

# PART1: Getting started with node-RED

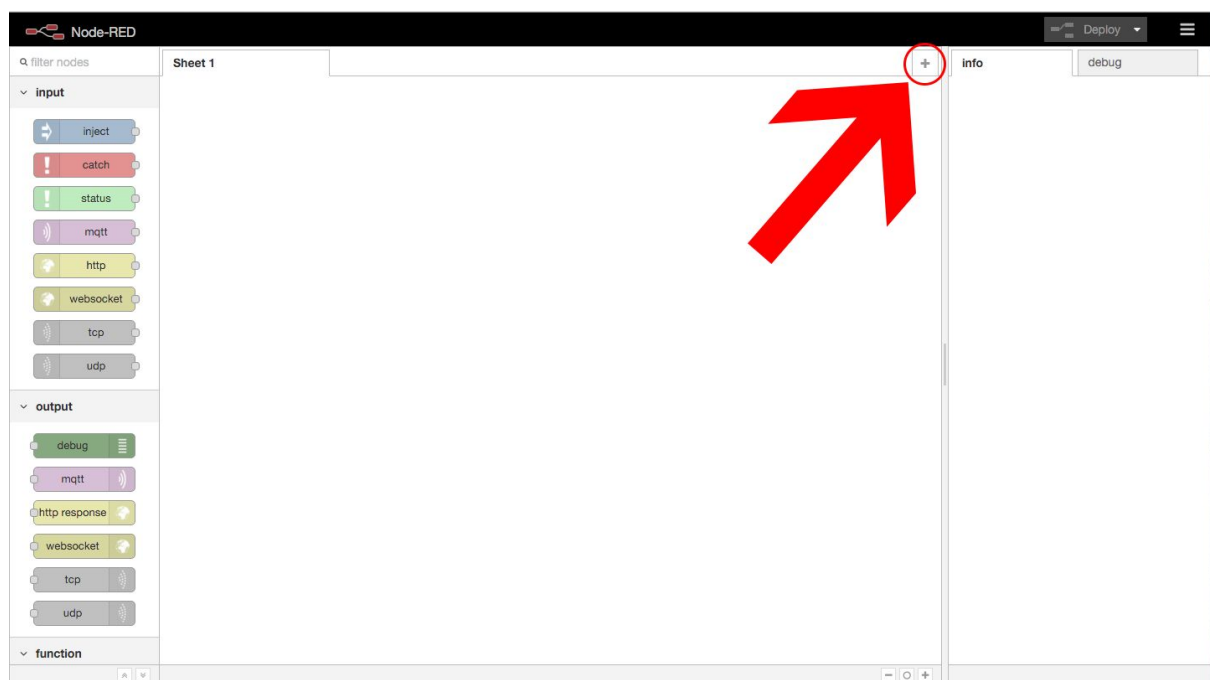
*“Node-RED is a tool for wiring together hardware devices, APIs and online services in new and interesting ways.”*

Prerequisites:

Download and install the open energy playground either as an virtual machine or as an Raspberry Pi. See <https://op-en.se>

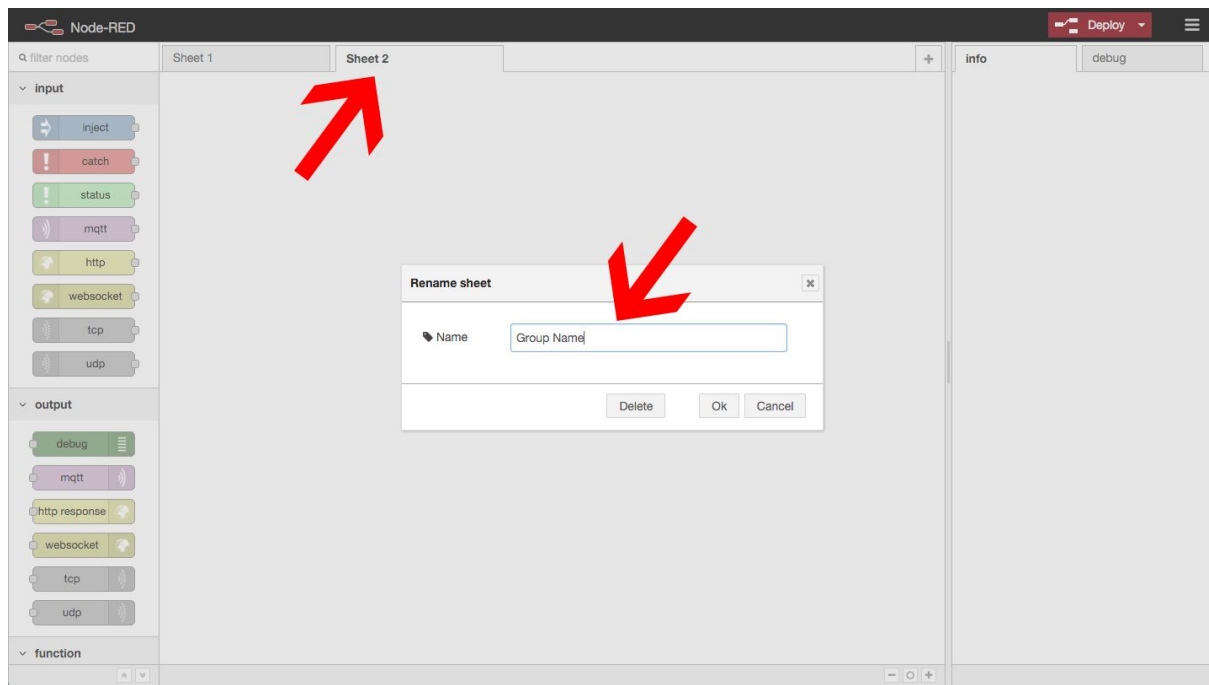
We start by logging into the node-red on our own or the workshop server running on the raspberry pi or the Raspberry Pi.

1. Open a browser and go to the ip of the server port 1880 e.g. <http://192.168.1.235:1880>

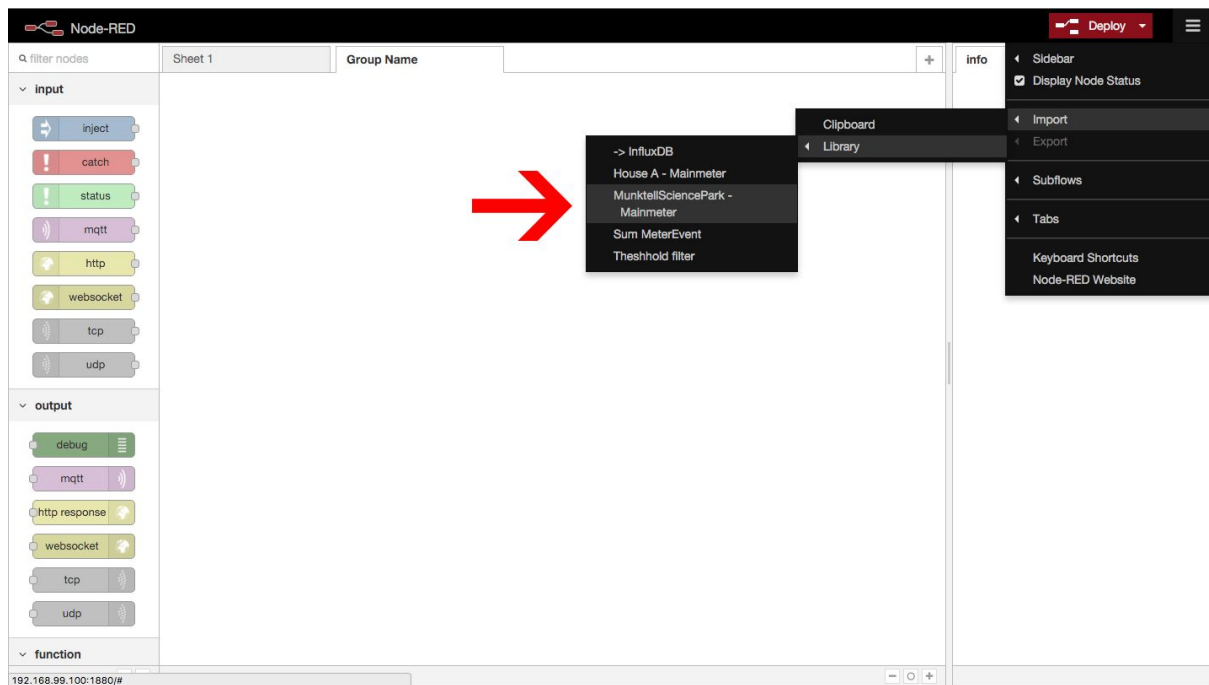


This is the node-red control panel! Now we are going to add a small “flow” that subscribes to energy updates from an energy meter in a house in Eskilstuna.

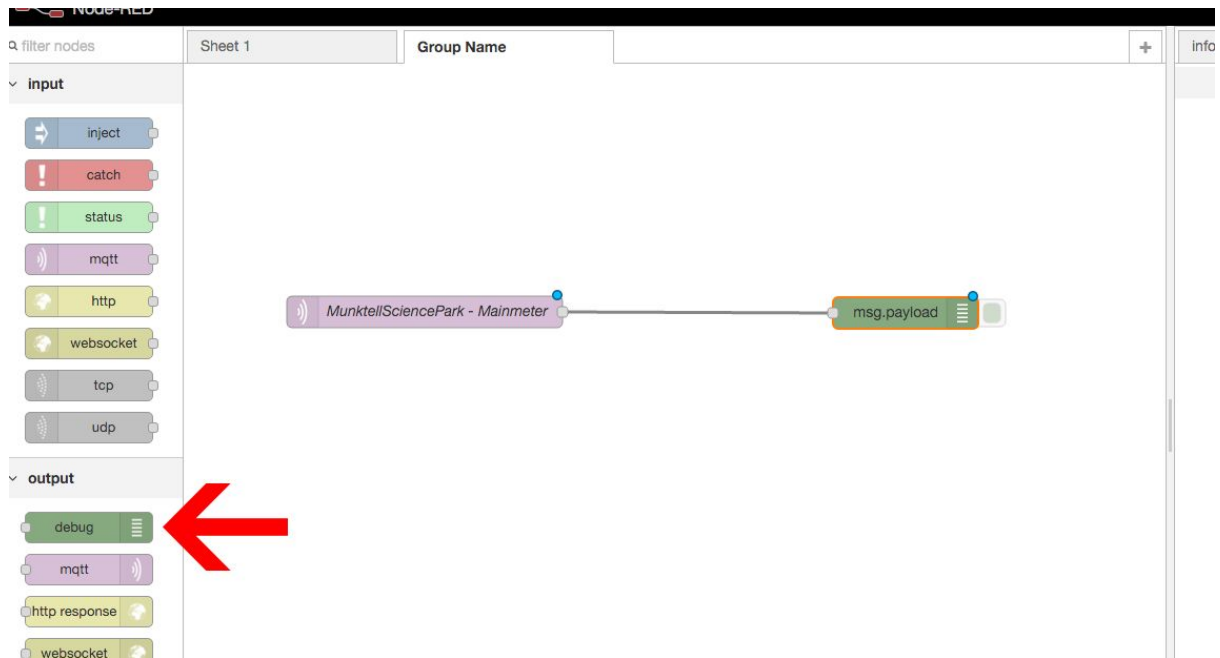
2. Create a new tab by pressing the + in the upper right corner, as shown in the image above.



3. Double click on the name of the new sheet (in this image Sheet 2). This will open up a dialog where we can name the sheet. Call it something new! Perhaps the name of your group? Then press OK.

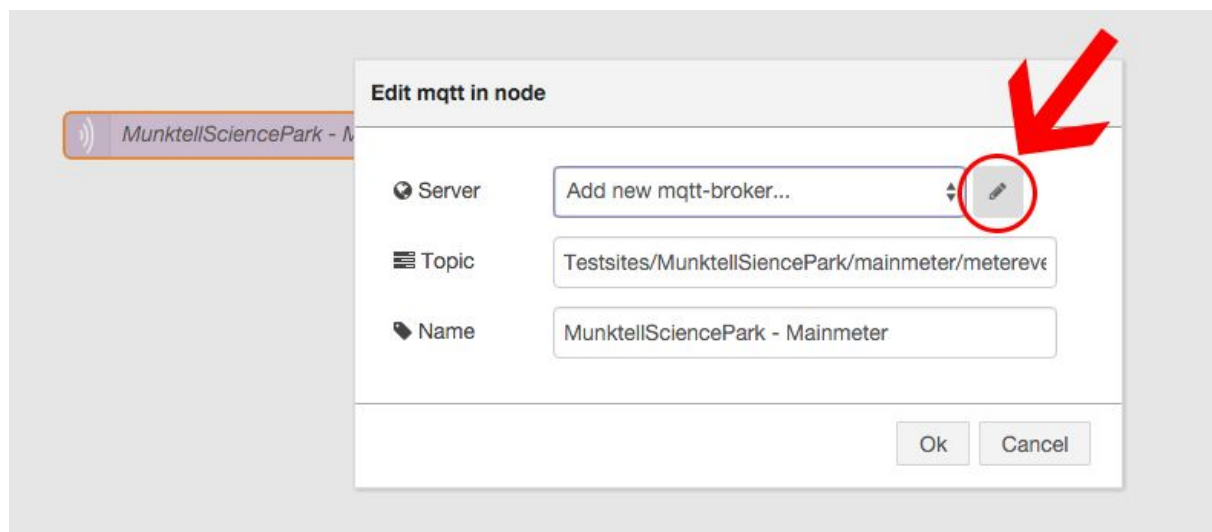


5. To get started quickly we can import a node that is already configured with the name of the meter in Eskilstuna. Under the hamburger menu in the upper right corner, navigate to “MuntellSciencePark - Mainmeter” and press it! This will create an mqtt node that you can place on the sheet. Place it anywhere on the sheet. You can instead of also select one of the smart plugs supplied in the box. But then you also need to plug them into an outlet.



6. Drag a debug node (from where the arrow points in the image), and place it anywhere on the sheet.

7. Connect the MunktellSciencePark - Mainmeter node with the debug node (now called msg.payload) by dragging the wire from the right side of the left node, to the left side of the right node. You should end up with something like the image above, where the two nodes are connected together with a wire.



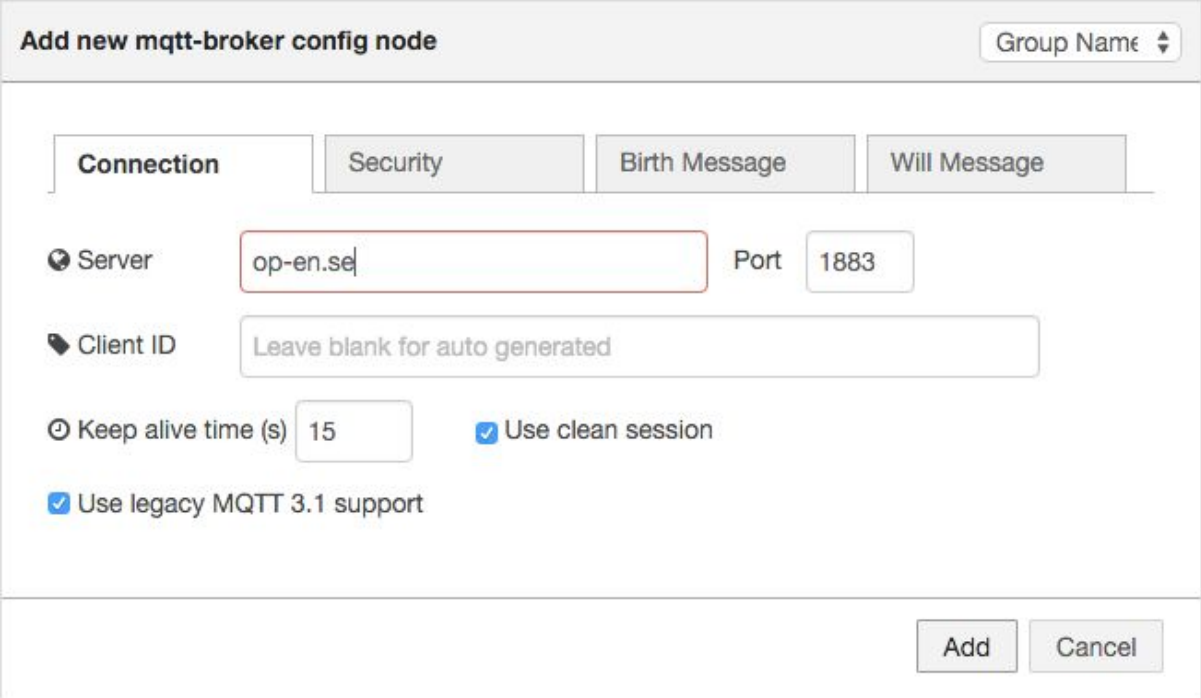
8. Double click the MunktellSciencePark node. This opens up a dialog where we can edit the settings of the node. The settings are:

- a. What server will this data come from? (this we need to configure)
- b. What topic on that server are we listening to? (this is already configured for us since we took this node from the library)
- c. What is the name of the node? (This is just to make it look nice on the screen)

Make sure that the broker called Open Energy or op-en.se is selected and jump to 14.  
If there is none we need to configure a new server.

9. In the server dropdown list, select “Add new mqtt-broker...”

10. Press the edit button (the pen) next to the server dropdown, as marked in the image above.



The screenshot shows a dialog box titled "Add new mqtt-broker config node". It has a "Group Name" dropdown at the top right. Below the title bar are four tabs: "Connection", "Security", "Birth Message", and "Will Message". The "Connection" tab is active. It contains the following fields and controls:

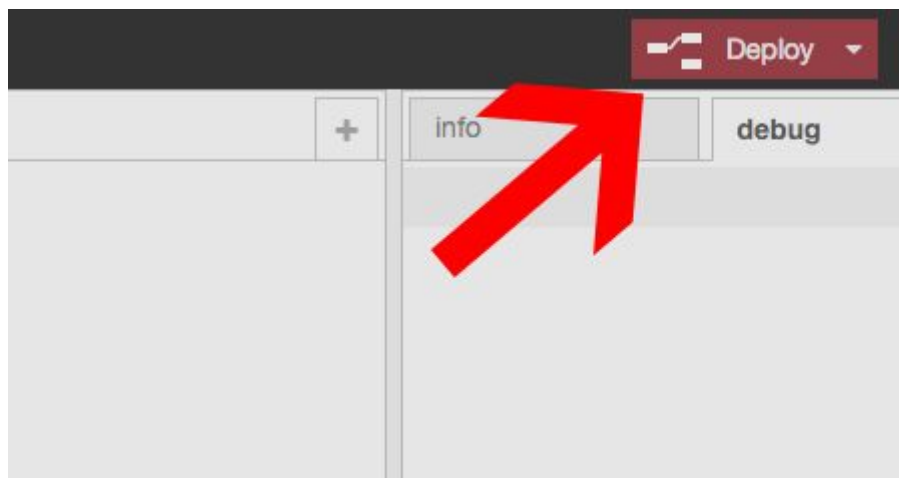
- Server:** A text field containing "op-en.se".
- Port:** A text field containing "1883".
- Client ID:** A text field with the placeholder text "Leave blank for auto generated".
- Keep alive time (s):** A text field containing "15".
- Use clean session:** A checked checkbox.
- Use legacy MQTT 3.1 support:** A checked checkbox.

At the bottom right of the dialog are two buttons: "Add" and "Cancel".

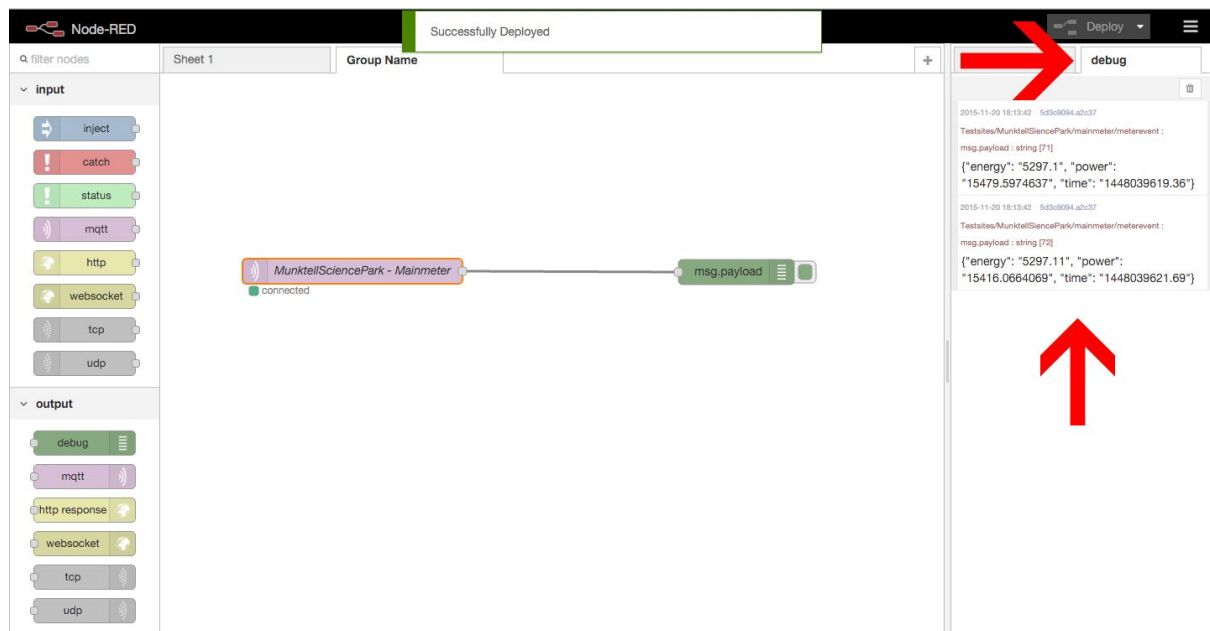
11. In the server field, enter “op-en.se”, and leave the rest of the settings as they are.

12. When the dialog looks like the screen above, press “Add”

13. Press Ok, finishing the edit of the mqtt node.



14. Now everything is done! Press the deploy button in the upper right corner.



15. You should get a message saying “successfully deployed”, like the screen above. Under the “Debug” tab to the right, data messages should now be pouring in.

We are now watching LIVE how much energy is being spent in the Munktell Science Park in Eskilstuna.

A message might look like this:

```
{ "energy": "5319.7", "power": "25265.9571994", "time": "1448042745.73" }
```

### Time

is the time of this message, as a Unix Timestamp. This is a common way of representing time in computers. It is defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970. If we convert this to a normal date, using [www.unixtimestamp.com](http://www.unixtimestamp.com), we see that it is equivalent to Fri, 20 Nov 2015 18:05:45 (UTC). The time that this tutorial was written :)

### Energy

is the accumulated energy in kWh that the Munktell Science Park has used.

### Power

is the current power of the build in watts (w). At the moment of writing, ~25kW of power was being used.

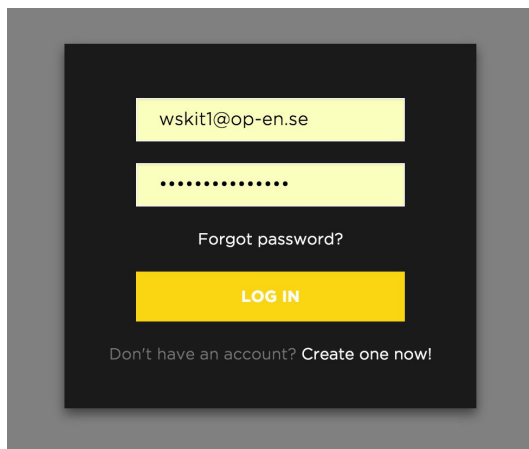
This is the end of the first tutorial.

A quick recap! What we have learned:

1. We imported an mqtt node that listens to a specific energy meter from a house in Eskilstuna.
2. We configured what server these updates are coming from.
3. We connected that data to the debug output, so that we could see what data was coming in.

In the next tutorial, we will connect this data to something new. Doing something with it instead of just looking at it!

## PART2: Programming the Particle.io

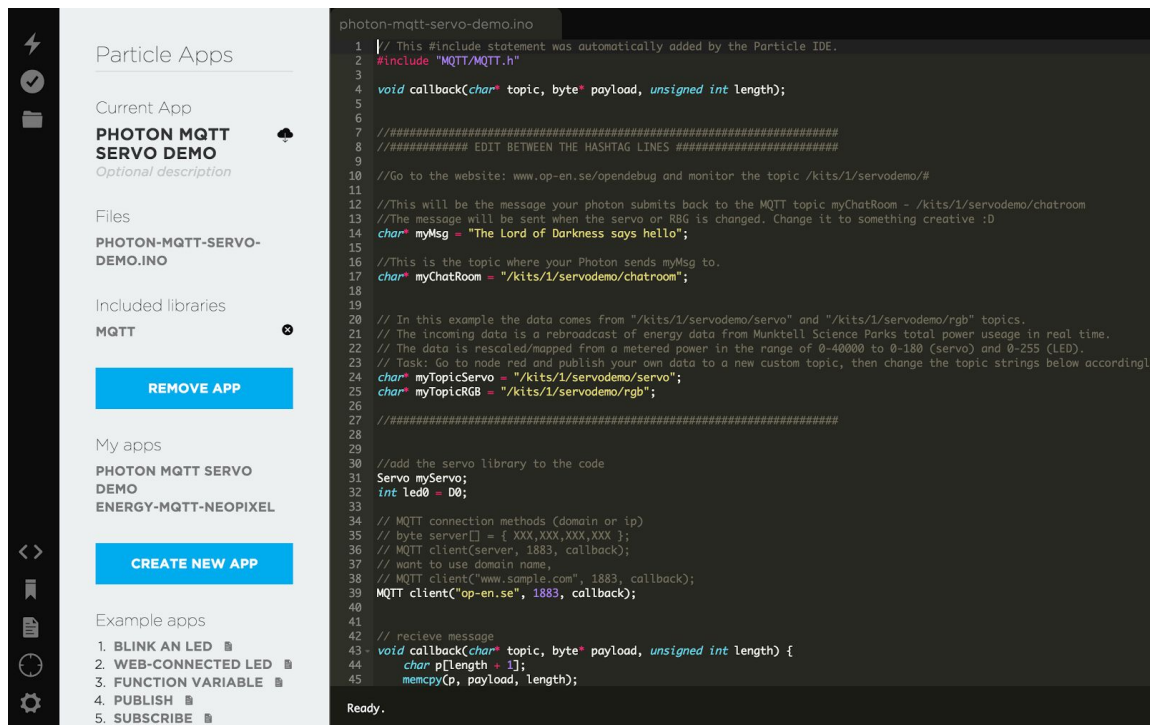


1. Visit <https://particle.io/build>
2. Log in with the following credentials:

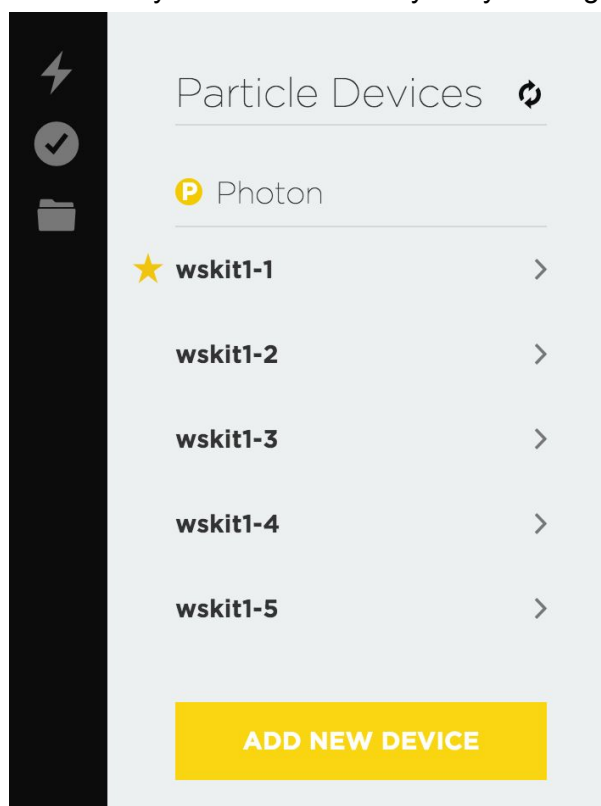
User: [wskit1@op-en.se](mailto:wskit1@op-en.se)

Password: internetdagarna

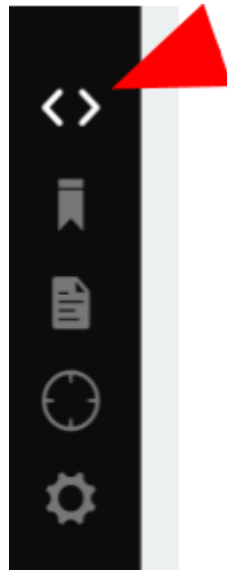
3. You will be presented with a view that looks like this:



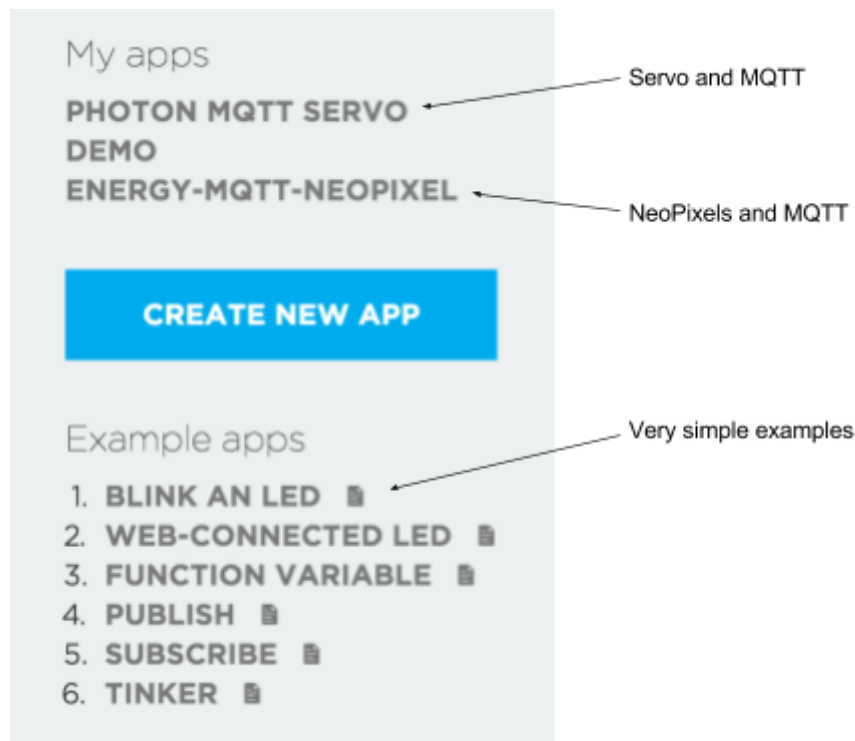
4. First, select a device that you want to use. This is done by clicking the cross-hair in the bottom left corner to show devices for this account. Here you will find a list of devices. Select the device you have in front of you by making it starred.



5. Loading an example “Hello World” application: Now, click the code button to load the first example:



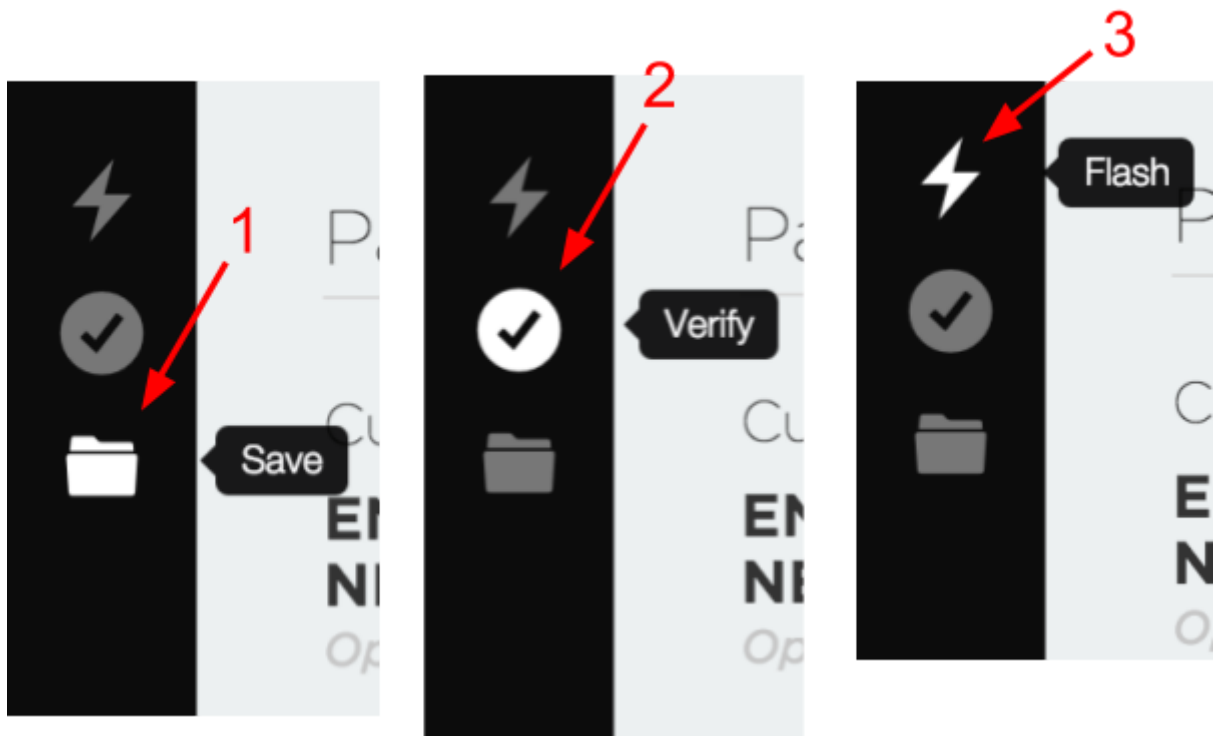
Now you can choose a very simple example like blinking a led. We have also prepared some more involving demos for connecting to MQTT, controlling servos and neopixels:



6. When you are ready to deploy some code. First save your changes by clicking the folder icon, then verify your code and correct any errors. Last but not least, flash the



firmware to the hardware:



As a preparation to part 3 we will now open the “Photon MQTT servo demo”. In the editor then if not already done set the server on line 39 to point at our local server “192.168.1.234”

Also change the topic on line 25 to /myapp/rgb and line 24 to /myapp/servo


## PART 3: Connecting data to a visualization

So we will now return to NodeRED again as in part one.

1. First create a new tab calling it “part 3”.
2. Add an inject node from the nodes on the left.
3. Click on the node to open up its settings.


4. Select string in the payload option. And enter 0,0,255 as the string to be sent on the row just below. As the topic enter /myapp/rgb

### Edit inject node


 Payload

string

(0,0,255)


 Topic

/myapp/rgb

 Repeat

none

☐ Fire once at start ?

 Name

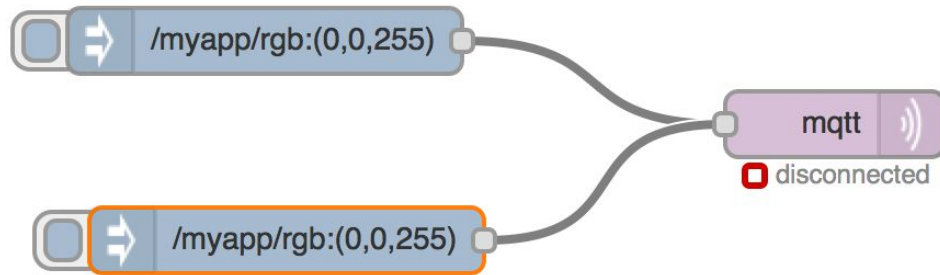
name

**Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

Ok

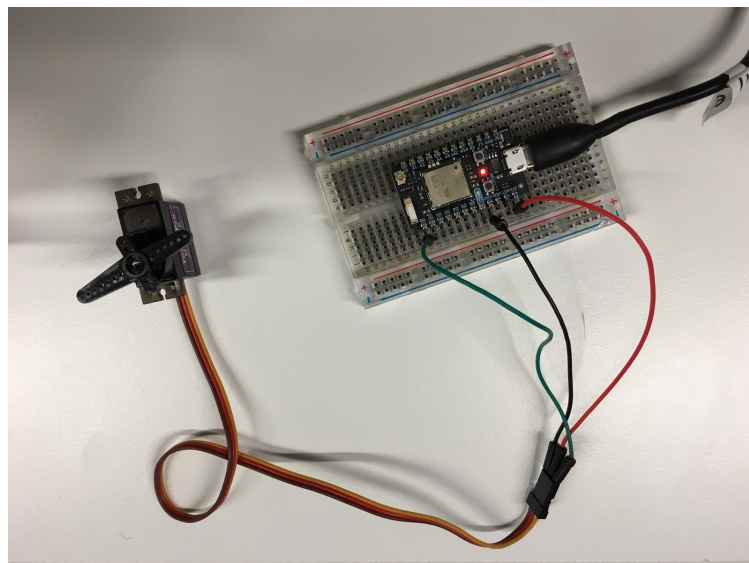
Cancel

5. Now copy the node and paste in a copy of it by using the copy pasted buttons on your computer (⌘c and ⌘v on OSX and ctrl-c ctrl-v on windows).
6. Double click the new node and change the string to be sent to 0,0,0.
7. Now add an outgoing mqtt node. Select the local broker (usually called localhost) in its config and connect both inject buttons output to its input.
8. Now press deploy.



If everything went well you should now be able to turn the LED on the photon on off by pressing the two input nodes.

10. Next select the “extract power” node from the library and place it in your sheet.
11. Then add on of the plugwise smart plugs in from the library.
12. Add one more MQTT output node and set the broker to localhost and the topic to /myapp/servo. Also label it servo.
13. Connect the smart plug node to the extract power node and then the “extract power” to the Servo node.



Finally connect the servo to the photon as in the picture. Red cable to the Vin, black to the GND and orange to the D0. If everything went well your servo should move in relation to the power use of the plug. If you don't see much movement you can add a range node just

before the servo node and rescale the direction of the servo i relation to the power value. Remember that you can always use the debug node to see what data the nodes outputs.

A live version of this document is also available at:

<https://docs.google.com/document/d/1es84qU-NZ3fbVRDDb-QjBd-LHrFasZYOKPYN28B1MKs/edit?usp=sharing>