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





**Franco Arboleda**

**13-Aug-15**

Table of Contents

1 Introduction 3

2 Using the RS232 interface from the front panel. 5

2.1 Modifying the configuration under the Advanced Editor. 8

2.1.1 Verifying if firewall hole is openned by LuvitRED 9

2.1.2 Inactivity timeout on the TCP node. 10

3 Using the USB ports for storage. 11

3.1 Writing data to the mass storage device. 11

3.2 Reading data from the mass storage device. 13

4 Configuring the I/O interfaces. 15

# Introduction

This document covers the configuration of the telematics card using LuvitRED.

There are currently three versions of the telematics card:

1. Telematics base board (CG1106-11957):



Figure : Telematics base board.

1. Telematics card with I/O expander (CG5106-11983):



Figure : Telematics card with I/O expander.

1. Telematics card with CAN I/O expander (CG5106-11984):



Figure : Telematics card with CAN I/O expander.

There are two differences between the Telematics card with I/O expander and the Telematics card with CAN I/O expander:

1. 2 of the digital outputs of the I/O expander are used for the CAN BUS protocol on the CAN I/O expander.
2. The Auxiliary serial port available on the molex connector of the I/O expander is switcheable between RS232 and RS485 (2 Wires) on the CAN I/O expander.

Visit the CloudGate Universe (http://cloudgate.option.com/) for more information about these different versions of the telematics card and their configurations.

For this document, the telematics card with I/O expander is going to be used as a base and any little difference in configuration that might be necessary when working with the other versions will be mentioned.

There is an extra card that will be used during this document. This card is a breakout board (CG7101-12018) designed specifically for the telematics card with I/O and CAN I/O expanders:



Figure : Breakout board.

# Using the RS232 interface from the front panel.

Go to the "Plugin" tab, under it one will find a sub-tab called "Serial and GPS settings" or "LuvitRED" (The name depends on the LuvitRED version being used):



Figure : Plugin tab, Serial and GPS settings.



Figure : Plugin tab, LuvitRED.

Without any configuration, the basic interface looks as follows:

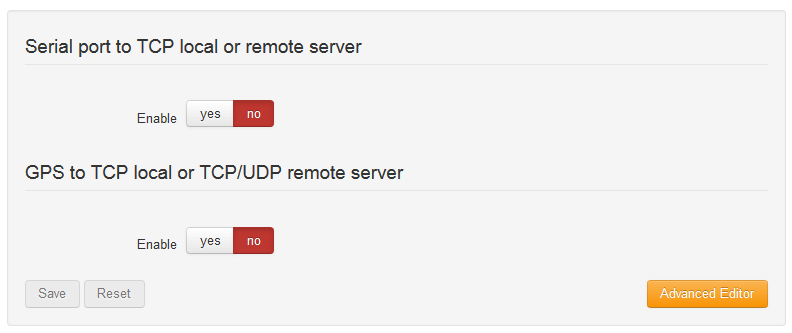


Figure : Basic interface.

For the configuring the RS232 interface we are going to focus on the section called "Serial port to TCP local or remote server". This section allows the configuration of one single serial port, the RS232 (**/dev/ttySP0** by default), to be accessible remotely via a local TCP server running on the CloudGate (See Figure 8) or a remote TCP server, running at another location (See Figure 9).

**NOTE**: Other, more advanced configurations, can be achieved by using the "Advanced Editor" of LuvitRED.



Figure : Serial to local TCP server.



Figure : Serial to Remote TCP server.

On both configurations, one can find the configuration of the serial interface (***Serial port settings****)*:

* Baud rate
* Data bits
* Stops bits
* Parity
* Flow control

These settings need to match the setting of the device connected to the serial interface.

On Figure 8, the CloudGate is running a local TCP server that will listen for incoming connections and forward them to the serial port. The Port number of the TCP server is, by default, **8889**, but it can be changed by the customer at any moment.

If access from the WAN interface (internet) is needed, an appropriate firewall rule needs to be in place to allow the connection to the port:



Figure : Inbound port forwarding rule.

**NOTE:** Recent versions of LuvitRED already open a firewall hole to allow remote access from the WAN interface. This can be verified only under the advanced editor, not on the basic interface (see section 2.1.1).

In Figure 9, the CloudGate will connect to a remote TCP server running on the specified port and send all the information that arrives from the device connected to the serial interface. **Be aware that this configuration may cause high data traffic.**

## Modifying the configuration under the Advanced Editor.

After configuring the serial port under the basic interface, one can go to the Advance Editor and edit the configuration. The configuration made on the basic interface will be reflected under the Advanced editor in the following way:



Figure : Same configuration under Advanced editor.

### Verifying if firewall hole is openned by LuvitRED

Double click on the ***tcpin*** node:

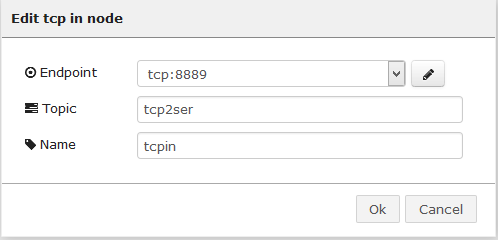


Figure : Tcpin node general configuration.

One can access the "Endpoint" configuration by clicking on the pencil icon:



Figure : Endpoint configuration.

Check if the configuration item called "Automatically open a hole in firewall?" is checked or modify it according to the needs of the configuration.

### Inactivity timeout on the TCP node.

Open the "Endpoint" configuration as explained on section 2.1.1. Once there, add a timeout, in seconds (30 on the example below), on the "After \_\_\_ seconds without activity disconnect session" configuration item so that it closes any open connection that is not generating traffic:



Figure : Adding connection timeout.

# Using the USB ports for storage.

From firmware version 1.46.0/2.46.0 onwards, automount for USB and SD mass storage devices (FAT file systems only) is supported on the CloudGate hardware.

Any new FAT formatted drive will be mounted under the ***/nmt/*** directory. Normally these drives are mounted as ***sdX#***:



Figure : Example of two FAT drives mounted on the linux system (sda1 and sdb1).

On Figure 15, one can see two drives mounted on the system, sda1 and sdb1. The sda1 drive is connected to the USB Type A interface while the sdb1 drive is connected on the USB OTG interface using a micro-USB to USB adapter.

Under the "Advanced editor" of LuvitRED, there are some nodes that are in charge of data storage and others for parsing data (See Figure 16):

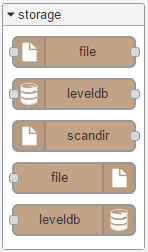
 

Figure : Storage and parsing nodes.

## Writing data to the mass storage device.

Let's say we want to write a file to the ***sda1*** drive which currently looks like this:



Figure : Current view of sda1.

We can drop a file out node:



Figure : File out node.

Configure the file out node like this:

1. Add the filename to write using the full location: ***/mnt/sda1/file.txt***
2. Choose an action for the node (append to file in our case), there are three actions available:
   1. append to file
   2. overwrite file
   3. delete file
3. Choose if you want to add a ***newline*** character at the end of every line written to the file.
4. Change the node's name.

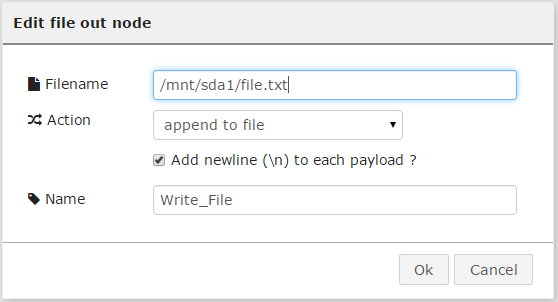


Figure : File out node configuration.

Now, let's drop an inject node to send some data to the file out node. In this case the Inject node is configure to send a string "write test" every time we press the inject button:

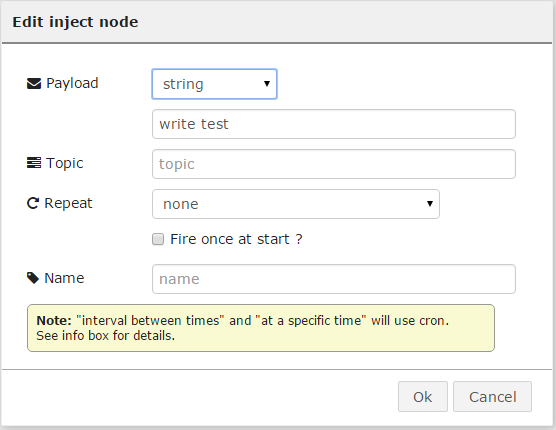


Figure : Inject node configuration.

Now, connect both nodes together the following way:



Figure : Write flow.

Deploy the configuration.

After pressing the inject button next to the inject node a few times (3 times in this example). The file is containing the following information when reading it on a SSH session:



Figure : File containing new information on the sda1 drive.

## Reading data from the mass storage device.

Now that we have written information to a file in the ***sda1*** drive on section 3.1, we want to read it back.

For reading the ***/mnt/sda1/file.txt*** we need first to drop a file in node:



Figure : File in node.

Configure the file in node like this:

1. Add the filename to read using the full location: ***/mnt/sda1/file.txt***
2. Choose a Read file action for the node (once per message in our ase), there are two actions available:
   1. once per message
   2. continuosly
3. Choose if you want to delete the file after a successful read (leave it blank for our example).
4. Change the node's name.

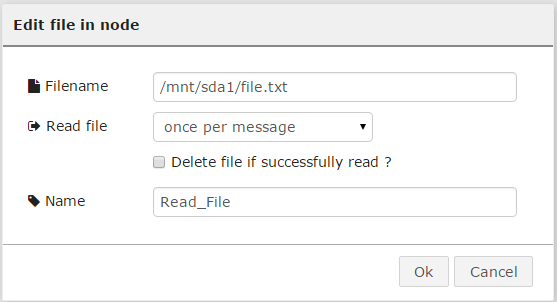


Figure : File out node configuration.

Now, let's drop an inject node to trigger the file in node to read (this will be the message that the node is waiting for reading "once per message"). In this case the Inject node is configure with its default values, so no change on its configuration:

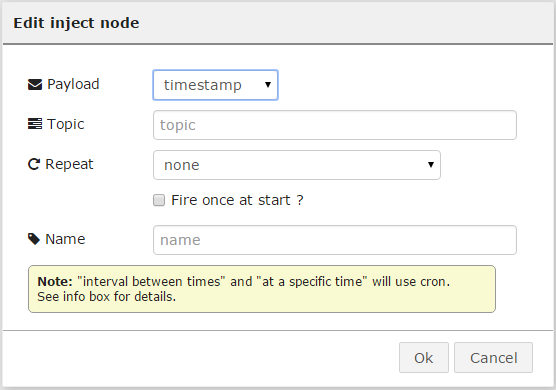


Figure : Inject node configuration.

Let's also drop an debug node to view the result of reading the file.

Connect the three nodes together the following way:



Figure : Read flow.

Deploy the configuration.

After pressing the inject button next to the inject node, the debug node should print the reading made by the file in node and print the result on the debug tab:

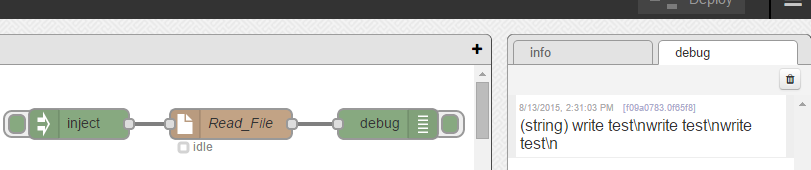


Figure : Debug node printing the result of reading the file.

Of course, printing the file to debug might not be something useful, but it is a good step to show that the file was correctly read. Instead of a debug node, or together with it, one could place other kind of nodes to, for example, send the file to a remote server, or make it available for a remote incoming connection.

# Configuring the I/O interfaces.