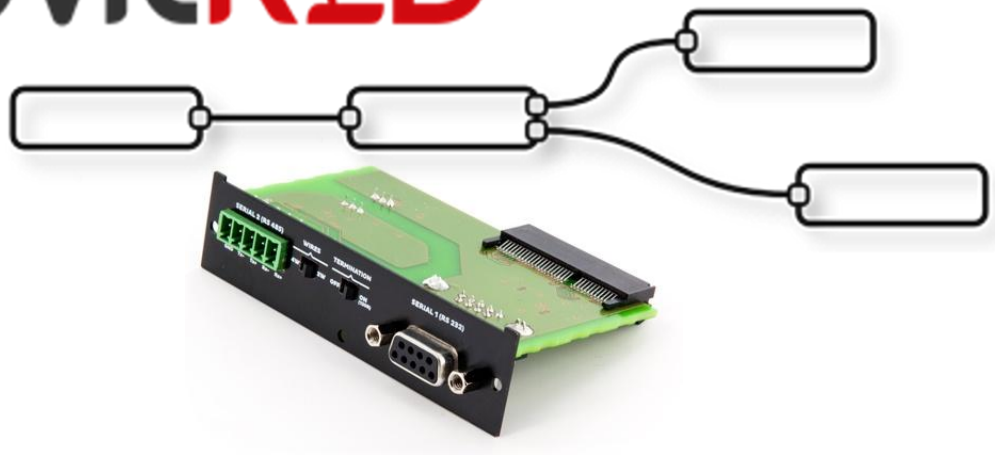


OPTION

LuvitRED



Modbus Poll using the industrial serial card

Franco Arboleda

04-Sep-15

Table of Contents

1	Introduction	3
2	Modbus configuration using the RS485 port.	4
3	Remote access to the Modbus data using a local TCP server	9

1 Introduction

This document covers the configuration of the industrial serial card using LuvitRED for a Modbus poll application:



Figure 1: Infinite serial cable application.

The industrial serial card (CG1102-11920) is shown on Figure 2 below:



Figure 2: Industrial Serial Card

- The **RS232** interface is shown on the operating system of the CloudGate as **/dev/ttySP0**
- The **RS485** interface is shown on the operating system of the CloudGate as **/dev/ttySP4**
- The **RS485** interface also has two switchable items that need to be set correctly on the plate:
 - The first item is the amount of wires to use: 4W or 2W.
 - The second item is the termination: OFF or ON.

For this configuration and any other Modbus configuration, one needs to use the "Advanced Editor" of LuvitRED.

2 Modbus configuration using the RS485 port.

For this configuration it is better to start from a real configuration. Let's say we have a Modbus RTU capable device (Slave ID 1) that has four holding registers (16 bits). The specification of the device mentions the following:

Slave ID	Holding Register	Contained Value
1	1	Temperature
1	2	Humidity
1	3	RPM
1	4	Current

- The "Temperature" register needs to be divided by 10 to obtain the real temperature in °C.
- The bytes of "Current" register need to be switched and the resulted value needs to be divided by 100 to obtain the real current in A.

This Modbus capable device also has a 2-Wire RS485 serial port with the following configuration:

Item	Value
Baud rate	9600
Parity	None
Data bits	8
Stop bits	1
RTS control	Disable

For this application we are going to use two Modbus nodes from LuvitRED: modbus poll and modbus extract.

The "modbus poll" node (See Figure 3) is the one that will be in charge of polling the information out of the Modbus RTU device.

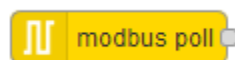


Figure 3: Modbus Poll node.

The "modbus extract" node (See Figure 4) is the one that will be in charge of extracting the information out of the polling and placing them into items inside a message that can be then deliver to other nodes.

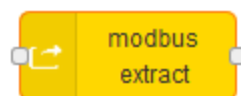


Figure 4: Modbus extract node.

Let's first configure our "modbus poll" node. Drag and Drop one "modbus poll" node into the LuvitRED editor and double click on it (See Figure 5).

Edit modbus poll node

Serial Port: Add new modbus requester... (dropdown) [pencil icon]

Topic: topic (text input)

Operation: Read Coils (dropdown)

Slave id: 1 (1 to 247)

Start at: 0 offset (0 to 65535)

Read: 1 coil

Tip: For 1 bit values (coils and discrete inputs) you can read from 1 to 2000 registers at a time.

Poll Rate: 5 seconds

Send: Always (dropdown)

Output: Raw string (dropdown)

Name: Name (text input)

Ok Cancel

Figure 5: New modbus poll node configuration.

The first item that needs to be changed is the serial configuration to match the one from the Modbus RTU device. For this, click on the pencil icon to edit the port configuration (To change the serial port selected, click on the magnifying glass):

Edit modbus requester config node

Protocol: Modbus RTU serial (dropdown)

Serial Port: /dev/ttySP4 (text input) [magnifying glass icon]

Settings: Baud Rate: 9600, Data Bits: 8, Parity: None, Stop Bits: 1

Leave: 100 milliseconds between queries

Tip: Time to wait between multiple queries. If multiple queries and timeouts, increase this value.

After: 1000 milliseconds query will timeout

Tip: Timeout for a single query. If a query timeouts on a single request (f.e. a slow device), increase this value.

Port mode: Do not switch mode (dropdown)

☐ Log raw PDUs (for debugging) ?

1 node uses this config

Delete Update Cancel

Figure 6: Serial configuration for the RS485 interface.

OPTION

After making sure the serial port is correctly updated, let's focus on the Modbus part of the polling:

1. We need to change the "Operation" from "Read coils" to "Read holding Registers"
2. Change the "Slave id" parameter to 1 since the slave id of the Modbus RTU device is 1.
3. Change the "Start at" parameter to 0. Zero is normally the first holding register on a Modbus device. In our case 0 points to register 1.
4. Change the "Read" parameter to 4 since we are reading 4 consecutive holding registers. If there is a need to read non-consecutive registers, then the best way is by using multiple 'modbus poll' nodes.
5. For this example we are going to keep the "Poll Rate", "Send" and "Output" parameters to their default values.
6. Change the "Name" parameter of the node to "Modbus_poll".

The configuration of the "modbus poll" node should look like the following:

The screenshot shows the 'Edit modbus poll node' configuration window. It contains the following fields and settings:

- Serial Port:** /dev/ttySP4:9600-8N1
- Topic:** topic
- Operation:** Read Holding Registers
- Slave id:** 1 (range 1 to 247)
- Start at:** 0 (range 0 to 65535)
- Read:** 4 registers
- Tip:** For 16 bit values (holding and input registers) you can read from 1 to 125 registers at a time.
- Poll Rate:** 5 seconds
- Send:** Always
- Output:** Raw string
- Name:** Modbus_Poll

At the bottom right, there are 'Ok' and 'Cancel' buttons.

Figure 7: Modbus poll node configured.

Click on "OK" to close the node configuration.

OPTION

Now it is time to add the "modbus extract" node and edit its content:

1. Change the "Name" of the node to something descriptive like "Extraction"
2. Add a new extraction by clicking on the little button on the bottom left corner.
3. The first extraction will start reading at byte 1, we will read 1 register and interpret it as a 16 bit unsigned integer, we will not change the order of the bytes received (AB). Then as mentioned on the description of the value, it needs to be divided by 10 (multiply by 0.1). The last part is to change the name of the value being extracted (Temp).
4. Add a second extraction for the humidity value. That one will start at byte 3 (default since we are reading registers of 2 bytes in size), we will read 1 register and interpret it as a 16 bit unsigned integer and we will not change the order of the bytes. We are going to change the name of the value (Hum).
5. Add a third extraction that will be the same as the one for the humidity value. The only difference is the starting point and the name of the value (RPM).
6. The last extraction will be for the current. It starts at byte 7 and we are reading 1 register interpreted as 16 bit unsigned register, but for this item we are going to change the byte order from AB to BA (simply because the specification of our Modbus RTU device specifies that the byte order is inverted). We now need to device the value by 100 (Multiply by 0.01) and change the name of the value (Curr).

The configuration of the "modbus extract" node should look like the following:

Edit modbus extract node

Name:

@ byte	1	read	1	16 bit unsigned i	order	AB	<input type="button" value="X"/>
multiply by	0.1	and add	0	name		Temp	
@ byte	3	read	1	16 bit unsigned i	order	AB	<input type="button" value="X"/>
multiply by	1	and add	0	name		Hum	
@ byte	5	read	1	16 bit unsigned i	order	AB	<input type="button" value="X"/>
multiply by	1	and add	0	name		RPM	
@ byte	7	read	1	16 bit unsigned i	order	BA	<input type="button" value="X"/>
multiply by	0.01	and add	0	name		Curr	

Figure 8: Modbus extract node configured.

Click on "OK" to close the node configuration.

OPTION

Now, for the last part, let's drop a debug node and connect the three nodes the following way:



Figure 9: Final configuration.

At this point, the "Modbus_Poll" node should be getting values from the Modbus RTU device via the RS485 interface, then it is passing those values to the "Extraction" node which is performing the extraction of the values and then we are printing these values to the debug tab on LuvitRED:

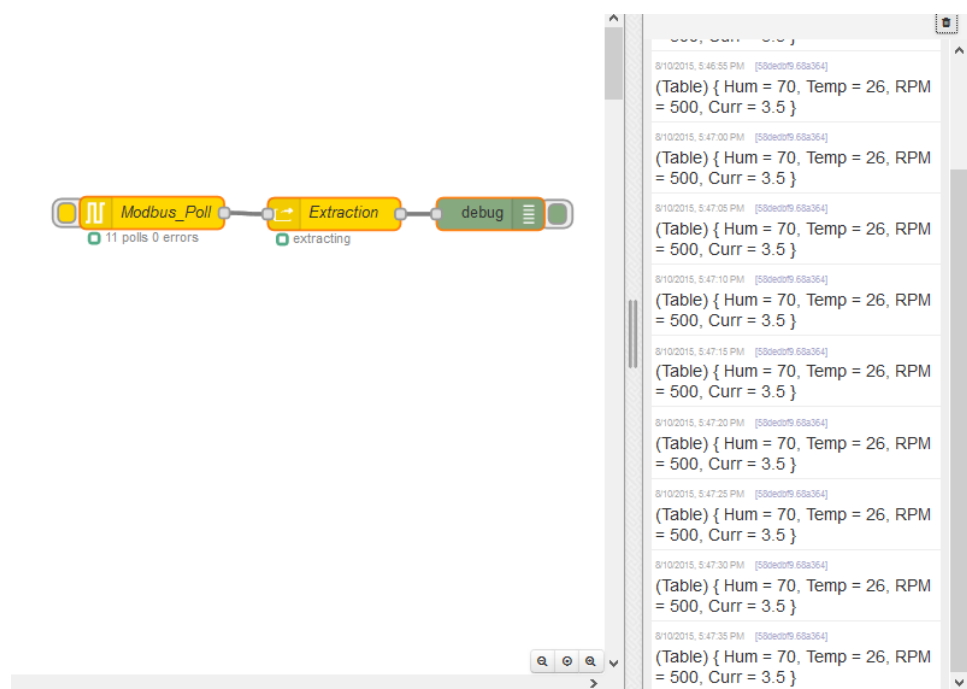


Figure 10: Debug tab output with Modbus values.

3 Remote access to the Modbus data using a local TCP server

For this example we are going to start a local TCP server running on the CloudGate that will basically print the same information we saw over the debug tab, but on a TCP connection. This TCP connection can be started remotely by a computer that wants to obtain the readings from the Modbus RTU device.

Of course, this is not the only possibility with LuvitRED, we could also send this information to remote server or take an action (send an SMS or an email) if a value is higher or lower than a threshold, etc.

Let's start by dropping a TCP output node:

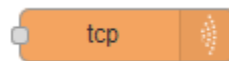


Figure 11: TCP output node.

In order to start a local TCP server on the CloudGate, we need to modify the following parameters on the TCP output node:

Edit tcp out node

Endpoint

Add new tcp endpoint...

Send

always to all

Enable queueing when no connection ?

Name

Name

Ok

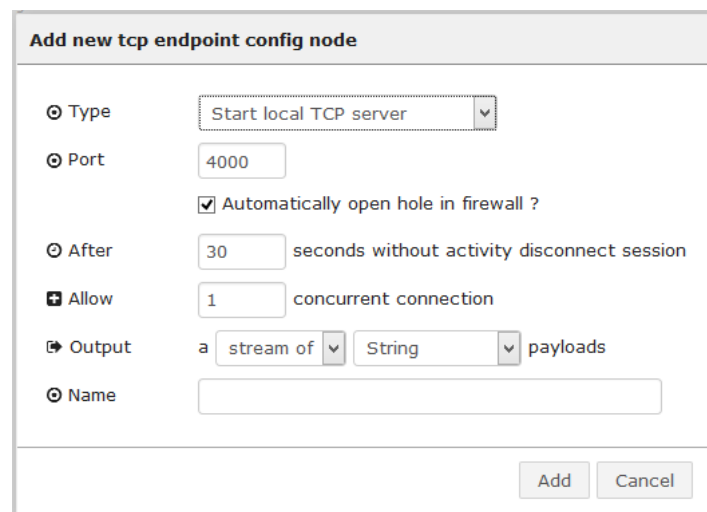
Cancel

Figure 12: Non-configured TCP output node.

OPTION

Add a new "Endpoint" by clicking over the pencil icon. The new endpoint should have the following configuration:

1. Change the "Type" to "Start local TCP server"
2. Change the "Port" to any port you would like to use for this test. On this example we are going to use TCP port 4000
3. Check the "Automatically open hole in firewall?" setting to allow incoming WAN connections to the port.
4. Add a 30 seconds activity timeout on the "After ___ seconds without activity disconnect session" setting.



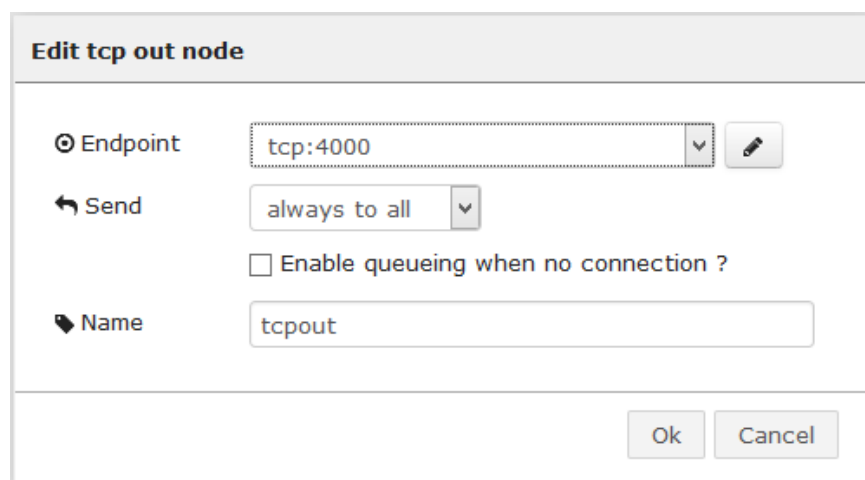
The dialog box is titled "Add new tcp endpoint config node". It contains the following fields and controls:

- Type:** A dropdown menu with "Start local TCP server" selected.
- Port:** A text input field containing "4000".
- Automatically open hole in firewall ?** A checked checkbox.
- After:** A text input field containing "30", followed by the text "seconds without activity disconnect session".
- Allow:** A text input field containing "1", followed by the text "concurrent connection".
- Output:** A label "a" followed by a dropdown menu with "stream of" selected, another dropdown menu with "String" selected, and the text "payloads".
- Name:** An empty text input field.
- Buttons:** "Add" and "Cancel" buttons at the bottom right.

Figure 13: Configuration of the TCP endpoint.

Click on "Add" to close the "Endpoint" configuration.

Once back on the node configuration, change the name of the node to something descriptive like **tcpout**:



The dialog box is titled "Edit tcp out node". It contains the following fields and controls:

- Endpoint:** A dropdown menu with "tcp:4000" selected, and a pencil icon button to the right.
- Send:** A dropdown menu with "always to all" selected.
- Enable queueing when no connection ?** An unchecked checkbox.
- Name:** A text input field containing "tcpout".
- Buttons:** "Ok" and "Cancel" buttons at the bottom right.

Figure 14: TCP output node configuration finished.

Click on "OK" to close the node configuration.

OPTION

Now, we just simply need to drop a json node and connect it in between the Extraction and the tcpout nodes as show below:

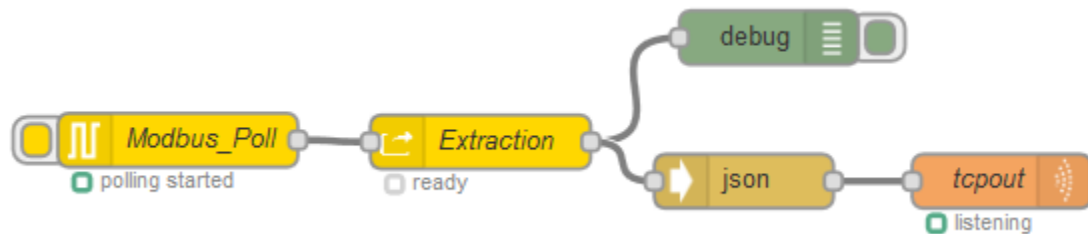


Figure 15: TCP output node connected to Extraction node.

The json node is used to transform the output of the Extraction node from a table to a String that can be printed on the tcpout node.

If we now try to open a TCP connection to the CloudGate on port 4000, we should be able to get the same output as was shown on Figure 10:

```
192.168.1.1 - PuTTY
{"Hum":70,"Temp":26,"RPM":500,"Curr":3.5}{"Hum":70,"Temp":26,"RPM":500,"Curr":3.5}{"Hum":70,"Temp":26,"RPM":500,"Curr":3.5}{"Hum":70,"Temp":26,"RPM":500,"Curr":3.5}
```

Figure 16: Modbus output on a local TCP server.

○ ○
○ ○ P T I O N
○ ○