Bohunt School *(Wokingham)*

Controlling LEDs using a Temperature Value

This practical session should be a bit of fun for you. It involves using an output from the weather reporting station to operate a set of three LEDs.

Objective

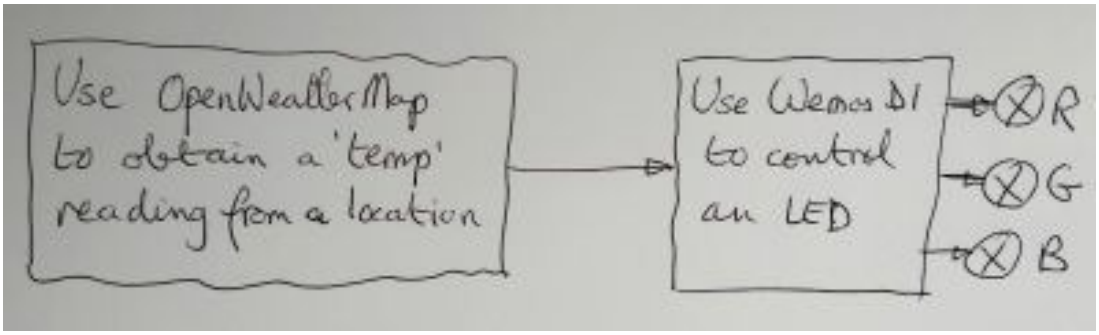
The main objective for this tutorial is to show how a temperature value (obtained from the Weather Reporting Station you built in a previous exercise) can be used to operate a set of three coloured LEDs (like the ones you used when you built the Traffic Light node).

The way the system will work is… if the temperature is greater than 30 degrees centigrade a 'red' LED is turned ON. If the temperature is greater than 10 degrees centigrade and less than 30 a 'green' LED is turned ON and if the temperature is less than 10 degrees centigrade a 'blue' LED is turned ON. Obviously you can use different temperature bands to match the geographical location where you are obtaining/reading the ambient temperature.

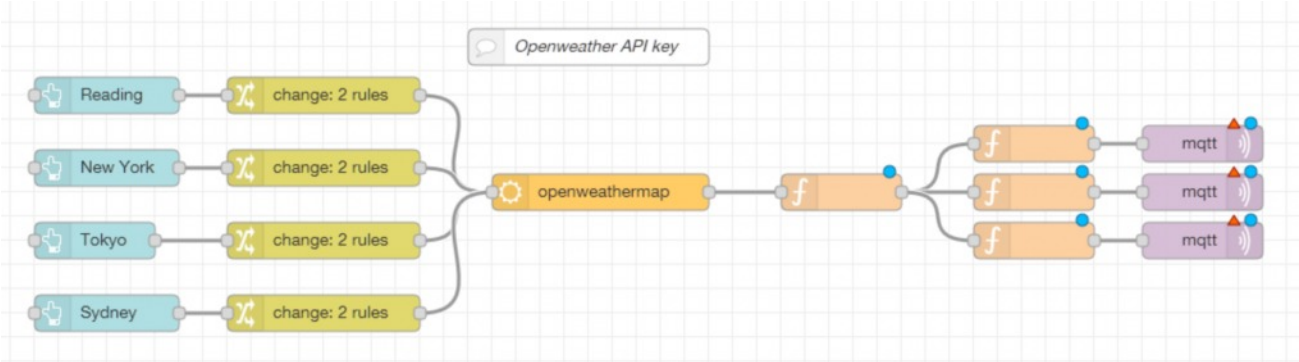
A secondary objective is to show how pseudo code can be used to rough-out a solution before coding using your chosen language (probably JavaScript).

Block Layout of the System

Here's a rough sketch of the basic system.



Here's a screen-shot of the Node-RED flow that is used to extract the temperature and send it, via MQTT, to the Wemos D1 Mini.

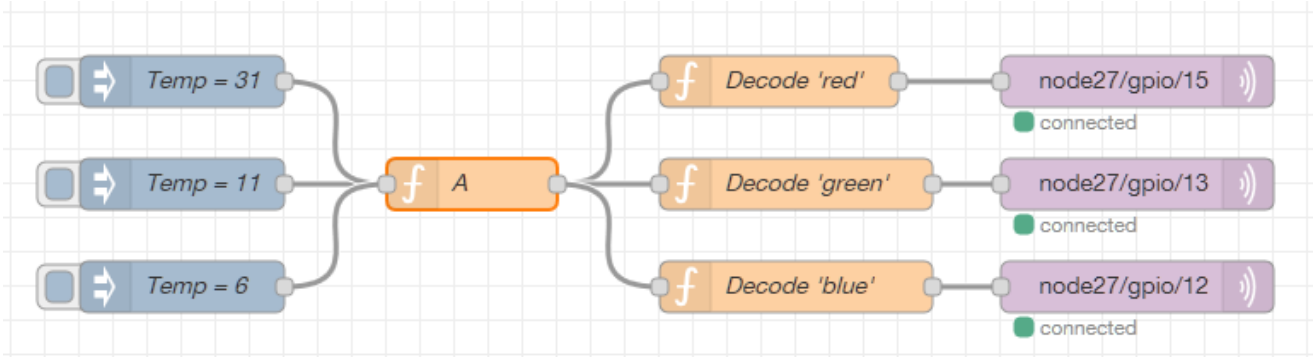


For the sake of this tutorial, the flow will simplified by replacing the OpenWeatherMap section with three inject nodes to simulate the temperature.

Bohunt School *(Wokingham)*

Controlling LEDs using a Temperature Value

Here's a screen-shot of the simplified flow.



Function node A is used to determine which band the temperature is in.

Name A

Function

```
1  if (msg.payload > 30) {
2      msg.payload = "red";
3  }
4  else if (msg.payload >= 10) {
5      msg.payload = "green";
6  }
7  else {
8      msg.payload = "blue";
9  }
10 return msg;
```

As can be seen from the JavaScript listing above this is achieved using a set of IF constructs. The resultant output is a text string: "red", "green" or "blue".

The output is connected to the three function blocks that decode the various colours and send a numeric value to the MQTT-Out blocks.

Name Decode 'red'

Function

```
1  if (msg.payload == "red") {
2      msg.payload = 1;
3  }
4  else {
5      msg.payload = 0;
6  }
7  return msg;
```

All three decode blocks are very similar the only difference is on line-1.

E.g. The Decode 'red' block has "red" at the end of line-1.

You can probably guess what appears at the end of the line in the other two blocks.

Each Decode block outputs a numeric 1 or a 0 to turn the appropriate LED on or off.

Controlling LEDs using a Temperature Value

By now you should be familiar as to how to set-up the MQTT-Out block.
If you need some help ask one of your colleagues or speak to Mr D.

If you have followed these instructions accurately you should find that when you click one of the 'inject' nodes a certain LED is illuminated. The 'inject' nodes are used to simulate a temperature value to test your system.

If you have time you could try removing the 'inject' nodes and connecting your system to the output from OpenWeatherMap - it should work in the same way.

Shortcoming

If you experiment with the system (i.e. click the various 'inject' nodes) you may find that for a short period of time two LEDs are illuminated. You may notice the system is a bit slow and takes a fraction of a second to switch over.

The reason for this is the Node-RED flow contains three MQTT-Out blocks that are used to send messages (one after the other) to the Wemos D1 Mini. This all takes time as each message is routed across the WiFi network a number of times. If you look carefully at your Wemos D1 Mini you should be able to see the small blue LED (the WiFi activity indicator) wink on and off three times.

Sometimes the message being sent is not really needed and is a complete waste of time. For example, a message might be sent to turn OFF the 'red' LED when it's actually already extinguished, is just silly. The same observation could be applied to the other coloured LEDs.

One way to overcome this shortcoming would be to 'remember' the state of each LED and only send a message if the state of an LED needed to change. This would reduce the amount of traffic going across the network.

Another method would be to send a code to the Wemos D1 Mini and get that device to do the decoding. This would mean only a single message would be sent rather than three consecutive messages - a considerable saving in time.

Sending a unique code to the Wemos D1 Mini

Up to this point you've probably only used the Wemos D1 Mini to turn LEDs on and off and maybe drive an OLED panel. Well it can actually do a number of other things like sensing an analog input voltage, controlling a servo motor or producing a pulse-width modulated output via the D pins.

The Wemos D1 Mini is really a small microcontroller and can be used to execute small programs inside itself. These programs are made-up of rules and are held in the 'Rules' section of the Wemos. One powerful feature of 'rules' is the ability to get the Wemos to 'listen' for an event to occur. Although it might sound complicated, it is very easy to set-up and get working.

Controlling LEDs using a Temperature Value

Using 'Rules' and sending 'events'

The first thing you need to do is decide on a name for the event that will be sent from Node-RED, via a MQTT-Out node, to the Wemos. This tutorial uses the event name… 'ledcolour_code' followed by a value or code to indicate which LED is to be illuminated.

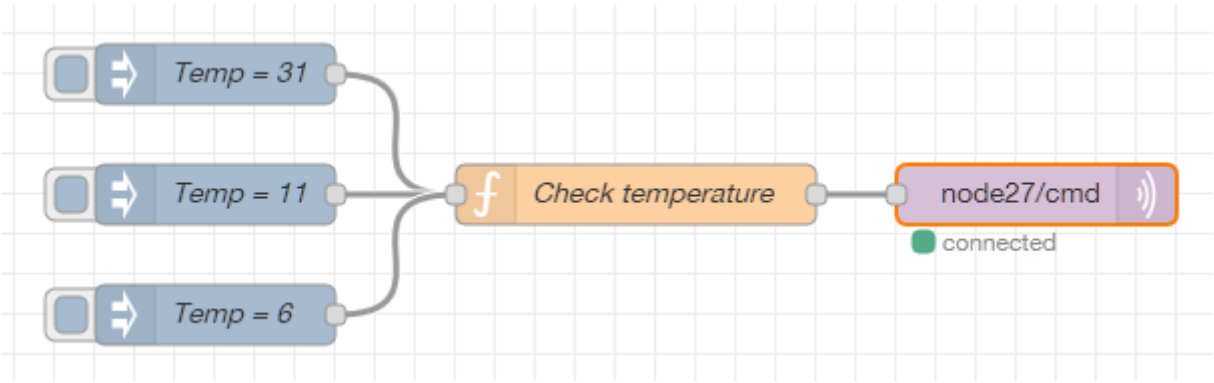
A value or code of '1' means the 'red' LED should be turned on.

A value or code of '2' means the 'green' LED should be turned on.

A value or code of '3' means the 'blue' LED should be turned on.

A value of '0' could be used to indicate ALL the LEDs should be turned off.

The MQTT-Out node needs to be set with the topic in the following flow.



As can be seen from the above flow, the 'Check temperature' function node performs all the work (i.e. performs all the magic). Here's the JavaScript.

Name: Check temperature

Function

```
1  if (msg.payload > 30) {
2      msg.payload = "event,ledcolour_code=1";
3  }
4  else if (msg.payload >= 10) {
5      msg.payload = "event,ledcolour_code=2";
6  }
7  else {
8      msg.payload = "event,ledcolour_code=3";
9  }
10 return msg;
```

If you look back to the JavaScript code for 'function' node 'A', on page-2, you should see that the above listing is very similar. The only differences are on line-2, line-5 and line-8 as they contain the msg.payload to call the event.

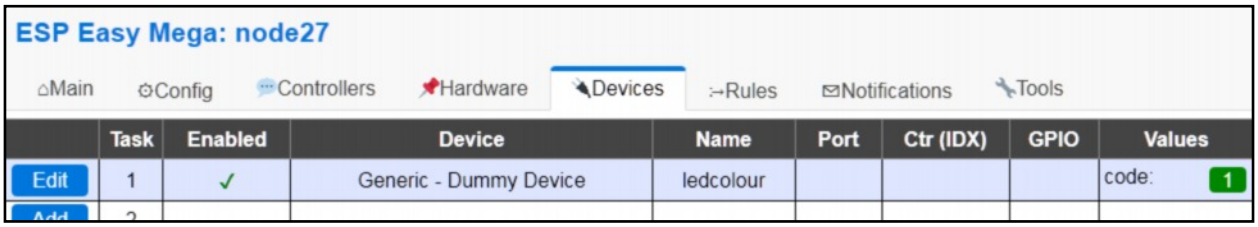
Bohunt School (Wokingham)

Controlling LEDs using a Temperature Value

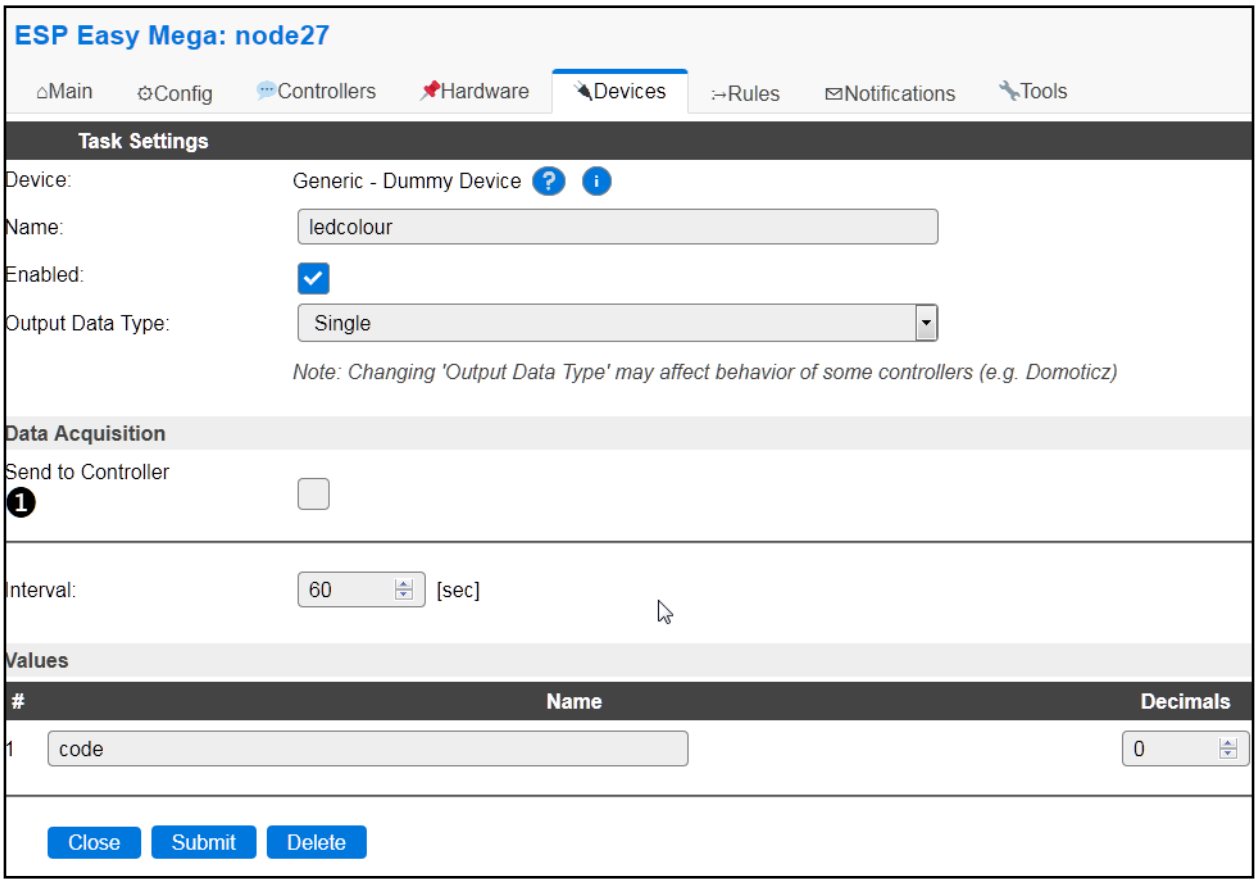
The next thing two things you need to do are inside the Wemos, so you need to login to the device by entering its IP address.

In this example node-27 is being used, so enter 192.168.1.27 in your browser.

Then select the 'Devices' tab as shown below.



Using the drop-down menu, scroll down to select 'Generic - Dummy Device' and fill it in as per this screen-shot.



This action creates a variable named… 'ledcolour#code' which will be used to hold the numeric value sent as part of the event string from Node-RED.

Next select the 'Rules' tab as shown below and select one of the Rule Sets.

Bohunt School *(Wokingham)*

Controlling LEDs using a Temperature Value

Carefully type the following into the Rule Set window.

```
On ledcolour_code Do
TaskValueSet,1,1,%eventvalue%          // Store %eventvalue% in dummy variable ledcolour#code

if [ledcolour#code]=0                  // Clear all LEDs
    gpio,15,0
    gpio,13,0
    gpio,12,0
else
    if [ledcolour#code]=1              // Turn on RED LED
        gpio,15,1
        gpio,13,0
        gpio,12,0
    else
        if [ledcolour#code]=2          // Turn on GREEN LED
            gpio,15,0
            gpio,13,1
            gpio,12,0
        else
            if [ledcolour#code]=3      // Turn on BLUE LED
                gpio,15,0
                gpio,13,0
                gpio,12,1
            endif
        endif
    endif
endif

endon
```

The first line is a command to 'listen' for the particular event 'ledcolour_code' to arrive at the Wemos via the MQTT communication channel.

The second line stores the numeric value (sent as part of the event string) in the Generic Dummy Device as a named variable (i.e. 'ledcolour#code').

The next few lines are a series of IF, THEN, ELSE statements that check for the numeric value… 1, 2, 3 or 0. As you can see from the JavaScript listing these tests are followed by the commands to control the GPIO pins on the Wemos.

As the Wemos is running with an onboard 40MHz clock the time it will take to execute the three GPIO commands is very short. So the overall effect is the LEDs will change rapidly - considerably faster than the method used on page-2.

If for some reason your system doesn't work - have a work with one of your colleagues or speak to Mr D.

Summary

This short exercise should have been "fun" for you and hopefully showed you how to adapt an existing flow (that you produced in an earlier exercise) to become something a bit more useful - using 'Rules' and 'Events'.

Node-RED is a brilliant tool for trying out your ideas quickly and easily.

Kind regards from Mr D.