IoT Workshop Session 1

By Tim Minter

Setting Up, Connecting and Controlling Basic Stuff with an ESP8266 Device

---

Prerequisites

Laptop with the following installed

- Arduino Programming Software (Arduino IDE)
  - https://www.arduino.cc/en/main/software

- ESP8266 Driver
  - Install the relevant driver for here https://wiki.wemos.cc/downloads

---

What is an ESP8266?

It is a basic tiny computer “Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.”
There are many different Arduino devices out there, all work in the same way and use the same programming techniques. The ESP8266 is one of smallest devices and comes with wifi making ideal for IoT projects.

Setting up the Software

Adding support for the ESP8266.

- Open the Arduino software and open the **Preferences** window. (From the menu in Mac, this is accessible via the **Arduino** menu then **Preferences**. For Windows this would be the **File** menu, then **Preferences**).
- In the **Additional Boards Manager URLs** add or enter “http://arduino.esp8266.com/stable/package_esp8266com_index.json” and click OK.
- Click **Tools** then **Board** then **Board Manager**.
- Enter esp8622 in the search box and then select **esp8266 by ESP Community** and **Install.**

In the Arduino software, we will now configure the connection to the ESP8266. We will select the correct “Board” to tell the software how to handle the ESP8622, the right speed to upload the code and the port the board is connected to.

- Connect the USB cable to the ESP8266 and the laptop and open the Arduino Program.
- Select **Tools** then **Board** from the menu and then select **LOLIN(WEMOS) D1 Mini Lite**. This is the actual name of the type of ESP8266 we are using.
- Select **Tools** then **Upload Speed** and then select **230400**
- Select **Tools** then **Port** then /dev/cu.wchusbserial1420 (for Mac), COM3 or similar (for Windows).

- *Note the brand name of this kind of board recently changed from Wemos to Lolin.*
- *Note there are other Lolin/Wemos ESP8266 devices that have different processing power, memory and connectivity options. We are using the basic “Lite” version which is fine for most applications.*

What the ESP8266 can do is controlled by what software libraries are available to you. Some libraries are installed by default.
To see what libraries are installed and add more, select **Sketch** from the menu and then **Include Library** and then **Manage Libraries** to add more. In Arduino world, a program is called a “Sketch”.

We will add a library that will allow us to control the LED strip. These LEDs are called “Neopixels”.

- Select **Sketch** and then **Include Library** then **Manage Libraries** from the menu.
- In the search box type “neopixel” then select **Adafruit Neopixel by Adafruit** and then **Install**.

Next we’ll install the library that allows us to communicate with IoT networks.

- Select **Sketch** and then **Include Library** then **Manage Libraries** from the menu.
- In the search box type “pubsubclient” then select **PubSubClient by Nick O’Leary** and then **Install**.

Next we'll install the library that allows us to handle json data.

- Select **Sketch** and then **Include Library** then **Manage Libraries** from the menu.
- In the search box type "ArduinoJson" then select **ArduinoJson by Benoit Blanchon** and then SELECT VERSION and choose 5.13.4 (not any of the beta versions) and **Install**.

Trying Out the ESP8266

We will now connect the LED strip, upload some code and show it controlling some basic lights.

- Connect the LED strip 5v, GND pins to the corresponding ESP8266 pins and the LED strip data pin to the D1 pin on the ESP8266.

- *Note the LED strip is a strip of 3 addressable LEDs which means you can send data down the data wire and control each LED on it's own.*

Next we will open one of the Example Sketches that comes with the Noepixel library and upload it to the ESP8266.

- Select **File** then **Examples** then **Adafruit NeoPixel** then **Simple**.
- The example code is displayed on the screen.
- On line 11 the pin that the LED strip is connected to is defined. Change this to "D1".
- On line 14 the number of LEDs we have connected is defined. Change this to 3.
- Underneath the "void setup()" line (line 23) there is some code we can safely delete to keep it simple. It is for a specific "board" and we are not using that board. Delete line 24-28.
- Check/Verify the code by clicking on the **Tick** icon at the top left of the Arduino software.
- Upload this code to the ESP8266 by clicking in the **Arrow** icon at the top left of the Arduino software.

Setting Up an IoT Network

We will be using the free version of the IBM Watson IoT Platform as our IoT network.
You will need an account in the IBM Cloud.

- Create an account at http://cloud.ibm.com
- Click on **Catalogue** then **Starter Kits** then **Internet of Things Platform Starter**. This creates an IoT network and an instance of NodeRED, which is open source software commonly used to "wire together" the internet of things.
- Give your Internet of Things Platform a name and accept the defaults for the rest of settings. You may be prompted to create an organisation and/or space in the cloud. Choose any names you want.

Next we will take a look at the IoT platform cloud interface and create what we need to connect our esp8266 device.

- Click on the left hand menu (three horizontal lines) in the IBM Cloud web page and select Dashboard to see the details of your account and services.
- Click on Cloud Foundry Services to see the IoT service you just created.
- Click on the IoT service you created and then click on the Launch button near the middle of the screen. This opens the interface that allows you to create and manage IoT devices.
- Click the Add Device button (top right of the page)
- When you first start with the service you will need to create a Device Type. For example "Door Bell" or "Thermostat" or, in our case we will choose to use "esp8266". In reality this is just a label used for grouping your devices if you have lots of different devices.
- Next enter a name for your specific device eg "IoTWorkshopDevice" (in the real world this would be created by a sign up process using the IoT platform APIs etc).
- Click Next and you can leave all fields blank.
- Click Next and on this Security tab you will need to enter a token (password) that your device will use to connect to the IoT network. Leaving it blank creates a random password. To keep this workshop simple please enter the password supplied to you in the workshop. Click Done.
- A summary is shown and it's worth looking over the various info sections shown under **Device Drilldown.**

- Click on the **Gear** icon on the left hand side of the page (second for bottom) and then **Connection Security**.
- There is a dropdown option lots near the top of the page. Select **TLS Optional**. This effects the security of the connection form your device to the IoT platform and TLS Optional is the least secure but fine for this demo. Using different security is out of scope of this workshop.
- Note down the IoT Network **Organisation ID** shown at the top right of the page eg “zgf4hg”

Connecting to Wifi and an IoT network.

We now have everything we need to connect your esp8622 to your instance of the Internet of Things.

We will now load a new Sketch onto your esp8266 and set it up to connect your IoT platform.

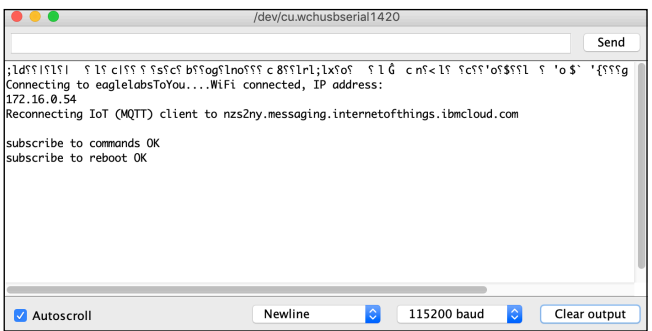
Copy the code from the IoTWorkshop1ArudinoCode file located here…
https://github.com/tim-minter/IoTWorkshop

In the Aruino software Click File/New and replace everything in the file with what you just copied (remove any code that already in the new file).

Save the file and then locate this line….
//-------- Customise these values ———————

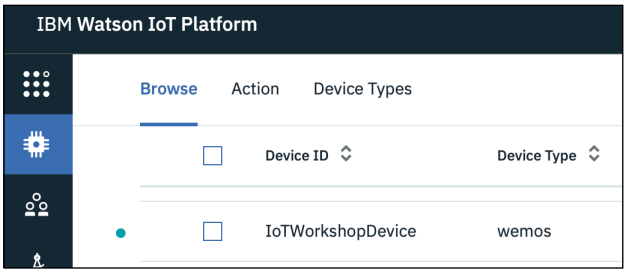
As you can see, this is where you enter the wifi and IoT details you noted down previously.
Enter these values and save again then click the Tick at the top left of the window to check the code. If it all checks to OK and there are no errors click the Arrow button at the top left of the window to upload this new code to your esp8266.

When done, the LED strip should turn red. To check what is going on click **Tools** then **Serial Monitor** in the arduino software. You should see something like this if connection was successful …



Errors will be shown here if there was a problem.

If we go back to the Watson IoT platform in IBM Cloud (https://internetofthings.ibmcloud.com - *you may have to select your org name form the top*



right of the page) you will see the device is connected (a green circle will be shown).

You can click on the device to see more info and logs and also perform a remote reboot.

Clicking this icon on the right hand side of the device info line shows much more info and the ability to perform the reboot.

Controlling the device

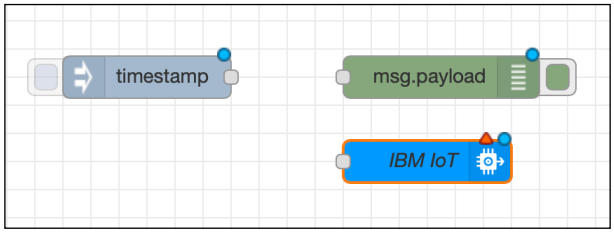
We will be using NodeRED, which is open source software for “wiring together the internet of things”.
We created an instance of this earlier.
In IBM Cloud go to your **Dashboard** and click on **Cloud Foundry Apps**. Click on the **NodeRED** instance you created earlier. This shows information about the application in the cloud. Click on the Visit App URL link at the top of the page.

You will be guided through some set up. Complete that and you will be presented with the NodeRED interface.
We wont describe NodeRED in detail here but will go over it in the workshop.

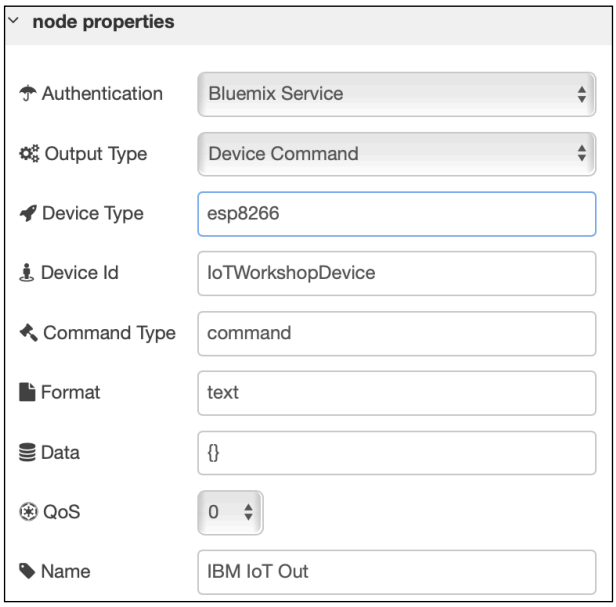
NodeRED consists of a pallet of “nodes” and the ability to connect them together visually.

We will be using three nodes initially. Drag the **ibmiot** Output node to the middle section (the

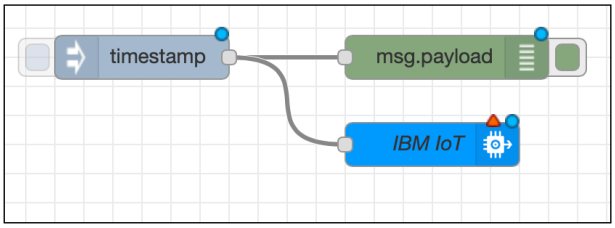
workspace) of nodeRED. Do the same for a **debug** node and and **inject** node.



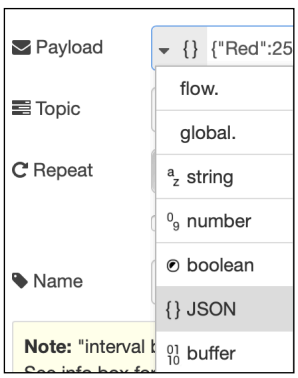
Next we will configure the IBM IoT node. Double click the node and set it up as follows… (replacing Device Type and Device ID with your values).



Connect the nodes by dragging the connectors so they look like this…



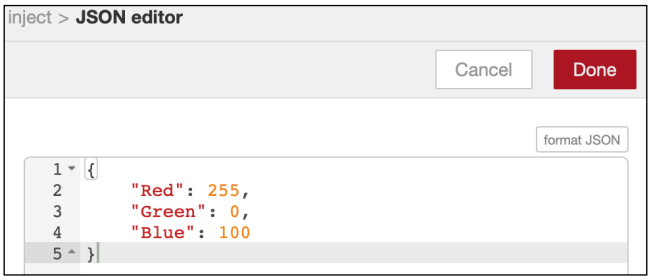
Then double click the inject “timestamp” node and click the Payload drop down. Select JSON as the type and then click the three dots on the right hand side of the drop down…
Then enter this



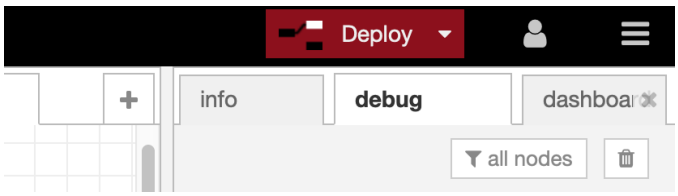
text…

```
{
   "Red": 255,
   "Green": 0,
   "Blue": 100
}
```

So it looks like this…



Click the deploy button at the top right to save/ deploy this new “flow” to the cloud.

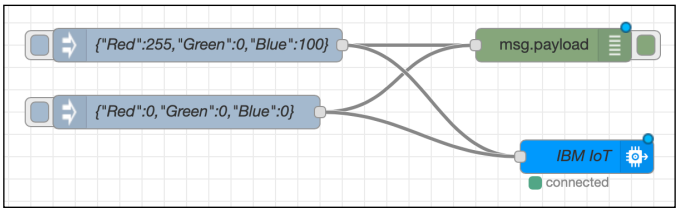


We can now test the whole system!

Watch the LED strip connected to the esp8266 and click the tab on the left side of the inject node. this sends the data you entered to the IoT node (and the debug node).
The LED strip should change colour!

Now add another inject node and enter this text into it…

```
{
   "Red": 0,
   "Green": 0,
   "Blue": 0
}
```



Connect it as shown below.

You should now be able to turn the LEDS off!