Implementing WebIOPi Framework on Raspberry Pi

For Internet of Things

By

Ubaier Ahmad Bhat

Ali Belakehal

Contents

Introduction

The aim of our project is to build an Internet of Thinks application that will allow a user to control and monitor devices using a web based interface. The original inspiration for the project came from the YouTube video shown in the class title "Use a Raspberry Pi to Fix Everyday Problems. Become the Office Hero!". In the video the guys were sensing the status of a switch and saving the status in an html file which was uploaded into a public DrobBox folder and that's how it was shared on the internet. This approach is simple however the main problem with this approach was that we could only monitor the device and not control them.

In order to overcome the problem of controlling the devices we researched into how a RESTful application can be developed on a Raspberry Pi and during our research we came to know about Apache based WebIOPi framework which we decided to use for our project.

In this report we will present all the steps required from connecting to the Raspberry Pi to building a web based application using WebIOPi that works in a Local Area Network.

The project code can be downloaded from: https://github.com/ubaierbhat/webiopi-ttu

Overview of Raspberry Pi

Raspberry Pi is a powerful single-board computer developed by Raspberry Pi Foundation. It has a 32-bit ARM processor and uses a Fedora distribution of Linux for its default operating system. It can be programmed with Python and other languages. The Raspberry Pi computer is essentially a system-on-a-chip (SoC) with connection ports. It can be operated by hooking up a USB keyboard and plugging the computer into a television. The Raspberry Pi Foundation has said it will be issuing two version of the computer: Model A and come with 256Mb RAM and one USB port but no network connection. Model B has 256Mb RAM, comes with two USB ports and can be connected to an Ethernet network. For this project we have used Model B Revision 2.

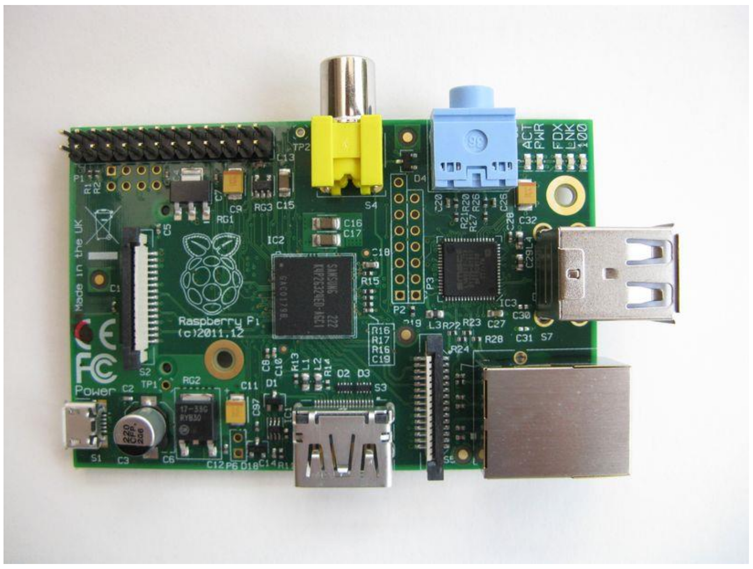


Figure 1 Raspberry Pi Model B Rev2 used for this project

Overview of WebIOPi Framework

WebIOPi is an open source framework to develop Internet of Things applications on Raspberry Pi. WebIOPi includes an HTTP server (Apache) that provides both HTML resources and a REST API to control things. The application runs within a web browser will first load a HTML file, then the included Javascript will make Asynchronous calls to the REST API to control and update the UI.

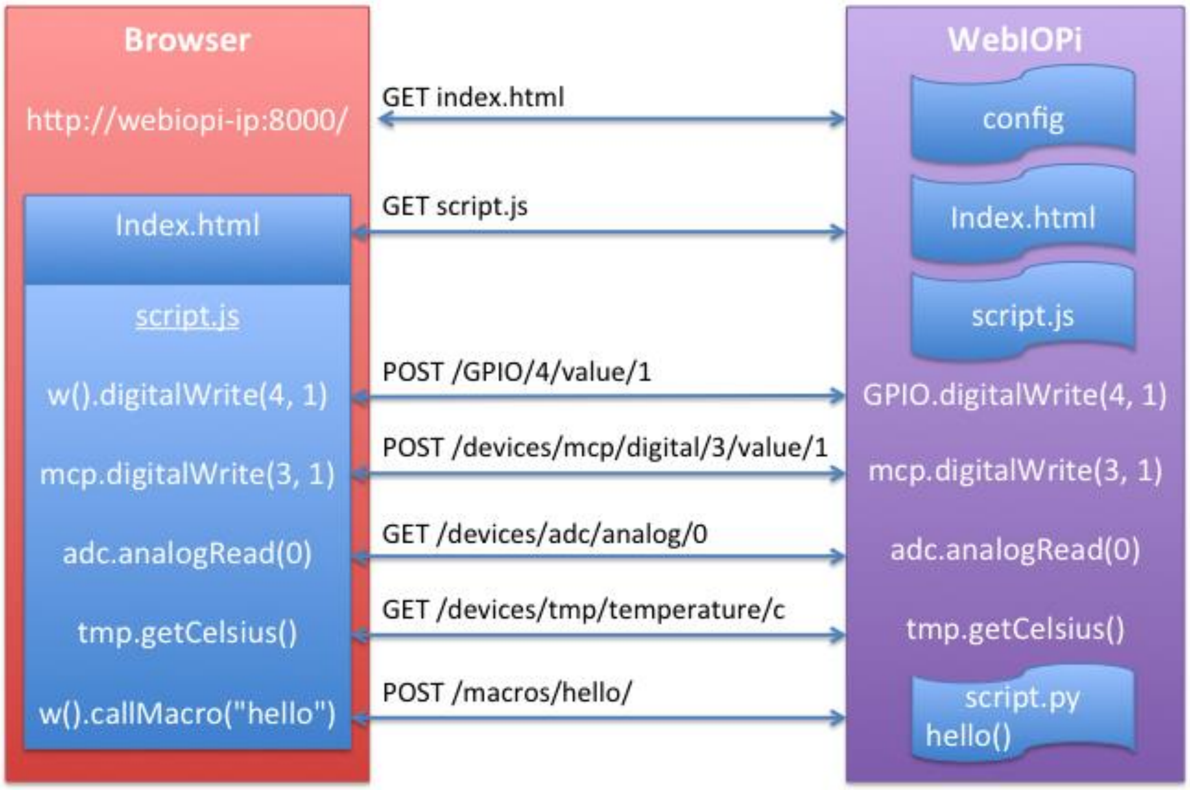


FIGURE 2 CLIENT SERVER INTERACTIONS IN WEBIOPI

Two main features of WebIOPi are:

- Ability to build HTML / Web UI from scratch, using the WebIOPi JS library or not.
- Extend the WebIOPi behaviour by loading custom Python script using an Arduino like syntax with setup/loop functions

Connecting to Raspberry Pi via LAN

In order to connect the Raspberry Pi via LAN the following software should be installed on the computer:

1. Putty (download from: http://www.chiark.greenend.org.uk/~sgtatham/putty/)
2. Xming X Server (download from: http://sourceforge.net/projects/xming/)

The network setup for the project can be seen in the figure below

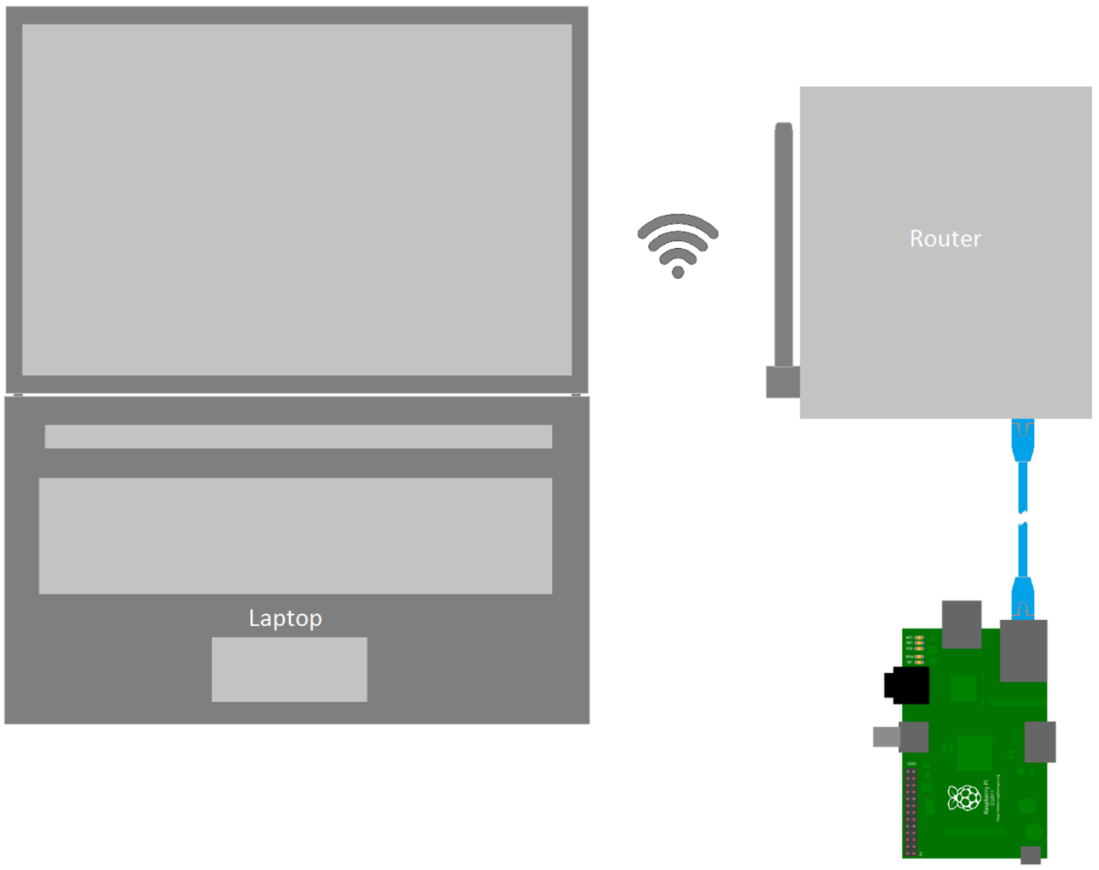
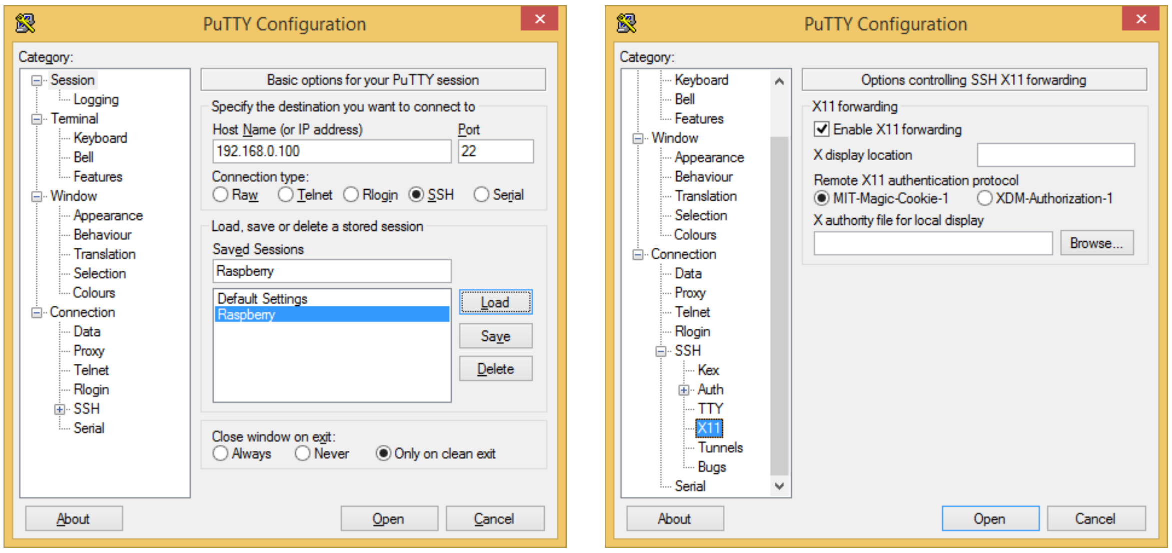


FIGURE 3 CONNECTING TO RASPBERRY PI VIA LAN

In order to connect to the Raspberry Pi the following settings should be used. The IP address will depend on the address assigned by the DHCP server (router). It is better to bind a particular IP address to the Raspberry Pi's MAC address. In a router is not available the steps in this tutorial "Guide To…Direct Network Connection" can be used to assign a fixed IP to the Raspberry Pi during boot up.



PuTTY Configuration — Session: Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.100 Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

Saved Sessions: Raspberry

Default Settings, Raspberry

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

PuTTY Configuration — Connection > SSH > X11: Options controlling SSH X11 forwarding

X11 forwarding

[✓] Enable X11 forwarding

X display location:

Remote X11 authentication protocol: [✓ MIT-Magic-Cookie-1] ( ) XDM-Authorization-1

X authority file for local display:

FIGURE 4 SETTINGS FOR PUTTY

Once a connection has been established username and password are required to start the session. The default credentials for Raspbian are:

username	pi
password	raspberry

In order to use the graphic user interface of Raspbian the Xming X server should be running on the computer. In order to start the GUI the following command has to be used:

```
startlxde
```

In order to use an cli application as a super user the following command should be used

```
sudo command
```

In order to use an GUI application as a super user the following command should be used

```
gksudo command
```

Setting up WebIOPi

Installing

1. Download the latest version of WebIOPi from http://sourceforge.net/projects/webiopi/files/. For this project we used the version 0.7 of WebIOPi (WebIOPi -0.7.0.tar.gz)
2. Unzip the files using following command: `tar xvzf WebIOPi-x.y.z.tar.gz`
3. Change directory to the new folder: `cd WebIOPi-x.y.z`
4. Run the following command to install the application: `sudo ./setup.sh`

Hardware Setup

For this project we used three LEDs as our output devices. These can be replaced with relays in order to control some other device like a light bulb, a motor or some other machine.

As our input device we just decided to have a simple switch to represent a digital input device. We can connect other types of sensors and even analogue sensors for advance applications.

Device	GPIO #	Pin
LED 1 (red)	2	3
LED 2 (green)	3	5
LED 3 (green)	4	7
Switch/sensor	17	11

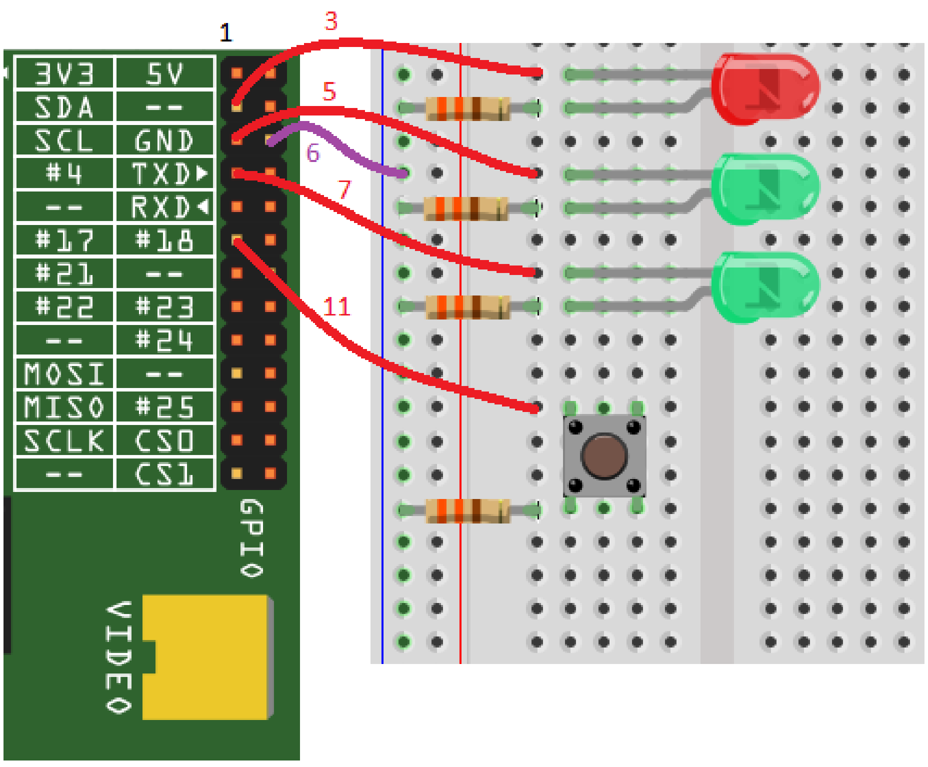
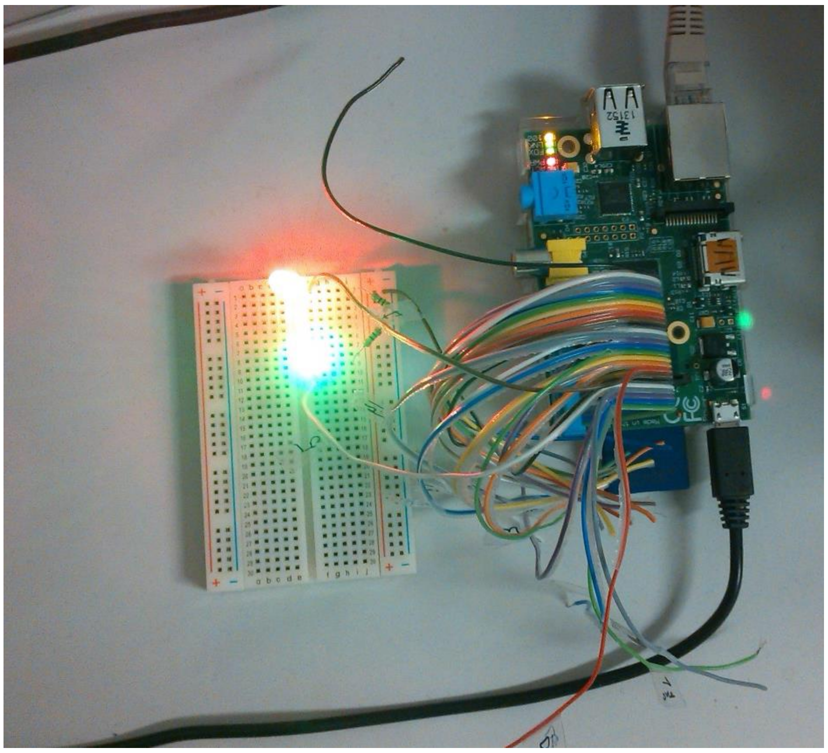


Figure 5 Circuit Diagram showing different GPIO connections



Starting the application server

Command to start the server:

```
sudo /etc/init.d/webiopi start
```

Command to stop the server:

```
sudo /etc/init.d/webiopi stop
```

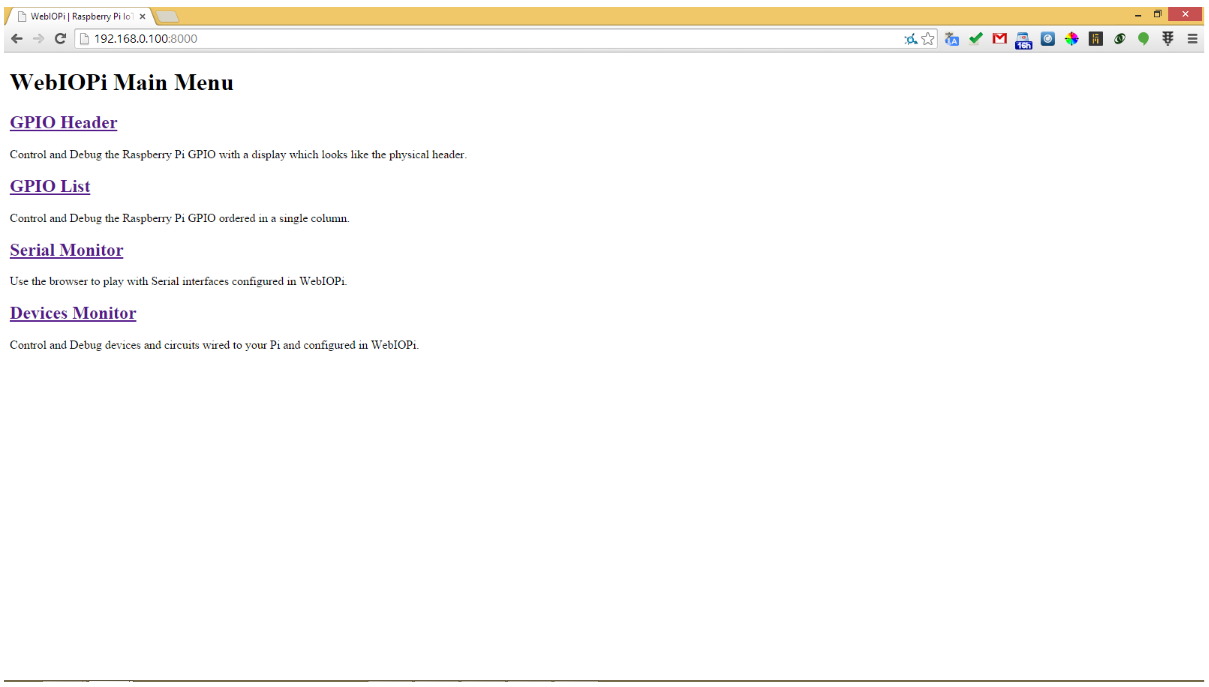
Command the check the status of the server:

```
sudo /etc/init.d/webiopi status
```

Launching the default web application

By default the application is served on port 8000. To launch the application in the web browser got to your Raspberry Pi ip address eg: 192.168.0.100:8000 in our case. The user is prompted for username and password. The default credential for WebIOPi are:

username	webiopi
password	raspberry


WebIOPi | Raspberry Pi Io

192.168.0.100:8000

WebIOPi Main Menu

GPIO Header

Control and Debug the Raspberry Pi GPIO with a display which looks like the physical header.

GPIO List

Control and Debug the Raspberry Pi GPIO ordered in a single column.

Serial Monitor

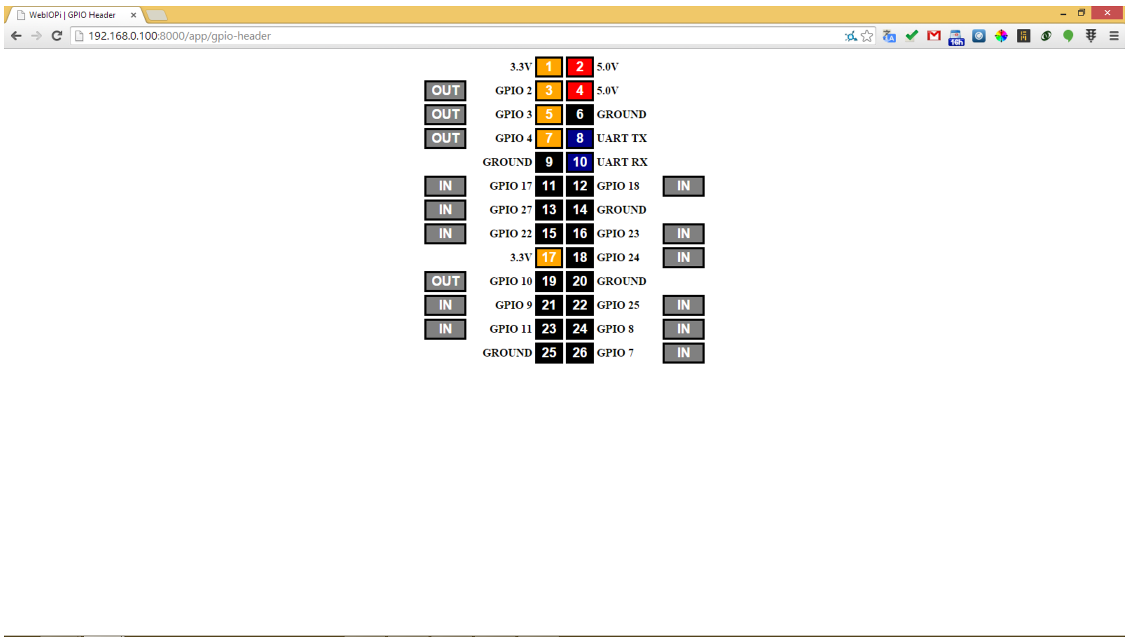
Use the browser to play with Serial interfaces configured in WebIOPi.

Devices Monitor

Control and Debug devices and circuits wired to your Pi and configured in WebIOPi.

Figure 6 Default Home Page of WebIOPi application

The default homepage enables the user to check the setting of the GPIO heads and monitor additional serial devices. The GPIO header page will enable the user to configure and test inputs and outputs as shown in the figure below.



	3.3V	1	2	5.0V	
OUT	GPIO 2	3	4	5.0V	
OUT	GPIO 3	5	6	GROUND	
OUT	GPIO 4	7	8	UART TX	
	GROUND	9	10	UART RX	
IN	GPIO 17	11	12	GPIO 18	IN
IN	GPIO 27	13	14	GROUND	
IN	GPIO 22	15	16	GPIO 23	IN
	3.3V	17	18	GPIO 24	IN
OUT	GPIO 10	19	20	GROUND	
IN	GPIO 9	21	22	GPIO 25	IN
IN	GPIO 11	23	24	GPIO 8	IN
	GROUND	25	26	GPIO 7	IN

Figure 7 GPIO header page

The user can change the setting just by clicking on the buttons which will toggle the settings. The default app is located at the following location:

/usr/share/webiopi/htdocs

Designing Custom UI in WebIOPi

The best feature about WebIOPi framework is that it enables a developer to easily build a custom application using JavaScript, HTML and CSS.

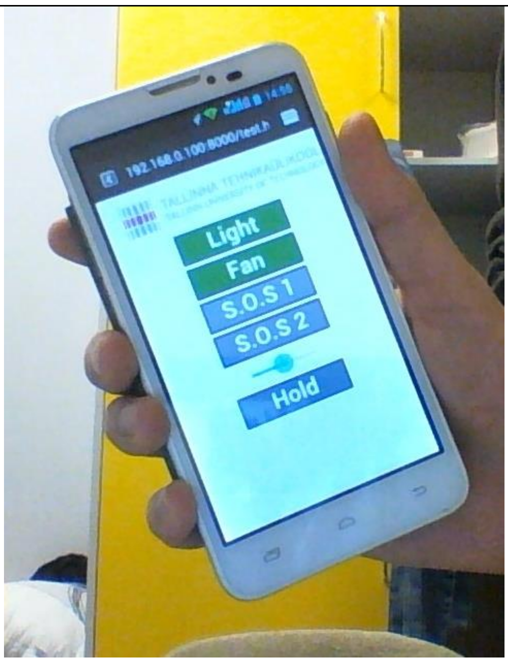


Figure 8 Testing different UI customisations on a mobile device

Updating Configuration

In order to customize and build our own application we need to update the configuration file located at: `/etc/webiopi/config`

This file can be used to change the default port for the application. Set the location of the custom app etc. For our application we are only changing the HTML configuration to:

```
[HTTP]

# HTTP Server configuration

enabled = true

port = 8000



# File containing sha256(base64("user:password"))

# Use webiopi-passwd command to generate it

passwd-file = /etc/webiopi/passwd

# Change login prompt message
```

```
prompt = "WebIOPi"

# Use doc-root to change default HTML and resource files location

#doc-root = /home/pi/myproject

# Use welcome-file to change the default "Welcome" file

#welcome-file = index.html
```

As can be seen from the configuration setting the root directory for our project is located at:

```
/home/pi/myproject
```

On the next page you will find the source code of our application

Source Code

```
<!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 content="width=device-width, initial-scale=1, user-scalable = no " name="viewport" />

    <title>WebIOPi | Home Automation Controls</title>

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

    <script type="text/javascript">

    webiopi().ready(function() {

        // Settup GPIO Pins

                webiopi().setFunction(2,"out");

                webiopi().setFunction(3,"out");

                webiopi().setFunction(4,"out");

                webiopi().setFunction(17,"in");


                // Create a button title "LED 1" labeled button for GPIO 2

        var button = webiopi().createGPIOButton(2, "LED 1");

                // Append button to HTML element with ID="controls" using jQuery

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


                // Create a button title "LED 2" labeled button for GPIO 3

                button = webiopi().createGPIOButton(3, "LED 2");

        // Append button to HTML element with ID="controls" using jQuery

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


                // Create a button title "LED 3" labeled button for GPIO 3
                button = webiopi().createGPIOButton(4, "LED 3");

        // Append button to HTML element with ID="controls" using jQuery

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


                // Create a status box title "Sensor" labeled button for GPIO 2

        var indicator = webiopi().createGPIOButton(17, "Sensor");

                // Append indicator to HTML element with ID="controls" using jQuery

                $("#controls").append(indicator);



                // Refresh GPIO buttons

        // pass true to refresh repeatedly of false to refresh once

        webiopi().refreshGPIO(true);

});



</script>

<style type="text/css">

        button {

                display: block;

                margin: 5px 5px 5px 5px;

                width: 160px;

                height: 45px;

                font-size: 24pt;

                font-weight: bold;

                color: white;

        }
```

```
		#gpio2.LOW {
			background-color: Red;
		}

		#gpio2.HIGH {
			background-color: Green;
		}

		#gpio3.LOW {
			background-color: Red;
		}

		#gpio3.HIGH {
			background-color: Green;
		}

	#gpio4.LOW {
	    background-color: Red;
	}

	#gpio4.HIGH {
	    background-color: Green;
	}

		#gpio17.LOW {
			background-color: Red;
```

```
                    }


                    #gpio17.HIGH {

                              background-color: Green;

                    }




      </style>

</head>

<body>

      <div align="center"><img src="http://www.ttu.ee/extensions/ttu/assets/img/ttu-logo.png"></div>

            <div id="controls" align="center"></div>

</body>

</html>
```

Final Output:

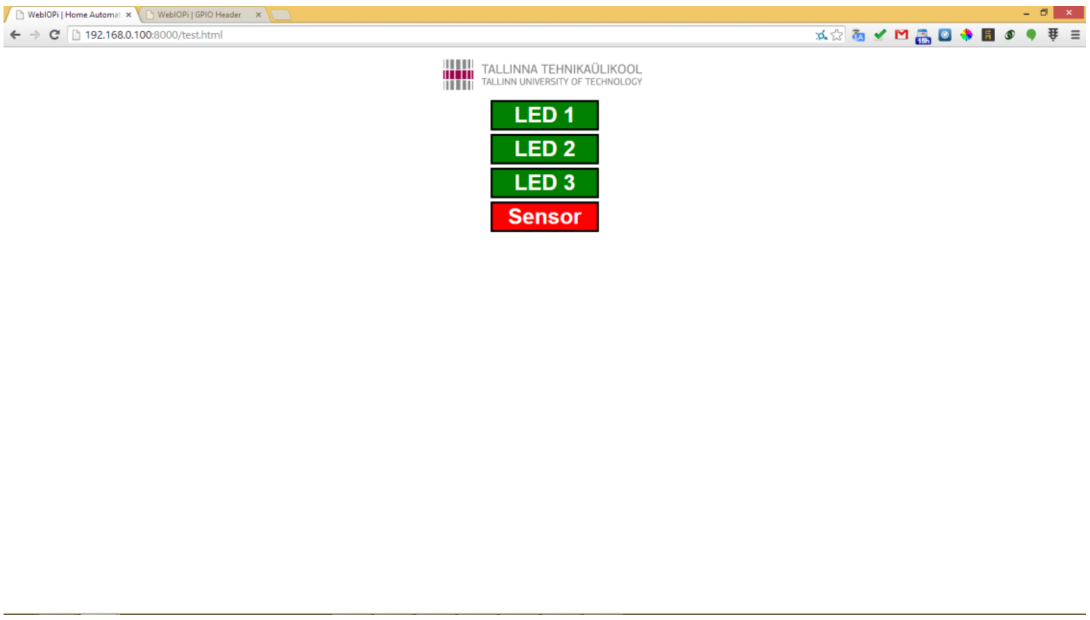


Figure 9 Web interface in Chrome browser

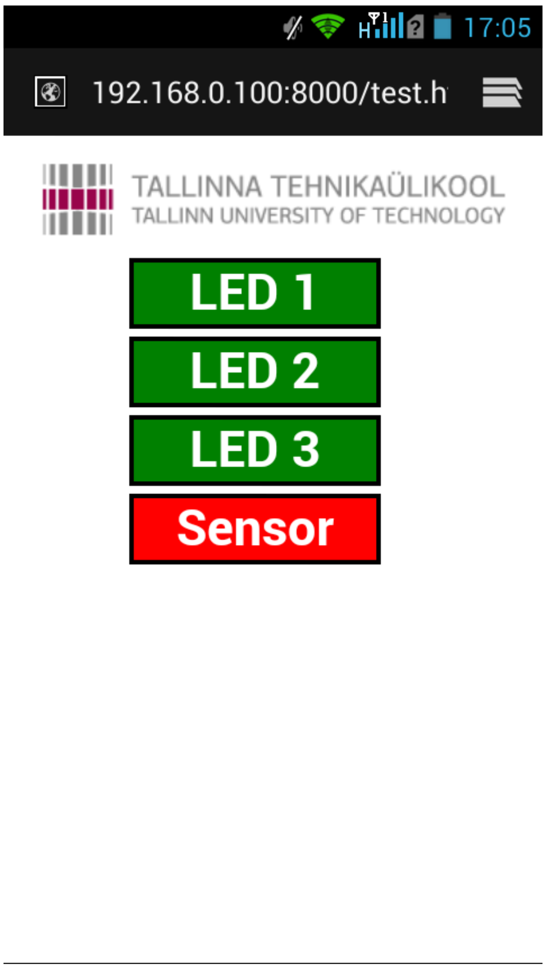


Figure 10 Web Interface on a Mobile device

Conclusion

In this project we developed a simple application to control couple of LEDs and read input from a sensor. The WebIOPi interface can used to create very advance applications. The application can also be configured to work over the internet.

We would have liked to connect some more sensors and actuators to the application, e.g. Temperature sensor or pressure sensors and build a fully functional closed loop control system.

The developers of WebIOPi are also planning to launch a new application IOMOTIX which will enable to build rapid applications for multiple single-board computers like Raspberry Pi, Beagle Bone black, Banana Pi. This will be great as it will remove the hardware dependence of the WebIOPi framework.

References

- YouTube video: Use a Raspberry Pi to Fix Everyday Problems. Become the Office Hero!, https://www.youtube.com/watch?v=ZszlVVY1LXo
- Raspberry Pi main website, http://www.raspberrypi.org/
- WebIOPi project website, https://code.google.com/p/webiopi/
- Bathroom Status App source code repository, https://github.com/ubaierbhat/BathroomStatus
- Guide To…Direct Network Connection, http://pihw.wordpress.com/guides/direct-network-connection/
- How to find MAC address of Raspberry Pi, http://www.raspberrypi-spy.co.uk/2012/06/finding-the-mac-address-of-a-raspberry-pi/
- Official Tutorial on installing WebIOPi, https://code.google.com/p/webiopi/wiki/INSTALL
- YouTube video Raspberry Pi Custom Web Control, https://www.youtube.com/watch?v=6RaBz01pi4E
- IOMotix, http://www.iomotix.com/
- Project Repository, https://github.com/ubaierbhat/webiopi-ttu/tree/master
-