Setting up the Node

CONTENTS

1. Programming the NodeMCU/ESP8266

When first programming your ESP8266, uncomment the directives:

```
//#define _ForceRocnetNodeID_to_subIPL
//#define _ForceDefaultEEPROM
//#define _Force_Loco_Addr 8
```

Then, monitor the arduino serial monitor. You should see the code looking for your Router SSID, logging in, then trying to connect to your MQTT broker. It will sequentially try IP addresses on your router between 0 and 50 (set in MQTT:reconnect().). It should connect and then throw up a table showing how the IO is organised. At this point, you know the code is working so note the IP address of the hardware and comment out the three directives above. RE-program the device and again watch as it starts up. It should connect to the router with the same IP address as before and will again look for the MQTT broker. But this time it will save these values un EEPROM so that subsequent start ups will be quicker.

You can now modify the IO settings using Rocrail.

If you set the hardware as a loco, the Arduino Serial monitor should show something like this:

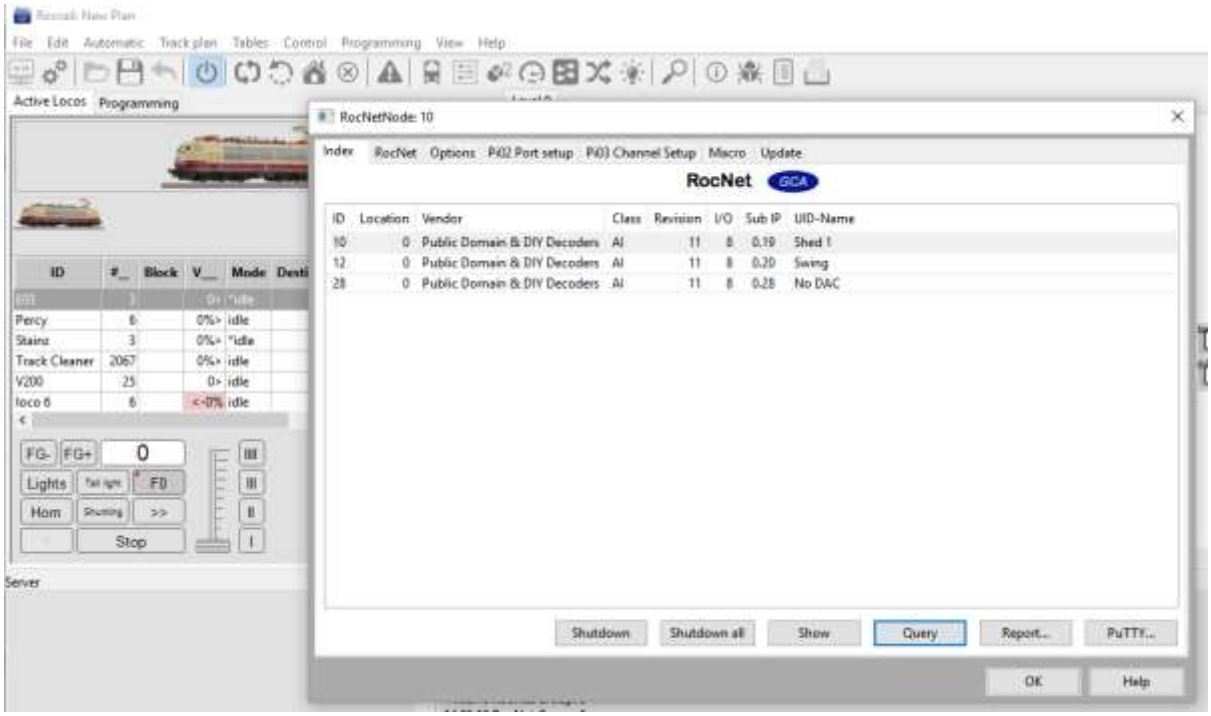
```
------------------------------------------------------------
          ESP8266 MQTT Rocnet Node with Sound
------------------------------------------------------------
               revision:11
------------------------------------------------------------
Trying to connect to {linksys-25}
----------------------------Connected-----------------------
 Connected to SSID:linksys-25  IP:192.168.0.28
 Mosquitto will first try to connect to:192.168.0.18
------------------------- RocNet Node ---------------------
              My ROCNET NODE ID:28
---------------------- LOCO Setup   ----------------------
        Short 'Locomotive Address' is7
-------------------------- PORT Setup ----------------------
--- Setting OTA Hostname <RN LOCO(No DAC)> -------------
------------------------------------------------------------
 Port :0 is SignalLED  Output (FIXED) NodePortType :0  Pi03_Setting_options0
 Port :1 used as LOCO MOTOR (FIXED) NodePortType :0  Pi03_Setting_options42
 Port :2 is BACKLight  Output (FIXED) NodePortType :0  Pi03_Setting_options0
 Port :3 is FRONTLight  Output (FIXED) NodePortType :0  Pi03_Setting_options0
 Port :4 is Output NodePortType :0  Pi03_Setting_options0
 Port :5 is Output NodePortType :0  Pi03_Setting_options0
 Port :6 is Input with pullup NodePortType :1  Pi03_Setting_options0
 Port :7 is SteamPulse  Output (FIXED) NodePortType :0  Pi03_Setting_options0
 Port :8 is Servo NodePortType :0  Pi03_Setting_options32
 ----------Entering Main Loop----------
@<00:00:00s>:Start-- Sound System Initiating --
Playing </initiated.wav> @Gain:1.00
@<00:00:00s>: Attempting MQTT connection attempt #1 trying:192.168.0.18
 *Debug Message: Time not synchronised yet  Node:28 Loco:7(No DAC) Msg<RocNetESPNode:28 Connected at:192.168.0.28>
..

 *Debug Message:<14:19:00s>  Node:28 Loco:7(No DAC) Msg< IPaddr .28  Time Synchronised >
```

This shows a decoder that is set for a Rocrail “Node” of 28, and is also a Loco at DCC address 7.

The code tries to show if any ports are “FIXED” and being used for special purposes, such as the Loco Motor, or front and back lights etc.

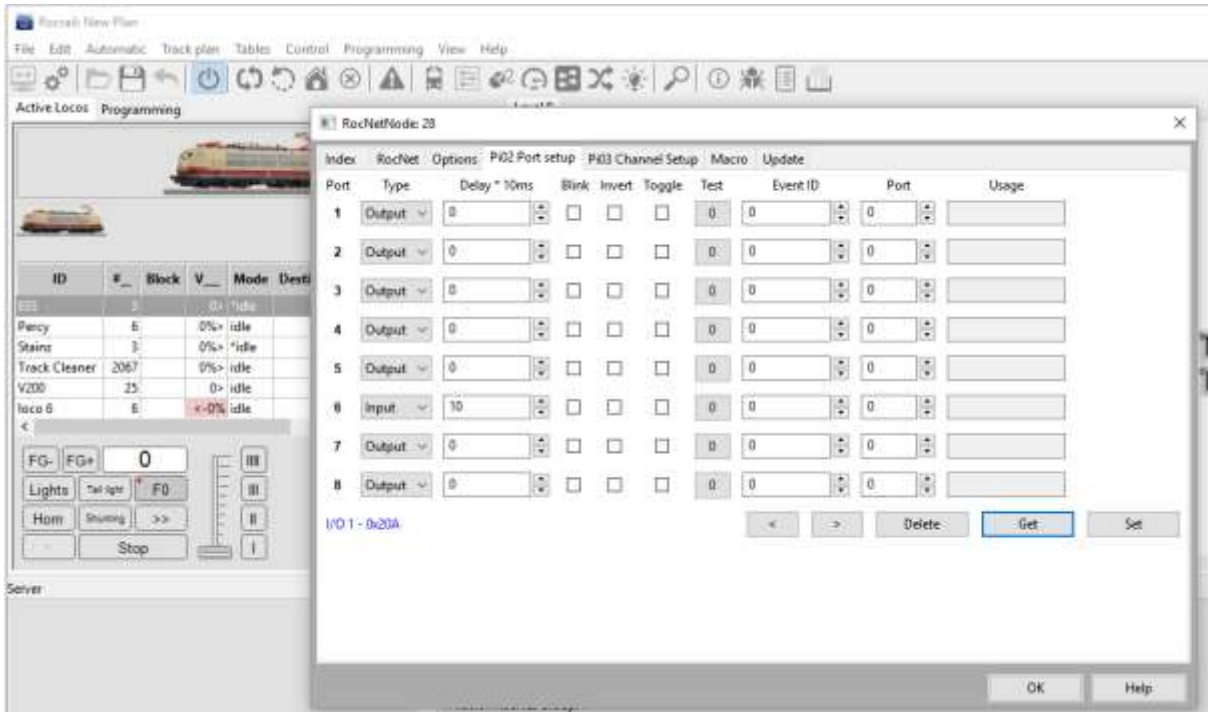
Here I have set pins 6 as input and 8 as servo. If this was a Stationary node, the display would be similar, but all ports would be usable.

Going to Rocrail and pressing Programming / Rocnet / Setup and then “Query” will bring up a page that shows all the nodes connected to the MQTT broker. (Note this does not update actively so may show nodes that are currently off or not powered up).



The decoder we were looking at via the Arduino Serial monitor is ID 28, and has the nickname “NoDAC”. The Roc Node address was set initially to the last byte of the IP address because of the _ForceRocnetNodeID_to_subIPL directive. This prevents nodes being on the same “node” address. Once the unit has the directive disables, you can set the Node Address via the Rocrail “Rocnet” ID interface, as can the node “nickname or “UID-Name” as Rocrail calls it. I would recommend leaving the Node ID as the IP address as it may help debugging later..

If we now highlight this node, and Click “Pi02 Port Setup”, it will show us how the various ports are set up:



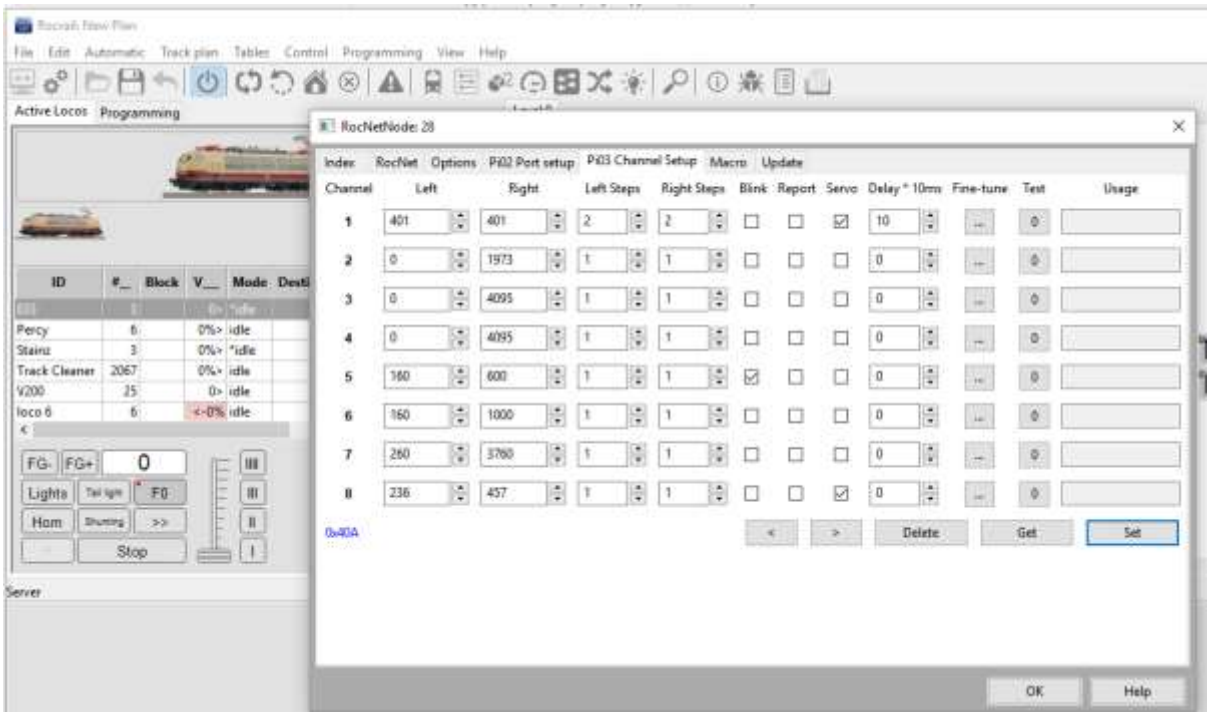
Port 6 is an input, but all others are outputs.

If any of the Pi02 ports were set with “Blink”, they will blink/flash with a delay set by the Delay setting.

“Invert” would invert the port phase.

Note Rocrail does not know that some ports are “FIXED”, but if you try to modify them they will revert to their “FIXED” settngs.

Pressing “Pi03 Channel Setup” shows further information about our 8 io pins:

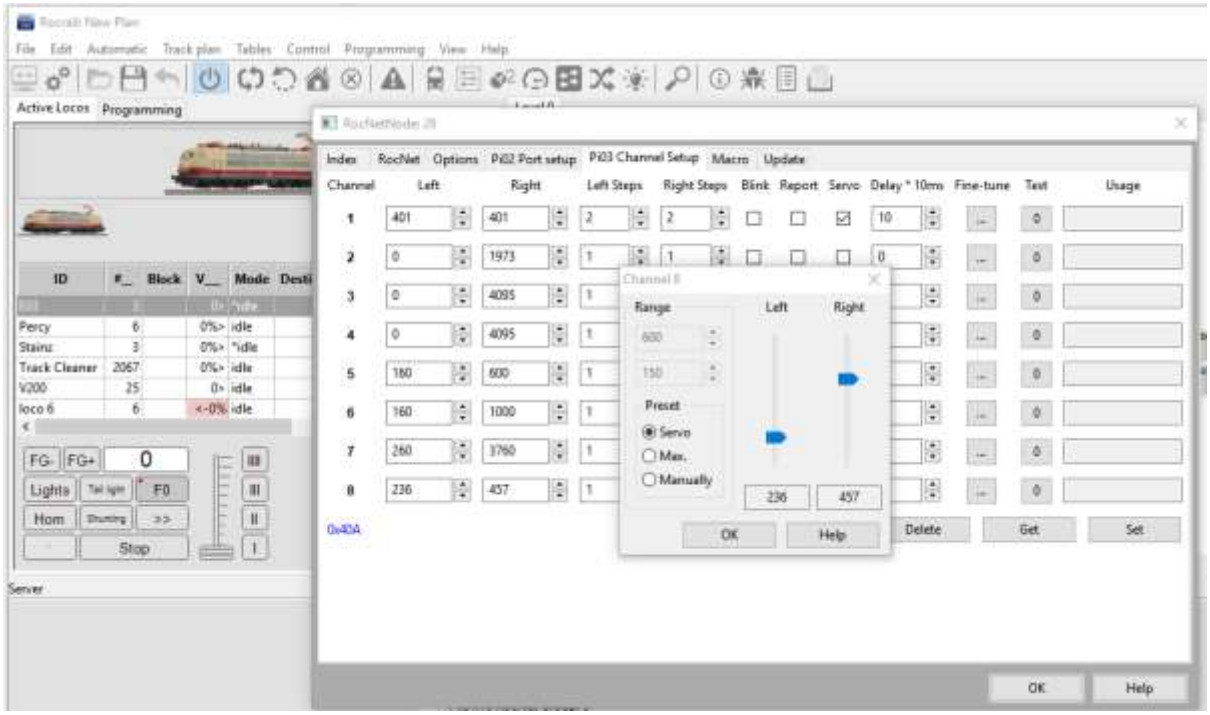


Here we see that Pin 1 and 8 are defined as a servos.. Pin 1 is actually the “FIXED” Motor drive servo, and is a special case where Left Right, Left steps Right steps etc are defined by CV values so these values cannot be changed here.

Pin 5 has Blink set, so will be a PWM output, switching between the Left and Right PWM settings of 160 and 600.

Pin 8 is a “normal” Servo output and will switch between positions 236 and 457. Note that in Rocrail, Min and Max positions for a servo are 150 and 600.

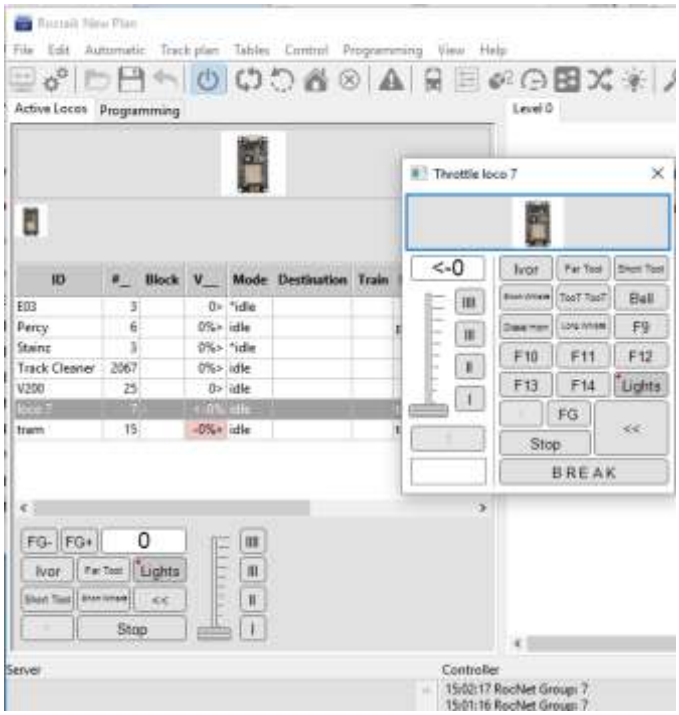
If you press the” …” box marked “Fine-Tune”, a very useful slider interface appears:



This is especially useful for adjusting endstops on points. My code should move the servo to the new position once the slider is moved.

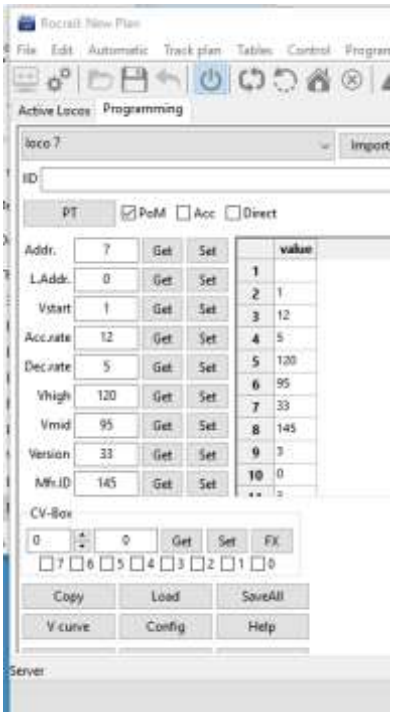
Because this is a “loco” at address 7 we can look at the throttle:

I have set this one in the Rocrail “Locomotives table” with the default sound effect names.



Clicking on “programming” will open the CV programming interface.

To make this work, you need to select “POM” and the loco to be programmed in the top left box, and Rocrail will interrogate the set loco address and get the main CV settings:



You can now “set” and “get” these CV settings.

The key ones are the Vstart and VMid, which for my code set the initial servo positions in forwards and reverse. Here the “reverse setting “Vmid” is set at 95 and needs to be changed.

Some Special CV’s:

CV [100] is the Overall Sound volume:

CV [101] is the volume for sound “F1”

CV [102] is the volume for sound “F2” (etc)

CV[110] is the volume for chuffs

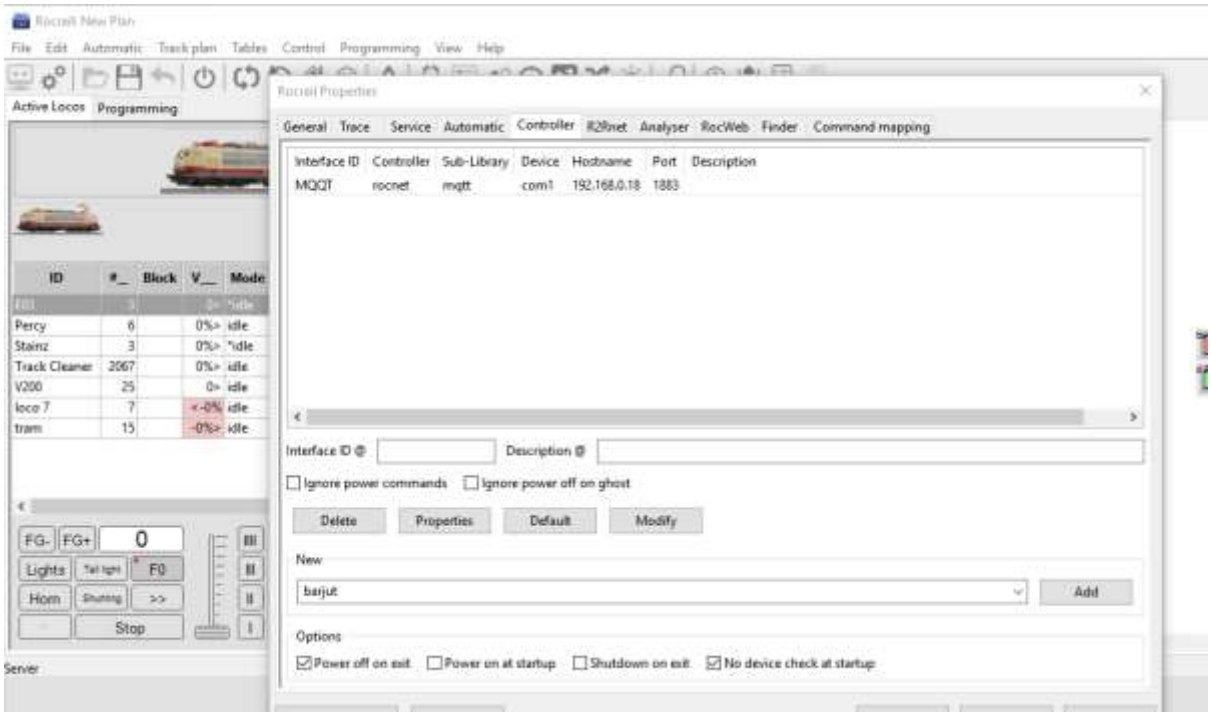
CV[111] is the volume for the brake squeal

.

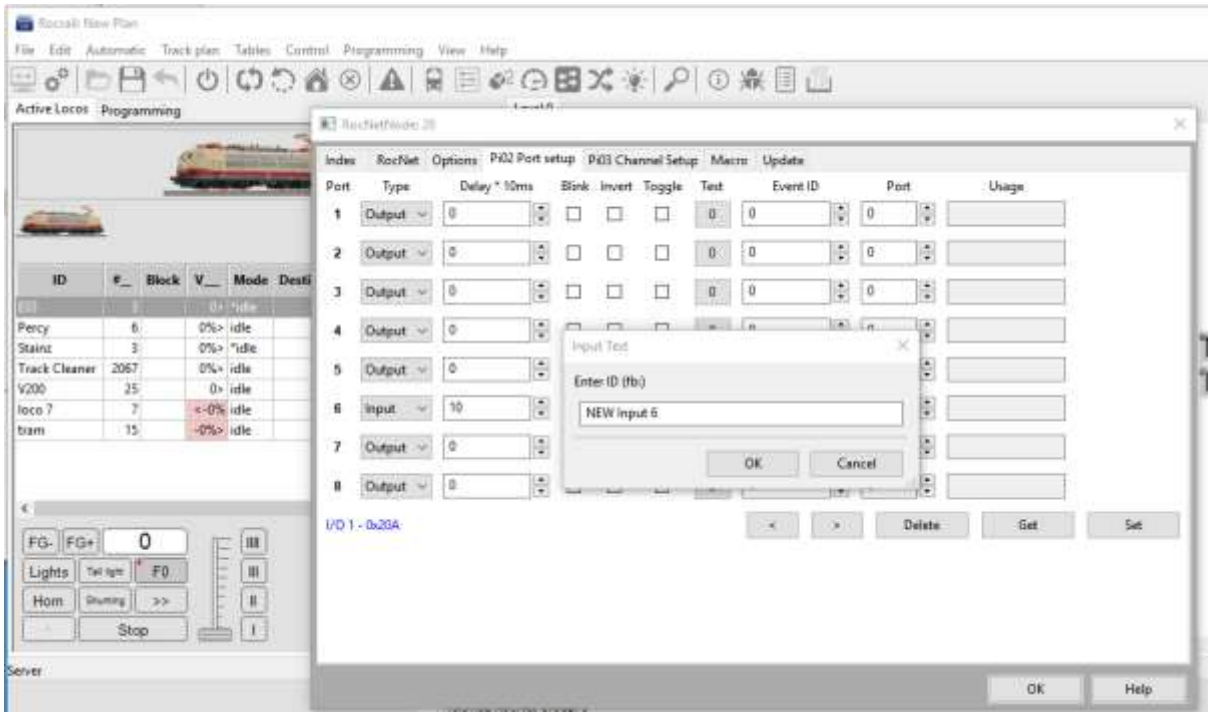
2. How Rocrail addresses the ports in its interface:

The terminology Rocrail uses when you set an interface can be confusing: Here I will set up one sensor and one servo using the hardware as used in the previous examples.

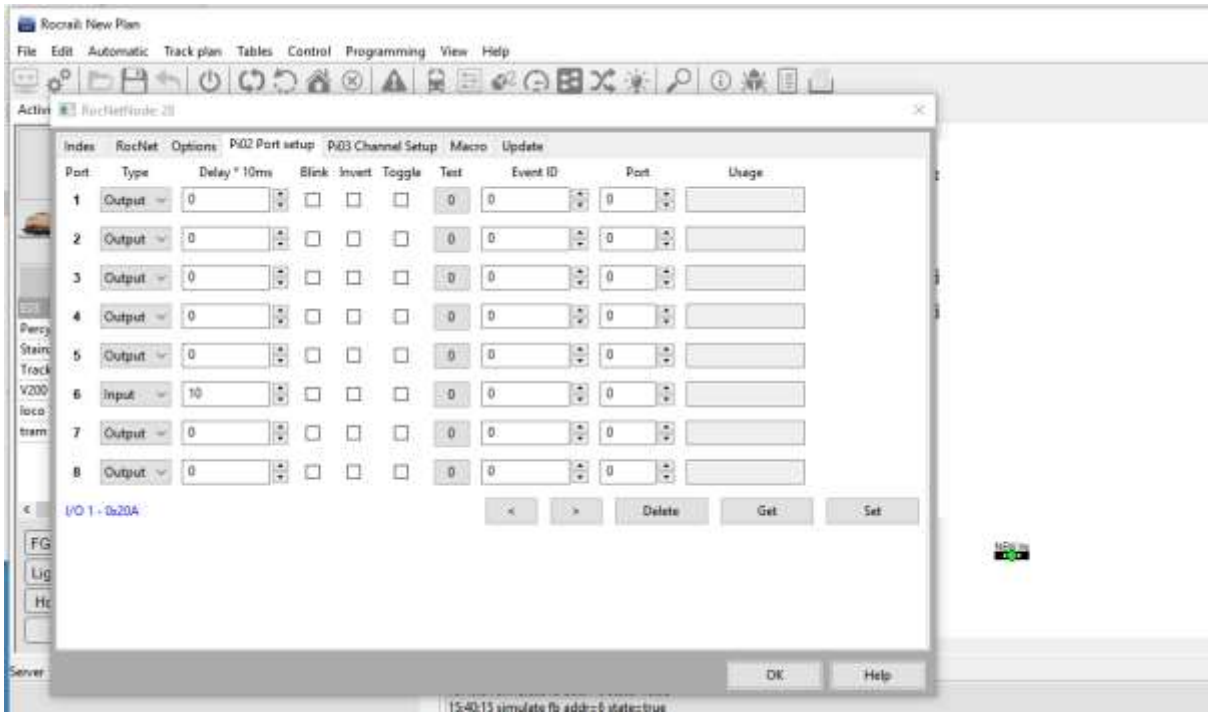
I have set my “controller” (File/Rocrail properties/ controller) as “MQQT” and set the correct MQQT broker address for me of 192.168.0.18, and port at 1883.



Now going to the Pi02 setup noted earlier, I click on the number “6” on the left and drag until it appears on my plan. On releasing, a box is opened asking for an ID:

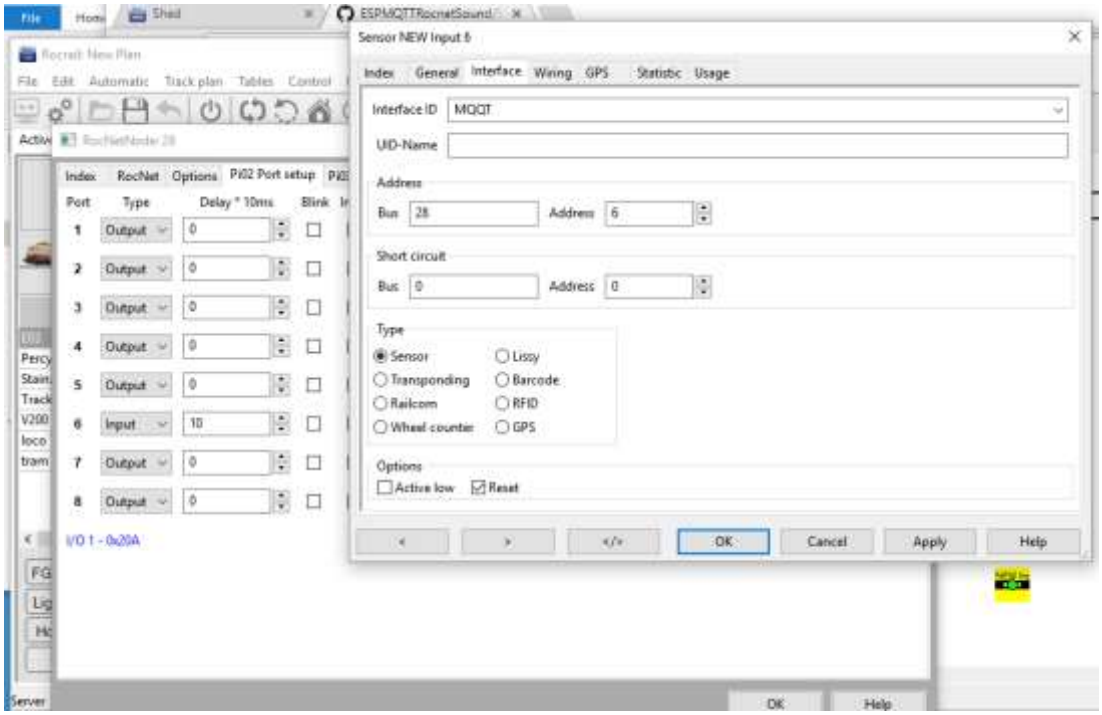


Clicking OK leaves me with a new sensor on the track plan :



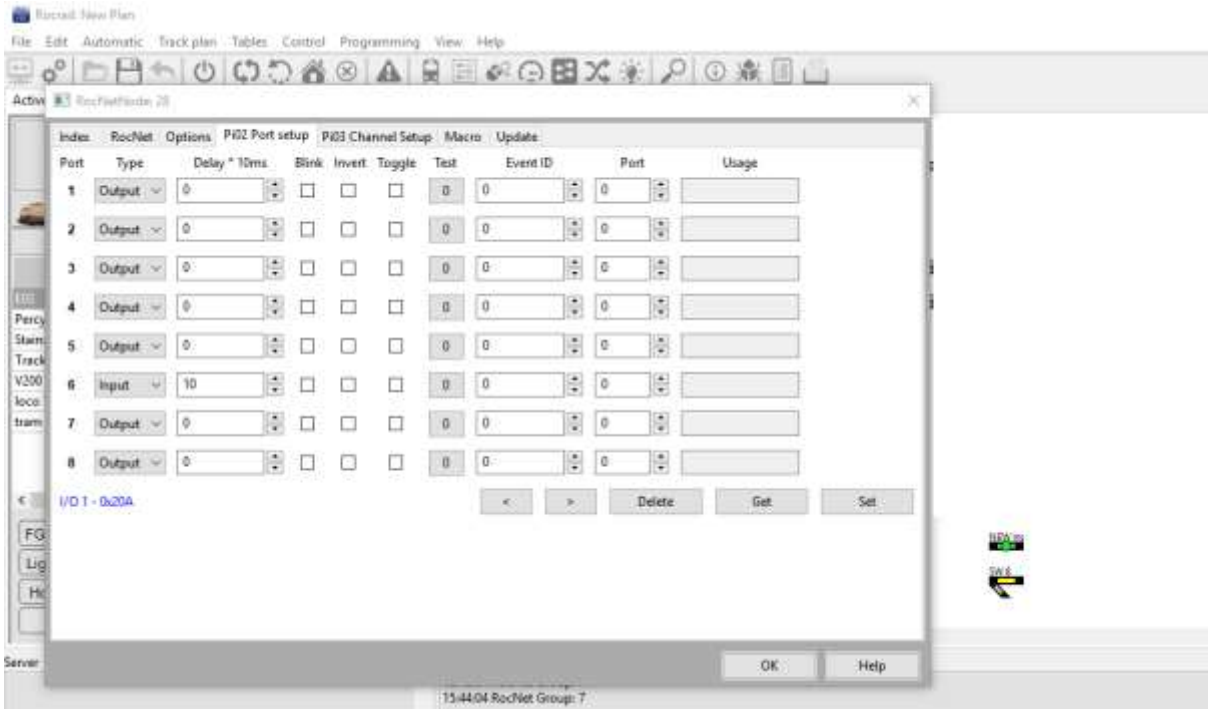
The name was too long for all of it to show…

Looking at the interface properties for this sensor you can see it is set to : “BUS” 28, which is the “Node” address of the hardware, and to “Address” 6 which is “port” 6 or PIN D6 on the NodeMCU.

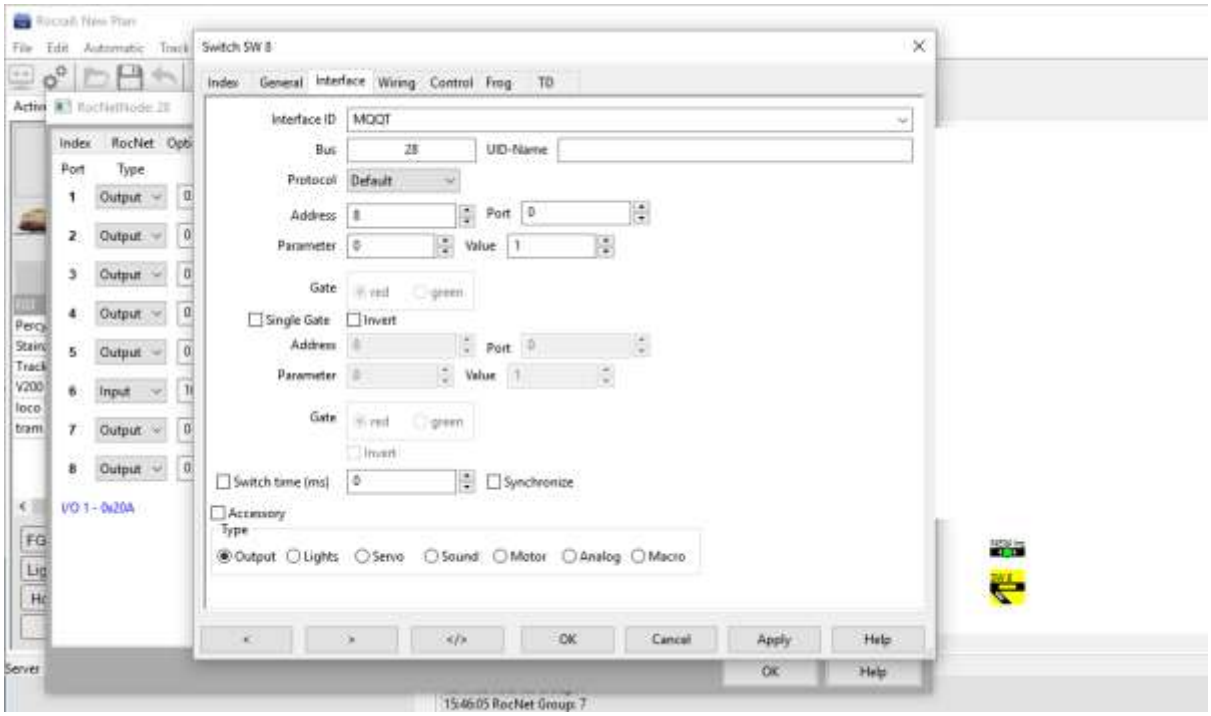


Clicking on the “8” on the Pi02 Setup page and dragging to the track plan will add a Point.

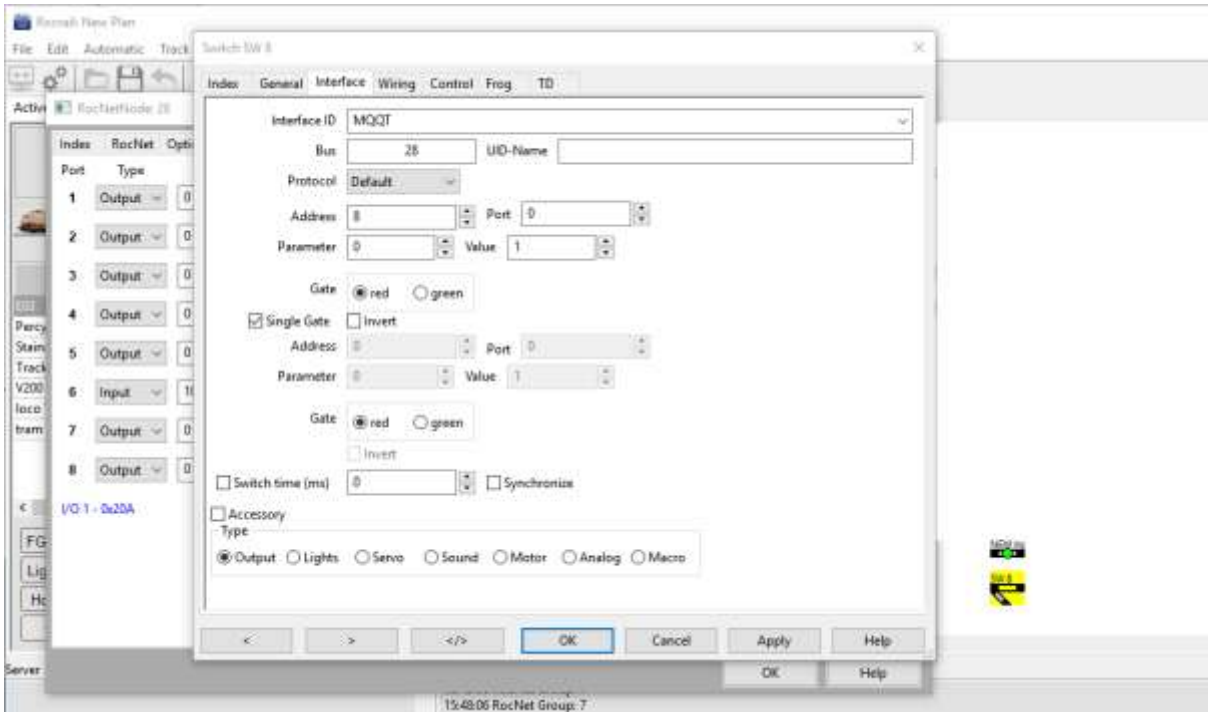
I named this one “SW 8” .



Clicking on the interface properties for SW 8 and you will see the default is as follows:



You MUST now click on “Single Gate” (and “Apply”) to enable Rocrail to control the servo.



Clicking on the point on the track plan will now activate the servo. This can be checked using the debug monitor (explained later) which will show

```
<15:50:41s>  Node:28 Loco:7(No DAC) Msg<Setting Output 8 (SERVO) to State(0) = Position:34 >
<15:50:42s>  Node:28 Loco:7(No DAC) Msg<Setting Output 8 (SERVO) to State(1) = Position:122 >
```

: Note that 34 and 122 are the “degrees” conversion from the Rocrail Left and Right settings of 236 and 457.

3. Wireless Debugging

I have allowed the code to send quite a lot of “debug” messages to the MQTT broker as a service “DebugMsg"

By opening a Command Prompt window on the PC running the MQTT debug server and entering: mosquitto_sub -h 127.0.0.1 -i "CMD_Prompt" -t debug -q 0,

You will open a useful debug service hat gives and indication of what’s going on:

If you want to do this on a device that is not running the MQTT broker, and where the MQTT broker is running on 192.168.0.18 run:

mosquitto_sub -h 192.168.0.18 -i "CMD_Prompt" -t debug -q 0

```
C:\Users>cd..
C:\>cd mosquitto
C:\mosquitto>mosquitto_sub -h 192.168.0.18 -i "CMD_Prompt" -t debug -q 0
<15:18:00s>  Node:28 Loco:7(No DAC) Msg< IPaddr .28  Time Synchronised >
<15:18:27s>  Node:2 Loco:3(3 DAC) Msg<SFX-F changed <8>h <0>h <0>h>
<15:18:27s>  Node:2 Loco:3(3 DAC) Msg<sfx-F4>
<15:18:28s>  Node:2 Loco:3(3 DAC) Msg<SFX-F changed <0>h <0>h <0>h>
<15:18:31s>  Node:2 Loco:3(3 DAC) Msg<Speed set to:48 Dir:1 Lights:1 Servo:111>
<15:18:34s>  Node:2 Loco:3(3 DAC) Msg<Speed set to:0 Dir:0 Lights:1 Servo:90>
<15:19:00s>  Node:2 Loco:3(3 DAC) Msg< IPaddr .16  Time Synchronised >
<15:19:00s>  Node:12 (Swing) Msg: IPaddr .20  Time Synchronised
<15:19:00s>  Node:10 (Shed 1) Msg: IPaddr .19  Time Synchronised
<15:19:00s>  Node:28 Loco:7(No DAC) Msg< IPaddr .28  Time Synchronised >
<15:20:00s>  Node:2 Loco:3(3 DAC) Msg< IPaddr .16  Time Synchronised >
```

This example shows what happened when I pressed “F4” and then changed the loco speed up and down.

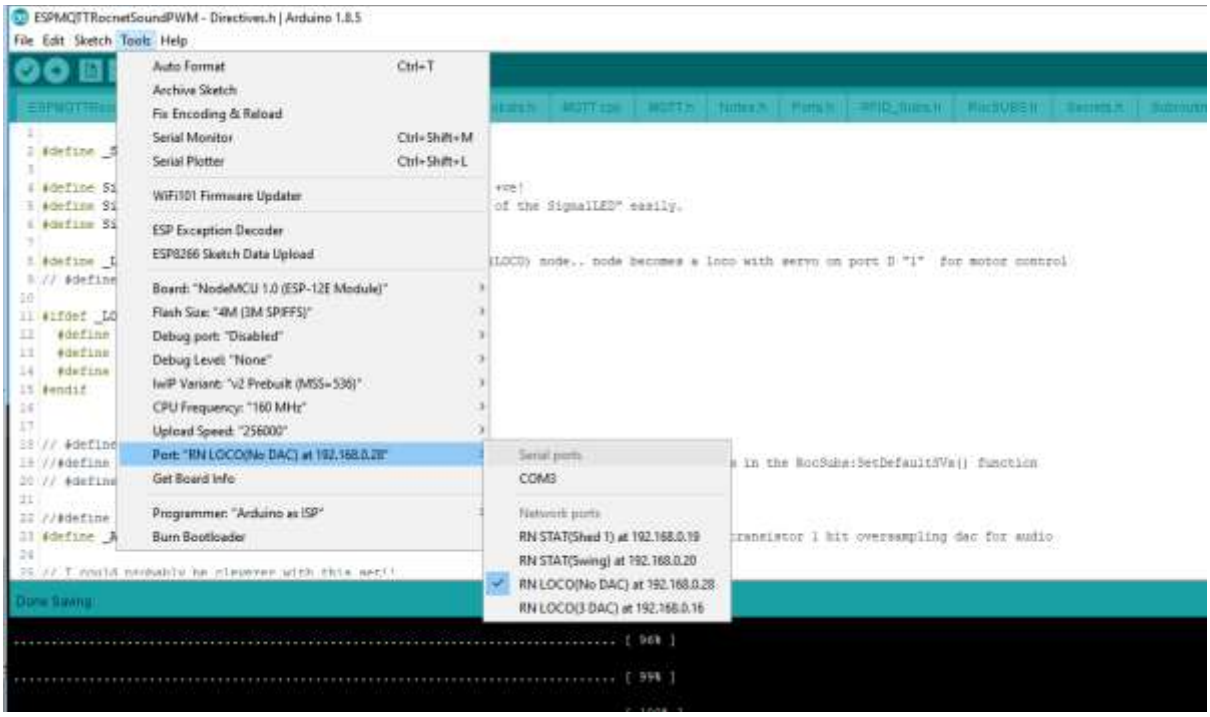
This interface is helpful when debugging the code, or just to check that nodes are connected, as they all report synchronisation with the Rocrail clock every minute.

Here I had two “station” nodes “Swing and Shed 1” connected as well as two Locos (“3” and “7”).

4. Wireless Updating

The code includes the ability to wirelessly update the code using the Arduino “OTA” programming.

To use this you must select the required hardware from the “ports” setup in Arduino. This displays the ports by their IP address and is one reason it is a good idea to keep the Rocnet Node ID the same as the IP address.



I have made the hardware report the Rocnode Nickname as part of the response to help identify the nodes. – but the nickname can only be 6 characters in my code.

Once you are ready, simply press the upload button.