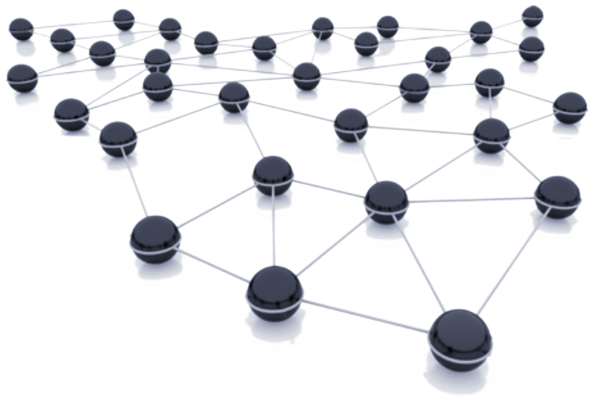
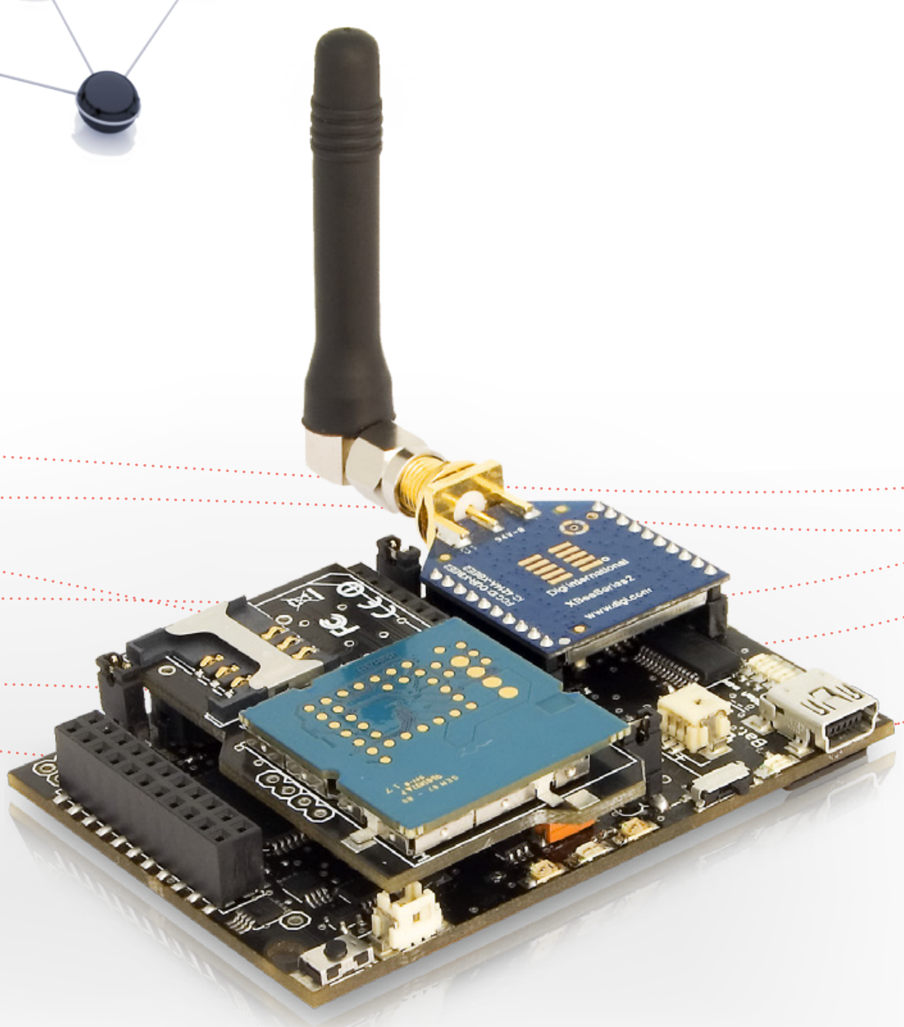
Over the Air Programming with 802.15.4 and ZigBee

Laying the groundwork





Document version: v0.9 - 07/2012

INDEX

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 reprogrammation 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.

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

Acknowledgement:
Libelium wants to thank to the Network Planning and Mobile Communications Group from the University of Cantabria on behalf of the Smart Santander project who contributed to the testing and performance improvement of the OTA technology.

2. Benefits

Libelium OTA Benefits:

- Enables the upgrade or change of firmware versions without physical access.
- Enables to recover to any sensor node which gets stuck.
- Discover nodes in the area just sending a broadcast discovery query.
- Upload new firmware in just a couple of 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.

3. Concepts

The idea is simple. When the programmer (normally the Gateway) sends a new program it is stored in the SD card. A second command "star_new_firmware" is needed in order to make them start. Then, the nodes copy the program from the SD card to the Flash memory and start the new program.

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 (ZigBee, Bluetooth, GPRS) and frequencies (2.4GHz, 868MHz, 900MHz) being capable of getting links up to 12km. It counts with sleep modes of few µA which allow to save battery when it is not transmitting . More than 50 sensors already available and a complete open source IDE (API libraries + compiler) made really easy to start working with the platform.

Note: to run programs which make use of the hibernate mode, it is needed to take the hibernate jumper out, so this feature cannot be used with OTA. Besides, if a new code is uploaded when the programming jumper is not set, then Waspmote could be left in an unstable mode. We do not advise to use hibernate mode with OTA; however, sleep or deepsleep modes can be performed.

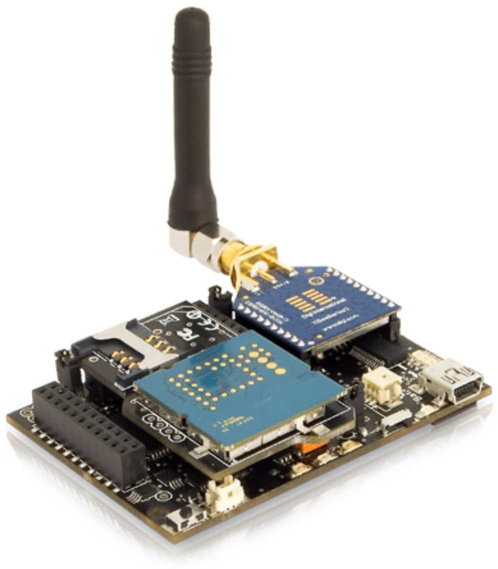


Figure 1: Waspmote Sensorial Device

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).
- DigiMesh - 2.4GHz (Worldwide).
- RF - 868MHz (Europe).
- RF - 900MHz (US, Canada, Australia).

Storage System:

Once we have sent the program to Waspmote 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 Waspmote'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 Waspmote 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:

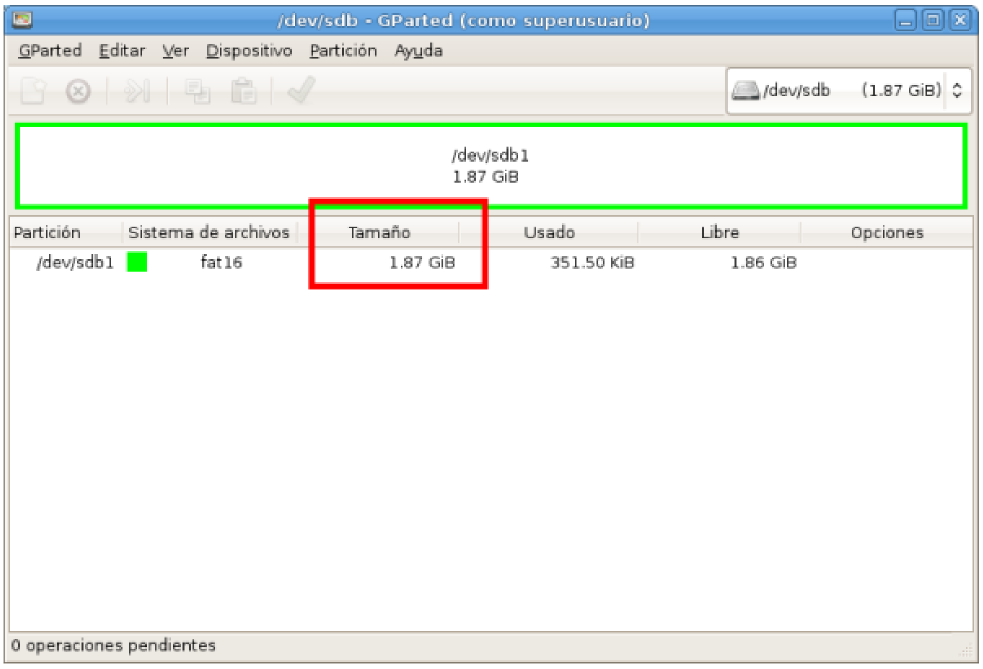
The OTA-Shell application can be used in Windows, Linux and MacOS. It allows to control in a quick and powerful way all the options available in OTA. If you are using Meshlium as the Gateway of the network, the OTA-Shell environment comes already preinstalled and ready to use. This is the recommended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

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

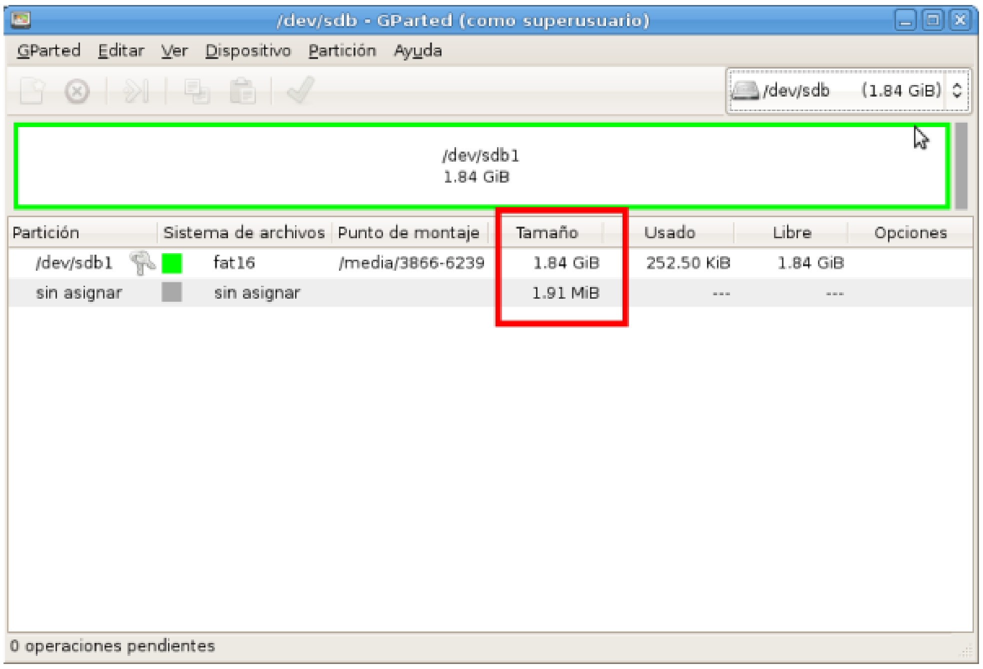
It necessary to install a program like 'gparted' or similar. When checking the SD card you can see two possibilities. If the capacity is 1.87 GB, the SD card is correct:


/dev/sdb - GParted (como superusuario)
GParted Editar Ver Dispositivo Partición Ayuda
/dev/sdb (1.87 GiB)
/dev/sdb1
1.87 GiB

Partición	Sistema de archivos	Tamaño	Usado	Libre	Opciones
/dev/sdb1 🟩	fat16	1.87 GiB	351.50 KiB	1.86 GiB	

0 operaciones pendientes

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 Waspmote OTA feature:

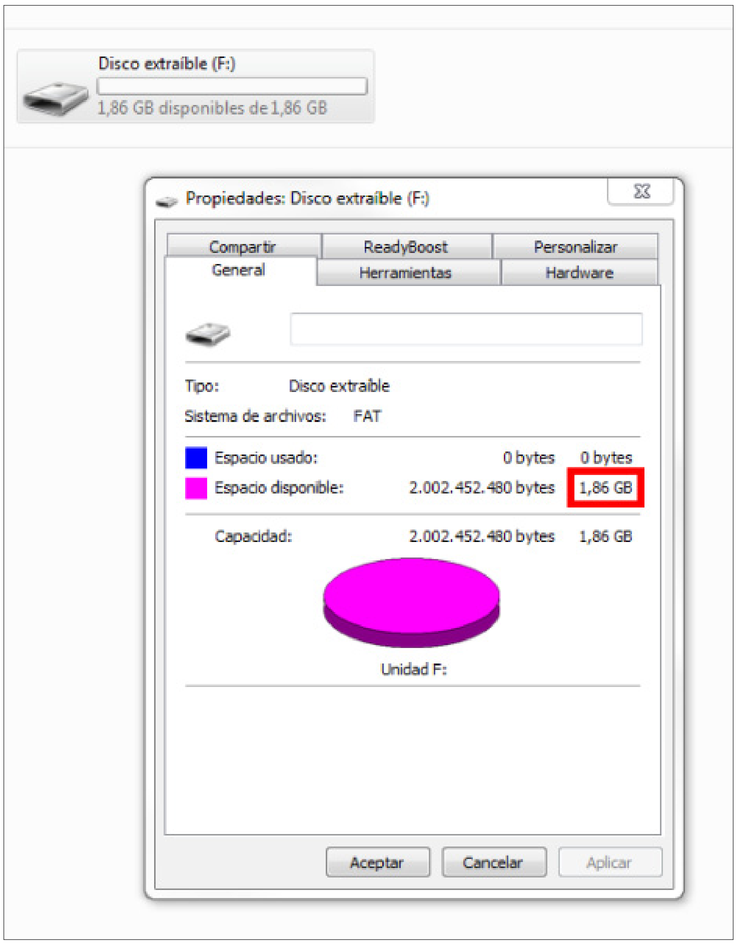

/dev/sdb - GParted (como superusuario)
GParted Editar Ver Dispositivo Partición Ayuda
/dev/sdb (1.84 GiB)
/dev/sdb1
1.84 GiB

Partición	Sistema de archivos	Punto de montaje	Tamaño	Usado	Libre	Opciones
/dev/sdb1 🟩	fat16	/media/3866-6239	1.84 GiB	252.50 KiB	1.84 GiB	
sin asignar ▨	sin asignar		1.91 MiB	---	---	

0 operaciones pendientes

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:



Disco extraíble (F:)
1,86 GB disponibles de 1,86 GB

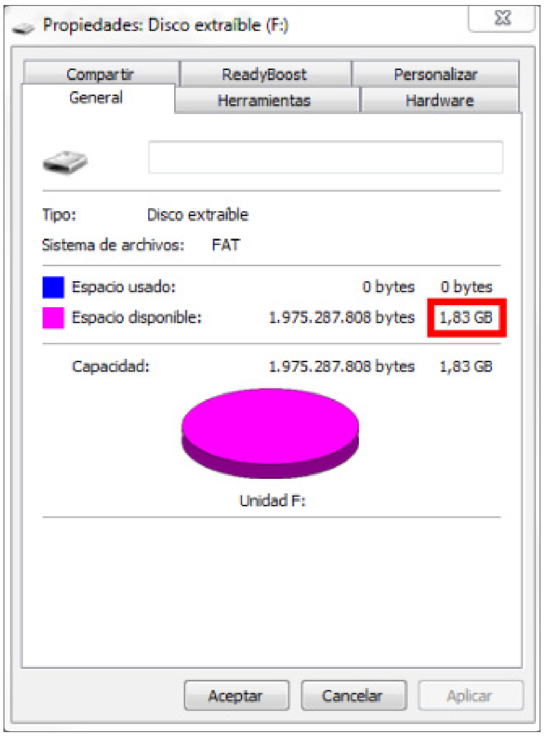
Propiedades: Disco extraíble (F:)

Tipo: Disco extraíble
Sistema de archivos: FAT

🟦	Espacio usado:	0 bytes	0 bytes
🟪	Espacio disponible:	2.002.452.480 bytes	1,86 GB
	Capacidad:	2.002.452.480 bytes	1,86 GB

Unidad F:

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



Propiedades: Disco extraíble (F:)

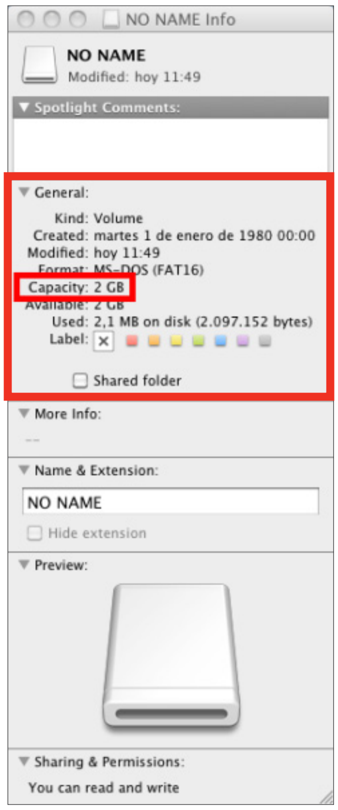
Tipo: Disco extraíble
Sistema de archivos: FAT

🟦	Espacio usado:	0 bytes	0 bytes
🟪	Espacio disponible:	1.975.287.808 bytes	1,83 GB
	Capacidad:	1.975.287.808 bytes	1,83 GB

Unidad F:

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:



NO NAME Info

NO NAME
Modified: hoy 11:49

Spotlight Comments:

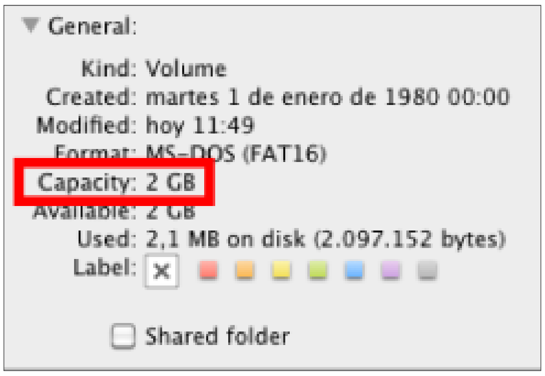
General:
Kind: Volume
Created: martes 1 de enero de 1980 00:00
Modified: hoy 11:49
Format: MS-DOS (FAT16)
Capacity: 2 GB
Available: 2 GB
Used: 2,1 MB on disk (2.097.152 bytes)
Label:
Shared folder

More Info:
--

Name & Extension:
NO NAME
Hide extension

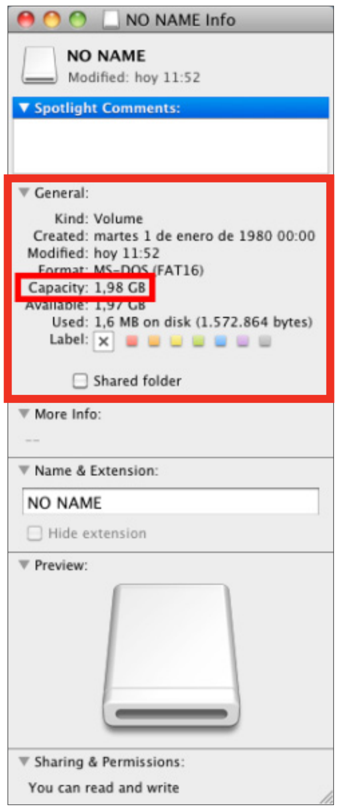
Preview:

Sharing & Permissions:
You can read and write



General:
Kind: Volume
Created: martes 1 de enero de 1980 00:00
Modified: hoy 11:49
Format: MS-DOS (FAT16)
Capacity: 2 GB
Available: 2 GB
Used: 2,1 MB on disk (2.097.152 bytes)
Label:
Shared folder

Otherwise, if the size is 1.98 GB, there are some defective sectors and the OTA operations will fail:



NO NAME Info

NO NAME
Modified: hoy 11:52

Spotlight Comments:

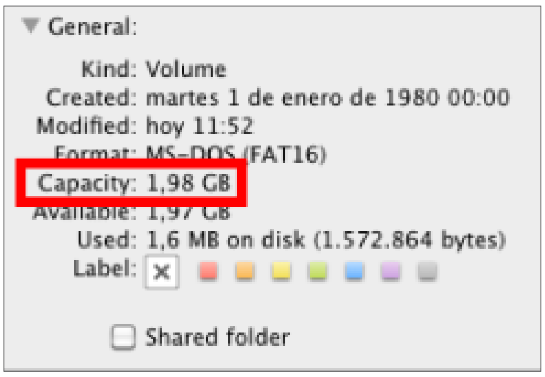
General:
Kind: Volume
Created: martes 1 de enero de 1980 00:00
Modified: hoy 11:52
Format: MS-DOS (FAT16)
Capacity: 1,98 GB
Available: 1,97 GB
Used: 1,6 MB on disk (1.572.864 bytes)
Label:
Shared folder

More Info:
--

Name & Extension:
NO NAME
Hide extension

Preview:

Sharing & Permissions:
You can read and write

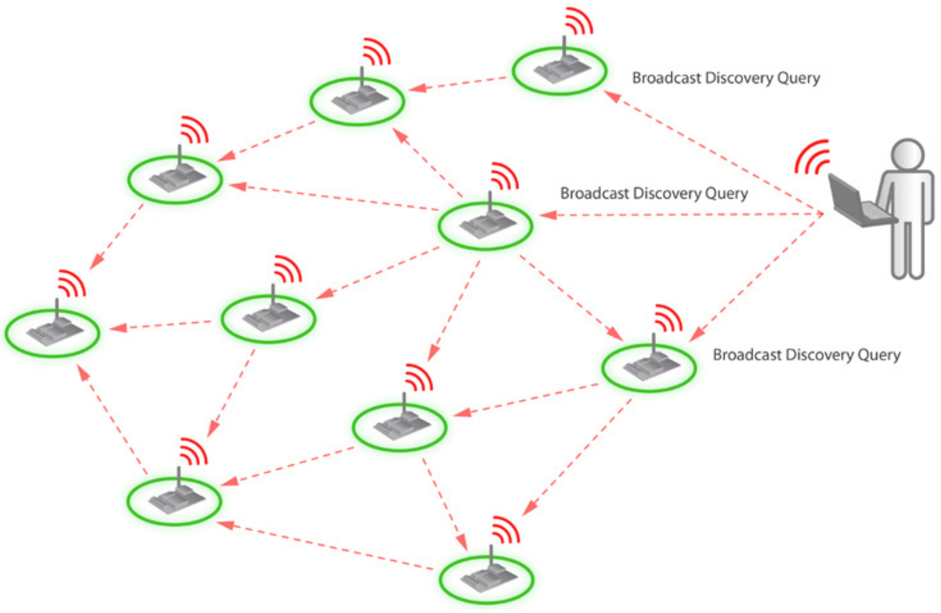


General:
Kind: Volume
Created: martes 1 de enero de 1980 00:00
Modified: hoy 11:52
Format: MS-DOS (FAT16)
Capacity: 1,98 GB
Available: 1,97 GB
Used: 1,6 MB on disk (1.572.864 bytes)
Label:
Shared folder

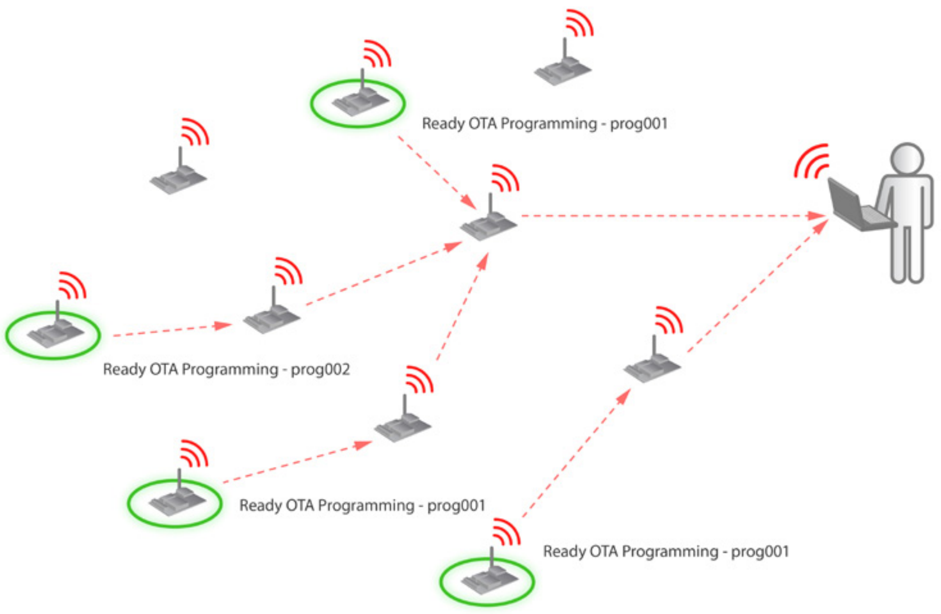
5. OTA Step by Step

- 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.

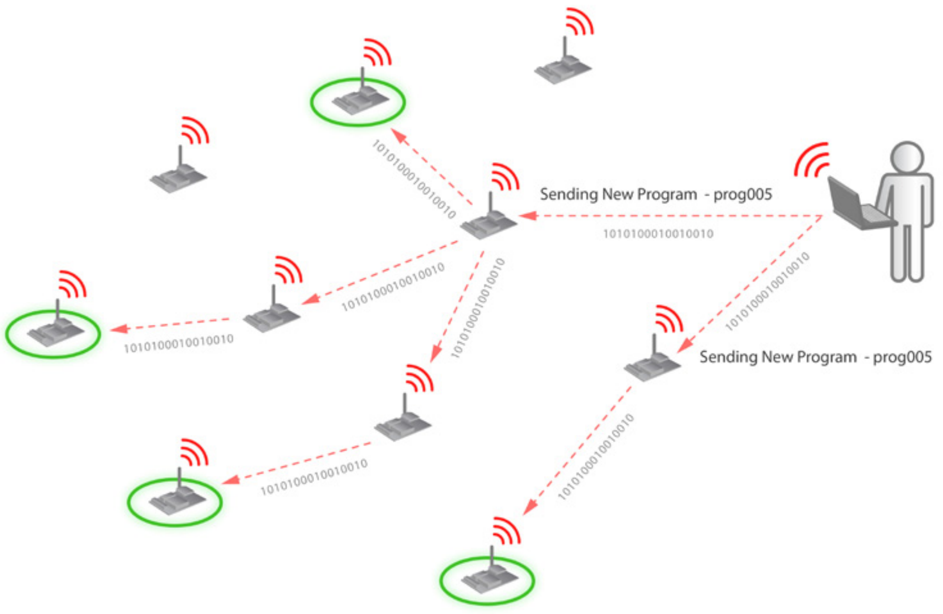

Broadcast Discovery Query
Broadcast Discovery Query
Broadcast Discovery Query

The nodes which are ready at this moment will answer with a "Ready to OTA" frame.

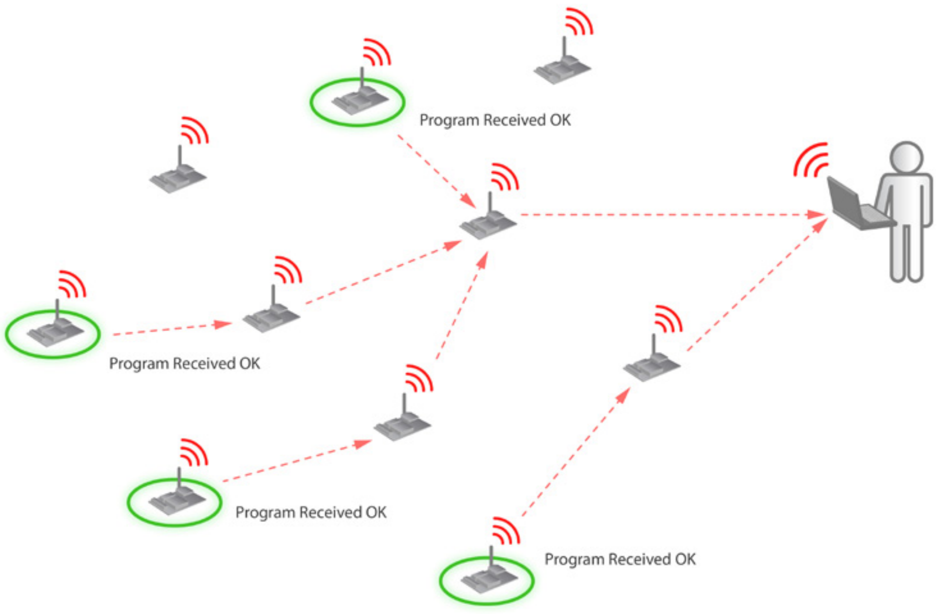

Ready OTA Programming - prog001
Ready OTA Programming - prog002
Ready OTA Programming - prog001
Ready OTA Programming - prog001

- **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.

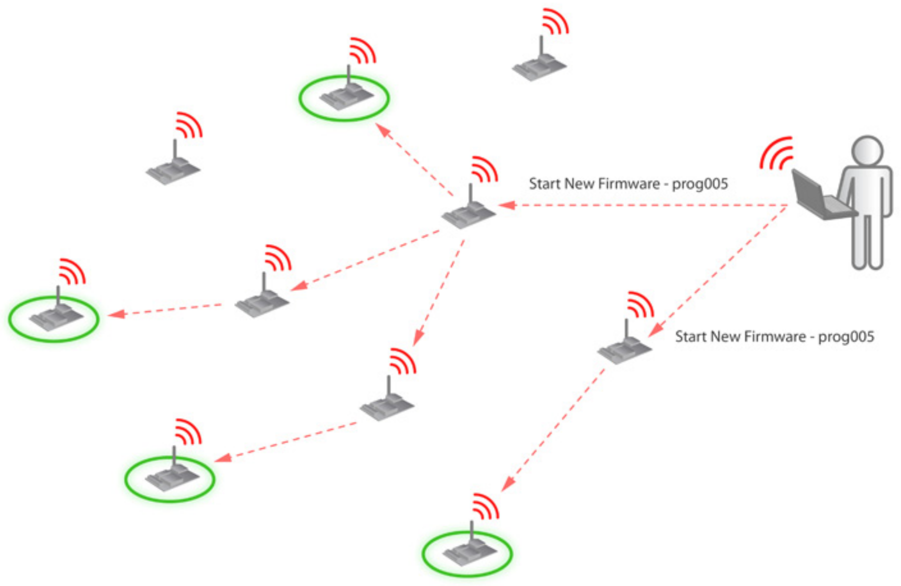


Each node which receives the program sends a message to the gateway to inform of the success of the process.

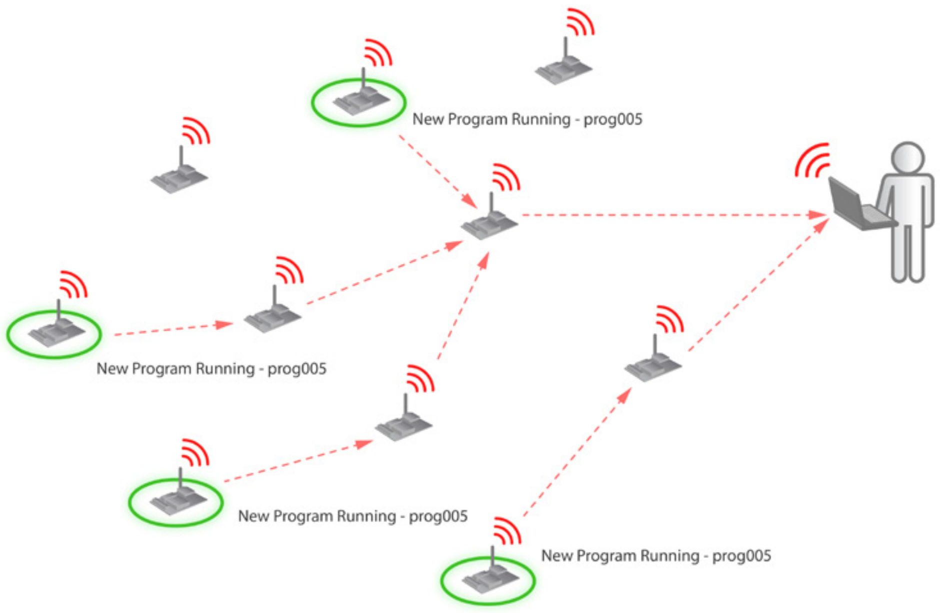


- **Reboot and start with the new program**

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.


New Program Running - prog005
New Program Running - prog005
New Program Running - prog005
New Program Running - prog005

6. Memory and Storage System in Waspmote

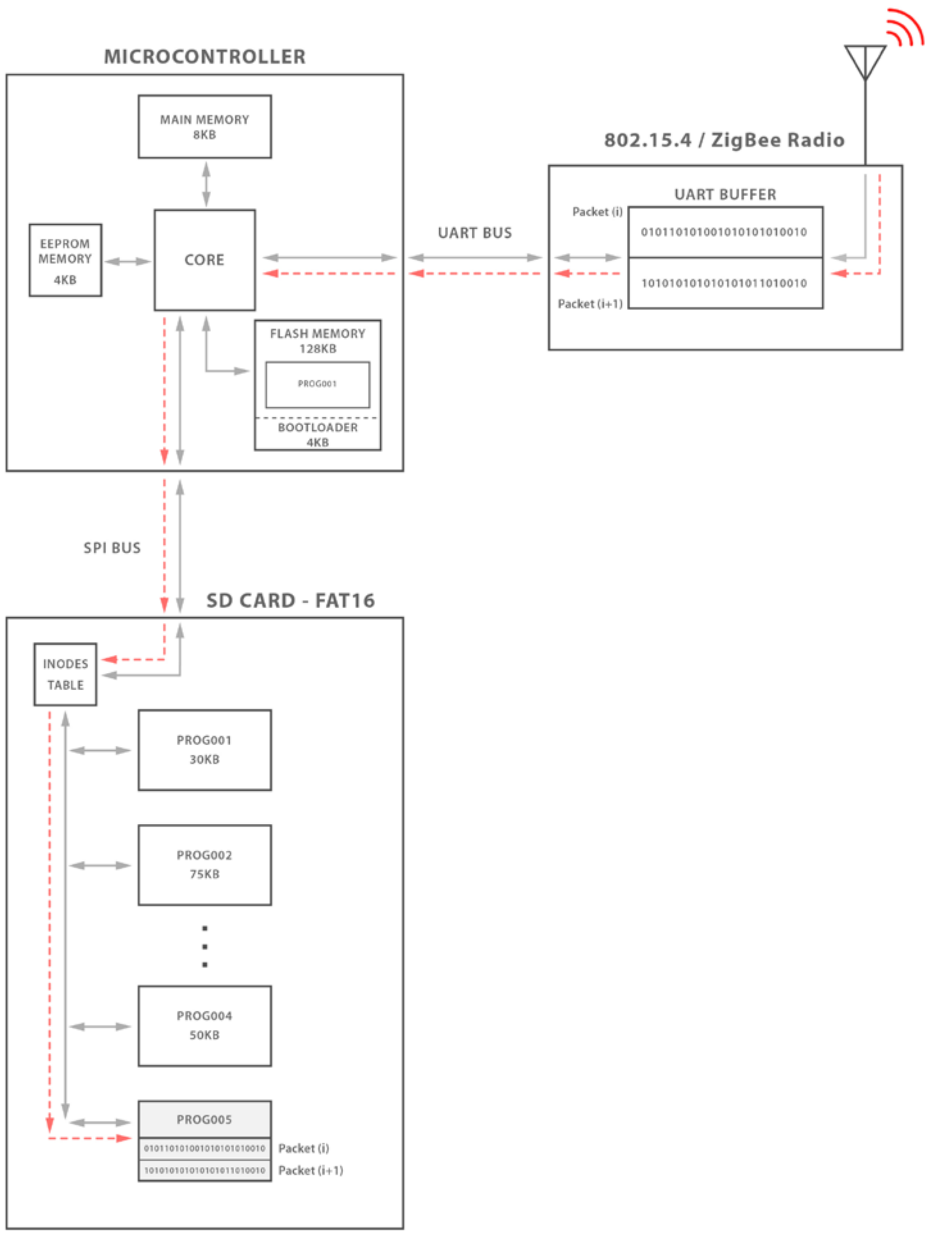
There are four different memory systems in Waspmote:

- 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.

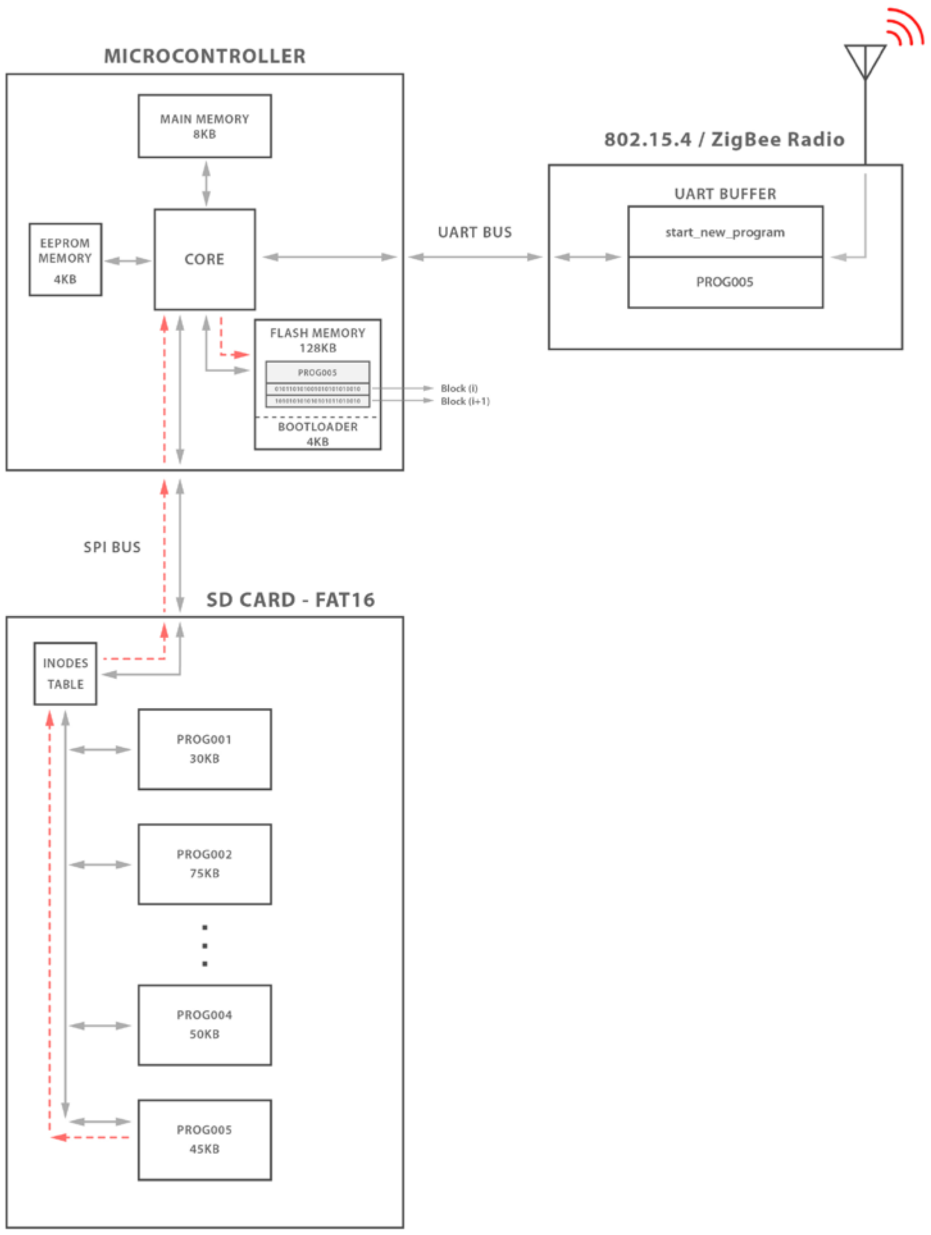
There a special file called "boot.txt" which contains the information of all the programs available in Waspmote. Each time we add a new program a reference line is added. The content of this file can be accesses using the "get_boot_list" function.

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.



7. 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:

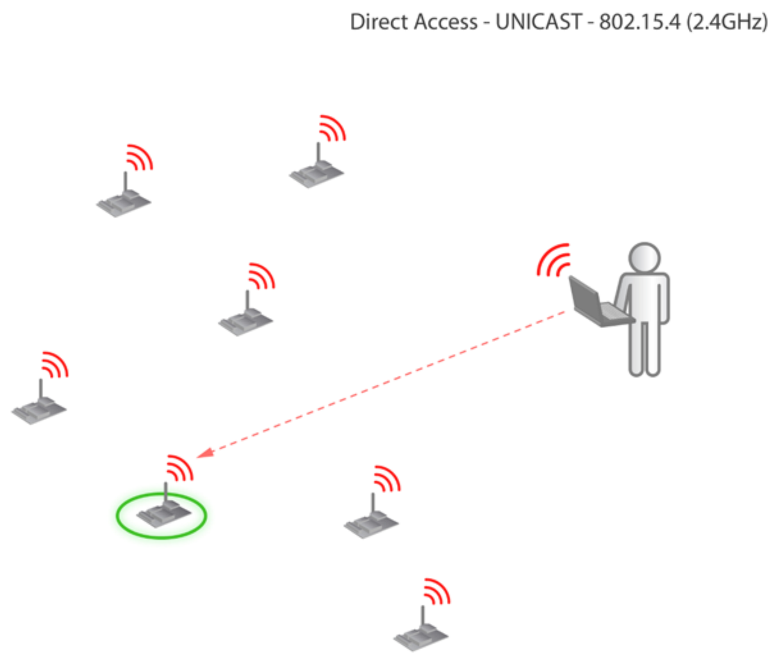
- Direct access: when the nodes are accessed in just one hop (no forwarding of the packets is needed). The protocols which work with this mode are: 802.15.4-2.4GHz, RF-868MHz and RF-900MHz.
- Multihop: when the nodes are accessed in two or more hops. In this mode some nodes have to forward the packets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

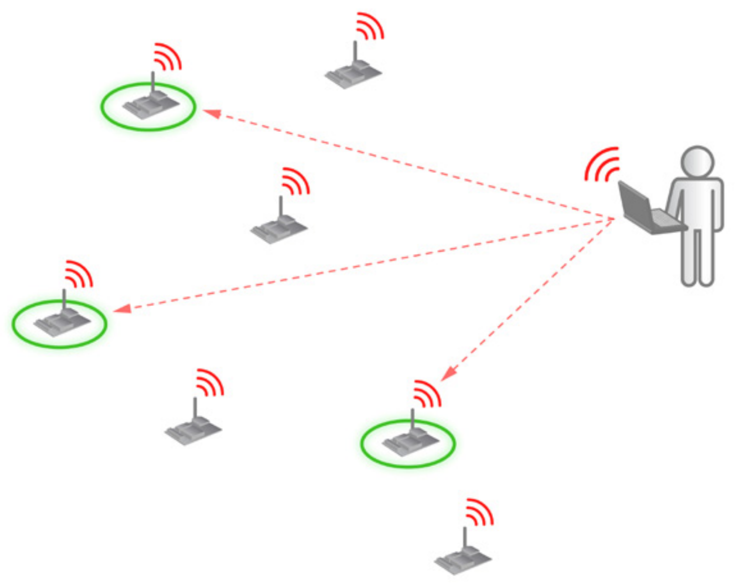
In order to send the firmware in MULTICAST mode what we make is to create a subnetwork formed by the nodes to be reprogrammed and then we send the new firmware using the BROADCAST transmission in order to make each packet reach all nodes at the same time. In this mode no ACK's packets are expected so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

Direct Access - MULTICAST - 802.15.4 (2.4GHz)



Direct Access - BROADCAST - 802.15.4 (2.4GHz)

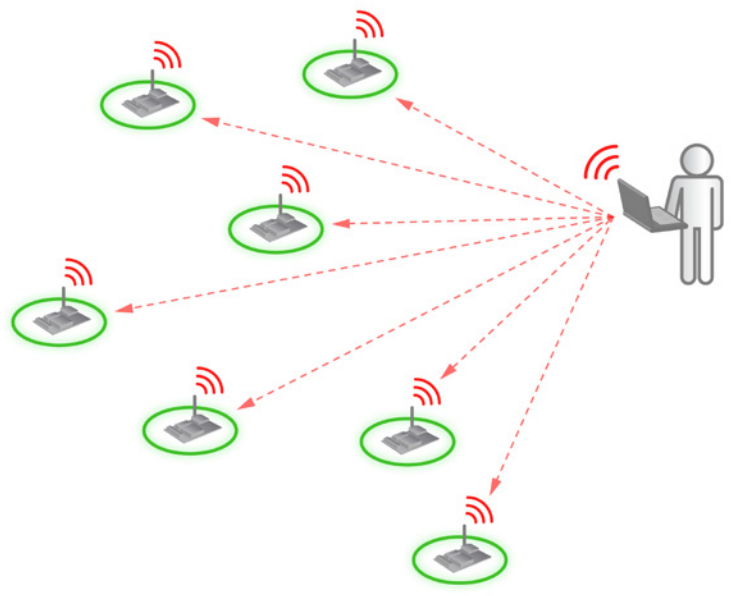
Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of channel as all the nodes in range are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

In this mode no ACK's packets are expected so we will have to wait to the end to see if all the nodes where reprogrammed successfully.

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

- Auth key
- New firmware

Direct Access - BROADCAST - 802.15.4 (2.4GHz)



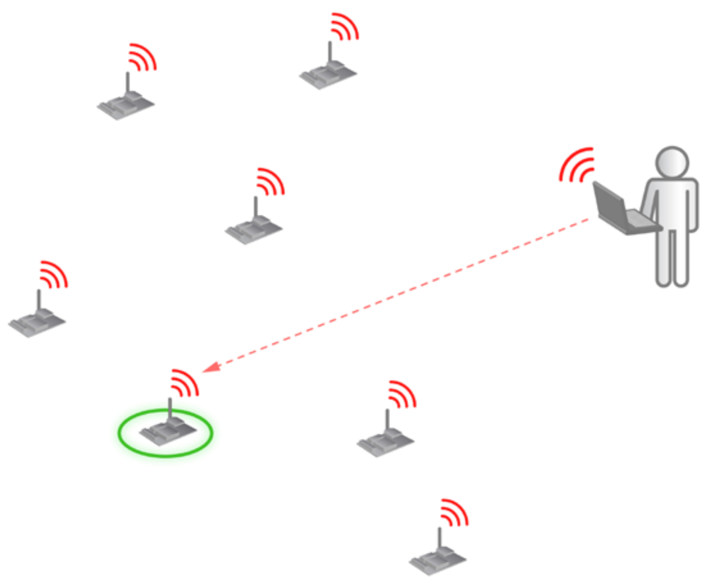
Direct Access - UNICAST - 868MHz/900MHz

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 - UNICAST - 868MHz/900MHz



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's packets are expected so we will have to wait to the end to see if all the nodes where reprogrammed successfully.

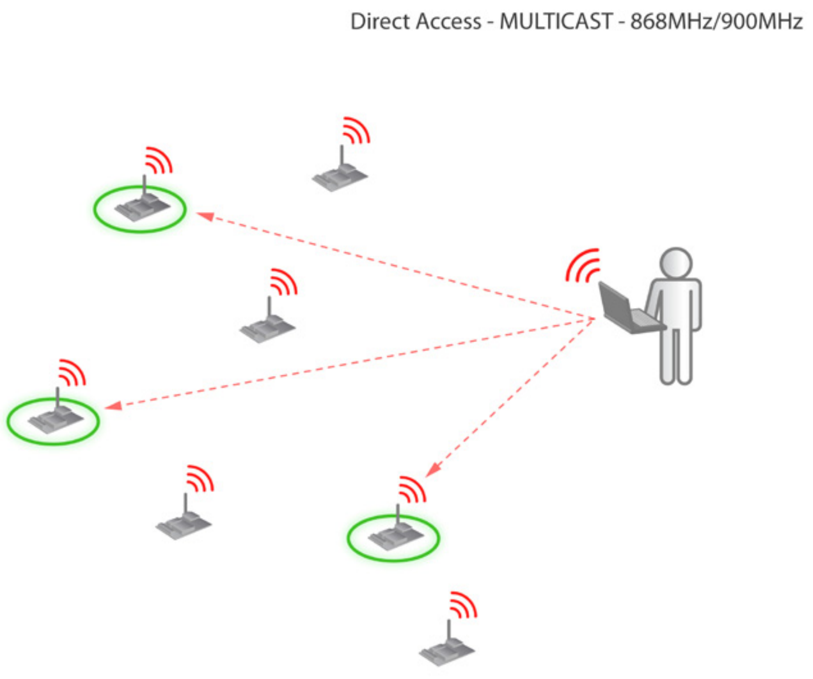
In the 868 and 900MHz models we can not change the frequency band, so what we will make is to change the Encryption Key used in order to created a subnetwork with the nodes to be reprogrammed. This way the nodes which are not involved in the OTA process do not have to be discarding the packets sent in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

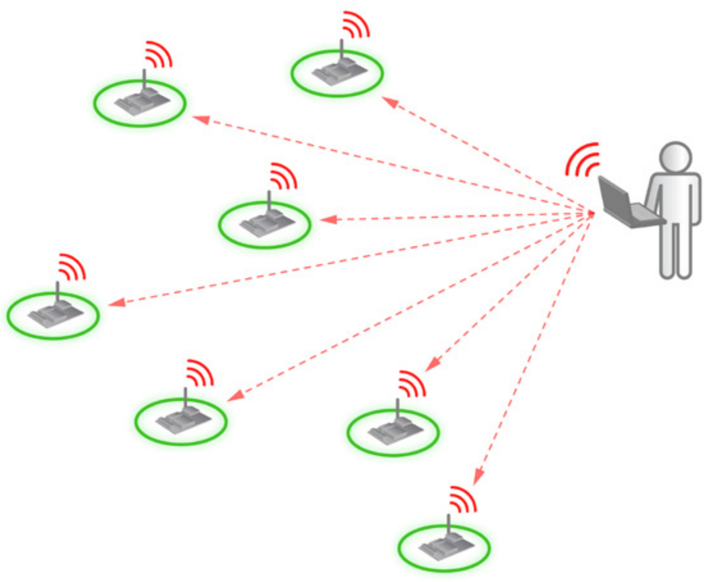
Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of Encryption Key as all the nodes in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

In this mode no ACK's packets are expected so we will have to wait to the end to see if all the nodes where reprogrammed successfully.

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

- Auth key
- New firmware

Direct Access - BROADCAST - 868MHz/900MHz



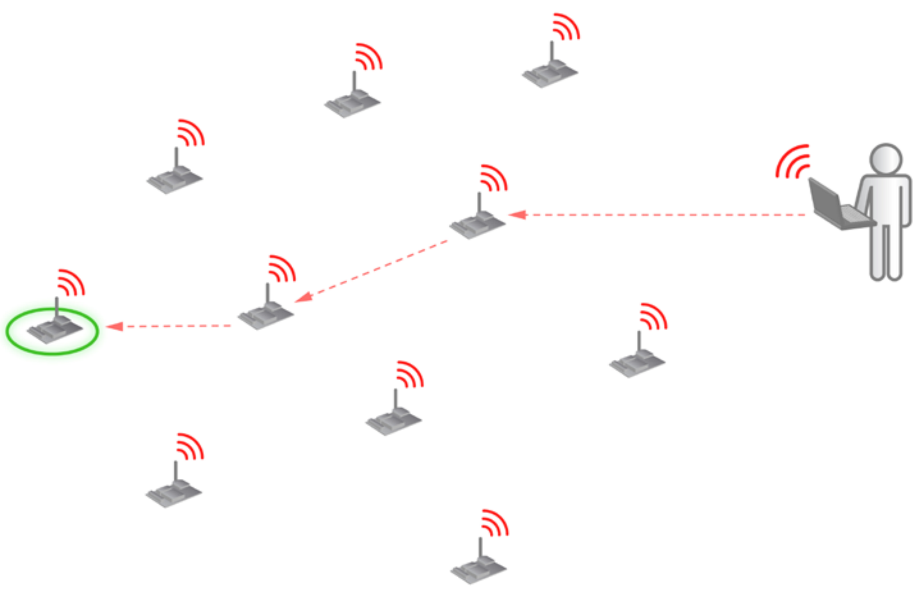
Multihop Access - UNICAST - ZigBee & DigiMesh (2.4GHz)

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 nor the channel or 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 - UNICAST - ZigBee & DigiMesh (2.4GHz)



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's packets are expected so we will have to wait to the end to see if all the nodes where reprogrammed successfully.

In a Multihop network nor the channel or 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.

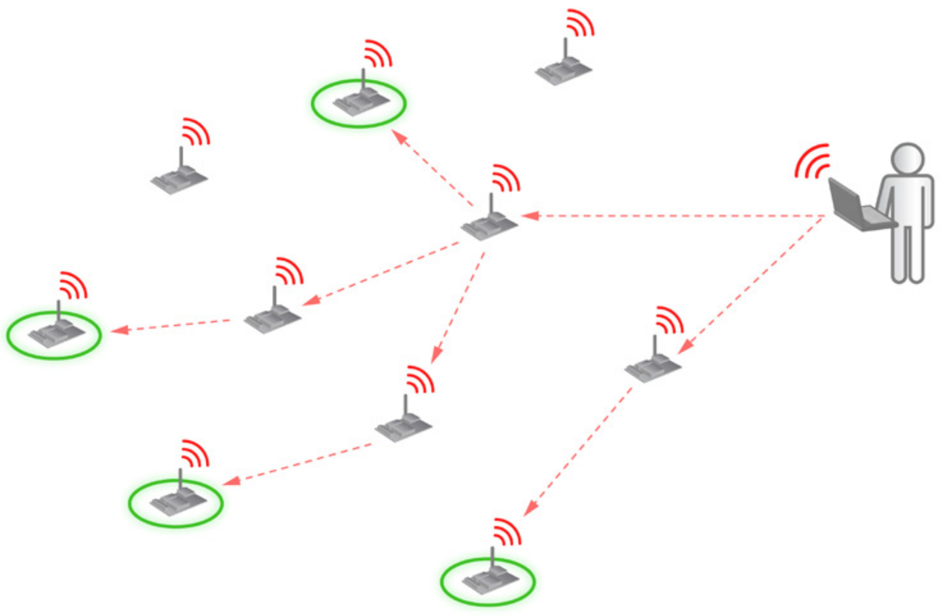
Before starting with the OTA process the Gateway sends a special packet to the node to be reprogrammed where it is specified a New Auth Key along with the Old Auth Key to be used by the node to validate if the OTA programmer is a trusted source. Both Gateway and nodes change the Auth Key to the new specified; 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 - MULTICAST - ZigBee & DigiMesh (2.4GHz)



Multihop Access - BROADCAST - ZigBee & DigiMesh (2.4GHz)

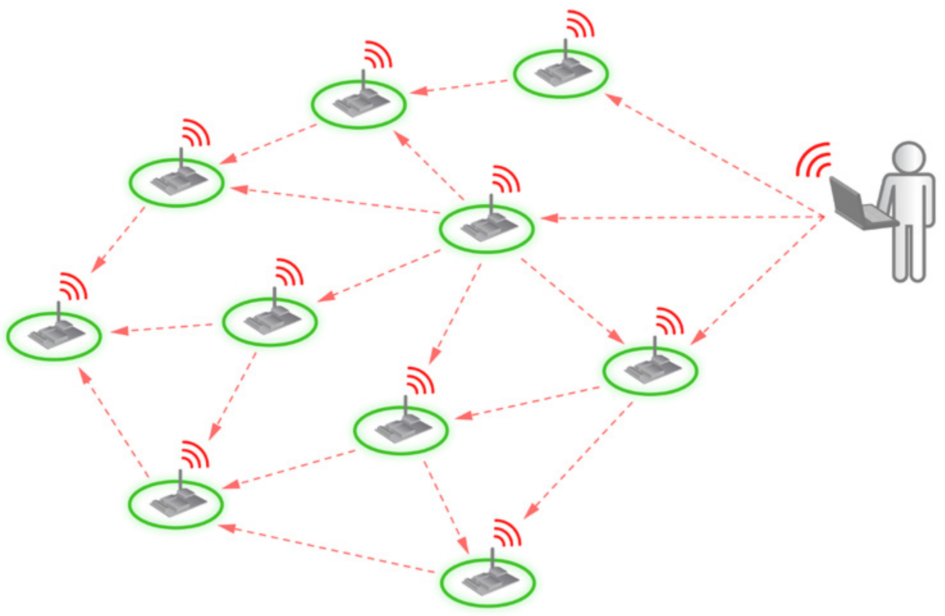
Before starting with the OTA process the Gateway sends a special packet to the nodes to be reprogrammed where it is specified the Auth Key which will be used by each of the nodes to validate if the OTA programmer is a trusted source. This time there is no change of the Auth Key as all the nodes in the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

In this mode no ACK's packets are expected so we will have to wait to the end to see if all the nodes were reprogrammed 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)



8. OTA Shell

8.1. 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.

8.2. 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
 -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
```

8.3. 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".
- PAN ID: It is the ID of the current network (2B).
- XBee model: 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 Flag: [on, off] It specifies if the Encryption is activated or not.
- Encryption Key: It is the key (16B) used by the AES 128b algorithm to encrypt the frames in the Link Layer (IEEE 802.15.4).
- Discarded Data File: 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 saved in ascii-hex format (one frame per line).

8.4. 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 Waspmote. If the node is in this state when we send the "scan_nodes" query it will answer with the next information: ID - MAC - PID.

```
#./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)
```

The answer is of the next form: MAC - ID - PID.

Let's see some examples:

Scan during 3 second for any node:

```
 #./otap -scan_nodes --mode BROADCAST --time 3
------------------------------------------------------------
 Total Nodes: 2 - Time elapsed 3s
 0 - Node 0013a2004061097c - waspmote001 - prog001 - READY
 1 - Node 0013a20041615623 - waspmote002 - prog001 - READY
------------------------------------------------------------
```

Search during 2 seconds 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 OTA Programming
---------------------------------------------------------------------------
```

8.5. Getting the bootlist

There is 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. 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
```

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:20110331140623
 1 -  PID: prog002 - Date:20110331140623
 2 -  PID: prog003 - Date:20110404103335
 3 -  PID: prog004 - Date:20110404103335
 4 -  PID: prog005 - Date:20110404103335
-----------------------------------------------------------------
```

Ask for the bootlist to two different nodes node:

```
 #./otap -get_boot_list -mode UNICAST --mac 0013a2004061097c,0013a20041615623
------------------------------------------------------------------------------
 Bootlist Node 0013a2004061097c - length: 5

 0 - PID: prog001 - Date:20110331140623
 1 - PID: prog002 - Date:20110331140623
 2 - PID: prog003 - Date:20110404103335
 3 - PID: prog004 - Date:20110404103335
 4 - PID: prog005 - Date:20110404103335
------------------------------------------------------------------------------
 Bootlist Node 0013a20041615623 - length: 5

 0 - PID: progr11 - Date:20110331140623
 1 - PID: progr12 - Date:20110331140623
 2 - PID: progr13 - Date:20110404103324
 3 - PID: progr14 - Date:20110404103325
 4 - PID: progr15 - Date:20110404103327
------------------------------------------------------------------------------
```

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.

8.6. 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 test.cpp.hex
---------------------------------------------------
 Name:              test.cpp.hex
 Date:              Wed Apr 27 13:24:18 CEST 2011
 Size:              2100 bytes
 Packets to send:   17
 Estimated time:    3.77 seconds
---------------------------------------------------
```

8.7. 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:

Time = time_per_packet * number_packets * 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>      6 ascii characters
    --pid <PROGRAM_1>                    program identifier, 7 characters in length
    --send_mode <BINARY|ASCII>           specify in what format the data will be sent
```

Examples:

Direct Access UNICAST - 802.15.4 + Change of channel

```
#./otap -send --file prog001.cpp.hex --mac 0013a2004061097c --mode UNICAST --channel
0x0C --deliveries 1 --pid prog001
------------------------------------------------------------------------------------------
Name:              test.cpp.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
------------------------------------------------------------------------------------------
```

Direct Access MULTICAST - 802.15.4 + Change of channel

```
 #./otap -send --file test.cpp.hex --mac 0013a2004061097c,0013a20041615623 --mode MULTICAST
 --channel 0x0C --deliveries 2 --pid prog001
-------------------------------------------------------------------------------
 Name:              test.cpp.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

```
 #./otap -send --file test.cpp.hex --mode BROADCAST --deliveries 2 --pid prog001
-------------------------------------------------------------------------------
 Name:              test.cpp.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 0013a2004061097c: PROGRAM RECEIVED OK
-------------------------------------------------------------------------------
 Node 0013a20041615623: PROGRAM RECEIVED OK
-------------------------------------------------------------------------------
 Node 0013a20041744564: PROGRAM RECEIVED OK
-------------------------------------------------------------------------------
```

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

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

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

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

Direct Access BROADCAST - 868/900MHz

```
#./otap -send --file test.cpp.hex --mode BROADCAST --deliveries 2 --pid prog001
```

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

```
#./otap -send --file prog001.cpp.hex --mac 0013a2004061097c --mode UNICAST --new_authkey
secret00 --deliveries 1 --pid prog001
```

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

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

Multihop BROADCAST- ZigBee & DigiMesh (2.4GHz)

```
#./otap -send --file test.cpp.hex --mode BROADCAST --deliveries 2 --pid prog001
```

8.8. Starting a new program

After sending a new program it does not start automatically. We have to send the "star_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.

```
#./otap -start_new_program
usage: otap -start_new_program
    --mac <0000000000000000>              mac address, 16 hex characters
    --macs_file <list.txt>                file containing list of macs (if MULTICAST eneabled)
    --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:

```
 #./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:

```
 #./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
------------------------------------------------------------
```

8.9. Reset

If the new program load fails and the node does not respond we can use the "reset" command in order to reboot the mote and make it accessible again. Using this command is like pressing physically the reset button on the Waspmote board, it causes a hardware reset independently of the microcontroller.

If this command is sent before a new program starts for the first time it will restore the previous stable program. The rest of the times the same program will start again.

This command will reset the mote even though the microcontroller is blocked.

```
# ./otap -reset
usage: otap -restore
    --mac <0000000000000000>              mac address, 16 hex characters
    --macs_file <list.txt>                file containing list of macs (select MULTICAST)
    --mode <UNICAST|MULTICAST|BROADCAST>  restore mode


#./otap -reset --mode UNICAST --mac 0013a2004061097c

Node 0013a2004061097c - prog001  - RESTARTING

#./otap -reset --mode BROADCAST

Node 0013a2004061097c - prog001  - RESTARTING
Node 0013a20041615623 - prog001  - RESTARTING
Node 0013a20041744564 - prog001  - RESTARTING
```

8.10. 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
```

8.11. 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 OTA PROGRAMMING

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.

STARTING NEW FIRMWARE

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.

BOOTLIST COMPLETE

It is sent by the nodes if the bootlist file has been correctly sent when the "get_bootlist" command is executed.

RESTARTING

It is sent by the nodes before rebooting when we execute the "reset" command.

Failure Responses:

PROGRAM UPLOAD FAILURE

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

PREVIOUS PROGRAM RESTORED

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 estable program is restored and this message sent the OTA programmer.

BOOTLIST FAILURE

It is sent by the nodes when we call the "get_bootlist" command and one or more packets are lost in the transmission.

ERROR STARTING

It is sent by the nodes when we call the "start_new_program" command with an invalid PID.

9. Setting the environment

9.1. Meshlium

Meshlium is a Linux router developed by Libelium which acts as the ZigBee Gateway of the sensor network.

It can contain 5 different radio interfaces: Wifi 2.4GHz, Wifi 5GHz, GPRS, Bluetooth and ZigBee. As well as this, Meshlium can also integrate a GPS module for mobile and vehicular applications and be solar and battery powered. These features a long with an aluminium IP-67 enclosure allows Meshlium to be placed anywhere outdoor. Meshlium comes with the Manager System, a web application which allows to control quickly and easily the Wifi, ZigBee, Bluetooth and GPRS configurations a long with the storage options of the sensor data received.

Figure 2: Meshlium Router

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

First of all stop the default ZigBee daemon:

```
#/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

9.2. 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 Waspmote 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
```

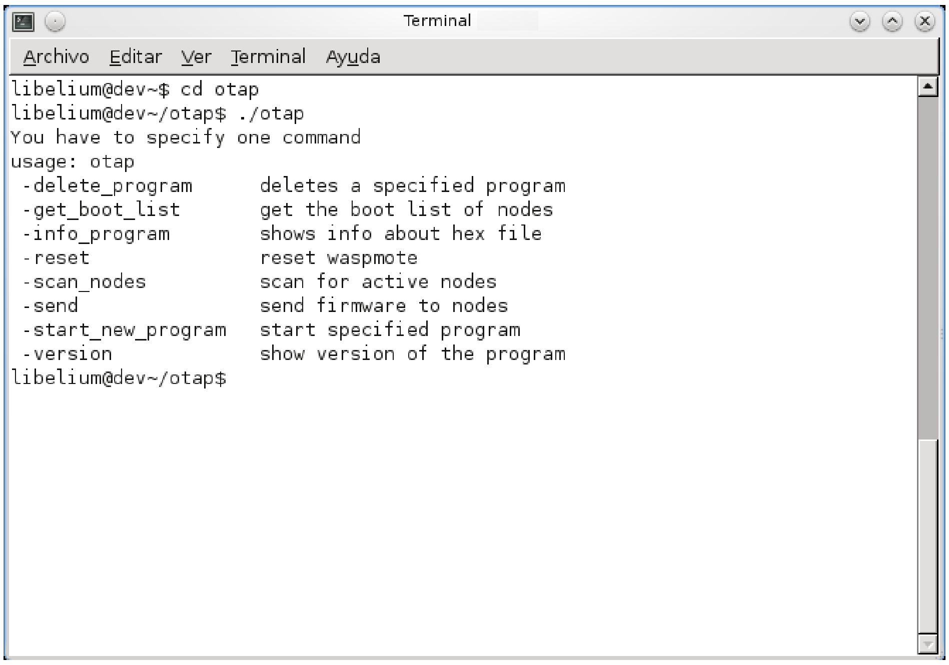
Extract all the files of the otap.tar.gz file executing the following in the directory where you have the compressed file:

```
#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
```

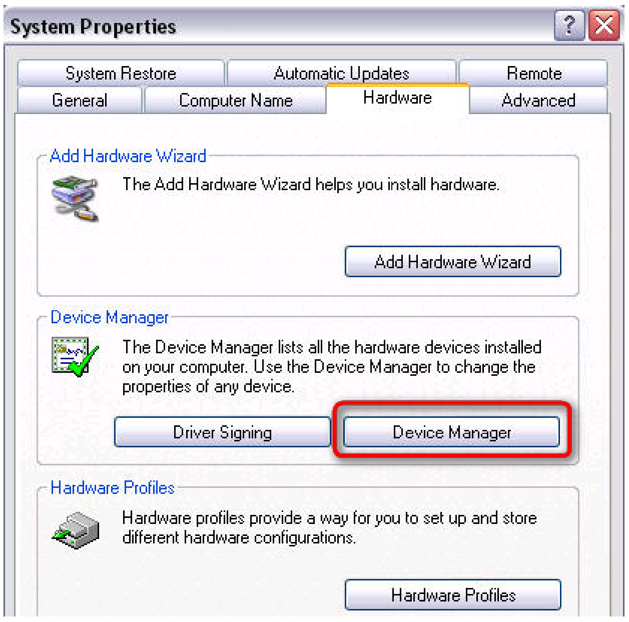
Now you can run the otap program simply executing ./otap (or ./otap64 if you've a 64 bits OS):



```
libelium@dev~$ 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$
```

9.3. 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.

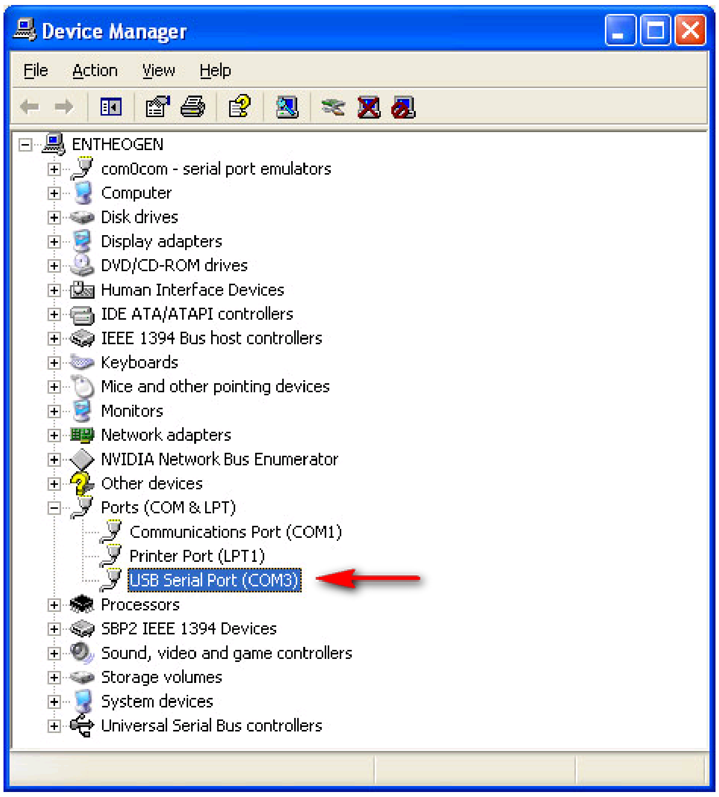


The Add Hardware Wizard helps you install hardware.

The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Hardware profiles provide a way for you to set up and store different hardware configurations.

- 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).

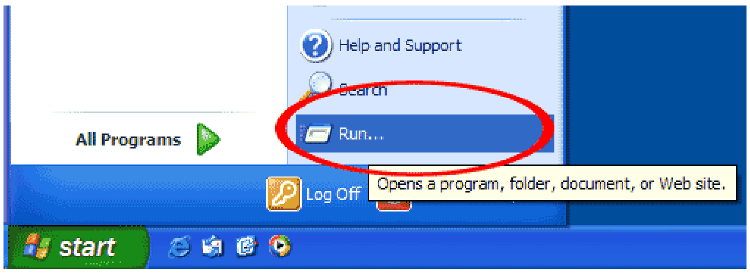


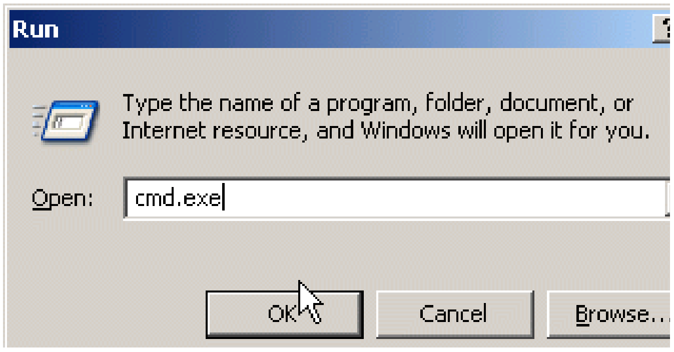
```
ENTHEOGEN
  com0com - serial port emulators
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  IEEE 1394 Bus host controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  NVIDIA Network Bus Enumerator
  Other devices
  Ports (COM & LPT)
    Communications Port (COM1)
    Printer Port (LPT1)
    USB Serial Port (COM3)
  Processors
  SBP2 IEEE 1394 Devices
  Sound, video and game controllers
  Storage volumes
  System devices
  Universal Serial Bus controllers
```

- 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

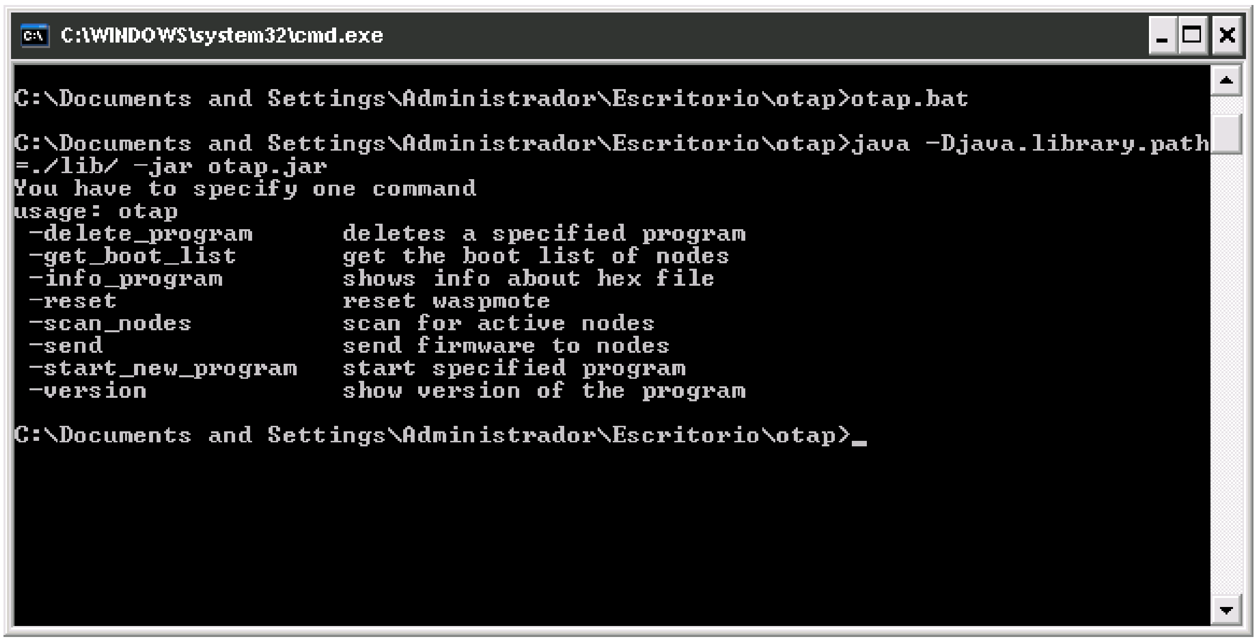




Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: 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):



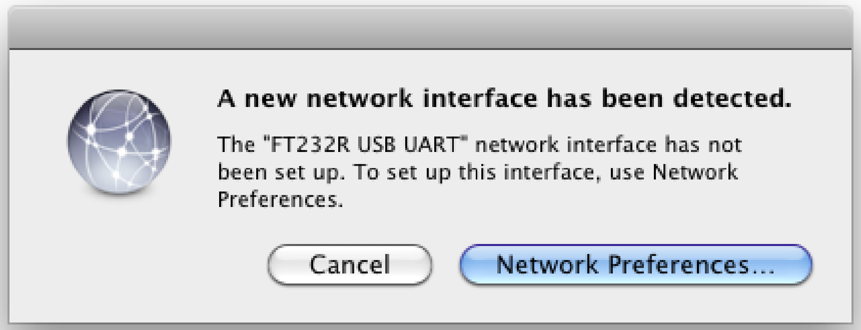
```
C:\Documents and Settings\Administrador\Escritorio\otap>otap.bat

C:\Documents and Settings\Administrador\Escritorio\otap>java -Djava.library.path
=./lib/ -jar otap.jar
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

C:\Documents and Settings\Administrador\Escritorio\otap>_
```

9.4. 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 you mac machine. Then a screen showing the following can appear:

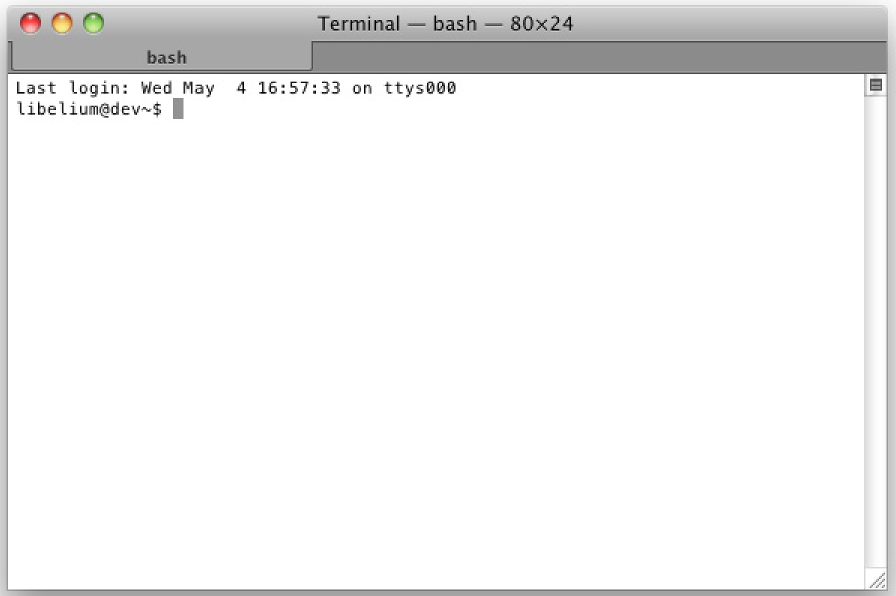


A new network interface has been detected.

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

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 ttys000
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:

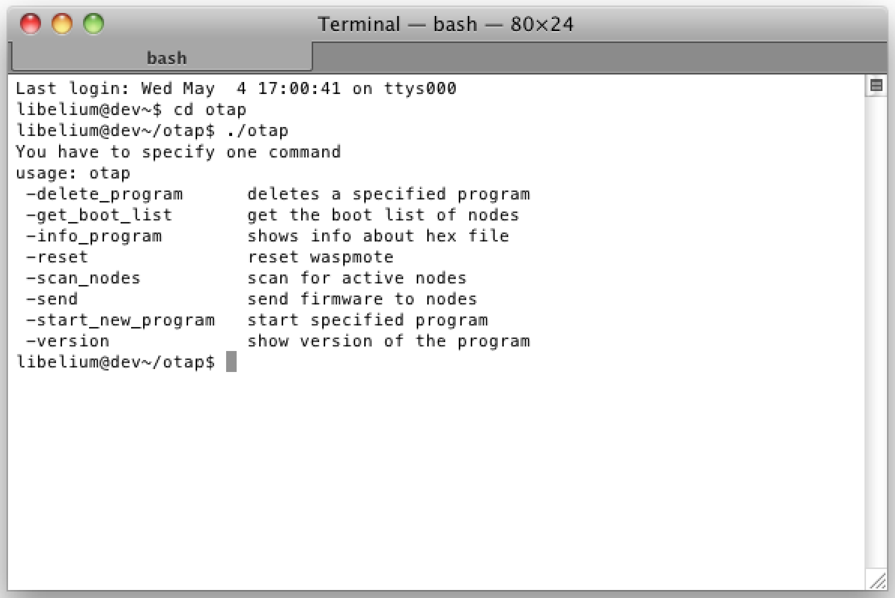
```
~/otap$ ls /dev/tty.usbserial*

/dev/tty.usbserial-A8003LP0
```

- Put the showed path in the 'xbee.conf' file in the following line:

```
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):



```
Last login: Wed May  4 17:00:41 on ttys000
libelium@dev~$ 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$ 
```

10. Hardware Setup

10.1. Important dates and versions

Waspmotes 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 Waspmote kits before that date and you want them to run OTA please contact our Commercial Department at: **http://www.libelium.com/company/contact**

Waspmotes bought after the 02/05/2011 come with the hardware ready to execute the 'reset' function. If you bought your Waspmote kits before that date and you want them to run the 'reset' command please contact our Commercial Department at: **http://www.libelium.com/company/contact**

NOTE: The reset command is not necessary to run OTA. It simply adds an extra option which will be useful just in certain cases.

10.2. Waspmote Setup

Make sure that you are executing the API version 0.18 or older. Download the last version of the API at:

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

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 Waspmote.

The 'Auth Key' must be set in each Waspmote to support OTA. By default, this key is set as 'LIBELIUM' and it is stored in the EEPROM.

The 'Mote ID' is an identifier to distinguish every Waspmote in a unique way. It is stored in the EEPROM and by default it is set as '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 Waspmotes in the network. This process is done through special XBee frames which are identified with special packetIDs.

   The OTA communication is transparent for the user, but he will need to avoid to create frames with one of the following packetIDs (also known as Application ID):

   - 0xF8
   - 0xF9
   - 0xFA
   - 0xFB
   - 0xFC
   - 0xFD
   - 0xFE
   - 0xFF

2. OTA uses space in the EEPROM of each Waspmote 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:

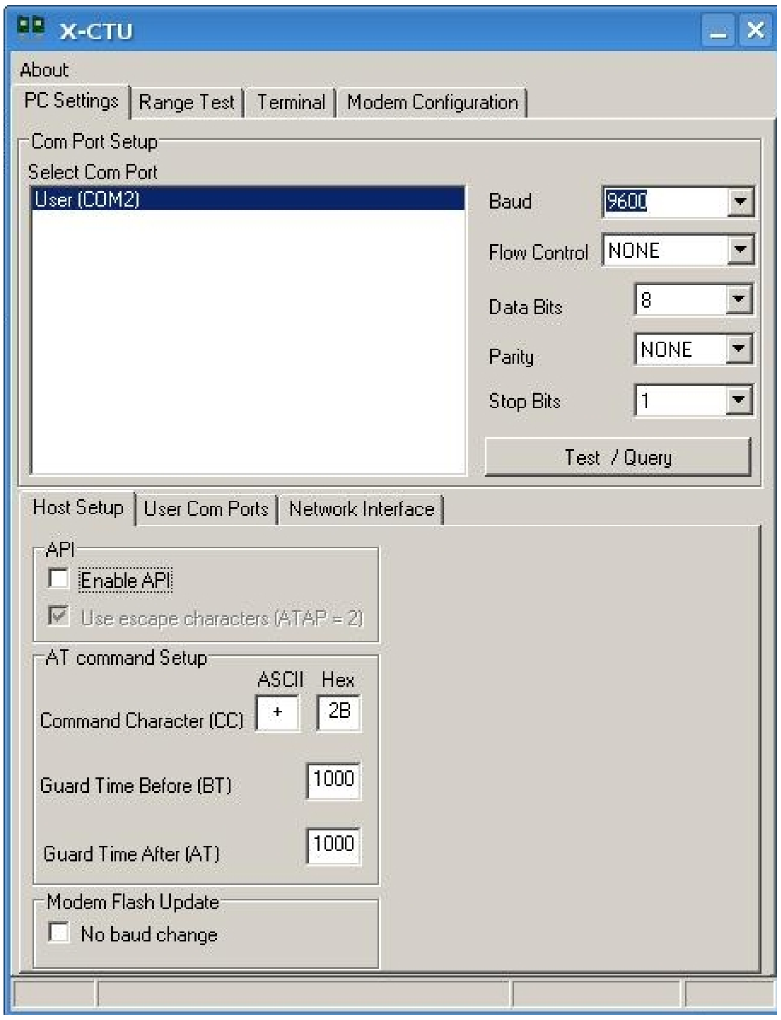
```
#define auth_key “LIBELIUM”
#define id_mote “WASPMOTE00000001”
      void setup()
      {
          for(int i=0;i<8;i++)
          {
            Utils.writeEEPROM(107+i,auth_key[i]);
          }
          for(int i=0;i<16;i++)
          {
            Utils.writeEEPROM(147+i,id_mote[i]);
          }
          xbee802.init(XBEE_802_15_4,FREQ2_4G,NORMAL);
          xbee802.ON();
          xbee802.checkNewProgram();
      }
      void loop()
      {
 if( XBee.available() )
{
    xbee802.treatData();
    while( xbee802.programming_ON  && !xbee802.checkOtapTimeout() )
    {
        if( XBee.available() )
        {
           xbee802.treatData();
        }
    }
}
       }
```

10.3. 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 38400bps (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.



Com Port Setup
Select Com Port: User (COM2)
Baud: 9600
Flow Control: NONE
Data Bits: 8
Parity: NONE
Stop Bits: 1

API
☐ Enable API
☑ Use escape characters (ATAP = 2)

AT command Setup

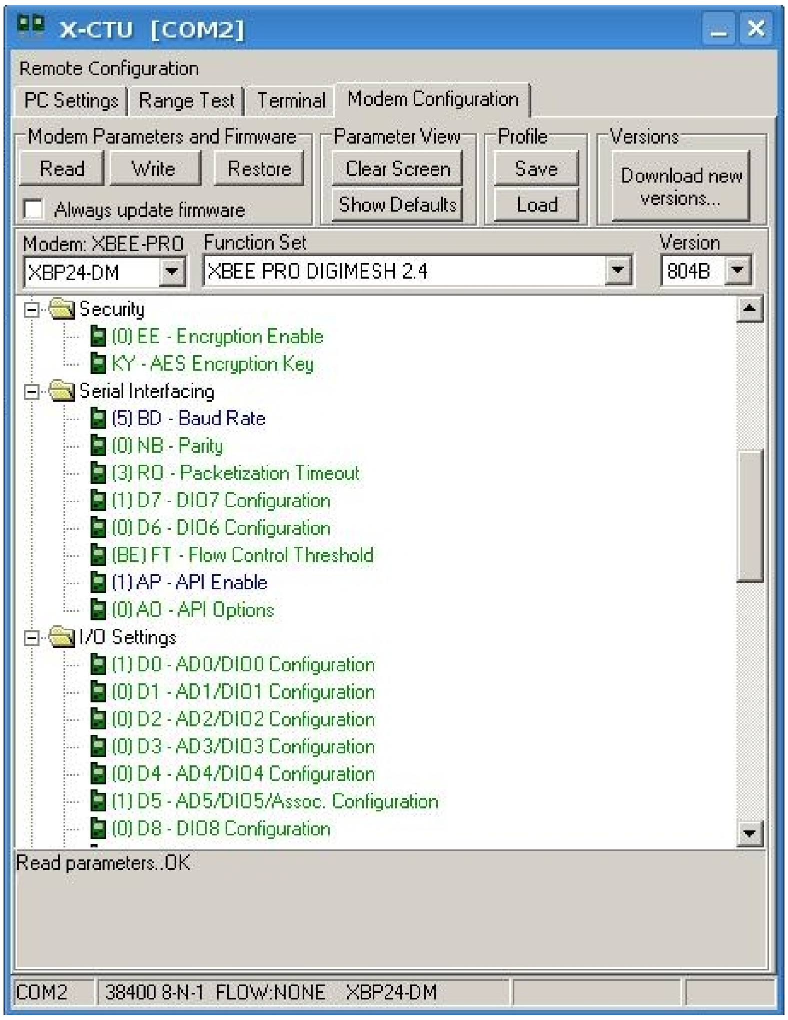
	ASCII	Hex
Command Character (CC)	+	2B
Guard Time Before (BT)		1000
Guard Time After (AT)		1000

Modem Flash Update
☐ No baud change

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.



Modem: XBEE-PRO XBP24-DM
Function Set: XBEE PRO DIGIMESH 2.4
Version: 804B

```
Security
    (0) EE - Encryption Enable
    KY - AES Encryption Key
Serial Interfacing
    (5) BD - Baud Rate
    (0) NB - Parity
    (3) RO - Packetization Timeout
    (1) D7 - DIO7 Configuration
    (0) D6 - DIO6 Configuration
    (BE) FT - Flow Control Threshold
    (1) AP - API Enable
    (0) AO - API Options
I/O Settings
    (1) D0 - AD0/DIO0 Configuration
    (0) D1 - AD1/DIO1 Configuration
    (0) D2 - AD2/DIO2 Configuration
    (0) D3 - AD3/DIO3 Configuration
    (0) D4 - AD4/DIO4 Configuration
    (1) D5 - AD5/DIO5/Assoc. Configuration
    (0) D8 - DIO8 Configuration
Read parameters..OK
```

```
COM2    38400 8-N-1  FLOW:NONE    XBP24-DM
```