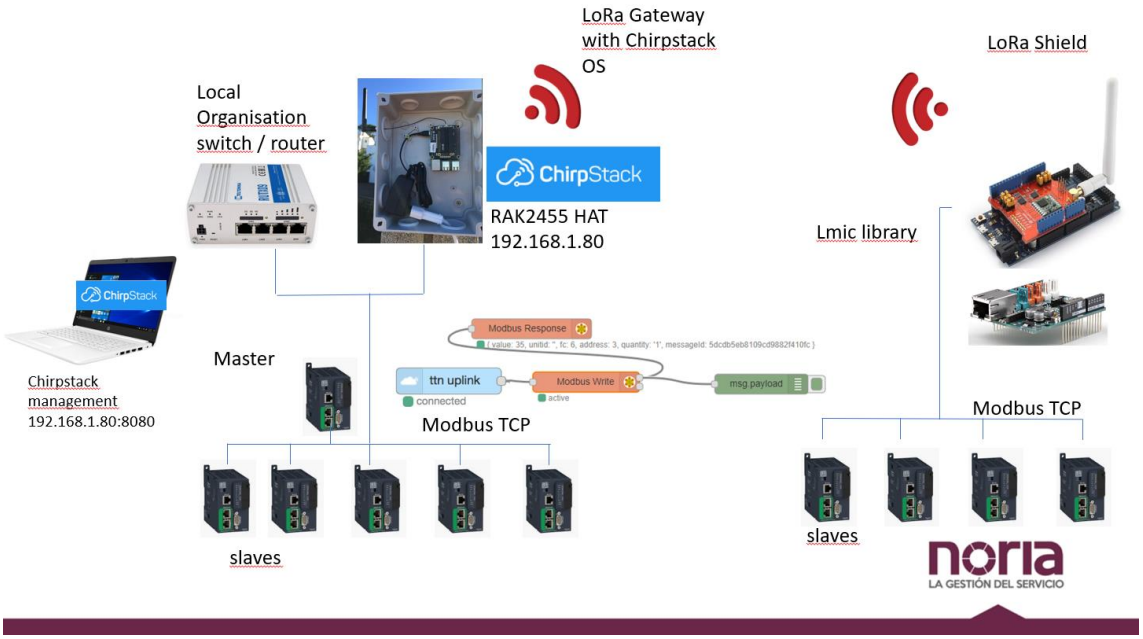
PLC alarms communication thru LoRa Server Gateway

SYSTEM ARCHITECTURE

We will see how to transmit PLC data with LoRa Server on a closed network

Topology



We need a node (PLC emitter) and a Gateway.

The Gateway is a RAK2245 with a raspberry and an operating system to perfom the LoRa server

TESTING LMIC LIBRARY with LoRa Server

We adapt LMIC example to our needs

From

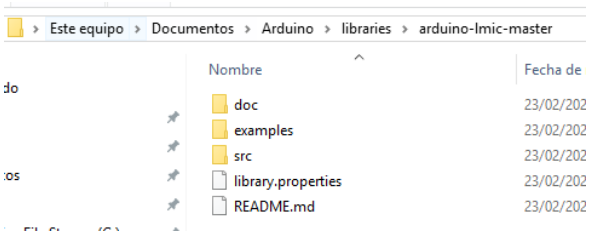
Download the zip file from

https://github.com/matthijskooijman/arduino-lmic

copy the zip folder and paste it on the Arduino libraries folder

unzip it

open the folder and copy the folder to the Arduino libraries folder so there is only one directory Arduino-lmic-library



First we try with the ABP example

https://github.com/matthijskooijman/arduino-lmic/tree/master/examples/ttn-abp

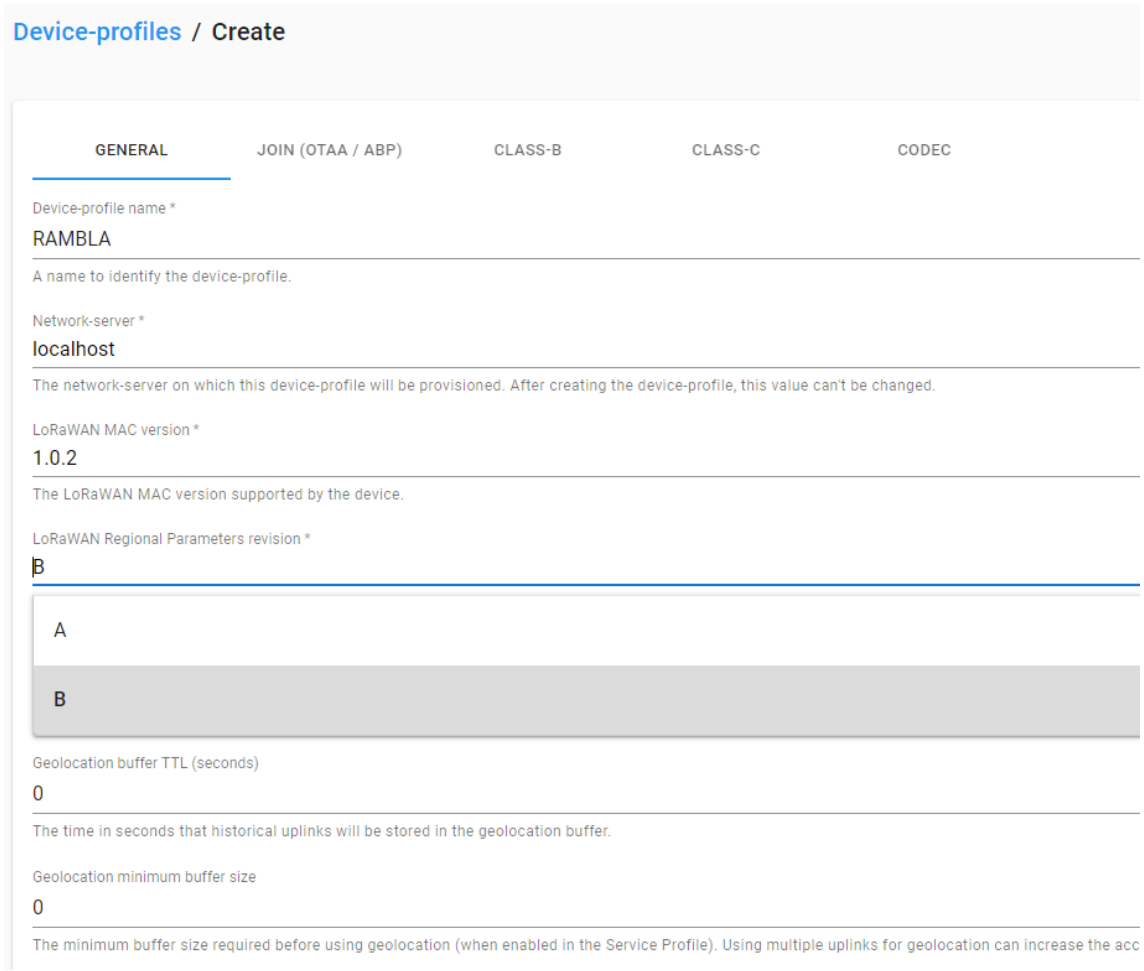
First we try with an Arduino UNO

We have to change the keys on the sketch, accordingly to the keys on the new LoraServer created application

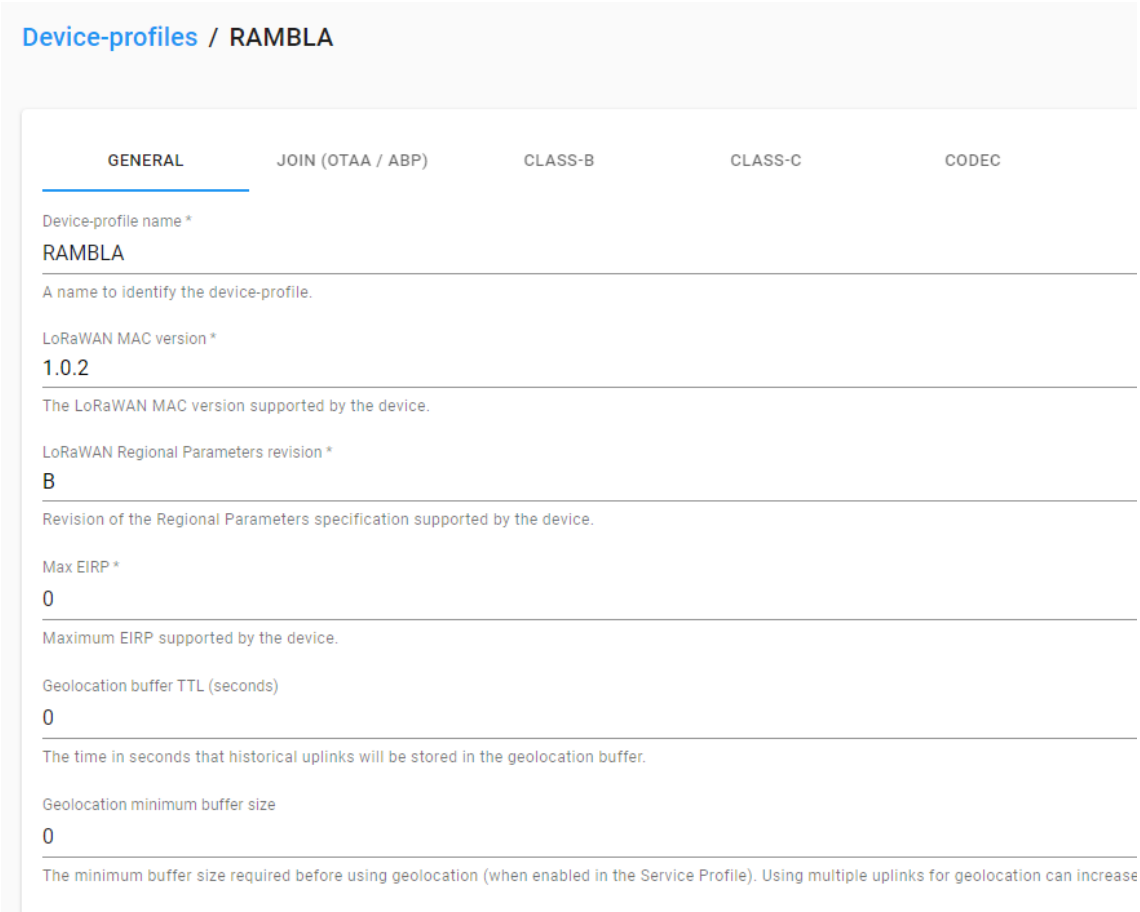
We create first a Device-profile

Follow these steps to create your device

https://www.youtube.com/watch?v=mkuS5QUj5Js



Once created



Device-profiles / RAMBLA

GENERAL | JOIN (OTAA / ABP) | CLASS-B | CLASS-C | CODEC

☐ Device supports OTAA

RX1 delay *
0
RX1 delay (valid values are 0 - 15).

RX1 data-rate offset *
0
Please refer the LoRaWAN Regional Parameters specification for valid values.

RX2 data-rate *
0
Please refer the LoRaWAN Regional Parameters specification for valid values.

RX2 channel frequency (Hz) *
0

Factory-preset frequencies (Hz) *
868000000
List of factory-preset frequencies (Hz), comma separated.

Device-profiles / RAMBLA

GENERAL | JOIN (OTAA / ABP) | CLASS-B | CLASS-C | CODEC

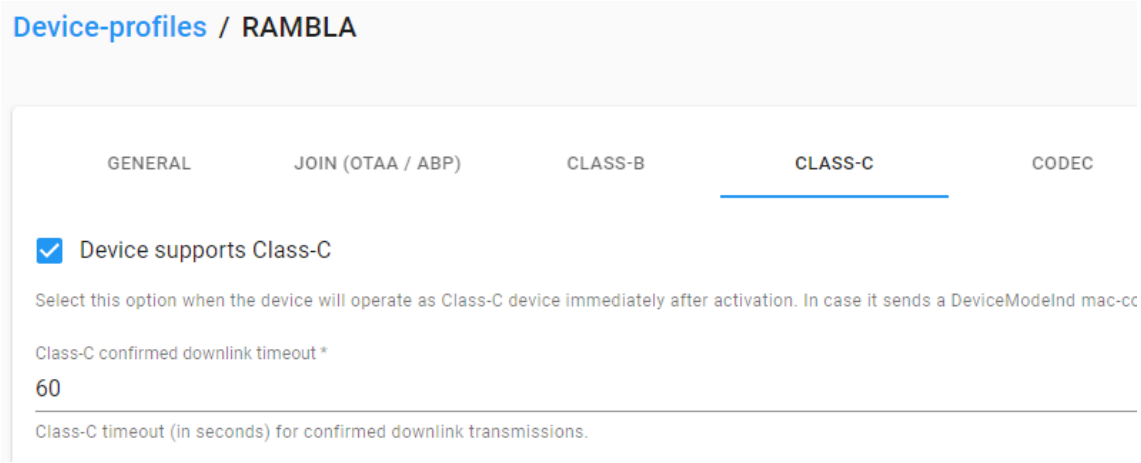
☑ Device supports Class-B

Class-B confirmed downlink timeout *
30
Class-B timeout (in seconds) for confirmed downlink transmissions.

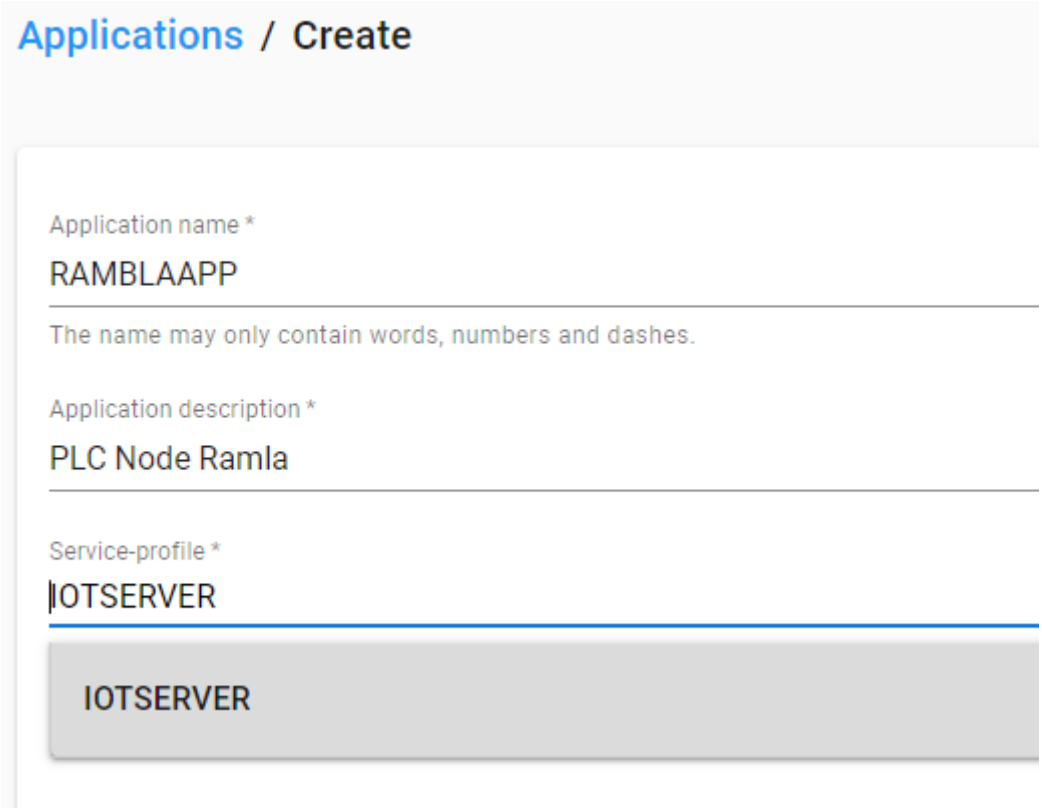
Class-B ping-slot periodicity *
every 2 seconds
Class-B ping-slot periodicity.

Class-B ping-slot data-rate *
5

Class-B ping-slot frequency (Hz) *
1



Now we create an Application



We create a new device

Applications / RAMBLAAPP / Devices / Create

GENERAL VARIABLES TAGS

Device name *

Rambla_node

The name may only contain words, numbers and dashes.

Device description *

PLC on Rambla device

Device EUI *

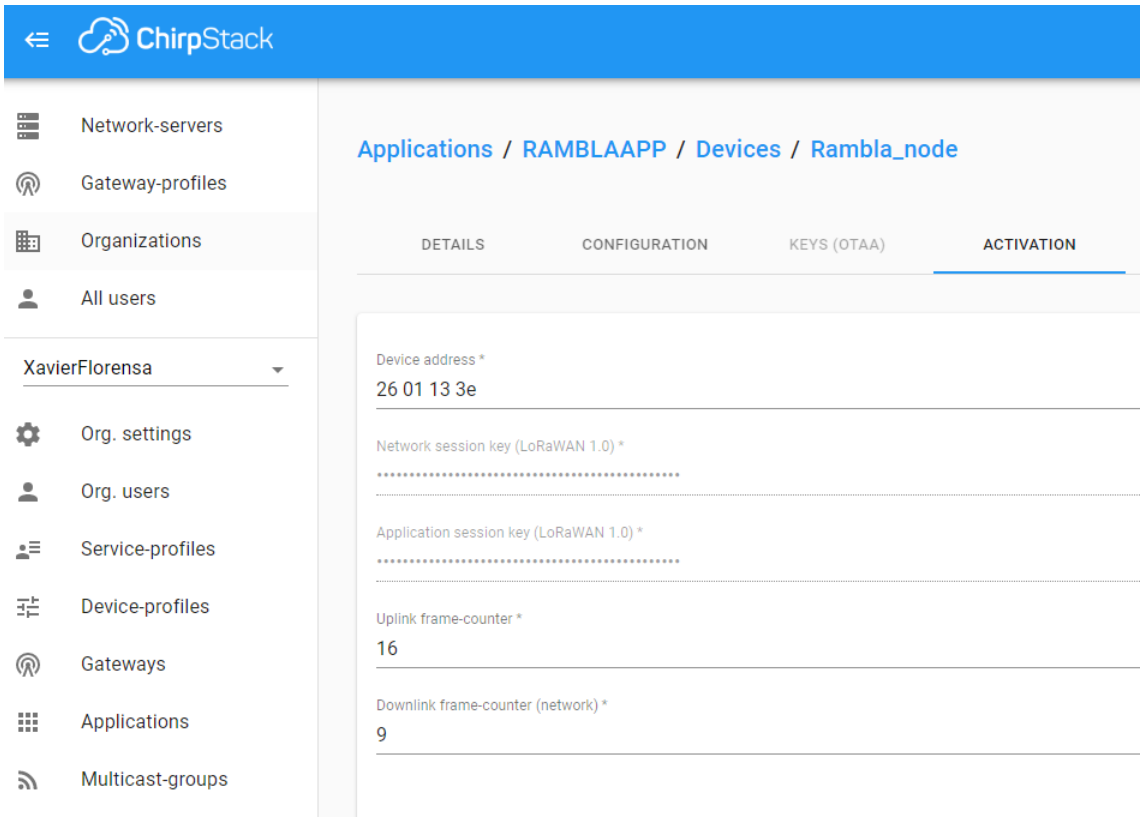
3e 78 6a 68 f5 1c 27 57

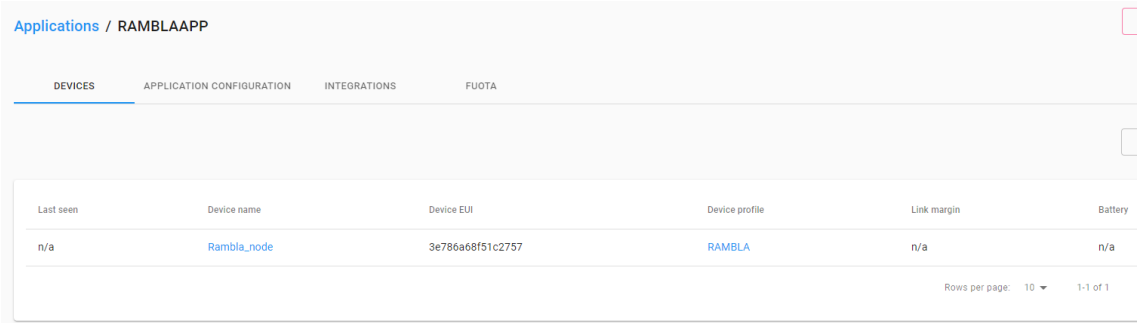
Device-profile *

RAMBLA

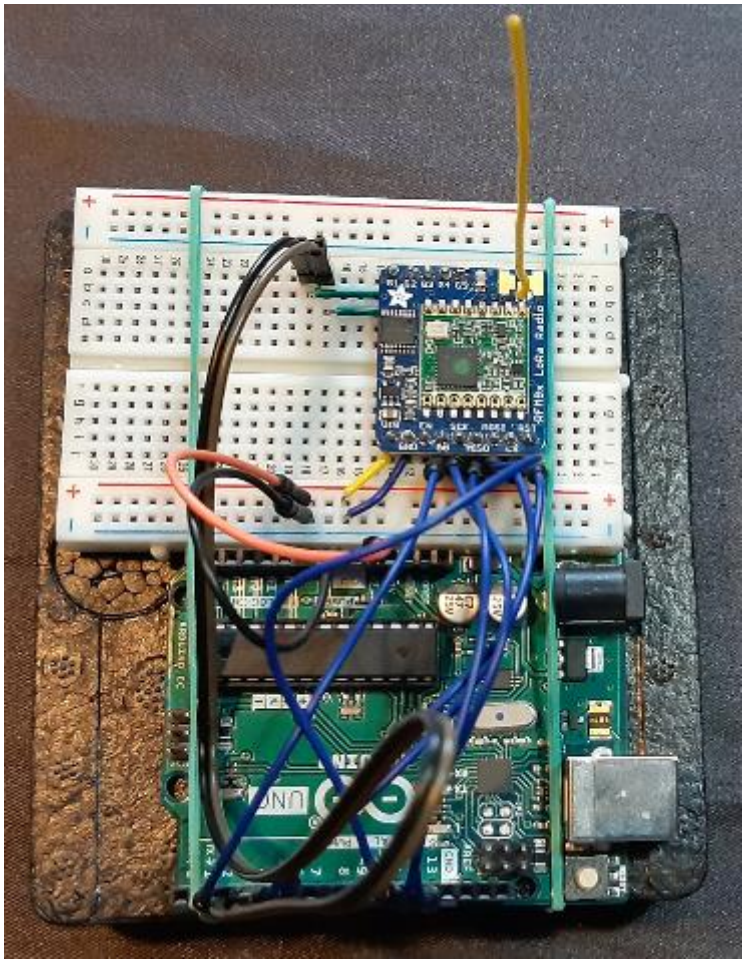
☑ Disable frame-counter validation

Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.

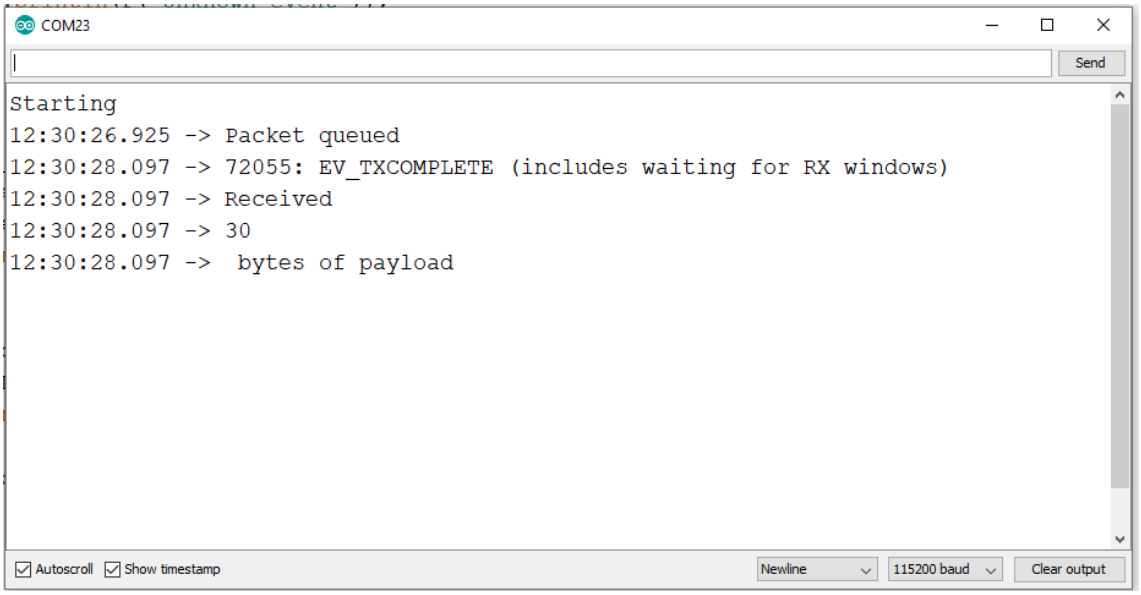




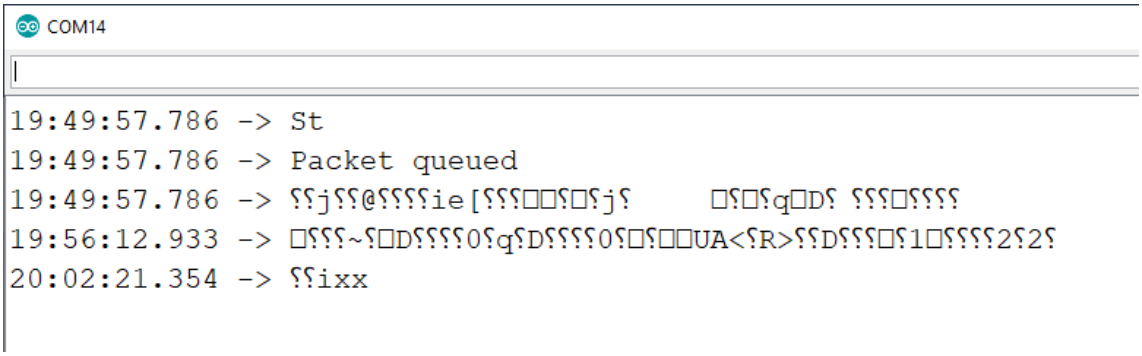
TESTING WITH ARDUINO UNO and RFM95 LoRa chip



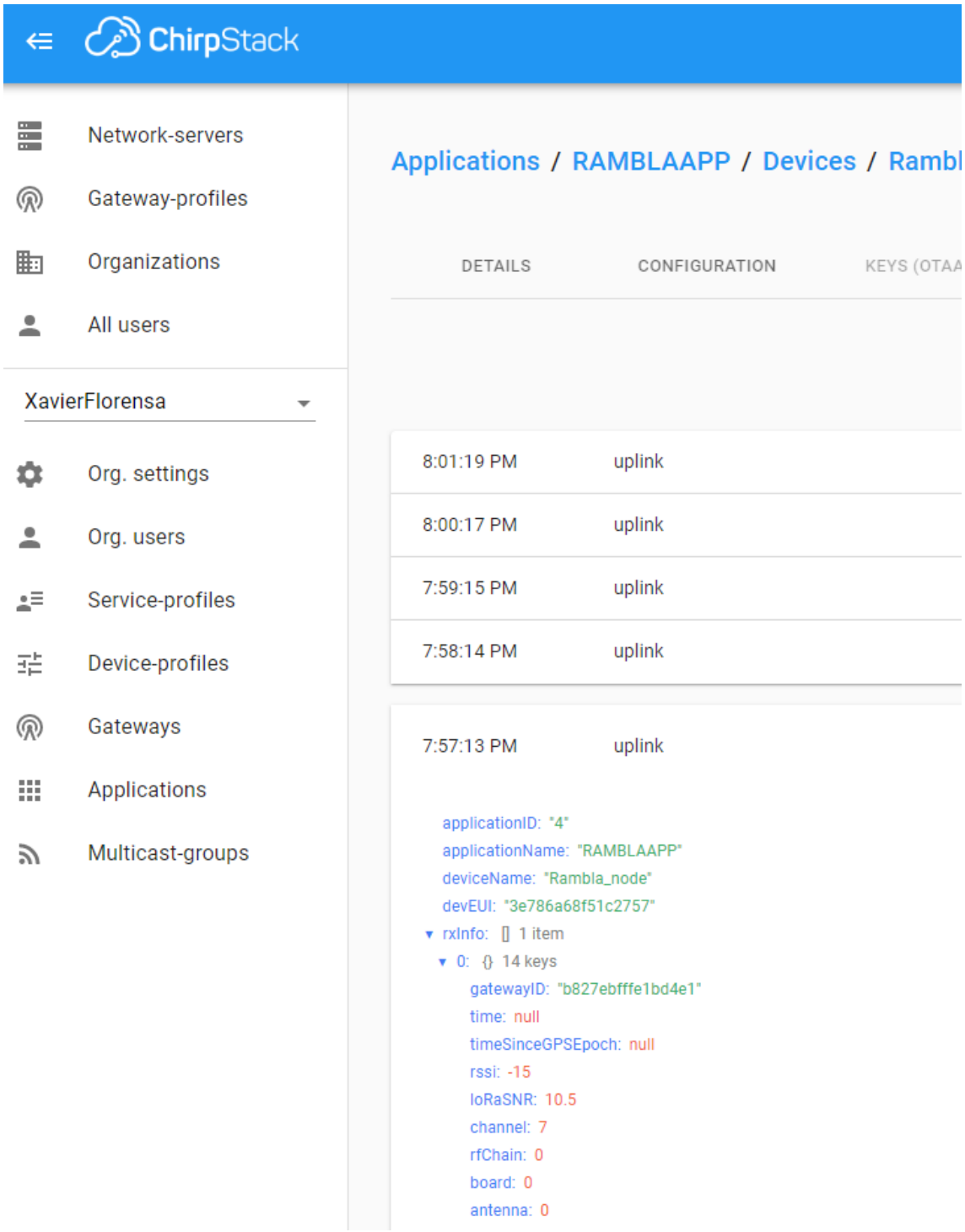
Power Up Arduino and… it Works



Attention, select 115200 bauds on the terminal, if not, you will see this

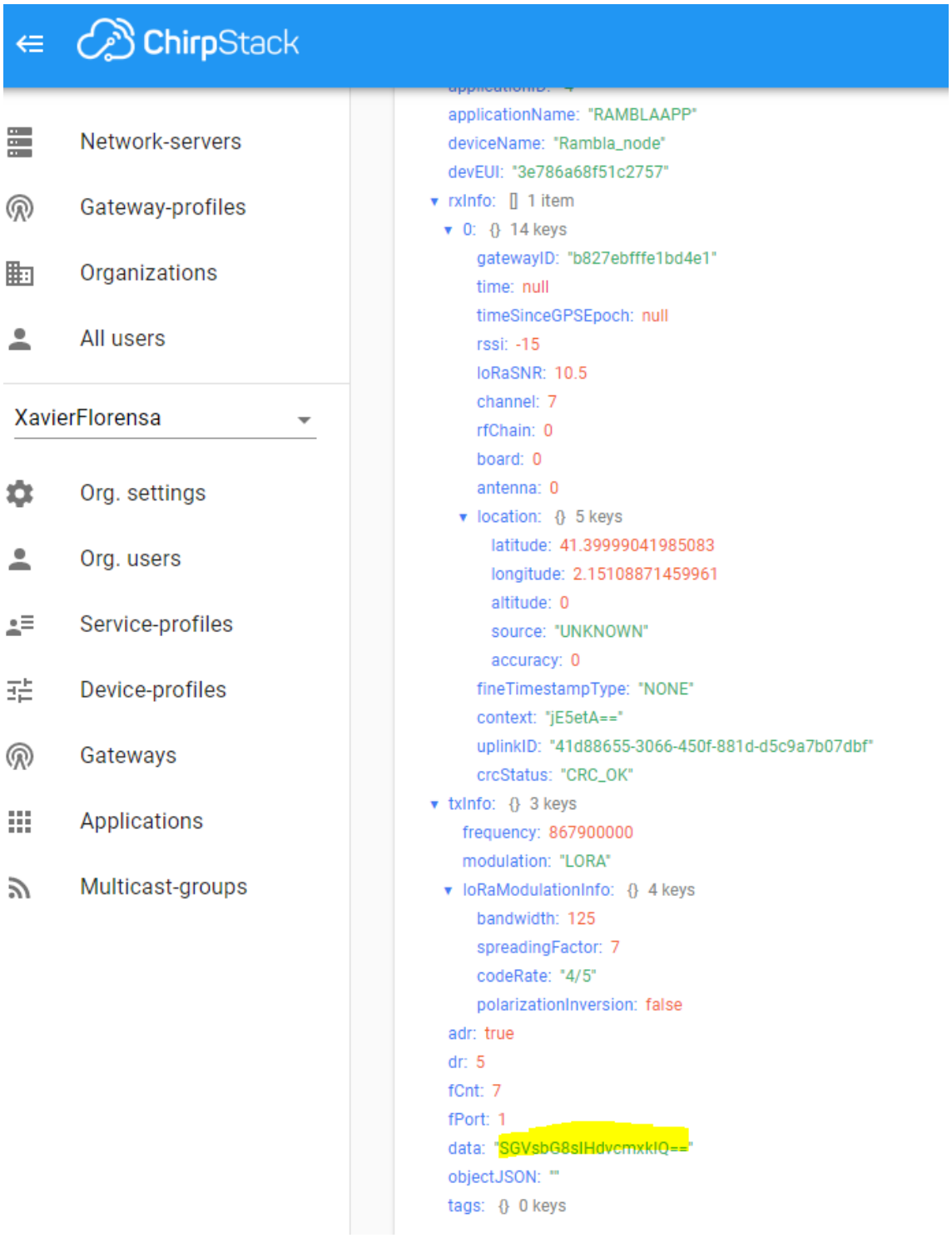


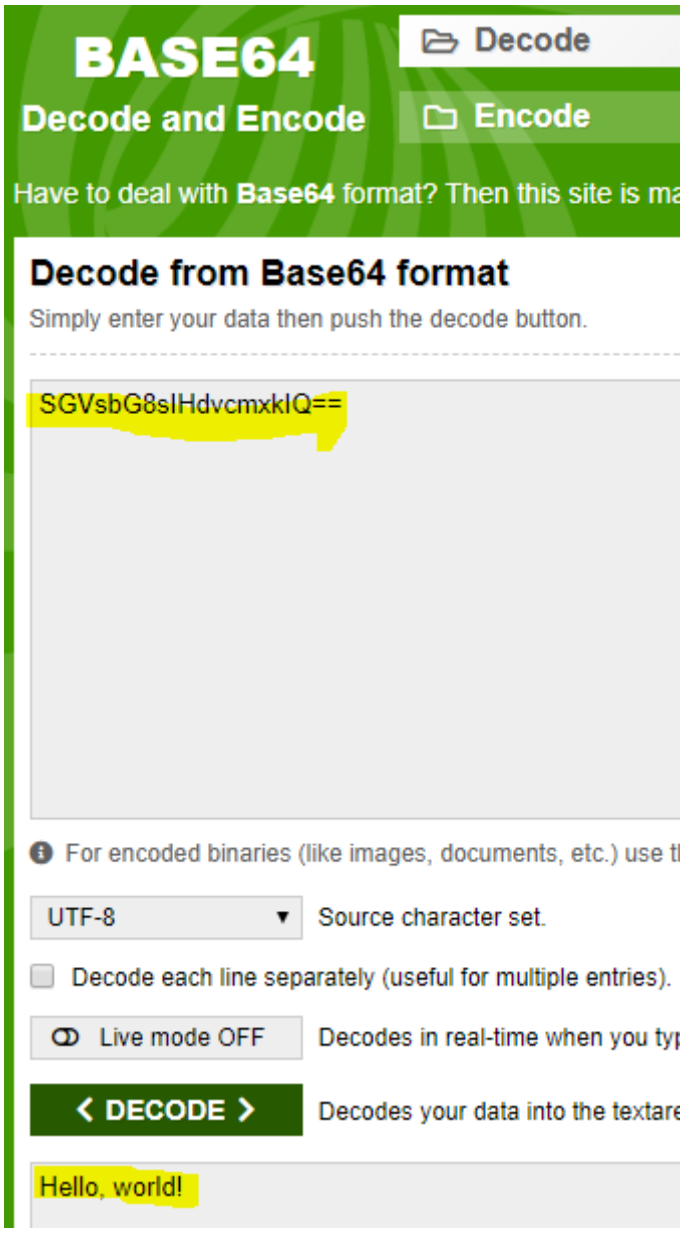
Taking a look at the application console



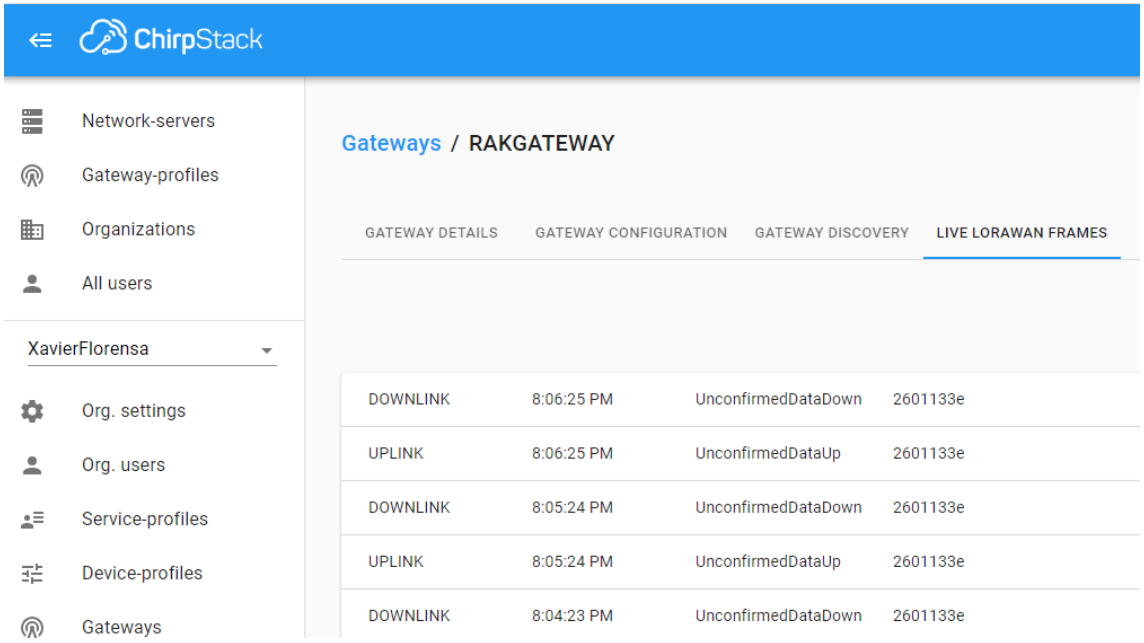
Taking a look at the payload under DATA

SGVsbG8sIHdvcmxkIQ==


ChirpStack
Network-servers
Gateway-profiles
Organizations
All users
XavierFlorensa
Org. settings
Org. users
Service-profiles
Device-profiles
Gateways
Applications
Multicast-groups
applicationName: "RAMBLAAPP"
deviceName: "Rambla_node"
devEUI: "3e786a68f51c2757"
rxInfo: [] 1 item
0: {} 14 keys
gatewayID: "b827ebfffe1bd4e1"
time: null
timeSinceGPSEpoch: null
rssi: -15
loRaSNR: 10.5
channel: 7
rfChain: 0
board: 0
antenna: 0
location: {} 5 keys
latitude: 41.39999041985083
longitude: 2.15108871459961
altitude: 0
source: "UNKNOWN"
accuracy: 0
fineTimestampType: "NONE"
context: "jE5etA=="
uplinkID: "41d88655-3066-450f-881d-d5c9a7b07dbf"
crcStatus: "CRC_OK"
txInfo: {} 3 keys
frequency: 867900000
modulation: "LORA"
loRaModulationInfo: {} 4 keys
bandwidth: 125
spreadingFactor: 7
codeRate: "4/5"
polarizationInversion: false
adr: true
dr: 5
fCnt: 7
fPort: 1
data: "SGVsbG8sIHdvcmxkIQ=="
objectJSON: ""
tags: {} 0 keys



And the Gateway

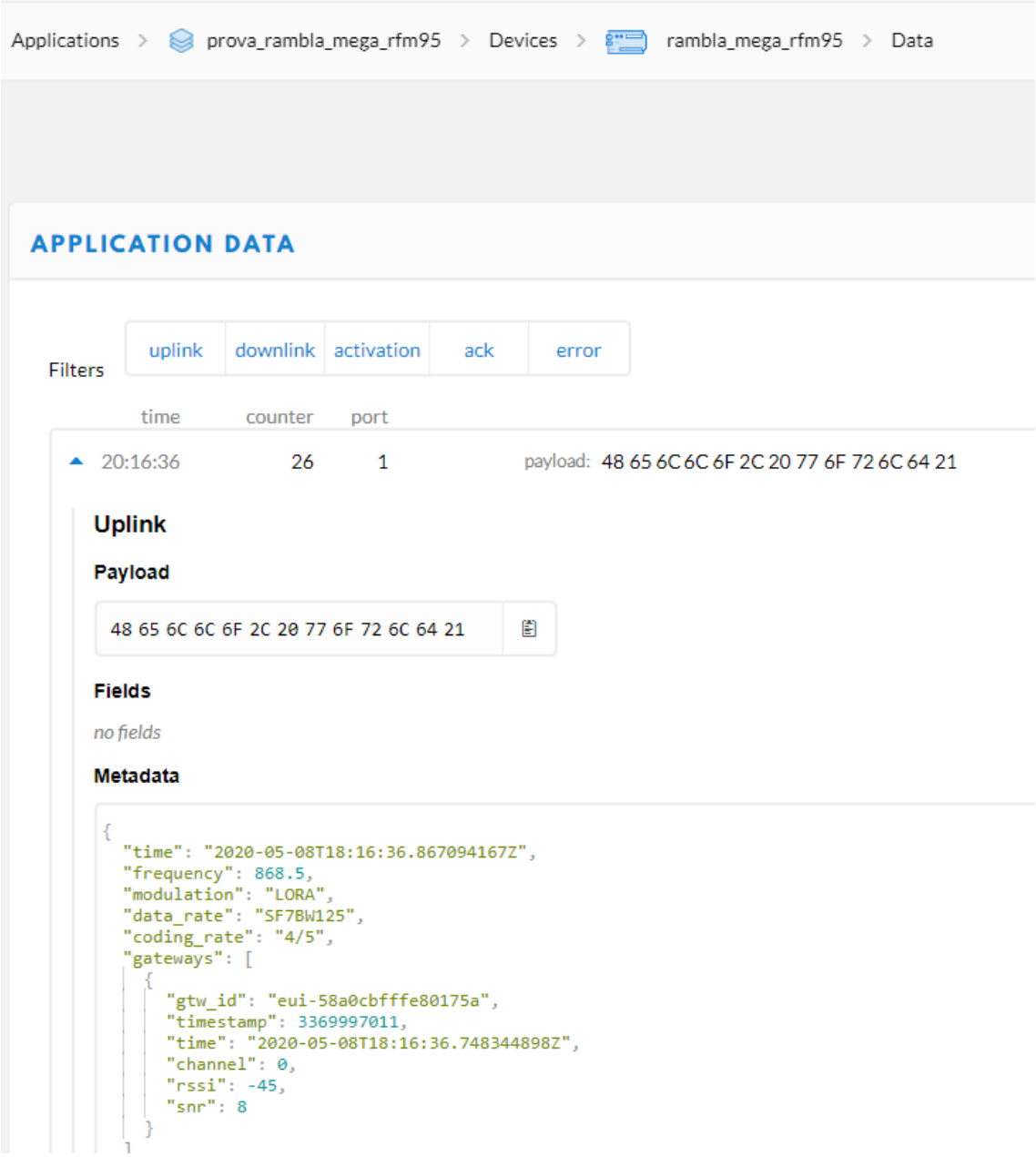


Even I can see the same on the other TTN Gateway I'm testing

48=h

65=e

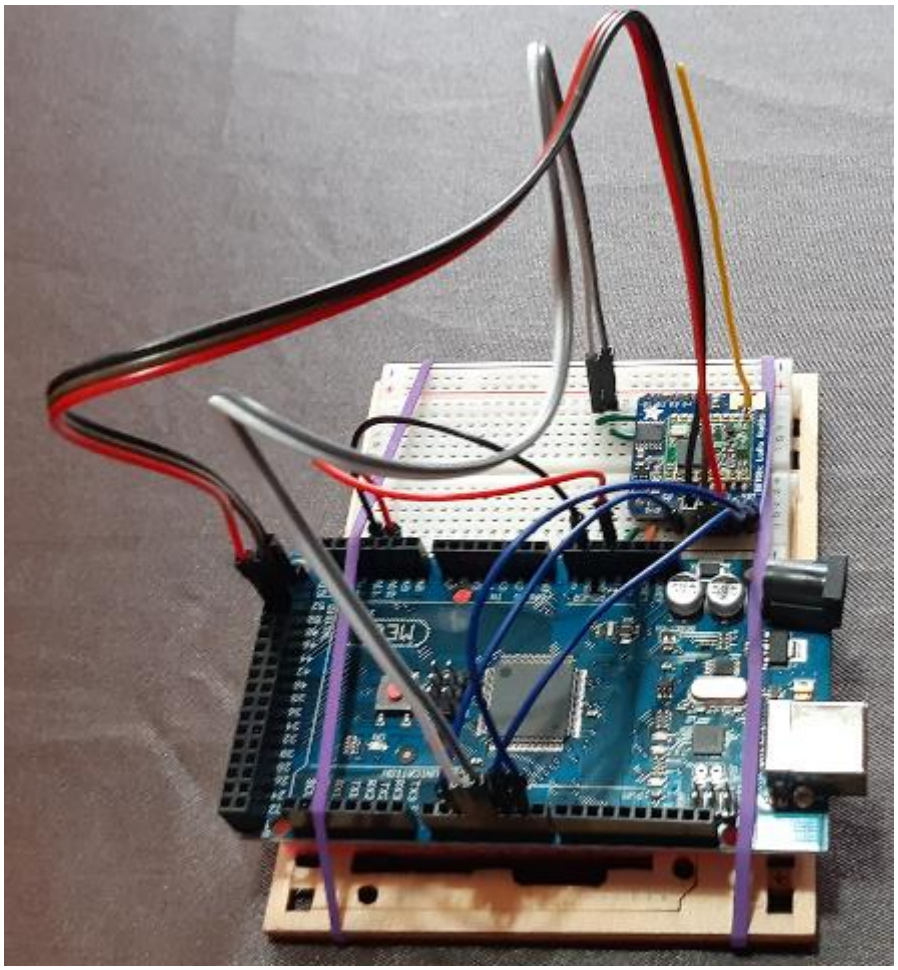
Llo world



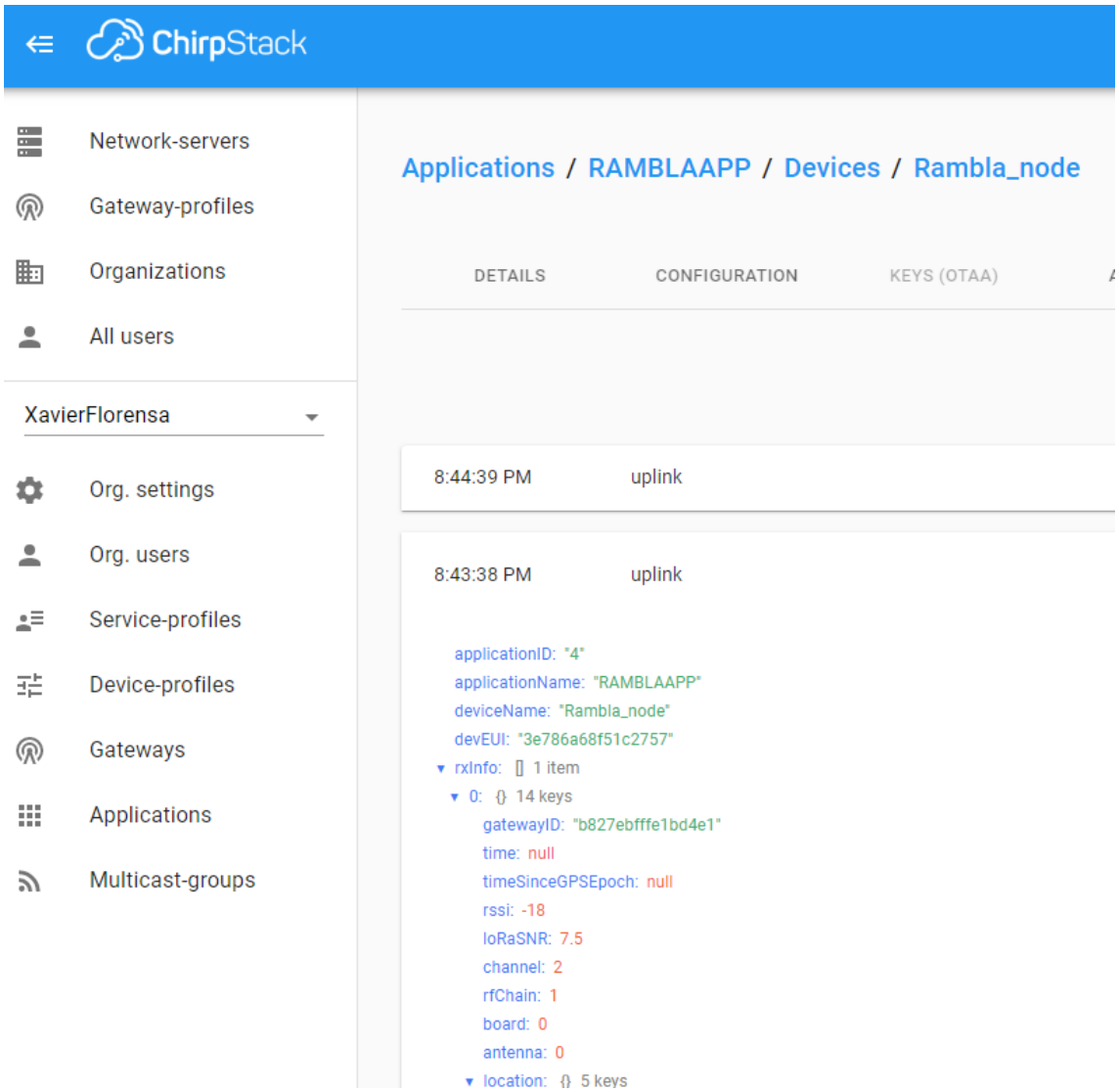
Now we test with Arduino MEGA

TESTING WITH ARDUINO MEGA and RFM95 LoRa chip

Attention, SPI pins are different from Arduino UNO. On the MEGA SPI are 50 to 52 and also available at the central connector on the MEGA



Now it Works with the MEGA

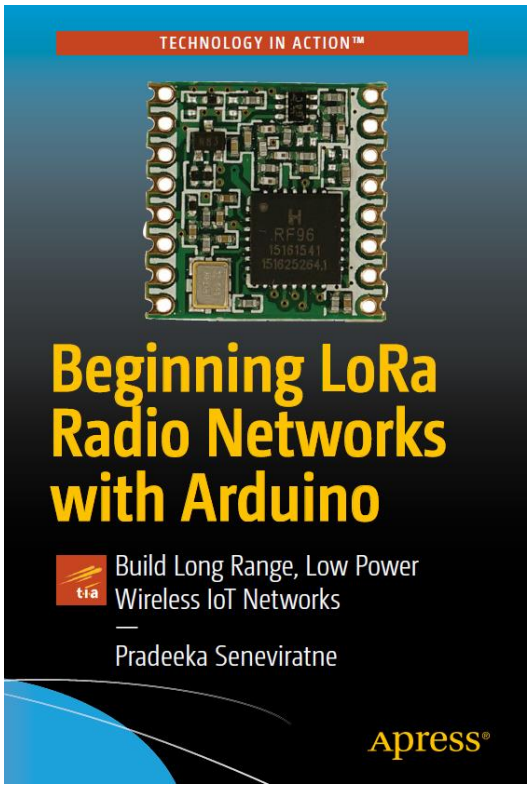


Now we try with the Draguino LoRa Shield

TESTING ARDUINO MEGA and DRAGUINO LoRa SHIELD

There is a Little diference

According to this guide



The pin connections between The Arduino and the LoRa are the following for the LMIC library

With 50, 51 and 52 instead of 11, 12 and 13.

CHAPTER 5 BUILDING A LO

Table 5-2. Wiring Co

Arduino	RFM9x
2	G0
3	G1
4	G2
5	RST
6	CS
8	
11	MOSI
12	MISO
13	SCK
5V	
GND	

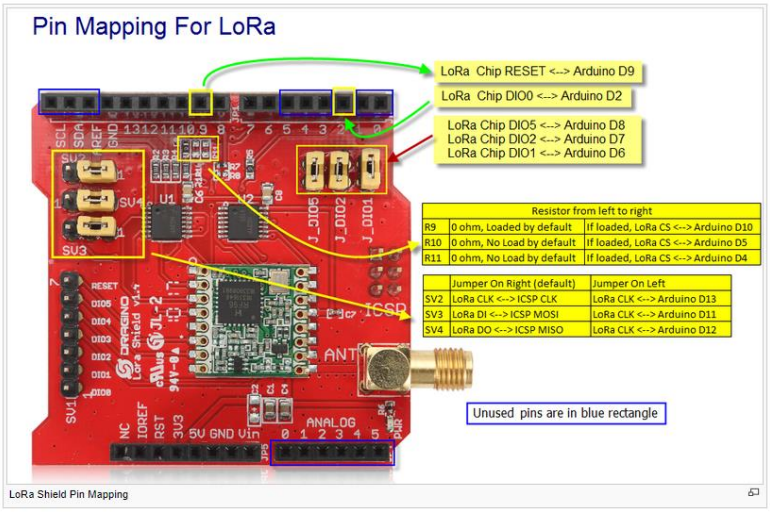
And on the point to point communications with the Radiohead library the pinout was different

So only the SPI pinout is the same

Table 4-1. Wiring Connections for

Arduino/Metro	Radio Transceiver
VIN	VIN
GND	GND
2	RST
3	G0
4	CS
8	
11	MOSI
12	MISO
13	SCK

We have top ay attention to the Draguino pinout to check if it is like described for the first table for LMIC



Since we will be using the ICSP connector, we can forget about the following signals

11	MOSI
12	MISO
13	SCK

They are connected on the shield

But we have to adjust these

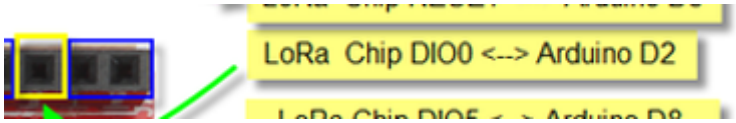
Table 5-2. Wiring Co

Arduino	RFM9x
2	G0
3	G1
4	G2
5	RST
6	CS

Let’s check all these pins before using the LMIC library with Draguino

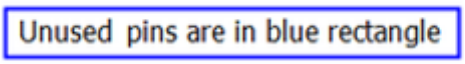
PIN 2

We are lucky, by default Draguino Lora connects Pin 2 with DI0 (G0)

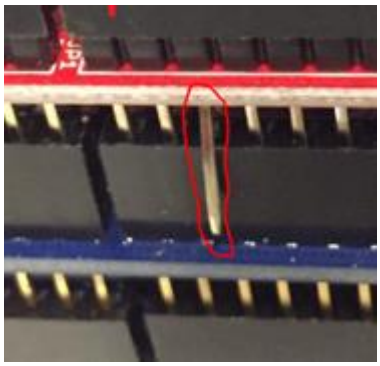


PIN 3

Pin 3 is unused by Draguino



We can connect pin 3 with G1 (DI01) on the Draguino shield externally

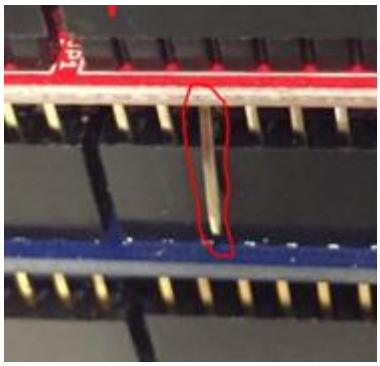


PIN 4

Pin 4 is unused by Draguino

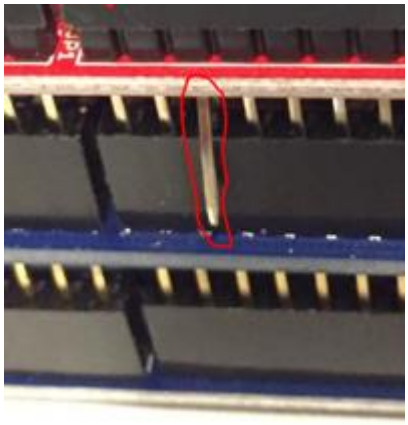
Unused pins are in blue rectangle

So we do the same we did with pin 3

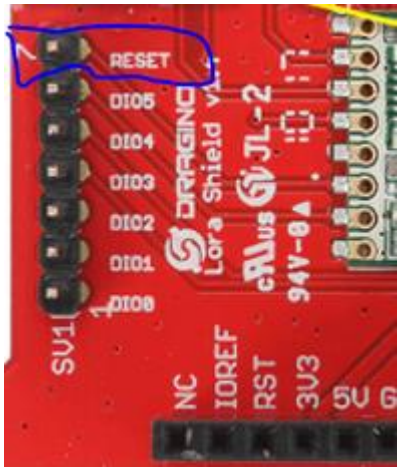


PIN 5

We have a conflict with pin 5 since it will be used by LoRa RST but we can leave the pin like this



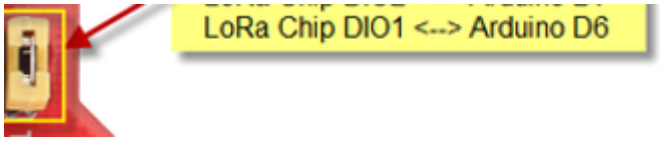
Conect to the RESET PIN of the Draguino



And we have to isolate pin 9 since it is connected to the LoRa RESET

PIN 6

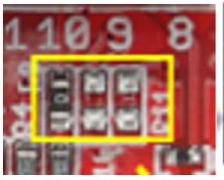
Pin 6 is busy



So we have to open this jumper

And connect pin 6 with LoRa CS

CS LoRa will be connected according to this jumper

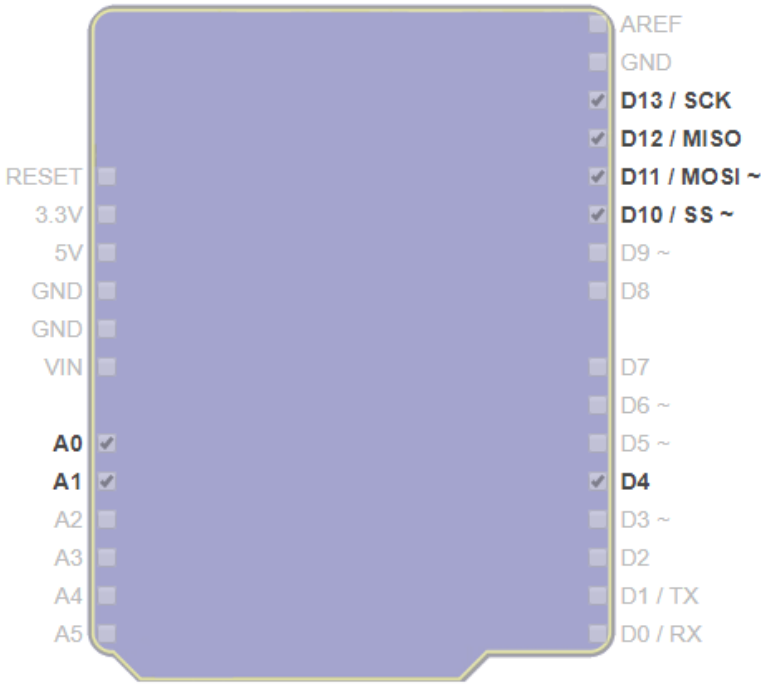


Resistor from left to right		
R9	0 ohm, Loaded by default	If loaded, LoRa CS <--> Arduino D10
R10	0 ohm, No Load by default	If loaded, LoRa CS <--> Arduino D5
R11	0 ohm, No Load by default	If loaded, LoRa CS <--> Arduino D4

We have moved this jumper on the Radiohead library to position R10.

So we have to restore the jumper to position R10, isolate the pin and connect to pin 6

But then we have to isolate pin 10 of Ethernet Shield and connect to pin 10 of Arduino



Note:
Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on D11, D12, and D13 on "classic" format Arduino models such as the Duemilanove, and pins D50, D51, and D52 on the Arduino Mega.

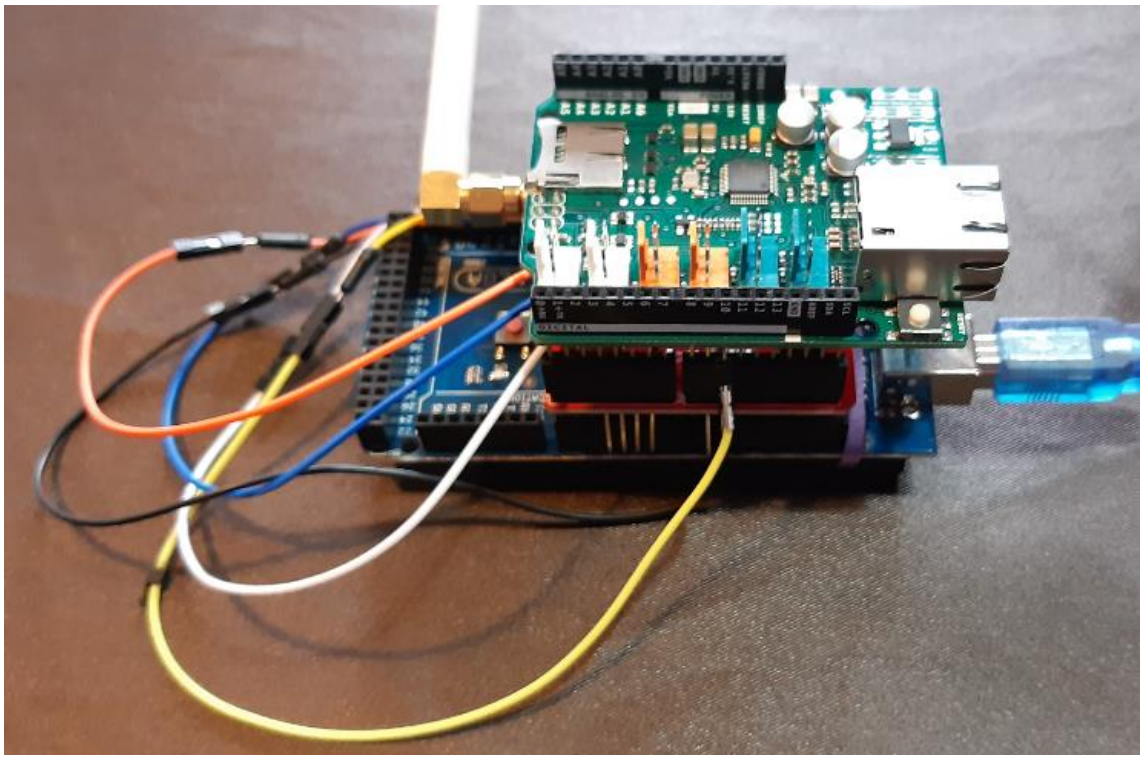
D10 is used to select the W5100 and cannot be used for general I/O.

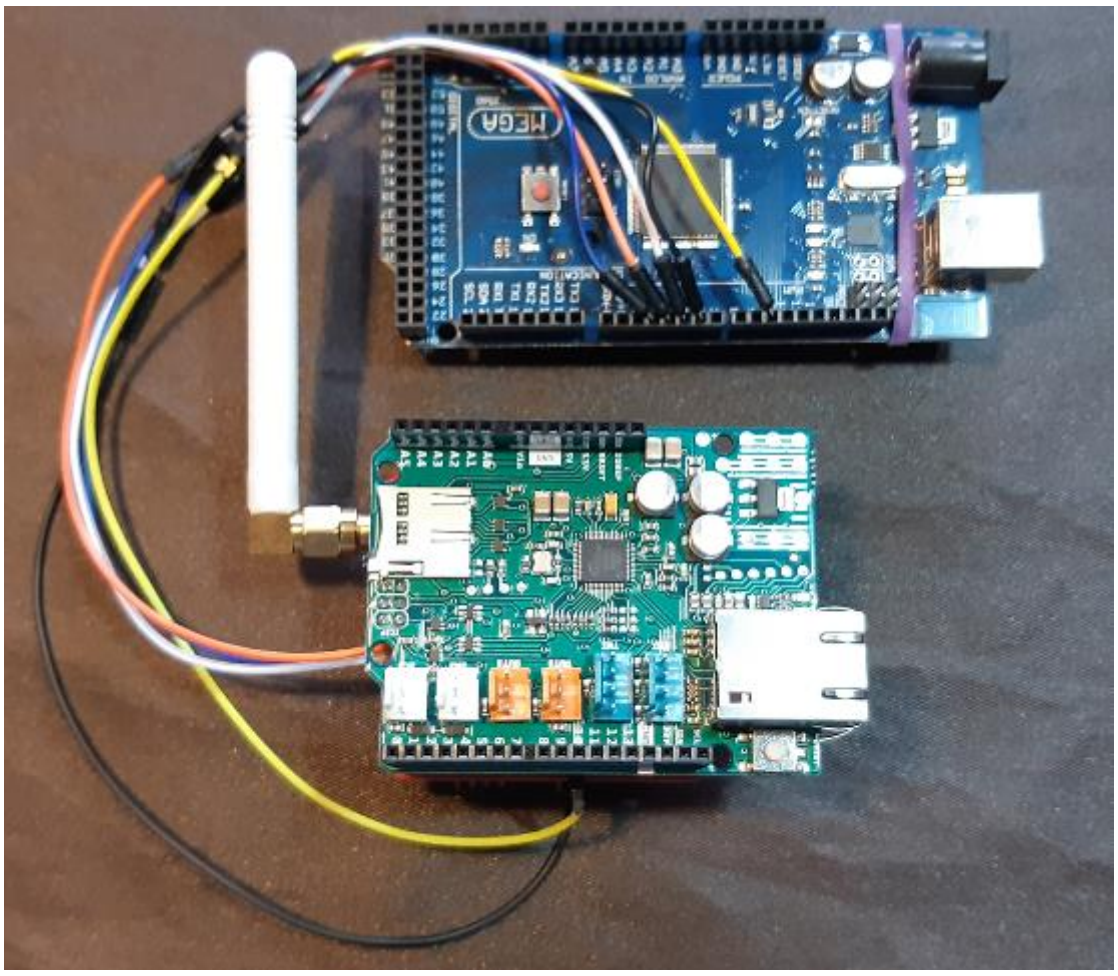
D4 is used for the SD card and can only be used for general I/O if the SD slot is not occupied.

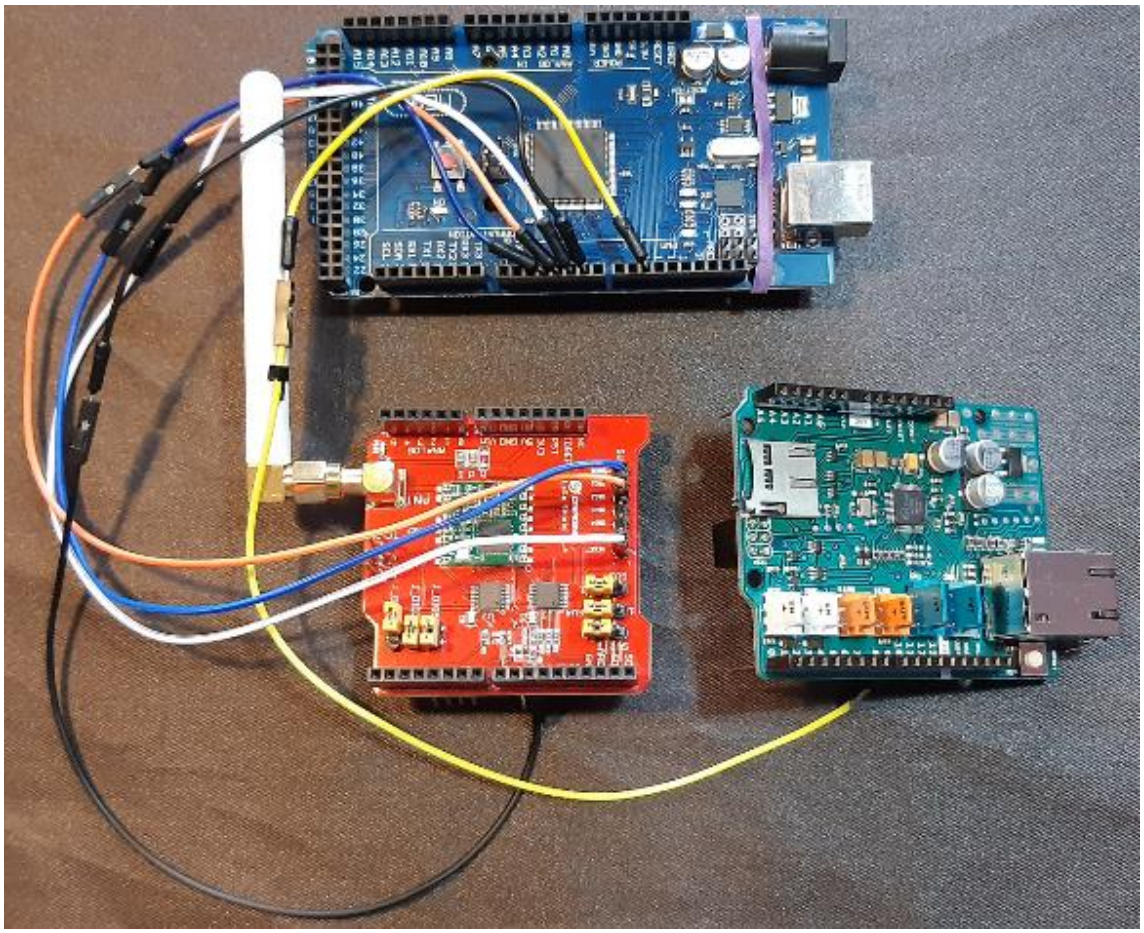
D2 is used if a solder bridge is placed across the "INT" pads to connect it to the W5100's INT pin.

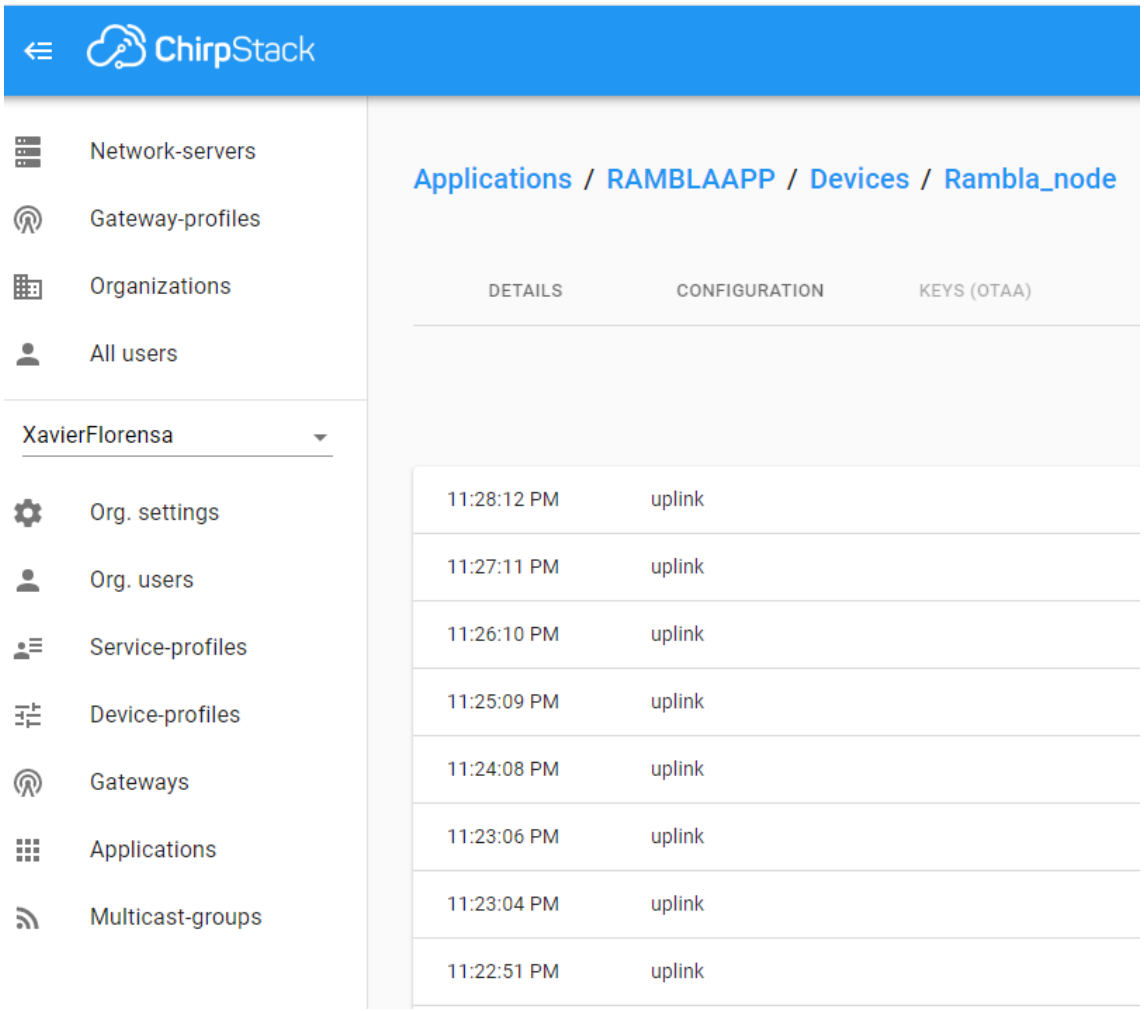
On the Mega, the hardware SS pin, D53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

Like this




MEGA





And it Works ¡!!!

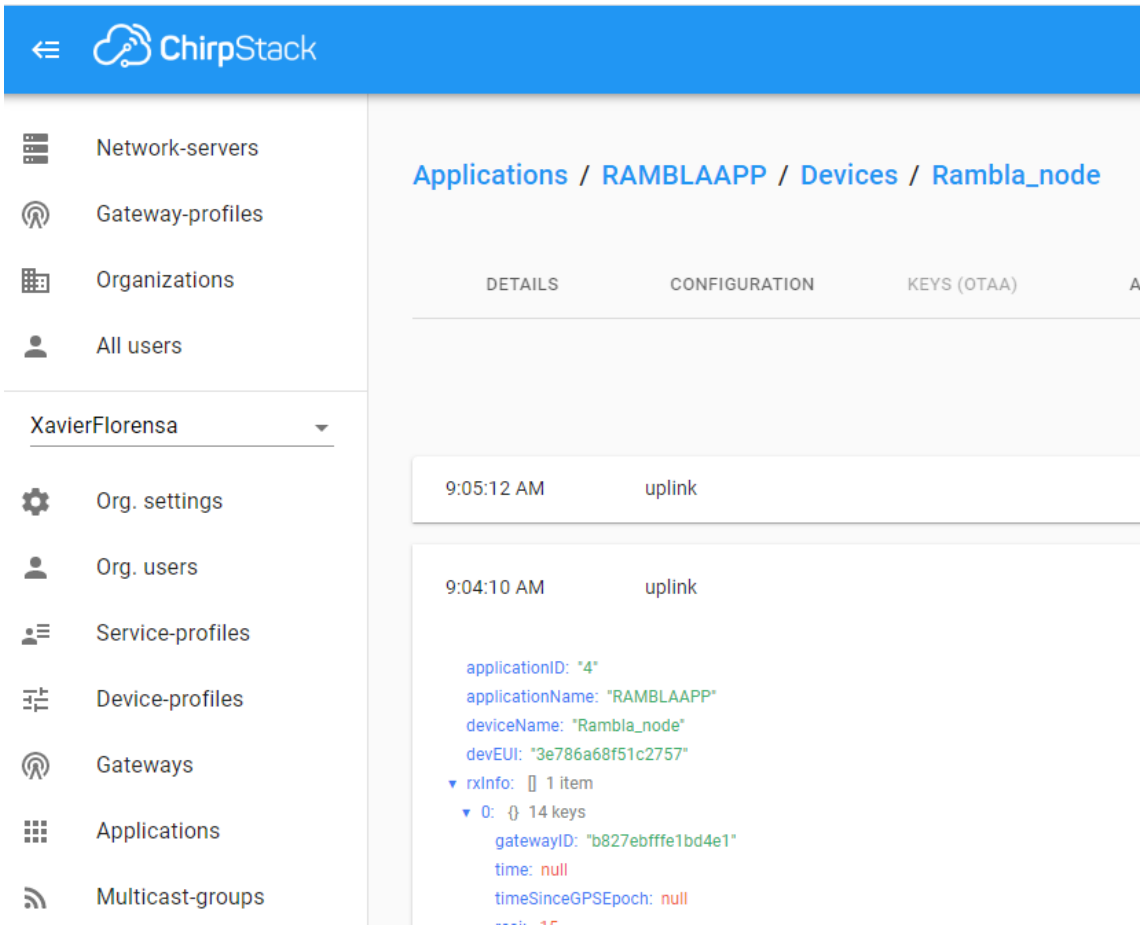
Next step, Ethernet Shield on the node

ETHERNET SHIELD

COMPATIBILITY TEST

Let’s insall Ethernet shield which is normally incompatible with Draguino LoRa due to pin 10, and see if both boards are able to exist together. And that LoRa does not stop working.

So it Works



16 bit INT TRANSMIT TEST

Now let's try to transmit a 16 bit number instead of "Hello World", for example number 85

Let`s modify the standard sketch

https://github.com/matthijskooijman/arduino-lmic/tree/master/examples/ttn-abp

before we just had

```
static uint8_t mydata[] = "Hello, world!";
```

Now we will add the PLC variable and convert as a string

```
int PLCMW100 = 85;
//static uint8_t mydata[] = "Hello, world!";
static uint8_t mydata[3];
char* formato="%i";
```

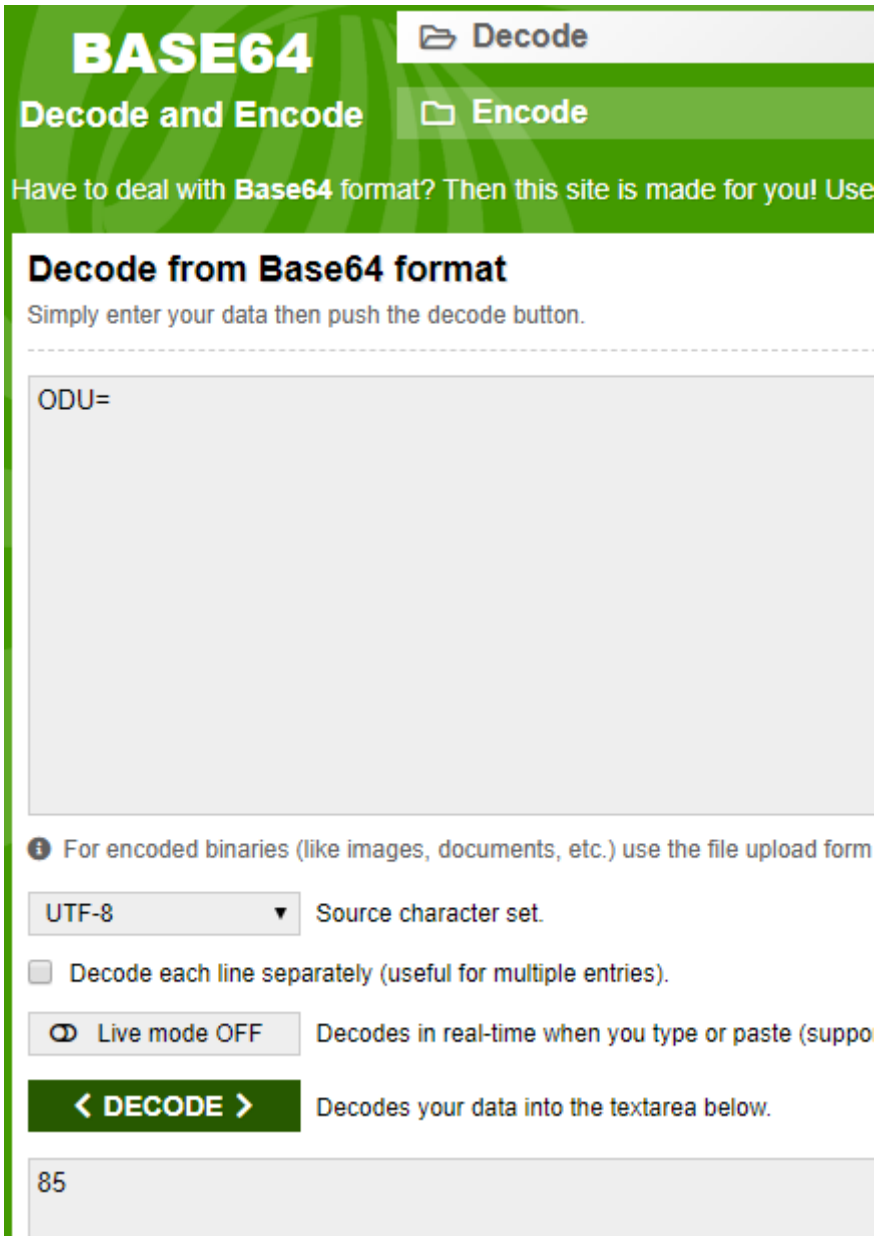
And inside setup() for example we start with the conversión from integer to string

You can not place this sentence outside of a function

```
void setup() {
    sprintf(mydata, formato, PLCMW100);
    Serial.begin(115200);
```

Yes, it Works, and if we take a look at the payload

```
    polarizationInversion: false
adr: true
dr: 5
fCnt: 0
fPort: 1
data: "ODU="
objectJSON: ""
tags: {} 0 keys
```



But the contents of PLCMW100 will be changing

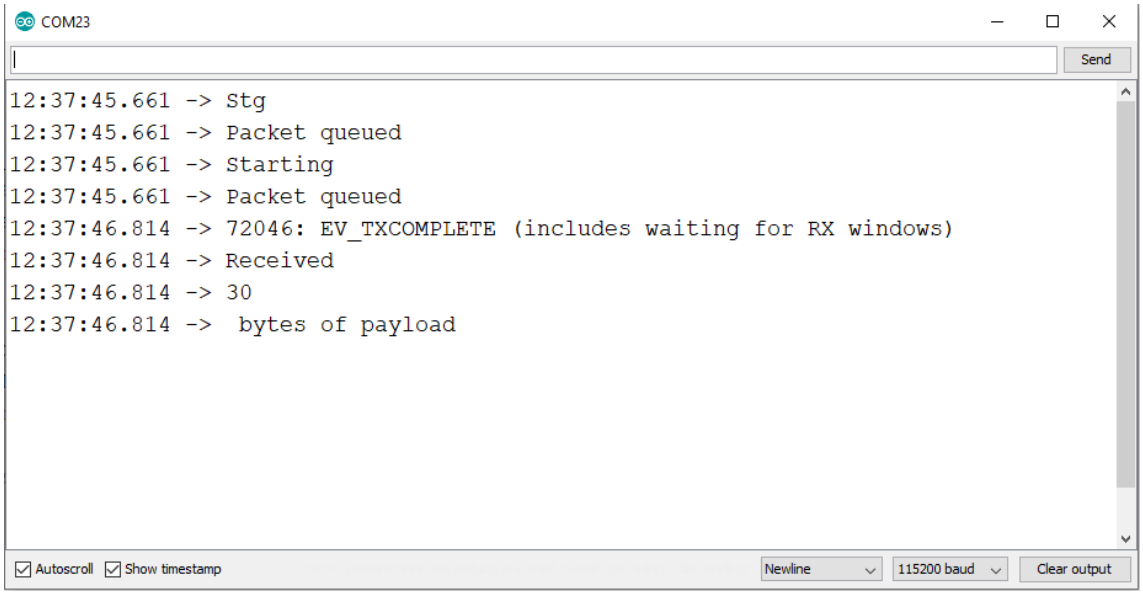
So we will place the conversión sentence somewhere else

Here is a nice place

```
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
        sprintf(mydata, formato, PLCMW100);
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
```

And still working

The name of the modified sketch is Nodo_Rambla_16bit_int.ino from 9/5/2020



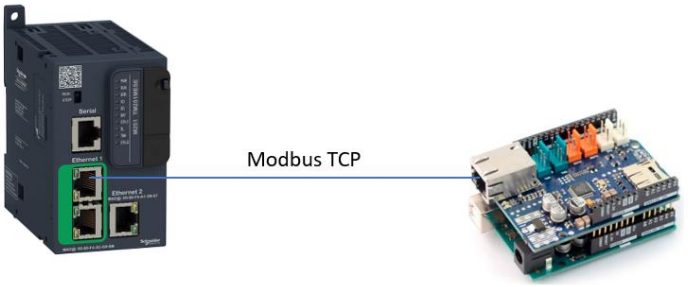
Now we forget about Lora and let's concentrate on the Modbus TCP and skecth to get data from PLC

CONNECTING PLC ON MODBUS TCP TO ARDUINO

Vamos a hacer una prueba de pasar datos del PLC al Arduino en Modbus TCP.

Es suficiente conectar directamente el PLC con el shield ethernet de Arduino mediante un Patchcord.

Asegurarse de usar el shield con chip W5500 (Con el W5100 no tomara la dirección IP que le designemos en el sketch y toma la 0.0.0.0)



192.168.1.58
PLC M251

192.168.1.57
Arduino + Ethernet shield

Asignamos direcciones IP

Maestro PLC Schneider

192.168.1.58

Esclavo Arduino

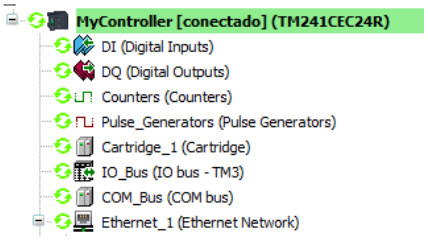
192.168.1.57

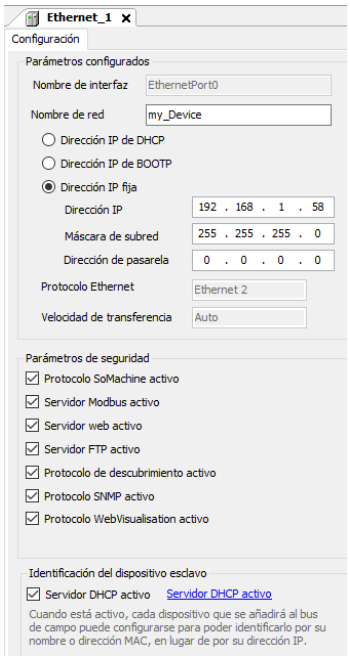
Asegurarse de usar el shield con chip W5500 (Con el W5100 no tomara la dirección IP que le designemos en el sketch)

CONFIGURACION PLC

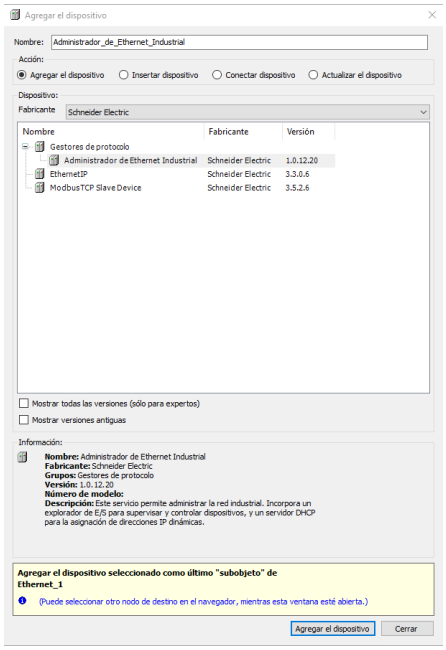
Vamos a configurar nuestro PLC para Modbus TCP IO Scanner

En SOMACHINE vamos a Dispositivos, para introducir la dirección IP del PLC

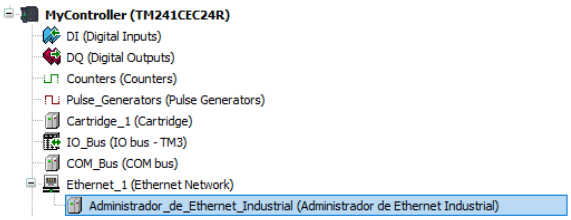




A continuación añadimos mediante el signo + en verde, el administrador de ethernet Industrial

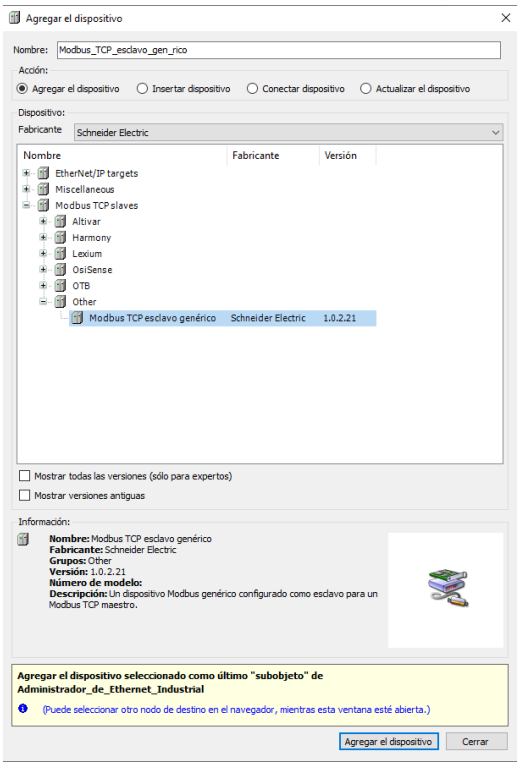


Nos quedará así

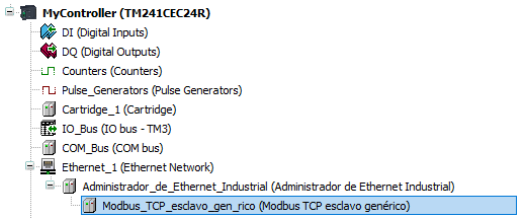


No es necesario configurar nada en este paso

Ahora añadimos el esclavo genérico Modbus (Nuestro Arduino)

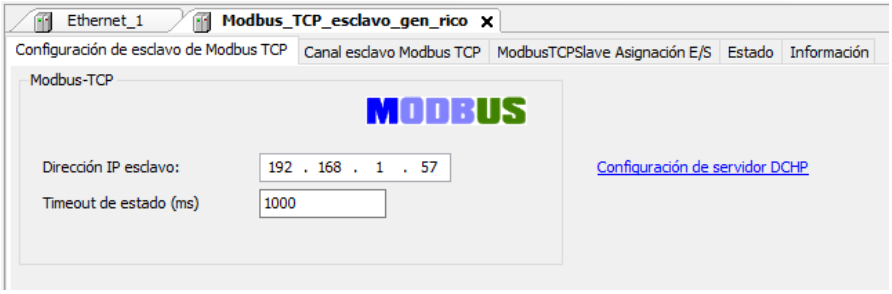


Con lo que nos quedará así:

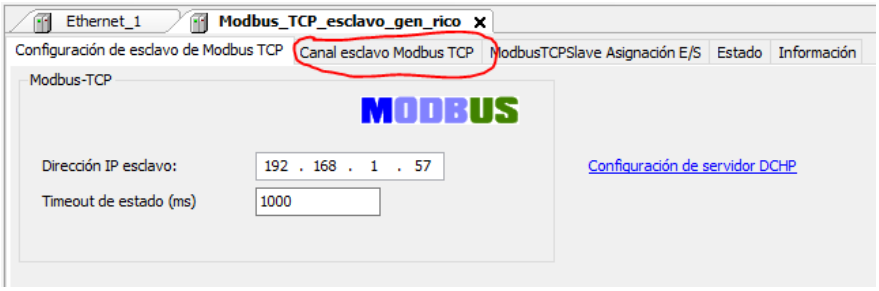


Ahora vamos a configurar el esclavo (nuestro Arduino) haciendo doble click

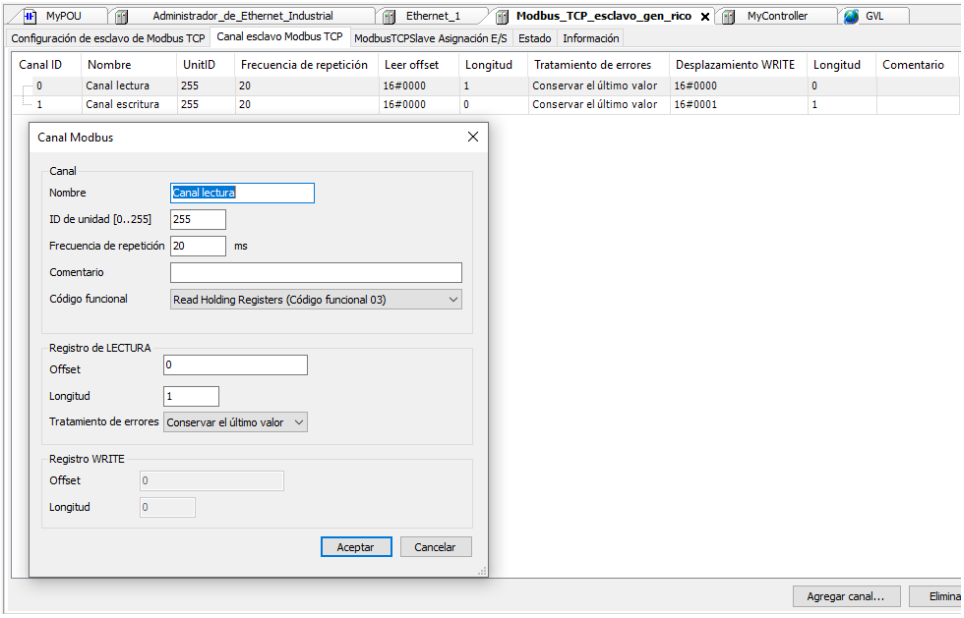
E introducimos la dirección IP de nuestro Arduino



Ahora definimos 2 canales uno para lectura y otro para escritura en esta pestaña

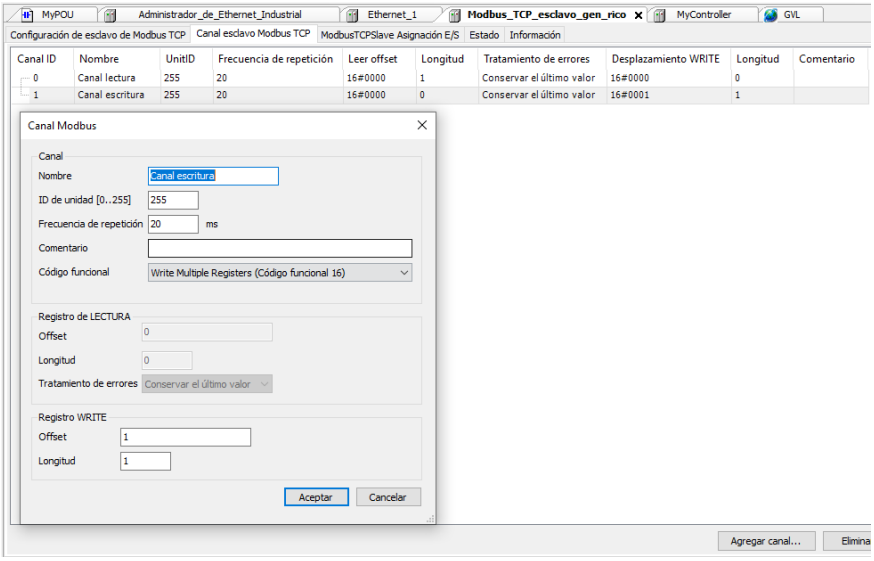


Agregamos un canal de lectura con un offset y un numero de palabras (1 word).

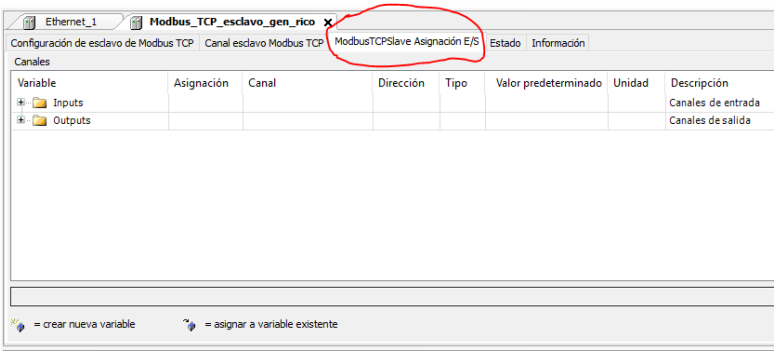


Y un canal de escritura, en una posición distinta pues el buffer que usa Arduino para la comunicación Modbus es el mismo tanto para entradas de datos como para salidas de datos.

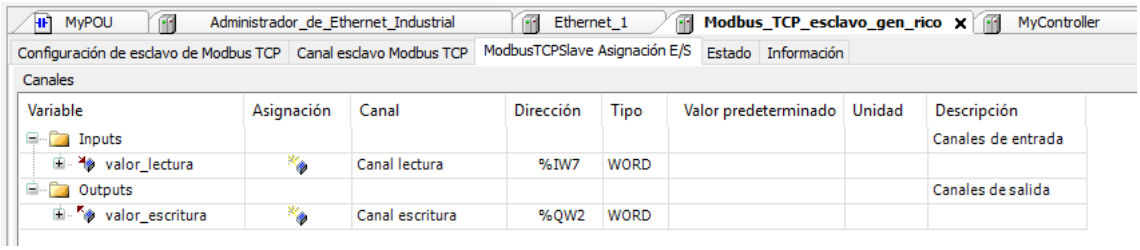
También asignamos 1 word.



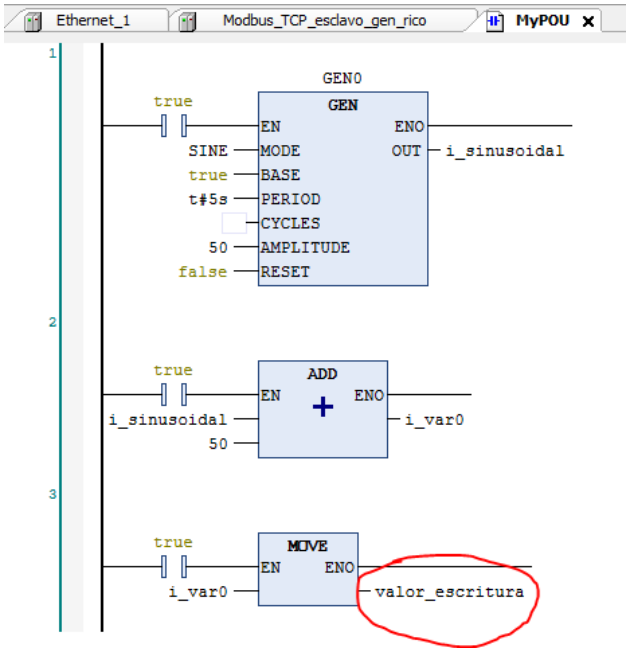
Podemos a su vez declarar variables para usar en nuestro programa de PLC mediante esta pestaña



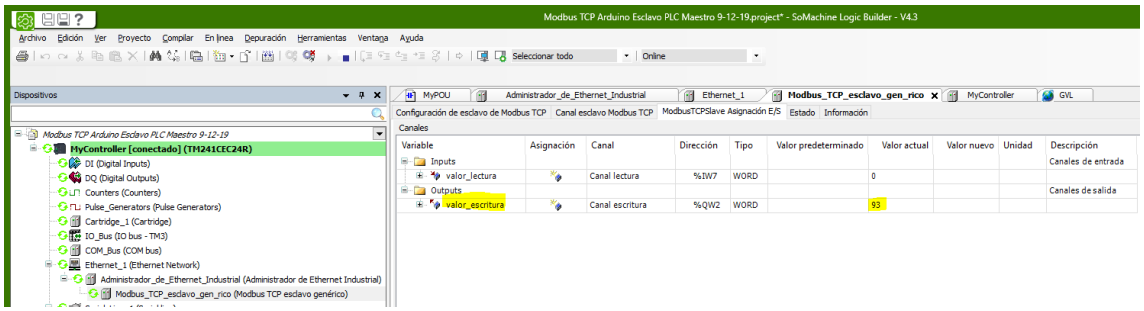
Lo vemos en detalle



Vamos a construir un programa para que vaya cambiando los valores de esa variable “valor_escritura”. Mediante una función sinusoidal que fluctúe en el tiempo, para simular la evolución de una variable de proceso.



Vemos cómo va evolucionando es labor de nuestra variable

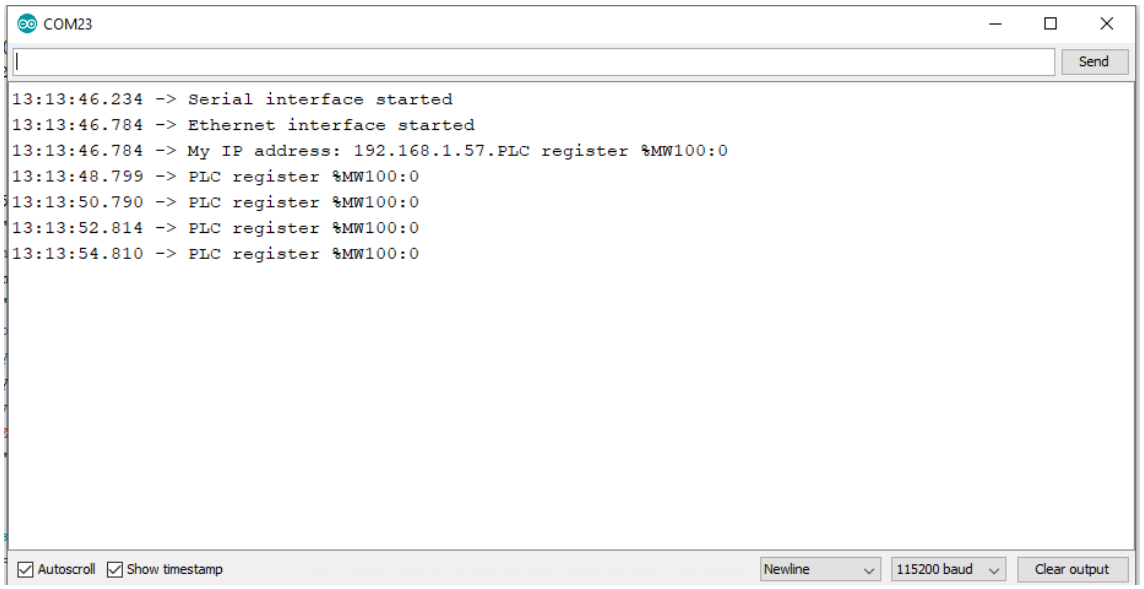


Con esto ya tendríamos todo preparado en el lado del PLC

PLC TEST on MODBUS TCP

Let's try with a PLCM251, before uploading the PLC programm

But let's change the terminal speed on the sketch to 115200 bauds, wich is the one used on the LMIC sketch



So we start with this sketch

Modbus_Rambla | Arduino 1.8.10

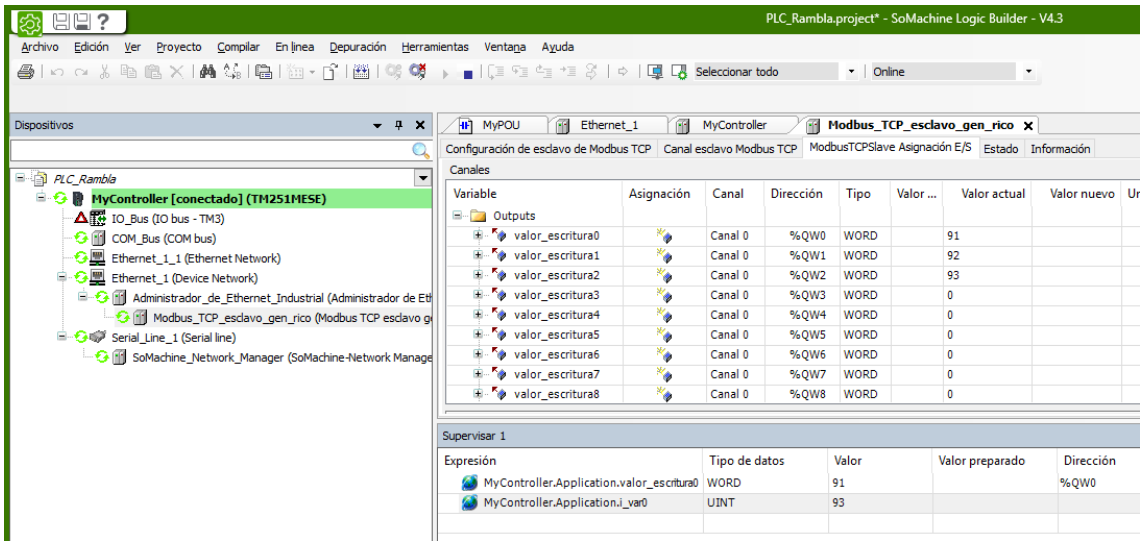
File Edit Sketch Tools Help

Modbus_Rambla § | MgsModbus.cpp | MgsModbus.h

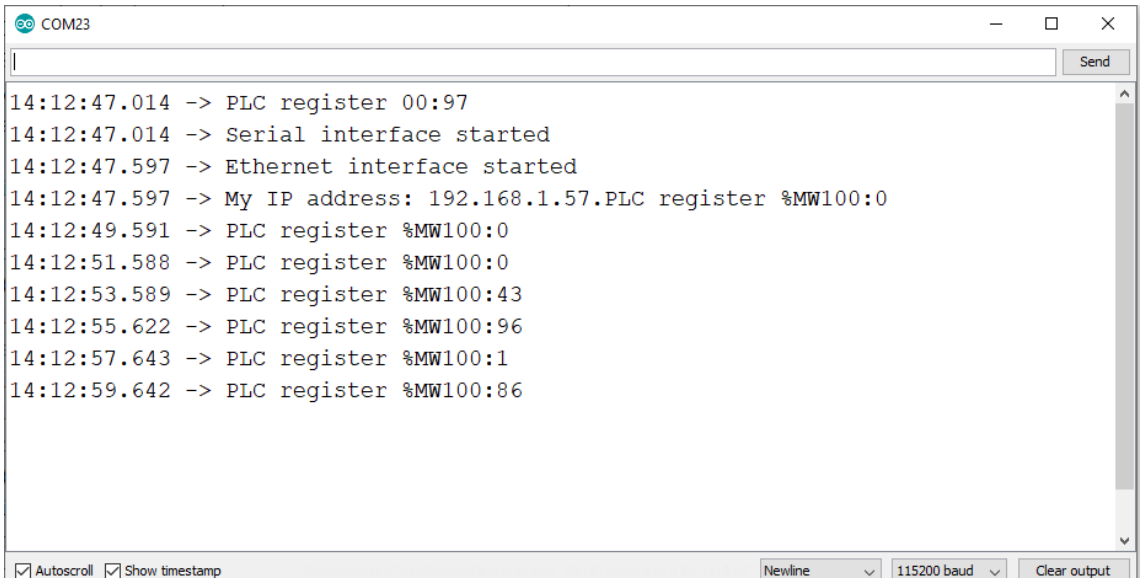
```
#include <SPI.h>
#include <Ethernet.h>
#include "MgsModbus.h"
MgsModbus Mb;
// Ethernet settings (depending on MAC and Local network)
byte mac[] = {0xA8, 0x61, 0x0A, 0xAE, 0x4E, 0x73 };
IPAddress ip(192, 168, 1, 57);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);

 void setup(void)
{
  // serial setup
  Serial.begin(115200);
  Serial.println("Serial interface started");
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet);   // start ethernet interface
  Serial.println("Ethernet interface started");
  // print your local IP address:
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  // Fill MbData
  Mb.SetBit(0,false);
  Mb.MbData[100] = 0;
}
void loop()
{
  Mb.MbsRun();
  Serial.print("PLC register %MW100:");
  Serial.println(Mb.MbData[100]);
  delay (2000);
}
```

Now let's upload the PLC programm to the M251 PLC to change the values on %MW100 based on a sine wave.

Partimos de un programa de PLC que manda valores del registro %MW100 por modbus TCP, como aquí



Y que nuestro Arduino recibe correctamente con shield Ethernet



Asegurarse de usar el shield con chip W5500 (Con el W5100 no tomará la dirección IP que le designemos en el sketch)

Next step:

Introducing LoRa LMIC library to the modbus sketch, in order to send PLC data per LoRa

MIXING MODBUS TCP AND LMIC LORA

We take the previous sketch and introduce these changes:

```
#include <lmic.h>

#include <hal/hal.h>
```

Then the LoRa Keys we have from LoRa server

```
// LoRaWAN NwkSKey, network session key

// This is the default Semtech key, which is used by the early prototype TTN
```

```
// network.
static const PROGMEM u1_t NWKSKEY[16] = { yours};
// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the early prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = { yours };
// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0xyours ; // <-- Change this address for every node!
// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }
```

Then

Etc, etc.

After combining both sketches,

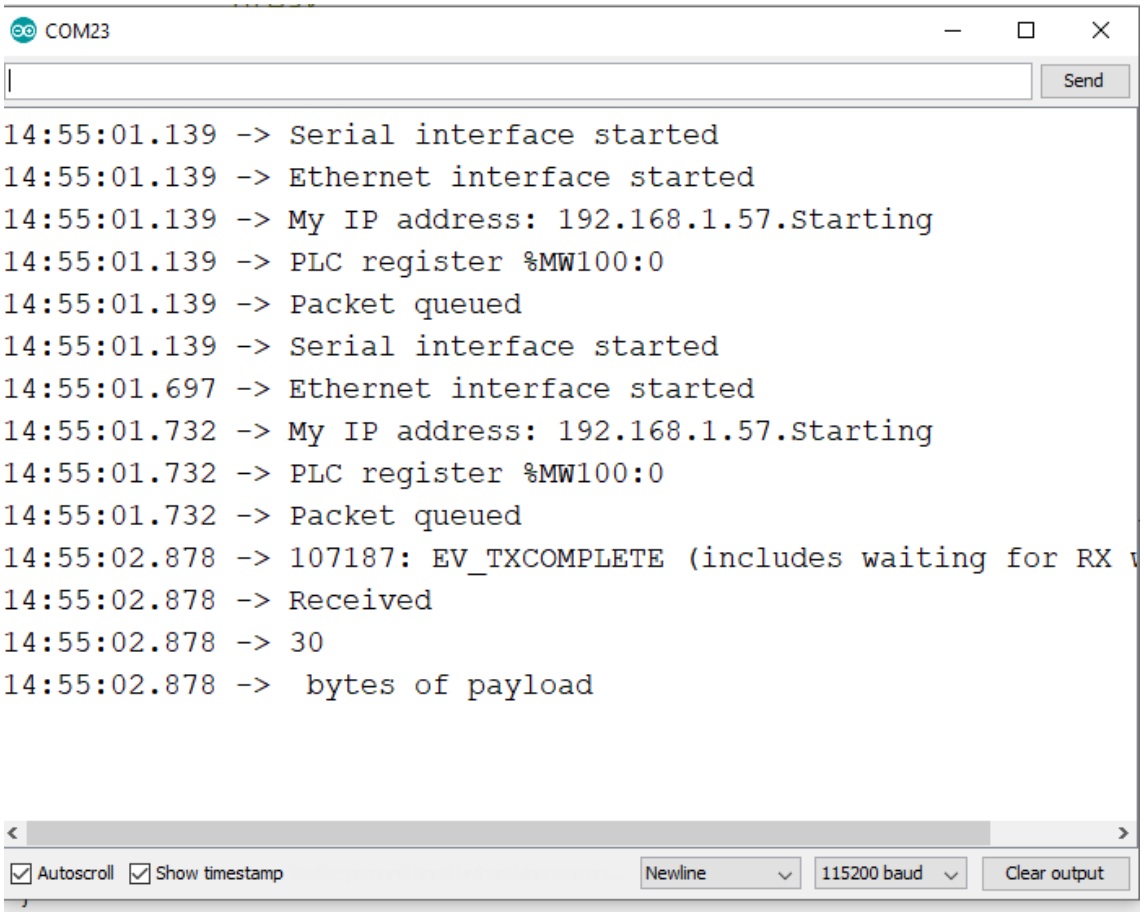
On first attemp

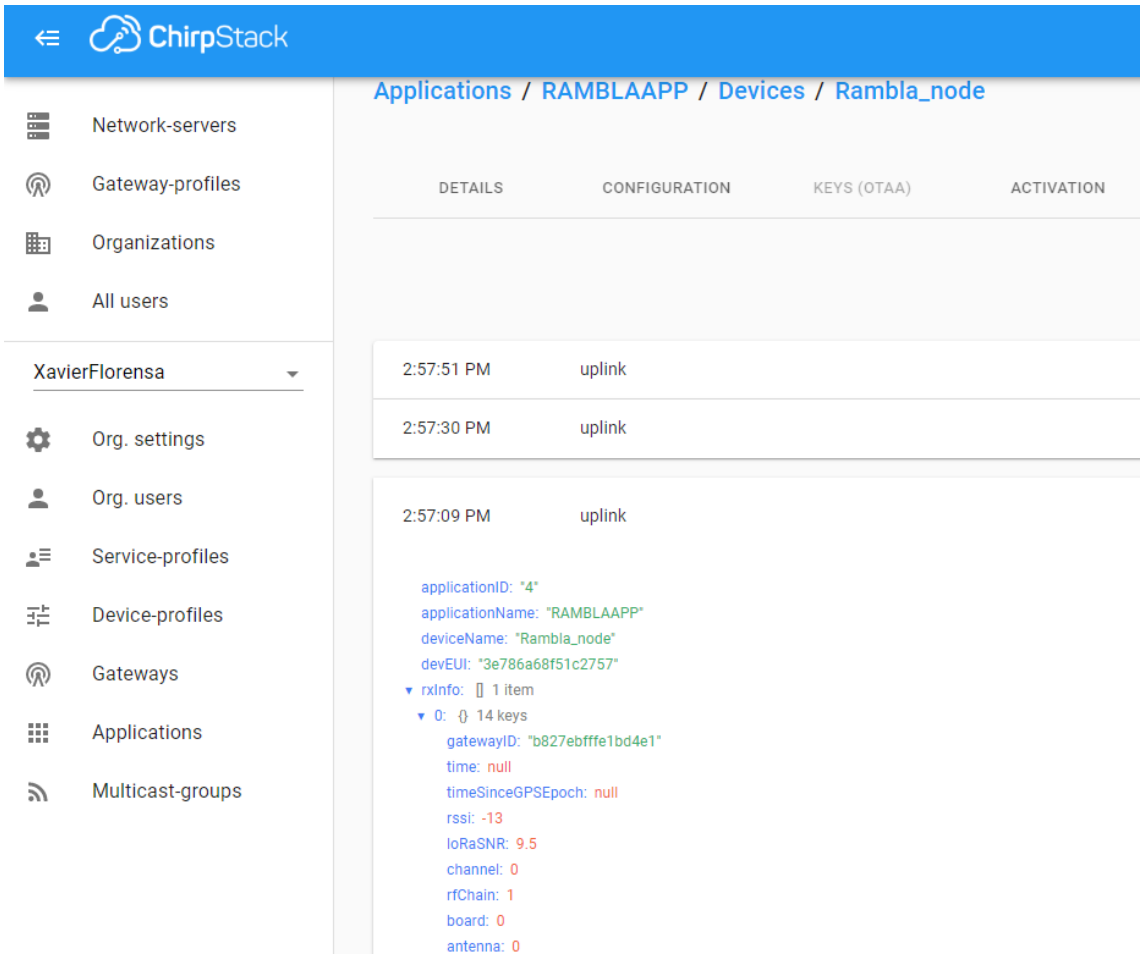
It does not work (Modbus TCP yes but none for LoRa)

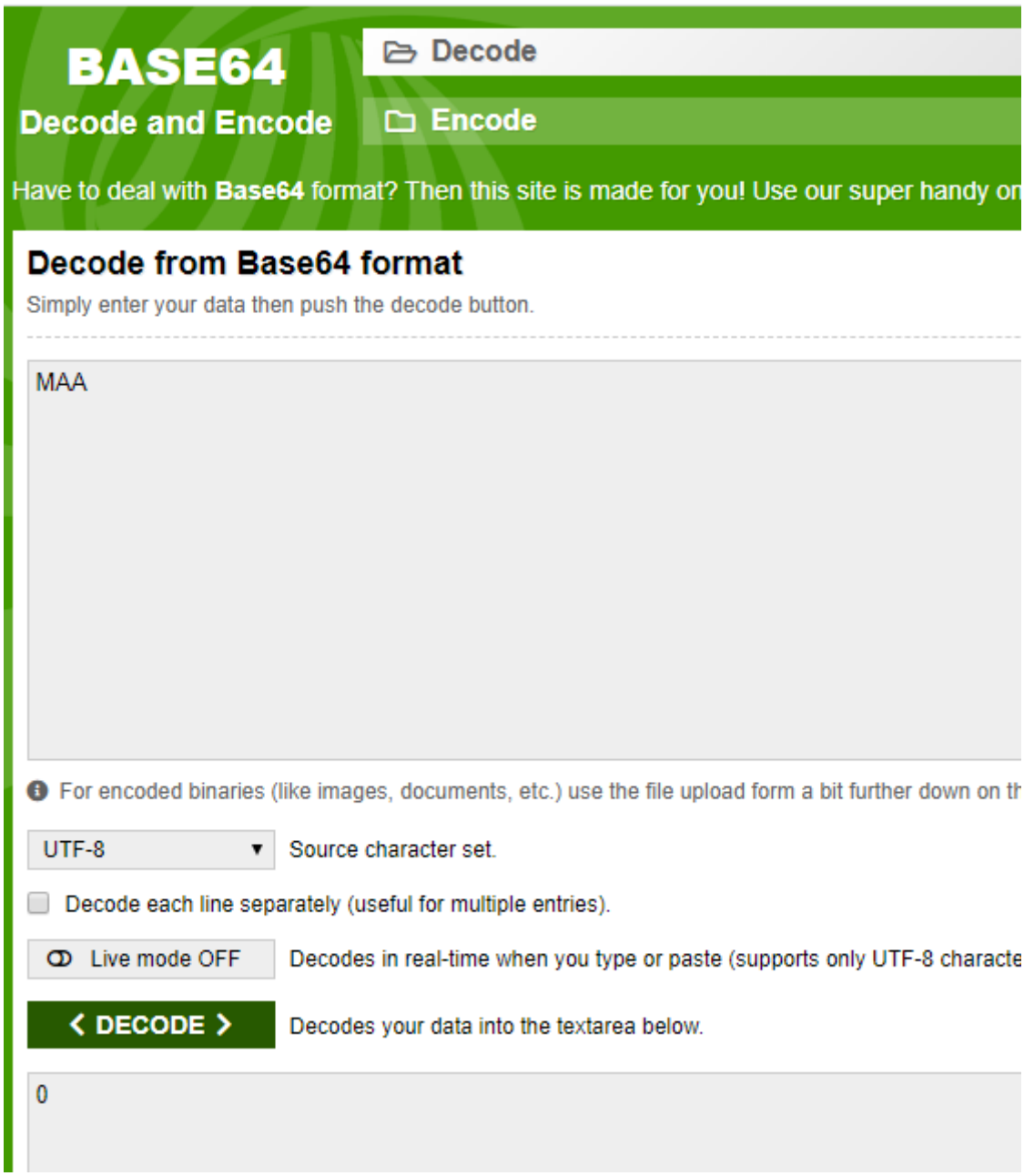
If I try to comment this line

```
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possibl
        //Mb.MbsRun();
        Serial.print("PLC register %MW100:");
        Serial.println(Mb.MbData[100]);
        sprintf(mydata, formato, Mb.MbData[100]);
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
```

Then it Works at least with 0



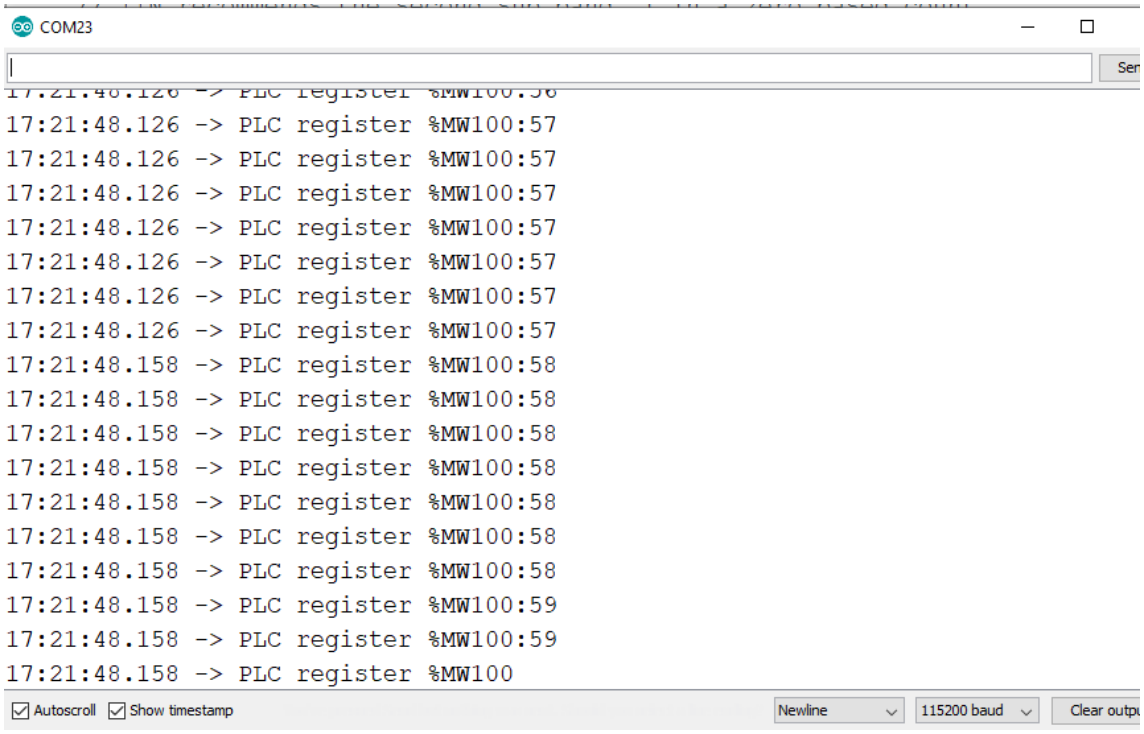

ChirpStack
Network-servers
Gateway-profiles
Organizations
All users
XavierFlorensa
Org. settings
Org. users
Service-profiles
Device-profiles
Gateways
Applications
Multicast-groups
Applications / RAMBLAAPP / Devices / Rambla_node
DETAILS
CONFIGURATION
KEYS (OTAA)
ACTIVATION
2:57:51 PM uplink
2:57:30 PM uplink
2:57:09 PM uplink
applicationID: "4"
applicationName: "RAMBLAAPP"
deviceName: "Rambla_node"
devEUI: "3e786a68f51c2757"
rxInfo: [] 1 item
0: {} 14 keys
gatewayID: "b827ebfffe1bd4e1"
time: null
timeSinceGPSEpoch: null
rssi: -13
loRaSNR: 9.5
channel: 0
rfChain: 1
board: 0
antenna: 0

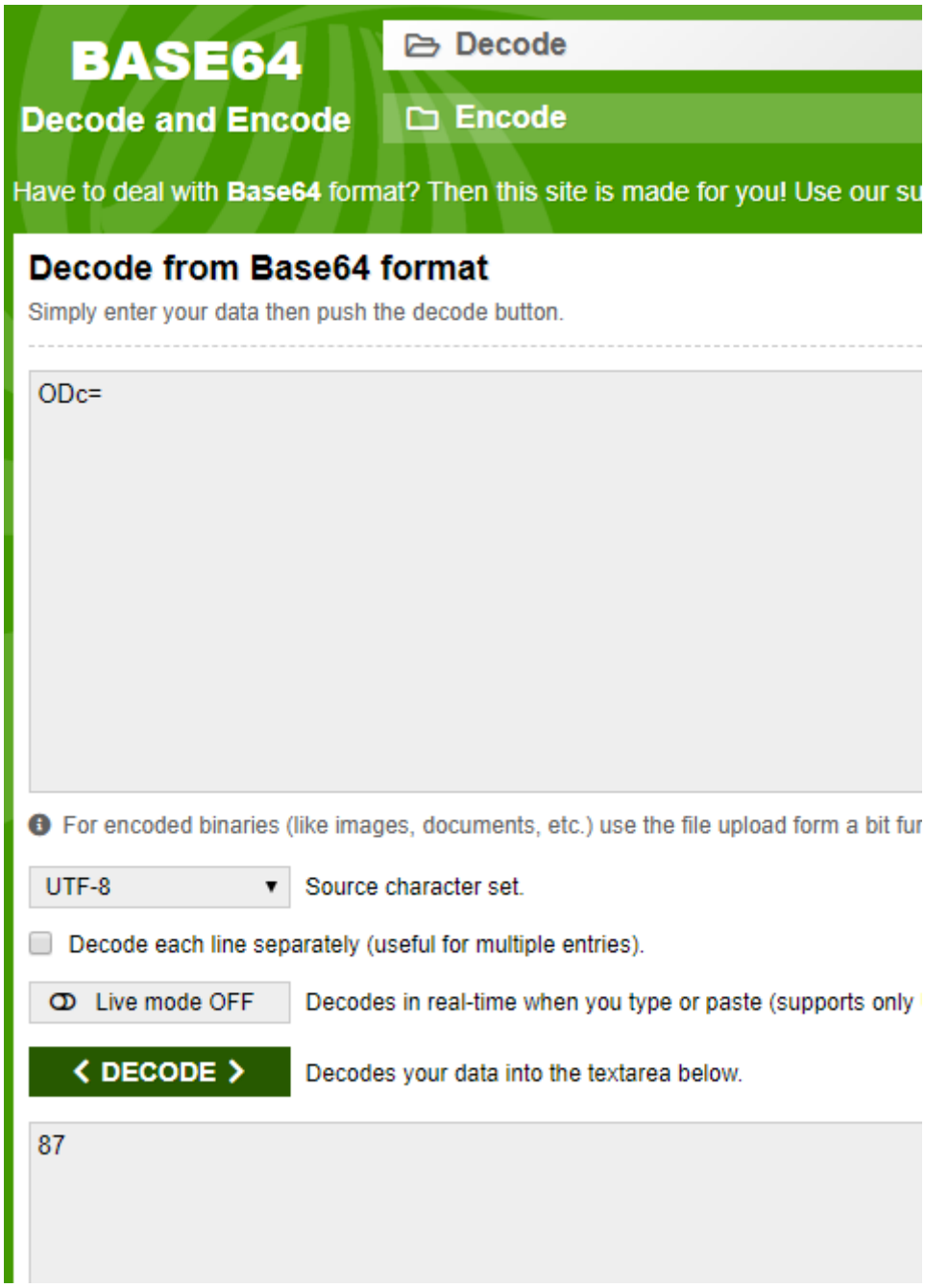


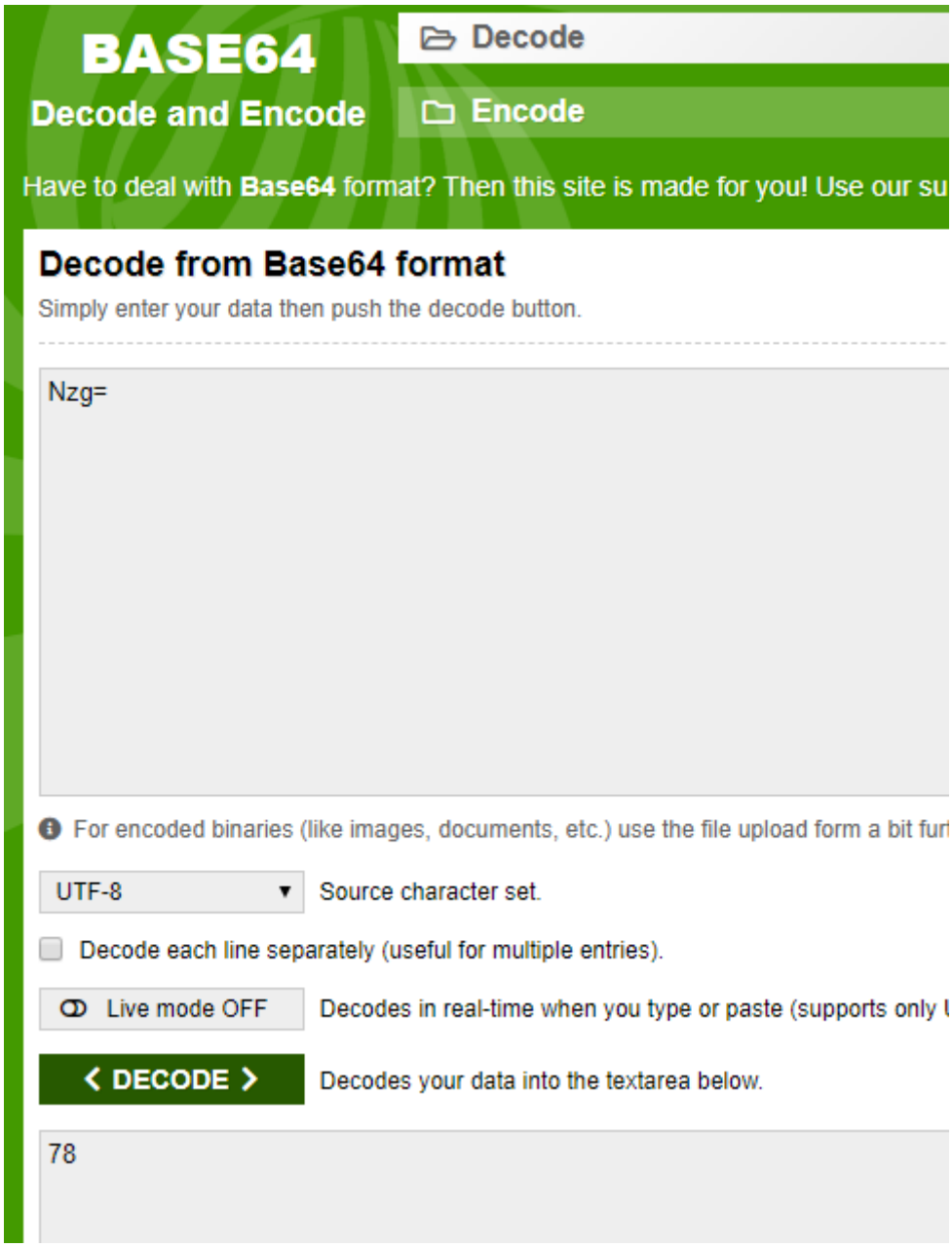
If we put the Modbus read here, it Works, but it's sending every time (there is no delay)

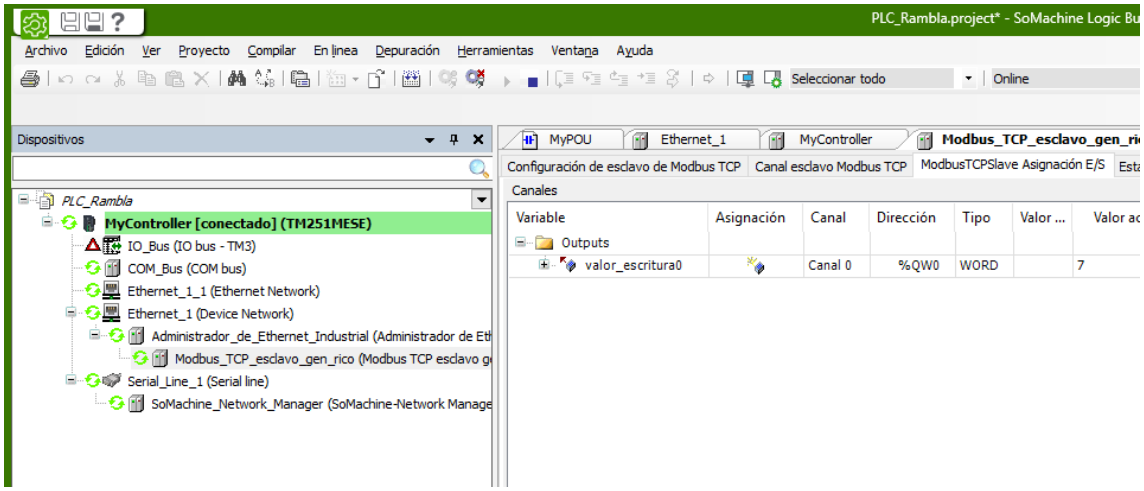
Modbus comm is Ok

```
void loop()
{
  os_runloop_once();
  Mb.MbsRun();
  Serial.print("PLC register %MW100:");
  Serial.println(Mb.MbData[100]);
}
```


COM23
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.126 -> PLC register %MW100:57
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:58
17:21:48.158 -> PLC register %MW100:59
17:21:48.158 -> PLC register %MW100:59
17:21:48.158 -> PLC register %MW100
Autoscroll
Show timestamp
Newline
115200 baud


BASE64
Decode and Encode
Decode
Encode
Have to deal with Base64 format? Then this site is made for you! Use our su
Decode from Base64 format
Simply enter your data then push the decode button.
ODc=
For encoded binaries (like images, documents, etc.) use the file upload form a bit fur
UTF-8
Source character set.
Decode each line separately (useful for multiple entries).
Live mode OFF
Decodes in real-time when you type or paste (supports only
DECODE
Decodes your data into the textarea below.
87

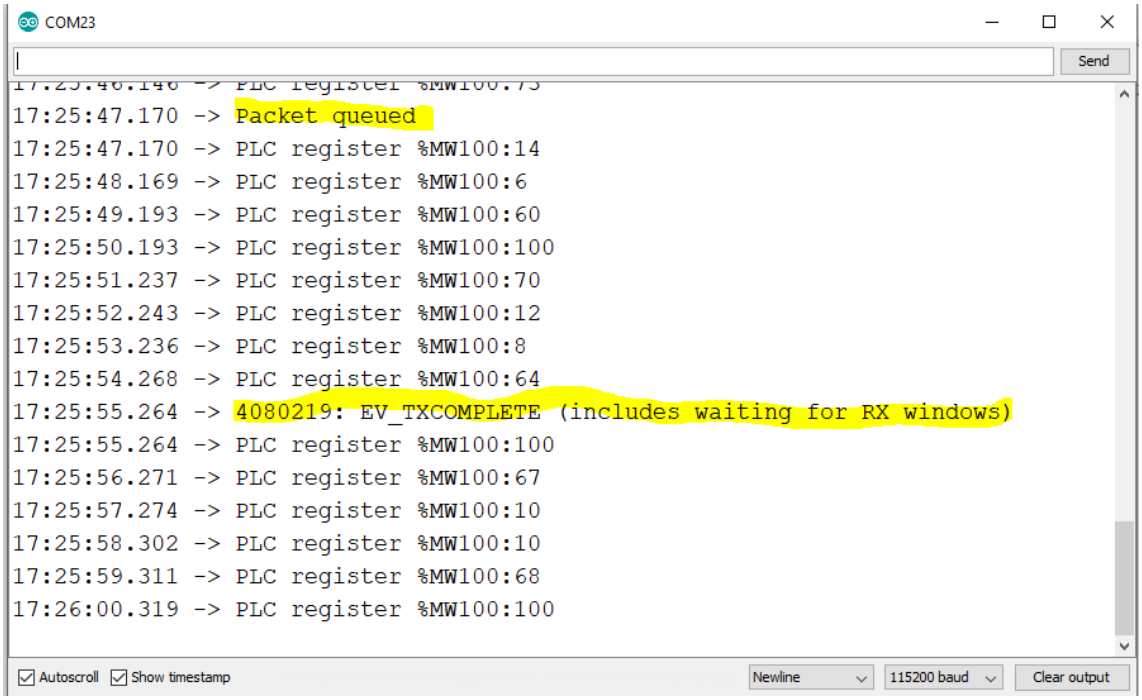


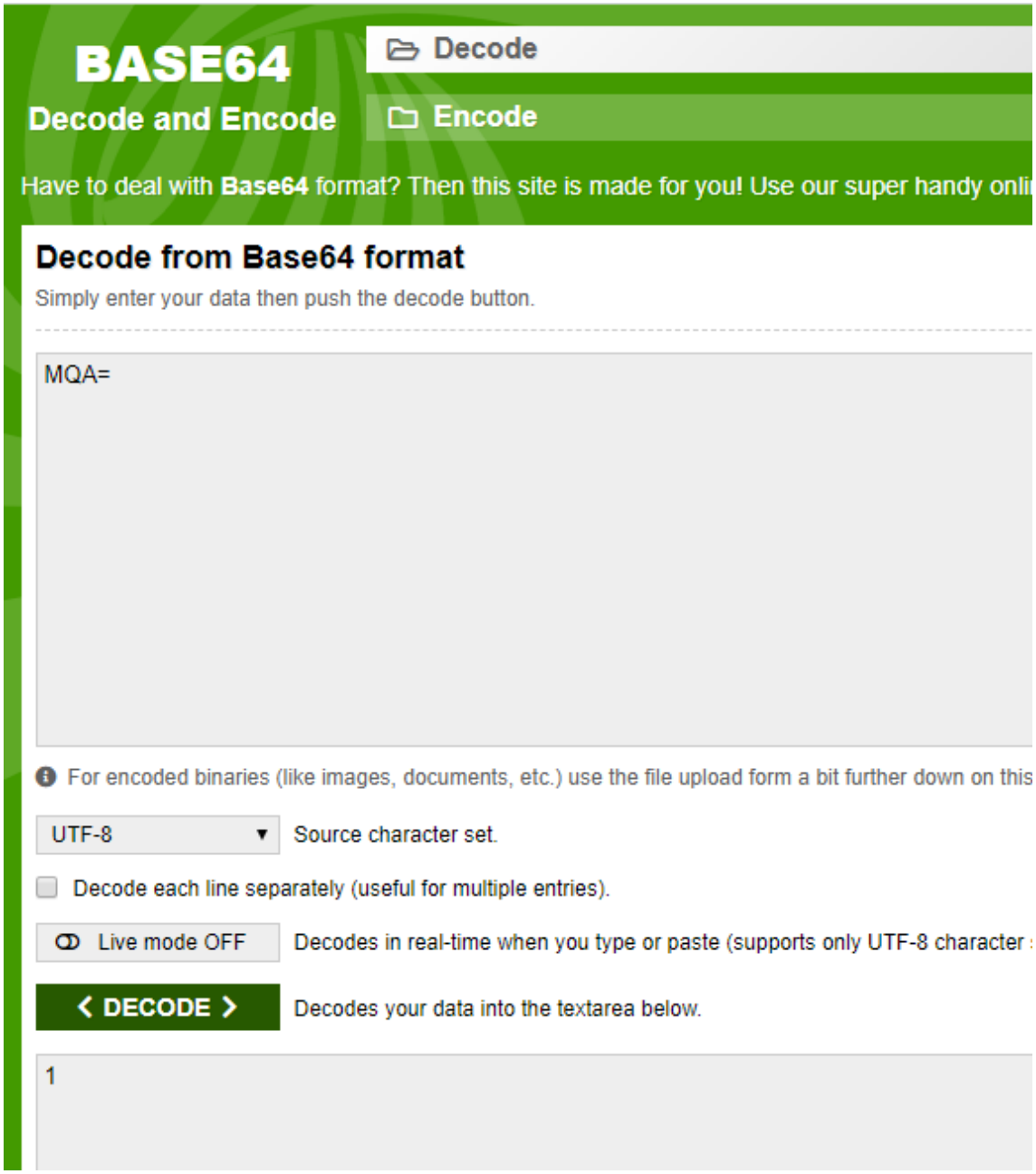


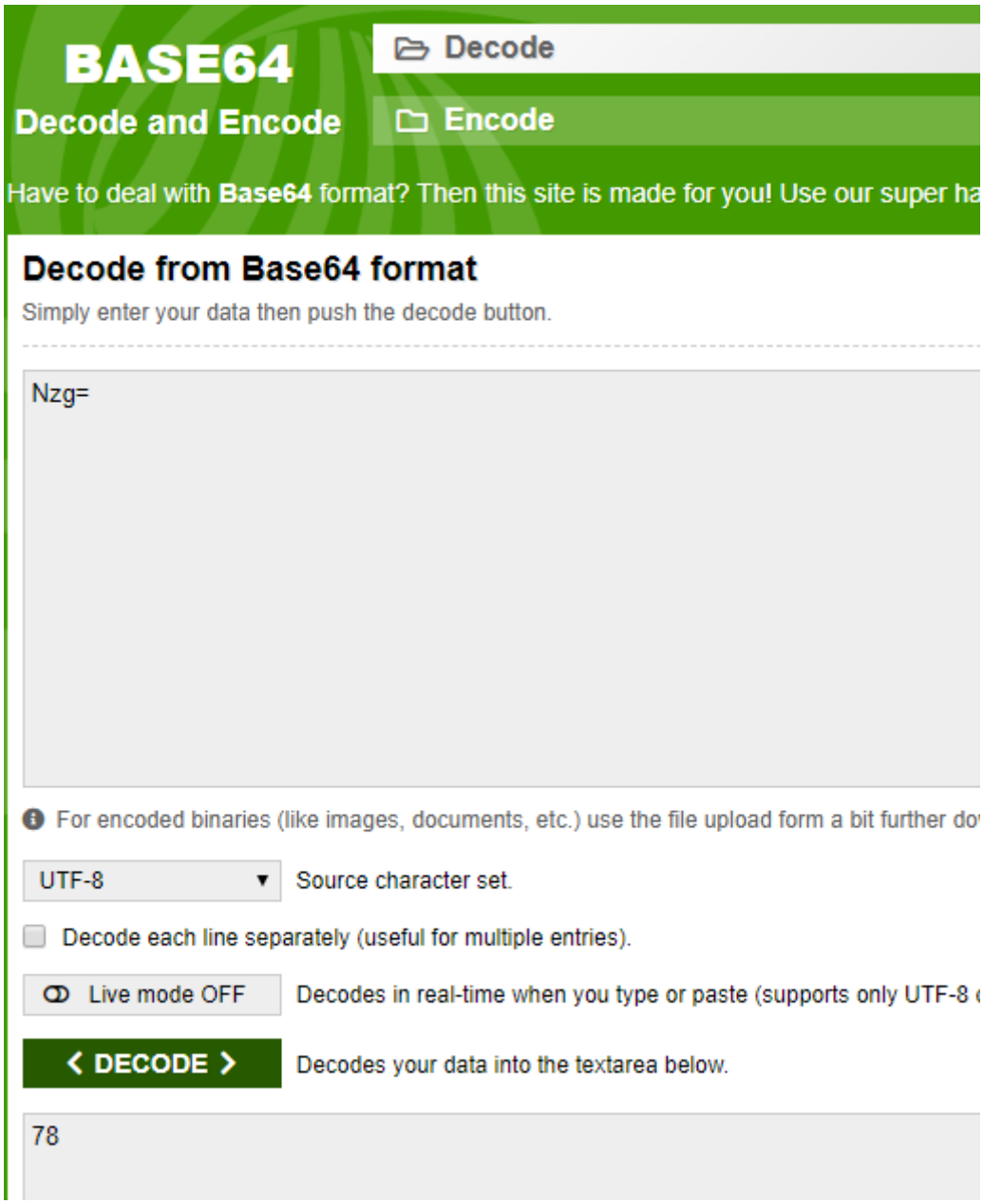
Let’s put a delay

Delay (1000)

Now ir looks better both working, Modbus and LMIC!!




BASE64
Decode and Encode
Decode
Encode
Have to deal with Base64 format? Then this site is made for you! Use our super handy onli
Decode from Base64 format
Simply enter your data then push the decode button.
MQA=
For encoded binaries (like images, documents, etc.) use the file upload form a bit further down on this
UTF-8
Source character set.
Decode each line separately (useful for multiple entries).
Live mode OFF
Decodes in real-time when you type or paste (supports only UTF-8 character
DECODE
Decodes your data into the textarea below.
1



Until now we are transmitting only one byte. And the integer number is between 1 and 100

We want to transmit now 2 bytes. A 16 bit integer.

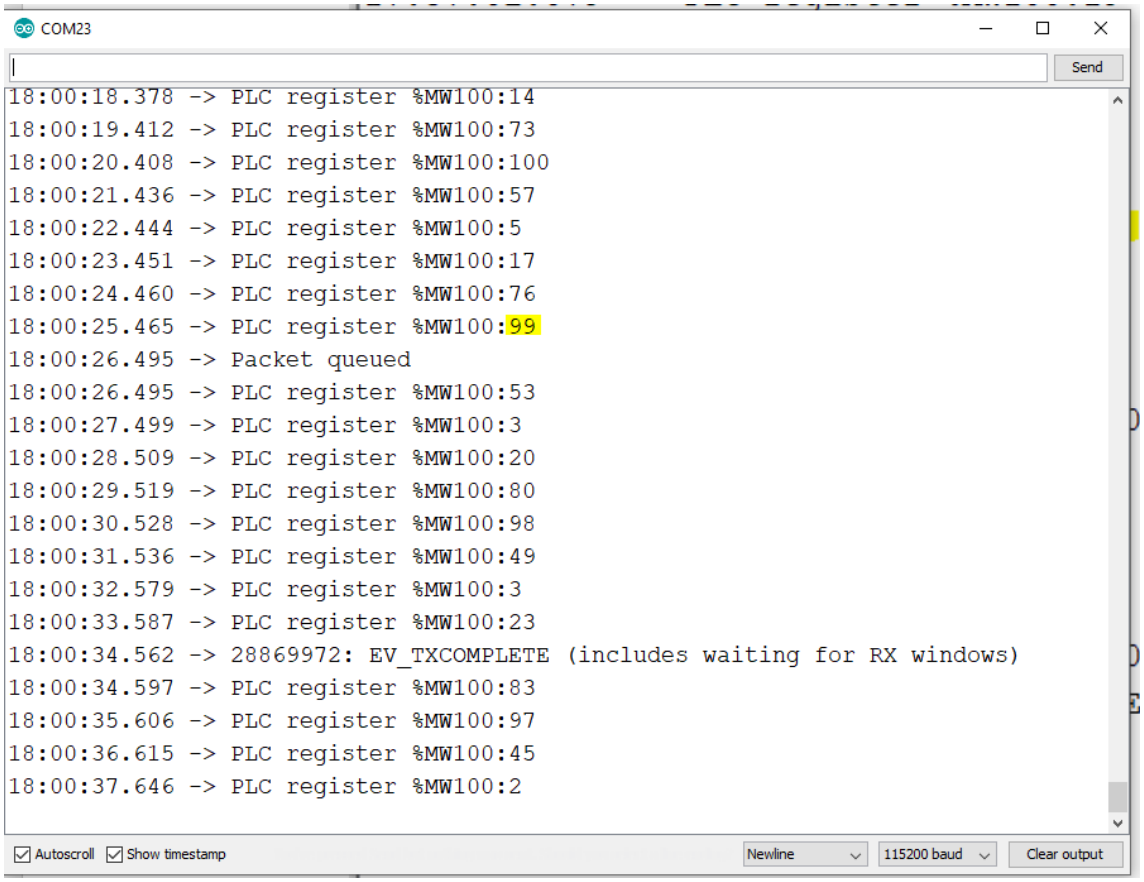
Let's change the programm on the sketch to do so.

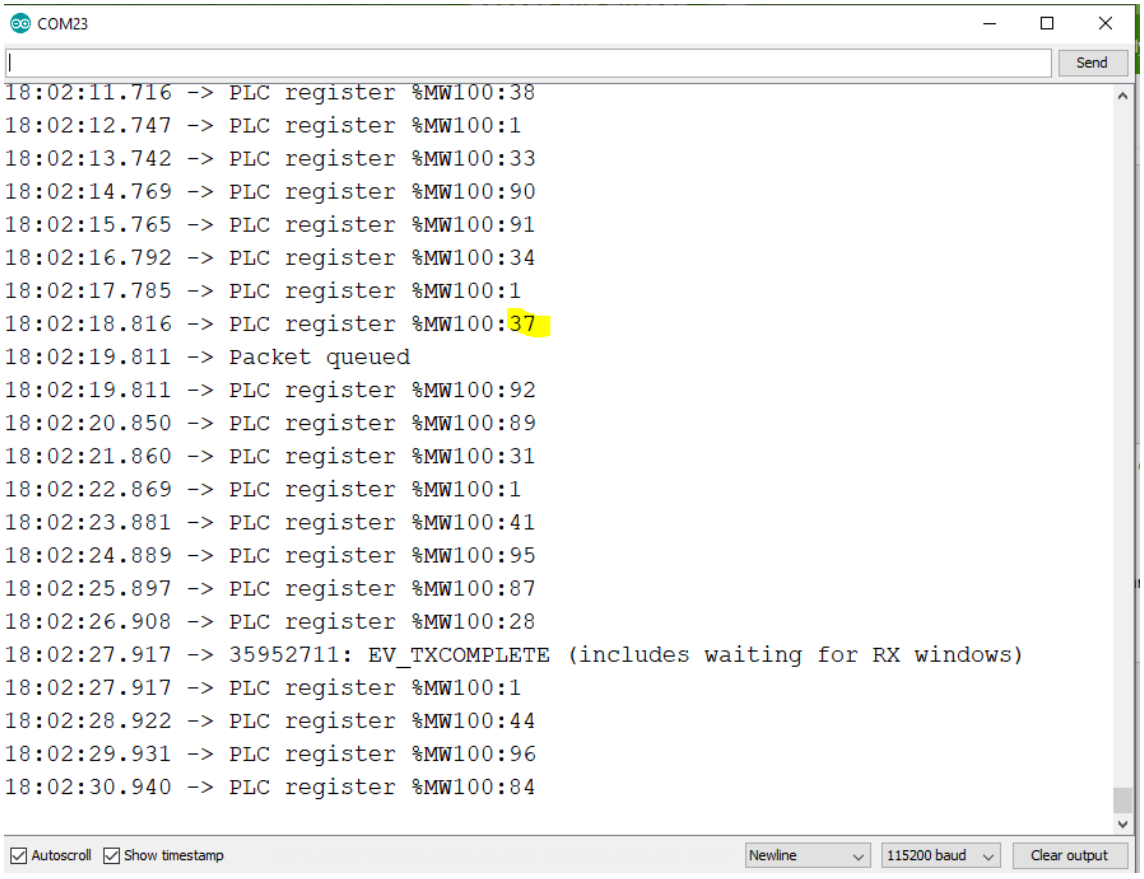
Now it Works but after decoding from Base64 we see a char

For example

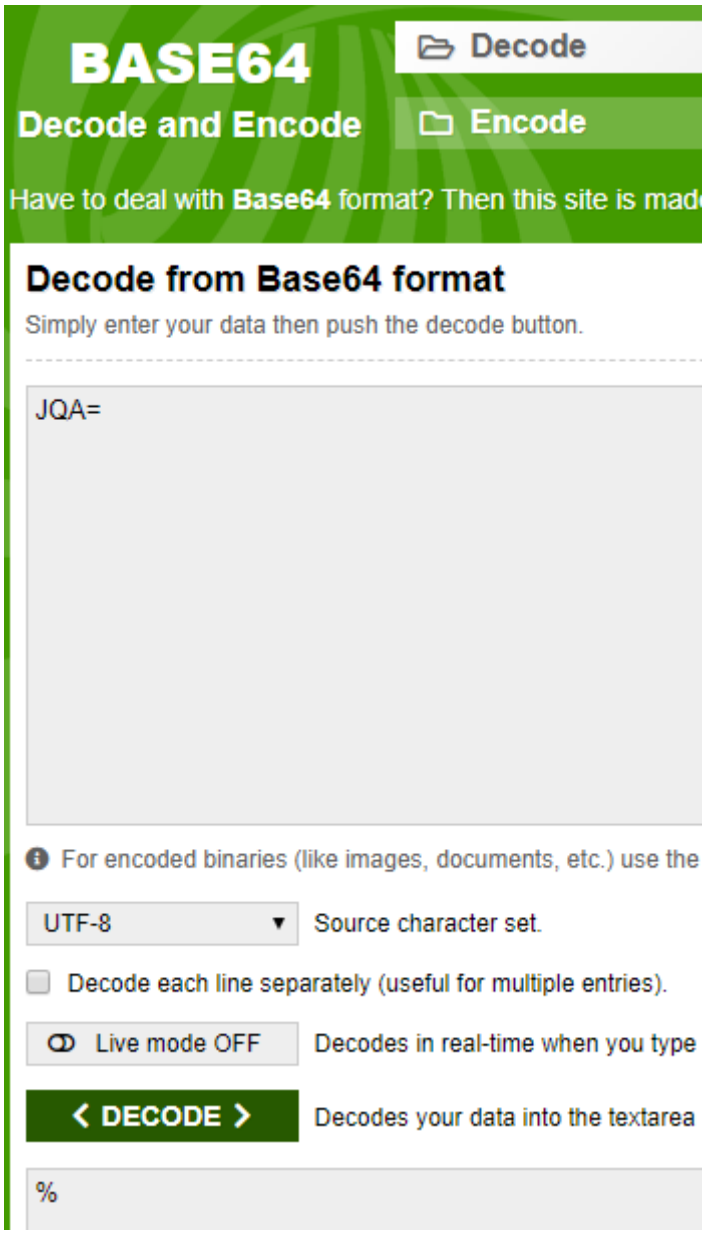
99 decimal we see a "c"

For example 99

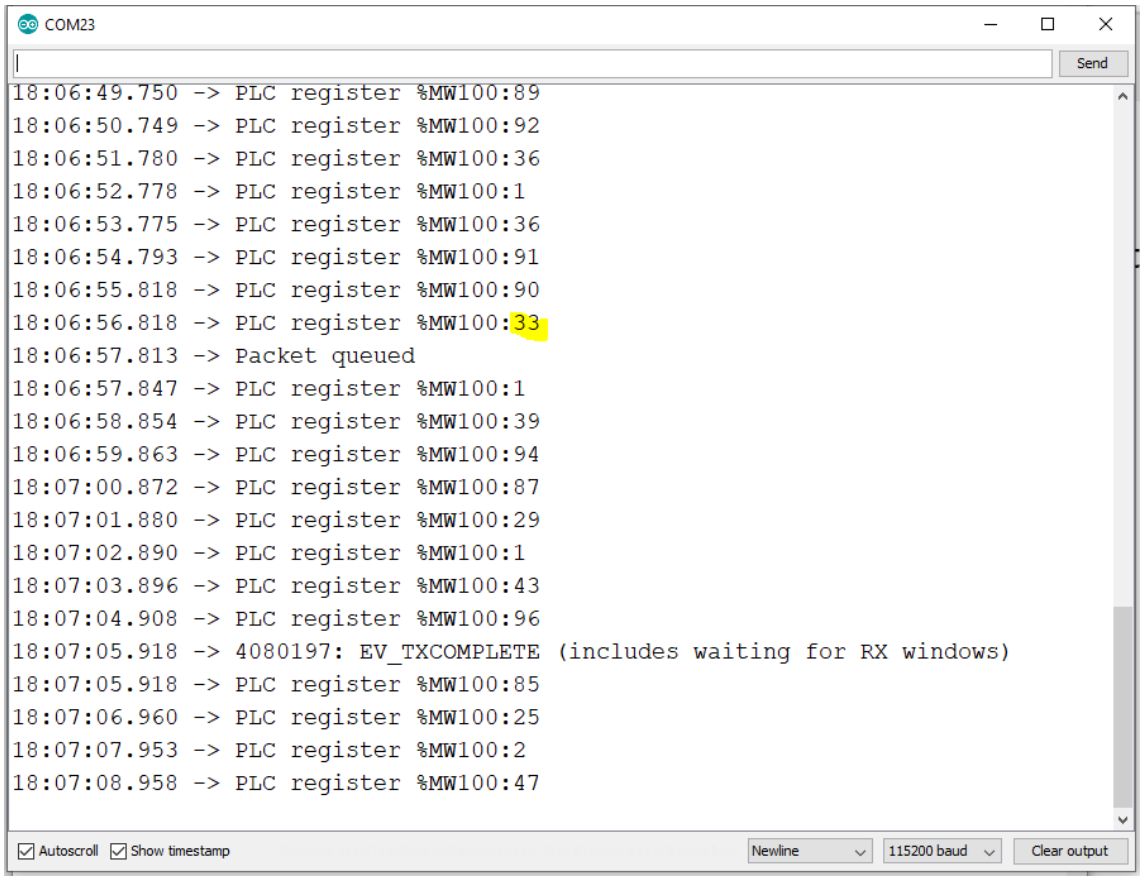

COM23
Send
18:00:18.378 -> PLC register %MW100:14
18:00:19.412 -> PLC register %MW100:73
18:00:20.408 -> PLC register %MW100:100
18:00:21.436 -> PLC register %MW100:57
18:00:22.444 -> PLC register %MW100:5
18:00:23.451 -> PLC register %MW100:17
18:00:24.460 -> PLC register %MW100:76
18:00:25.465 -> PLC register %MW100:99
18:00:26.495 -> Packet queued
18:00:26.495 -> PLC register %MW100:53
18:00:27.499 -> PLC register %MW100:3
18:00:28.509 -> PLC register %MW100:20
18:00:29.519 -> PLC register %MW100:80
18:00:30.528 -> PLC register %MW100:98
18:00:31.536 -> PLC register %MW100:49
18:00:32.579 -> PLC register %MW100:3
18:00:33.587 -> PLC register %MW100:23
18:00:34.562 -> 28869972: EV_TXCOMPLETE (includes waiting for RX windows)
18:00:34.597 -> PLC register %MW100:83
18:00:35.606 -> PLC register %MW100:97
18:00:36.615 -> PLC register %MW100:45
18:00:37.646 -> PLC register %MW100:2
Autoscroll
Show timestamp
Newline
115200 baud
Clear output


COM23
Send
18:02:11.716 -> PLC register %MW100:38
18:02:12.747 -> PLC register %MW100:1
18:02:13.742 -> PLC register %MW100:33
18:02:14.769 -> PLC register %MW100:90
18:02:15.765 -> PLC register %MW100:91
18:02:16.792 -> PLC register %MW100:34
18:02:17.785 -> PLC register %MW100:1
18:02:18.816 -> PLC register %MW100:37
18:02:19.811 -> Packet queued
18:02:19.811 -> PLC register %MW100:92
18:02:20.850 -> PLC register %MW100:89
18:02:21.860 -> PLC register %MW100:31
18:02:22.869 -> PLC register %MW100:1
18:02:23.881 -> PLC register %MW100:41
18:02:24.889 -> PLC register %MW100:95
18:02:25.897 -> PLC register %MW100:87
18:02:26.908 -> PLC register %MW100:28
18:02:27.917 -> 35952711: EV_TXCOMPLETE (includes waiting for RX windows)
18:02:27.917 -> PLC register %MW100:1
18:02:28.922 -> PLC register %MW100:44
18:02:29.931 -> PLC register %MW100:96
18:02:30.940 -> PLC register %MW100:84
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

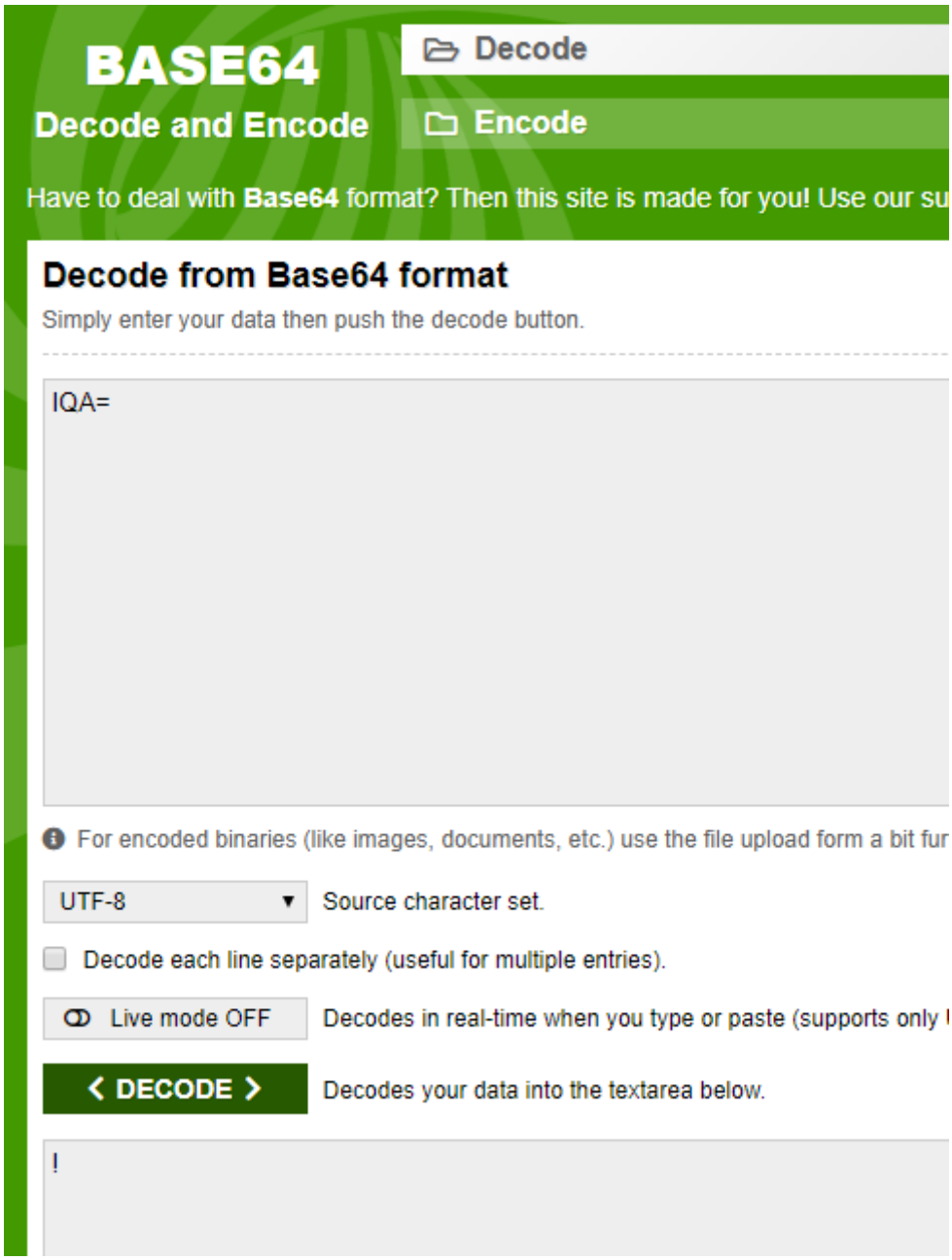
ASCII

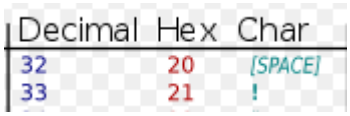


So now after decoding, we get the ascii carácter


COM23
Send
18:06:49.750 -> PLC register %MW100:89
18:06:50.749 -> PLC register %MW100:92
18:06:51.780 -> PLC register %MW100:36
18:06:52.778 -> PLC register %MW100:1
18:06:53.775 -> PLC register %MW100:36
18:06:54.793 -> PLC register %MW100:91
18:06:55.818 -> PLC register %MW100:90
18:06:56.818 -> PLC register %MW100:33
18:06:57.813 -> Packet queued
18:06:57.847 -> PLC register %MW100:1
18:06:58.854 -> PLC register %MW100:39
18:06:59.863 -> PLC register %MW100:94
18:07:00.872 -> PLC register %MW100:87
18:07:01.880 -> PLC register %MW100:29
18:07:02.890 -> PLC register %MW100:1
18:07:03.896 -> PLC register %MW100:43
18:07:04.908 -> PLC register %MW100:96
18:07:05.918 -> 4080197: EV_TXCOMPLETE (includes waiting for RX windows)
18:07:05.918 -> PLC register %MW100:85
18:07:06.960 -> PLC register %MW100:25
18:07:07.953 -> PLC register %MW100:2
18:07:08.958 -> PLC register %MW100:47
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

Yes

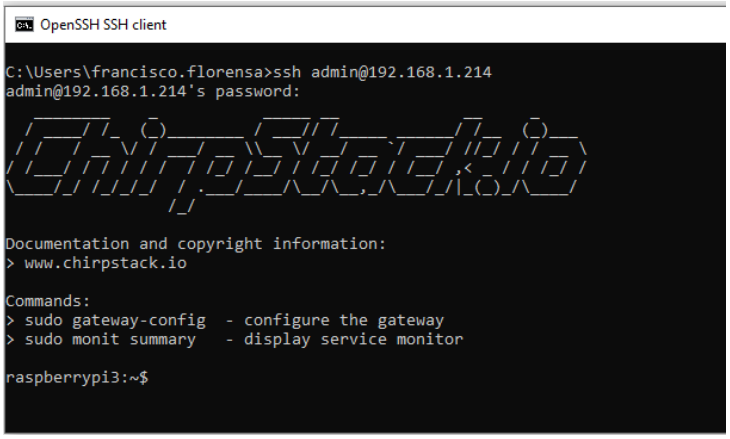




Decimal	Hex	Char
32	20	[SPACE]
33	21	!

Before continouing let's install node-red on the Gateway so it will be easier to decode the payload. And write to the destination PLC

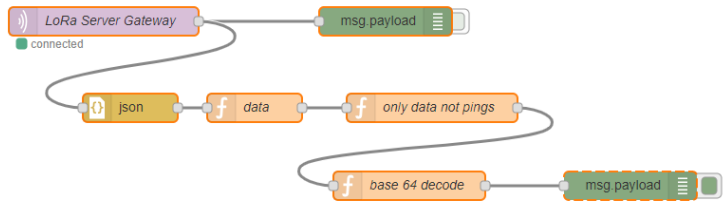
INSTALLING NODE-RED ON THE GATEWAY



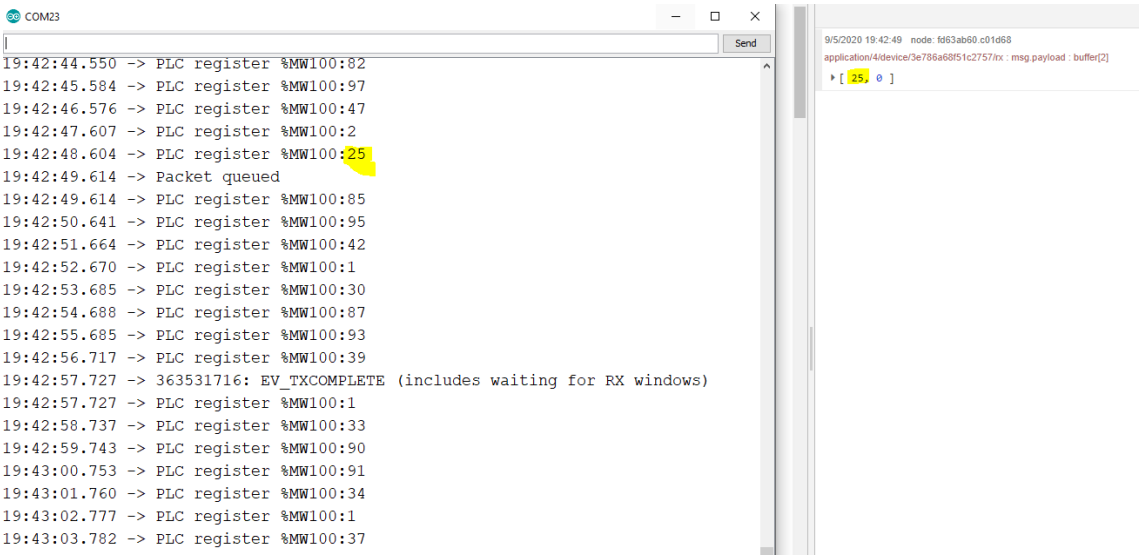
But the LoRa server SD image is build on YOCTO, so a knowledge of Yocto is needed to install node-red.

Let's install another Raspberry Pi with Node-red on the same Gateway Box, or on any location of the same network.

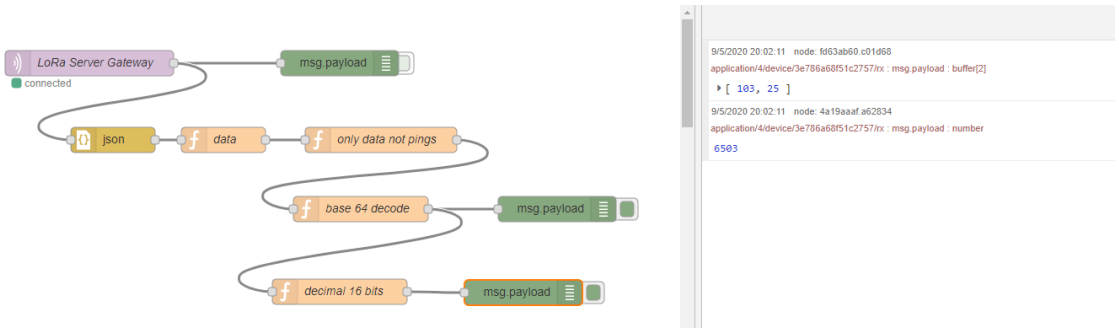
Now we have a nice decoder



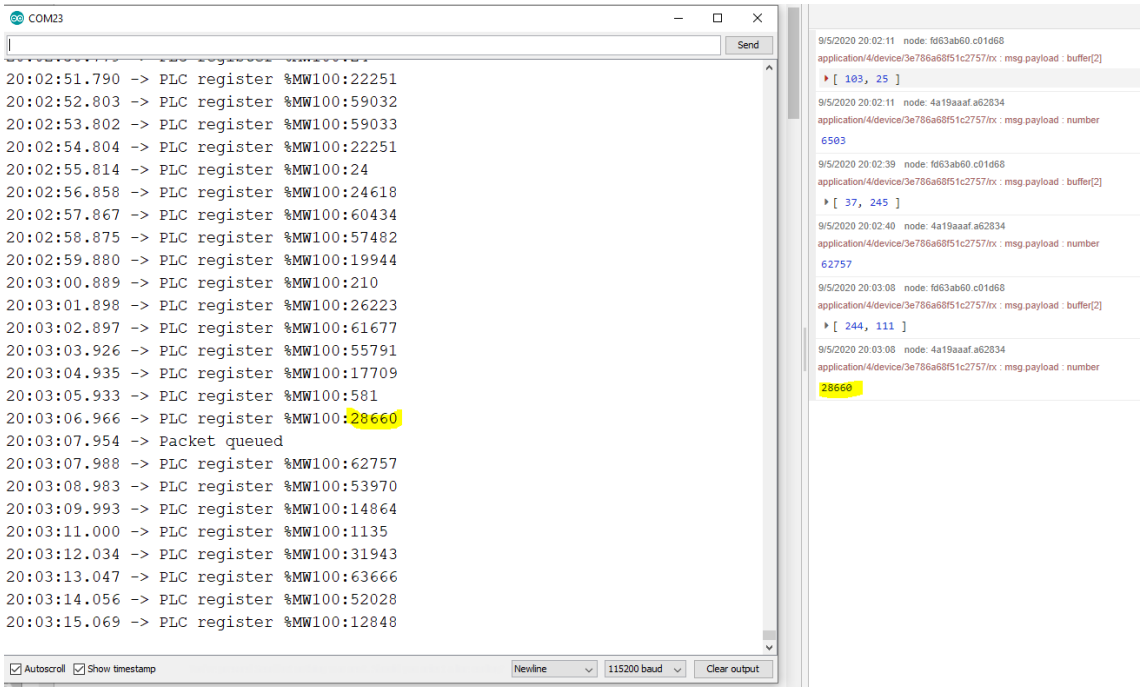
Yes

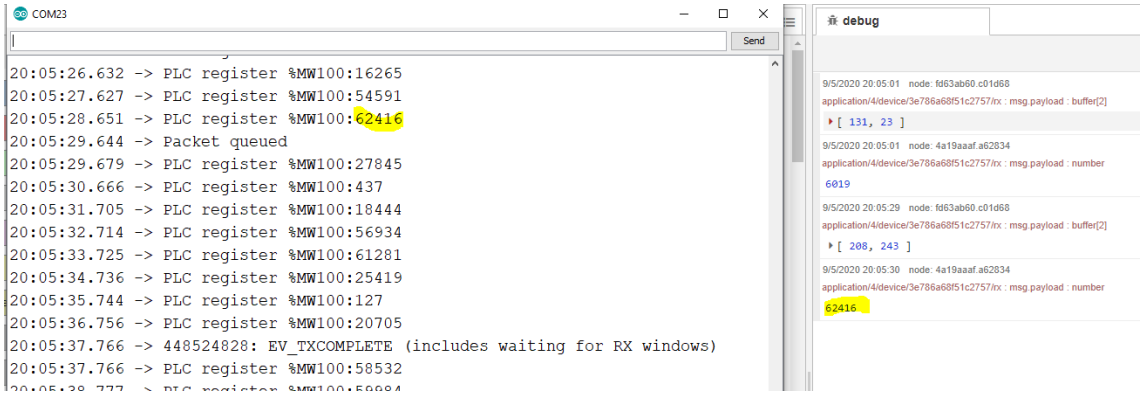


Now let’s modify the PLC program to generate numbers between 1 and 65535 maximum 16 bits int

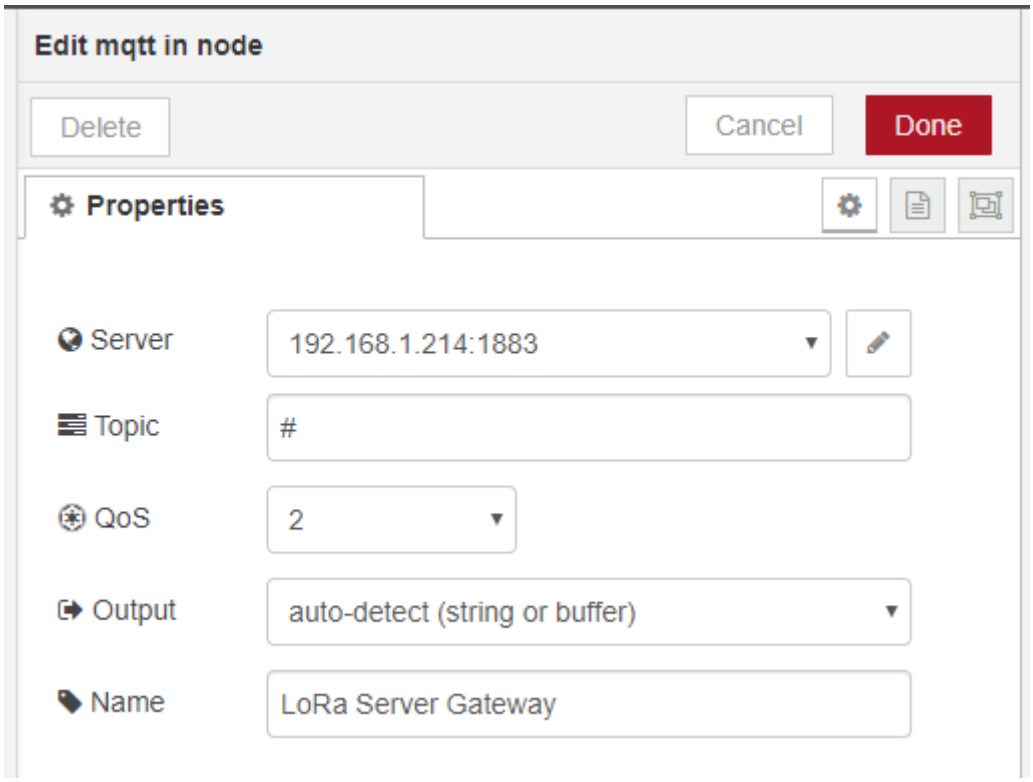


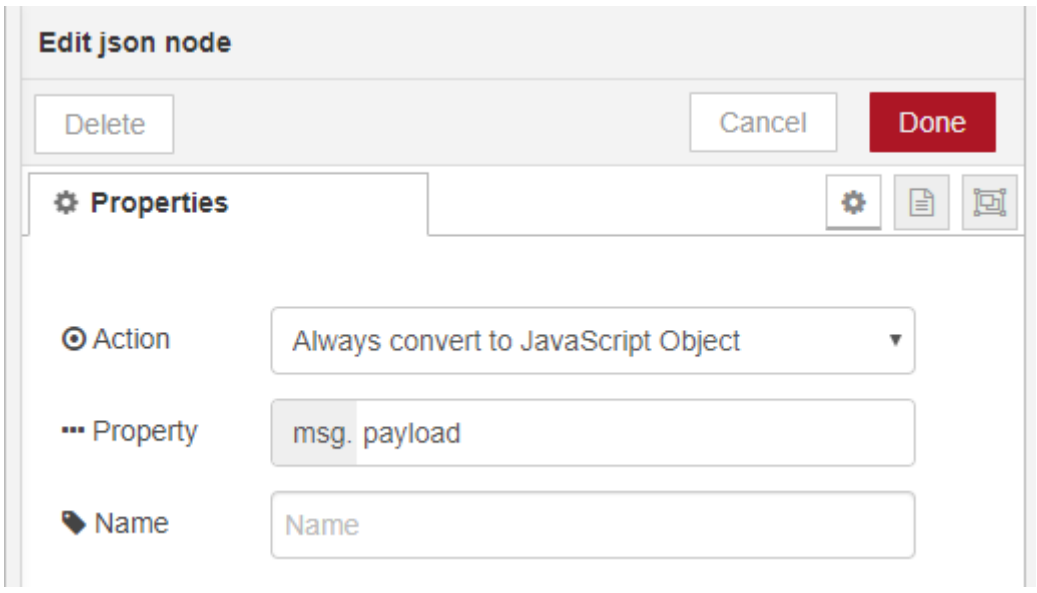
Yes

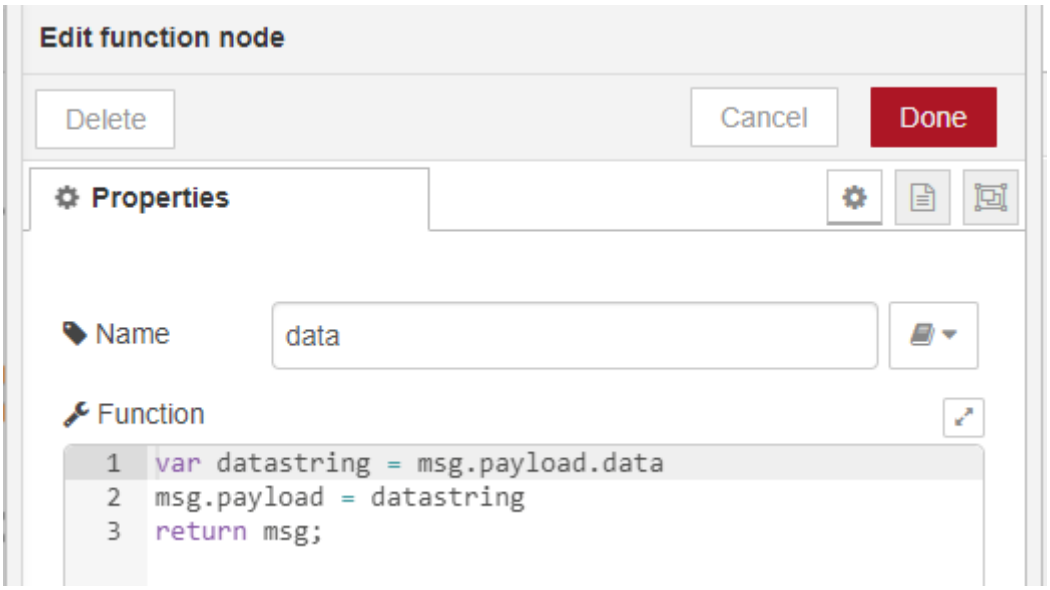


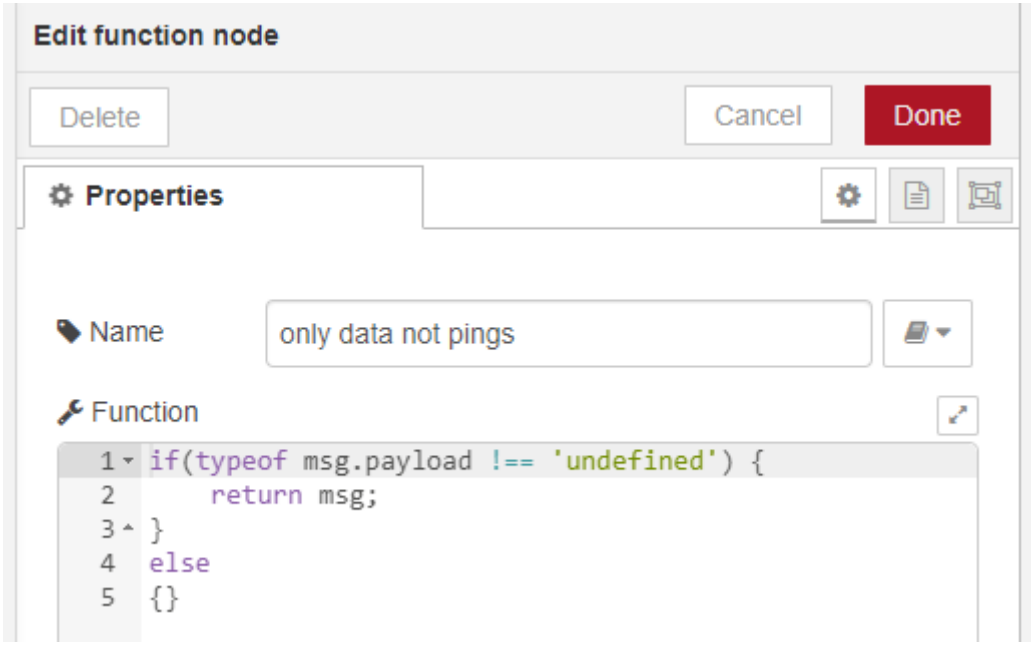


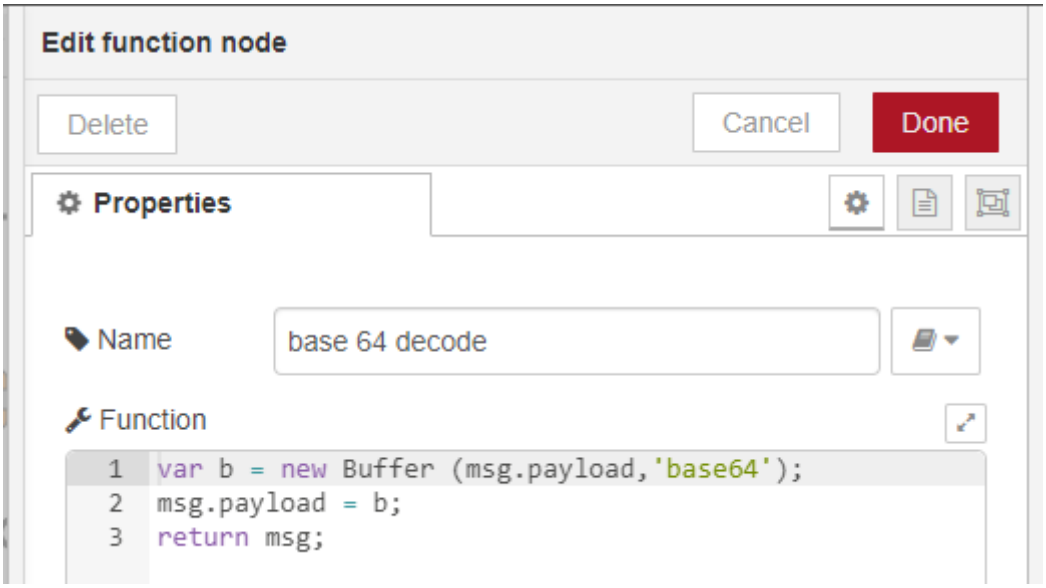
And this is the node-red Flow

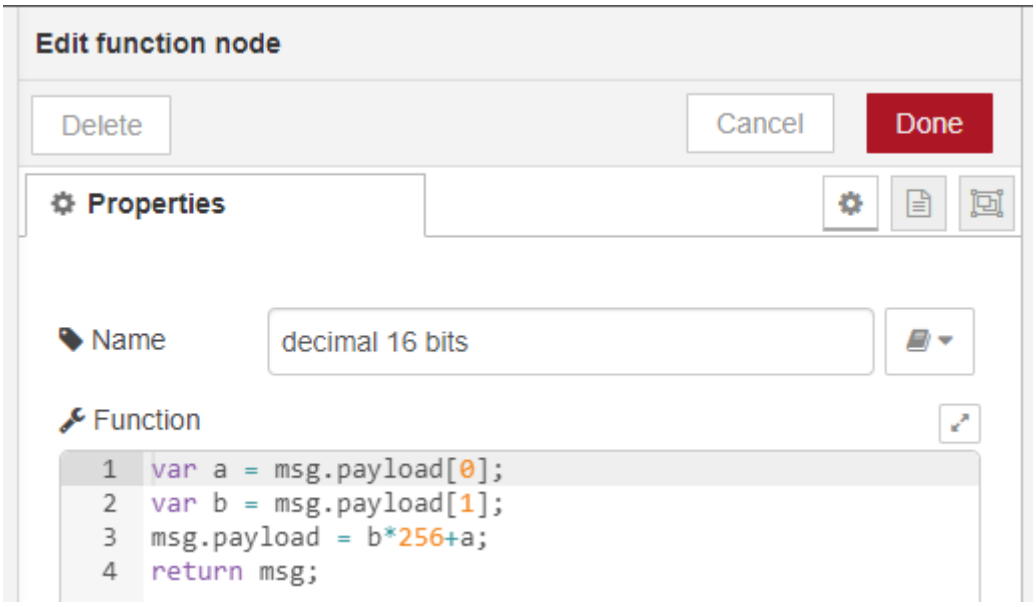

Edit mqtt in node
Delete
Cancel
Done
Properties
Server
192.168.1.214:1883
Topic
#
QoS
2
Output
auto-detect (string or buffer)
Name
LoRa Server Gateway


Edit json node
Delete
Cancel
Done
Properties
Action
Always convert to JavaScript Object
Property
msg. payload
Name
Name

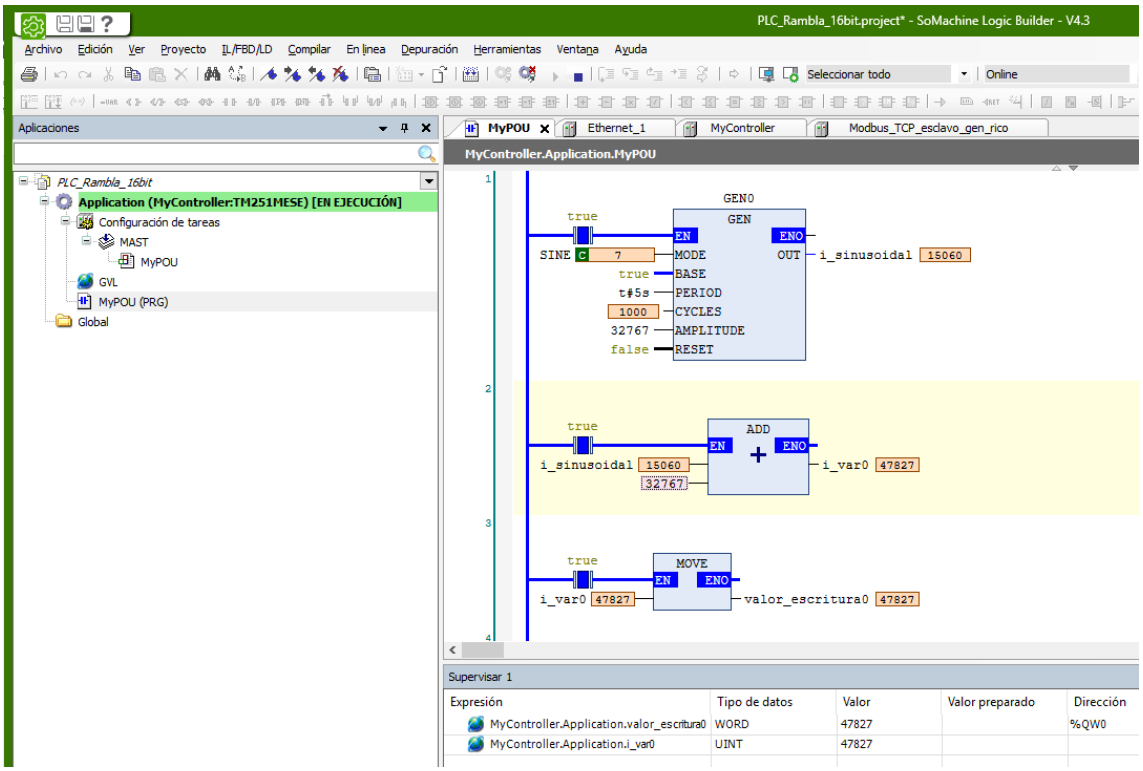

Edit function node
Delete
Cancel
Done
Properties
Name
data
Function
1 var datastring = msg.payload.data
2 msg.payload = datastring
3 return msg;

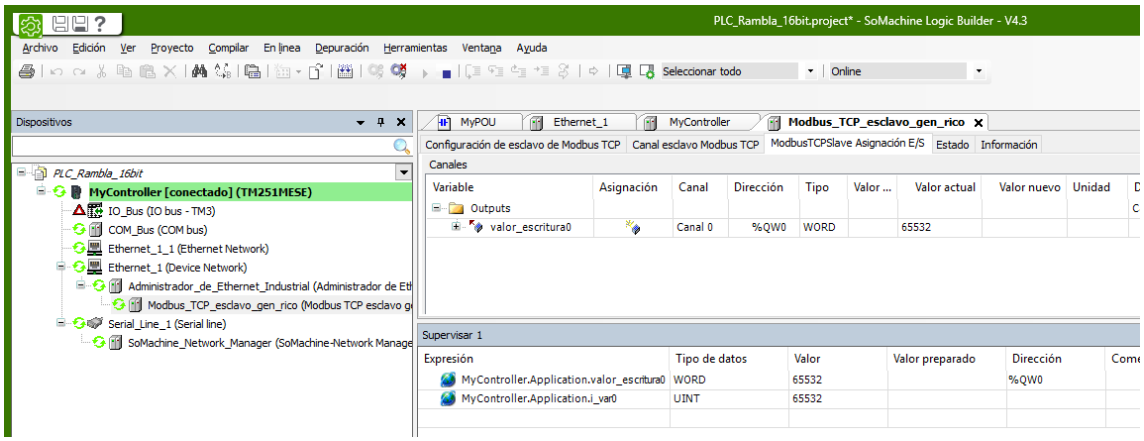

Edit function node
Delete
Cancel
Done
Properties
Name
only data not pings
Function
1 if(typeof msg.payload !== 'undefined') {
2 return msg;
3 }
4 else
5 {}

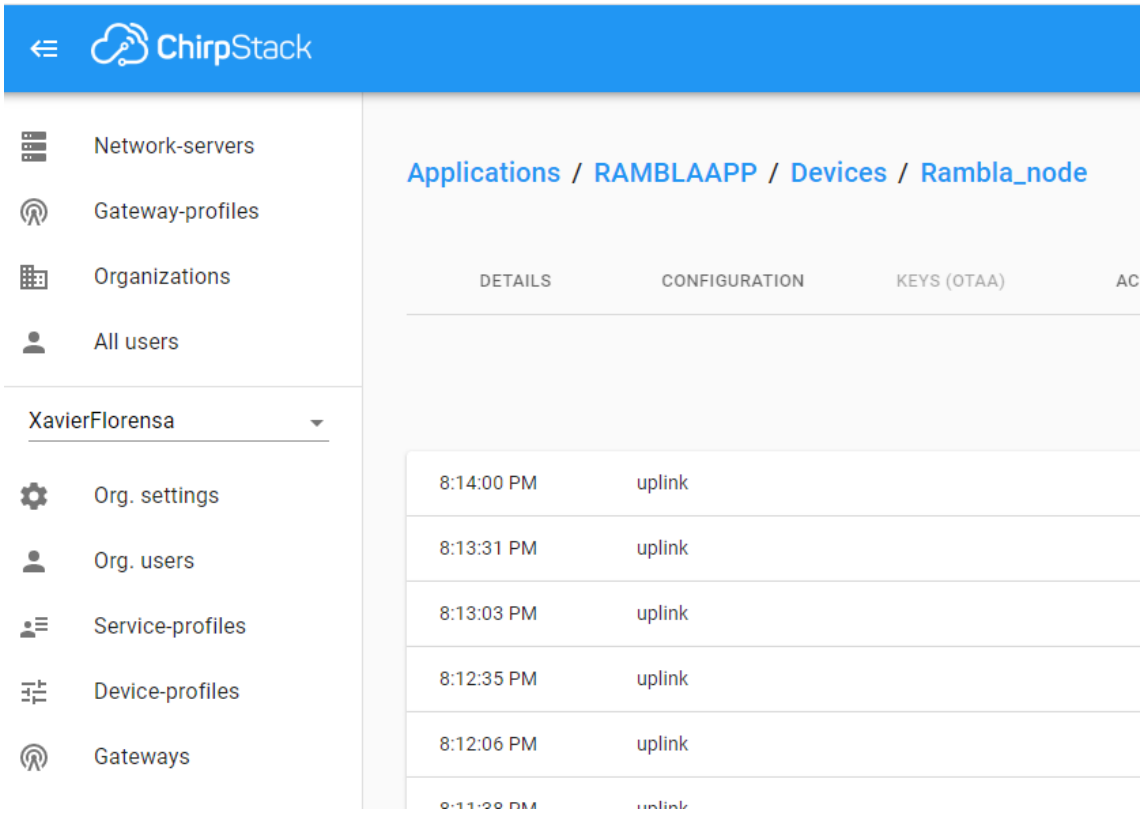

Edit function node
Delete
Cancel
Done
Properties
Name
base 64 decode
Function
1 var b = new Buffer (msg.payload,'base64');
2 msg.payload = b;
3 return msg;



This is the PLC programm

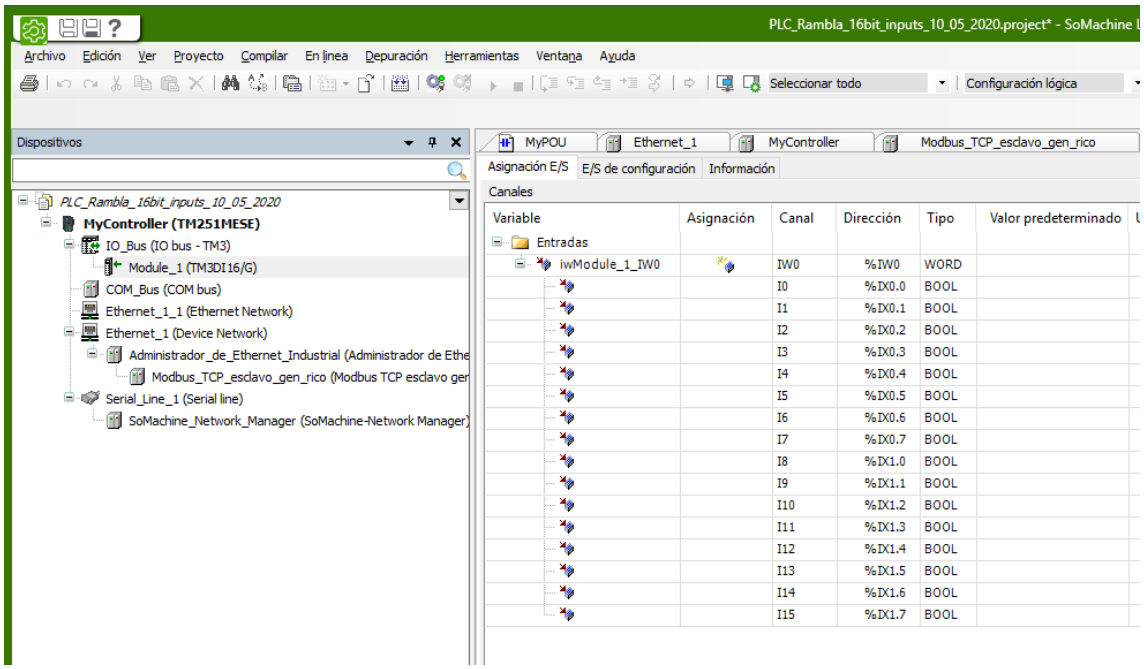


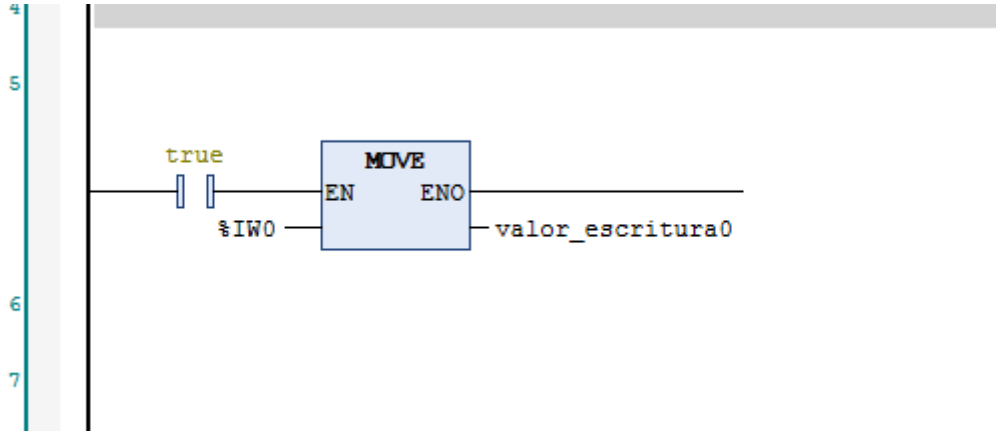




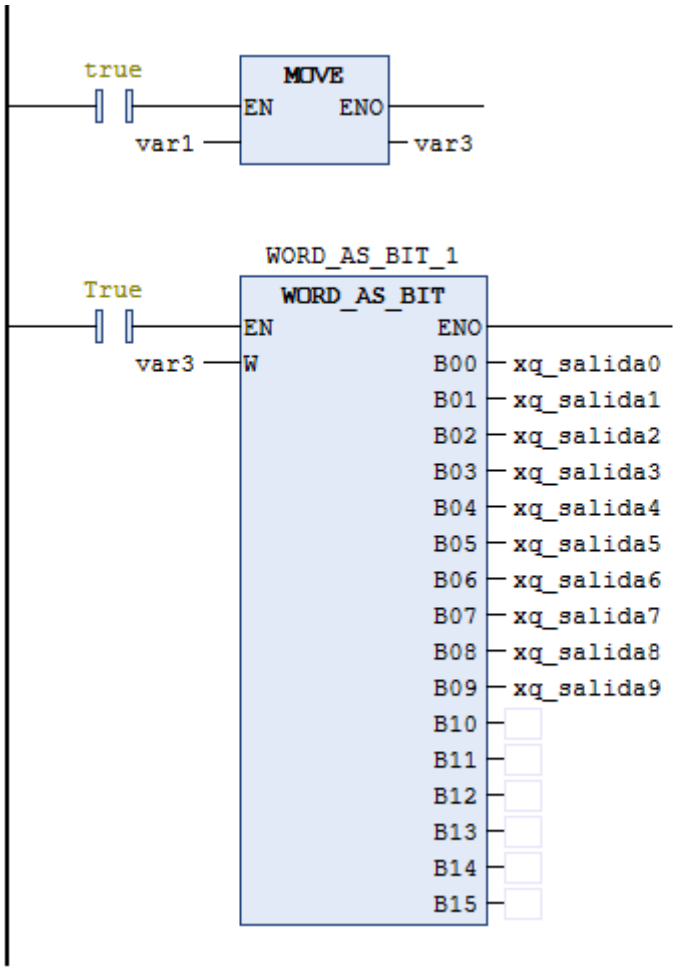
Next step is to use the digital inputs and translate to the digital outputs of a master PLC

LoRa PLC programm





This is the PLC progamm on the Master side



Here you can find the test video on 10/05/2020

https://youtu.be/gVRArAS1oRc

Here you can find the Arduino sketch

https://github.com/xavierflorensa/Modbus-TCP-to-LoRa-converter/tree/master/Modbus_TCP_LMIC_16bit_integer_no_keys

Here you can find the Node PLC programm

https://github.com/xavierflorensa/Modbus-TCP-to-LoRa-converter/blob/master/PLC_Rambla_16bit.project

Here you can find the Master (Gateway) PLC programm

https://github.com/xavierflorensa/Modbus-TCP-to-LoRa-converter/blob/master/Edificio%20central%20Gateway%20PLC%20Maestro%20v0.project

Here you can find the Node-red Flow

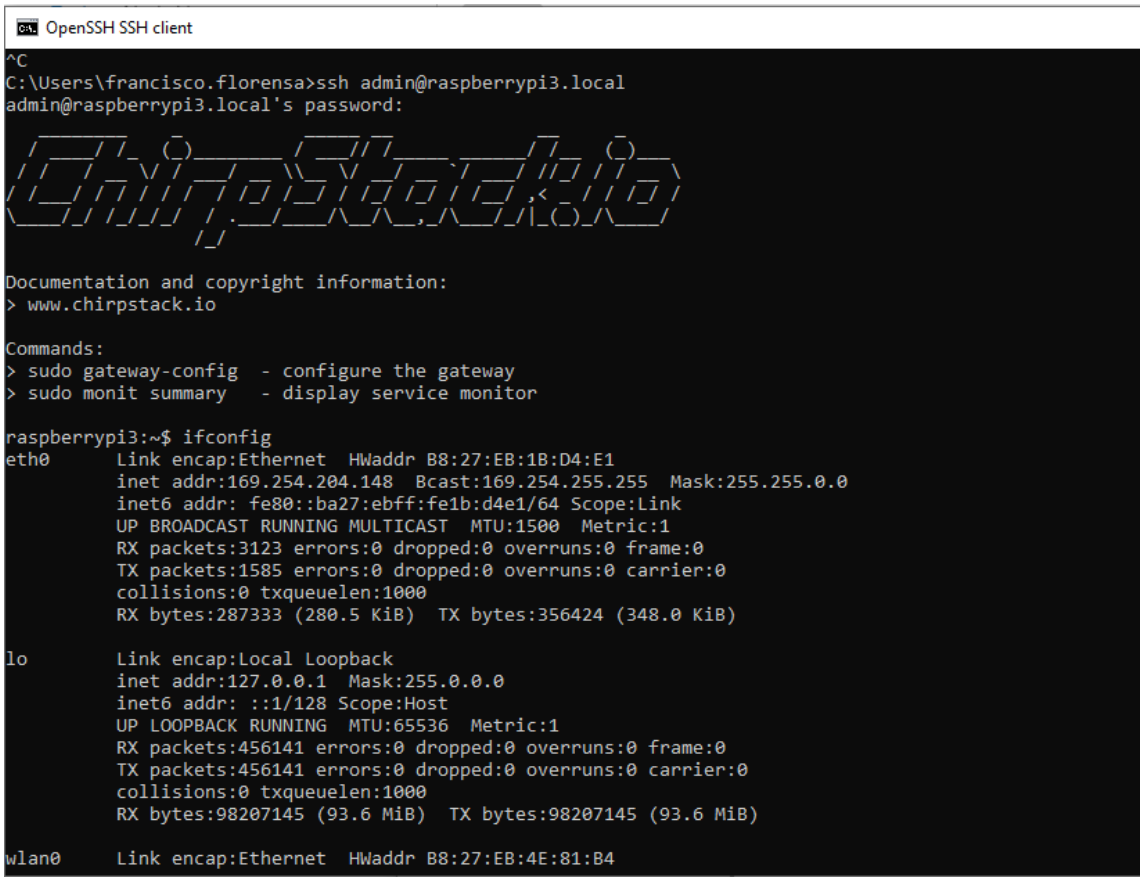
https://github.com/xavierflorensa/Modbus-TCP-to-LoRa-converter/blob/master/PLC_data_decoding_node_red_flow.txt

TEST WITHOUT ROUTER

The customer soes not have any router on the PLC’s network.

Next step is to assign a fixed IP to be in the same range of the PLC’s network.

Connect to the Gateway from the PC with a patchcord



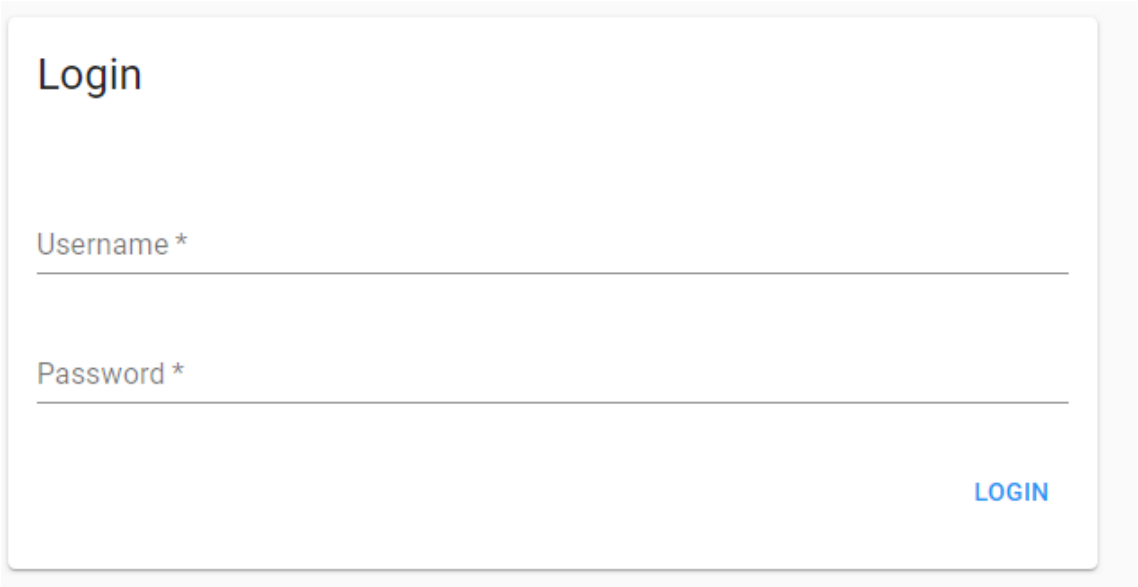
These are the IP’s from the PLC’s

TM251MESE	192.168.1.201	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:B7:C6
TM251MESE	192.168.1.202	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:B7:DE
TM251MESE	192.168.1.203	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:B7:E0
TM251MESE	192.168.1.204	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:B7:D2
TM251MESE	192.168.1.205	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:47:8E
TM251MESE	192.168.1.206	TELEMECANIQUE ELECTRIQUE	00:80:F4:3C:B7:DC

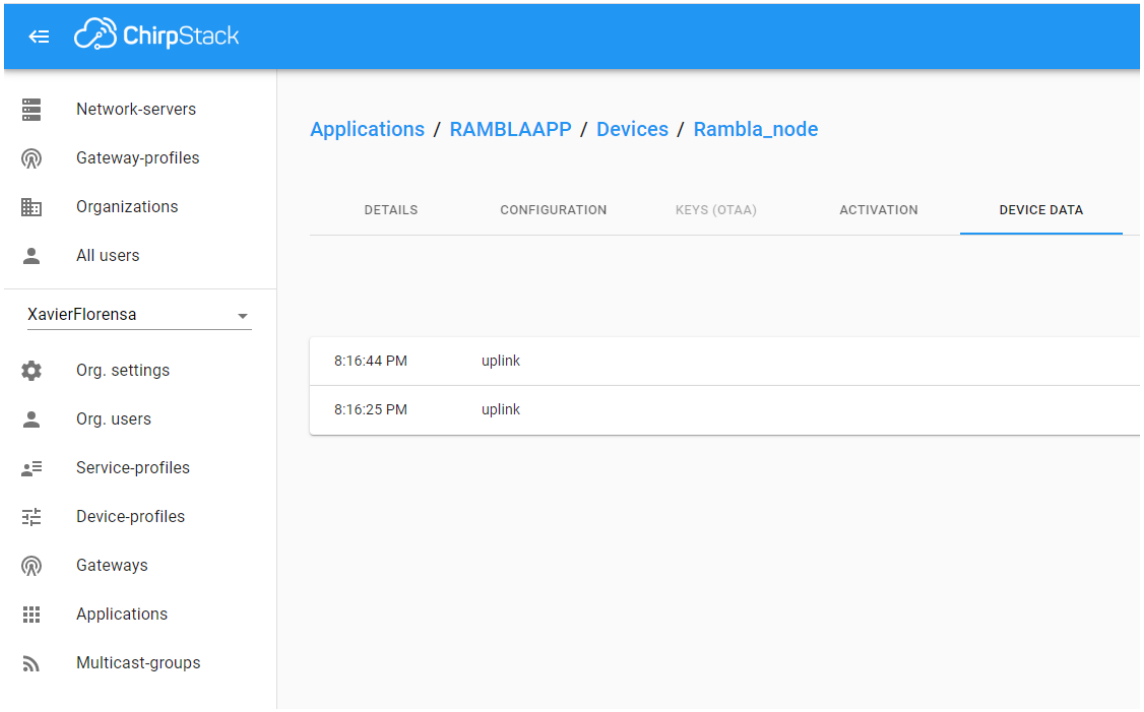
We can set up a fixed address on the Gateway, at least temporarily

```
raspberrypi3:~$ sudo ifconfig -a eth0 192.168.1.100
Password:
raspberrypi3:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr B8:27:EB:1B:D4:E1
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ba27:ebff:fe1b:d4e1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17557 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4643 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1384397 (1.3 MiB)  TX bytes:1071262 (1.0 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1142274 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1142274 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:246552344 (235.1 MiB)  TX bytes:246552344 (235.1 MiB)

wlan0     Link encap:Ethernet  HWaddr B8:27:EB:4E:81:B4
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

raspberrypi3:~$
```

Now I can connect from the PC



And it is still working!



If you want to make the IP address permanente each time you start the Gateway

https://forum.chirpstack.io/t/static-ip-and-dns-on-raspberrypi-with-lora-gateway-os/4310/2

So create a file named network.sh

Thank you but the regular ways wont work as the Gateway OS Runs on an older Version of Linux.
But we found a way by adding a bash script to the System Startup:

```
#!/bin/sh
DESC=“FIX NETWORK”

case “$1” in
start)
echo “Starting $DESC”
ifconfig eth0 “ip” netmask “mask” up
;;
stop)
echo “Stopping $DESC”
do_stop
;;
restart|force-reload)
echo “Restarting $DESC”
do_stop
sleep 1
do_start
;;
*)
echo “Usage: $0 {start|stop|restart|force-reload}” >&2
exit 1
```

;;
esac

exit 0

CLI:
raspberrypi3:/etc/init.d# ./network start
Starting FIX NETWORK
raspberrypi3:/etc/init.d# sysctl
raspberrypi3:/etc/init.d# chmod 755 network
raspberrypi3:/etc/init.d# update-rc.d network defaults
Adding system startup for /etc/init.d/network.
raspberrypi3:/etc/init.d# ll /etc/init.d/network

```
#!/bin/sh
DESC="FIX NETWORK HSMA"

case "$1" in
  start)
    echo "Starting $DESC"
    ifconfig eth0 141.19        netmask 255.255.255.0 up
    ;;
  stop)
    echo "Stopping $DESC"
    do_stop
    ;;
  restart|force-reload)
    echo "Restarting $DESC"
    do_stop
    sleep 1
    do_start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|force-reload}" >&2
    exit 1
    ;;
esac

exit 0
```

Let’s assign a fixed IP address to our Raspberry Pi (who has Node-red)

Let’s assign 192.168.1.101

With ifconfig -a eth0 192.168.1.101

To make this change permanente

Re: How to Configure Pi with Static IP Address?

Sat Aug 25, 2018 10:47 am

Simply:

Code: Select all

```
sudo nano /etc/dhcpcd.conf
```

Code: Select all

```
#Config for static IP on eth0

interface eth0
static ip_address=192.168.1.121/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

In this case the Pi will appear as 192.168.1.121

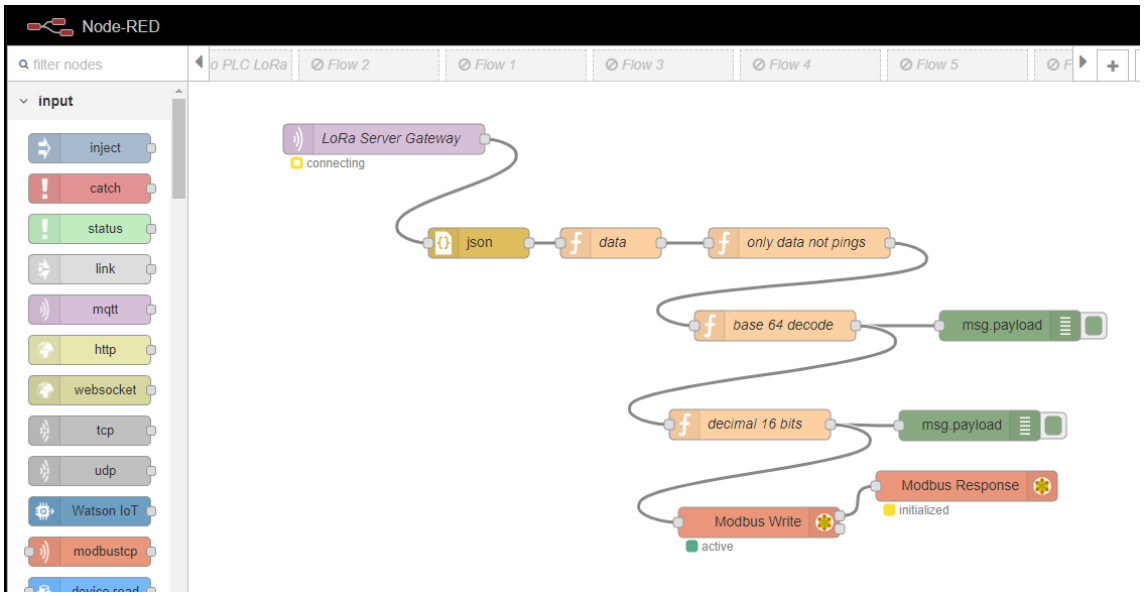
https://www.raspberrypi.org/forums/viewtopic.php?t=221060#p1357512

And stop node red

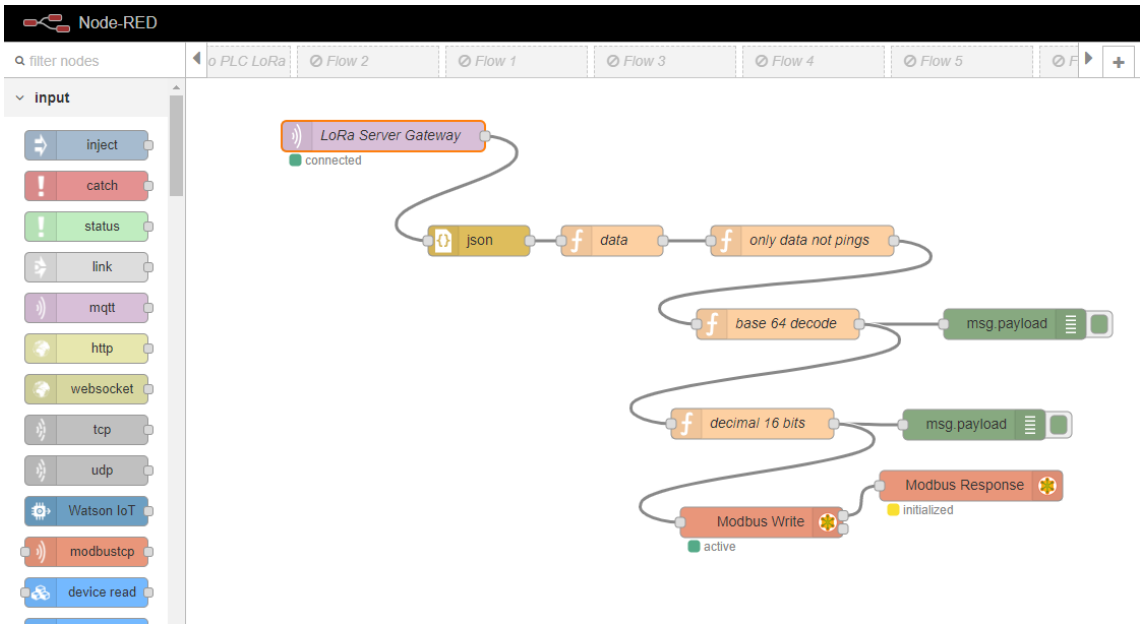
systemctl stop nodered

Start node red

Systemctl start nodered



Now we have to change the address of mqtt it is address 100



Voila connected

And running

¡!!

But let’s make it all in one Raspberry

INSTALLATION OF CHIRPSTACK OVER RASPBIAN

sudo apt update

sudo apt upgrade

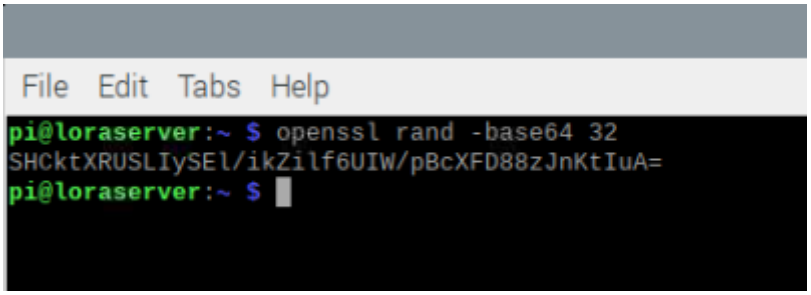
https://www.chirpstack.io/guides/debian-ubuntu/

Also explained here

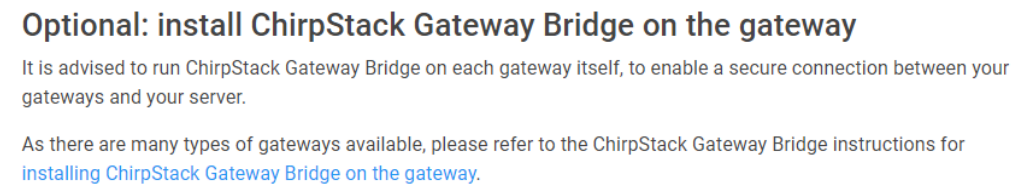
https://www.youtube.com/watch?v=FnTP7t47DlI

jwt

SHCktXRUSLIySEl/ikZilf6UIW/pBcXFD88zJnKtIuA=



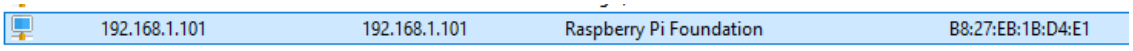
Missing (optional)



Create Gateway and device

MAC address of Gateway

Advanced IP scanner



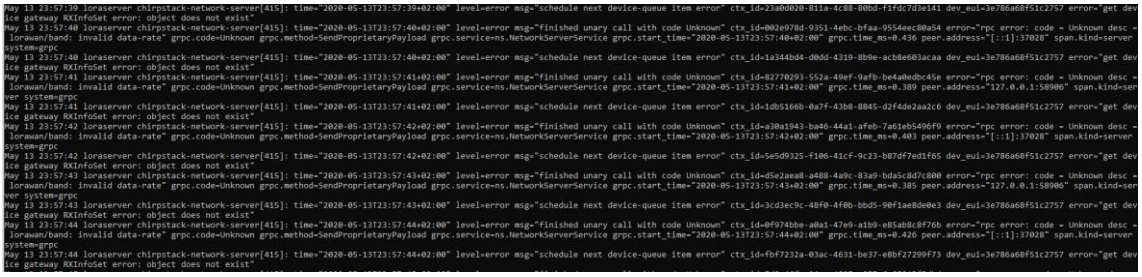
https://www.youtube.com/watch?v=mkuS5QUj5Js

It is not working

We get following errors

Print the ChirpStack Network Server log-output:

```
sudo journalctl -f -n 100 -u chirpstack-network-server
```



Print the ChirpStack Application Server log-output:

```
sudo journalctl -f -n 100 -u chirpstack-application-server
```