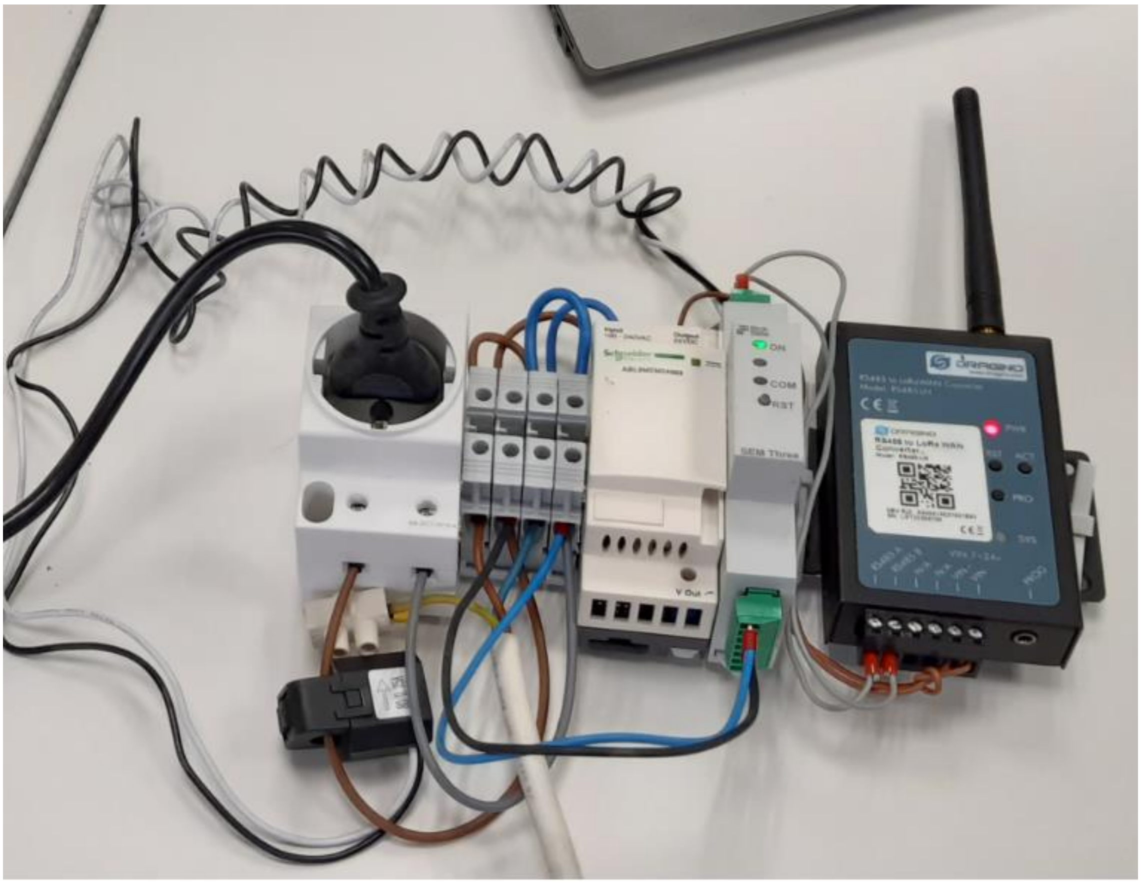
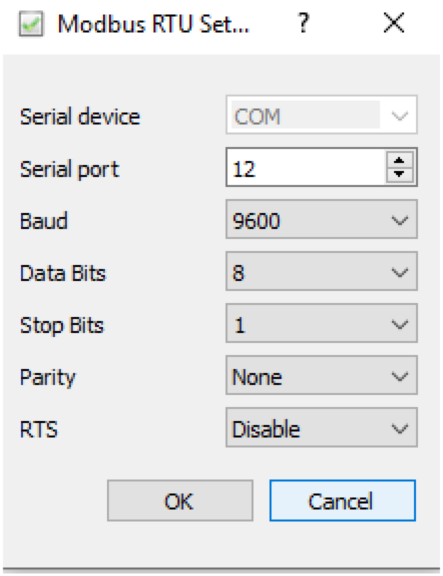
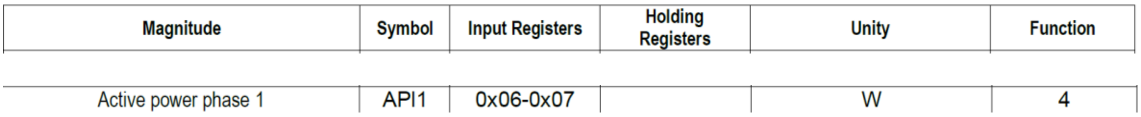
SEM three and Dragino RS-485



Use this Settings on QModMaster to validate and configure the Energy meter

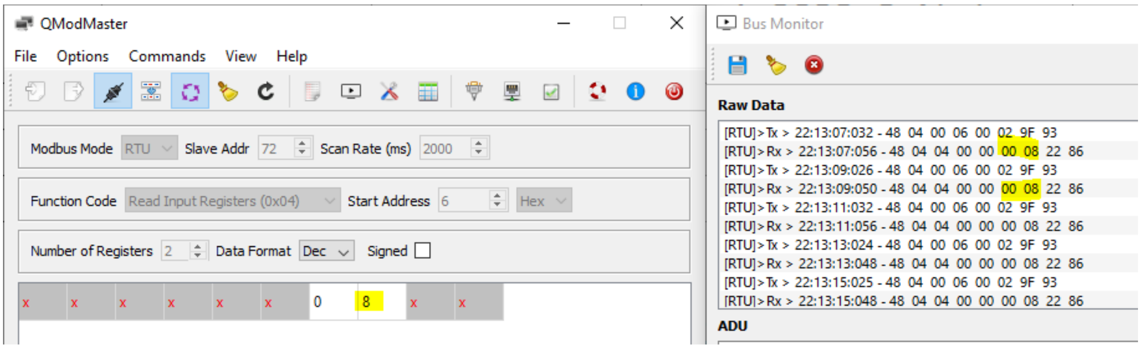


Serial device	COM
Serial port	12
Baud	9600
Data Bits	8
Stop Bits	1
Parity	None
RTS	Disable



Magnitude	Symbol	Input Registers	Holding Registers	Unity	Function
Active power phase 1	API1	0x06-0x07		W	4

Measuring power: 8 Watts at 230V



Modbus Mode RTU Slave Addr 72 Scan Rate (ms) 2000

Function Code Read Input Registers (0x04) Start Address 6 Hex

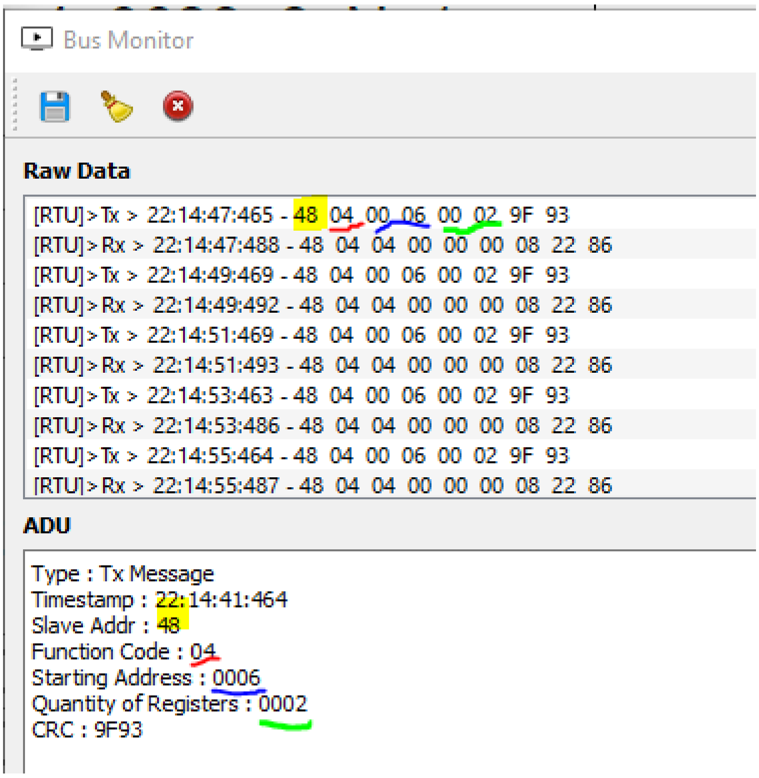
Number of Registers 2 Data Format Dec Signed ☐

x x x x x x 0 8 x x

Raw Data

```
[RTU]>Tx > 22:13:07:032 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:13:07:056 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:13:09:026 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:13:09:050 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:13:11:032 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:13:11:056 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:13:13:024 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:13:13:048 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:13:15:025 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:13:15:048 - 48  04  04  00  00  00  08  22  86
```

ADU

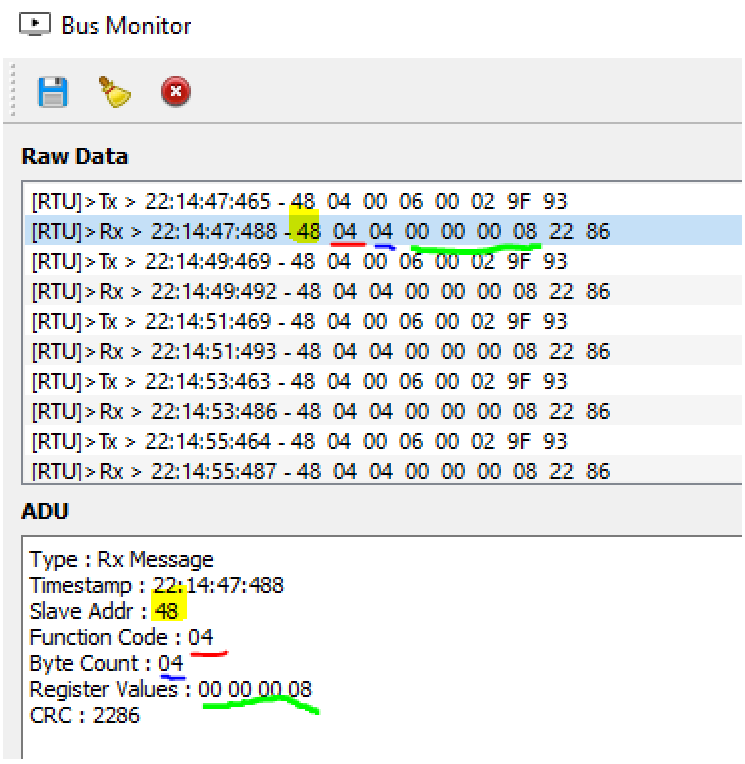


Raw Data

```
[RTU]>Tx > 22:14:47:465 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:47:488 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:49:469 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:49:492 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:51:469 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:51:493 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:53:463 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:53:486 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:55:464 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:55:487 - 48  04  04  00  00  00  08  22  86
```

ADU

```
Type : Tx Message
Timestamp : 22:14:41:464
Slave Addr : 48
Function Code : 04
Starting Address : 0006
Quantity of Registers : 0002
CRC : 9F93
```



```
Raw Data
[RTU]>Tx > 22:14:47:465 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:47:488 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:49:469 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:49:492 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:51:469 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:51:493 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:53:463 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:53:486 - 48  04  04  00  00  00  08  22  86
[RTU]>Tx > 22:14:55:464 - 48  04  00  06  00  02  9F  93
[RTU]>Rx > 22:14:55:487 - 48  04  04  00  00  00  08  22  86
ADU
Type : Rx Message
Timestamp : 22:14:47:488
Slave Addr : 48
Function Code : 04
Byte Count : 04
Register Values : 00 00 00 08
CRC : 2286
```

So the right command for the Dragino RS485-LN are

Active Power Phase 1

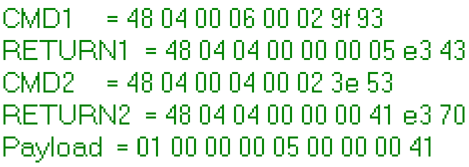
AT+COMMAND1=48 04 00 06 00 02,1

AT+DATACUT1=9,1,4+5+6+7

Current Phase 1

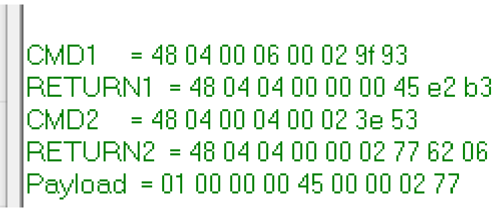
AT+COMMAND2=48 04 00 04 00 02,1

AT+DATACUT2=9,1,4+5+6+7



```
CMD1     = 48 04 00 06 00 02 9f 93
RETURN1  = 48 04 04 00 00 00 05 e3 43
CMD2     = 48 04 00 04 00 02 3e 53
RETURN2  = 48 04 04 00 00 00 41 e3 70
Payload  = 01 00 00 00 05 00 00 00 41
```

Let’s plug a load (A 0,06KW motor)

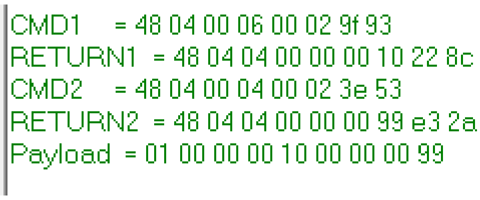


```
CMD1     = 48 04 00 06 00 02 9f 93
RETURN1  = 48 04 04 00 00 00 45 e2 b3
CMD2     = 48 04 00 04 00 02 3e 53
RETURN2  = 48 04 04 00 00 02 77 62 06
Payload  = 01 00 00 00 45 00 00 02 77
```

So we have 45 Hex or 69 Dec Watts Active power

And we have 2 77 wich is 2 119 in Decimal so 2*256+119 = 631 mA so 0,631 Amperes

Let’s try with a Laptop

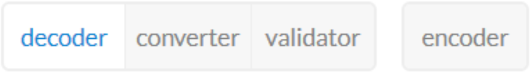


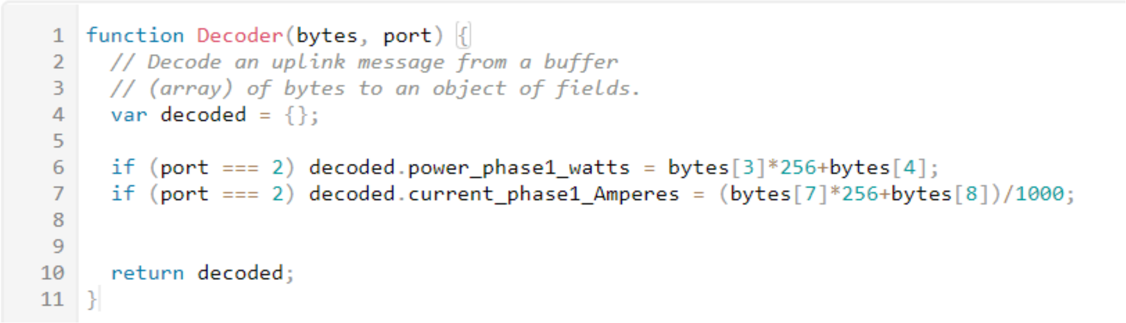
```
CMD1     = 48 04 00 06 00 02 9f 93
RETURN1  = 48 04 04 00 00 00 10 22 8c
CMD2     = 48 04 00 04 00 02 3e 53
RETURN2  = 48 04 04 00 00 00 99 e3 2a
Payload  = 01 00 00 00 10 00 00 00 99
```

10 in Hex so 16 Watts

153 mA so 0,153 A

Let’s adjust the Payload

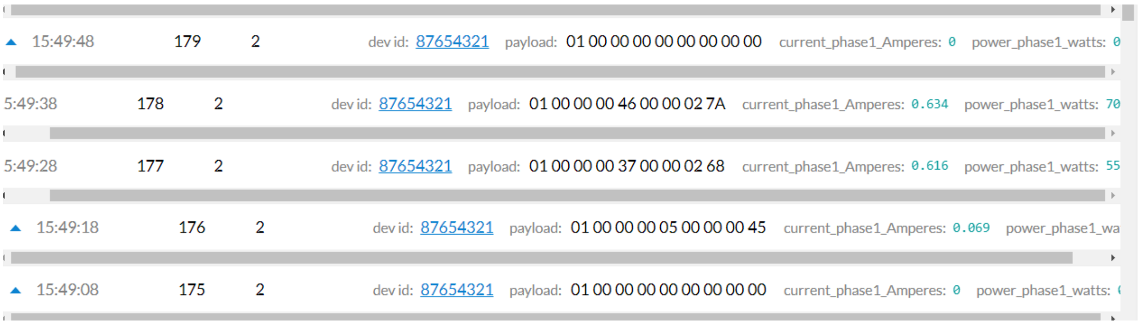




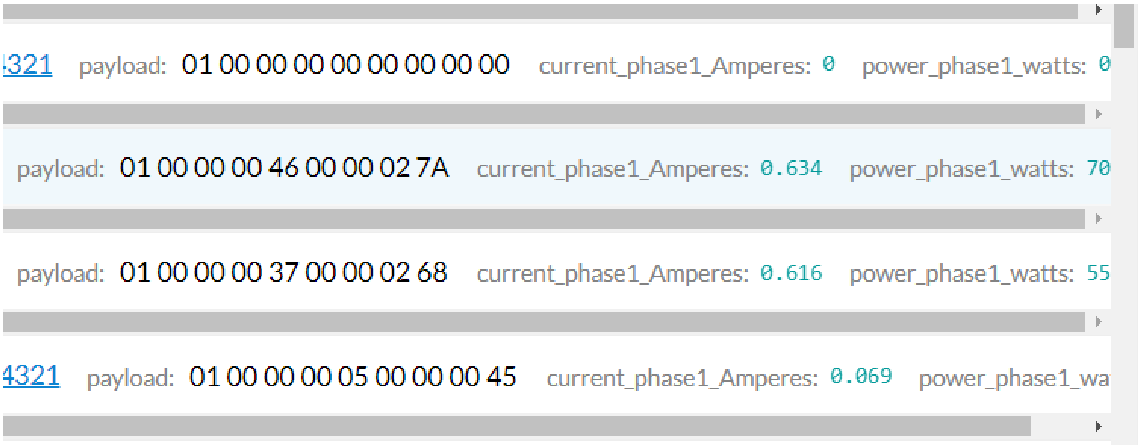
```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};

  if (port === 2) decoded.power_phase1_watts = bytes[3]*256+bytes[4];
  if (port === 2) decoded.current_phase1_Amperes = (bytes[7]*256+bytes[8])/1000;


  return decoded;
}
```



```
15:49:48    179    2    dev id: 87654321    payload: 01 00 00 00 00 00 00 00 00    current_phase1_Amperes: 0    power_phase1_watts: 0
5:49:38    178    2    dev id: 87654321    payload: 01 00 00 00 46 00 00 02 7A    current_phase1_Amperes: 0.634    power_phase1_watts: 70
5:49:28    177    2    dev id: 87654321    payload: 01 00 00 00 37 00 00 02 68    current_phase1_Amperes: 0.616    power_phase1_watts: 55
15:49:18    176    2    dev id: 87654321    payload: 01 00 00 00 05 00 00 00 45    current_phase1_Amperes: 0.069    power_phase1_wa
15:49:08    175    2    dev id: 87654321    payload: 01 00 00 00 00 00 00 00 00    current_phase1_Amperes: 0    power_phase1_watts:
```



```
321   payload:  01 00 00 00 00 00 00 00 00   current_phase1_Amperes: 0   power_phase1_watts: 0
payload:  01 00 00 00 46 00 00 02 7A   current_phase1_Amperes: 0.634   power_phase1_watts: 70
payload:  01 00 00 00 37 00 00 02 68   current_phase1_Amperes: 0.616   power_phase1_watts: 55
4321   payload:  01 00 00 00 05 00 00 00 45   current_phase1_Amperes: 0.069   power_phase1_wa
```

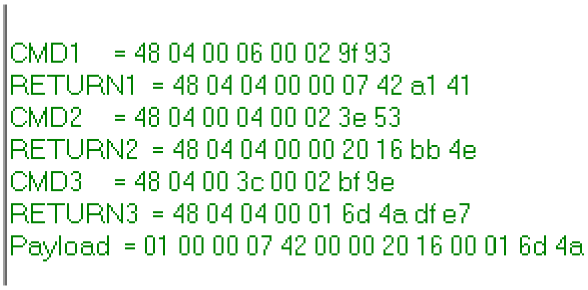
Now let’s read the active energy phase 1

The right parametres are:

AT+COMMAND3=48 04 00 3C 00 02,1

AT+DATACUT3=9,1,4+5+6+7

Now let’s connect a heater

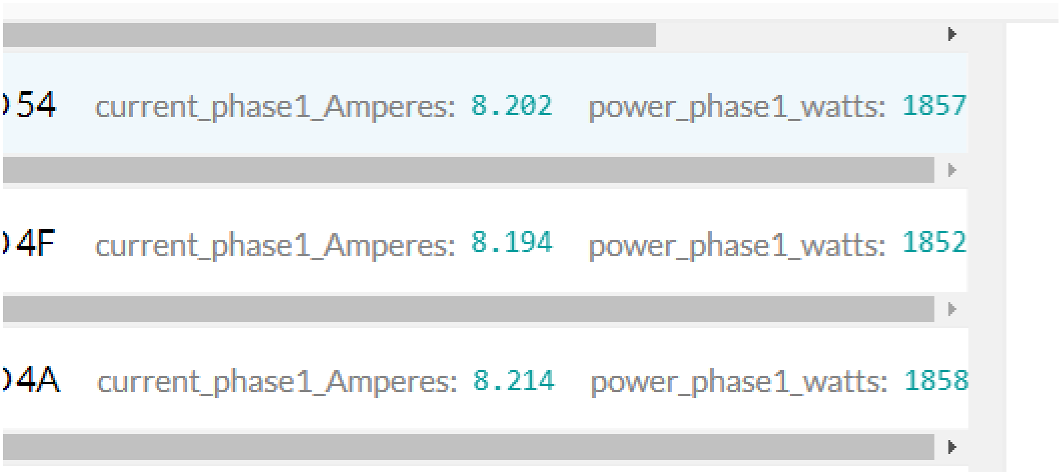


```
CMD1     = 48 04 00 06 00 02 9f 93
RETURN1  = 48 04 04 00 00 07 42 a1 41
CMD2     = 48 04 00 04 00 02 3e 53
RETURN2  = 48 04 04 00 00 20 16 bb 4e
CMD3     = 48 04 00 3c 00 02 bf 9e
RETURN3  = 48 04 04 00 01 6d 4a df e7
Payload  = 01 00 00 07 42 00 00 20 16 00 01 6d 4a
```

256x7=1792

42 Hex = 66

1792+66=1858Watts

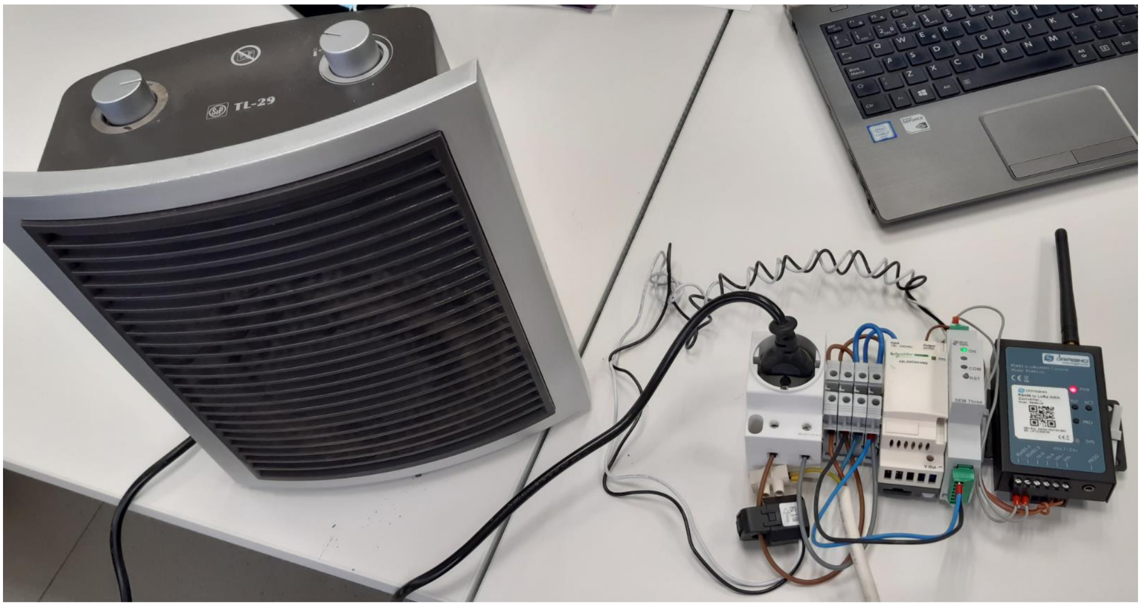


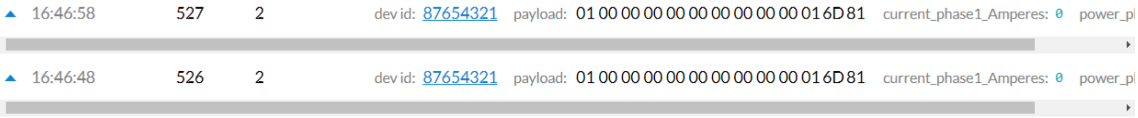
```
54   current_phase1_Amperes: 8.202   power_phase1_watts: 1857
4F   current_phase1_Amperes: 8.194   power_phase1_watts: 1852
4A   current_phase1_Amperes: 8.214   power_phase1_watts: 1858
```

Energy

1 6D 4A Hex = 1 109 74

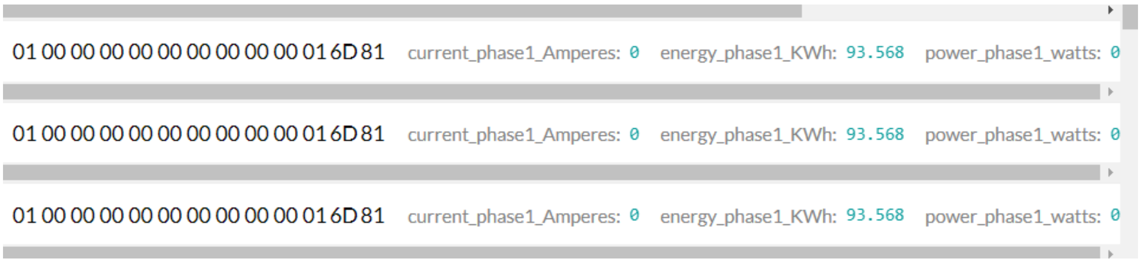
65535 + 109*256 + 74 =65.535 + 27.904 + 74=93.513 Wh= 93,513KWh



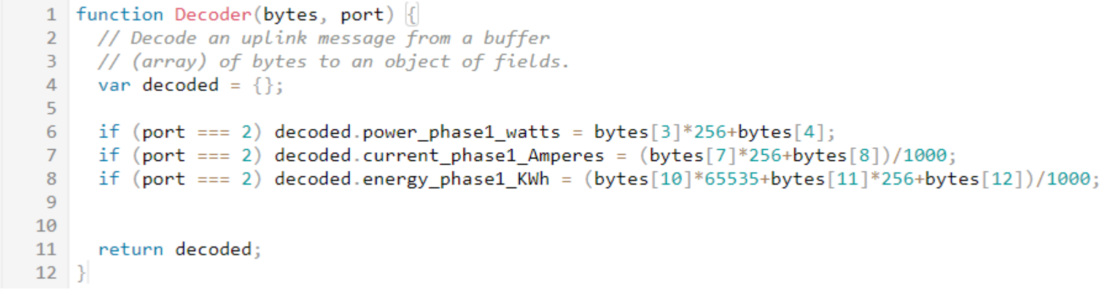


```
▲ 16:46:58    527    2    dev id: 87654321   payload: 01 00 00 00 00 00 00 00 00 00 01 6D 81   current_phase1_Amperes: 0   power_pl
▲ 16:46:48    526    2    dev id: 87654321   payload: 01 00 00 00 00 00 00 00 00 00 01 6D 81   current_phase1_Amperes: 0   power_pl
```

Energy is acumulative



```
01 00 00 00 00 00 00 00 00 00 01 6D 81   current_phase1_Amperes: 0   energy_phase1_KWh: 93.568   power_phase1_watts: 0
01 00 00 00 00 00 00 00 00 00 01 6D 81   current_phase1_Amperes: 0   energy_phase1_KWh: 93.568   power_phase1_watts: 0
01 00 00 00 00 00 00 00 00 00 01 6D 81   current_phase1_Amperes: 0   energy_phase1_KWh: 93.568   power_phase1_watts: 0
```

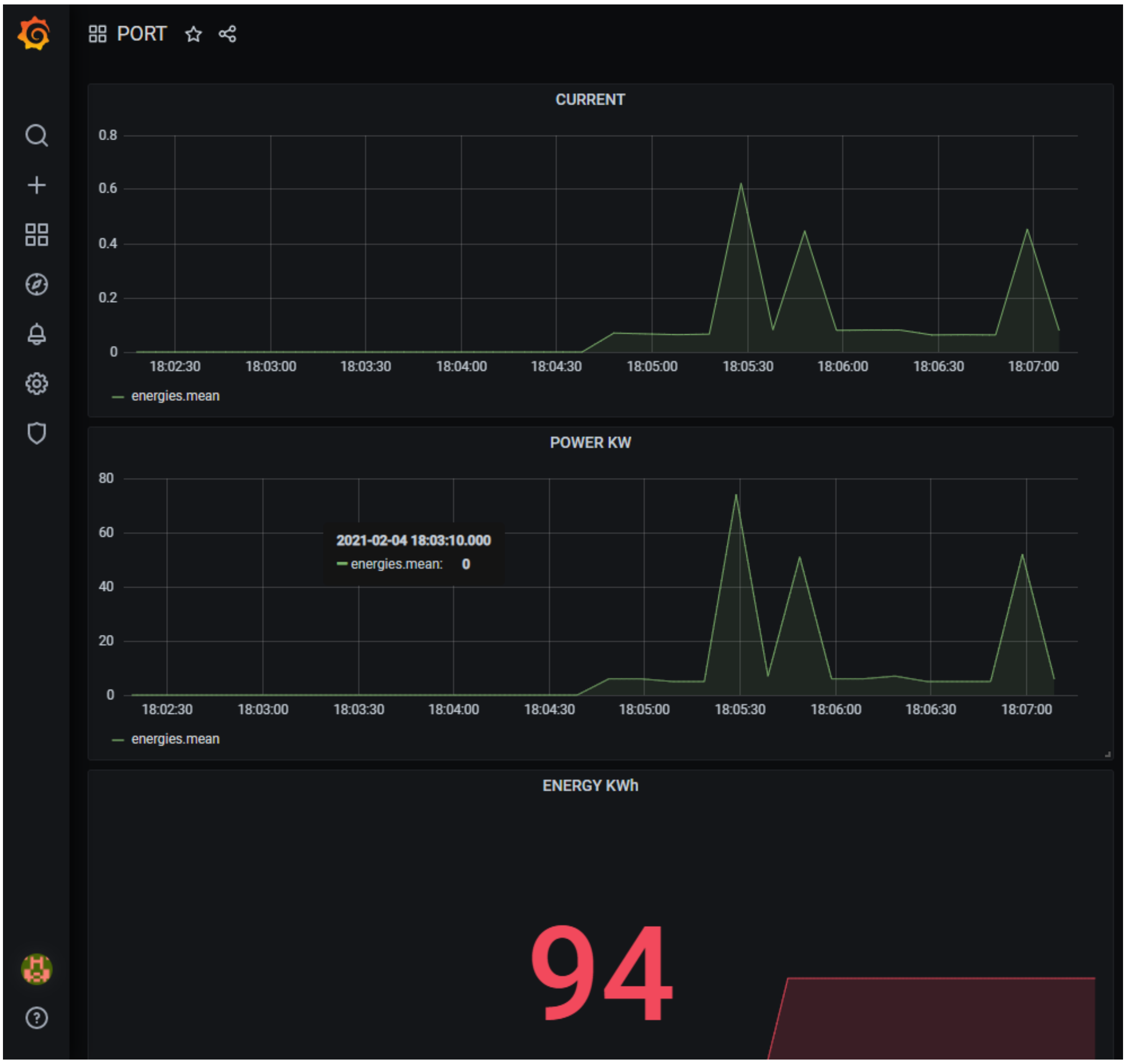


```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};

  if (port === 2) decoded.power_phase1_watts = bytes[3]*256+bytes[4];
  if (port === 2) decoded.current_phase1_Amperes = (bytes[7]*256+bytes[8])/1000;
  if (port === 2) decoded.energy_phase1_KWh = (bytes[10]*65535+bytes[11]*256+bytes[12])/1000;


  return decoded;
}
```

Storing to InfluxDB and Grafana



You have the code here:

https://github.com/xavierflorensa/PICKDATA-SEM-Three-to-LoRaWAN-energy-metering