|  |
| --- |
| Telematics card on LuvitRED |
|  |





**Franco Arboleda**

**13-Aug-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

# Introduction

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



Figure : Infinite serial cable application.

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



Figure : 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.

# 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 ºC.
* 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 : 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 : 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).

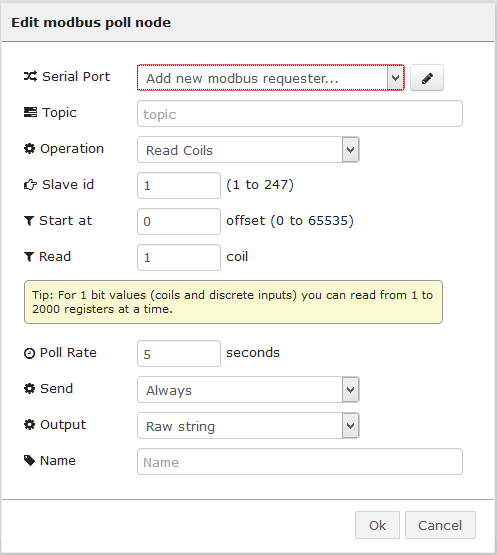


Figure : New modbus poll node configuration.

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):

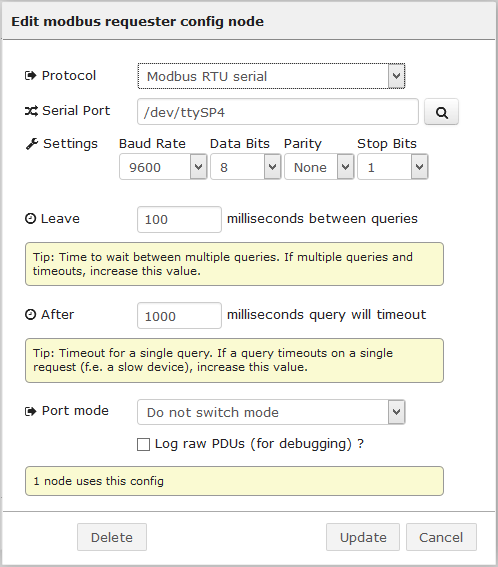


Figure : Serial configuration for the RS485 interface.

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 keet 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:

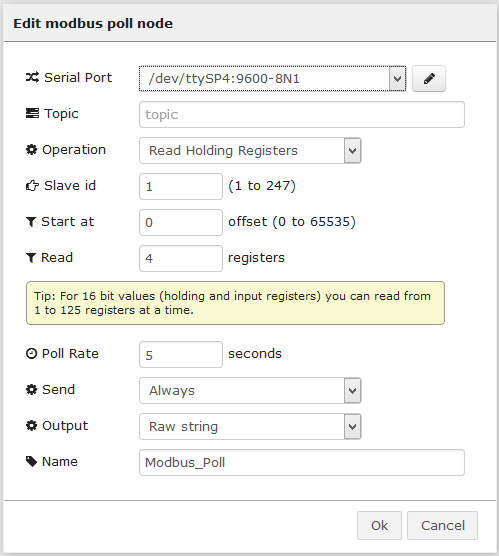


Figure : Modbus poll node configured.

Click on "OK" to close the node configuration.

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 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 byters 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 is 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 unsinged 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:

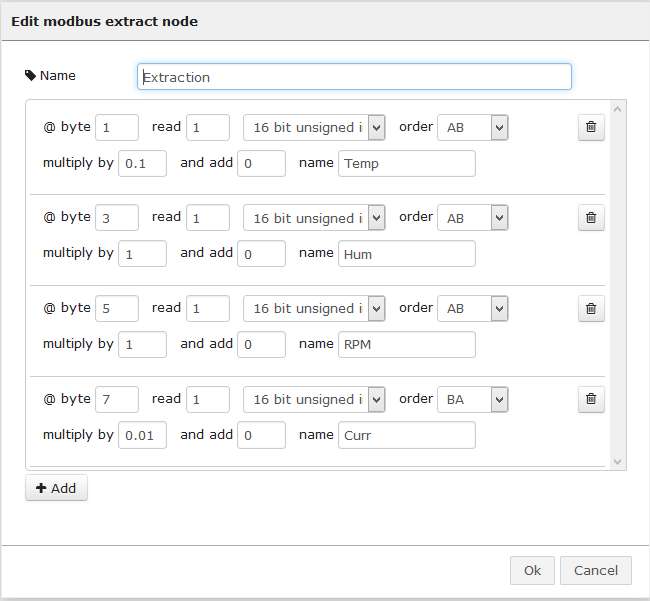


Figure : Modbus extract node configured.

Click on "OK" to close the node configuration.

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



Figure : Final configuration.

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

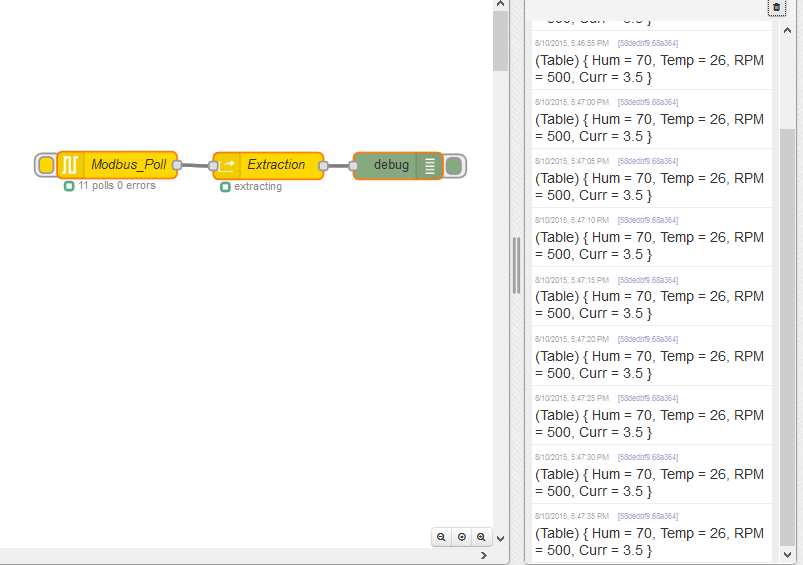


Figure : Debug tab output with Modbus values.

# 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 : 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:

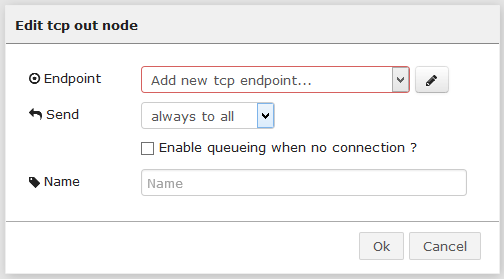


Figure : Non-configured TCP output node.

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.

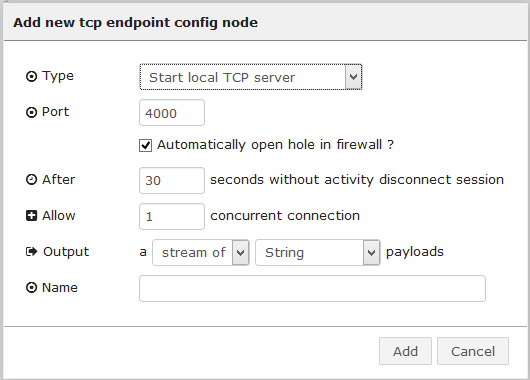


Figure : Configuration of the TCP endpoint.

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****:*

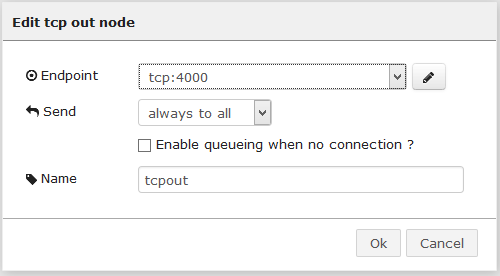


Figure : TCP output node configuration finished.

Click on "OK" to close the node configuration.

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

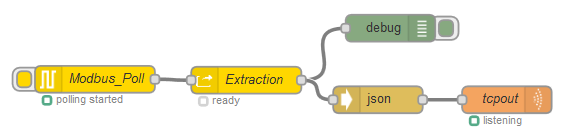


Figure : TCP output node connected to Extraction node.

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:

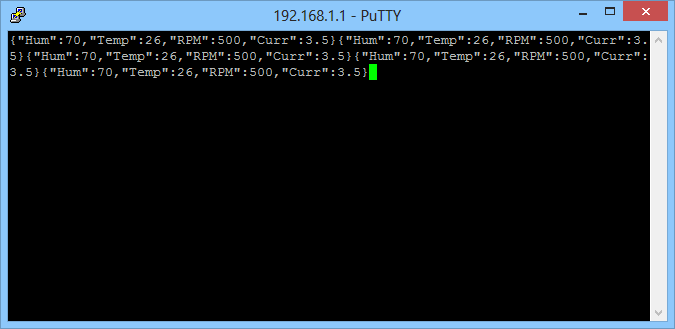


Figure : Modbus output on a local TCP server.