NXT2WIFI User Guide

latest firmware version 1.0.130126
Daniele Benedettelli – 26 January 2013

INTRODUCTION

NXT2WIFI is a new Wi-Fi sensor for LEGO® MINDSTORMS® NXT. It allows you to control your robot using any browser-enabled device (iOS devices as iPhone and iPad included!), without having to install anything on your device. This is possible thanks to its graphic

webserver. It has full Wi-Fi secure networking functionality (WEP, WPA, WPA2), UDP TCP sockets, real time clock, and onboard NiMh battery.

- You can turn the device on and off using its switch on the side opposite to the plugs.
- You can reset the device by shortly pressing the reset button placed beside the power switch.
- You can recharge the onboard battery at any time, connecting a cable from your computer to the miniUSB plug on the device. A red led will light up when charging, and will turn off when the battery is fully charged.
- You can connect your NXT2WIFI device to the NXT port 4 using any NXT connector cable.

Communication Protocol Conventions

This section describes the communication protocol of the NXT2WIFI device. The NXT2WIFI device has two serial ports: a miniUSB plug to communicate with the PC (virtual COM port), and a NXT plug to exchange data with your NXT brick, using the RS485-enabled port 4. Sending commands from NXT or from a serial terminal on computer is transparent for the NXT2WIFI.
In the following, the messages are going from NXT (or from the computer) to NXT2WIFI. The responses are from NXT2WIFI to NXT (or to the computer). Every command must start with the $ symbol, and must be terminated with the linefeed character (0x0A). The NXT2WIFI controls the flow toggling the read/write mode on RS485. The commands that do not have an explicit response, can have this reply:
$<E><NL> (E = 0 everything went fine, E=1 the command has one or more wrong parameters,E=2 the command resulted with an error, E=3 the command is illegal, cannot be performed in the current state of things).

- You can connect to the USB virtual COM port using any serial terminal (like Termite) with settings 115200 Baud, 8N1, no handshake.
- Set the terminal so it sends a Line Feed Byte (0x0A) at the end of each command.
- Debug stream on USB is disabled by default.

Once connected to the correct virtual COM port, reset the device by pressing its reset button, or sending the $RST command: you will see the startup text. Then type the command

$DBG<S> sets debug on (1) or off (0)
$DBG1 enables the debug strings.

To check communication at any time, you can ping the device sending the command $???

OUT OF THE BOX SETUP

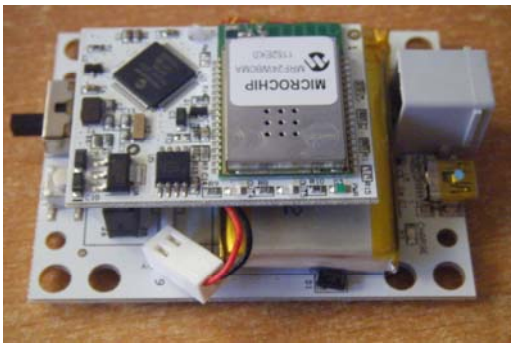
This guide will lead you through the setup of your NXT2WIFI.
You can find all the downloadable material you need on
http://www.robotics.benedettelli.com/NXT2WIFI.htm

To start, you need:

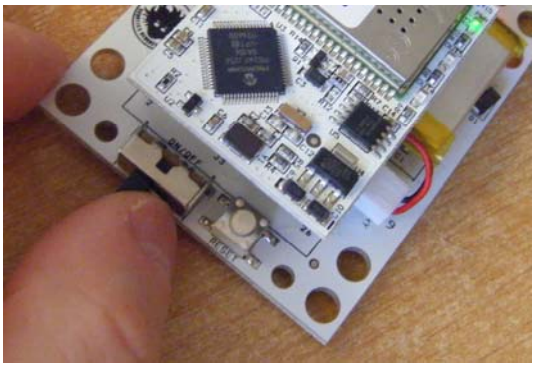
- a USB A-B cable (the one for the NXT)
- a miniUSB cable (the charge and communicate with the NXT2WIFI)
- a NXT connector cable (to connect the NXT2WIFI to the NXT port 4)
- a NXT brick
- the NXT2WIFI device itself

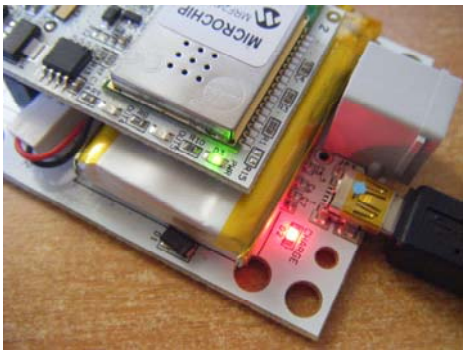
First Time Setup

1. Take the NXT2WIFI out of its ESD-protective envelope.
2. The device is shipped with the battery disconnected for safety reasons. Connect the battery plug to the board (it's one way only, you can't make mistakes!)

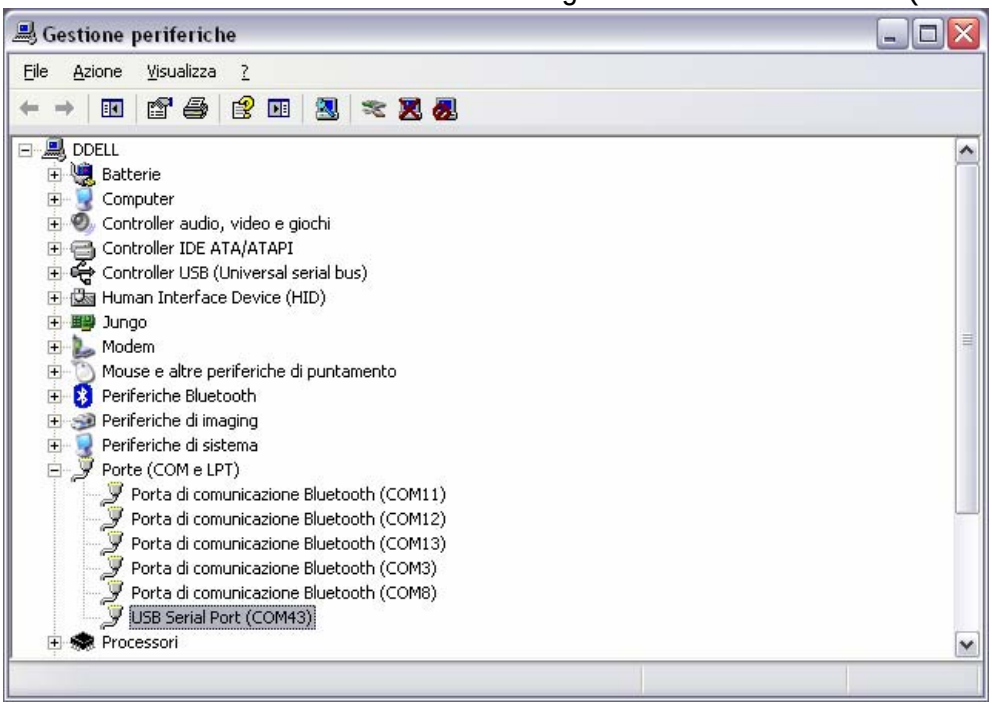


3. The new battery might be not completely charged. Connect the device to your computer using a miniUSB cable: the red "Charge" led is on until the battery is fully charged. Turn the device on using the power switch: the green light on the Wi-Fi module will turn on. The first time you turn the device on, it automatically connects using the DEFAULT connection profile to an adhoc network called "NXT2WIFI". After activation, you have to connect it manually (keep reading).

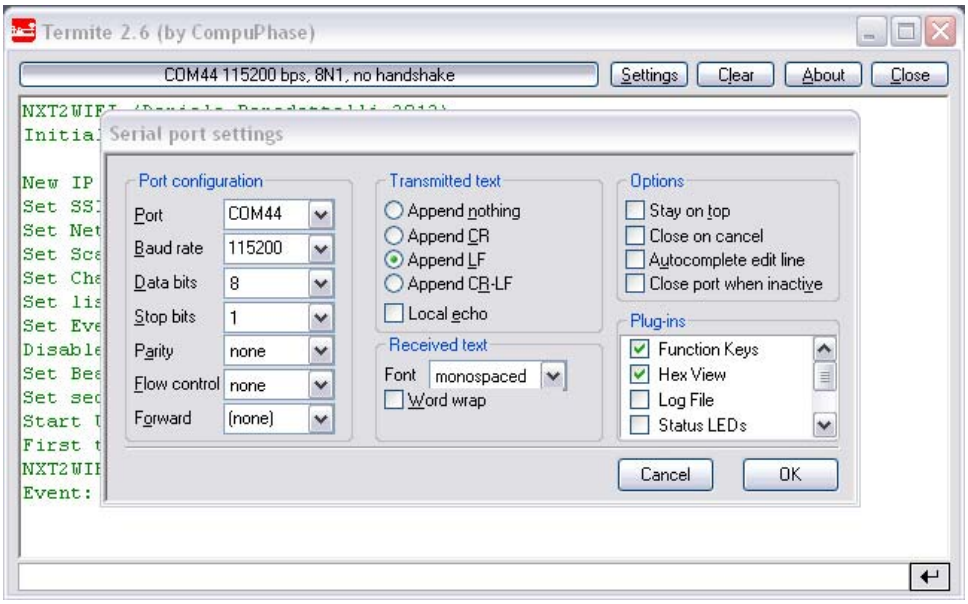




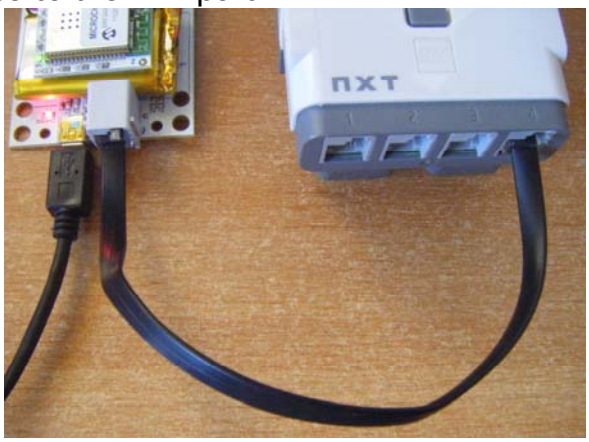
4. You might be asked to install the USB virtual serial port driver. You can download it from http://www.ftdichip.com/Drivers/VCP.htm
   The device will be listed in the Device Manager as a USB Serial Port (COM port)



5. If you are curious, you can connect to the NXT2WIFI using any serial terminal, for example Termite (by CompuPhase), that is a freeware software.
   https://www.google.it/search?q=termite+download&oq=termite+download
   The connection settings are: 115200 baud, 8 data bit, no parity, 1 stop bit. Append LF (0x0A) to every command sent. If you want to see the debug stream, issue the command $DBG1. Every command you send from the computer or from the NXT will be shown in the terminal log.



6. Connect the device to the NXT port 4.



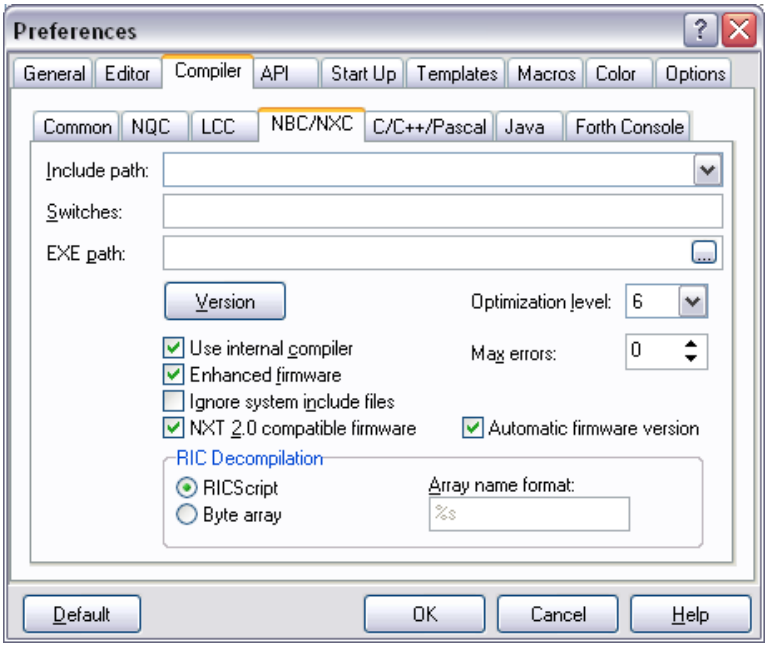
Now the guide changes depending on the programming language you prefer. The supported languages are NXC, ROBOTC and LejOS.

Custom Wi-Fi Setup using NXC

1. Download BricxCC IDE from http://sourceforge.net/projects/bricxcc/files/bricxcc/ and install it.
2. Download the latest BricxCC test release from http://bricxcc.sourceforge.net/test_releases/ and extract it to the main BricxCC installation folder (replacing items)
3. Open BricxCC and connect the NXT using the USB cable A-B.
4. Replace the firmware of the NXT with John Hansen's enhanced firmware (Tools->Download firmware). This is needed to enable serial communication on port 4. The last version (known to work with NXT2WIFI) is http://bricxcc.sourceforge.net/test_releases/lms_arm_nbcnxc_132_20120810_0021.rfw
5. Download the latest NXC library from http://www.robotics.benedettelli.com/NXT2WIFI.htm, unpack it and open `N2W_setup.nxc` in BricxCC. Make sure that the library file `NXT2WIFI.nxc` is in the same folder.
6. Edit the definitions and the CreateCustomWIFI() function to reflect your network configuration

```
#define MY_ADHOC false
#define MY_SSID "Blessed"
#define MY_WPA_KEY "d4d3a089b20d91ef63bd6045457556a9294355bf63e936e0bb0e952f31071f55"
#define MY_WPA_PASS "abcdef12"
#define MY_WEP_KEY_40 "0123456789"
#define MY_DHCP true

// customize your Wi-Fi network settings
void CreateCustomWIFI() {

        WIFI_SetAdHoc(MY_ADHOC); // set to Infrastructure
        WIFI_SetSSID(MY_SSID); // set the network name

// Uncomment the Security option you need
        WIFI_Security(WF_SEC_WPA_AUTO_KEY,MY_WPA_KEY,0);
        //WIFI_Security(WF_SEC_WPA_AUTO_PASSPHRASE,MY_WPA_PASS,0);
        //WIFI_Security(WF_SEC_WEP_40,MY_WEP_KEY_40,1);
        //WIFI_SecurityOpen();

        WIFI_EnableDHCP(MY_DHCP); // enable DHCP
        WIFI_Save(); // store custom settings to retentive memory
}
```

7. Change the NXC compiler settings (Edit->Preferences) to target the current enhanced firmware: check Use Internal compiler, Enhanced firmware, NXT 2.0 compatible firmware, Automatic firmware version checkboxes, set Optimization level to 6. (see snapshot)



8. Download and run the program by pressing CTRL+F5.

Custom Wi-Fi Setup using ROBOTC

1. Download ROBOTC IDE from http://www.robotc.net/ and install it.
2. Download and install the latest ROBOTC Drivers Suite by Xander Soldaat. You can get it from http://botbench.com/blog/category/programming/robotc-drivers/
3. Open, compile and run the program `N2W_setup.c`

Custom Wi-Fi Setup using LejOS

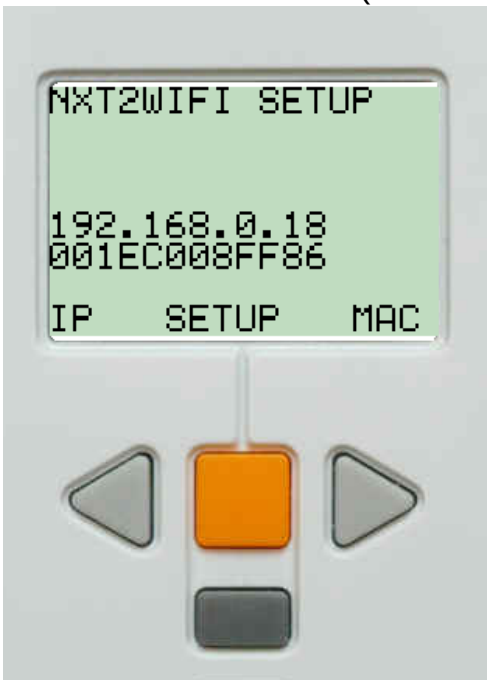
4. The description on how to install the LejOS system and IDE (Eclipse or Netbeans), and how to download the LejOS firmware onto the NXT brick is beyond the scope

of this document. For details about that, refer to the LejOS NXJ tutorial http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

5. Get the `NXT2WIFI.java` class from LejOS SVN repository and put it in your classes source folder (lejos/nxt/addon) in the LejOS source tree. Recompile all the library in order to include the new class.
6. Get the `n2w_setup.java` program from the LejOS SVN repository (until it will be included in next releases)
7. Compile and download the setup program to the NXT brick.

The Setup Program on the NXT

1. Just press the central button to send and save the Wi-Fi configuration to the device. If the configuration you set is correct, the NXT will play a sound when the NXT2WIFI is connected (its red led will turn on).
2. Press left arrow to retrieve current IP address (needed to connect with your browser)
3. Press right arrow to retrieve the MAC address (needed to activate the product)



Activation

NXT2WIFI needs to be activated the first time you use it, and every time you updgrade the firmware, otherwise many functionalities won't work.

You can always ping the device, setup the network and retrieve information, such as MAC address, or firmware version.

To activate the product, do the following:

1. In your web browser address bar, type the IP you got from the Setup Program on the NXT followed by `/activate.htm` for example `http://192.168.0.18/activate.htm`
   If you are using a Windows PC, you can type `http://nxt2wifi/activate.htm` as the address, `nxt2wifi` is the NETbios name of the device.
2. You should see NXT2WIFI activation page. The device MAC address is be displayed there.

3. Alternatively, you can obtain your device MAC from the serial terminal. Enable debug stream with the `$DBG1` command and then issue the command `$MAC` returns MAC=XXXXXXXXXXXX where XX is a byte in hex ascii format, e.g. `MAC=001EC002DA44`
4. In another tab of your browser, go to the online activation webpage `http://dev.openpicus.com/Firmware/NXT2WIFI` enter your NXT2WIFI MAC address, press compute, select and copy that 40-char-long code.
5. Enter (paste) the code on the NXT2WIFI activation page (see above). Alternatively, you can enter the code sending the following command using any serial terminal:
   `$REG?<KEY>` where <KEY> is the 40-char-long code obtained from the online activation code generator. The key is 40 chars long and must contain only the symbols `0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f`
   Example: `$REG?39d22a9dc0882686cc84076c57aac5807e02bce2`
6. If the inserted code is correct, the page will change like this, and in the terminal log you will see the message `NXT2WIFI activated successfully!`

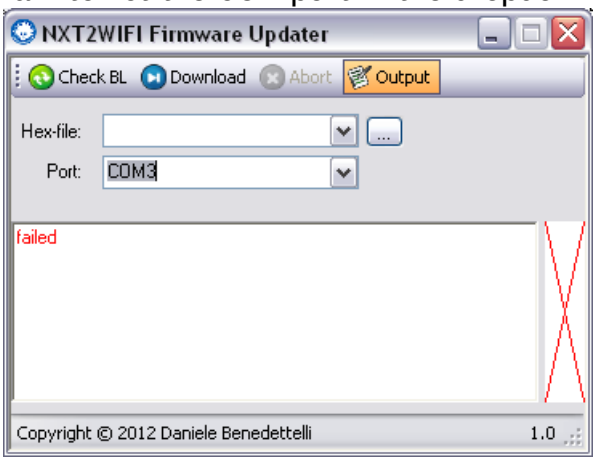
NOTE:
The first time you turn the device on, it automatically connects using the DEFAULT connection profile to an adhoc network called “NXT2WIFI”. After activation, the device

does not connect to any network at startup. You should command it to connect issuing the command $WFC<T> either from NXT or from terminal.

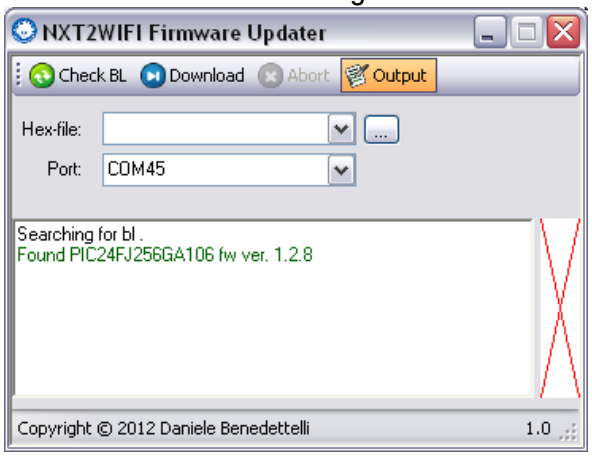
UPDATING THE FIRMWARE

You should keep your NXT2WIFI firmware up-to-date: firmware updates are needed to fix eventual bugs or to add new features. You can find the latest firmware for your device on the official support webpage
http://robotics.benedettelli.com/NXT2WIFI.htm
WARNING: updating the firmware deletes the user settings! After the update, the device must be activated again.

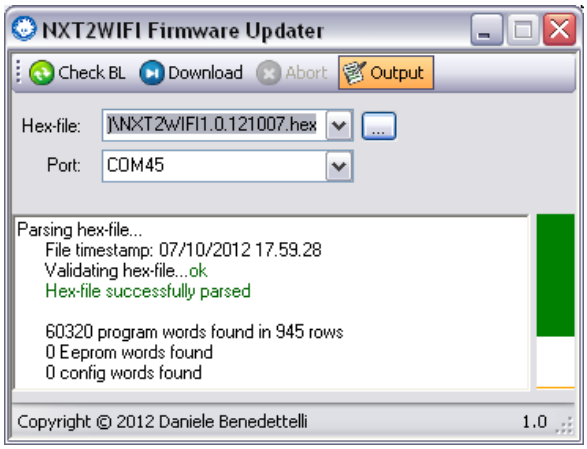
1. Download the NXT2WIFI Firmware Update Tool from the support webpage.
2. No need to install it, just extract the downloaded archive.
3. Connect the NXT2WIFI to your computer using the miniUSB cable, and turn the device on.
4. Double click on the NXT2WIFI-updater.exe program (the device must be already on, for the program to list the COM port in the dropdown menu)



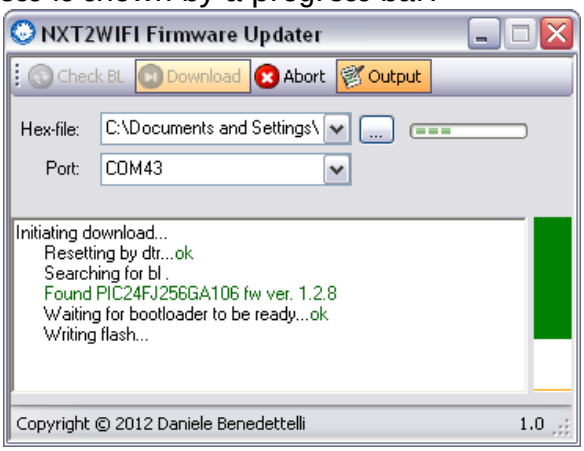
5. Select the COM port of your device from the Port drop down menu. To check if the device is correctly connected and communicating with the Firmware Updater program, press the reset button on the device and click the Check BL button **at once** (actually within 1 second since the reset). You should see the PIC type and the bootloader version in the log window.



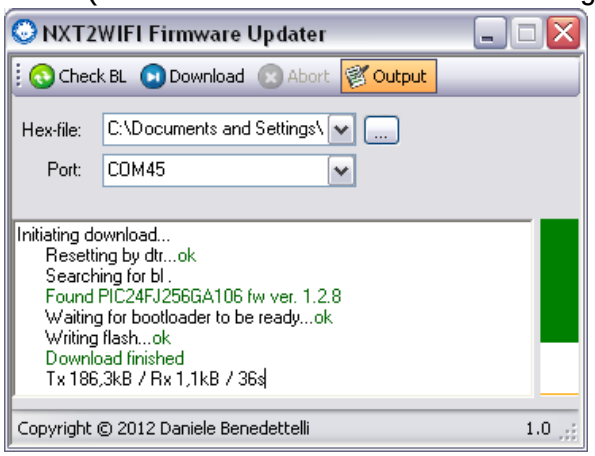
6. Browse for the firmware image clicking on the small button beside the Hex-file drop-down menu. Select the latest .hex file you downloaded from the support page. In the log window you can see the details about the firmware image. The bar on the right shows the memory occupied by the firmware.



7. All is ready to flash the new firmware onto the device. Click the Download button. The process is shown by a progress bar.



8. When the update is complete, the NXT2WIFI will restart. You can then proceed to activate the device (see the Activation section of this guide for help).



NXT2WIFI COMMUNICATION PROTOCOL

Utilities

$SCB<N>?<baudrate> changes the baudrate (need to reconnect PC serial)
<N> is 1 (PC), or 2 (NXT), maximum baudrate is 320400.
e.g. $SCB1?115200

$??? pings the NXT2WIFI. The response to ping are three exclamation marks !!!

$MAC returns MAC=XXXXXXXXXXXX where XX is a byte in hex ascii format, e.g. MAC=001EC002DA44

$FW gets the firmware version string, e.g. FW=1.0.120407

$DBG<S> sets debug on (1) or off (0)

$RST restarts the device

Network

The NXT2WIFI has two Wi-Fi connection profiles: the default one is adhoc, has a DHCP server, has a standard SSID name "NXT2WIFI"; you can change the custom profile from serial, and save it permanently.
There are 4 sockets available, that can be used either as TCP/UDP clients/servers.

Connecting to Wi-Fi Networks

$WFC<T> start WIFI connection using the <T> profile:
0 for DEFAULT (adhoc network with DCHP enabled and SSID "NXT2WIFI")
1 for CUSTOM (user customizable)
e.g. $WFC1 connects using the CUSTOM profile settings.

$WFX disconnect WIFI

$WFQ stop trying to connect

Getting Connection Status

$WFGS gets connection status.
Returns WFGS=<N>

NOT_CONNECTED	0
CONNECTING	1
CONNECTED	2
CONNECTION_LOST	3
CONNECTION_FAILED	4
STOPPING	5
TURNED_OFF	6

Power Saving

$WFH go into Hibernation mode, to save battery.

$WFO exit Hibernation mode. You will need to reconnect the device using $WFC<T>

Scanning for networks

$WFSCAN scan WIFI networks. Must be issued when WIFI is not connected to any network.

$WFNN Returns the number of retrieved networks WFNN=<P>, <P> is 255 when scan is pending. You can retrieve network info using the $WFNI command.

$WFNI<N> get info of the <N> network obtained with a scan.
Returns in sequence
BSSID=<BSSID>
SSID=<SSID>
CHN=<channel>
SGN=<signal>
SEC=<security>
TYPE=<type>
BCN=<beacon>
PRE=<preamble>

Managing the custom profile settings

$WFKD delete custom profile

$WFKE check if WF_CUSTOM profile exists, returns WFKE=<B> 1 if valid data exists, 0 if does not exist, e.g. WFKE=1

$WFKL load custom profile

$WFKS save custom profile to permanent memory

Changing the custom profile settings

- Network type (adhoc or infrastructure)

$WFE?TYPE=<E> 1 if ADHOC, 0 if INFRASTUCTURE e.g. WFE?TYPE=0

- IP address of the device

$WFE?IPAD=<address> e.g. WFE?IP=192.168.1.100

- Primary DNS server

$WFE?DNS1=<address> e.g. WFE?DNS1=192.168.1.1

- Secondary DNS server

$WFE?DNS2=<address> e.g. WFE?DNS2=192.168.1.1

- Default gateway

$WFE?GWAY=<address> e.g. WFE?GW=192.168.1.1

- Subnet mask

$WFE?MASK=<address> e.g. WFE?MASK=255.255.255.0

- Netbios name

$WFE?NAME=<name> e.g. WFE?NAME=NXT2WIFI

- SSID

$WFE?SSID=<name> e.g. WFE?SSID=NXT2WIFINet

- DHCP client enabled or not

$WFE?DHCP=<E> 1 if enabled, 0 if disabled e.g. WFS?DHCP=1

Getting the current IP

$WFIP get the current IP (DHCP may have assigned a new one)

$WFAA checks if DCHP assigned an address. This is helpful to poll the status of DHCP client, waiting for the new address to be assigned, before calling $WFIP.
Returns WFAA=<S>, <S> being 1 if new address was assigned, 0 if not assigned yet.

Changing the Custom Profile Security settings

The following command allow to change custom profile security settings

$WFS?<MODE>:<KEYPASS>:<KEYIND>

NOTE!
<KEYPASS> and <KEYIND> can be omitted if MODE is 0 (OPEN)
<KEYIND> can be omitted if MODE is not WEP
the key length is computed as string length for passphrases, and as half the number of chars for hex keys.
keys must be even-long and must contain only the symbols
0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f

Parameters
<MODE> is the security mode. Valid security mode are the following:
0 = no security.
1 = WEP security, with 40 bit key (64bit) 10 hex chars (5Bytes)
2 = WEP security, with 104 bit key (128 bit) 26 hex chars (13Bytes)
3 = WPA-PSK personal security, the user specifies the hex key.
4 = WPA-PSK personal security, the user specifies only the passphrase.
5 = WPA2-PSK personal security, the user specifies the hex key.
6 = WPA2-PSK personal security, the user specifies only the passphrase.
7 = WPA-PSK personal or WPA2-PSK personal (autoselect) with hex key.
8 = WPA-PSK personal or WPA2-PSK personal (autoselect) with pass phrase.

<KEYPASS> the key or passphrase for the network. A key must be specified also for open connections (you can put a blank string, like "").

<KEYIND> index of the key (used only for WEP security)

NOTE!
For WPA/WPA2 with passphrase, the NXT2WIFI must calculate the hex key the first time it connects. The calculation is long and difficult, so it will take about 30 seconds to connect. Once connected, the key is saved, and it will connect at once!

Checking current custom profile settings

The following commands allow to check the custom profile (address is always in the format xxx.xxx.xxx.xxx, without 0 padding)

$WFKC? (no argument) returns all the following.

- Network type (adhoc or infrastructure):
$WFKC?TYPE returns TYPE=<E> 1 if ADHOC, 0 if INFRASTUCTURE

- IP address of the device:
$WFKC?IPAD=<address> returns IPAD=<address>

- Primary DNS server:
$WFKC?DNS1 returns DNS1=<address>

- Secondary DNS server:
$WFKC?DNS2 returns DNS2=<address>

- Default gateway:
$WFKC?GWAY returnrs GWAY=<address>

- Subnet mask:
$WFKC?MASK returns MASK=<address>

- Netbios name:
$WFKC?NAME returns NAME=<name> e.g. NAME=NXT2WIFI

- SSID:
$WFKC?SSID returns SSID=<name> e.g. SSID=NXT2WIFINet

- DHCP client enabled or not:
$WFKC?DHCP returns DHCP=<E> 1 if enabled, 0 if disabled

- Security settings:
$WFKC?SCRT returns SCRT=<MODE>:<KEYPASS>:<KEYLEN>:<KEYIND>

Example configuration 1 (ADHOC, no security)

Send the following commands, one at a time (remember to have the line feed at the end, and the debug stream turned on to check the device responses)

```
$WFX
$WFKD
$WFS?0: :0:0 (notice the space!)
$WFE?TYPE=1
$WFE?IPAD=192.168.0.1
```

```
$WFE?MASK=255.255.255.0
$WFE?DHCP=1
$WFE?SSID=ADHOCFP
$WFE?NAME=NXT2WIFI
$WFKS
$WFC1
```

Example configuration 2 (INFRASTRUCTURE, WPA2-PSK)

Send the following commands, one at a time (remember to have the line feed at the end, and the debug stream turned on to check the device responses)

```
$WFX
$WFKD
```

WITH HEX KEY

```
$WFS?5:d4d3a089b20d91ef62bd6245467556a9294355bf63e936e0bb0e952f31051f35
```

OR WITH PASSPHRASE

```
$WFS?6:cippalippa:10

$WFE?SSID=Danny
$WFE?DHCP=1
$WFE?TYPE=0
```

(the following are optional)

```
$WFE?MASK=255.255.255.0
$WFE?GWAY=192.168.0.1
$WFE?DNS1=192.168.0.1
$WFE?NAME=NXT2WIFI
$WFKS
$WFC1
```

Example configuration 3 (INFRASTRUCTURE, WEP 40bit)

Send the following commands, one at a time (remember to have the line feed at the end, and the debug stream turned on to check the device responses)

```
$WFX
$WFKD
$WFS?1:d4d3a089b2:1
$WFE?SSID=Danny
$WFE?DHCP=1
$WFE?TYPE=0
$WFKS
$WFC1
```

WIFI Connection Failure codes (only displayed in computer terminal)

JOIN FAILURE	2
AUTHENTICATION FAILURE	3
ASSOCIATION FAILURE	4
WEP HANDSHAKE FAILURE	5
PSK CALCULATION FAILURE	6
PSK HANDSHAKE FAILURE	7
ADHOC JOIN FAILURE	8
SECURITY MISMATCH	9
NO SUITABLE AP FOUND	10
RETRY FOREVER NOT SUPPORTED	11

Forcing ARP request

$ARP?<IP> force an ARP request for the specified <IP> address in the format xxx.xxx.xxx.xxx, e.g. $ARP?192.168.0.12

TCP sockets

The device provides 4 sockets with 700Bytes input buffer, and 100Bytes output buffer. All commands are non-blocking. You can create a new socket using an already used ID, the old socket will be closed without complaining. The port 0 cannot be used. You can pass the address of the socket as a numeric IP, or as the name. The DNS will retrieve the address automatically.

$TCPOC<N>?<IP>,<PORT> opens socket <N> as TCP client of <IP> server address, <PORT> server port, e.g. $TCPOC1?10.23.0.12,1234
$TCPOC1?http://www.google.com,80
Returns TCPOC<N>=<OK> 1 is success, 0 if failure, e.g. TCPOC1=1

$TCPOS<N>?<PORT> opens socket <N> as TCP server with <PORT> , e.g. $TCPOS1?1234
Returns TCPOS<N>=<OK> 1 is success, 0 if failure, e.g. TCPOS1=1

$TCPD<N> detaches the client from server socket <N>(1,2,3,4, 0 detaches all sockets), e.g. $TCPD1
Returns TCPD<N>=<OK> 1 is success, 0 if failure (maybe socket <N> isn't a server)

$TCPX<N> closes the socket <N>, e.g. $ TCPX1 (<N>= 0 closes all sockets)
Returns TCPX<N>=<OK> 1 is success, 0 if failure, e.g. TCPX1=1

$TCPR<N>?<LEN> read <LEN> Bytes of data from socket <N>. Returns them in <data>
If <N> is zero,it first verifies how many bytes are available to be read and returns the <datalen> and <data> byte array
$TCPR1?0 reads all bytes available on socket 1
$TCPR2?10 reads 10 bytes from socket 2
Returns TCPR<N>=<datalen>,<data> e.g. TCPR3=13,Hello World!(\0)

$TCPL<N> get the available length of incoming message in Bytes of socket <N>
Returns TCPL<N>=<LEN> e.g. TCPL2=10

$TCPW<N>?<datalen>,<data> write <message> to socket <N> (1,2,3,4)
Returns TCPW<N>=<LEN> number of bytes written
e.g. $TCPW1?hello should return TCPW1=5

$TCPI<N> retrieves info on socket <N>
Returns TCPI<N>=<T><C> where <T> is 'C' for client, 'S' for server, 'F' for free (unused), <C> is 1 if connected , 0 if disconnected.

$TCPF<N> flushes socket <N> input buffer
Returns TCPF<N>=<OK> 1 is success, 0 if failure

$TCPSM<N> gets the MAC address of the remote socket <N>
Returns TCPSM<N>=XXXXXXXXXXXX where each XX couple is a byte in hex ascii format, e.g.
TCPSM2=0022F5DD97A3
Returns TCPSM<N>=000000000000 if socket <N> is closed.
Returns TCPSM<N>=000000000001 if socket <N> is open, but no client is connected

$TCPSI<N> gets the IP address of the remote socket <N>
Returns TCPSI<N>=<IP_ADDR> where <IP_ADDR> is in the dotted format
TCPSI1=192.168.0.23
Returns TCPSI<N>=0.0.0.0 if socket <N> is closed.
Returns TCPSI<N>=0.0.0.1 if socket <N> is open, but no client is connected

UDP sockets

The device can handle 4 UDP sockets with a 200Byte input buffer.
The 4 available sockets can be opened as servers, as clients or broadcast. You can create a new socket using an already used ID, the old socket will be closed without complaining. UPDR reads might put multiple packets together in the same buffer. Writing on UDP server socket sends data to the last UDP client that sent data. All commands are non-blocking. The port 0 cannot be used.

$UDPOB<N>?<PORT> opens socket <N> as UDP broadcast on spec. <PORT>, e.g.
$UDPOB1?123
Returns UDPOB<N>=<OK> 1 is success, 0 if failure , e.g. UDPOB1=1

$UDPOC<N>?<IP>,<PORT> opens socket <N> as UDP client of <IP> server address, <PORT> server port, e.g. $ UDPOC1?10.23.0.12,1234
Returns UDPOC<N>=<OK> 1 is success, 0 if failure, e.g. UDPOC1=1

$UDPOS<N>?<PORT> opens socket <N> as UDP server with <PORT> , e.g.
$UDPOS1?1234
Returns UDPOS<N>=<OK> 1 is success, 0 if failure, e.g. UDPOS1=1

$UDPX<N> closes the socket <N>, e.g. $ UDPX1 (<N>= 0 closes all sockets)
Returns UDPX<N>=<OK> 1 is success, 0 if failure, e.g. UDPX1=1

$UDPR<N>?<LEN> read <LEN> Bytes of data from socket <N>. Returns them in <data>
If <N> is zero,it first verifies how many bytes are available to be read and returns the data length and a <data>, e.g.
$ UDPR1?0 reads all bytes available on socket 1
$ UDPR2?10 reads 10 bytes from socket 2
Returns UDPR<N>=<datalen>,<data> e.g. UDPR3=13,Hello World!(\0)

$UDPL<N> get the available length of incoming message in Bytes of socket <N>
Returns UDPL<N>=<LEN> e.g. UDPL2=10

$UDPW<N>?<datalen>,<data> write <message> to socket <N> (1,2,3,4)
Returns UDPW<N>=<LEN> number of bytes written

e.g. $UDPW1?hello should return UDPW1=5

$UDPI<N> retrieves info on socket <N>
Returns UDPI<N>=<T> where <T> is 'C' for client, 'S' for server, 'O' for broadcast, 'F' for free (unused)

$UDPF<N> flushes socket <N> RX buffer
Returns UDPF<N>=<OK> 1 is success, 0 if failure

$UDPV<N> check if socket <N> RX buffer overflowed
Returns UDPV<N>=<OK> 1 if overflow occurred, 0 if not occurred

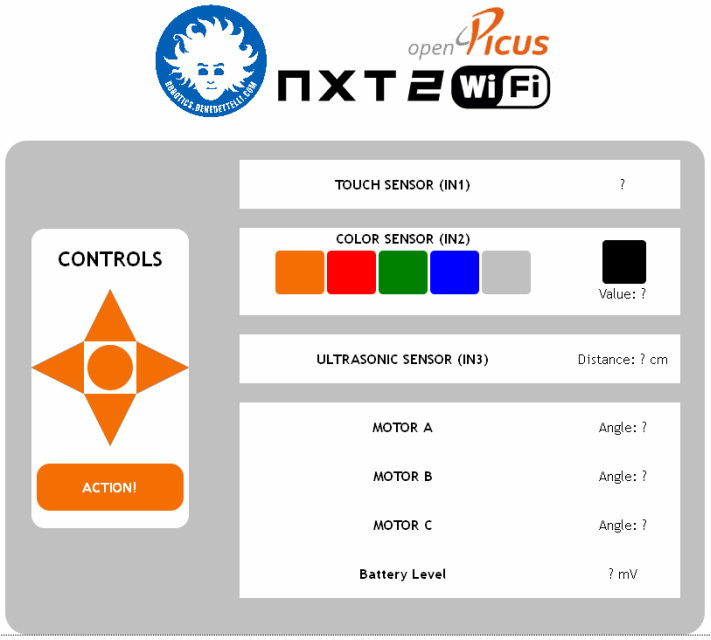
Realtime Clock

The Real Time Clock Calendar is a PIC24F module that permits to keep track of the time using the secondary oscillator. It synchronizes automatically with a time server, when connected to a network that has internet access.

$GTIM?<K> get the current time with <K> hours of offset from GMT (can be positive or negative
Returns GTIM=YYYY-MM-DD/hh:mm:ss
e.g. GTIM=2012-05-23/19:32:10

GRAPHIC WEBSERVER

The webserver pages are stored in NXT2WIFI external 2MB Flash memory. You can access the main page index.htm from any browser-enabled device, by typing the IP address of the NXT2WIFI in the browser address field. In Windows networks, you can type the NETbios name of the device (default name is nxt2wifi).



Besides the main page, there are special pages such as config.htm and activate.htm.
NXT2WIFI always listens and executes the commands sent over the serial ports (NXT or computer), responding to the requests. The only spontaneous messages are sent to the

NXT when events are generated from the webpage (pressing or releasing buttons, moving sliders, toggling checkboxes). This might cause collisions, since RS485 is half duplex communication. To toggle the spontaneous messages on and off, use the command $SRV<X> where <X> can be 1 to turn it on, 0 to turn it off. Everytime you turn the NXT2WIFI on, the webserver spontaneous messages are enabled.

Managing Webserver Events

Whenever the user interacts with the webpage, the NXT2WIFI sends a special Direct Command to the NXT.

0x00 (HS_NXT_ADDRESS)	0x80 DC_NO_REPLY	0x14 WEB_EVENT	0-255 (0,1,2) controlType	0-255 control ID	0-255 event	0-255 value

`controlType` can be
0 if the event is generated by a slider
1 if the event is generated by a button
2 if the event is generated by a checkbox
The first byte in the header is needed for future compatibility with a special feature in John Hansen's enhanced firmware for NXT (http://bricxcc.sourceforge.net/), that allows the NXT to execute direct commands received on port 4 instead of from the Bluetooth.

Retrieving Webpage Data

If you need to get the data from a webpage widget, issue the command `$WEBGET<TYPE>?<ID>` where `<TYPE>` is the widget type and `<ID>` is the widget ID

Widget	TYPE
Button	0
Slider	1
Checkbox	2
Label	3
Bargraph	4

The command returns
`WEBGET=<data>`
Notice that if the webpage widget does not exist or has never been used (touched on page or updated from the device), the device will return junk data.

Updating Webpage Widgets Data

On the webserver page you can show data coming from the NXT (motors, sensors, variables, messages), and command the NXT to perform actions.
The NXT can update the state and the content of the webpage elements such as labels (strings), sliders and bargraphs, buttons and checkboxes. that are retrieved as strings.

You can send strings, sensor and motor data to NXT2WIFI to be shown on the webserver page. Also it can update various controls status with this simple protocol:

Widget	Direction	Data type	Quantity
Label	output	string	20 (64Bytes-long)
Button	input/output	bool	20
Checkbox	input/output	bool	20
Slider	input/output	int	20
Bargraph	input/output	int	20

$WEBLBL<N>?<datalen>,<data>[0x1B] where <N> is the input field ID, <datalen> is the length of <data>, usually a text string containing numbers or a message.
Upto 10 64-Byte strings can be sent; the ID can be 0-9.

$WEBBTN<N>?<S> where <N> is the button ID, and <S> is the boolean status, 1 or 0. Buttons can be used to display two different images, so they can become a graphic indicator with two states.

$WEBCHK<N>?<S> where <N> is the checkbox ID, and <S> is the boolean status, 1 fror checked or 0 for unchecked.

$WEBSLD<N>?<V> where <N> is the checkbox ID, and <v> is the integer value for the slider position. Maximum and minimum values are set in the web editor.

$WEBBAR<N>?<V> where <N> is the checkbox ID, and <v> is the integer value for the bar fill level. Maximum and minimum values are set in the web editor.

$WEBCLR resets all webpage fields to zero (or empty).

CUSTOMIZING THE WEBSERVER

The webserver pages can be fully customized to your needs. You can display images, add buttons, sliders, forms and other widgets, and make them generate events that are sent to the NXT2WIFI, to command your robot. You can also display sensor values on the webpage.

Creating your webpage using the Drag & Drop Web Editor

The Web Editor is a simple and powerful tool that allows you to create a rich webpage without having to write a single line of code! You can add labels, buttons, checkboxes,
Web Editor tool is coming soon!

Creating your custom webpage

While waiting for the Web Editor tool to be ready, you can create your custom webpage using any html editor or text editor. You can use the provided webpage source code as reference. You can get the webpage source code from the support page
http://robotics.benedettelli.com/NXT2WIFI.htm
In the archive you find two folders:

- *Default webserver* includes the source code for the default webserver (created manually)
- *Custom webserver* includes the source code for a custom webserver. The index.htm was created using the Web Editor.

After making your edits, you can upload the project (and create a .bin image) using the Webpages Converter Tool (see section below).
Before uploading, make sure that in the folder are present together with index.htm:

- scripts, images, img folders
- activate.htm : the product activation page
- config.htm : the Wi-Fi connection settings page
- sld.cgi, chk.cgi, btn.cgi
- activation.xml, status.xml
- jquery-1.5.min.js, json2.min.js, lib.min.js, jquery-ui-1.8.21.custom.min.js
- content.css, jquery-ui-1.8.21.custom.css

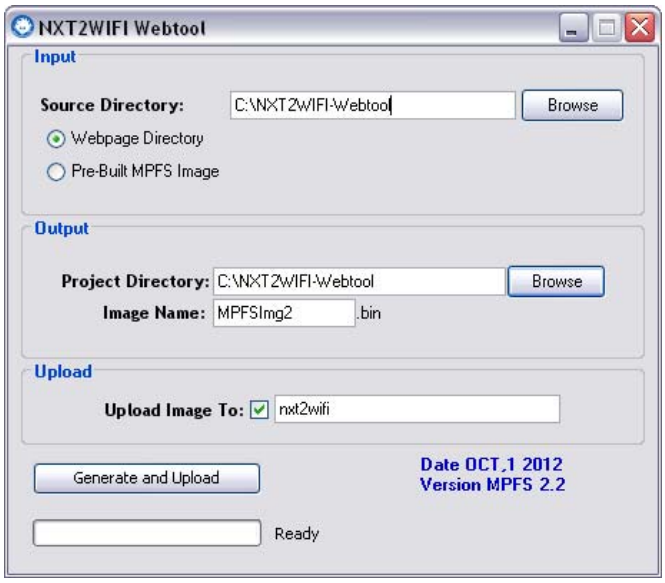
UPLOADING WEBPAGES

Replacing the webserver pages is very easy, using the provided NXT2WIFI Webpages Converter. This tool allows you to pack the webserver project folder into a .bin image and upload it to the device via Wi-Fi. You can get the tool from the support page
http://robotics.benedettelli.com/NXT2WIFI.htm
To upload a new webserver image, do the following:

1. Make your NXT2WIFI connect to your Wi-Fi network, the same your computer is connected to. To do this, you can use the terminal, or the NXT setup program. (see sections above)
2. Open the NXT2WIFI Webpages Converter tool. Hovering on the various controls will make hint boxes to pop up.

Create webserver from project folder

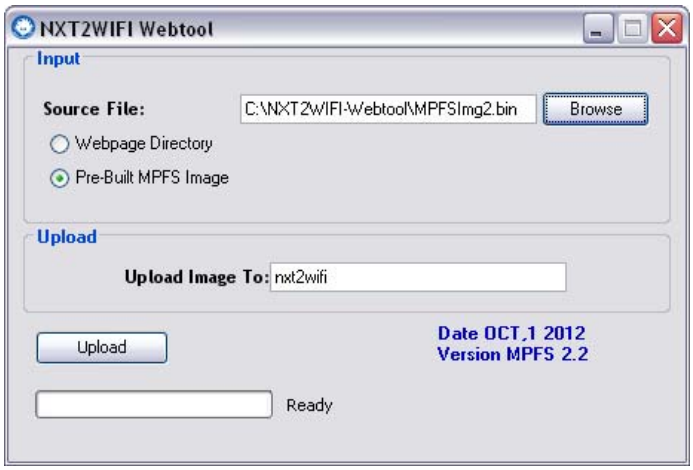
1. You can create the .bin image and upload it, by selecting the WebPage Directory radio button.



2. In the Input panel, click on Browse button to select the folder that contains all the webserver files.
3. In the Output panel, you can specify the Project Directory where to save the .bin image, and the Image Name. You can safely keep the defaults.
4. In the Upload panel, check the box beside the address field if you want to upload the newly created .bin image to the NXT2WIFI device. If you don't check the box, the tool will just pack the project folder into a .bin image, without uploading it to the device. This is useful if you want to share a webserver image, without distributing your source code.
5. You can edit the IP address of your device. You can get the IP address from the NXT setup program or by issuing the commands `$DBG1` and `$WFIP` on the terminal. If you are in a Windows network, you can type the NETbios name of the device instead of the IP address (the default name is nxt2wifi).
6. Check again that your NXT2WIFI is connected to the same network as your computer. If not, make it connect using the terminal, or the NXT setup program.
7. Click the Generate and Upload button and wait for completion.
8. Upon successful upload, a log screen will pop up. If something goes wrong, an error diagnostic message will pop up.

Upload webserver image file

1. To upload a previously created .bin image select the Pre-built MPFS Image radio button.



2. In the Input panel, browse to the .bin image you want to upload.
3. In the Upload panel, you can edit the IP address of your device. You can get the IP address from the NXT setup program or by issuing the commands `$DBG1` and `$WFIP` on the terminal. If you are in a Windows network, you can type the NETbios name of the device instead of the IP address (the default name is nxt2wifi).
4. Check again that your NXT2WIFI is connected to the same network as your computer. If not, make it connect using the terminal, or the NXT setup program.
5. Click the Upload button and wait for completion.
6. Upon successful upload, a log screen will pop up. If something goes wrong, an error diagnostic message will pop up.