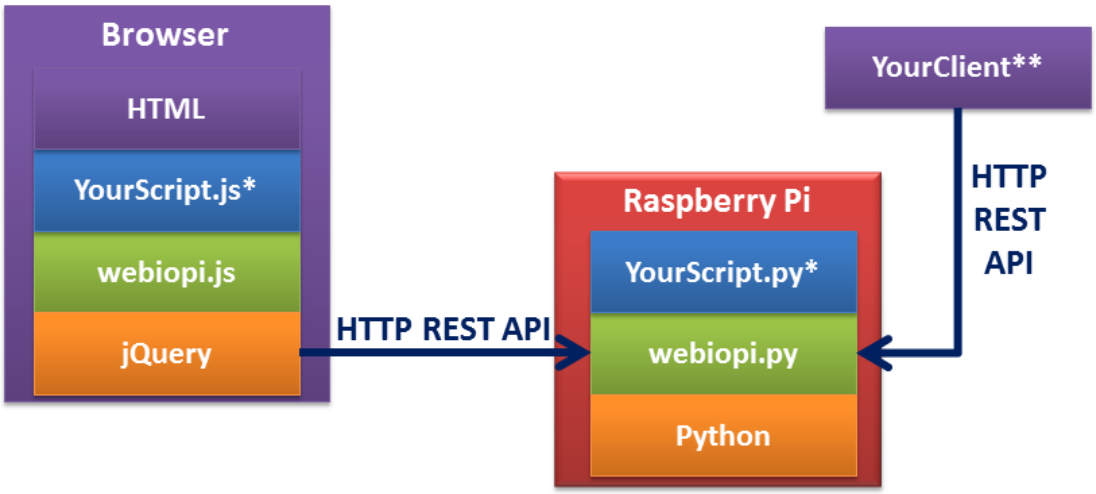
CUSTOMIZE

Updated Jan 8, 2013 by tro...@trouch.com

WebIOPi customization is easy but requires some Javascript and CSS knowledge for the UI. You can also create your own Python script to change the server behavior.



** Optional, allows you to customize the interface and the behavior*
*** Use REST API from any technology/language*

HTML/Javascript : Client and UI

The WebIOPi Javascript library allows you to make your own interface easily, only requiring one single script tag to include /webiopi.js

WebIOPi will automatically load and add required jquery.js and webiopi.css into your HTML file at runtime ! Then you can use the Javascript library to build the interface, creating buttons and registering callbacks.

This example displays big buttons arranged in a column. It also changes the background color of the GPIO 7 for both low and high states. There is two buttons which call a macro and requires to use the associated server script you'll found below. You can use the pre-installed WebIOPi service if you don't plan to use macros.

You have to put both HTML file and Python server script in the same folder and execute the script in that folder to ensure your server will found your HTML file. You can also put your HTML file in /usr/share/webiopi/htdocs

After unpacking WebIOPi, you'll find the following code in examples/custom/index.html :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content = "height = device-height, width = 420, user-scalable = no" />
        <title>WebIOPi | Demo</title>
        <script type="text/javascript" src="/webiopi.js"></script>
        <script type="text/javascript">
        webiopi().ready(function() {
                var content, button;
                content = $("#content");

                // create a "SWITCH" labeled button for GPIO 0
                button = webiopi().createGPIOButton(0, "SWITCH");
                content.append(button); // append button to content div

                // create a "LED" labeled button for GPIO 7
                button = webiopi().createGPIOButton(7, "LED");
                content.append(button); // append button to content div

                // create a button that output a single pulse
                button = webiopi().createPulseButton("pulse", "Pulse", 7);
                content.append(button); // append button to content div

                // create a button which output a bit sequence on GPIO 7 with a 100ms period
                button = webiopi().createSequenceButton("sos", "S.O.S 1", 7, 100, "01010100110011001100101010");
                content.append(button); // append button to content div

                // the previous button will always output the same sequence
                // you can also create a simple button with your own function
                button = webiopi().createButton("sos2", "S.O.S 2", outputSequence);
                content.append(button); // append button to content div
```

```
                // create a button which call myMacroWithoutArgs
                button = webiopi().createMacroButton("macro", "Macro 1", "myMacroWithoutArgs");
                content.append(button); // append button to content div

                // create a button which call myMacroWithArgs with "1,2,3" as argument
                button = webiopi().createMacroButton("macro", "Macro 2", "myMacroWithArgs", [1,2,3]);
                content.append(button); // append button to content div

                // the previous button will always call myMacroWithArgs with the same "1,2,3" argument
                // you can also create a simple button with your own function
                button = webiopi().createButton("macro2", "Macro 3", callMacro);
                content.append(button); // append button to content div

                // you can also create a button which calls a different function for mouse down and up events
                button = webiopi().createButton("hold", "Hold", mousedown, mouseup);
                content.append(button);

                // Only for Chrome and Safari, create a slider that pulse out a 0-100% duty cycle ratio on GPIO 8
                button = webiopi().createRatioSlider(8);
                content.append(button);

                // Only for Chrome and Safari, create a slider that pulse out a -45 to +45° angle on GPIO 9
                button = webiopi().createAngleSlider(9);
                content.append(button);
        });

        function mousedown() {
                webiopi().setValue(7, 1);
        }

        function mouseup() {
                webiopi().setValue(7, 0);
        }

        function outputSequence() {
                var sequence = "01010100110011001100101010" // S.O.S. morse code or whatever you want
                // output sequence on gpio 7 with a 100ms period
                webiopi().outputSequence(7, 100, sequence, sequenceCallback);
        }

        function sequenceCallback(gpio, data) {
                alert("sequence on " + gpio + " finished with " + data);
        }

        function callMacro() {
                var args = [1,2,3] // or whatever you want
                // call myMacroWithArgs(arg)
                webiopi().callMacro("myMacroWithArgs", args, macroCallback);
        }

        function macroCallback(macro, args, data) {
                alert(macro + " returned with " + data);
        }

        </script>
        <style type="text/css">
                button {
                        display: block;
                        margin: 5px 5px 5px 5px;
                        width: 160px;
                        height: 45px;
                        font-size: 24pt;
                        font-weight: bold;
                        color: black;
                }

                input[type="range"] {
                        display: block;
                        width: 160px;
                        height: 45px;
                }

                #gpio7.LOW {
                        background-color: White;
                }

                #gpio7.HIGH {
                        background-color: Red;
                }
        </style>
</head>
<body>
        <div id="content" align="center"></div>
</body>
</html>
```

Python Server script : Extend WebIOPi server

WebIOPi server can also be used as a Python library you can import in your own script. Doing that, you will be able to add initialization and termination code and also to extend WebIOPi functionalities by adding new functions to the REST API.

After unpacking WebIOPi, you'll find the following code in examples/custom/webiopi_custom.py, you can execute from two ways :

- Foreground (debugging):

```
$ sudo python webiopi_custom.py
```

- Background (production)

```
$ sudo ./webiopi_custom.daemon start
```

```
# Imports
import webiopi
import time

# Retrieve GPIO lib
GPIO = webiopi.GPIO

# ----------------------------------------------- #
# Macro definition part                           #
# ----------------------------------------------- #

# A custom macro which prints out the arg received and return OK
def myMacroWithArgs(arg1, arg2, arg3):
    print("myMacroWithArgs(%s, %s, %s)" % (arg1, arg2, arg3))
    return "OK"

# A custom macro without args which return nothing
def myMacroWithoutArgs():
    print("myMacroWithoutArgs()")

# Example loop which toggle GPIO 7 each 5 seconds
def loop():
    GPIO.output(7, not GPIO.input(7))
    time.sleep(5)

# ----------------------------------------------- #
# Initialization part                             #
# ----------------------------------------------- #

# Setup GPIOs
GPIO.setFunction(1, GPIO.IN)
GPIO.setFunction(7, GPIO.OUT)
GPIO.setFunction(8, GPIO.PWM)
GPIO.setFunction(9, GPIO.PWM)

GPIO.output(7, GPIO.HIGH)
GPIO.pulseRatio(8, 0.5) # init to 50% duty cycle ratio
GPIO.pulseAngle(9, 0)   # init to neutral

# ----------------------------------------------- #
# Main server part                                #
# ----------------------------------------------- #

# Instantiate the server on the port 8000, it starts immediately in its own thread
server = webiopi.Server(port=8000, login="webiopi", password="raspberry")
# or     webiopi.Server(port=8000, passwdfile="/etc/webiopi/passwd")

# Register the macros so you can call it with Javascript and/or REST API
server.addMacro(myMacroWithArgs)
server.addMacro(myMacroWithoutArgs)


# ----------------------------------------------- #
# Loop execution part                             #
# ----------------------------------------------- #

# Run our loop until CTRL-C is pressed or SIGTERM received
webiopi.runLoop(loop)

# If no specific loop is needed and defined above, just use
# webiopi.runLoop()
# here instead

# ----------------------------------------------- #
# Termination part                                #
# ----------------------------------------------- #

# Cleanly stop the server
server.stop()

# Reset GPIO functions
GPIO.setFunction(1, GPIO.IN)
GPIO.setFunction(7, GPIO.IN)
GPIO.setFunction(8, GPIO.IN)
GPIO.setFunction(9, GPIO.IN)
```