The configuration file allows to setup the WebIOPi server when using webiopi command or service. The syntax is the same as windows INI files. Several section (section.html) containing key=value pairs. The webiopi service use **/etc/webiopi/config** as a configuration file.

HTTP server

You can enable or disable the HTTP as well as changing the port. You can also change the passwd file location and folder to look up for HTML files.

```
[HTTP]
enabled = true
port = 8000
passwd-file = /etc/webiopi/passwd
doc-root = /home/pi/webiopi/examples/scripts/macros
welcome-file = index.html
```

CoAP server

You can also enable or disable the CoAP server and change its port. By default, multicasting is enabled, you can disable it here.

```
[COAP]
enabled = true
port = 5683
multicast = true
```

GPIO setup

You can set default GPIO setup at WebIOPi startup.

```
[GPIO]
21 = IN
23 = OUT 0
24 = OUT 0
25 = OUT 1
```

GPIO reset

You can also reset GPIO with WebIOPi shutdown.

```
[~GPIO]
21 = IN
23 = IN
24 = IN
25 = OUT 0
```

Script loading

Loading custom script is easy. You can add setup/loop/destroy function as well as register macros.

```
[SCRIPTS]
# name = sourcefile
myscript = /home/pi/webiopi/examples/scripts/macros/script.py
```

Device registration

See supported devices (DEVICES.html) page for more information about devices, board and components you can use with webiopi.

```
[DEVICES]
# Device configuration syntax:
# name = device [args...]
#   name   : used in the URL mapping
#   device : device name
#   args   : (optional) see device driver doc
# Devices configured here are mapped on REST API /device/name
# Devices are also accessible in custom scripts using deviceInstance(name)
# See device driver doc for methods and URI scheme available

# Raspberry native UART on GPIO, uncomment to enable
# Don't forget to remove console on ttyAMA0 in /boot/cmdline.txt
# And also disable getty on ttyAMA0 in /etc/inittab
serial0 = Serial device:ttyAMA0 baudrate:9600

# USB serial adapters
usb0 = Serial device:ttyUSB0 baudrate:9600

adc = MCP3008
dac = MCP4922 chip:1

gpio0 = MCP23017
gpio1 = MCP23017 slave:0x21
gpio2 = MCP23017 slave:0x22

pwm0 = PCA9685
pwm1 = PCA9685 slave:0x41
```

REST API configuration

By default, REST API allows to GET/POST on all GPIOs, use gpio-export to limit GPIO available. You can also disable GPIO setup and toggling on the REST API.

```
[REST]
gpio-export = 21, 23, 24, 25
gpio-post-value = false
gpio-post-function = false
device-mapping = false
```

URL re-routing

Adding routes allows to simplify access with Human comprehensive URLs

```
[ROUTES]
# Custom REST API route syntax :
# source = destination
#   source      : URL to route
#   destination : Resulting URL

# In the next example with have the bedroom light connected to GPIO 25
# and a temperature sensor named temp2, defined in [DEVICES] section
#  - GET  /bedroom/light       => GET  /GPIO/25/value, returns the light state
#  - POST /bedroom/light/0     => POST /GPIO/25/value/0, turn off the light
#  - POST /bedroom/light/1       => POST /GPIO/25/value/1, turn on the light
#  - GET  /bedroom/temperature => GET  /devices/temp2/sensor/temperature/c, returns the temperatu
re in celsius
/bedroom/light = /GPIO/25/value
/bedroom/temperature = /devices/temp2/sensor/temperature/c
```