



Application Note: OP-AN-0009

Webserver with Wi-Fi Parameter Configuration

Device: Openpicus Flyport Wi-Fi

IDE version: 2.1

Author - version: Simone Marra - v1.1

External libs: -

Connections: -

Description:

One of the great features of the openPICUS framework with Wi-Fi devices is the ability to handle 2 different configurations for the Wi-Fi network parameters.

The “**WF_DEFAULT**” can be configured inside the IDE Wizard tool, and it can not be changed at run-time. It is a fixed configuration that must be stored inside the microcontroller with the download of the firmware.

The second configuration, called “**WF_CUSTOM**”, can be changed at run-time, saved inside the internal Flash memory of the microcontroller, loaded and deleted. This configuration is used to let the user change the information related to the router or access point where the Flyport Wi-Fi has to connect itself, or if Flyport should create an adhoc connection, etc.

The “**WF_CUSTOM**” parameters can be handled by a user task, and the user can provide for example serial commands to change them when the “*taskFlyport*” firmware is executed, but this solution requires a hardware connection, so the customization should be done in the vicinity of any Flyport module.

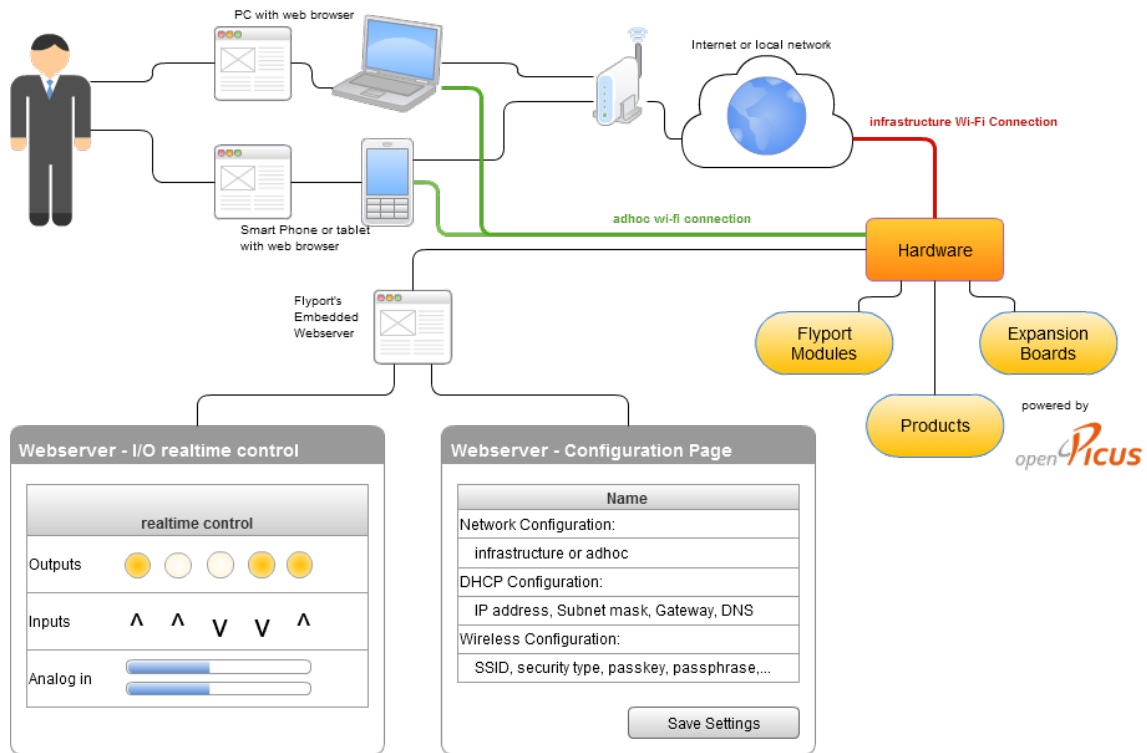
The solution for this problem could be to permit the change of “**WF_CUSTOM**” parameters using the Wi-Fi connection. In this application note it is shown how a smart configuration web page can be integrated to configure Flyport Wi-Fi Network parameters.

The *configuration page* has some input forms to enter the new parameters, and some scripts to check the validation of the data added by the user. In fact, if a parameter is filled with an incompatible value, the browser will automatically show a message of what parameter must be corrected. In this way, the human errors are reduced.

The proposed solution also has a configurable starting time for the firmware, where the Flyport Wi-Fi connects only to its **WF_DEFAULT** configuration (the parameters entered inside the IDE Wizard settings); On the other hand, when the code is in execution, the pressure of an input button for more than 3 seconds forces the microcontroller to delete the **WF_CUSTOM** customizable parameters, and after connects again to the “**WF_DEFAULT**” settings.

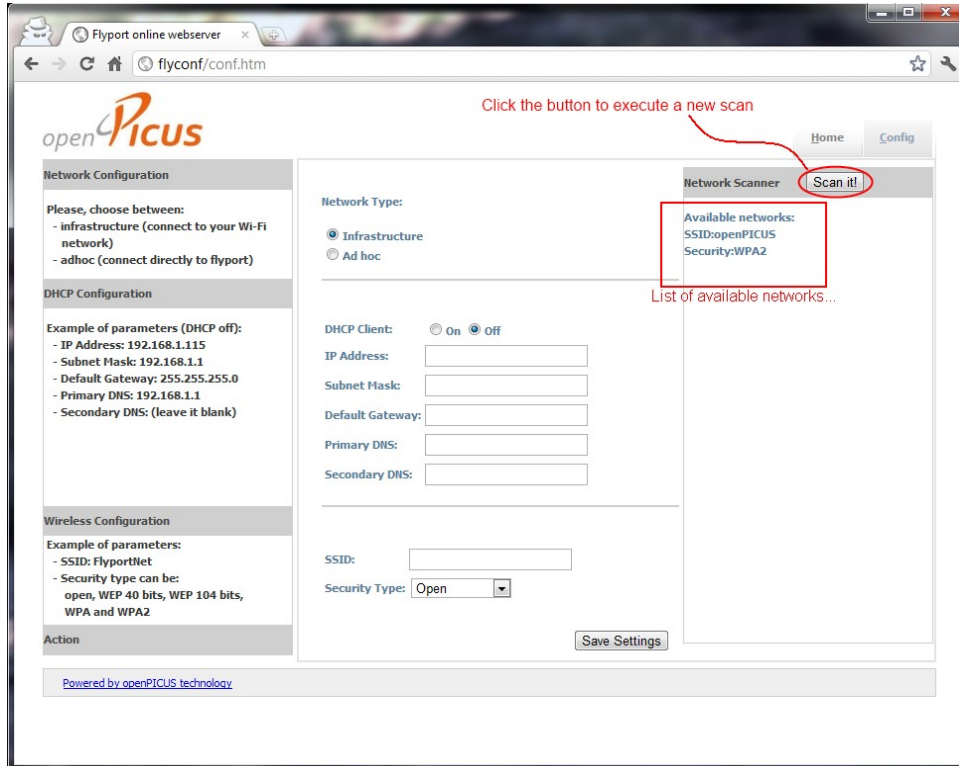
In a typical application with more Flyport Wi-Fi modules used, or with a physical position not easily reachable from the user, the replacement of Wi-Fi routers is an easy task. The customer can change the configuration of every Flyport one by one, using the web browser of a PC or SMARTPHONE or TABLET, without the worry about the hardware cable connections.

Using the Flyport Wi-Fi embedded web server, the user can anytime not only control the hardware I/O status, but also change the Wi-Fi network parameters.



The above picture shows the typical use of the webserver with the configuration page. The user can access to the Flyport Wi-Fi using a PC, a Smart-Phone, or any other internet enabled device. Surfing with a web browser to the Flyport Wi-Fi IP address or host name, the user can access to the first page of the webserver (the index.htm file); this page is related to the "I/O real-time control", and has a tab to access the "Configuration Page" (the conf.htm file).



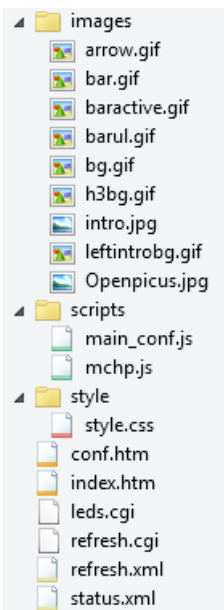


The *conf.htm* page has a router-like interface, with all the customizable parameters accessible by the user.

The section related to the SSID and the Security type changes dynamically in relation of the security type chosen by user, and enables the forms to fill with the pass-phrases and passkeys only where expected.

On the right side of the page, it is located a section called "Network Scanner". This tool reports a list of all the Wi-Fi network reachable from the Flyport Wi-Fi module. The list is updated by the pressure of the button "Scan it!" inside the web-page, or with the pressure of the button at the pin "scanPinNumber" (defined in firmware) of the module.

Webserver Files structure:



The two pages of webserver are "index.htm" and "conf.htm".

Both pages share some files inside the *images* folder and the *style.css* file.

The *index.htm* page uses the script *mchp.js*, inside the "scripts" folder, and the *leds.cgi* and *status.xml* files too.

The *conf.htm* page uses instead the scripts *main_conf.js* and *mchp.js*, inside the "scripts" folder, and *refresh.cgi* and *refresh.xml* files too.

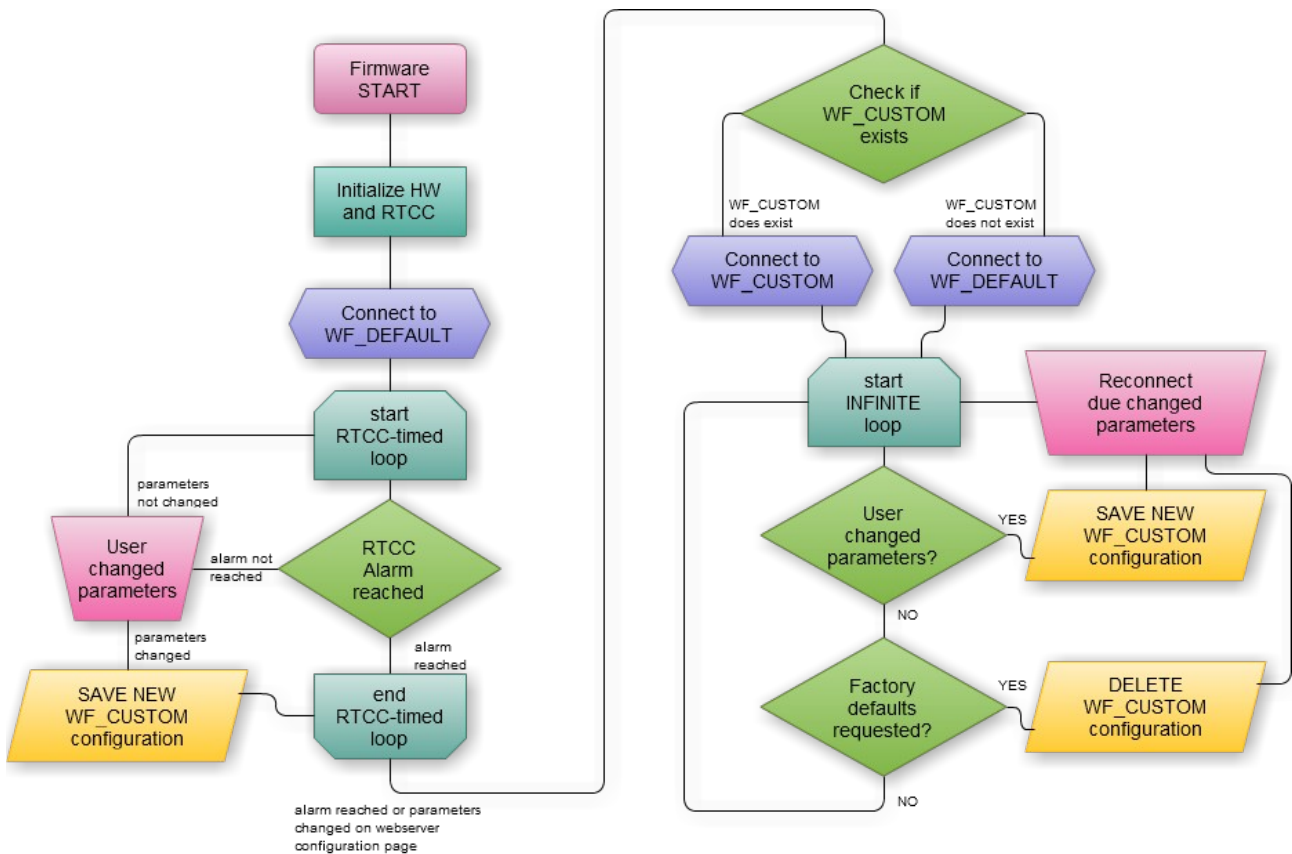
The *mchp.js* file is the scripting file responsible of the handle of I/Os control and any other related hardware/firmware resource that user would like to embed into the webpage. The code of *index.htm*, in conjunction with *status.xml* and the microcontroller *firmware* can be modified by the user to show and to control the other dynamic resources, like sensor values (temperature, humidity, etc...), relays or PWMs (for example to turn on/off a light, or take advantage of a dimmer to control the brightness of a lamp). The script updates also the network list of the *Network Scanner* tool inside the *config.htm*, using *refresh.xml*

The **main_conf.js** is the scripting file responsible to pass the Wi-Fi parameters to the Flyport's firmware, using a POST method; it checks if a data is correct for every specific parameter, for example it checks if an IP address is written in the correct format (clearly it can't check if an IP address is correct, but only if it was written in a correct format). This file is very important since it checks the human input, so it can reduce a lot the probability of errors (like bad typing, or parameter exchange...).

NOTE: When the user press the "Save Parameters" button on the bottom-right side of the configuration page, the parameters validation is done by a JavaScript, so all the calculation is done by the browser (the PC or the Smart-Phone or the Tablet, etc...but outside of the Flyport). If the data is verified and correct, they are sent to the Flyport firmware *HTTPApp* with the using of a POST method.

Firmware of Flyport:

The Flyport has to execute some functions to store, to change and to reload the *WF_CUSTOM* parameters. When the user changes the network parameters inside the web-server configuration page, the Flyport receives data and store them inside the *WF_CUSTOM* configuration.



Above is a schematic diagram of the flow of the *flyportTask.c* execution firmware.

After the initializations of hardware and RTCC module, the Flyport Wi-Fi connects to *WF_DEFAULT* parameters to permit the user to configure the network parameters, before using the customizable configuration. The Flyport waits for a time (customizable in firmware) before to proceed with the code execution. The time is configurable with two *const int* variables: **setuptimeSec** and **setuptimeMin**. Changing those variables the developer can change the RTCC alarm, and so the waiting time for this "setup pause" phase of the firmware. At the end of this process, the RTCC module is stopped.

The next step on firmware execution is to choose the network parameters to connect with.

If the WF_CUSTOM configuration does exist, it means that a custom configuration was previously saved inside the flash memory of the Flyport Wi-Fi microcontroller. In this case, the firmware connects the module to the *customized* network parameters.

In case WF_CUSTOM does not exist it could mean that it is the first start of Flyport Wi-Fi, or that WF_CUSTOM configuration was never saved. In this case, the firmware connects the module to the *default* network parameters.

After the Wi-Fi connection successful created, the firmware enters in a “infinite loop” and checks at every cycle the state of the variable “ParamSet” and the state of the p5 input pin.

If the input pin is tied low for a time greater then “waitbuttonSec” seconds, the WF_CUSTOM configuration is deleted, and the Flyport returns in its “factory default” (or first start) state.

The “ParamSet” variable is changed by the HTTPApp.c file if the user provide a new configuration inside the configuration webpage of Flyport Wi-Fi's webserver. The function responsible of the handle of the update of parameters is the HTTPExecutePost that changes all the WF_CUSTOM parameters, and the saving of the new configuration is done by the taskFlyport.c.

Conclusions:

A special attention should be payed when you choose the WF_DEFAULT parameters. This configuration is prepared inside the openPICUS IDE, using the Wizard tool. All the information related to the WF_DEFAULT configurations is written directly inside the microcontroller firmware, and cannot be changed at run-time. The only way to change the default configuration is to prepare and download again the microcontroller firmware inside the Flyport Wi-Fi, but it needs a hardware connection between a PC and the Flyport Wi-Fi module.

In the specific context of this application note, it is suggested to use the WF_DEFAULT parameters as an “ad hoc” connection, to be sure that Flyport Wi-Fi is reachable from every PC; Due to this specific network type, Flyport Wi-Fi acts as a one-to-one access point and the customer can configure the parameters easily. This solution is very comfortable in an environment with few Flyport Wi-Fi modules, so every module can change its network configuration one by one.

A solution for an architecture with more modules could be to setup the default parameters to access to a “infrastructure” network, with a access point with fixed SSID and protection settings (like the safe WPA2), specifically designed for setup purpose (it is not necessary to have a internet connection). In this case, every Flyport can connect with default parameters at the “always accessible” access point, and also the customer can connect to the same network to setup all the Flyport Wi-Fi modules to connect to a new network.

In this case, one access point is used to setup the Flyport Wi-Fi modules, and another access point is used on the specific application, maybe with internet connection available.

The “Network Scanner” tool embedded inside the web-server configuration page, helps the customer to choose the network parameters, and to check them if a specific network is reachable from the Flyport Wi-Fi module.