

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
4.1	Digital outputs	16
4.2	Digital inputs	19
4.3	Analog inputs	22

1 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 1: Telematics base board.

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



Figure 2: Telematics card with I/O expander.

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



Figure 3: 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 4: Breakout board.

2 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):

oning	System	Plugin +	VF	N
		Serial and	d GP	S settings
Figure	e 5: Plugin	tab, Serial	and	GPS settings.
oning	System	n Plugin	•	VPN
		Luvit	RED	
		SNM	Р	

Figure 6: Plugin tab, LuvitRED.

Without any configuration, the basic interface looks as follows:

Serial port to TCP local or remote server	
Enable yes no	
GPS to TCP local or TCP/UDP remote server	
Enable yes no	
Save	Advanced Editor



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/ttySPO 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.

Enable Serial port settings Baud rate 9600 Data bits 0 Data bits 7 0 at bits 9 1000000000000000000000000000000000000
Serial port settings Baud rate 9600 Data bits 0
Baud rate 9600 V Data bits 7 0 8 Stop bits 0 1 2 Parity 0 none 0 even 0 odd 0 mark 0 space Flow control 0 none
Data bits 07 08 Stop bits 01 2 Parity 0 none 0 even 0 odd mark 0 space Flow control 0 none
Stop bits 0 2 Parity 0 none 0 even 0 odd 0 mark 0 space Flow control 0 none 0 voluyOEE
Parity onone overn odd mark space Flow control Onone Overn
Flow control
O CTS/RTS
TCP settings
TCP server is Local Remote
Port 8889 🗎
GPS to TCP local or TCP/UDP remote server
Enable yes no Advanced Editor

Figure 8: Serial to local TCP server.

Serial port to TCP	ocal or remote server
Enable	yes no
Serial port settings	
Baud rate	9600
Data bits	○ 7 ● 8
Stop bits	● 1 ○ 2
Parity	 none even odd mark
Flow control	 ○ space ● none ○ XON/XOFF ○ CTS/RTS
TCP settings	
TCP server is	Local Remote
Hostname	RemoteServerIP
Port	8889
GPS to TCP local of	or TCP/UDP remote server
Enable	yes no
Save Reset	Advanced Editor

Figure 9: 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:

Edit inbound p	ort forwarding r	ule ×
Protocol	ТСР	~
Inbound interface	ALL	~
Source IP	 Any Specific: 	
Destination port	8889	
Target IP address	192.168.1.1	
Target destination port	8889	
		Cancel Add

Figure 10: 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.**

2.1 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 11: Same configuration under Advanced editor.

2.1.1 Verifying if firewall hole is openned by LuvitRED

Double click on the **tcpin** node:

Edit tcp in node	1	
⊙ Endpoint	tcp:8889	 Image: Image: Ima
📰 Торіс	tcp2ser	
Name	tcpin	
		Ok Cancel

Figure 12: Tcpin node general configuration.

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

Edit tcp endpoint config node				
⊙ Туре	Start local TCP server 🗸			
⊙ Port	8889			
	Automatically open hole in firewall ?			
O After	seconds without activity disconnect session			
Allow	1 concurrent connection			
🕞 Output	a stream of 💙 String 💙 payloads			
⊙ Name				
2 nodes use t	his config			
	Delete Update Cancel			

Figure 13: 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.

2.1.2 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 <u>seconds</u> without activity disconnect session" configuration item so that it closes any open connection that is not generating traffic:

Edit tcp endpoint config node			
🖸 Туре	Start local TCP server		
⊙ Port	8889		
	\checkmark Automatically open hole in firewall ?		
O After	30 seconds without activity disconnect session		
Allow	1 concurrent connection		
🕞 Output	a stream of 🗸 String 🗸 payloads		
⊙ Name			
2 nodes use	this config		
	Delete Update Cancel		

Figure 14: Adding connection timeout.

3 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#*:

admin@cgate:~\$	ls /mnt/		
base_cfg cust	cust_cfg	sda1	sdb1
admin@cgate:~\$			

Figure 15: 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):

	▼ parsers		
	nmea parser		
▼ storage	og≝ compress		
file	csv		
leveldb	json 🛉		
scandir	message pack		
file	xml		
leveldb S	nmea crc		

Figure 16: Storage and parsing nodes.

3.1 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:

admin@cgate:	-\$ ls /mnt/sd	a1/	
1_DATA	HBCD	drivers	home
CLEAN	autorun.inf	grldr	menu.lst
admin@cgate:	~\$ <mark>-</mark>		

Figure 17: Current view of sda1.

We can drop a file out node:



Figure 18: 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:
 - a. append to file
 - b. overwrite file
 - c. 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.

Edit file out no	de
Filename	/mnt/sda1/file.txt
X Action	append to file •
	$\ensuremath{\mathbb{Z}}$ Add newline (\n) to each payload ?
🗣 Name	Write_File
	Ok Cancel

Figure 19: 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:

Payload	string 🔻
	write test
🛢 Topic	topic
C Repeat	none •
	Fire once at start ?
Name	name
Note: "interval See info box fo	between times" and "at a specific time" will use cron. r details.

Figure 20: Inject node configuration.

Now, connect both nodes together the following way:



Figure 21: 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:

admin@cgate:	-\$ ls /mnt/sd	a1/	
1_DATA	HBCD	drivers	home
CLEAN	autorun.inf	grldr	menu.lst
admin@cgate:	-\$ cat /mnt/s	da1/file.txt	
write test			
write test			
write test			
admin@cgate:/	~\$ <mark> </mark>		

Figure 22: File containing new information on the sda1 drive.

3.2 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 23: 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:
 - a. once per message
 - b. 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.

Edit file in nod	e
Filename	/mnt/sda1/file.txt
🗭 Read file	once per message
	Delete file if successfully read ?
🗣 Name	Read_File
	Ok Cancel

Figure 24: 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:

Edit inject noo	le
💌 Payload	timestamp 🔹
nter Topic	topic
C Repeat	none •
	Fire once at start ?
🗣 Name	name
Note: "interva See info box fo	I between times" and "at a specific time" will use cron. r details.
	OK Cancer

Figure 25: 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 26: 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 27: 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.

4 Configuring the I/O interfaces.

On the telematics cards, models CG5106-11983 and CG5106-11984, there are three different types of IO interfaces as explained on the CloudGate universe:

- 5 Digital inputs (11, 12, 13, 14 and 15)
- 2 Analog inputs (Al1 and Al2)
- 6 Digital outputs (DO1, DO2, DO3, DO4, DO6 and DO7) 4 DOs in the case of the telematics with CAN IO expander DO3 and DO4 are used for the CAN BUS interface.

There are three GPIO related node on LuvitRED, GPIO in, GPIO out and GPIO query:



Figure 28: GPIO nodes.

- The GPIO in node is used to read the values of either a digital input or an analog input pin.
- The GPIO out node is used to write a value to a Digital output pin.
- The GPIO query node is used to query the status of an IO pin.

On the breakout board, there is access to those IO pins, for the Digital output there are some LEDs, for the Digital inputs there are some buttons and for the Analog inputs there are also some connectors too:



Figure 29: Front of the breakout board.

4.1 Digital outputs

Let's say that we want to turn on the LEDs of the breakout board ON. We first need to change the LED switch to the "ON" position (pressed towards the top side), when this is done, the LED number 5 is lit:



Figure 30: Switch on "ON" position and LED number 5 ON.

Let's start by turning on and off one single LED using two inject nodes and one GPIO out node.

Drop a GPIO out node and configure it the following way:

Edit gpio out r	ode	
🗱 Pin	Add new gpio out pin	▼ 8
🗩 Message	msg. payload	
	Invert output ?	
Name	Name	
		Ok Cancel

Figure 31: Default GPIO out node.

1. Add a new gpio out pin by pressing on the pencil icon and configure it for O1 (DO1):

Add new gpio out pin config node		
# Pin number	01 •	
⊘ Output type	Push pull •	
O Pin is	Off • at star	t
		Add Cancel

Figure 32: DO1 configuration.

2. Then let the rest of the configuration as it is and change the name of the node

🗱 Pin	Output pin O1	▼ #
🗩 Message	msg. payload	
	Invert output ?	
Name	DO1	

Figure 33: Final configuration of the GPIO out node.

NOTES:

- 1. The "Message" option of the configuration is determining what part of the message will be use as a trigger. In our case we will use the payload, but on more sofisticated configurations, one might need to use something else like *msg.payload.trigger*.
- 2. The "Invert output?" option of the configuration is simply inverting the meaning of the message received. If *msg.payload* is equal to 1 in our case, then the node will understand that 1 as 0.

Drop an Inject node and configure it the following way:

- 1. Change the Payload to "number"
- 2. Then add a "1" as the payload.
- 3. Leave the rest as it is and change the name of the node to "ON"

Edit inject node	2
► Payload	number •
	1
nterio El Topic	topic
C Repeat	none •
	Fire once at start ?
🗣 Name	ON
Note: "interval b See info box for	between times" and "at a specific time" will use cron. details.
	Ok Cancel

Figure 34: Inject node "ON" configuration.

Drop a second Inject node and configure it the same way as the first inject, but in this case set the payload to "0" and change the name to "OFF":

Edit inject noo	de
► Payload	number •
	0
nter Topic	topic
C Repeat	none
	Fire once at start ?
🗣 Name	OFF
Note: "interva See info box fo	I between times" and "at a specific time" will use cron. r details.
	Ok Cancel

Figure 35: Inject node "OFF" configuration.

Now let's connect the three nodes together the following way:





Deploy the new configuration and test the DO1 by pressing on the "ON" and "OFF" injects:



Figure 38: Off inject pressed.



This configuration can be reproduced to all the other LEDs by simply adding more GPIO out nodes and configure them by adding a new gpio out pin as shown on Figure 32. For this example we are connect the same inject nodes to all GPIO pins:



Figure 39: All Digital outputs to the same triggers.



Figure 40: All LEDs ON.

4.2 Digital inputs

Let's say that instead of turning the LEDs on by using an inject node we would like to do it by pressing a button (Digital Input) and turning them off pressing another button.

Drop a GPIO in node and configure it the following way:

🛱 Pin	Add new gpio in pin	▼ ♂
🗲 Message	always sent 🔹	
🛢 Topic	topic	
Name	Name	

Figure 41: Default GPIO in node.



- 3. Add a new gpio in pin by pressing on the pencil icon and configure it for 11:
 - a. Set the mode to Digital (digital inputs can also work as analog inputs)
 - b. Set the Pull mode to "pull the input up"
 - c. Set the read interval to 15 ms

Add new gpio in	pin config node
🗱 Pin number	I1 • Mode Digital •
🖋 Pull mode	pull the input up
② Read pin	every
	Add Cancel

Figure 42: 11 configuration.

4. After adding the pin, change the Message to only sent on change and then rename the node:

Edit gpio in nod	e	
☆ Pin	Input pin I1	▼ 8
🖋 Message	only sent on change 🔹	
🛢 Topic	topic	
Name	DIN1	
		Ok Cancel

Figure 43: Final configuration of the first GPIO in node.

Drop a second GPIO in node and configure it the exact same way as the first node, but now adding the I2 pin instead:

dit gpio in no	de	
♯ Pin	Input pin I2	▼ #
🔑 Message	only sent on change	
E Topic	topic	
🗣 Name	DIN2	
		Ok Cancel

Figure 44: Final configuration of the second GPIO in node.

Now, in order to make the second GPIO in to have a reversed action as for the first GPIO in pint, we need to add another node that can make such change for us. There are several, but for this example we are going to use a "change" node:

<mark>o</mark> χ¦ ch	ange
----------------------	------

Figure 45: Change node.

We need to configure it the following way:

it change node		
Set the va	lue of the message property •	
called	msg. payload	
to	0	
Tip: expects name of ano	a new property name and either a fixed value OR the full ther message property eg: msg.sentiment.score	
🗣 Name	Change	
	Ok Cancel	

Figure 46: Change node configuration.

We are basically telling the node to change the **payload** of the message to 0 when triggered.

Connect the nodes the following way and deploy:



Figure 47: Final configuration using Digital inputs for one output.



Figure 48: Final configuration using digital inputs for multiple outputs.

4.3 Analog inputs

For this example we are going to use the first input (11) of the breakout board connected to the sixth input connector (A11) using a resistor the following way:



Figure 49: Resistor connected between the digital and analog inputs.

The idea behind doing this is that this setup will show a voltage variation over time on the digital input the same way a sensor will do, and when the button is pressed the voltage will go down to zero.

Drop a GPIO in node and configure it the following way:

Edit gpio in no	de	
🗱 Pin	Add new apio in pin	▼ ▲
ℯ Message	always sent 🔻	
🛢 Topic	topic	
Name	Name	
		Ok Cancel

Figure 50: Default GPIO in node.

- 1. Add a new gpio in pin by pressing on the pencil icon and configure it for Al1:
 - a. Set the mode to Analog (default after selecting AI1)
 - b. Set the read interval to 500 ms

🗱 Pin number	AI1	• M	ode A	nalog	•
ට Read pin	every	▼ 500	milli	seconds	

Figure 51: Al1 configuration.



2. After adding the pin, rename the node:

Edit gpio in no	de	
# Pin	Input pin AI1	▼ 🖉
📕 Message	only sent on change 🔹	
🛢 Topic	topic	
Name	AIN1	
		Ok Cancel

Figure 52: Final configuration of the first GPIO in node.

Let's say that we add a GPIO out node and configure it the same way as explained on section 4.1:

Edit gpio out n	node
🗱 Pin	Output pin O1
🗩 Message	msg. payload
	Invert output ?
Name	DO1
	Ok Cancel

Figure 53: Digital output configured.

Connect both nodes together and "Deploy" the configuration:



Figure 54: Final configuration for Analog input.

At this point we should see that the digital output 1 is ON all the time because the value of the Digital input is always higher than "1". The only way to turn the digital output 1 OFF is by pressing the digital input button, this is not directly related to the button itself, but because the analog input is going to have a reading of "0":



Figure 55: Analog input reading "0" after pressing the digital input.

00 PTION