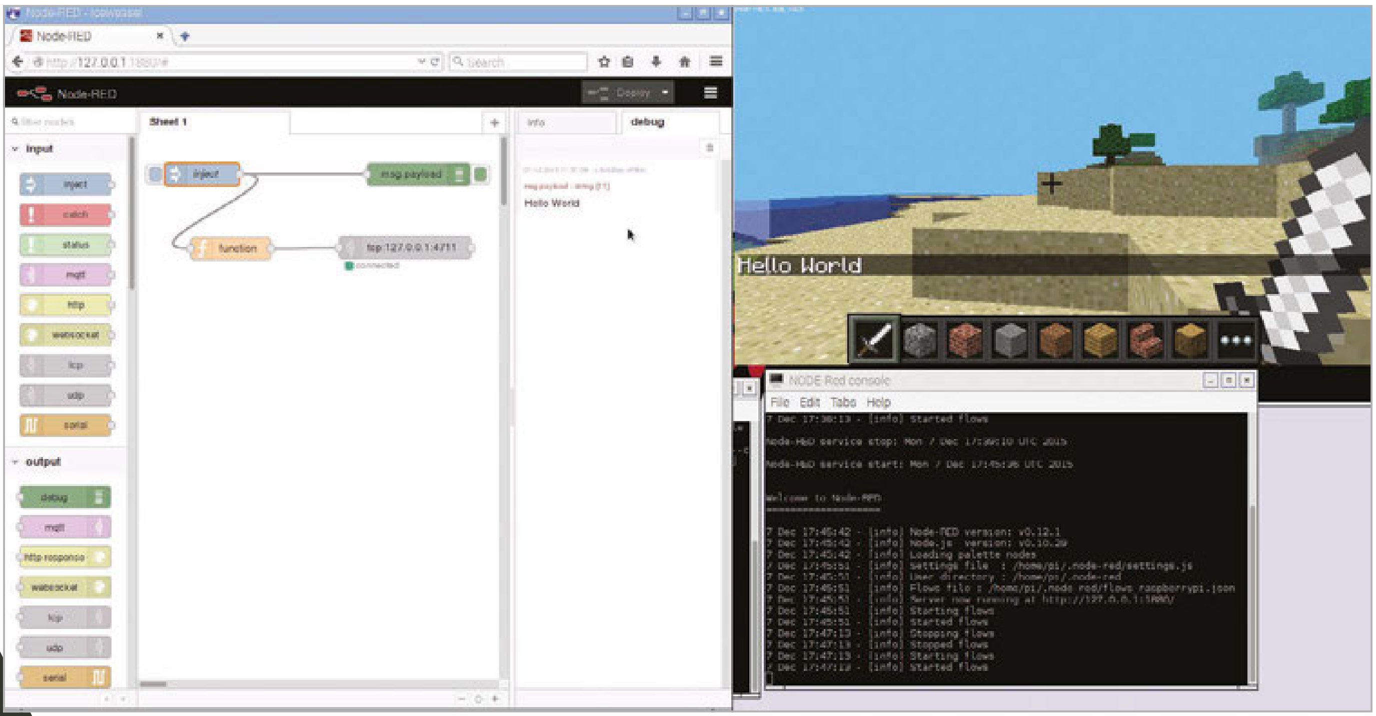
The MagPi ESSENTIALS

[CHAPTER ELEVEN] NODE-RED AND CONTROLLING MINECRAFT WITH JAVASCRIPT PART 1

JavaScript is an interpreted programming language and one of the main components of interactive websites. With Node-RED, you can easily use it to control Minecraft

Below Node-RED and Minecraft on one screen can look a bit busy. Once you're comfortable, however, Steve's world is under your control



```
7 Dec 17:38:13 - [info] Started flows

Node-RED service stop: Mon 7 Dec 17:39:10 UTC 2015

Node-RED service start: Mon 7 Dec 17:45:36 UTC 2015

Welcome to Node-RED
===================

7 Dec 17:45:42 - [info] Node-RED version: v0.12.1
7 Dec 17:45:42 - [info] Node.js  version: v0.10.29
7 Dec 17:45:42 - [info] Loading palette nodes
7 Dec 17:45:51 - [info] Settings file  : /home/pi/.node-red/settings.js
7 Dec 17:45:51 - [info] User directory : /home/pi/.node-red
7 Dec 17:45:51 - [info] Flows file : /home/pi/.node-red/flows_raspberrypi.json
7 Dec 17:45:51 - [info] Server now running at http://127.0.0.1:1880/
7 Dec 17:45:51 - [info] Starting flows
7 Dec 17:45:51 - [info] Started flows
7 Dec 17:47:13 - [info] Stopping flows
7 Dec 17:47:13 - [info] Stopped flows
7 Dec 17:47:13 - [info] Starting flows
7 Dec 17:47:13 - [info] Started flows
```

[MINECRAFT & NODE-RED]

Node-RED is a visual tool for wiring the Internet of Things. It takes care of technicalities, like GPIO access or internet protocols, and lets you focus on your workflow.

This article should give you a taste of a workshop Boris taught at a recent CamJam. Here, we'll show you just how easy Node-RED is to use. You can refer to the original tutorial (**magpi.cc/**1jLWFST) for a lot more detail and theory.

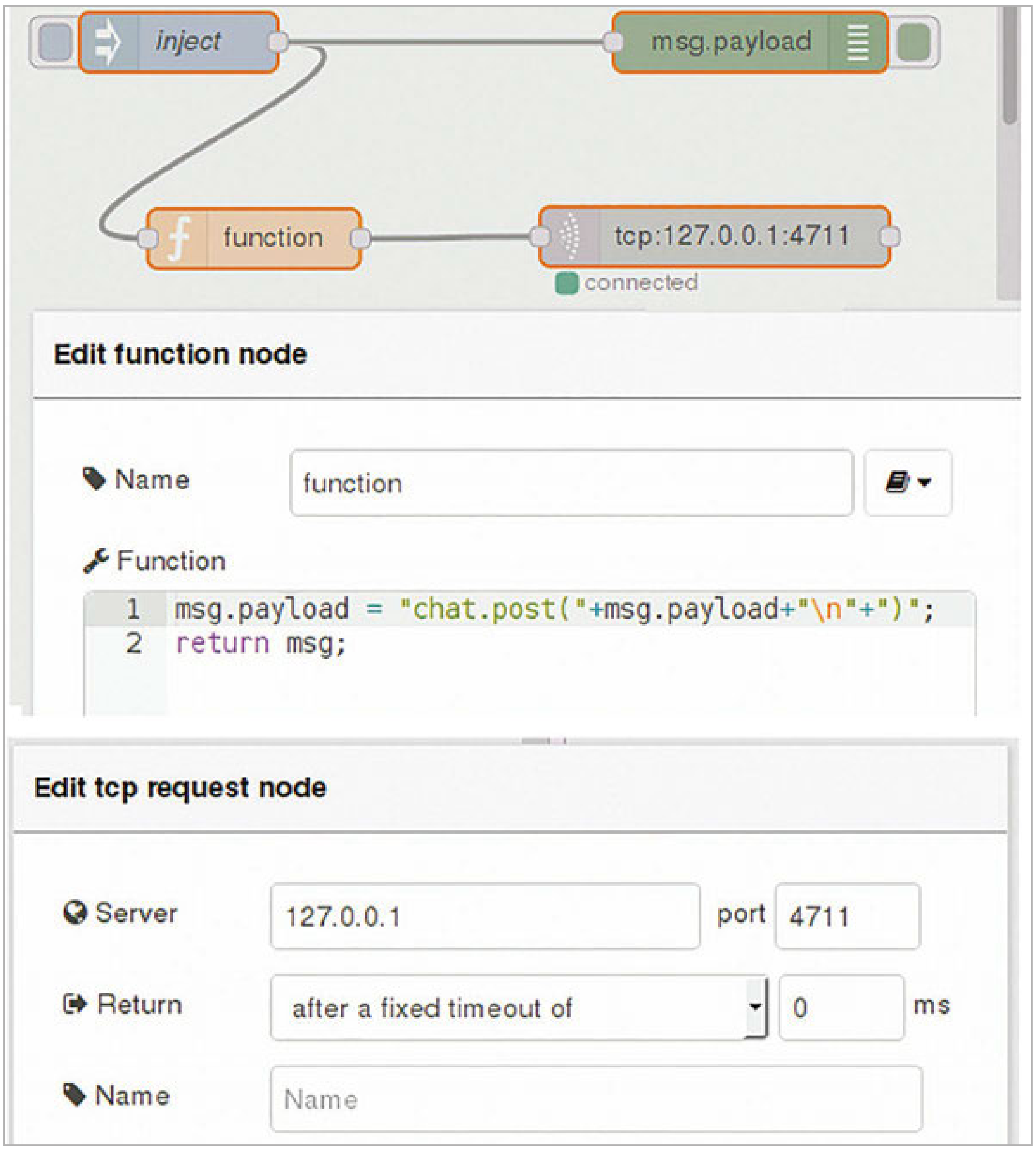
Preparation

Start Minecraft and open a new world. Fire up Node-RED from the **Programming** submenu on the Raspbian desktop. After a few moments, Node-RED is going to run as a service in the background. Open a web browser (Iceweasel works best with Node-RED) and direct it to 127.0.01:1880 to see the programming environment. On the left, there's a panel with nodes. These are visual components that you can use to build your flow. In the middle is the flow editor where you can simply drag and drop your flow together. On the right, you can switch between the help panel and the debug panel. Note that while 'debug' often has some negative connotation, it's your default text output in Node-RED.

[A TOOL FOR THE INTERNET OF THINGS]

Node-RED encapsulates great functionality in already existing nodes. Let the Twitter output node feed into the Minecraft chat, or the GPIO input node trigger your flow with a physical button!

Right Your first flow: the code you need to write into your function node, and configuration details for the TCP request node



Edit function node

Name: function

Function

```
msg.payload = "chat.post("+msg.payload+"\n"+")";
return msg;
```

Edit tcp request node

Server: 127.0.0.1 port 4711

Return: after a fixed timeout of 0 ms

Name: Name

Hello World in Minecraft

Drag and drop an inject node from the nodes panel into the flow editor. Once you've selected that inject node, you should see a general explanation about its functionality in the help panel – this is especially useful info when you select complex nodes with many different options.

Double-click your inject node in the flow editor. Once the associated dialogue opens, change the Payload to type 'string' and write 'Hello World' in the empty text field below.

The flow of information is modelled through the exchange of messages in Node-RED, which happens by passing along a variable called **msg**. It has two main properties: topic and payload. In simple terms, these could be interpreted like the subject and body of an email. Drag and drop a debug node from the nodes library into the flow editor.

Drag and drop a function node into the editor. This is the node type that allows you to directly interact with the **msg** object in JavaScript. In your function node, before the line with **return msg;**, write:

```
msg.payload = "chat.post("+msg.payload+")\n";
```

This line is going to take the incoming payload 'Hello World', and assign the new content **"chat.post(Hello World)\n"** to the variable. **chat.post** is a command from the Minecraft API, which you can read about in a file called **mcpi_protocol_spec.tx**t in the Minecraft API directory. The end of our command is indicated by a line break, the Unix character **\n**.

Drag and drop a TCP request node from the function section of the node panel. The server is 127.0.0.1, and the port is 4711. As we're not expecting a return value, we'll Return after a fixed timeout of 0ms. This connects Node-RED to the Minecraft TCP socket. If you're keen to understand what a socket is, have a look at the original workshop material: **magpi.cc/**1jLWFST.

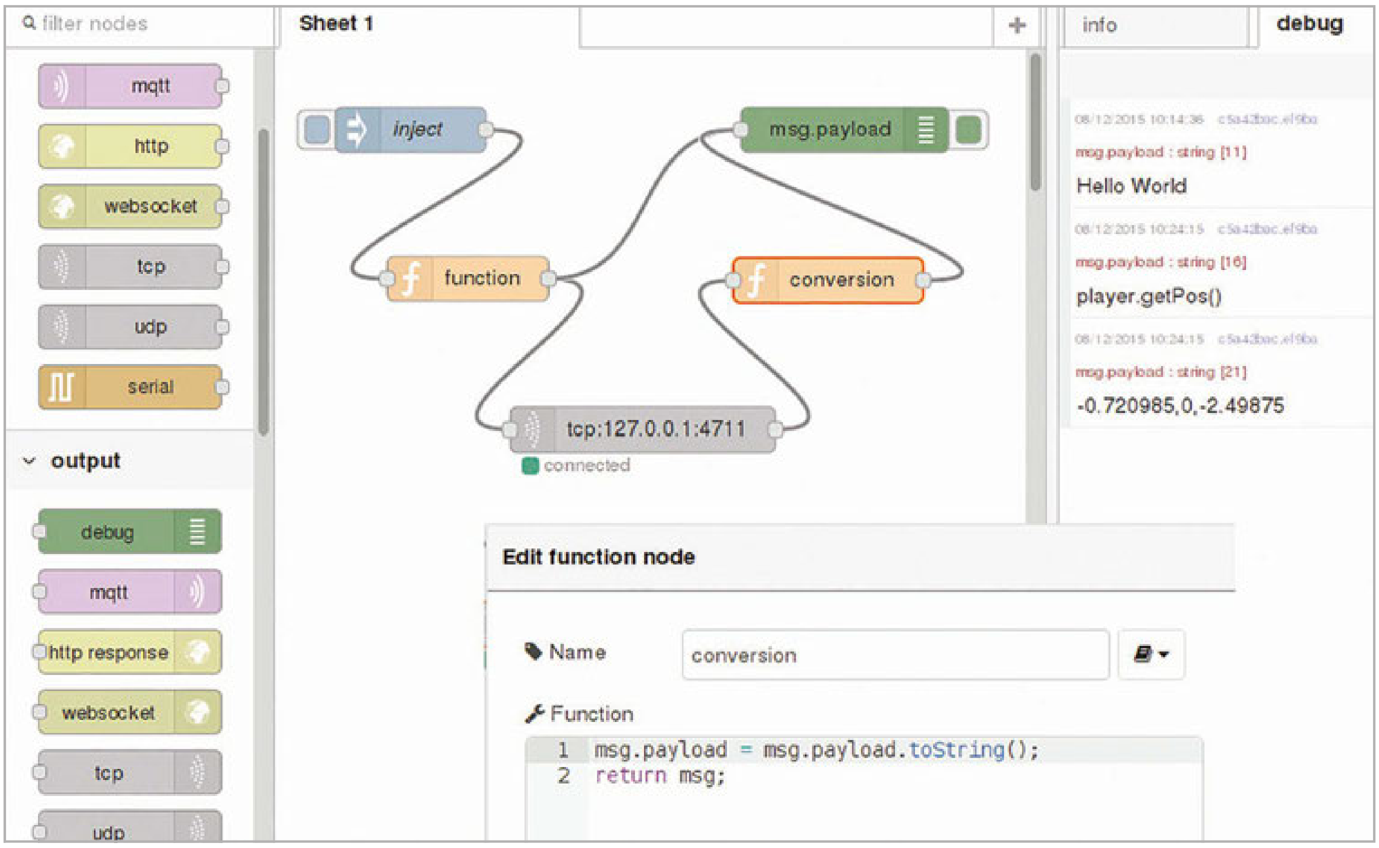
Connect the nodes by drawing connections, as in the screenshots here. Start your flow by pressing the red Deploy button in the upper-right corner. Trigger your flow by pressing the rounded rectangle to the immediate left of your inject button. Do you see your message in Minecraft?

Retrieve values from Minecraft

So far, our interaction with Minecraft has been rather one-directional. We just sent a command string that had an effect on the Minecraft world. Now we're going to modify our flow so we can query values like our own position via the API.

In the function node, before the line with **return msg;**, set:

```
msg.payload = "player.getPos()\n";
```



Name: conversion

```
msg.payload = msg.payload.toString();
return msg;
```

```
08/12/2015 10:14:36  c5a42bac.el9ba
msg.payload : string [11]
Hello World
08/12/2015 10:24:15  c5a42bac.el9ba
msg.payload : string [16]
player.getPos()
08/12/2015 10:24:15  c5a42bac.el9ba
msg.payload : string [21]
-0.720985,0,-2.49875
```

Instead of leaving the TCP request node after some time, we expect our Return 'when character is received', namely **\n.**

By default, the TCP request node returns a buffer, and we need to convert the information from Node-RED using:

```
msg.payload = msg.payload.toString();
```

in a new function node. The result of this new function node goes straight to our debug node.

Deploy your Node-RED flow. Now have a walk around Minecraft and trigger your flow with the inject node. Do you see what you expected in the debug panel?

Look out for the second part of this tutorial, which will cover aspects of event-driven development with Node-RED, including the interaction of Minecraft with the physical world.

Hello World

```
[{“id”:”945e5f77.182da8”,”type”:”function”,”z”:”a59a50d2
.5a65b”,”name”:”function”,”func”:”msg.payload = \”chat.
post(\”+msg.payload+\”\\n\”+\”)\”;\nreturn msg;”,”outputs”
:1,”noerr”:0,”x”:124,”y”:152,”wires”:[[“ec2d4ccc.7365b”]]}
,{“id”:”c5a42bac.ef9ba”,”type”:”debug”,”z”:”a59a50d2.5a65b
”,”name”:””,”active”:true,”console”:”false”,”complete”:”fa
lse”,”x”:374,”y”:53,”wires”:[]},{“id”:”f1098e71.8eb9d”,”ty
pe”:”inject”,”z”:”a59a50d2.5a65b”,”name”:”inject”,”topic”:
””,”payload”:”Hello World”,”payloadType”:”string”,”repeat”
:””,”crontab”:””,”once”:false,”x”:86,”y”:53,”wires”:[[“945
e5f77.182da8”,”c5a42bac.ef9ba”]]},{“id”:”ec2d4ccc.7365b”,”
type”:”tcp request”,”z”:”a59a50d2.5a65b”,”server”:”127.0.0
.1”,”port”:”4711”,”out”:”time”,”splitc”:”0”,”name”:””,”x”:
356,”y”:151,”wires”:[[]]}]
```

Download

Hello World
magpi.cc/1Qr4USK

Player Position
magpi.cc/1Qr4Xht

Player Position

```
[{"id":"945e5f77.182da8","type":"function","z":"a59a50d2.
5a65b","name":"function","func":"msg.payload = \"player.
getPos()\\n\";\nreturn msg;","outputs":1,"noerr":0,"
x":128,"y":153,"wires":[["ec2d4ccc.7365b","c5a42bac.
ef9ba"]]},{"id":"c5a42bac.ef9ba","type":"debug","z":"a59
a50d2.5a65b","name":"","active":true,"console":"false","
complete":"false","x":378,"y":54,"wires":[]},{"id":"f109
8e71.8eb9d","type":"inject","z":"a59a50d2.5a65b","name":"i
nject","topic":"","payload":"Hello World","payloadType":"s
tring","repeat":"","crontab":"","once":false,"x":90,"y":54
,"wires":[["945e5f77.182da8"]]},{"id":"ec2d4ccc.7365b","ty
pe":"tcp request","z":"a59a50d2.5a65b","server":"127.0.0.1
","port":"4711","out":"char","splitc":"\\n","name":"","x":
244,"y":253,"wires":[["f4f5b8bd.154d68"]]},{"id":"f4f5b8bd
.154d68","type":"function","z":"a59a50d2.5a65b","name":"c
onversion","func":"msg.payload = msg.payload.toString();\
nreturn msg;","outputs":1,"noerr":0,"x":367,"y":154,"wires
":[["c5a42bac.ef9ba"]]}]
```