**WebIOPi - Raspberry Pi REST framework**

***Tutorial: Remote controlled robot cam***

WebIOPi is a REST framework which allows to control Raspberry Pi’s GPIO from a browser. It’s written in Javascript for the client and in Python for the server.

You can fully customize and easy build your own webapp. You can even use all the power of WebIOPi directly in your own Python script and register your functions so you can call them from the webapp. WebIOPi also includes some other features like software PWM for all GPIO.

# Installation

Installation on the Raspberry Pi is really easy, as it only requires Python.

On Raspbian Wheezy, you can use the PiStore to download and install WebIOPi.

You can also install it using a terminal or a SSH connection, check the project page for the latest version, then:

$ wget http://webiopi.googlecode.com/files/WebIOPi-0.5.3.tar.gz

$ tar xvzf WebIOPi-0.5.3.tar.gz

$ cd WebIOPi-0.5.3

$ sudo ./setup.sh

You should see some compile and install process, to finally get a success output with usage instructions:

WebIOPi successfully installed

\* To start WebIOPi with python:
sudo python -m webiopi

\* To start WebIOPi with python3:
 sudo python3 -m webiopi

\* To start WebIOPi at boot:
sudo update-rc.d webiopi defaults

\* To start WebIOPi service:
sudo /etc/init.d/webiopi start

\* Look in /home/pi/webiopi/examples for Python library usage examples

You will have a line for each installed Python version you can use to launch WebIOPi.

It’s time to start WebIOPi, for example with Python 2.X:

$ sudo python -m webiopi

WebIOPi/Python2/0.5.3 Started at http://[IP]:8000/webiopi/

# First use

Open a browser from any device of your network, and point it to the given URL: http://raspberrypi:8000/webiopi/

Replace raspberrypi by its IP. You can use localhost if you are connected to your Pi with a keyboard and a display plugged in it.

You will be asked to log in, **default user is webiopi and password is raspberry**. You should see the default header app:



With the default header app, you can toggle GPIO functions between input and output, and toggle pin states. Just click on the IN/OUT buttons and on each pin to change their state when set as output.

All GPIO can be directly used with the REST API. For instance, to setup GPIO 23 as an output, just make a HTTP POST request on /GPIO/23/function/out then to output a logical 1, make POST on /GPIO/23/value/1

To retrieve states, make HTTP GET on /GPIO/23/function and /GPIO/23/value

The included Javascript library also allows changing GPIO without caring REST calls.



# Robot cam

The next parts of this article are about a robotized webcam you can control from any web browser. You will need:

* Chassis
* Raspberry Pi with WebIOPi
* Operational USB Webcam
* Operational USB WiFi adapter
* L293 H-Bridge
* 2 sets of motor, reducer and wheel
* Battery and Power regulation
* Electronic prototyping parts

From the electronic point of view, L293 contains an electronic circuitry similarly to the Skutter’s H-Bridge from the December MagPi issue. L293 adds an enable input that can be used with a PWM signal to limit the speed.

It also has two power inputs, one for the logic (+V=5V), and one that suits the motors (+Vmotor<36V).



Motor rotation is controlled by IN\* and EN\*:



We can connect +V to Pi’s +5V and IN\*/EN\* to GPIOs. +Vmotor will be connected to the battery or a dedicated regulator. You will need at least one 5V regulator to power the Pi with a battery through the micro USB plug. You can use a 7805 with two capacitors to do that:



With an input voltage higher than 7V, you will have a 5V regulated output voltage for the Pi.



To control the H-Bridge with WebIOPi and create an interface, we have to write few Python lines for the server side, and some Javascript for the client.

# Writing the Python script

Start by creating a new Python file with your favourite editor. You will have to import webiopi then instantiate a Server.

One parameter is required: the port the server has to bind to. You can also change the defaults login and password. The server will start and run in its own thread until the end of the script. We need to add a loop to keep the server running. We use the webiopi.runLoop() function for that. It will sleep the main thread until CTRL-C is pressed. We can also pass a function to loop.

import webiopi

# Instantiate WebIOPi server

# It starts immediately

server = webiopi.Server(

 port=8000,

 login="cambot",

 password="cambot")

# Run the default loop

webiopi.runLoop()

# Cleanly stop the server

server.stop()

The previous script simply starts the WebIOPi server. We can use the default web app to interact with GPIOs, the REST API or the Javascript library. We could directly control H-Bridge lines from Javascript, but we’ll add go\_forward and stop REST macros to decrease latency.

To continue, we need a GPIO library. We can use RPi.GPIO or the integrated GPIO library, which is a fork from RPi.GPIO. Integrated library removes many sanity checks to give a full access from the server and to give more functionality.

Right after the import section:

# Integrated GPIO lib

GPIO = webiopi.GPIO

We add variables to ease H-Bridge control:

# Left motor GPIOs

L1=9 # L293 IN1 on GPIO 9

L2=10 # L293 IN2 on GPIO 10

LS=11 # L293 EN1 on GPIO 11

# Right motor GPIOs

R1=23 # L293 IN3 on GPIO 23

R2=24 # L293 IN4 on GPIO 24

RS=25 # L293 EN2 on GPIO 25

Before the Server call, we write functions for both left and right motors then wrap them into go\_forward and stop macros:

# Left motor functions

def left\_stop():

 GPIO.output(L1, GPIO.LOW)

 GPIO.output(L2, GPIO.LOW)

def left\_forward():

 GPIO.output(L1, GPIO.HIGH)

 GPIO.output(L2, GPIO.LOW)

# Right motor functions

def right\_stop():

 GPIO.output(R1, GPIO.LOW)

 GPIO.output(R2, GPIO.LOW)

def right\_forward():

 GPIO.output(R1, GPIO.HIGH)

 GPIO.output(R2, GPIO.LOW)

# Set the motors speed

def set\_speed(speed):

 GPIO.pulseRatio(LS, speed)

 GPIO.pulseRatio(RS, speed)

# Movement functions

def go\_forward():

 left\_forward()

 right\_forward()

def stop():

 left\_stop()

 right\_stop()

Then, and always before the Server call, we initialize GPIO:

# Setup GPIOs

GPIO.setFunction(LS, GPIO.PWM)

GPIO.setFunction(L1, GPIO.OUT)

GPIO.setFunction(L2, GPIO.OUT)

GPIO.setFunction(RS, GPIO.PWM)

GPIO.setFunction(R1, GPIO.OUT)

GPIO.setFunction(R2, GPIO.OUT)

set\_speed(0.5)

stop()

Finally, we have to register macros on the server to add it to the REST API. It will allow to call them from the web app:

server.addMacro(go\_forward)

server.addMacro(stop)

# Building the interface

Building your own interface is also easy, and is based on a HTML file embedding some Javascript. You only have to load the webiopi.js file from your HTML file to use the WebIOPi power. Create a new index.html file next to your Python script:

<html>

<head>

<title>CamBot</title>

<script type="text/javascript" src="/webiopi.js"></script>

<script type="text/javascript">

// Javascript code goes here

</script>

</head>

<body>

<div id=”box" align="center">

</div>

</body>

</html>

Take note of the starting slash when loading webiopi.js, to ensure it will be searched in the root of the server or it may be not found.

I added an empty script section; we will use the WebIOPi JS library here. There is also a div box, which will contain controls.

In the script section, we add an init function to build the interface using WebIOPi library. It contains many functions to ease creation of buttons that control GPIO. Here we use a basic button to call a different function on press and release. Each function calls a different macro on the server. Don’t forget to register the init function to WebIOPi. It will be called when everything is loaded and ready.

function init() {

 var button =

webiopi().createButton(

 "bt\_up", // id

 "/\\", // label

 go\_forward, // press

 stop); // release

 $("#box").append(button);

}

function go\_forward() {

 w().callMacro("go\_forward");

}

function stop() {

 w().callMacro("stop");

}

webiopi().ready(init);

Be careful that webiopi() is a function and a reserved word that need brackets to return the WebIOPi object. You can use w() to short the webiopi() call.

It’s now time to start the server and enjoy the interface. Open a terminal in the folder you created Python and HTML files and execute the script:

$ sudo python yourscript.py

Open a browser to the webiopi to control the chassis. Hold the button to go forward and release it to stop. The last piece missing is the webcam.

# Add a webcam stream.

There are many possibilities to stream a webcam, which may depend on the model you have. In my case, I have a recent webcam which outputs both RAW and MJPEG formats up to 1280x720@30fps.

First, check your webcam is installed with a terminal:

$ lsusb

[...]

Bus 001 Device 005: ID 046d:0825 Logitech, Inc. Webcam C270

$ ls /dev/video\*

/dev/video0

Then, to check it’s working, you can install uvccapture using apt-get or aptitude and take a single snapshot:

$ uvccapture -v

Using videodevice: /dev/video0

Saving images to: snap.jpg

Image size: 320x240

Taking snapshot every 0 seconds

Taking images using mmap

Resetting camera settings

Camera brightness level is 0

Camera contrast level is 255

Camera saturation level is 255

Camera gain level is 255

Saving image to: snap.jpg

If uvccapture returns without error, we can continue to stream the webcam.

I use MJPG-STREAMER, which is really easy to use. It gives me a 320x240@25fps pass-through MJPEG stream over HTTP. I tried FFMPEG but it takes the RAW output of the webcam to encode it in MJPEG with a framerate under 5fps.

You can download MJPG-STREAMER at http://sourceforge.net/projects/mjpg-streamer/

You will also need libjpeg8-dev you can install using aptitude/apt-get.

Uncompress and build MJPG-STREAMER using make command. Then execute it:

$ ./mjpg\_streamer -i "./input\_uvc.so –r 320x240 –f 25" -o "./output\_http.so –n –p 8001" &

Back to HTML file, add a img tag with src set to http://raspberrypi:8001/?action=stream replacing raspberrypi by your Pi’s IP. You can also directly try the URL in your browser.

...

<img src="http://raspberrypi:8001/?action=stream">

</body>

</html>

# Conclusion

With this article, you learned how to install WebIOPi and how to use it in your own Python scripts to write macros you can call from the web. The code is incomplete as it only allows to go forward and to stop. Just add left/right\_backward, turn\_left/right and go\_backward functions to move the robot in all directions.

You can download the complete code at http://files.trouch.com/webiopi/cambot.zip. You will find more information on the project wiki and in the examples folder of WebIOPi archive.

***Eric PTAK, creator of WebIOPi***

***http://trouch.com***

***http://code.google.com/p/webiopi/***



