

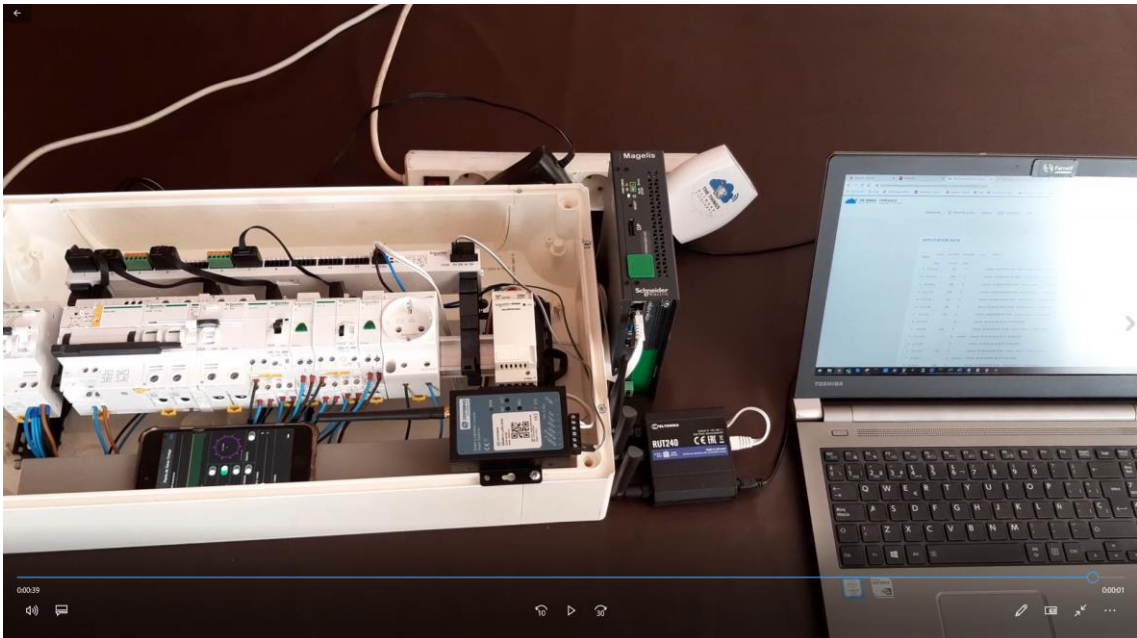
# SCHNEIDER SMART LINK TO LORAWAN

Monitor and control your cabinet remotely with no wires and with Dragino RS485-LN LoRaWAN technology

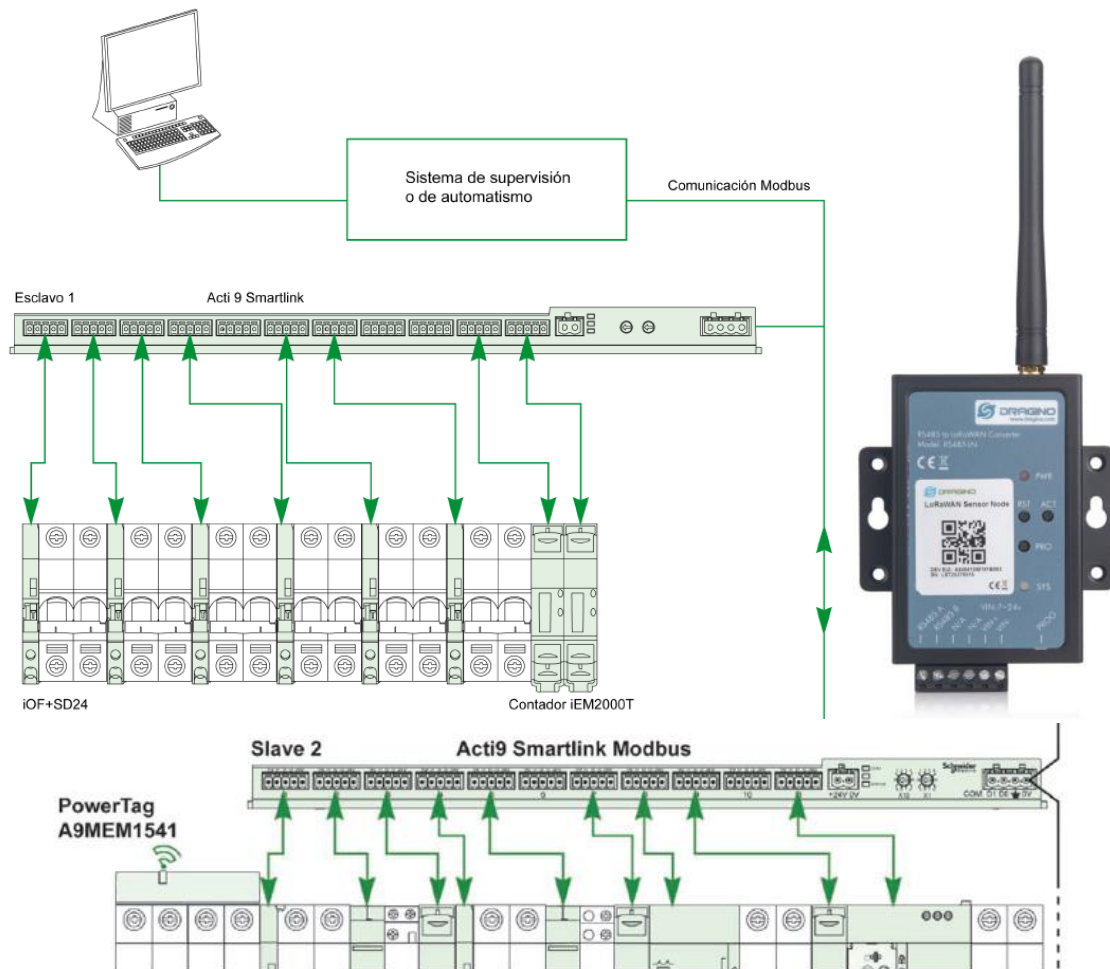
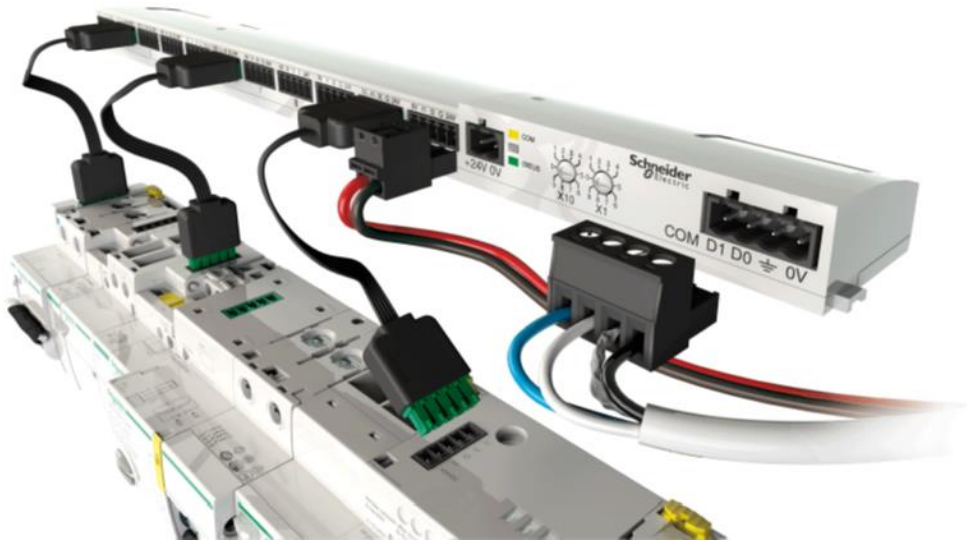
This is our workbench

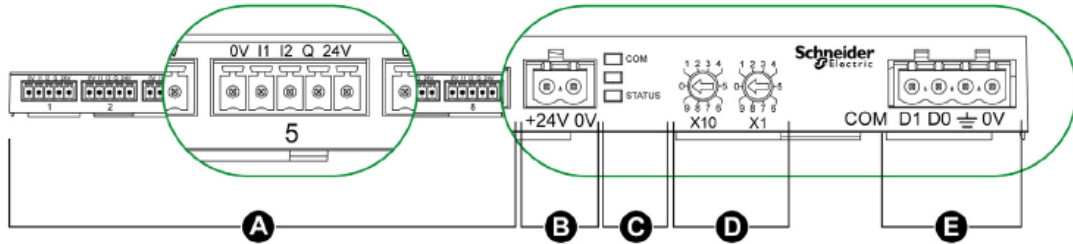


And this is the complete setup



This is Schneider Smartlink Modbus RTU system con connect and control protection cabinets





- A 11 canales de E/S
- B Un conector de alimentación eléctrica de 24 V de CC
- C LED que muestran el estado de funcionamiento del dispositivo Acti 9 Smartlink
- D 2 ruedas codificadoras para la dirección Modbus del dispositivo
- E Un conector Modbus de 4 patillas

## REGISTERING THE DRAGINO RS485-LN ON TTN

Applications > smartlink\_to\_lora > Devices > 87654321 > Data

Overview Data Settings

### APPLICATION DATA

Filters: uplink downlink activation ack error

time	counter	port	
▲ 13:00:05	1	2	payload: 01
▼ 12:50:07		0	
▲ 12:50:07	0	2	retry payload: 01
⚡ 12:49:57			dev addr: 26 01 2A 26 app eui: 15 F6 FF 6A 7C EF B4 78 dev eui: A8 40 41 2F E1 82 62 4B

## CHANGING LORA UPLINK PERIOD

Let's connect the control cable to a FDTI (USB to TTL converter) and start Termit terminal software on your PC

Default uplink period is 10 minutes

Use this setting on the Termit terminal

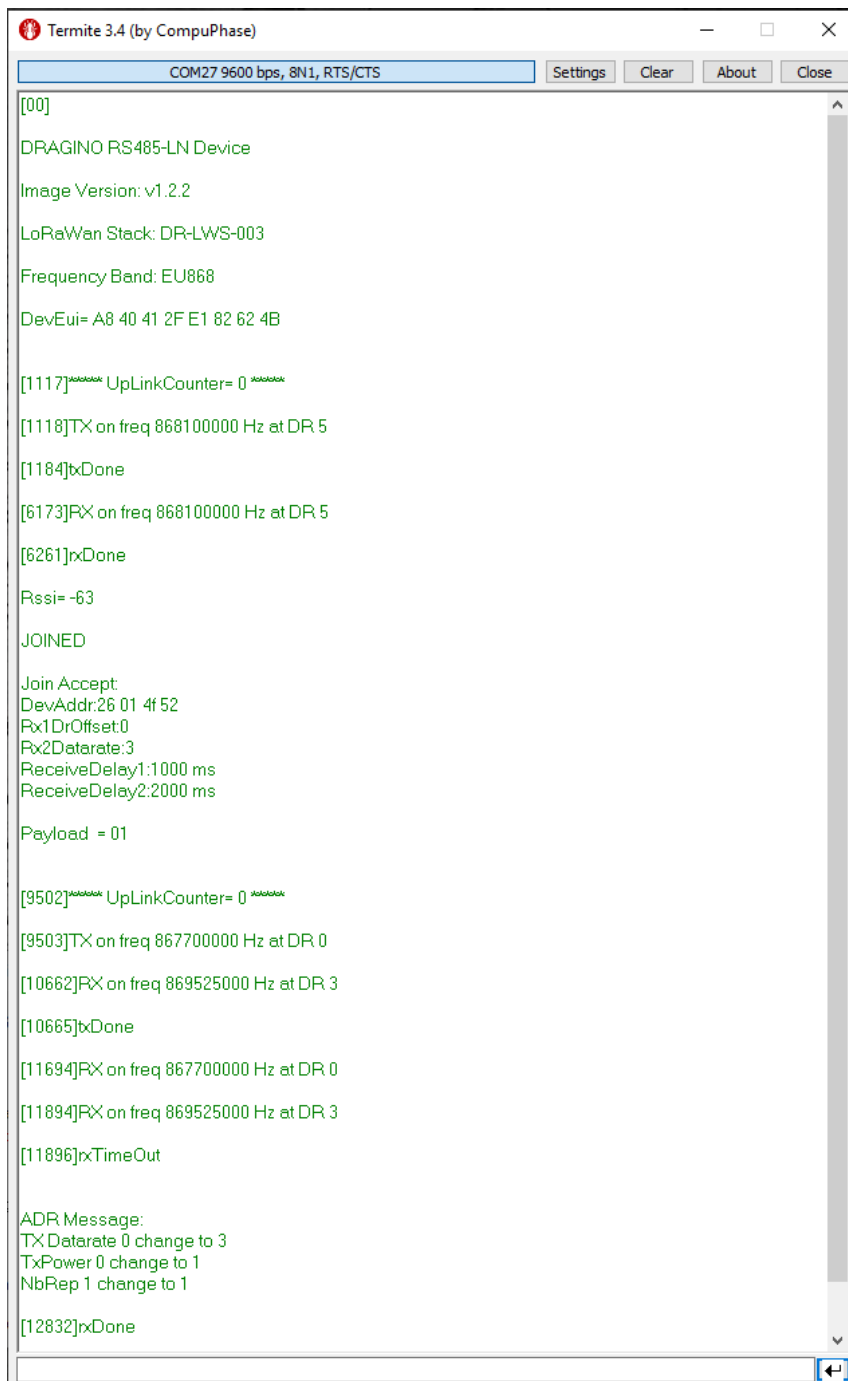
Serial port settings

<b>Port configuration</b> Port: COM27 Baud rate: 9600 Data bits: 8 Stop bits: 1 Parity: none Flow control: none Forward: none	<b>Transmitted text</b> <input type="radio"/> Append nothing <input checked="" type="radio"/> Append CR <input type="radio"/> Append LF <input type="radio"/> Append CR-LF <input checked="" type="checkbox"/> Local echo <b>Received text</b> Polling: 100 ms Max. lines: Font: default <input type="checkbox"/> Word wrap	<b>Options</b> <input type="checkbox"/> Stay on top <input checked="" type="checkbox"/> Quit on Escape <input checked="" type="checkbox"/> Autocomplete edit line <input checked="" type="checkbox"/> Keep history <input type="checkbox"/> Close port when inactive <b>Plug-ins</b>
--	---	--

User interface language: English (en) [Cancel] [OK]

You should see this screen.

If the screen is in blank, then unplug power OFF and ON on the Dragino RS485-LN

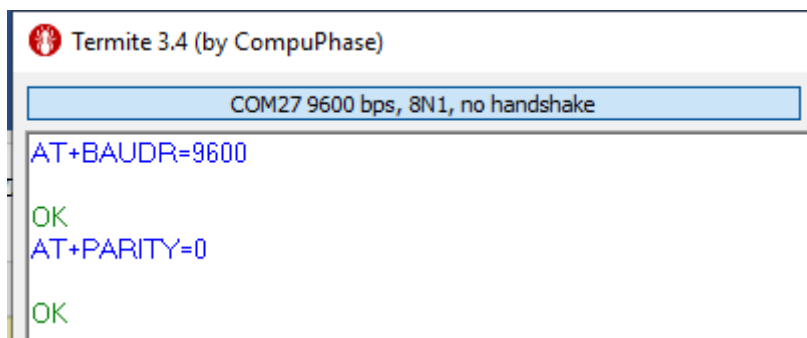


```
Termit 3.4 (by CompuPhase)
COM27 9600 bps, 8N1, RTS/CTS
Settings Clear About Close
[00]
DRAGINO RS485-LN Device
Image Version: v1.2.2
LoRaWan Stack: DR-LWS-003
Frequency Band: EU868
DevEui= A8 40 41 2F E1 82 62 4B

[1117]***** UpLinkCounter= 0 *****
[1118]TX on freq 868100000 Hz at DR 5
[1184]txDone
[6173]RX on freq 868100000 Hz at DR 5
[6261]rxDone
Rssi= -63
JOINED
Join Accept:
DevAddr:26 01 4f 52
Rx1DrOffset:0
Rx2Datarate:3
ReceiveDelay1:1000 ms
ReceiveDelay2:2000 ms
Payload = 01

[9502]***** UpLinkCounter= 0 *****
[9503]TX on freq 867700000 Hz at DR 0
[10662]RX on freq 869525000 Hz at DR 3
[10665]txDone
[11694]RX on freq 867700000 Hz at DR 0
[11894]RX on freq 869525000 Hz at DR 3
[11896]rxTimeOut
ADR Message:
TX Datarate 0 change to 3
TxPower 0 change to 1
NbRep 1 change to 1
[12832]rxDone
```

Let's send some commands to ensure we will have our Modbus settings correctly settled.



```
Termit 3.4 (by CompuPhase)
COM27 9600 bps, 8N1, no handshake
AT+BAUDR=9600
OK
AT+PARITY=0
OK
```

Now let's change the uplink period

Termite 3.4 (by CompuPhase)

```
COM27 9600 bps, 8N1, no handshake
AT+BAUDR=9600
OK
AT+PARITY=0
OK
AT+TDC=10000
OK
```

We see that it automatically reinitiates and changes are applied!

Applications > smartlink\_to\_lora > Devices > 87654321 > Data

Overview Data Settings

APPLICATION DATA || pause 🗑 clear

Filters uplink downlink activation ack error

time	counter	port	
13:16:05	2	2	payload: 01
13:15:55	1	2	payload: 01
13:05:57		0	
13:05:57	0	2	retry payload: 01
13:05:47			dev addr: 26 01 58 FB app eui: 15 F6 FF 6A 7C EF B4 78 dev eui: A8 40 41 2F E1 82 62 4B

Now we need to determine the right command to give to the Smart Link

First we connect a RS-485 converter to the PC and open qModMaster to test the dialogue between PC and Smart Link

And we want to read channel 2 leakage breaker mounted on Smart link

The unit ID is 01 (Slave node address) as set on the rotary dip switches

Modbus address to read the **channel 2** status is **14240** in decimal (37 A0 in Hex)

Description of terminals for each channel (Ti24 interface):

Terminal	Description
24 V	24 V of the 24 Vdc power supply
Q	Control output
I2	Input number 2
I1	Input number 1
0 V	0 V of the 24 Vdc power supply

### Status

	Channels						
	1	2	3	4	5	6	7
Input I1 (bit 0)	14200	14240	14280	14320	14360	14400	14440
Input I2 (bit 1)	14200	14240	14280	14320	14360	14400	14440

For Channel 1

Address	No.	RW	X	Unit	Type	Range	Default Value	Svd	Function Code	Description
14200	1	R	-	-	BITMAP	-	0x0000	N	03, 100-4	Electrical status of inputs 1 and 2 of all connected devices <sup>(1)</sup> .

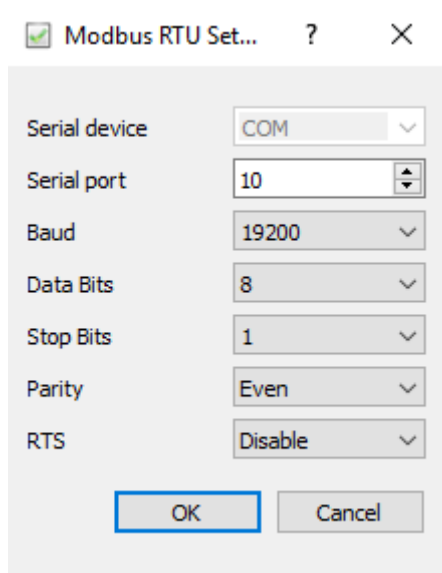
And these are the default Smartlink Modbus comm parameters

### Parámetros de comunicación

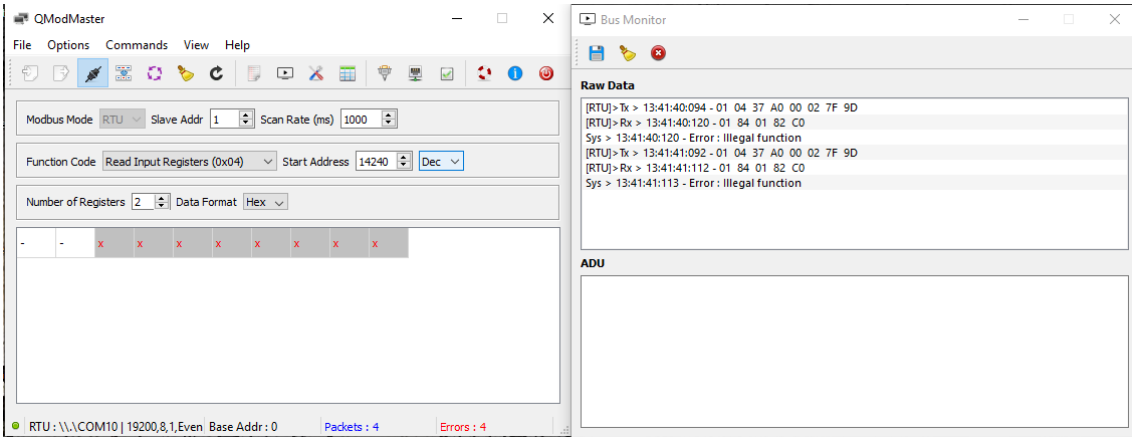
Los valores de los parámetros de comunicación son los siguientes:

Parámetros	Valores autorizados	Valor predeterminado
Velocidad de datos (en Baud)	4.800, 9.600 y 19.200	19.200
Paridad	<ul style="list-style-type: none"> <li>Par y un bit de parada</li> <li>Impar y un bit de parada</li> <li>Sin paridad (eliminación del bit de paridad), se necesitan 2 bits de parada.</li> </ul>	Par (con 1 bit de parada)

Let's assing these on qModMaster

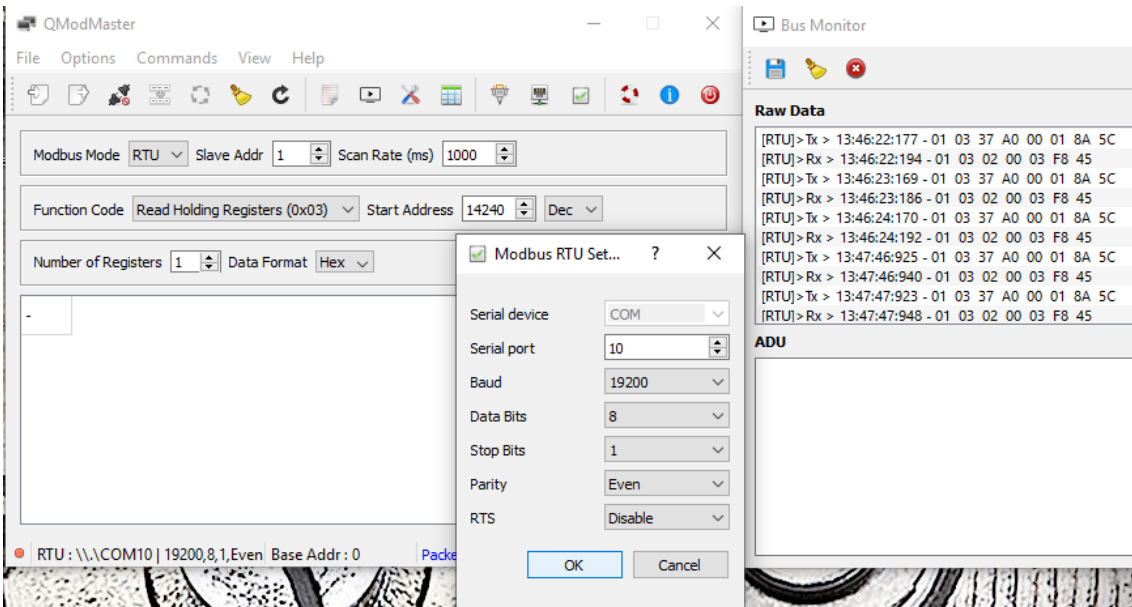


Something is wrong

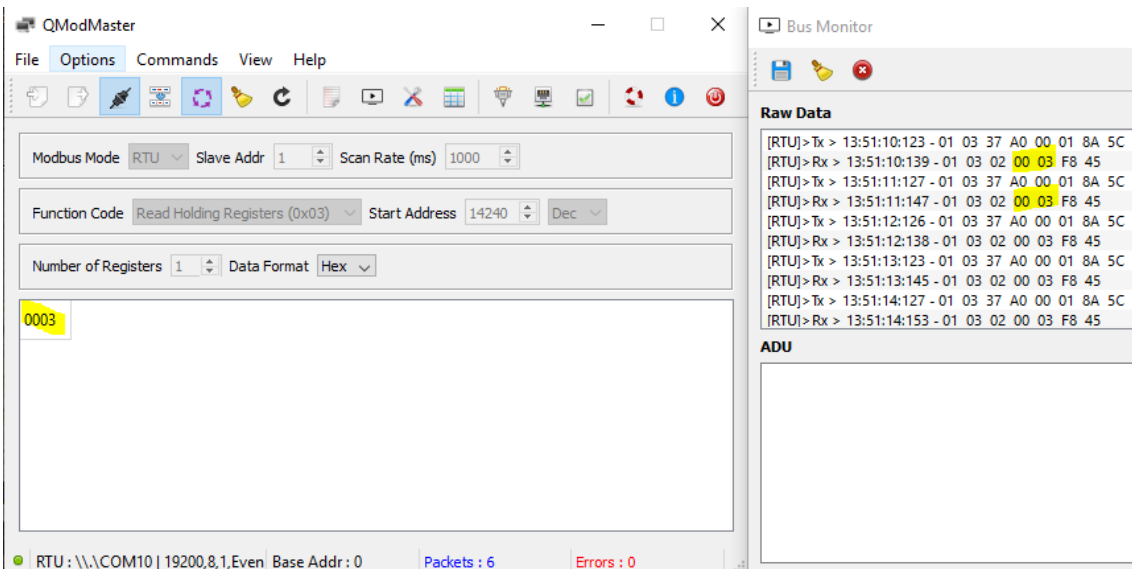


We try with function number 3 and only 1 register

Now it Works!



Let's check the ON status of leakage breaker



And now the OFF status of Leakage breaker

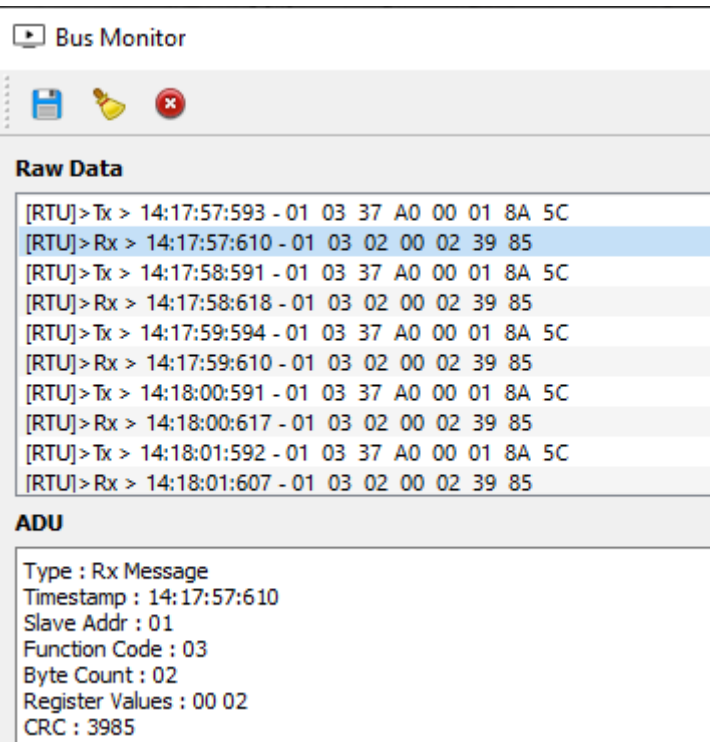
The screenshot shows the QModMaster software interface. The main window has a menu bar (File, Options, Commands, View, Help) and a toolbar. Below the toolbar are several configuration fields: Modbus Mode (RTU), Slave Addr (1), Scan Rate (ms) (1000), Function Code (Read Holding Registers (0x03)), Start Address (14240), Dec, Number of Registers (1), and Data Format (Hex). A large text area below these fields displays the value '0002'. At the bottom of the main window, a status bar shows 'RTU: \\.\COM10 | 19200,8,1,Even Base Addr: 0', 'Packets: 114', and 'Errors: 0'. To the right, a 'Bus Monitor' window is open, displaying a list of raw data packets under the heading 'Raw Data'. The packets are timestamped and show Tx and Rx data in hexadecimal. The status bar of the Bus Monitor window shows 'ADU'.

Transmit request message

The screenshot shows the Bus Monitor window. The 'Raw Data' section displays a list of packets. The first packet is highlighted in blue and is a Tx message: [RTU]>Tx > 14:17:57:593 - 01 03 37 A0 00 01 8A 5C. Below the 'Raw Data' section, the 'ADU' section provides a detailed view of the selected Tx message: Type : Tx Message, Timestamp : 14:17:57:593, Slave Addr : 01, Function Code : 03, Starting Address : 37A0, Quantity of Registers : 0001, CRC : 8A5C.

Response message





So the right commands are

AT+COMMAND1=01 03 37 A0 00 01,1 (or as downlink AF 01 01 06 01 03 37 A0 00 01 01)

AT+DATACUT1=7,2,4~5 (or as downlink AF 01 02 04 07 02 04 05 01)

Let's try

```
AT+COMMAND1=01 03 37 A0 00 01,1
```

```
OK
```

```
AT+DATACUT1=7,2,4~5
```

```
OK
```

```
CMD1 = 01 03 37 a0 00 01 8a 5c
RETURN1 = 00 00 00 00 00 00 00
Payload = 01 00 00
```

```
[4779429]***** UpLinkCounter= 424*****
```

```
[4779431]TX on freq 867700000 Hz at DR 5
```

```
[4779486]RX on freq 869525000 Hz at DR 3
```

```
[4779489]txDone
```

Our payload is higher now, but we still have to connect the Dragino to the Smartlink thru Modbus RTU cable

And we have to set this parameters on Dragino Modbus

### Parámetros de comunicación

Los valores de los parámetros de comunicación son los siguientes:

Parámetros	Valores autorizados	Valor predeterminado
Velocidad de datos (en Baud)	4.800, 9.600 y 19.200	19.200
Paridad	<ul style="list-style-type: none"><li>• Par y un bit de parada</li><li>• Impar y un bit de parada</li><li>• Sin paridad (eliminación del bit de paridad), se necesitan 2 bits de parada.</li></ul>	Par (con 1 bit de parada)

With Termitte

```
AT+BAUDR=19200
```

```
OK
```

```
AT+PARITY=1
```

```
OK
```

```
CMD1 = 01 03 37 a0 00 01 8a 5c  
RETURN1 = 00 00 00 00 00 00 00  
Payload = 01 00 00
```

```
AT+PARITY=2
```

```
OK
```

```
CMD1 = 01 03 37 a0 00 01 8a 5c  
RETURN1 = 00 00 00 00 00 00 00  
Payload = 01 00 00
```

Voilà this is the OFF state of leakage Breaker

```
CMD1 = 01 03 37 a0 00 01 8a 5c  
RETURN1 = 01 03 02 00 02 39 85  
Payload = 01 00 02
```

And this is the ON state of Leakage Breaker

```
CMD1 = 01 03 37 a0 00 01 8a 5c  
RETURN1 = 01 03 02 00 03 f8 45  
Payload = 01 00 03
```

Or we can see i ton TTN

## APPLICATION DATA

Filters

	time	counter	port	
▲	14:41:36	522	2	payload: 01 00 02
▲	14:41:26	521	2	payload: 01 00 02
▲	14:41:16	520	2	payload: 01 00 03

Now let's go to node-red to send the status to the Mobile phone IoT OnOff App

But first let's change the payload decoder to be able to get a readable value from TTN

## PAYLOAD FORMATS

### Payload Format

The payload format sent by your devices

Custom

```
1 function Decoder(bytes, port) {  
2   // Decode an uplink message from a buffer  
3   // (array) of bytes to an object of fields.  
4   var decoded = {};  
5  
6   if (port === 2) decoded.breaker_status = bytes[1]*256+bytes[2];  
7  
8   return decoded;  
9 }
```

▲ 14:53:16	593	2	dev id: <a href="#">87654321</a>	payload: 01 00 02	breaker_status: 2
▲ 14:53:06	592	2	dev id: <a href="#">87654321</a>	payload: 01 00 02	breaker_status: 2
▲ 14:52:55	591	2	dev id: <a href="#">87654321</a>	payload: 01 00 02	breaker_status: 2

egram / Pusi | ATV320 | Dragino VFD | SMARTLINK TO LORA

**debug**

```

17/10/2020 14:55:04 node: 25818b30.58eb14
msg.payload : Object
  ▶ { breaker_status: 3 }
17/10/2020 14:55:04 node: 39dfcae.c1be236
msg.payload : number
3
17/10/2020 14:55:15 node: 25818b30.58eb14
msg.payload : Object
  ▶ { breaker_status: 2 }
17/10/2020 14:55:15 node: 39dfcae.c1be236
msg.payload : number
2

```

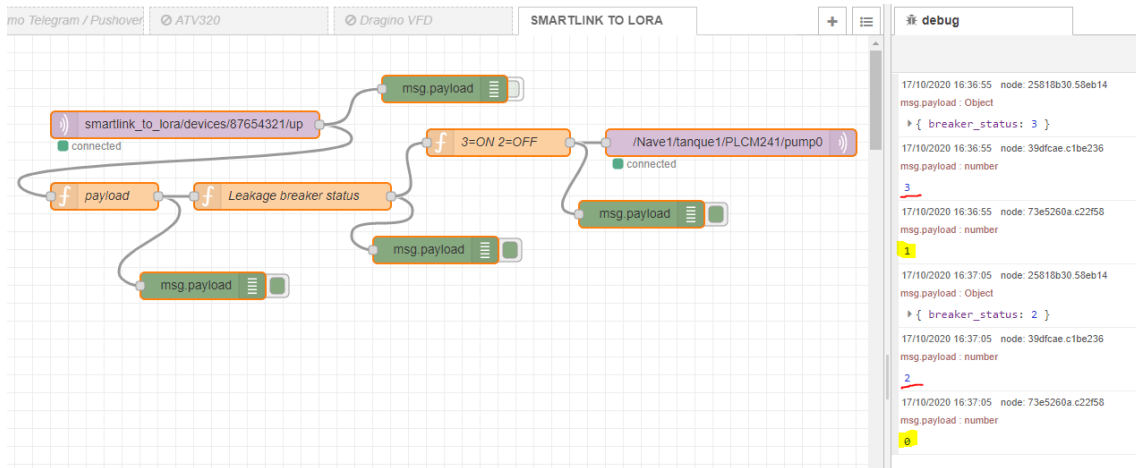
**debug**

```

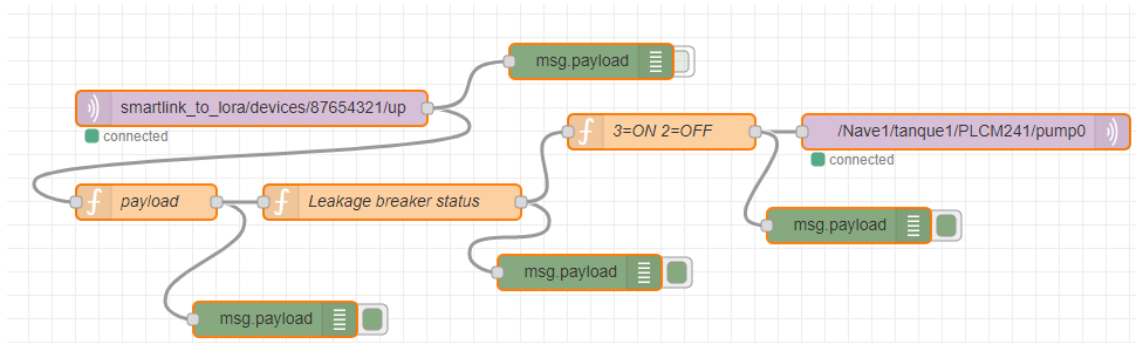
17/10/2020 14:56:05 node: 25818b30.58eb14
msg.payload : Object
  ▶ { breaker_status: 2 }
17/10/2020 14:56:05 node: 39dfcae.c1be236
msg.payload : number
2
17/10/2020 14:56:15 node: 25818b30.58eb14
msg.payload : Object
  ▶ { breaker_status: 3 }
17/10/2020 14:56:15 node: 39dfcae.c1be236
msg.payload : number
3

```

Now let's get a simple value 0 or 1 and send it to MQTT






Yes it is working!






And these are the configured node-red nodes



### Edit mqtt in node



Delete Cancel Done


**Properties**   

 Server eu.thethings.network:1883 

 Topic smartlink\_to\_lora/devices/87654321/up



 QoS 2 


 Output auto-detect (string or buffer) 

 Name Name


### Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update


**Properties**  

 Name Name

**Connection** **Security** **Messages**

 Server eu.thethings.network Port 1883

Enable secure (SSL/TLS) connection

 Client ID Leave blank for auto generated

Keep alive time (s) 60  Use clean session

Use legacy MQTT 3.1 support

Edit mqtt in node > **Edit mqtt-broker node**

Delete Cancel **Update**

**Properties** ⚙️ 📄

🔑 Name

Connection **Security** Messages

👤 Username

🔒 Password

**Edit function node**

Delete Cancel **Done**

**Properties** ⚙️ 📄 🖨️

🔑 Name  📄 ▼

🔧 Function ↗️

```

1 var msg1 = { payload: msg.payload.length };
2 msg1.payload = JSON.parse(msg.payload);
3 msg1.payload = msg1.payload.payload_fields;
4
5 return msg1;

```

**Edit function node**

Delete Cancel **Done**

**Properties** ⚙️ 📄 🖨️

🔑 Name  📄 ▼

🔧 Function ↗️

```

1 var a = msg.payload;
2 msg.payload=a.breaker_status;
3 return msg;

```

### Edit function node

Delete Cancel Done

**Properties**

Name 3=ON 2=OFF

Function

```
1 var a=msg.payload;
2 if (a==3)
3 {msg.payload=1}
4 else if (a==2)
5 {msg.payload=0}
6 return msg;
```

### Edit mqtt out node

Delete Cancel Done

**Properties**

Server broker.hivemq.com:1883

Topic /Nave1/tanque1/PLCM241/pump0

QoS Retain

Name Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

You can find the code here

<https://github.com/xavierflorensa/Schneider-SmartLink-to-LoRaWAN/tree/main>



## TWO CHANNELS



Let's try this remote control breaker

Now we will configure the RS485-LN to send the status of two breaker son same message uplink

We already know the right command

We want t oread the status of breaker on channel 1

Description of terminals for each channel (Ti24 interface):

Terminal	Description
24 V	24 V of the 24 Vdc power supply
Q	Control output
I2	Input number 2
I1	Input number 1
0 V	0 V of the 24 Vdc power supply

### Status

	Channels						
	1	2	3	4	5	6	7
Input I1 (bit 0)	14200	14240	14280	14320	14360	14400	14440
Input I2 (bit 1)	14200	14240	14280	14320	14360	14400	14440

For Channel 1

Address	No.	RW	X	Unit	Type	Range	Default Value	Svd	Function Code	Description
14200	1	R	-	-	BITMAP	-	0x0000	N	03, 100-4	Electrical status of inputs 1 and 2 of all connected devices <sup>(1)</sup> .

Channel 1 has address 14200 which is 37 78 in Hex

So the right command is

AT+COMMAND2=01 03 37 78 00 01,1

AT+DATACUT2=7,2,4~5

Let's try on Termite Terminal

```
AT+COMMAND2=01 03 37 78 00 01,1
```

```
OK
```

```
AT+DATA CUT2=7,2,4~5
```

```
OK
```

```
CMD1 = 01 03 37 a0 00 01 8a 5c  
RETURN1 = 01 03 02 00 03 f8 45  
CMD2 = 01 03 37 78 00 01 0a 67  
RETURN2 = 01 03 02 00 02 39 85  
Payload = 01 00 03 00 02
```

Now the Payload has two channels

On TTN

Applications > smartlink\_to\_lora > Data

Overview Devices Payload Formats Integrations Data Settings

**APPLICATION DATA** || pause 🗑 clear

Filters:

time	counter	port	dev id	payload	breaker_status
17:14:46	1456	2	<a href="#">87654321</a>	01 00 02 00 02	2
17:14:36	1455	2	<a href="#">87654321</a>	01 00 02 00 02	2
17:14:25	1454	2	<a href="#">87654321</a>	01 00 02	2
17:14:15	1453	2	<a href="#">87654321</a>	01 00 02	2

But we have to adjust the payload decoder in order to make the message readable

## PAYLOAD FORMATS

### Payload Format

The payload format sent by your devices

Custom

decoder converter validator encoder

```
1 function Decoder(bytes, port) {  
2   // Decode an uplink message from a buffer  
3   // (array) of bytes to an object of fields.  
4   var decoded = {};  
5  
6   if (port === 2) decoded.breaker_status = bytes[1]*256+bytes[2];  
7   decoded.remote_status = bytes[3]*256+bytes[4];  
8  
9   return decoded;  
10 }
```

Applications > smartlink\_to\_lora > Data

Overview Devices Payload Formats Integration

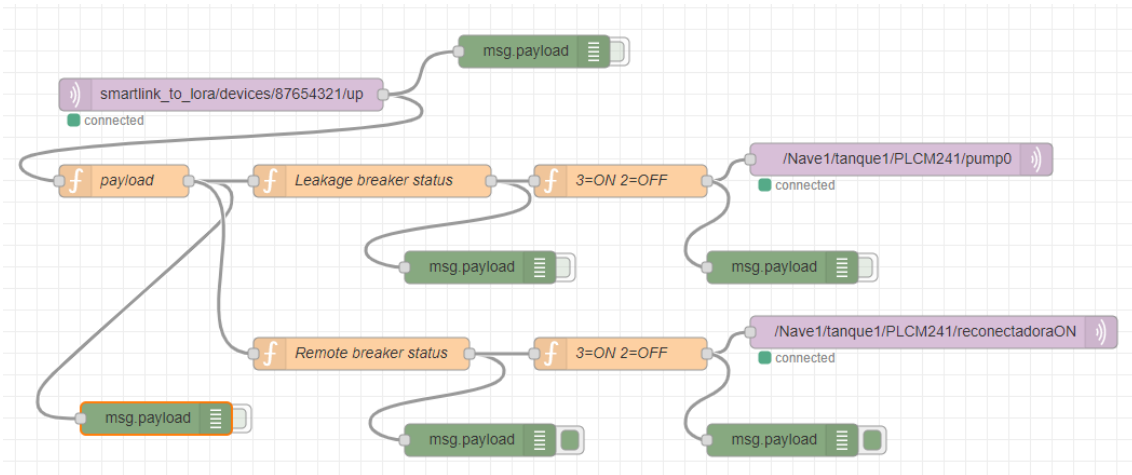
## APPLICATION DATA

Filters

uplink downlink activation ack error

	time	counter	port				
▲	18:47:46	2022	2	dev id: <a href="#">87654321</a>	payload: 01 00 03 00 03	breaker_status: 3	remote_status: 3
▲	18:47:36	2021	2	dev id: <a href="#">87654321</a>	payload: 01 00 03 00 03	breaker_status: 3	remote_status: 3
▲	18:47:25	2020	2	dev id: <a href="#">87654321</a>	payload: 01 00 03 00 03	breaker_status: 3	
▲	18:47:15	2019	2	dev id: <a href="#">87654321</a>	payload: 01 00 03 00 03	breaker_status: 3	

Now we can modify the node-red Flow as before to send the data to the mobile phone



## REMOTE CONTROL

Now we want not just read, but also change the status of the remote controlled circuit breaker

We will use the TTN downlink ,

But first let's test with the PC connected to the Smartlink

We want to write on a register to perform an order on this way

### Orders

	Channels						
	1	2	3	4	5	6	7
Output Q (bit 0 and bit 1): Acti9 product	14201	14241	14281	14321	14361	14401	14441

For Channel 1

Address	No.	RW	X	Unit	Type	Range	Default Value	Svd	Function Code	Description
14201	1	RW	-	-	BITMAP	-	0x0000	N	03, 06, 16, 100-4	Close and open order for products in the Acti9 range <sup>(1)</sup> .

(1)

- Bit 0 = open order
- Bit 1 = close order
- Bits 2 to 15 = no meaning

So since we have the remote circuit breaker on channel 1 we need to write on address 14201, and we have to use function number FC6 Preset single register

To close the circuit:

**QModMaster**

File Options Commands View Help

Modbus Mode: RTU Slave Addr: 1 Scan Rate (ms): 1000

Function Code: Write Single Register (0x06) Start Address: 14201 Dec

Number of Registers: 1 Data Format: Bin

0000000000000010

**Bus Monitor**

**Raw Data**

```
[RTU]>Tx > 20:11:55:546 - 01 06 37 79 00 02 D7 A6
[RTU]>Rx > 20:11:55:570 - 01 06 37 79 00 02 D7 A6
Sys > 20:11:55:570 - values written correctly.
[RTU]>Tx > 20:11:56:545 - 01 06 37 79 00 02 D7 A6
[RTU]>Rx > 20:11:56:561 - 01 06 37 79 00 02 D7 A6
Sys > 20:11:56:561 - values written correctly.
[RTU]>Tx > 20:11:57:547 - 01 06 37 79 00 02 D7 A6
[RTU]>Rx > 20:11:57:569 - 01 06 37 79 00 02 D7 A6
Sys > 20:11:57:569 - values written correctly.
[RTU]>Tx > 20:11:58:081 - 01 03 37 79 00 01 5B A7
```

**ADU**

To open the circuit:

**QModMaster**

File Options Commands View Help

Modbus Mode: RTU Slave Addr: 1 Scan Rate (ms): 1000

Function Code: Write Single Register (0x06) Start Address: 14201 Dec

Number of Registers: 1 Data Format: Bin

0000000000000001

**Bus Monitor**

**Raw Data**

```
[RTU]>Tx > 20:14:27:502 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:27:523 - 01 06 37 79 00 01 97 A7
Sys > 20:14:27:523 - values written correctly.
[RTU]>Tx > 20:14:28:499 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:28:514 - 01 06 37 79 00 01 97 A7
Sys > 20:14:28:514 - values written correctly.
[RTU]>Tx > 20:14:29:502 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:29:521 - 01 06 37 79 00 01 97 A7
Sys > 20:14:29:521 - values written correctly.
[RTU]>Tx > 20:14:30:214 - 01 03 37 79 00 01 5B A7
```

**ADU**

RTU: \\.\COM10 | 19200,8,1,Even | Base Addr: 0 | Packets: 35 | Errors: 0

And it Works!

And the telegram:

**Bus Monitor**

**Raw Data**

```
[RTU]>Tx > 20:14:27:502 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:27:523 - 01 06 37 79 00 01 97 A7
Sys > 20:14:27:523 - values written correctly.
[RTU]>Tx > 20:14:28:499 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:28:514 - 01 06 37 79 00 01 97 A7
Sys > 20:14:28:514 - values written correctly.
[RTU]>Tx > 20:14:29:502 - 01 06 37 79 00 01 97 A7
[RTU]>Rx > 20:14:29:521 - 01 06 37 79 00 01 97 A7
Sys > 20:14:29:521 - values written correctly.
[RTU]>Tx > 20:14:30:214 - 01 03 37 79 00 01 5B A7
```

**ADU**

Type : Tx Message  
 Timestamp : 20:14:27:502  
 Slave Addr : 01  
 Function Code : 06  
 Starting Address : 3779  
 Output Value : 0001  
 CRC : 97A7

So the right downlink value would be (We will use command 3 not to interfere with the others):

To Close the circuit

AF 03 01 06 01 06 37 79 00 02 00 and the datacut AF 03 02 04 07 02 04 05 00

To open the circuit

AF 03 01 06 01 06 37 79 00 01 00 and the datacut AF 03 02 04 07 02 04 05 00

You have to add the datacut at the end of the downlink if you want the command to be updated

### Type Code 0xAF

0xAF downlink command can be used to set AT+COMMANDx or AT+DATACUTx.

Note: if user use AT+COMMANDx to add a new command, he also need to send AT+DATACUTx downlink.

Format: AF MM NN LL XX XX XX XX YY

Where:

- ✧ MM: the ATCOMMAND or AT+DATACUT to be set. Value from 01 ~ 0F,
- ✧ NN: 0: no CRC; 1: add CRC-16/MODBUS ; 2: set the AT+DATACUT value.
- ✧ LL: The length of AT+COMMAND or AT+DATACUT command
- ✧ XX XX XX XX: AT+COMMAND or AT+DATACUT command
- ✧ YY: If YY=0, RS485-LN will execute the downlink command without uplink; if YY=1, RS485-LN will execute an uplink after got this command.

Example:

AF 03 01 06 0A 05 00 04 00 01 00: Same as AT+COMMAND3=0A 05 00 04 00 01,1

Now we connect the Dragino to the SmartLink thru Modbus RTU cable, and try:

In order to set a command3 you have to send also a datacut (at least first time)

**DOWNLINK**

**Scheduling**

replace first last

**FPort**

**Confirmed**

**Payload**

bytes fields  11 bytes

Send

If you do not send this second instruction it will not work

**DOWNLINK**

**Scheduling**

replace
first
last

**FPort**

**Confirmed**

**Payload**

bytes
fields

AF 03 02 04 07 02 04 05 0d
9 bytes

Send

After first datacut then you do not need to send a datacut after powering off. Unless you programm it with AT commands on the start up.

```

CMD1  = 01 03 37 a0 00 01 8a 5c
RETURN1 = 01 03 02 00 02 39 85
CMD2  = 01 03 37 78 00 01 0a 67
RETURN2 = 01 03 02 00 03 f8 45
Payload = 01 00 02 00 03

```

```
[3340092]***** UpLinkCounter= 326 *****
```

```
[3340094]TX on freq 867900000 Hz at DR 5
```

```
[3340149]RX on freq 869525000 Hz at DR 3
```

```
[3340152]txDone
```

```
[3341140]RX on freq 867900000 Hz at DR 5
```

```
[3341208]rxDone
```

```
Rssi= -57
```

```
Receive data
```

```
1:af 03 02 04 07 02 04 05 00
```

```

CMD1  = 01 03 37 a0 00 01 8a 5c
RETURN1 = 01 03 02 00 02 39 85
CMD2  = 01 03 37 78 00 01 0a 67
RETURN2 = 01 03 02 00 03 f8 45
CMD3  = 01 06 37 79 00 02 d7 a6
RETURN3 = 01 06 37 79 00 02 d7
Payload = 01 00 02 00 03 79 00

```

And it Works j! The remote breaker is now closed

```
CMD1 = 01 03 37 a0 00 01 8a 5c
RETURN1 = 01 03 02 00 02 39 85
CMD2 = 01 03 37 78 00 01 0a 67
RETURN2 = 01 03 02 00 03 f8 45
CMD3 = 01 06 37 79 00 02 d7 a6
RETURN3 = 01 06 37 79 00 02 d7
Payload = 01 00 02 00 03 79 00
```

Indeed we do not need a response for command 3

If we want to open the breaker (here we do not need the datacut command, is only the first time)

**DOWNLINK**

**Scheduling**    **FPort**   **Confirmed**

**Payload**    11 bytes

Now let's close again

**DOWNLINK**

**Scheduling**    **FPort**   **Confirmed**

**Payload**    11 bytes

Here is the result first with a open command (marked in yellow) and a close command (marked in red)



## APPLICATION DATA

|| pause  clear

Filters

	time	counter	port	
▲	11:02:23	845	2	payload: 01 00 02 00 03 79 00 breaker_status: 2 remote_status: 3
▲	11:02:13	844	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▼	11:02:02		1	payload: AF 03 01 06 01 06 37 79 00 02 00
▲	11:02:03	843	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▼	11:01:54		1 <i>scheduled</i>	payload: AF 03 01 06 01 06 37 79 00 02 00
▲	11:01:53	842	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▲	11:01:43	841	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▲	11:01:33	840	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▲	11:01:23	839	2	payload: 01 00 02 00 02 79 00 breaker_status: 2 remote_status: 2
▲	11:01:13	838	2	payload: 01 00 02 00 03 79 00 breaker_status: 2 remote_status: 3
▼	11:01:02		1	payload: AF 03 01 06 01 06 37 79 00 01 00
▲	11:01:03	837	2	payload: 01 00 02 00 03 79 00 breaker_status: 2 remote_status: 3

But the status changes after next uplink (no problem if we know that)

So it Works!

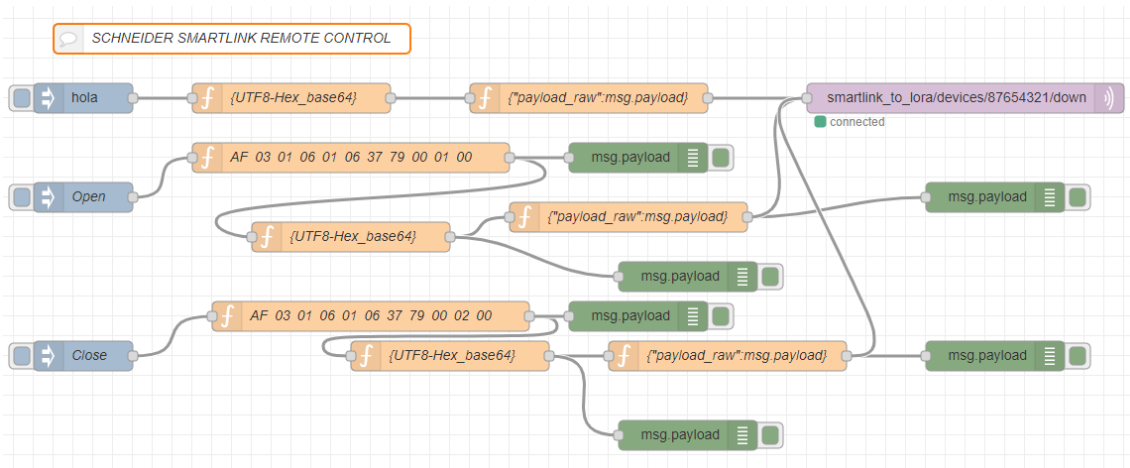
Now let's do it from node-red and later on from the mobile App

No matter the port used to send the downlink, Smartlink will react on same way

So we try sending the command on the downlink

First we prepare the command as a buffer on node-red then it must be converted to Base64 then format as "payload\_raw:"my encoded to Base64 command"

This way



## And it Works

```

18/10/2020 15:05:34 node: 82690894.a697a8
msg.payload : array[11]
  ▶ [ 175, 3, 1, 6, 1, 6, 55, 121, 0, 2 ... ]
18/10/2020 15:05:34 node: 3f193b4a.e1adf4
msg.payload : string[16]
  "rwMBBgEGN3kAAgA="
18/10/2020 15:05:34 node: 7ad08e23.5b9cd
msg.payload : Object
  ▶ { payload_raw: "rwMBBgEGN3kAAgA=" }
18/10/2020 15:06:06 node: abae706.cfe1f9
msg.payload : array[11]
  ▶ [ 175, 3, 1, 6, 1, 6, 55, 121, 0, 1 ... ]
18/10/2020 15:06:06 node: 683f8446.b2465c
msg.payload : string[16]
  "rwMBBgEGN3kAAQA="
18/10/2020 15:06:06 node: 8c3d7db.82b4b8
msg.payload : Object
  ▶ { payload_raw: "rwMBBgEGN3kAAQA=" }
18/10/2020 15:06:42 node: 82690894.a697a8
msg.payload : array[11]
  ▶ [ 175, 3, 1, 6, 1, 6, 55, 121, 0, 2 ... ]
18/10/2020 15:06:42 node: 3f193b4a.e1adf4
msg.payload : string[16]
  "rwMBBgEGN3kAAgA="

```

## APPLICATION DATA

Filters: uplink downlink activation ack error

	time	counter	port		payload	breaker_status	remote_status
▲	15:07:11	683	2		01 00 00 00 03 79 00	0	3
▲	15:07:02	682	2		01 00 00 00 02 79 00	0	2
▼	15:06:50		0		AF 03 01 06 01 06 37 79 00 02 00		
▲	15:06:52	681	2		01 00 00 00 02 79 00	0	2
▼	15:06:42		0	<i>scheduled</i>	AF 03 01 06 01 06 37 79 00 02 00		
▲	15:06:42	680	2		01 00 00 00 02 79 00	0	2
▲	15:06:32	679	2		01 00 00 00 02 79 00	0	2
▲	15:06:22	678	2		01 00 00 00 03 79 00	0	3
▼	15:06:10		0		AF 03 01 06 01 06 37 79 00 01 00		
▲	15:06:12	677	2		01 00 00 00 03 79 00	0	3
▼	15:06:06		0	<i>scheduled</i>	AF 03 01 06 01 06 37 79 00 01 00		
▲	15:06:02	676	2		01 00 00 00 03 79 00	0	3
▲	15:05:52	675	2		01 00 00 00 02 79 00	0	2

These are the configured node-red nodes

For the "Open" Flow

The screenshot shows the 'Edit function node' dialog in Node-RED. It includes a 'Delete' button, 'Cancel' and 'Done' buttons, and a 'Properties' section. The 'Name' field contains the hex string 'AF 03 01 06 01 06 37 79 00 01 00'. The 'Function' field contains the following JavaScript code:

```
1 var b = new Buffer(11);
2 b=[0xAF,0x03,0x01,0x06,0x01,0x06,0x37,0x79,0x00,0x01,0x00];
3 msg.payload = b;
4 return msg;
```

### Edit function node

Delete Cancel Done

**Properties**

Name {UTF8-Hex\_base64}

Function

```

1 var b = new Buffer(msg.payload);
2 msg.payload = b.toString('base64');
3 return msg;

```

### Edit function node

Delete Cancel Done

**Properties**

Name {"payload\_raw":msg.payload}

Function

```

1 msg.payload = {"payload_raw":msg.payload}
2 return msg;

```

### Edit mqtt out node

Delete Cancel Done

**Properties**

Server eu.thethings.network:1883

Topic smartlink\_to\_lora/devices/87654321/down

QoS Retain

Name Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

For the Close Flow just change 1 to 2 on the first node

**Edit function node**

Delete Cancel **Done**

**Properties**

Name

Function

```

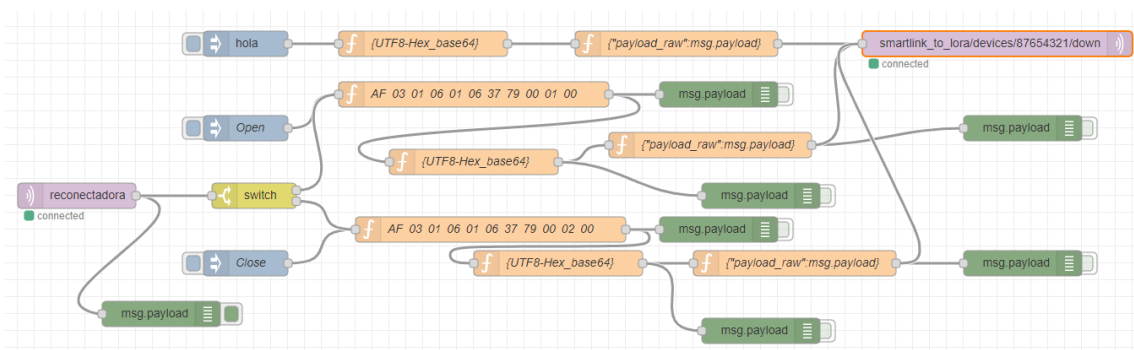
1 var b = new Buffer(11);
2 //b[0]=0xAF;
3 //b[1]=3;
4 b=[0xAF,0x03,0x01,0x06,0x01,0x06,0x37,0x79,0x00,0x02,0x00];
5 msg.payload = b;
6 return msg;

```

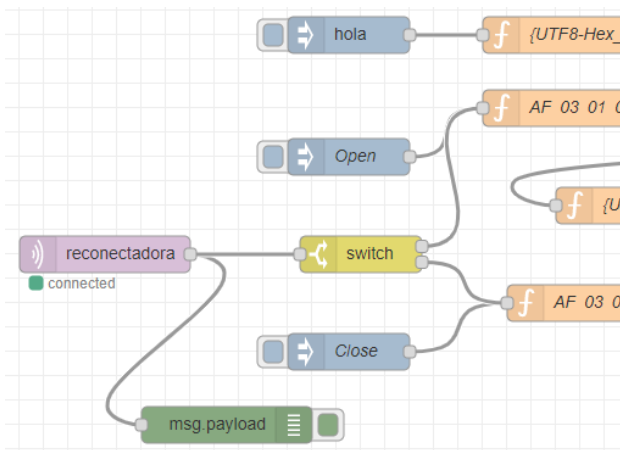
Now let's do it from Mobile Phone,

This is so easy with MQTT

## MOBILE PHONE



Just place a MQTT node to get the orders from the mobile phone IoT App

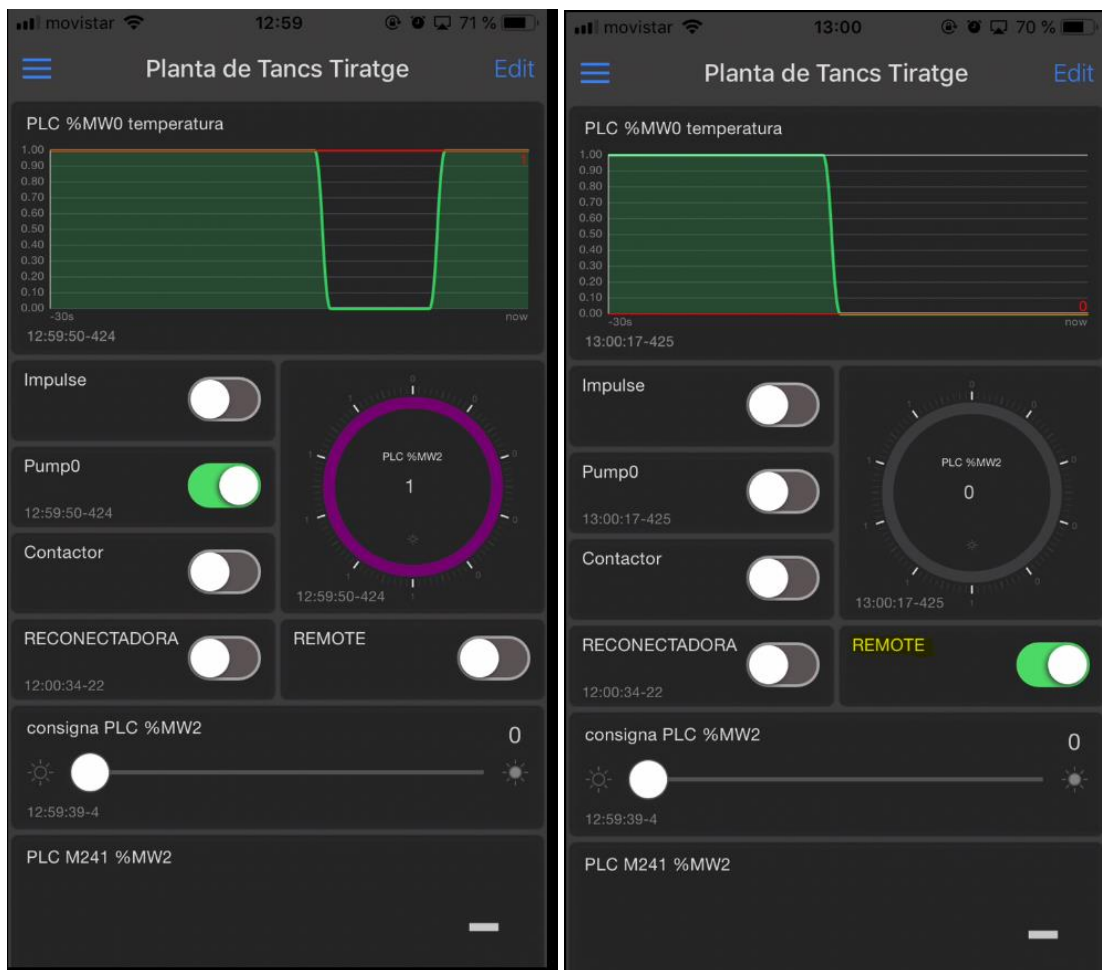


You can see the process on this video:

[https://www.youtube.com/watch?v=7UQAFyWkfhA&ab\\_channel=XavierFlorensaBerenguer](https://www.youtube.com/watch?v=7UQAFyWkfhA&ab_channel=XavierFlorensaBerenguer)

You can find the code here




<https://github.com/xavierflorensa/Schneider-SmartLink-to-LoRaWAN>




And this is how these new nodes are configured

### Edit mqtt in node

Delete Cancel Done

⚙️ Properties   

🌐 Server  

📄 Topic




⊕ QoS

➡ Output

👤 Name

### Edit switch node

Delete Cancel Done

⚙️ Properties   

👤 Name

⋮ Property

≡	==	▼	▼ 0 <sub>9</sub> 0	→ 1	✕
≡	==	▼	▼ a <sub>z</sub> 1	→ 2	✕

And that's all! Stay tuned