary to download it or not.

There is a restriction regarding the filename: it must be defined by 7 characters given by the Waspmote IDE when it generates the binary file. It is important not to change the file name. If no OTA file needs to be downloaded, then the FILE field must be filled with "NO_FILE" pattern. Thus, Waspmote will know that no downloading process is required. If no UPGRADE.TXT file is found in the FTP server, the OTA process will be called off as well.

Example 1: No binary file has to be downloaded

```
FILE: NO_FILE
PATH: 
SIZE: 
VERSION: 
```

Example 2: Filename and Path specified in the text file. Notice that no ‘/’ character appears at the end of the path.

```
FILE: PROG001
PATH: /demo/test
SIZE: 46999
VERSION: 10
```

Example 3: Filename and FTP Server’s Root Directory are specified in the text file. Notice that root directory is specified as a unique ‘/’ character.

```
FILE: PROG001
PATH: /
SIZE: 56999
VERSION: 11
```

During Waspmote program execution, several cases can occur when the mote requests OTA via 3G/GPRS/WiFi module:

- If there is no UPGRADE.TXT file or it is not found, Waspmote will not perform any OTA process and will carry on with the current program.
- If there is an UPGRADE.TXT file and NO_FILE is specified as filename, then Waspmote will carry on with the current program.
- If every label is correctly defined, Waspmote will attempt to download and update the new program automatically.
6.2.2. Meshlium FTP server setup

The other possibility is to setup the FTP server using Meshlium.

Meshlium works as the Gateway of the Waspmote Sensor Networks. It reads the sensor frames coming from the nodes and stores them in its internal database and in external cloud systems located on the Internet.

The frames coming from Waspmote are normally received by the 802.15.4/ZigBee radio and sent to the Internet using Ethernet, WiFi and 3G interfaces.

Meshlium allows us even to detect iPhone and Android devices and in general any device which works with WiFi or Bluetooth interfaces. The idea is to be able to measure the amount of people and cars which are present in a certain point at a specific time, allowing the study of the evolution of the traffic congestion of pedestrians and vehicles.

The Manager System is a web interface which comes with Meshlium. It allows you to control all the interfaces and system options in a secure, easy and quick way. For more information:


There is a plug-in for Meshlium’s Manager System which permits to set up the FTP server automatically. Thus, it is easier to prepare the binary files to be downloaded by Waspmote. This plug-in is already installed in Manager System version 3.0.7 or more. Please check the Technical Guide in order to update the Manager System to its last version.

OTA via GPRS, 3G or WiFi is supported by Meshlium Xtreme. If you have a previous version, you need to upgrade it; please contact the Sales Department for more information:

http://www.libelium.com/company/contact

Besides, a default user is configured in Meshlium FTP Server with the following settings:

user: ota
password: libelium
This user directly connects to the following path in Meshlium’s system directory where the application creates all the binary and UPGRADE.TXT files:

/mnt/user/ota

The OTA-FTP application can be found inside the Manager System -> Sensor Networks -> OTA-FTP

Firstly, there are three possibilities to be chosen:

- Select NO_FILE to inform Waspmote that no OTA is necessary
- Select a new file generated by Waspmote-IDE so as to update the Waspmote’s program.
- Select an existing binary if the user needs to update to an older program. The files are stored in the following path: /mnt/user/ota

Secondly, the program version is always set by the user before generating the new UPGRADE.TXT file. There is a specific input to indicate the program version. It must be defined as a 1-unsigned-byte number (range: from 0 to 255).

Finally, there is a button to generate the new UPGRADE.TXT file.

Once these steps have been followed, the binary file and the proper UPGRADE.TXT file are ready for the Waspmote devices deployed which try to perform OTA via FTP transmission. This file is shown in the last window of the application representing the actual binary prepared for OTA.
6.3. Waspmote basic sketch

Every Waspmote sketch which provides OTA via FTP must have the following appearance:

```cpp
// include and variable definitions

// setup function
void setup()
{
    // 1. Check if the program has been programmed successfully
    // 2. User code
    /*
    * Put your setup code here, to run once
    */
}

void loop()
{
    // 3. User code
    /*
    * Put your main code here, to run repeatedly
    */

    // 4. OTA request function
}
```

Step by step:

1. The first checking is done by calling a specific function defined in Utilities library. This function must be called at the beginning of the code. This function returns three possible values. This allows the user to perform an action when a reprogramming has taken place.

   ```cpp
   {
   Utils.checkNewProgram();
   }
   ```

Returns

0: Reprogramming error
1: Reprogramming ok
2: Normal starting (no OTA process)

2. The user must fill this point with the code which is executed just once

3. The user must fill this point with the code which is executed every loop repeatedly.

4. The OTA request function depends on the module used. This function should be called when the user considers it so as to keep battery, i.e. once a day or once a week.
6.4. OTA via GPRS and GPRS+GPS module

When using a GPRS module, Waspmote connects to the nearest Mobile Cell and follows the OTA procedure.

The GPRS_Pro and GPRS+GPS libraries implement the request function indicating four input parameters: FTP address, port, user and password. So a function example for GPRS_Pro is:

```c
{  
    GPRS_Pro.requestOTA("ftp_address","ftp_port","fpt_user","ftp_password");  
}
```

This function returns a int8_t variable which indicates the error found:

- 1: if success
- -2: if error
- -4: if error setting the type of internet connection,
- -5: if error setting the apn
- -6: if error setting the user name
- -7: if error setting the password
- -8: if error saving the configuration
- -9: if error opening connection with the GPRS provider
- -10: error downloading OTA version file
- -11: if error getting the IP address
- -12: if error setting the FTP/HTTP ID (-50 if CME error available)
- -13: if error setting the FTP mode (-51 if CME error available)
-14: if error setting the FTP type (-52 if CME error available)
-15: if error setting the FTP server (-53 if CME error available)
-16: if error setting the FTP port (-54 if CME error available)
-17: if error setting the user name (-55 if CME error available)
-18: if error setting the password (-56 if CME error available)
-21: if error setting the file name in the FTP server (-57 if CME error available)
-22: if error setting the path of the file in the FTP server (-58 if CME error available)
-23: if error opening the FTP session (-59 if CME error available)
-24: if error starting the SD
-25: if error creating the file
-26: error requesting data to the FTP (-60 if CME error available)
-27: if error saving data into the SD
-28: if error requesting more data to the FTP (-61 if CME error available)
-30: setting the file name in the FTP to get the file size (-62 if CME error available)
-31: setting the path in the FTP to get the file size (-63 if CME error available)
-32: if error getting the file size (-64 if CME error available)
-65: if FTP is busy
-66: if there isn't FILE tag
-67: if there isn't PATH tag
-68: if there isn't VERSION tag
-69: if OTA is not necessary
-70: if OTA files are the same program version
-71: if error opening connection with the GPRS provider
-72: error downloading OTA file
-73: if error getting the IP address
-74: if error setting the FTP/HTTP ID (-112 if CME error available)
-75: if error setting the FTP mode (-113 if CME error available)
-76: if error setting the FTP type (-114 if CME error available)
-77: if error setting the FTP server (-115 if CME error available)
-78: if error setting the FTP port (-116 if CME error available)
-79: if error setting the user name (-117 if CME error available)
-80: if error setting the password (-118 if CME error available)
-83: if error setting the file name in the FTP server (-119 if CME error available)
OTA with 3G/GPRS/WiFi via FTP

-84: if error setting the path of the file in the FTP server (-120 if CME error available)
-85: if error opening the FTP session (-121 if CME error available)
-86: if error starting the SD
-87: if error creating the file
-88: error requesting data to the FTP (-122 if CME error available)
-89: if error saving data into the SD
-90: if error requesting more data to the FTP (-123 if CME error available)
-92: setting the file name in the FTP to get the file size (-124 if CME error available)
-93: setting the path in the FTP to get the file size (-125 if CME error available)
-94: if error getting the file size (-126 if CME error available)
-127: if FTP is busy

Before calling the request function, it is necessary to switch on the GPRS module and connect to a GPRS network.

The following example shows how to perform OTA via GPRS_Pro module. Keep in mind that the user has to change the APN settings to their own (constants \texttt{AT\_GPRS\_APN}, \texttt{AT\_GPRS\_LOGIN} and \texttt{AT\_GPRS\_PASSW} into the file ”WaspGPRS\_Pro\_core.h”)
and must keep the \texttt{OTA\_FUSE} with ‘1’:

\texttt{www.libelium.com/development/waspmote/examples/ota-06-ota-via-gprs-pro-module}

\textbf{Note:} It is absolutely necessary to use the proper firmware in the GPRS module so as to perform the OTA process. The correct firmware version is 1137B01SIM900M64\_ST\_ENHANCE. You can check the firmware version executing the following example:


\textbf{Note:} It is absolutely necessary to use the proper firmware in the GPRS+GPS module so as to perform the OTA process. The correct firmware version is 1137B03SIM908M64\_ST\_ENHANCE. You can check the firmware version executing the following example:


\textbf{Note:} In the case you want to upgrade the firmware to a valid version, please ask Libelium Sales Department for more information.
6.5. OTA via 3G module

When using the 3G module, Waspmote connects to the nearest Mobile Cell and follows the OTA procedure.

![OTA with 3G module](image)

The 3G library implements the request function indicating four input parameters: FTP address, port, user and password. So a function example is:

```
{ 
   _3G.requestOTA("ftp_address","ftp_port","ftp_user","ftp_password"); 
}
```

This function returns a int8_t variable which indicates the error found:

- 1: on success
- 2: if error setting the connection parameters (APN)
- 3: if error setting the FTP server (‘-13’ if CME error available)
- 4: if error setting the FTP port (‘-14’ if CME error available)
- 5: if error setting the FTP mode (‘-15’ if CME error available)
- 6: if error setting the FTP type (‘-16’ if CME error available)
- 7: if error setting the user name (‘-17’ if CME error available)
- 8: if error setting the FTP password (‘-18’ if CME error available)
- 19: if error downloading the OTA version file
- 21: if error with CME code (FTP error) downloading the OTA version file
- 22: if error sending the OTA version file from 3G module to Waspmote's SD
- 23: if there isn't FILE tag
- 24: if there isn't PATH tag
OTA with 3G/GPRS/WiFi via FTP

-25: if there isn’t VERSION tag
-26: if OTA is not necessary
-27: if OTA files are the same program version
-28: if error downloading the OTA file
-31: if error with CME code (FTP error) downloading the OTA file
-32: if error sending the OTA file from 3G module to Waspmote’s SD

Before calling the request function, it is necessary to switch on the 3G module and connect to a 3G or GPRS network.

The following example shows how to perform OTA via 3G module. Keep in mind that the user has to change the APN settings to their own (constants _3G_APN, _3G_LOGIN and _3G_PASSW into the file "Wasp3G.cpp") and must keep the OTA_FUSE with ‘1’:

www.libelium.com/development/waspmote/examples/ota-07-ota-via-3g-module

6.6. OTA via WiFi module

When using the WiFi module, Waspmote connects to the Access Point set up by the user and follows the OTA procedure.

![OTA with 3G module](image)

**Note:** Be aware of the decrease in success percentage depending on the devices the user sets up in a WiFi network working together.

**Note:** OTA via WiFi feature was released on June 2013; modules purchased before that date may not support this feature.

**Note:** In the case you want to upgrade the firmware to a valid version, please ask Libelium Sales Department for more information.
The WiFi library implements the request function indicating no input parameters. It is supposed that the user has previously set the correct settings (FTP address, user, password and port). So the function is as simple as:

```cpp
WIFI.requestOTA();
```

This function returns an int8_t variable which indicates the error found:

-1: If error downloading UPGRADE.TXT
-2: If filename in UPGRADE.TXT is not a 7-byte name
-3: If no FILE label is found in UPGRADE.TXT
-4: If NO_FILE is defined as FILE in UPGRADE.TXT
-5: If no PATH label is found in UPGRADE.TXT
-6: If no SIZE label is found in UPGRADE.TXT
-7: If no VERSION label is found in UPGRADE.TXT
-8: If version indicated in UPGRADE.TXT is lower/equal to Waspmote's version
-9: If file size does not match the indicated in UPGRADE.TXT
-10: If error downloading binary file

Before calling the request function, it is necessary to switch on the WiFi module and join a network.

The following example shows how to perform OTA via WiFi module. Keep in mind that the user has to change the AP settings to their own. Besides, the FTP server settings must be changed in the case an extern FTP server is used. If not, the default Meshlium's parameters are set, but the Meshlium's network address must be specified as the real one:

www.libelium.com/development/waspmote/examples/ota-08-ota-via-wifi-module

**Note:** It is absolutely necessary to use the proper firmware in the WiFi module so as to perform the OTA process. The correct firmware versions are: &lt;2.32&gt; and &lt;2.36&gt;. You can check the firmware version executing the following example:

www.libelium.com/development/waspmote/examples/wifi-03-status/
7. Documentation changelog

From v4.3 to v4.4

- Added references to the GPRS+GPS module
- Added advice about Meshlium Sensor Parser, nodes password length, etc.

From v4.2 to v4.3

- Deleted references to Reset OTA

From v4.1 to v4.2

- Added the implementation of OTA via 3G/GPRS/WiFi modules.

From v4.0 to v4.1

- Added references to 3G/GPRS boarditaribus te et; Catiline nulla delius.