Welcome, **Guest** | Log in | Register | **Activity** | **Translate** |

Content Search within content, members or groups Search

Topics Resources Members Design Center Store

All Places > Raspberry Pi > Raspberry Pi Projects > Blog > 2014 > April > 30

Raspberry Pi Projects

Turn the Rapiro into an IoT device with Eclipse Orion and MQTT

Posted by kartben in Raspberry Pi Projects on Apr 30, 2014 9:38:21 AM

I finally found time to play with my Rapiro robot (thanks again Element14 for accepting me for this RoadTest!).
You can have a look at the review I wrote, but in a nutshell, after having figured out a few hardware issues, I was essentially ready to start hacking and code what I had in mind: I wanted my Rapiro to become an IoT (Internet of Things) device, that I could control for virtually anywhere in the world!

The project

Meet Rapiro. Rapiro is a nice little toy, that comes in a kit. It is, as you've guessed, a robot!

- The **robot** itself consists of 12 servos (3 in each arm, 2 in each leg, 1 for the head and 1 for the waist), enabling a wide range of movements. Not-so-wide movements actually, the robots' limb are **very** short!
- A **Raspberry Pi**, in the head of the robot, allows
- A **PiCam**, a 5Mpix camera extension to the Raspberry Pi, allows to take pictures, records videos, ...
- A **USB WiFi dongle** allows to control the robot completely wirelessly

I won't detail how to assemble the Rapiro, neither will I explain how to get your Raspberry Pi setup with a Linux distro since there are plenty of resources on the Internet. The fun part actually starts after you have finished assembling the Rapiro, and when the Raspberry Pi is properly running, allowing you to open a remote SSH connection to start hacking the beast!

The goal is very simple: my Rapiro is able to get Internet connectivity, thanks to its Raspberry Pi and WiFi dongle, so I want to make it controllable remotely. To that effect, I am going to use MQTT, a very lightweight protocol well-suited for Internet of Things communications, to send commands to the robot. Thanks to the PiCam, the robot should also, when asked to do so, take a picture and transmit it. I want to be able to open a browser from wherever in the world, and start controlling my Rapiro (because why not, eh?), so I will develop a simple web UI to send the actual MQTT commands.

Steps

Install Eclipse Orion

Eclipse Orion is an IDE that you can run from your favorite web browser. Once you have Orion running on a device (here it's going to be on the Raspberry Pi attached to my Rapiro, obviously), all you really need is a browser for writing.
We are going to run Orion on top of Node.js so we are going to need Node.js first.

```
+ expand source  view plain
01. cd
02. wget http://nodejs.org/dist/v0.10.26/node-v0.10.26-linux-arm-pi.tar.gz
03. tar -xvzf node-v0.10.26-linux-arm-pi.tar.gz
04. sudo mv node-v0.10.26-linux-arm-pi /opt/node/
```

In order to have the `node` and `npm` executables in your PATH, you can edit the `/etc/profile` file and add the following lines:

```
+ expand source  view plain
01. NODE_JS_HOME=/opt/node"
02. PATH="$PATH:$NODE_JS_HOME/bin"
03. export PATH
```

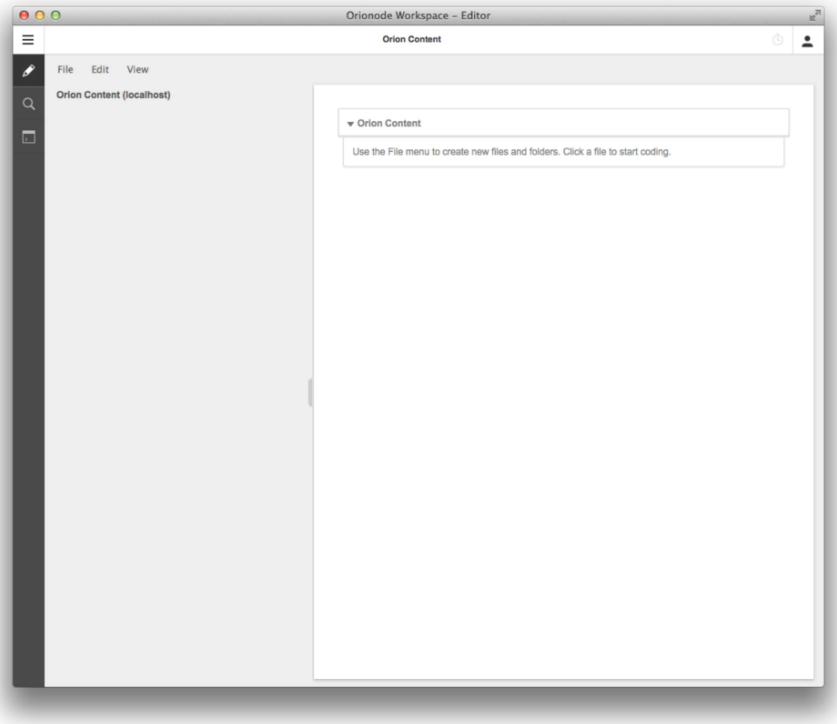
Great! Node is now correctly installed, and we can use the standard package manager (npm) to install and start Orion.

```
+ expand source  view plain
01. npm install orion
02. npm start orion
```

You should see an output similar to this:

```
+ expand source  view plain
01. > orion@0.0.30 start /home/pi/node_modules/orion
02. > node server.js
03.
04. Using workspace: /home/pi/node_modules/orion/.workspace
05. Listening on port 8081...
```

And if you open your favorite web browser (`http://ip-address-of-your-raspberrypi:8081`), you'll see that Orion is actually running – congrats!



So what's next? Orion is going to help us develop a very simple application that will expose Rapiro's Arduino-based controller, to the Internet.

Allow the Rapiro to be controlled from anywhere with MQTT

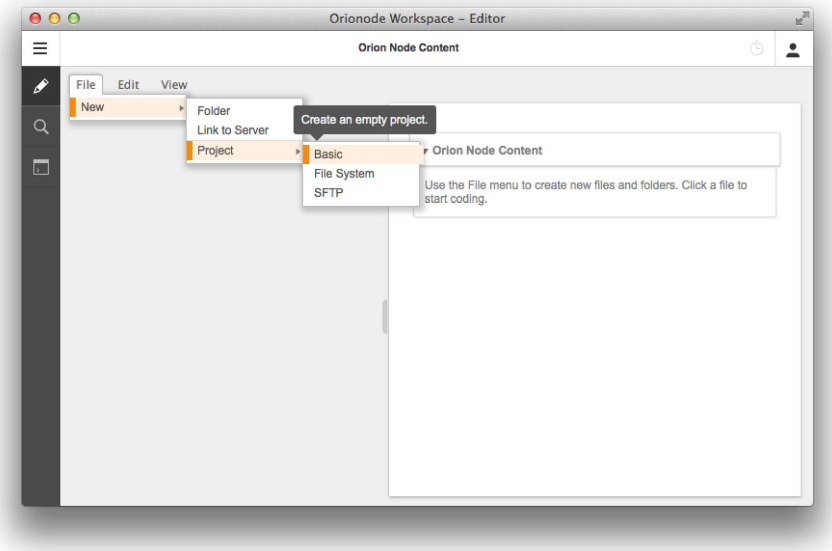
Rapiro built-in serial commands

By default, the Rapiro is meant to be controlled over a serial connection. Basically, the servo-motors are all attached to an Arduino board, onto which runs a program that listens to commands on the serial interface. These commands can be either built-in, like `#M1` to start walking or `#M5` to wave hands, or more complex sequences, like `#PR000G255B000T010` to set the eyes LEDs to Green (R=0, G=255, B=0), in 10 units of time (T=10).

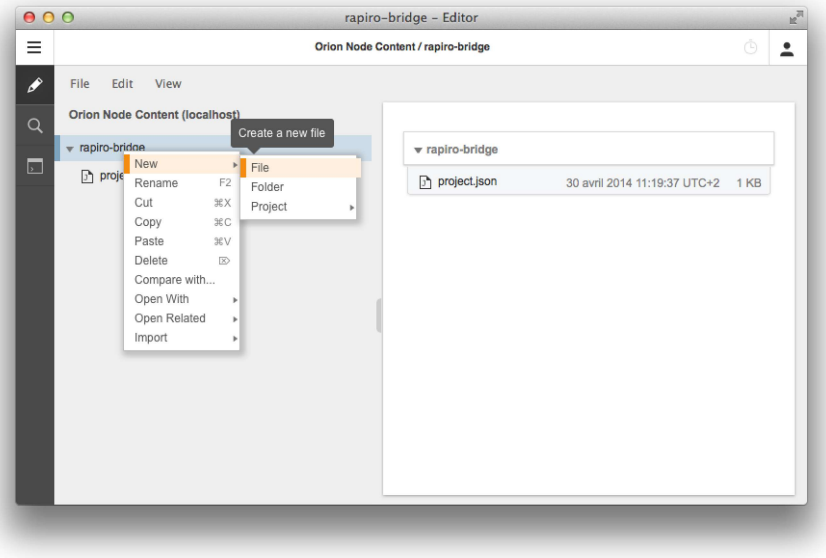
The Rapiro's head is home for its second brain, in addition to the Arduino controller: a RaspberryPi. It is powered by the same battery pack than the robot, and connected to the Arduino's serial line meaning that I can actually send serial commands from virtually any program that I'd run on the Raspberry Pi. As a very first test, you can ask the Rapiro to start walking from the command line of your Raspberry Pi:

```
+ expand source  view plain
01. echo "#M1" | sudo minicom -b 57600 -o -D /dev/ttyAMA0
```

So now the goal is to use Orion to write a simple app that will use MQTT to listen to commands that a web UI is going to send, and forward these commands to the Arduino controller over the serial port.
In Orion, we're going to create a new project (File > New > Project > Basic).



And in this project is going to be our main application, so let's create a file called e.g. `main.js`.



+ expand source view plain

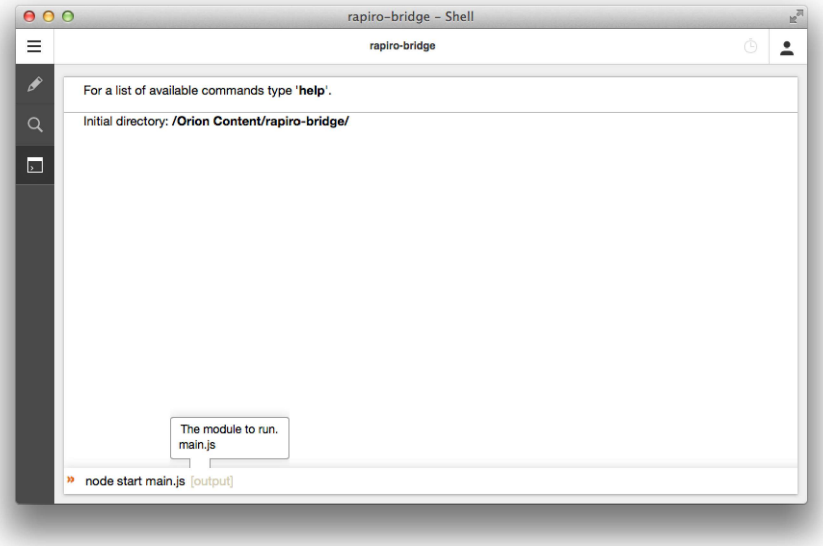
```
01.  /*******************************************************************************
02. * Copyright (c) 2013, 2014 Benjamin Cabé and others.
03. * All rights reserved. This program and the accompanying materials
04. * are made available under the terms of the Eclipse Public License v1.0
05. * which accompanies this distribution, and is available at
06. * http://www.eclipse.org/legal/epl-v10.html
07. *
08. * Contributors:
09. *     Benjamin Cabé - initial API and implementation
10. *******************************************************************************/
11.
12.
13. /** Serial port configuration **/
```

Yes, it's that simple! Not even 50 lines of Javascript code, cool eh?
Basically, we're using two NodeJS modules: `serialport` for manipulating the serial port of the Raspberry Pi to which the Arduino controller is attached, and `mqtt` for doing... MQTT communications!
The code is pretty straight forward:

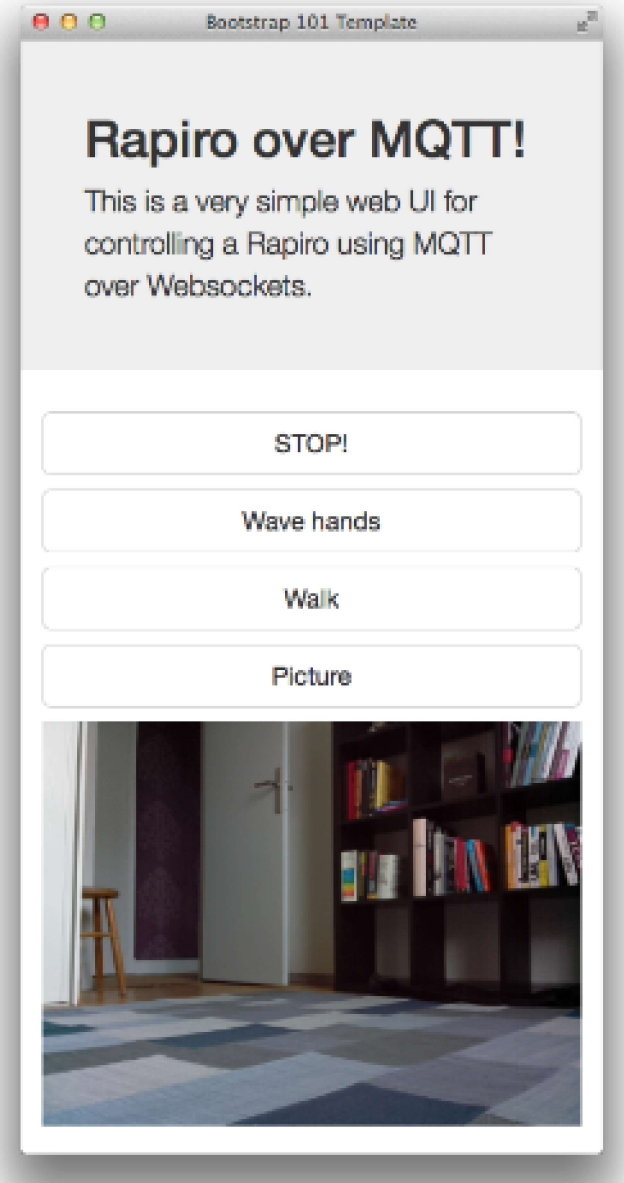
- lines 14-19: we setup the serialport module, indicating the name of the serial port, its baudrate, ...
- lines 22-25: we setup the MQTT client: we want to communicate with the MQTT broker available at iot.eclipse.org, and we subscribe to the topic where we are accepting to receive commands targeted at the controller
- lines 40-50: on line 50 we indicate the Javascript callback for when a new MQTT message is receive.
  In the callback, if the message we received has a "TAKE_PICTURE" payload, we're using raspistill to take a picture and output the binary JPEG encoded in base64 on stdout. `publishPicture` callback then takes this output, and publishes it in an MQTT message (see Lines 31-35).
  If the payload is something else than "TAKE_PICTURE", we just use the serialport module to write this on the serial port, assuming it's a serial command – hopefuly it's something Rapiro can understand!

Use Orion shell to run the app

Now that we have written our MQTT-to-serial bridging application, it's time to use the Orion shell to actually launch it.


rapiro-bridge – Shell
rapiro-bridge
For a list of available commands type 'help'.
Initial directory: /Orion Content/rapiro-bridge/
The module to run.
main.js
node start main.js [output]

Web UI


Bootstrap 101 Template
Rapiro over MQTT!
This is a very simple web UI for controlling a Rapiro using MQTT over Websockets.
STOP!
Wave hands
Walk
Picture

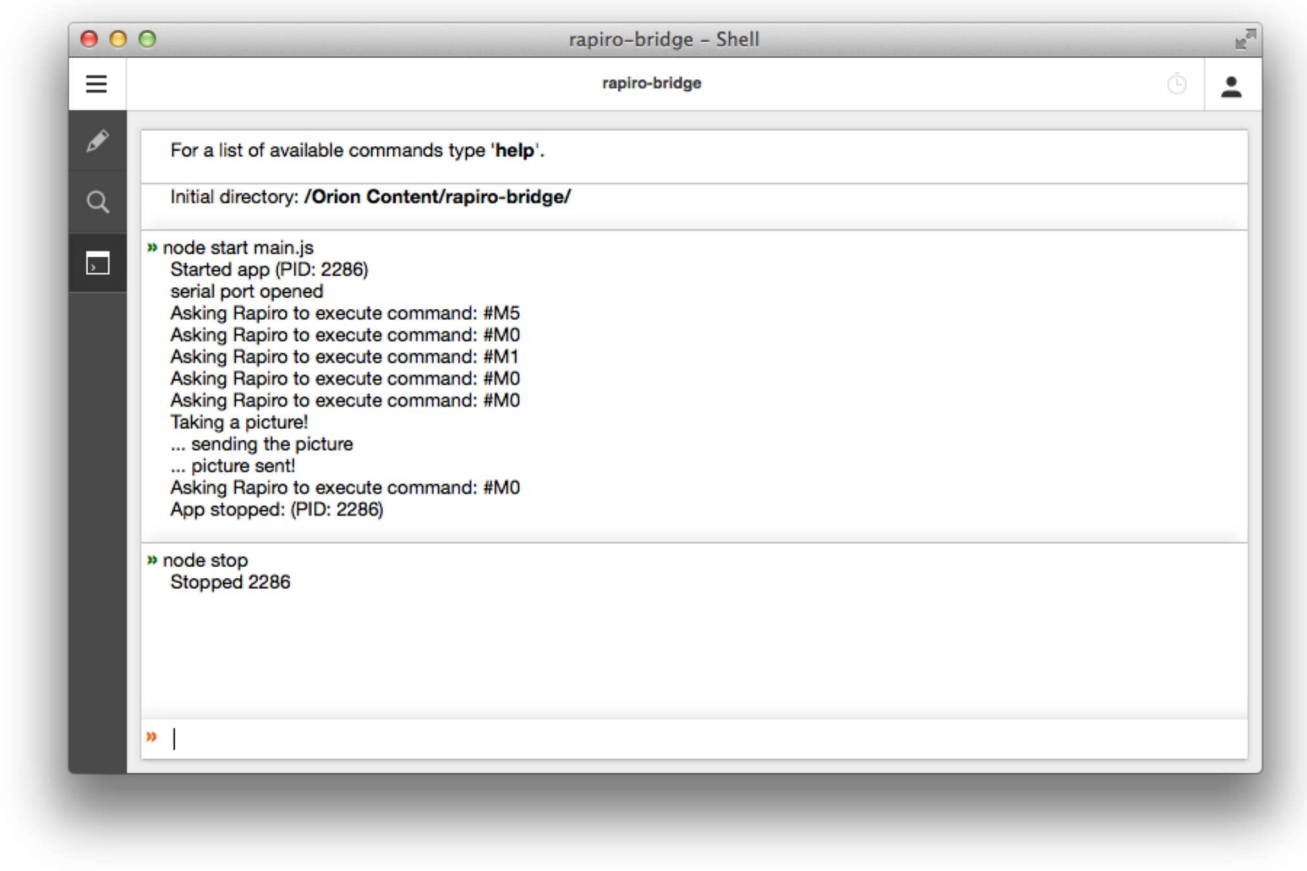
Here the idea is pretty straightforward. I want a dead simple web UI that will allow me to publish MQTT messages, and receive base64-encoded images that I can display in my page.
In order to perform MQTT communications from a web page, there in Eclipse Paho, a Javascript client that allows to do MQTT over WebSockets.
Without further ado, here is the actual web UI for my page:

```
+ expand source  view plain
01. <div class="container">
02.   <div class="row">
03.     <div class="col-xs-12 col-md-3">
04.       <p><a class="btn btn-default btn-lg btn-block" href="#" role="button" onclick='message = new Messaging.Message("#M0"); me
05.     </div>
06.     <div class="col-xs-12 col-md-3">
07.       <p><a class="btn btn-default btn-lg btn-block" href="#" role="button" onclick='message = new Messaging.Message("#M5"); me
08.     </div>
09.     <div class="col-xs-12 col-md-3">
10.       <p><a class="btn btn-default btn-lg btn-block" href="#" role="button" onclick='message = new Messaging.Message("#M1"); me
11.     </div>
12.     <div class="col-xs-12 col-md-3">
13.       <p><a class="btn btn-default btn-lg btn-block" href="#" role="button" onclick='message = new Messaging.Message("TAKE_PIC
```

It's very simple: when a button is clicked, an MQTT message is published on the topic `benjamin/rapiro/command` (remember that this is what my Rapiro's Raspberry Pi is subscribed to, right?), with a payload that is what action I want the Rapiro to perform
And below is the few lines of Javascript that will initialize my MQTT client, and listen to the topic on which a base64 image may show up.

```
+ expand source  view plain
01. <script type="text/javascript" src="http://iot.eclipse.org/demo/js/mqttws31.js"></script>
02. <script>
03.   client = new Messaging.Client("ws://iot.eclipse.org/ws", "rapiroweb" + new Date().getTime());
04.   client.onConnectionLost = onConnectionLost;
05.   client.onMessageArrived = onMessageArrived;
06.   client.connect({
07.     onSuccess: onConnect
08.   });
09.
10.   function onConnect() {
11.     // Once a connection has been made, make a subscription and send a message.
12.     console.log("onConnect");
13.     client.subscribe("benjamin/rapiro/pic");
```

When actually using the Web UI, granted that the NodeJS app is still running on the Raspberry Pi, the Rapiro will start doing what you're asking it too, and obviously you can use the Orion shell output to troubleshoot what's happening.



See it live!

So this is it! Check out the video below to see all of this in action!

Turn the Rapiro into an IoT device with Eclipse Orio...

More...

Possible improvements

It took me a couple hours to put together this project (and actually more than a couple hours to write this blog post!), so there is definitely room for improvement.

- Sending raw instructions (e.g. `#M0`) over MQTT is pretty convenient and very versatile, but the Node.js MQTT-to-serial bridge could be smarter, and instead of naively "forwarding" instructions received via MQTT to the Rapiro over serial, it could understand more advanced commands that would be transmitted as part of the MQTT topic. That way, an MQTT message received on the topic `benjamin/rapiro/command/walk` would be translated into an `#M1` command being written on the serial link.
- Sending webcam images over MQTT works great, but it might be more appropriate to actually expose a live MJPEG stream to the Internet

Source code

The source code for the Web UI is on Github https://github.com/kartben/rapiro-web-ui – it's very simple, but eh, feel free to improve it!

Upcoming webinar

If you would like to ask me questions about that particular project, or if you'd like to learn more about some other cool IoT projects that are part of the Eclipse IoT initiative, please join my webinar next week, on May 7 – From Arduinos to Raspberry Pis, IoT-ize Your Embedded Projects with Open IoT software!

8924 Views Tags: internet_of_things, iot, raspberry-pi, rpiexpert, mqtt, orion

Average User Rating

(0 ratings)

2 Comments

Login or Register to comment

Frederick Vandenbosch May 4, 2014 3:26 AM

Great post! Something I'll have to try with my Rapiro

Frederick

Actions Like (1)

esifts Jan 10, 2015 6:18 PM

I was able to install as well as run Orion on Rapiro. However, when I now try to run node.js, it gives me the error:

» node start main.js Started app (PID: 3179)module.js:340 throw err; ^Error: Cannot find module 'serialport' at Function.Module._resolveFilename (module.js:338:15) at Function.Module._load (module.js:280:25) at Module.require (module.js:364:17) at require (module.js:380:17) at Object.<anonymous> (/home/pi/node_modules/orion/.workspace/RJRemoteRapiro/main.js:8:18) at Module._compile (module.js:456:26) at Object.Module._extensions..js (module.js:474:10) at Module.load (module.js:356:32) at Function.Module._load (module.js:312:12) at Function.Module.runMain (module.js:497:10)App stopped: (PID: 3179)

I even installed serialport globally as mentioned online but that didn't resolve it. Any advice? Thanks!

Actions Like (0)

element14 is the first online community specifically for engineers. Connect with your peers and get expert answers to your questions.

Content | Topics | Resources | Design Center | Members | Store
About Us | Feedback & Support | FAQs | Terms of Use | Privacy Policy | Cookies | Sitemap

Follow element14

Powered by jive

A Premier Farnell Company
© 2009-2017 Premier Farnell Ltd. All Rights Reserved. ICP 备案号 10220084.
Premier Farnell Ltd, registered in England and Wales (no 00876412), registered office: Farnell House, Forge Lane, Leeds LS12 2NE