



How to use the PFC-200 serial port with Node-RED



Question: Why do I get a permission denied error when I try to use the onboard serial port in a Node-RED container?

In order to use the onboard serial port of the PFC200 in Node-RED some configuration changes must be made, and the container must have the device attached to it.

This procedure will explain the steps to get the serial port to work in Node-RED.

```
root@PFC200V3-46E709:/dev docker run --rm -t --network=host -v node_red_user_data:/data --device=/dev/tty01:/dev/tty01:rw nodered/node-red
```

```
11 Dec 12:11:56 - [info] Started flows  
11 Dec 12:11:56 - [error] serial port /dev/tty01 error: Error: Error: Permission denied, cannot open /dev/tty01
```

The screenshot shows the Node-RED web interface. On the left, a 'network' palette contains 'serial in', 'serial out', and 'serial request' nodes. The main workspace shows a flow with a '/dev/ttyO5' node connected to a 'msg.payload' node, and another flow with an 'abc' node connected to a '/dev/ttyO5' node. The right sidebar is open to the 'Edit serial out node > Edit serial-port node' configuration panel. The 'Serial Port' field is set to '/dev/ttyO5'. The 'Settings' field is set to a list containing '/dev/ttyO0', '/dev/ttyO1', and '/dev/ttyO5'. The 'DTR', 'RTS', 'CTS', and 'DSR' fields are all set to 'auto'. The 'Input' section is partially visible. On the far right, a 'debug' console shows three error messages:

```
12/4/2020, 4:08:22 PM  
msg : string[85]  
"serial port /dev/tty00 error:  
Error: Error: Permission denied,  
cannot open /dev/tty00"  
12/4/2020, 4:08:37 PM  
msg : string[85]  
"serial port /dev/tty01 error:  
Error: Error: Permission denied,  
cannot open /dev/tty01"  
12/4/2020, 4:08:47 PM  
msg : string[85]  
"serial port /dev/tty05 error:  
Error: Error: Permission denied,  
cannot open /dev/tty05"
```

Permission denied error in Node-RED



Question: How can I identify the onboard serial port /dev/ name?

The linux command `ls /dev -l` shows all the serial ports found on the controller (partial list shown below for clarity).

```
root@PFC200V3-46E709:/dev ls -l
c----- 1 root root 10, 235 Dec 11 02:48 autofs
lrwxrwxrwx 1 root root 10 Dec 11 02:48 serial -> /dev/ttyO1
lrwxrwxrwx 1 root root 10 Dec 11 02:48 service -> /dev/ttyO0
lrwxrwxrwx 1 root root 10 Dec 11 02:48 ttyKbus -> /dev/ttyO5
crw-rw---- 1 root dialout 247, 0 Dec 11 02:48 ttyO0
crw-rw---- 1 root dialout 246, 1 Dec 11 02:48 ttyO1
crw-rw---- 1 root dialout 247, 5 Dec 11 02:48 ttyO5
```

} Soft Links (blue) are used help you easily identifiable the tty serial ports.

The on-board serial port of the PFC200 is /dev/ttyO1.



Note: This is O as in Omega, not the number 0.



KBUS
/dev/ttyO5
Do not use

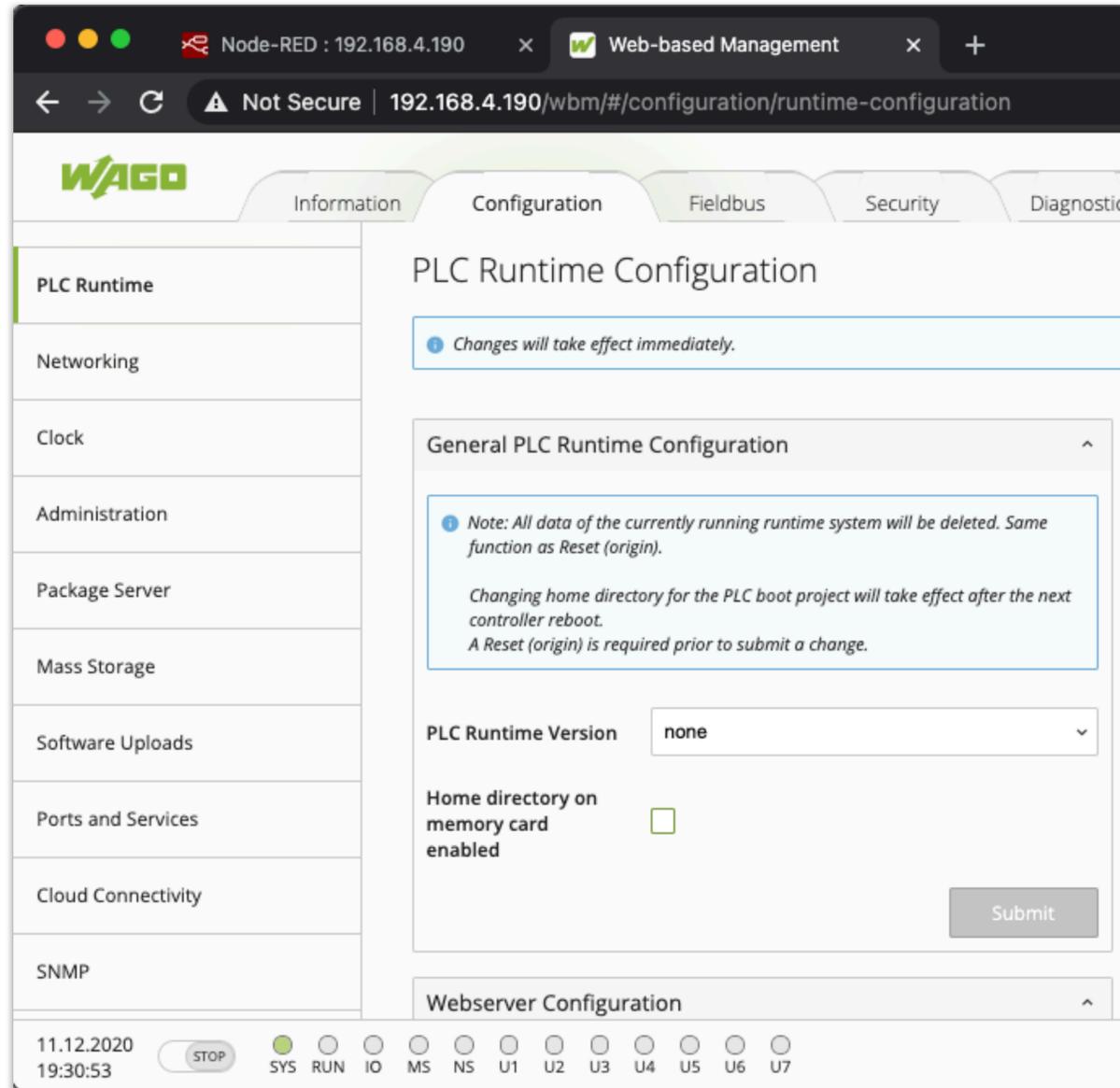
Service
/dev/ttyO0
Do not use

I/O Modules

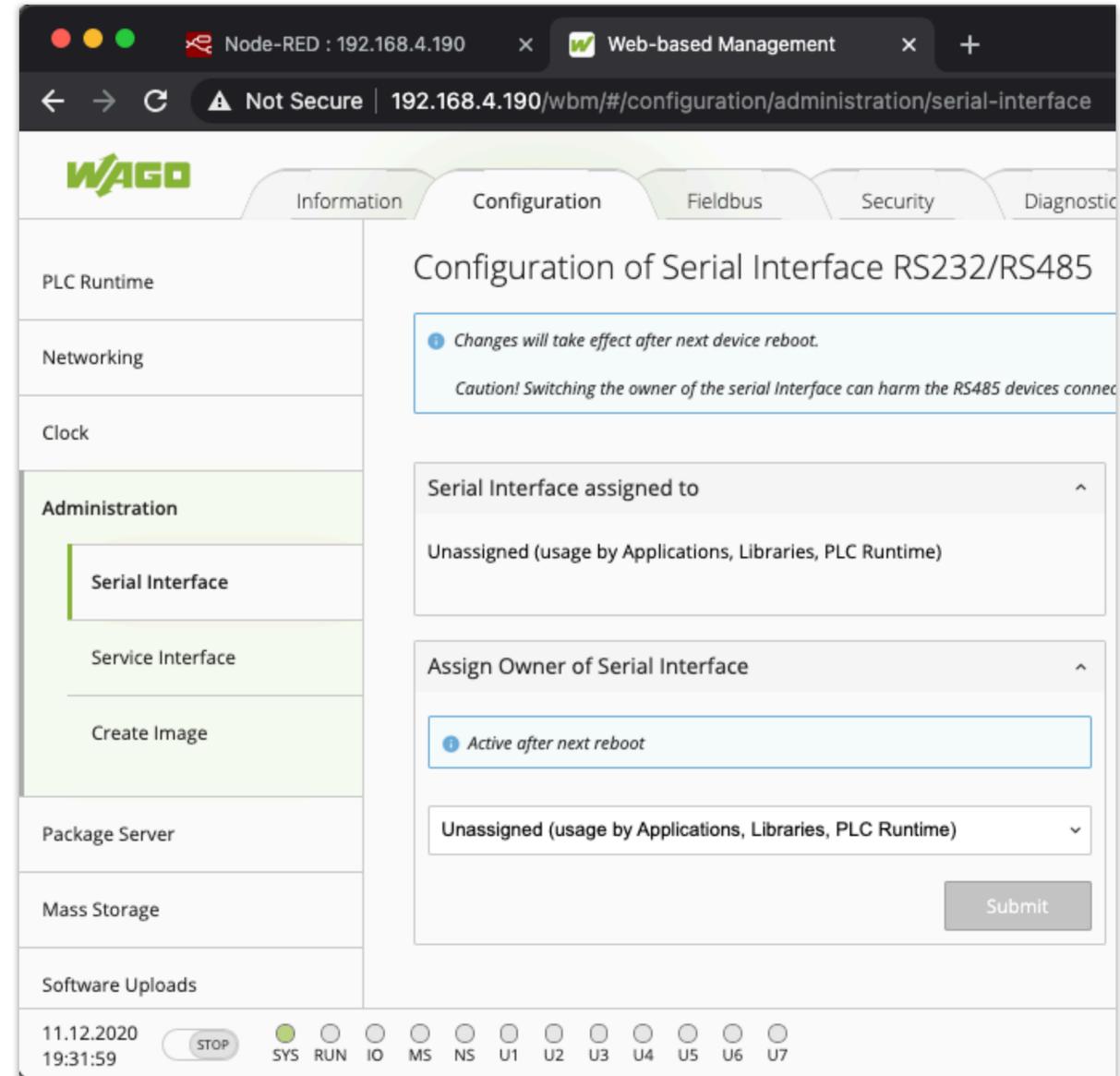


Step 1:

Set the runtime to none in the Web Based Management.



Set Serial Interface to "Unassigned"



Step 2:

The serial port device has insufficient permissions to use with Docker. This can be checked by looking at the results of `ls -l` command.

```
root@PFC200V3-46E709:/dev ls -l
crw-rw----- 1 root dialout 246, 1 Dec 11 02:48 tty01
```

We need the `tty01` to have read/write permission for **Other**, however it does not by default. Only owner (`root`) and group (`dialout`) has r/w access. *Other in this case is the Docker Node-RED container.*

The command `chmod` allows permissions to be changed. A 6 gives rw-privileges to the device, so 666 will give owner, group and other the r/w access we need.

```
root@PFC200V3-46E709:/dev chmod 666 /dev/tty01
```

Another `ls /dev -l` command verifies the new permissions for **Other**

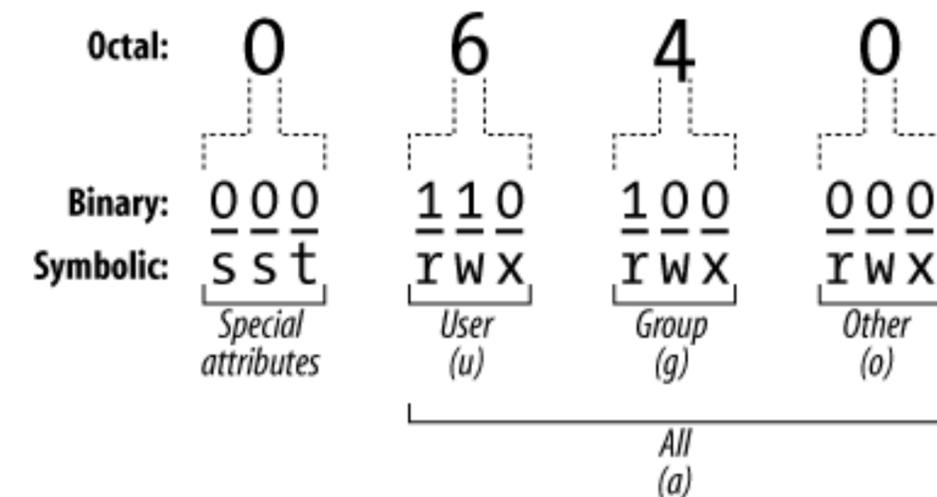
```
root@PFC200V3-46E709:/dev chmod 666 /dev/tty01
root@PFC200V3-46E709:/dev ls -l
crw-rw-rw----- 1 root dialout 246, 1 Dec 11 02:48 tty01
```

```
# ls -l file
-rw-r--r-- 1 root root 0 Nov 19 23:49 file
```

File type

Owner (rw-)
Group (r- -)
Other (r - -)

r = Readable
w = Writeable
x = Executable
- = Denied



Step 3:

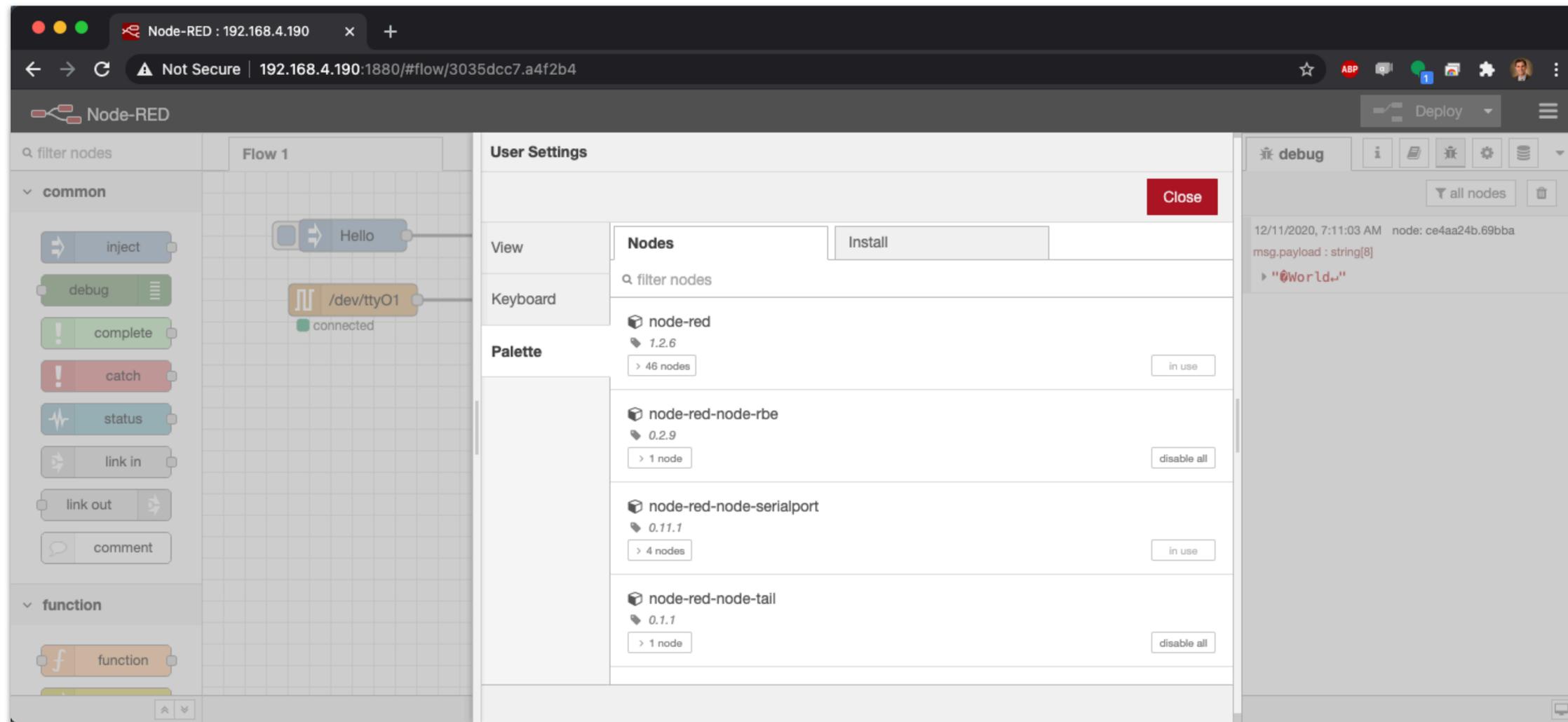
Install the Node-RED container with docker onto the PFC 200.

```
docker volume create node_red_user_data
```

```
docker run -d --restart unless-stopped --network=host -v node_red_user_data:/data --device=/dev/tty01:/dev/tty01:rw nodered/node-red
```

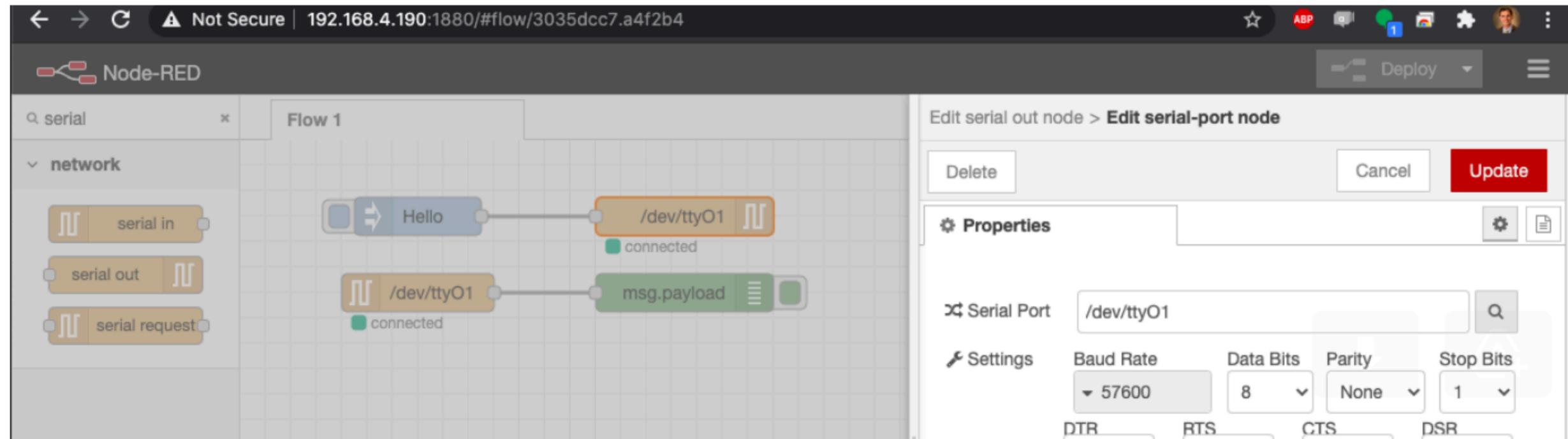
Step 4:

Install the Node-RED node called `node-red-serialport` in the palette manager.



Step 5:

Drag   in the order shown below for a simple test.



The screenshot shows the Node-RED web interface. On the left, a palette contains 'serial in', 'serial out', and 'serial request' nodes. The main workspace shows a flow with an 'inject' node containing 'Hello' connected to a '/dev/ttyO1' serial node, which is connected to another '/dev/ttyO1' serial node, which is connected to a 'msg.payload' debug node. The right-hand panel is open to the configuration for the selected '/dev/ttyO1' serial node. The 'Serial Port' is set to '/dev/ttyO1'. Under 'Settings', the 'Baud Rate' is set to 57600, 'Data Bits' is 8, 'Parity' is None, and 'Stop Bits' is 1. There are also fields for DTR, RTS, CTS, and DSR. Buttons for 'Delete', 'Cancel', and 'Update' are visible at the top of the configuration panel.

Configure the `serial-port` node for `/dev/ttyO1` with 56700 Baud Rate & 8N1

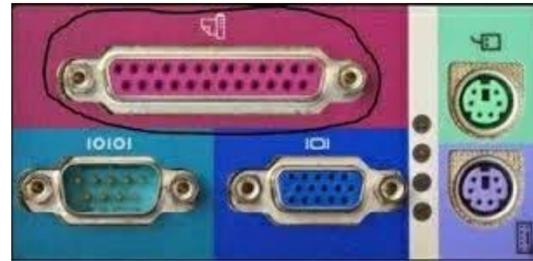
Step 6:

When you deploy, the serial nodes should show connected.



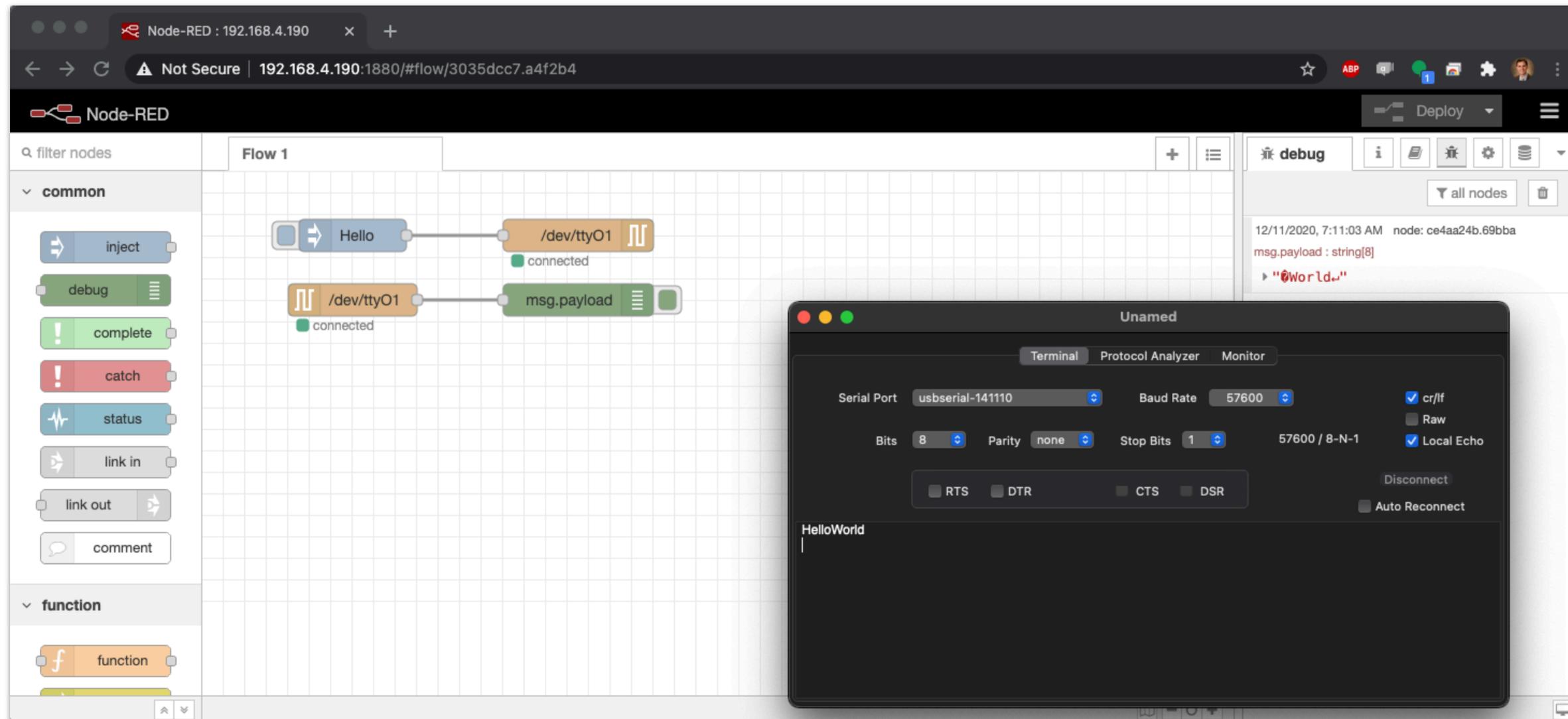
Step 7:

Connect a DB9 RS-232 cable to a computer (will a null modem inline) together with the PFC200 serial port.



Step 8:

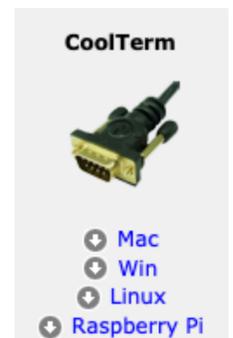
Using a serial terminal program on the computer, you should see ASCII in debug on both ends when you type and click the inject buttons. You now have a working serial port in Node-RED for your application development!



The screenshot shows the Node-RED web interface in a browser window. The main workspace displays a flow with two nodes: an 'inject' node with the text 'Hello' and a '/dev/ttyO1' node. Below this, another '/dev/ttyO1' node is connected to a 'msg.payload' node. The left sidebar shows various nodes under 'common' and 'function'. The right sidebar shows a 'debug' panel with a log entry: '12/11/2020, 7:11:03 AM node: ce4aa24b.69bba msg.payload : string[8] > "World"'. In the foreground, a terminal window titled 'Unnamed' shows the configuration for a serial port: 'Serial Port: usbserial-141110', 'Baud Rate: 57600', 'Bits: 8', 'Parity: none', 'Stop Bits: 1', and '57600 / 8-N-1'. The terminal output shows 'HelloWorld'.

If you need a serial terminal program, here is an open source one called CoolTerm

<https://freeware.the-meiers.org/>



Troubleshooting:

💡 If it does not work, you can troubleshoot with the following command to start your container in interactive mode, so you can see the error messages when node-red starts.

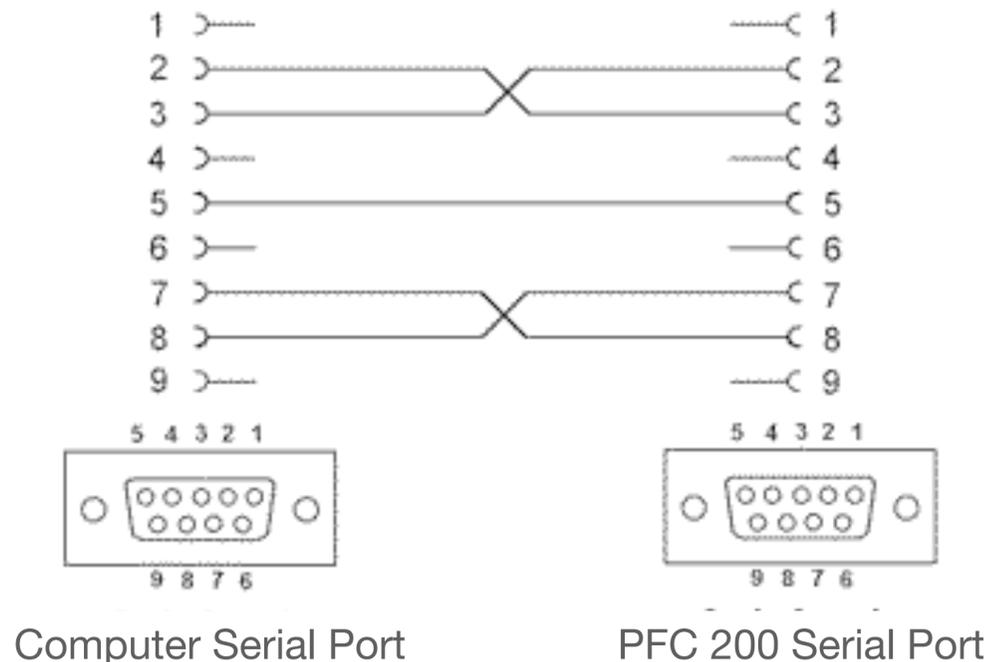
```
root@PFC200V3-46E709:/dev docker run --rm -t --network=host -v node_red_user_data:/data --device=/dev/tty01:/dev/tty01:rw nodered/node-red
```

```
11 Dec 12:11:56 - [info] Started flows
11 Dec 12:11:56 - [error] serial port /dev/tty01 error: Error: Error: Permission denied, cannot open /dev/tty01
```

💡 Make sure you press enter when sending strings from the computer to the PFC.

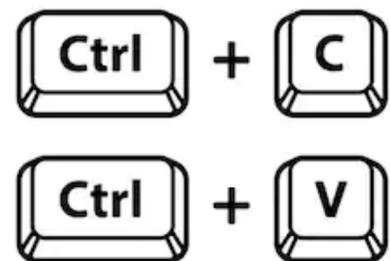
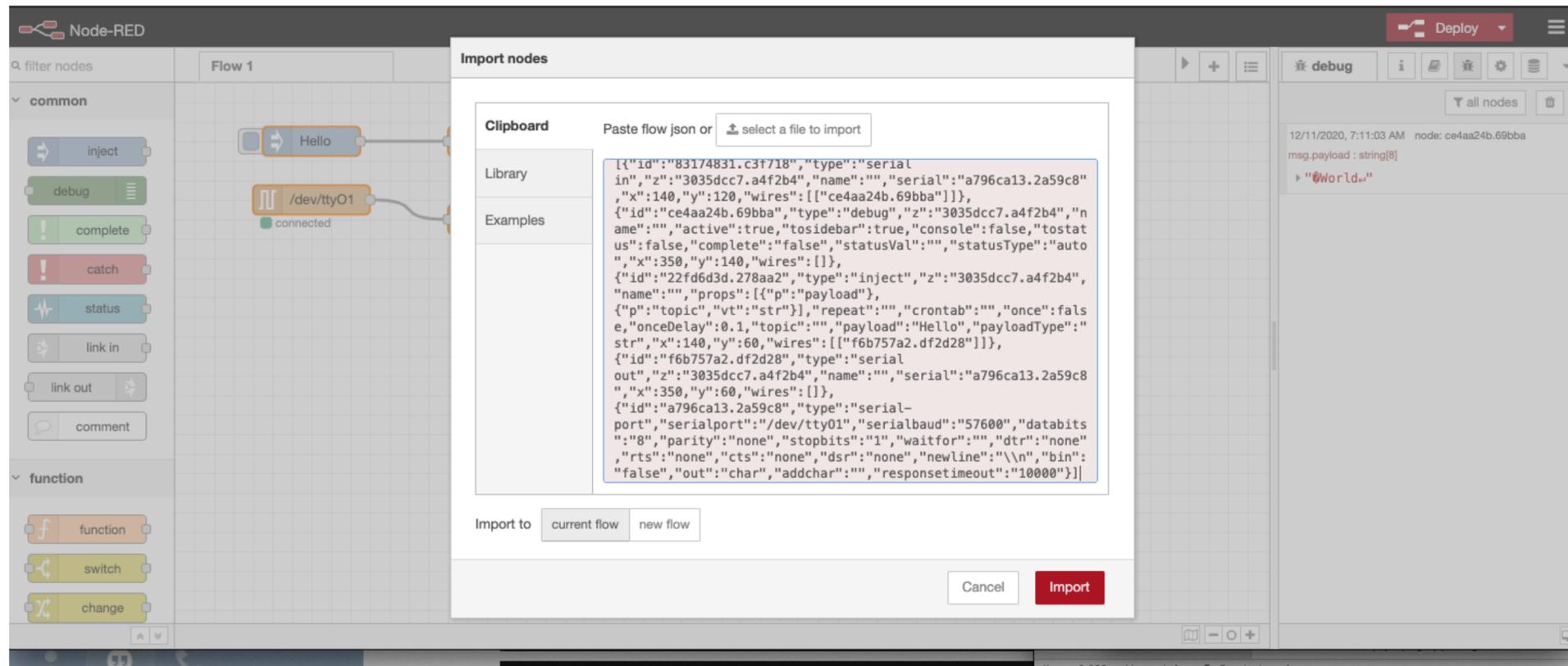


💡 If the serial node shows connected, but ASCII characters are not showing then check the wiring.



DCE Device (Modem)		DB9	DCE to DCE Connections	DCE Device (Modem)	DB9
Pin#	DB9	RS-232 Signal Names	Signal Direction	Pin#	DB9
#1	Carrier Detector (DCD)	CD		#1	Carrier Detector (DCD)
#2	Receive Data (Rx)	RD	→	#2	Receive Data (Rx)
#3	Transmit Data (Tx)	TD	←	#3	Transmit Data (Tx)
#4	Data Terminal Ready	DTR	→	#4	Data Terminal Ready
#5	Signal Ground/Common (SG)	GND		#5	Signal Ground/Common (SG)
#6	Data Set Ready	DSR	→	#6	Data Set Ready
#7	Request to Send	RTS	→	#7	Request to Send
#8	Clear to Send	CTS	→	#8	Clear to Send
#9	Ring Indicator	RI		#9	Ring Indicator
Soldered to DB9 Metal - Shield		FGND		Soldered to DB9 Metal - Shield	

Here is the example flow for this test. Import with copy/paste in Node-RED.



```
[{"id":"83174831.c3f718","type":"serial in","z":"3035dcc7.a4f2b4","name":"","serial":"a796ca13.2a59c8","x":140,"y":120,"wires":[["ce4aa24b.69bba"]]}, {"id":"ce4aa24b.69bba","type":"debug","z":"3035dcc7.a4f2b4","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":350,"y":140,"wires":[]}, {"id":"22fd6d3d.278aa2","type":"inject","z":"3035dcc7.a4f2b4","name":"","props":[{"p":"payload"}, {"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"Hello","payloadType":"str","x":140,"y":60,"wires":[["f6b757a2.df2d28"]]}, {"id":"f6b757a2.df2d28","type":"serial out","z":"3035dcc7.a4f2b4","name":"","serial":"a796ca13.2a59c8","x":350,"y":60,"wires":[]}, {"id":"a796ca13.2a59c8","type":"serial-port","serialport":"/dev/ttyO1","serialbaud":"57600","databits":"8","parity":"none","stopbits":"1","waitfor":"","dtr":"none","rts":"none","cts":"none","dsr":"none","newline":"\\n","bin":"false","out":"char","addchar":"","responsetimeout":"10000"}]
```