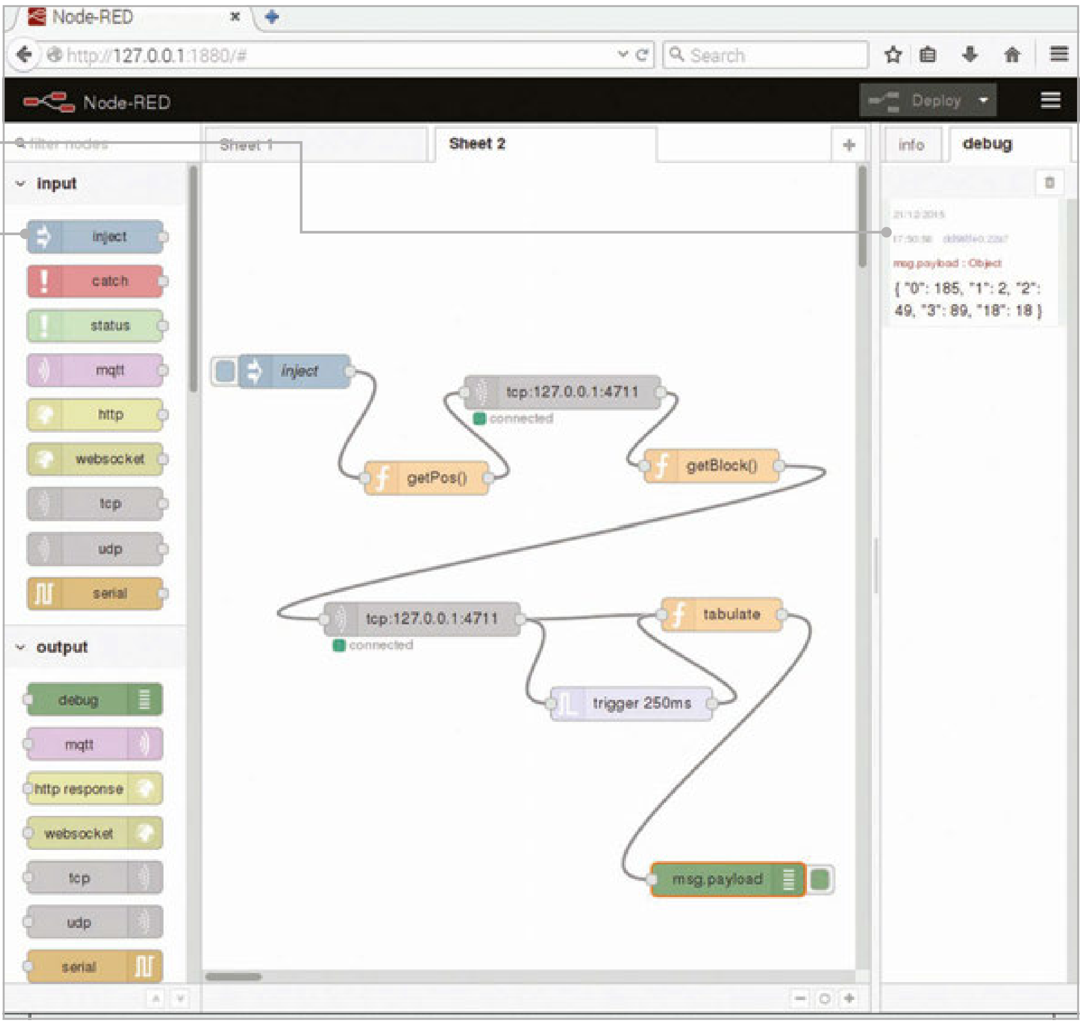
The MagPi ESSENTIALS

[CHAPTER SIX] NODE-RED AND CONTROLLING MINECRAFT WITH JAVASCRIPT PART 2

With Node-RED you can mine like a pro. Give Steve an additional sense to see if there are any exciting blocks in the neighbourhood

This is a summary of the block types. For example, we have 185 blocks of air (ID "0") around us

This is the flow that tabulates all block types around you



msg.payload : Object

{ "0": 185, "1": 2, "2": 49, "3": 89, "18": 18 }

You'll Need

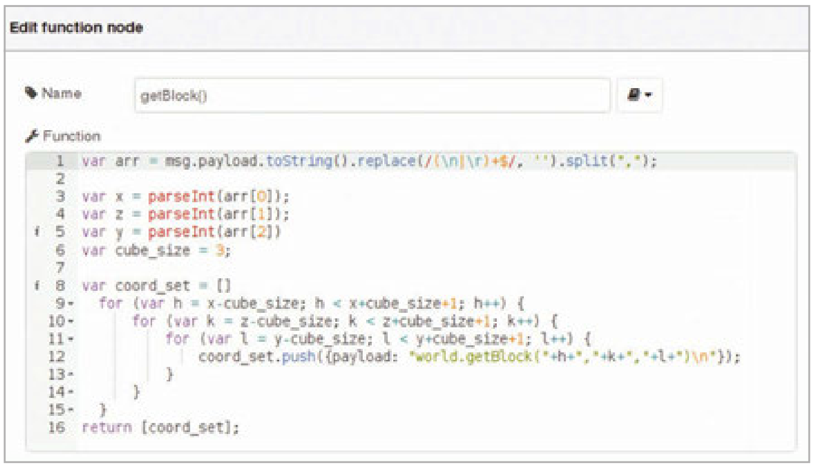
- Raspbian **raspberrypi.org/downloads**
- Node-RED – pre-installed on Raspbian **nodered.org**
- Iceweasel – installed via terminal **wiki.debian.org/Iceweasel**

Part one of this mini-series introduced linear flows with Node-RED and Minecraft, and you should go through it before embarking on the next adventure. In part two, you'll dive deeper into Node-RED and JavaScript. Learn how to react to events, loop over a set of variables, and retain values from previous executions of a node in your flow. With this tutorial, we've got you well covered for any further explorations with Node-RED!

This tutorial takes off where we left in part one. In that tutorial, we taught you how to make a Minecraft 'Hello World' and how to read data from it.

Preparation

To get back to where you were, start Minecraft and open a world. Fire up Node-RED from the Programming sub-menu. Open a web browser (Iceweasel works best with Node-RED) and direct it to **127.0.0.1:1880**


Edit function node

Name: getBlock()

Function

```
 1  var arr = msg.payload.toString().replace(/(\n|\r)+$/, '').split(",");
 2
 3  var x = parseInt(arr[0]);
 4  var z = parseInt(arr[1]);
 5  var y = parseInt(arr[2])
 6  var cube_size = 3;
 7
 8  var coord_set = []
 9    for (var h = x-cube_size; h < x+cube_size+1; h++) {
10        for (var k = z-cube_size; k < z+cube_size+1; k++) {
11            for (var l = y-cube_size; l < y+cube_size+1; l++) {
12                coord_set.push({payload: "world.getBlock("+h+","+k+","+l+")\n"});
13            }
14        }
15    }
16  return [coord_set];
```

Above The 'getBlock()' function. Lines 8-16 populate an array with 343 (7³) elements, each carrying a payload with a world.getBlock statement for the TCP node

to see the programming environment. Arrange the windows so you can see what's going on in Node-RED and the Minecraft world.

You should see your last flow, the one that retrieved Steve's position and printed the coordinates in the debug panel. If not, you can download the Player Position flow from the Node-RED flows directory (**magpi.cc/1Qr4Xht**) and import it before you proceed.

Calculate and store the coordinates

Edit the **conversion** function node, and rename it to **getBlock()** to reflect its new role. There's a bit of magic in this one. The **`replace()`** and **`split()`** functions remove the trailing line break from the Minecraft message and break the components of Steve's position into character strings. These are put into an array – a variable that saves a few values in a list – which is called **`arr`**. Convert the strings from the list into whole numbers and assign them to variables **`x`**, **`z`**, and **`y`** with **`parseInt()`** to be able to do calculations on them.

Define a variable **`cube_size`** that keeps half of the size of a cube in whose centre we're located. With three nested **`for`** loops, iterate over this cube that spans from *x - cube_size*, *z - cube_size*, *y - cube_size* to *x + cube_size*, *z + cube_size*, *y + cube_size*. For each tuple (**`h`**, **`k`**, **`l`**), create a **`payload`** object that carries a string of the format **`"world.getBlock(h,k,l)\n"`**, and store it as an element of an array, **`coord_set`**. The **function** node returns **`[coord_set]`**, which makes Node-RED invoke the next flow element for each list element.

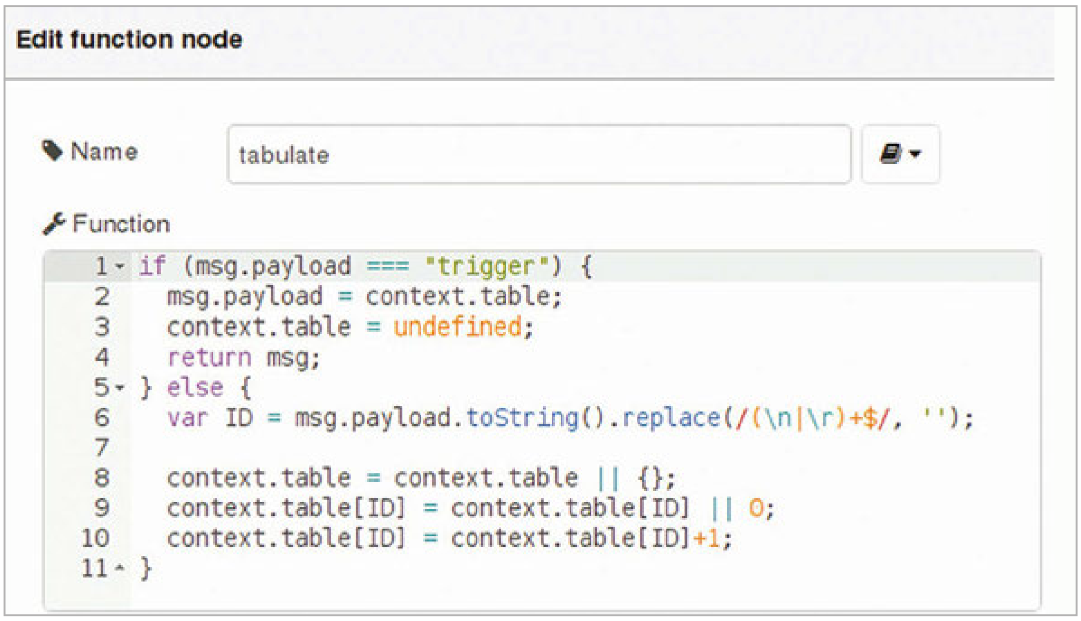
Drag and drop another **TCP request** node from the function section of the node panel. As before, the server is 127.0.0.1 and port 4711, and we expect the returned message to be finished with **\n**.

Drag and drop a **trigger** node. Configure it such that it sends nothing and waits for 250ms, and then sends the string payload **"trigger".** Check the extend delay option. This essentially means that as long as there are incoming messages at least every 250ms, the node will remain silent, but if there are no new messages, we send the trigger message. In our case, this will indicate that the TCP request node is done with all requests.

A final function node called **tabulate** does a few clever bits. First, it checks if the incoming payload is **"trigger"**. If that's the case, it publishes the results of a variable **`context.table`** as payload, but deletes **`context.table`**. Remember, that's only happening after all TCP requests are finished. If TCP requests are still incoming, the second part after the **`else`** statement is relevant: the request buffer with a Minecraft block ID is converted into a string. Have a look at **magpi.cc/1PvHAh4** to see what they mean; for example, 0 is air and 17 is wood. If it's the first time we're entering this part of the code, we'll create an empty variable **`context.table`**: **`context`** is a special variable in Node-RED that's persistent between different times a node is called up. If there's no table entry for ID, we'll create one and set it to 0, otherwise we take its current value, then we add 1. With that, the node expects more incoming IDs until the trigger message arrives.

Ultimately, we'll have a directory of elements which we can print with a debug node; this is happening in the **`return`** statement in the **tabulate** node. The block type IDs are shown in quotes; the number of occurrences follows the colon. These should add up to 343, the volume of the cube around us.

We hope that this tutorial has helped you understand a few concepts that aren't immediately obvious when using Node-RED for the first time. In part one, we promised interaction with the real world... well... just add the GPIO node. We're sure you'll figure it out!



```
1  if (msg.payload === "trigger") {
2    msg.payload = context.table;
3    context.table = undefined;
4    return msg;
5  } else {
6    var ID = msg.payload.toString().replace(/(\n|\r)+$/, '');
7
8    context.table = context.table || {};
9    context.table[ID] = context.table[ID] || 0;
10   context.table[ID] = context.table[ID]+1;
11 }
```

Left The 'tabulate' function. Lines 1-4 are executed if the trigger node doesn't receive any more results from the TCP node

Download

Block Inventory
magpi.cc/1n59yKr

Block Inventory

```
[{"id":"533edf4.facc12","type":"function","z":"e13b1b02.1ec4e8",
"name":"getPos()","func":"msg.payload = \"player.getPos()\\n\";\
nreturn msg;","outputs":1,"noerr":0,"x":207,"y":151,"wires":[["714
29f10.8ebd6"]]},{"id":"dd58ffe0.22a7","type":"debug","z":"e13b1b02
.1ec4e8","name":"","active":true,"console":"false","complete":"pay
load","x":642,"y":278,"wires":[]},{"id":"cc9ed7e5.336128","type":"
inject","z":"e13b1b02.1ec4e8","name":"inject","topic":"","payload"
:"trigger","payloadType":"string","repeat":"","crontab":"","once":
false,"x":98,"y":53,"wires":[["533edf4.facc12"]]},{"id":"71429f10.8
ebd6","type":"tcp request","z":"e13b1b02.1ec4e8","server":"127.0.0.
1","port":"4711","out":"char","splitc":"\\n","name":"","x":329,"y":
74,"wires":[["a9afe2c4.56502"]]},{"id":"a9afe2c4.56502","type":"fun
ction","z":"e13b1b02.1ec4e8","name":"getBlock()","func":"var arr =
msg.payload.toString().replace(/(\\n|\\r)+$/, ‘’).split(\",\");\n\
nvar x = parseInt(arr[0]);\nvar z = parseInt(arr[1]);\nvar y =
parseInt(arr[2])\nvar cube_size = 3;\n\nvar coord_set = []\n  for
(var h = x-cube_size; h < x+cube_size+1; h++) {\n      for (var k
= z-cube_size; k < z+cube_size+1; k++) {\n          for (var l =
y-cube_size; l < y+cube_size+1; l++) {\n              coord_set.
push({payload: \"world.getBlock(\"+h+\",\"+k+\",\"+l+\")\\n\"});\n
}\n      }\n  }\nreturn [coord_set];","outputs":1,"noerr":0,"x":46
3,"y":140,"wires":[["9713fdf5.68ec"]]},{"id":"9713fdf5.68ec","type
":"tcp request","z":"e13b1b02.1ec4e8","server":"127.0.0.1","port":
"4711","out":"char","splitc":"\n","name":"","x":203,"y":277,"wire
s":[["4958e94e.b6a718","e8e39686.171c68"]]},{"id":"4958e94e.b6a718"
,"type":"function","z":"e13b1b02.1ec4e8","name":"tabulate","func":"
if (msg.payload === \"trigger\") {\n  msg.payload = context.table;\n
context.table = undefined;\n  return msg;\n} else {\n  var ID = msg.
payload.toString().replace(/(\\n|\\r)+$/, ‘’);\n  \n  context.table
= context.table || {};\n  context.table[ID] = context.table[ID] ||
0;\n  context.table[ID] = context.table[ID]+1;\n}","outputs":1,"noer
r":0,"x":470,"y":278,"wires":[["dd58ffe0.22a7"]]},{"id":"e8e39686.17
1c68","type":"trigger","z":"e13b1b02.1ec4e8","op1":"1","op2":"trigge
r","op1type":"nul","op2type":"val","duration":"250","extend":true,"u
nits":"ms","name":"","x":362,"y":354,"wires":[["4958e94e.b6a718"]]}]
```