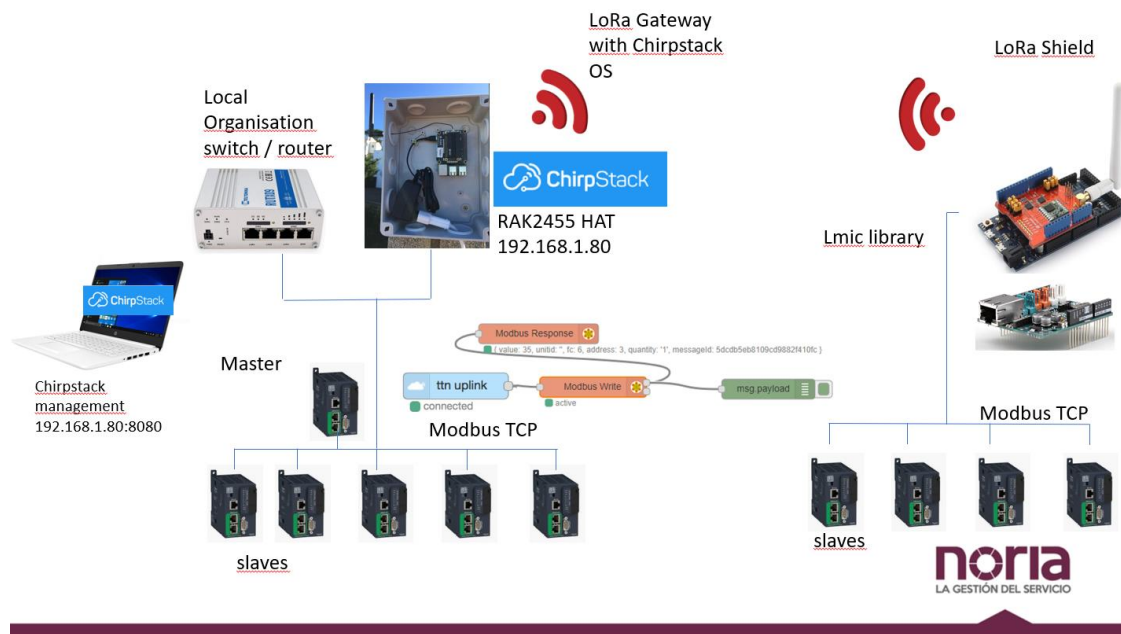


PLC alarms communication thru LoRa Server Gateway

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 perform the LoRa server

TESTING LMIC LIBRARY with LoRa Server

We adapt LMIC example to our needs

From

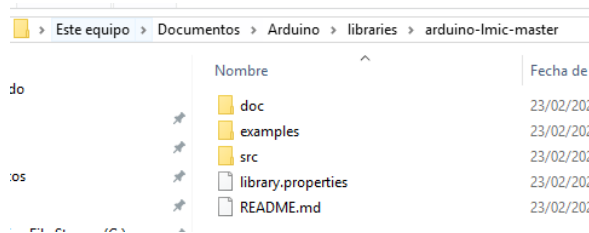
Download the zip file from

<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>

Device-profiles / Create

GENERAL	JOIN (OTAA / ABP)	CLASS-B	CLASS-C	CODEC
Device-profile name *				
RAMBLA				
A name to identify the device-profile.				
Network-server *				
localhost				
The network-server on which this device-profile will be provisioned. After creating the device-profile, this value can't be changed.				
LoRaWAN MAC version *				
1.0.2				
The LoRaWAN MAC version supported by the device.				
LoRaWAN Regional Parameters revision *				
B				
A				
B				
Geolocation buffer TTL (seconds)				
0				
The time in seconds that historical uplinks will be stored in the geolocation buffer.				
Geolocation minimum buffer size				
0				
The minimum buffer size required before using geolocation (when enabled in the Service Profile). Using multiple uplinks for geolocation can increase the acc				

Once created

Device-profiles / RAMBLA

GENERAL	JOIN (OTAA / ABP)	CLASS-B	CLASS-C	CODEC
Device-profile name *				
RAMBLA				
A name to identify the device-profile.				
LoRaWAN MAC version *				
1.0.2				
The LoRaWAN MAC version supported by the device.				
LoRaWAN Regional Parameters revision *				
B				
Revision of the Regional Parameters specification supported by the device.				
Max EIRP *				
0				
Maximum EIRP supported by the device.				
Geolocation buffer TTL (seconds)				
0				
The time in seconds that historical uplinks will be stored in the geolocation buffer.				
Geolocation minimum buffer size				
0				
The minimum buffer size required before using geolocation (when enabled in the Service Profile). Using multiple uplinks for geolocation can increase				

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|

Device-profiles / RAMBLA

GENERAL

JOIN (OTAA / ABP)

CLASS-B

CLASS-C

CODEC

☒ Device supports Class-C

Select this option when the device will operate as Class-C device immediately after activation. In case it sends a DeviceModelInd mac-cs

Class-C confirmed downlink timeout *

60

Class-C timeout (in seconds) for confirmed downlink transmissions.

Now we create an Application

Applications / Create

Application name *

RAMBLAAPP

The name may only contain words, numbers and dashes.

Application description *

PLC Node Ramla

Service-profile *

IOTSERVER

IOTSERVER

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.

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

Device address *

26 01 13 3e

Network session key (LoRaWAN 1.0) *

.....

Application session key (LoRaWAN 1.0) *

.....

Uplink frame-counter *

16

Downlink frame-counter (network) *

9

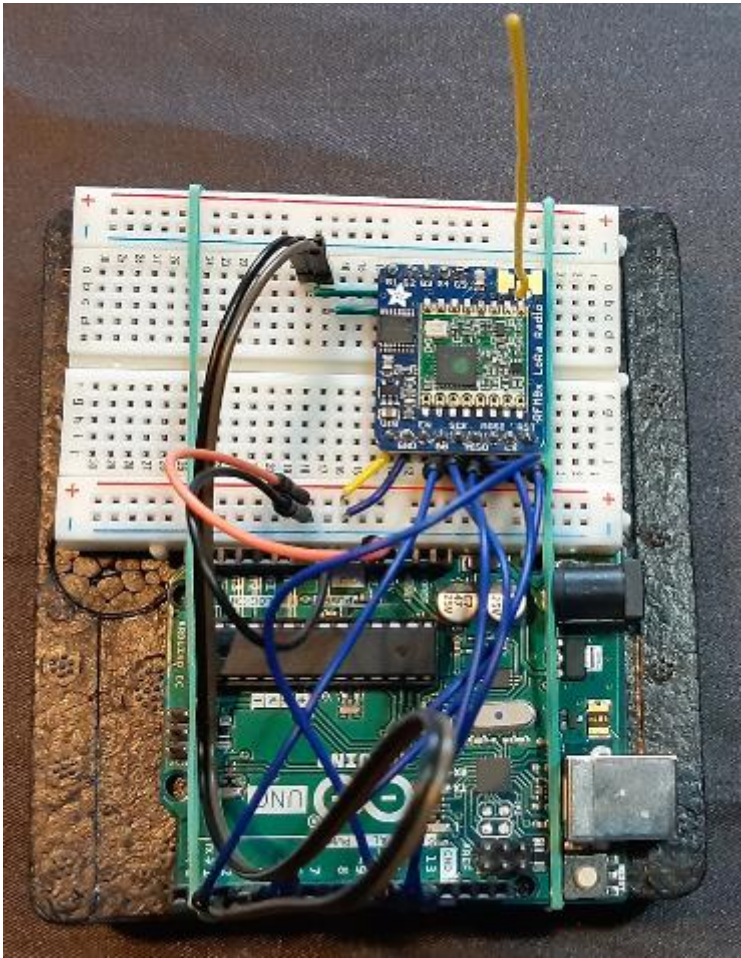
Applications / RAMBLAAPP

DEVICESAPPLICATION CONFIGURATIONINTEGRATIONSFUOTA

Last seen	Device name	Device EUI	Device profile	Link margin	Battery
n/a	Rambla_node	3e786a68f51c2757	RAMBLA	n/a	n/a

Rows per page: 101-1 of 1

TESTING WITH ARDUINO UNO and RFM95 LoRa chip



Power Up Arduino and... it Works

```
COM23
|
|
|
Starting
12:30:26.925 -> Packet queued
12:30:28.097 -> 72055: EV_TXCOMPLETE (includes waiting for RX windows)
12:30:28.097 -> Received
12:30:28.097 -> 30
12:30:28.097 -> bytes of payload
|
|
|
Autoscroll Show timestamp Newline 115200 baud Clear output
```

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

```
COM14
|
|
|
19:49:57.786 -> St
19:49:57.786 -> Packet queued
19:49:57.786 -> ??j??@????ie[???[]?j?   []?q[]D? ???[]???
19:56:12.933 -> []???~[]D????0?q[]D????0[]?[]UA<?R>??D???[]1[]????2?2?
20:02:21.354 -> ??ixx
|
|
|
```

Taking a look at the application console



Network-servers



Gateway-profiles



Organizations



All users

XavierFlorensa ▼



Org. settings



Org. users



Service-profiles



Device-profiles



Gateways



Applications



Multicast-groups

applicationID: "4"

applicationName: "RAMBLAAPP"

deviceName: "Rambla_node"

devEUI: "3e786a68f51c2757"

▼ rxInfo: [] 1 item

▼ 0: {} 14 keys

gatewayID: "b827ebffe1bd4e1"

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: "SGVsbG8slHdvcmxkIQ=="

objectJSON: ""

tags: {} 0 keys

BASE64

Decode and Encode

Decode


Encode

Have to deal with **Base64** format? Then this site is made for you.

Decode from Base64 format

Simply enter your data then push the decode button.

SGVsbG8slHdvcmxklQ==


 For encoded binaries (like images, documents, etc.) use the **Decode as binary** checkbox.

UTF-8

▼

Source character set.

☐ Decode each line separately (useful for multiple entries).

 Live mode OFF

Decodes in real-time when you type.

< DECODE >

Decodes your data into the text area below.

Hello, world!

And the Gateway

ChirpStack

Network-servers

Gateway-profiles

Organizations

All users

XavierFlorensa

Org. settings

Org. users

Service-profiles

Device-profiles

Gateways

Gateways / RAKGATEWAY

GATEWAY DETAILS

GATEWAY CONFIGURATION

GATEWAY DISCOVERY

LIVE LORAWAN FRAMES

DOWNLINK	8:06:25 PM	UnconfirmedDataDown	2601133e
UPLINK	8:06:25 PM	UnconfirmedDataUp	2601133e
DOWNLINK	8:05:24 PM	UnconfirmedDataDown	2601133e
UPLINK	8:05:24 PM	UnconfirmedDataUp	2601133e
DOWNLINK	8:04:23 PM	UnconfirmedDataDown	2601133e

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

48=h

65=e

Llo world

Applications >  prova_rambla_mega_rfm95 > Devices >  rambla_mega_rfm95 > Data

APPLICATION DATA

Filters

uplink

downlink

activation

ack

error

time

counter

port

▲ 20:16:36 26 1 payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

Uplink

Payload

48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21



Fields

no fields

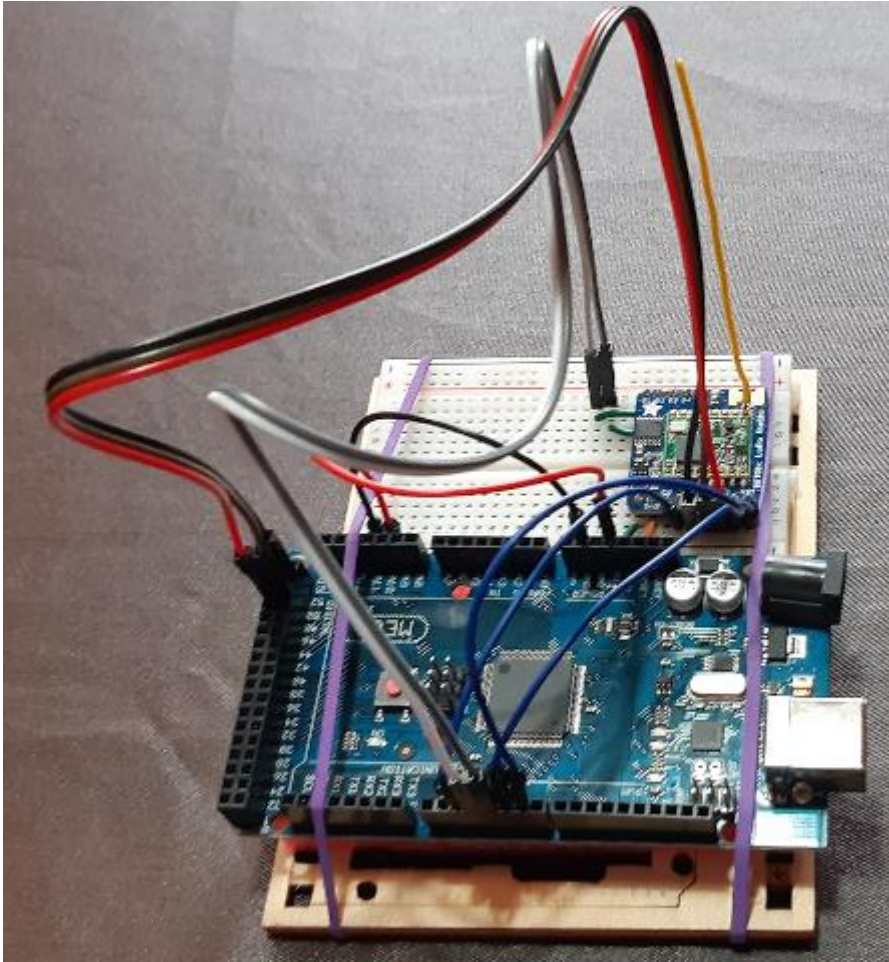
Metadata

```
{
  "time": "2020-05-08T18:16:36.867094167Z",
  "frequency": 868.5,
  "modulation": "LORA",
  "data_rate": "SF7BW125",
  "coding_rate": "4/5",
  "gateways": [
    {
      "gtw_id": "eui-58a0cbfffe80175a",
      "timestamp": 3369997011,
      "time": "2020-05-08T18:16:36.748344898Z",
      "channel": 0,
      "rssi": -45,
      "snr": 8
    }
  ]
}
```

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

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)

8:44:39 PM uplink

8:43:38 PM uplink

applicationID: "4"

applicationName: "RAMBLAAPP"

deviceName: "Rambla_node"

devEUI: "3e786a68f51c2757"

▼ rxInfo: [] 1 item

▼ 0: {} 14 keys

gatewayID: "b827ebfffe1bd4e1"

time: null

timeSinceGPSEPOCH: null

rssI: -18

IoRaSNR: 7.5

channel: 2

rfChain: 1

board: 0

antenna: 0

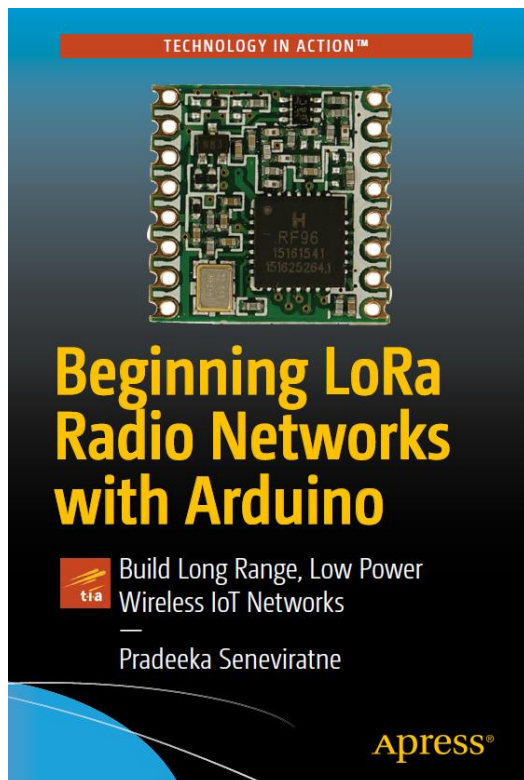
▼ location: {} 5 keys

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.

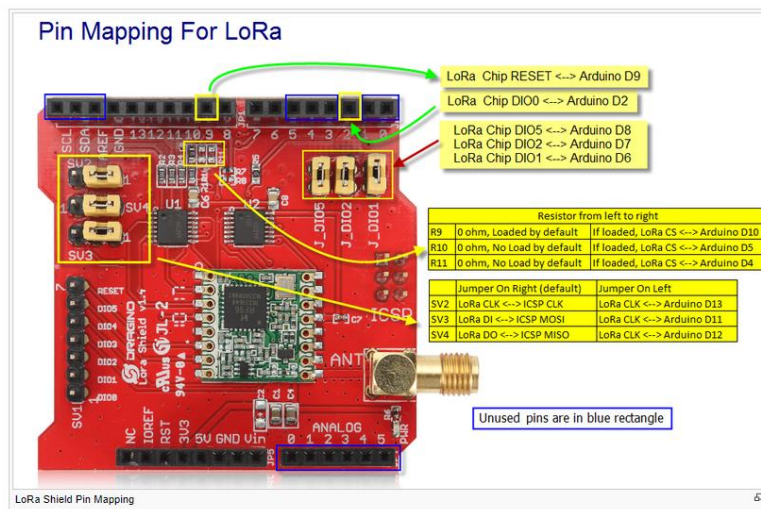
Arduino	RFM9x
2	G0
3	G1
4	G2
5	RST
6	CS
8	
11	MOSI
12	MISO
13	SCK
5V	
GND	

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	GO
4	CS
8	
11	MOSI
12	MISO
13	SCK

We have to pay 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

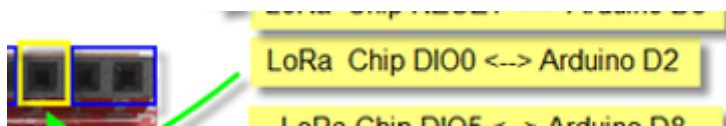
Table 5-2. Wiring Connections

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 DIO (G0)



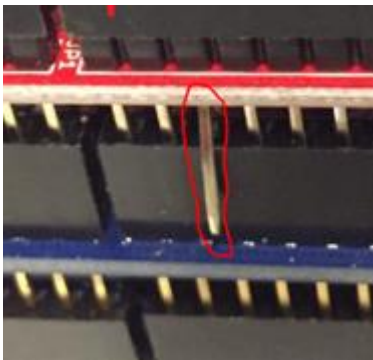
PIN 3



Pin 3 is unused by Draguino

Unused pins are in blue rectangle

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



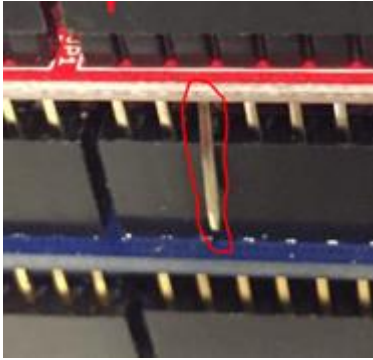
PIN 4

Pin 4 is unused by Draguno

Unused pins are in blue rectangle

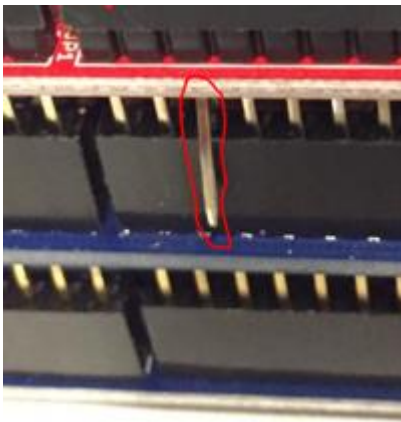
So we do the same we did with pin 3





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

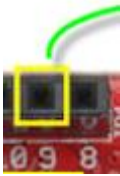


Connect to the RESET PIN of the Dragino



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





PIN 6

Pin 6 is busy



LoRa Chip DIO1 <--> Arduino D6

So we have to open this jumper

And connect pin 6 with LoRa CS

CS LoRa will be connected according to this jumper

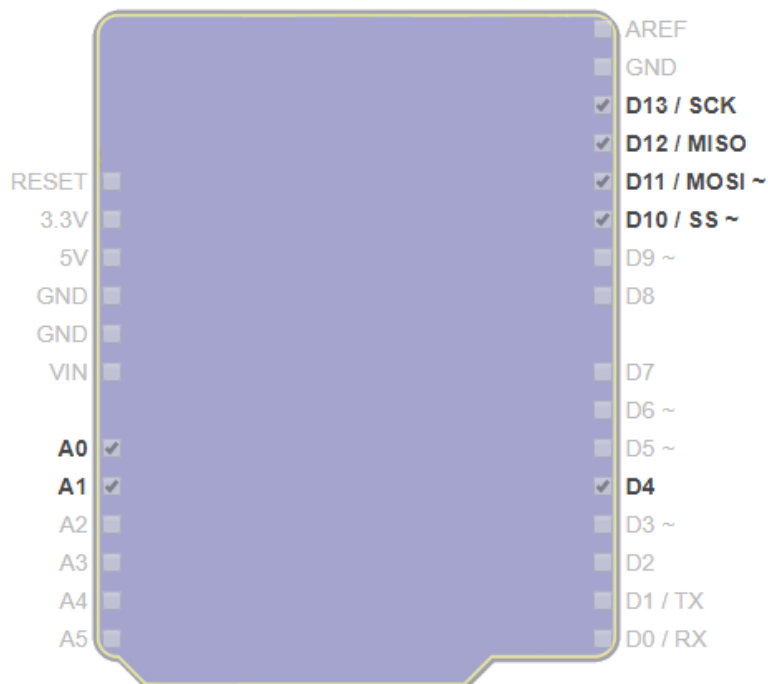


Resistor from left to right		
R9	0 ohm, Loaded by default	If loaded, LoRa CS <--> Arduino D10
R10	0 ohm, No Load by default	If loaded, LoRa CS <--> Arduino D5
R11	0 ohm, No Load by default	If loaded, LoRa CS <--> Arduino D4

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.

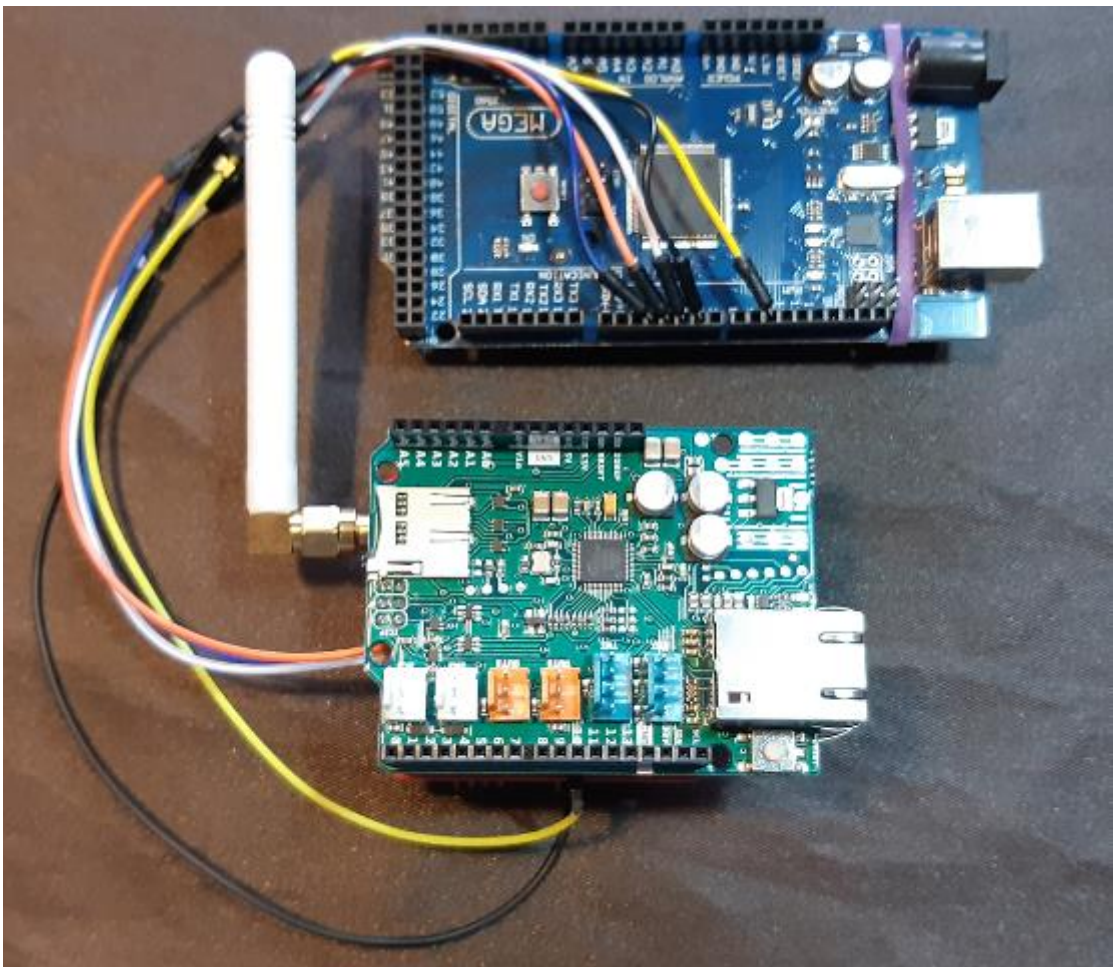
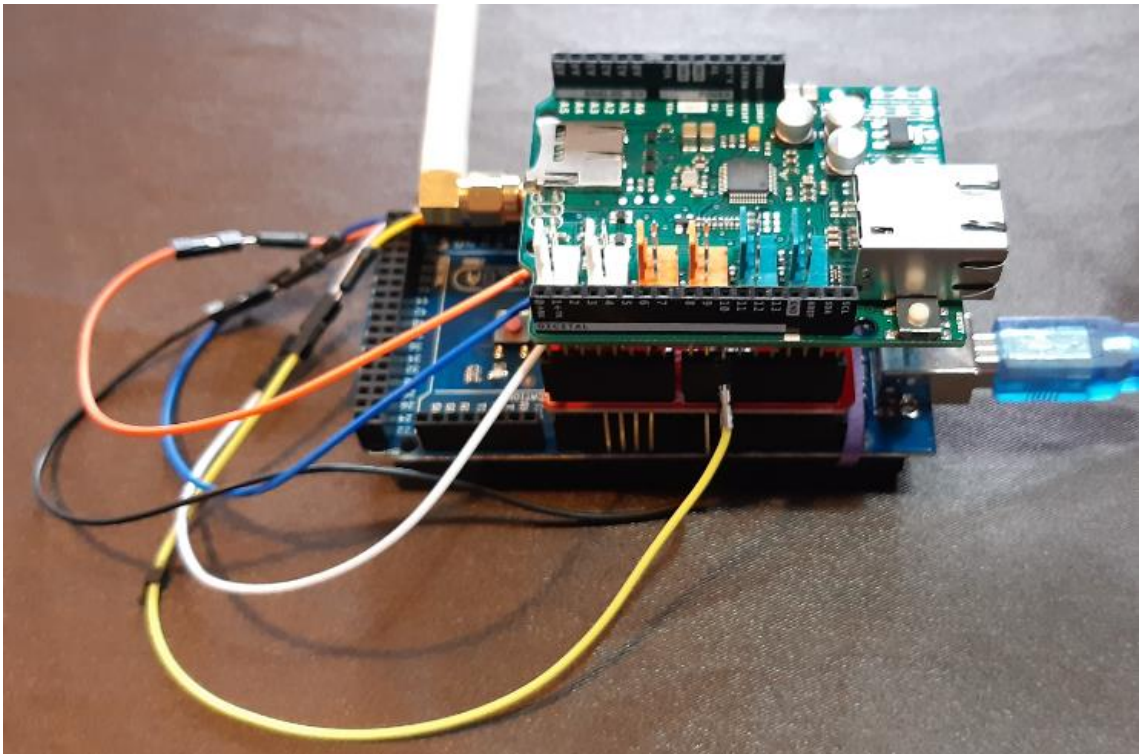
D10 is used to select the W5100 and cannot be used for general I/O.

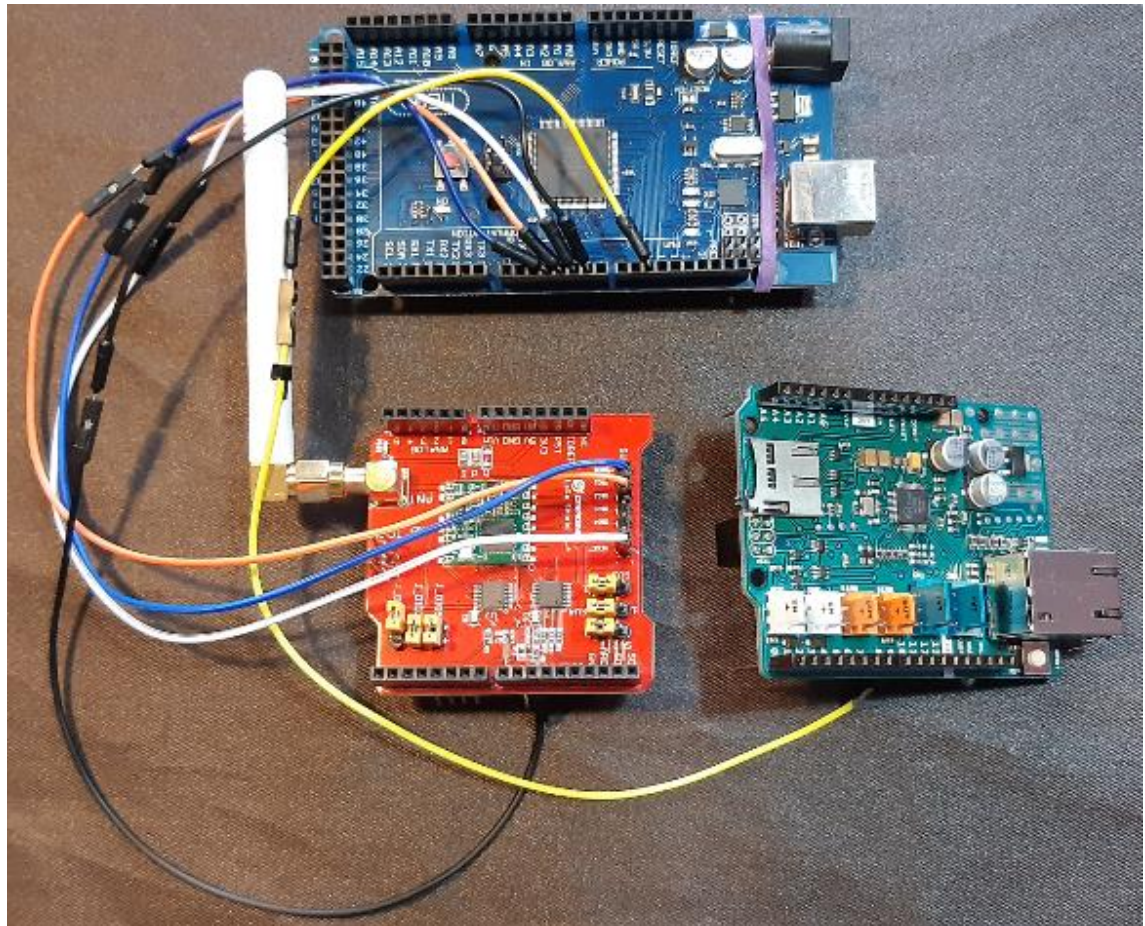
D4 is used for the SD card and can only be used for general I/O if the SD slot is not occupied.

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





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)

11:28:12 PM	uplink
11:27:11 PM	uplink
11:26:10 PM	uplink
11:25:09 PM	uplink
11:24:08 PM	uplink
11:23:06 PM	uplink
11:23:04 PM	uplink
11:22:51 PM	uplink

And it Works j!!!

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

The screenshot shows the ChirpStack web interface. On the left is a sidebar menu with options: Network-servers, Gateway-profiles, Organizations, All users, XavierFlorensa (selected), Org. settings, Org. users, Service-profiles, Device-profiles, Gateways, Applications, and Multicast-groups. The main content area shows the breadcrumb 'Applications / RAMBLAAPP / Devices / Rambla_node'. Below this are tabs for DETAILS, CONFIGURATION, and KEYS (OTAA). The DETAILS tab is active, showing a table of messages. The first message is at 9:05:12 AM, an uplink. The second message is at 9:04:10 AM, also an uplink, with its details expanded. The details show: applicationID: "4", applicationName: "RAMBLAAPP", deviceName: "Rambla_node", devEUI: "3e786a68f51c2757", rxInfo: [1 item], and a list of 14 keys. The first key is gatewayID: "b827ebfffe1bd4e1", time: null, timeSinceGPSEPOCH: null, and rxInfo: [1 item].

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 conversion 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
```

BASE64

Decode and Encode

Decode


Encode

Have to deal with **Base64** format? Then this site is made for you! Use

Decode from Base64 format

Simply enter your data then push the decode button.

ODU=


 For encoded binaries (like images, documents, etc.) use the file upload form

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 (suppo

< DECODE >

Decodes your data into the textarea below.

85

But the contents of PLCMW100 will be changing

So we will place the conversi3n sentence somewhere else

Here is a nice place

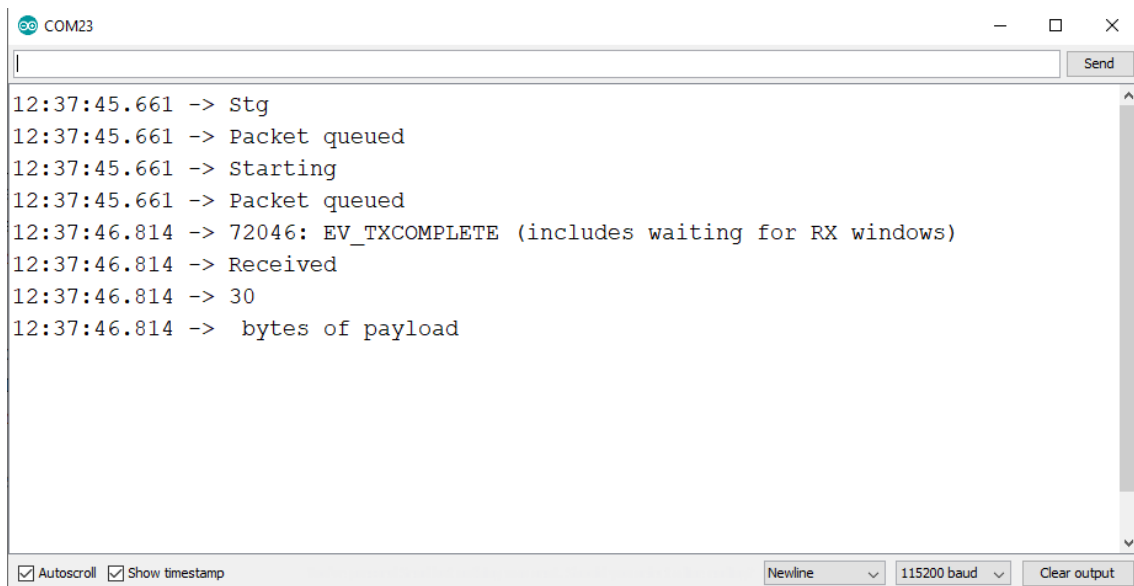
```

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, "%04d", 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



```

COM23
12:37:45.661 -> Stg
12:37:45.661 -> Packet queued
12:37:45.661 -> Starting
12:37:45.661 -> Packet queued
12:37:46.814 -> 72046: EV_TXCOMPLETE (includes waiting for RX windows)
12:37:46.814 -> Received
12:37:46.814 -> 30
12:37:46.814 -> bytes of payload

```

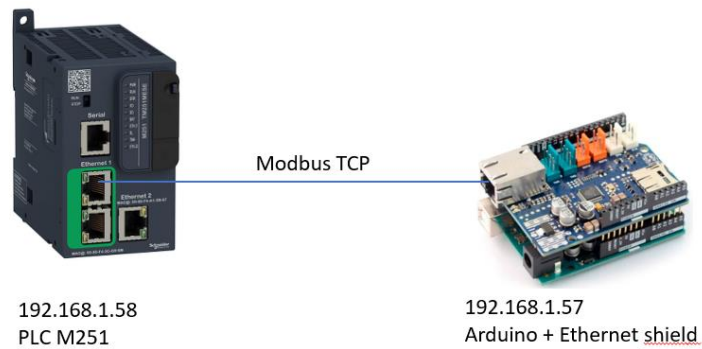
Now we forget about Lora and let's concentrate on the Modbus TCP and sketch to get data from PLC

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)



Asignamos direcciones IP

Maestro PLC Schneider

192.168.1.58

Esclavo Arduino

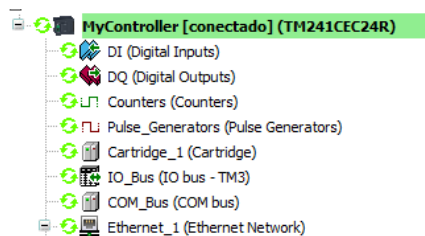
192.168.1.57

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



Ethernet_1 x

Configuración

Parámetros configurados

Nombre de interfaz: EthernetPort0

Nombre de red: my_Device

☐ Dirección IP de DHCP
☐ Dirección IP de BOOTP
☒ Dirección IP fija

Dirección IP: 192 . 168 . 1 . 58

Máscara de subred: 255 . 255 . 255 . 0

Dirección de pasarela: 0 . 0 . 0 . 0

Protocolo Ethernet: Ethernet 2

Velocidad de transferencia: Auto

Parámetros de seguridad

☒ Protocolo SoMachine activo
☒ Servidor Modbus activo
☒ Servidor web activo
☒ Servidor FTP activo
☒ Protocolo de descubrimiento activo
☒ Protocolo SNMP activo
☒ Protocolo WebVisualisation activo

Identificación del dispositivo esclavo

☒ Servidor DHCP activo [Servidor DHCP activo](#)
 Cuando está activo, cada dispositivo que se añadirá al bus de campo puede configurarse para poder identificarlo por su nombre o dirección MAC, en lugar de por su dirección IP.

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

Agregar el dispositivo

Nombre: Administrador_de_Ethernet_Industrial

Acción:
 ☒ Agregar el dispositivo
 ☐ Insertar dispositivo
 ☐ Conectar dispositivo
 ☐ Actualizar el dispositivo

Dispositivo: Schneider Electric

Nombre	Fabricante	Versión
Gestores de protocolo		
Administrador de Ethernet Industrial	Schneider Electric	1.0.12.20
EthernetIP	Schneider Electric	3.3.0.6
ModbusTCP Slave Device	Schneider Electric	3.5.2.6

☐ Mostrar todas las versiones (sólo para expertos)
☐ Mostrar versiones antiguas

Información:

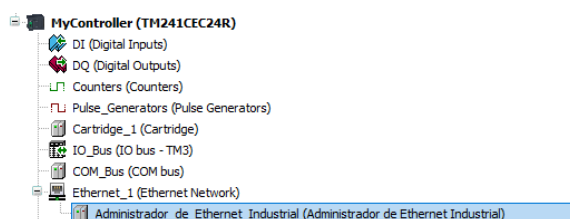
Nombre: Administrador de Ethernet Industrial
 Fabricante: Schneider Electric
 Grupo: Gestores de protocolo
 Versión: 1.0.12.20
 Número de modelo:
 Descripción: Este servicio permite administrar la red industrial. Incorpora un explorador de E/S para supervisar y controlar dispositivos, y un servidor DHCP para la asignación de direcciones IP dinámicas.

Agregar el dispositivo seleccionado como último "subobjeto" de Ethernet_1

☒ (Puede seleccionar otro nodo de destino en el navegador, mientras esta ventana esté abierta.)

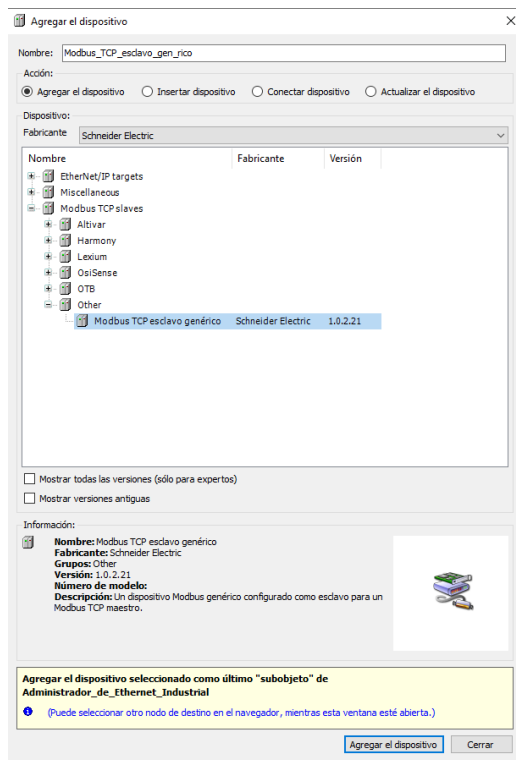
Agregar el dispositivo Cerrar

Nos quedará así

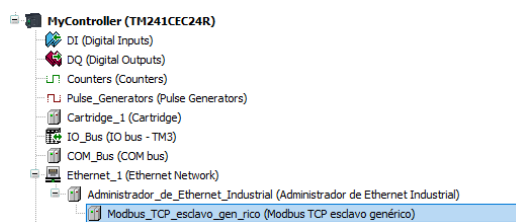


No es necesario configurar nada en este paso

Ahora añadimos el esclavo genérico Modbus (Nuestro Arduino)

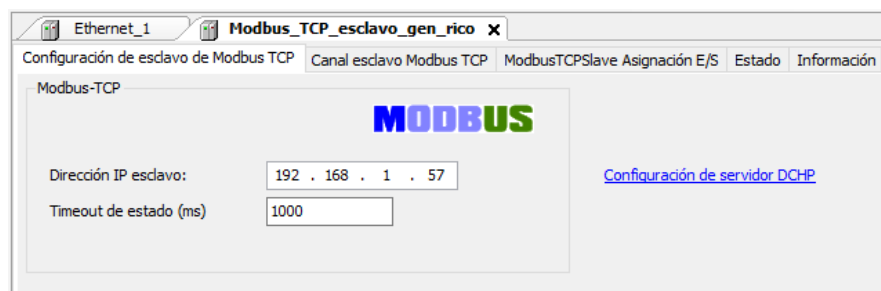


Con lo que nos quedará así:

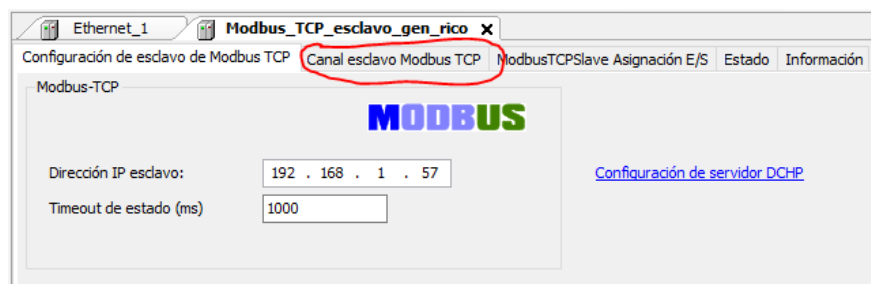


Ahora vamos a configurar el esclavo (nuestro Arduino) haciendo doble click

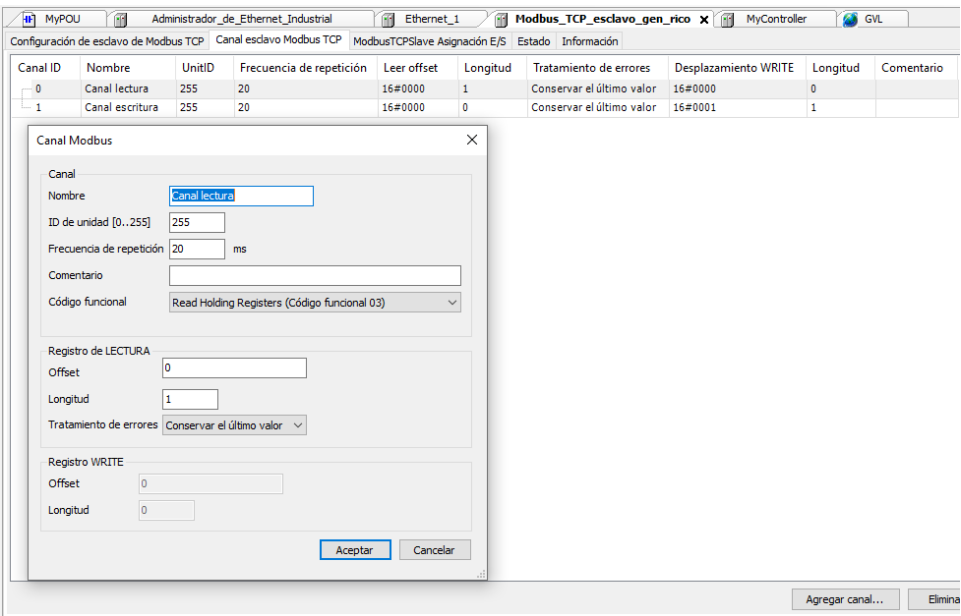
E introducimos la dirección IP de nuestro Arduino



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

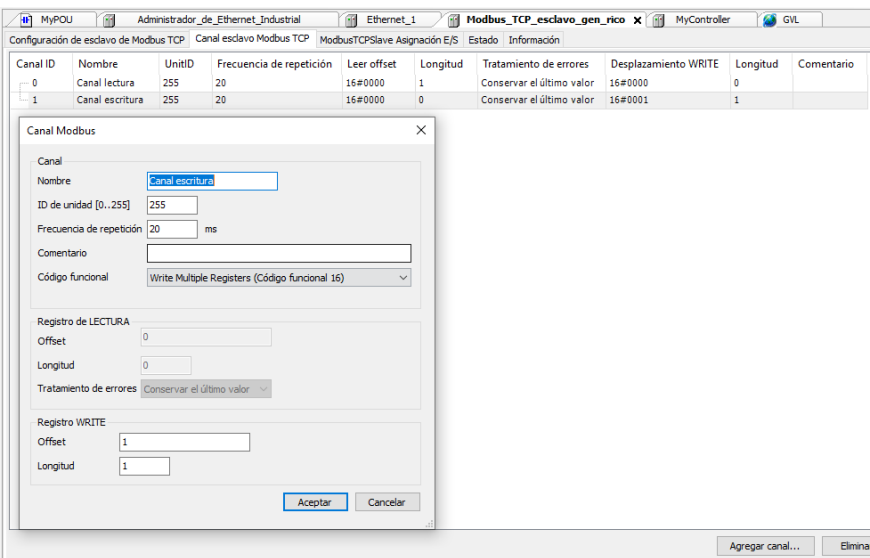


Agregamos un canal de lectura con un offset y un numero de palabras (1 word).

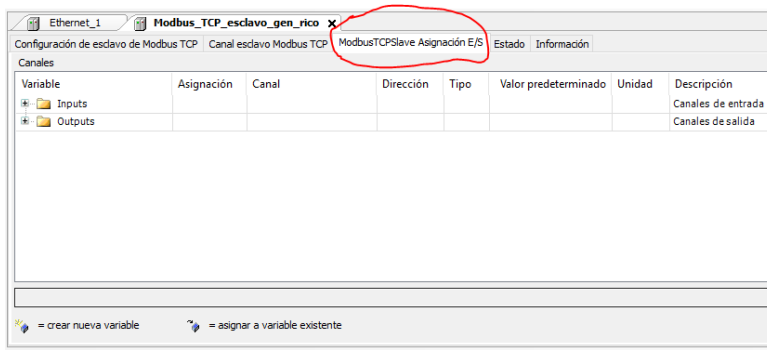


Y un canal de escritura, en una posición distinta pues el buffer que usa Arduino para la comunicación Modbus es el mismo tanto para entradas de datos como para salidas de datos.

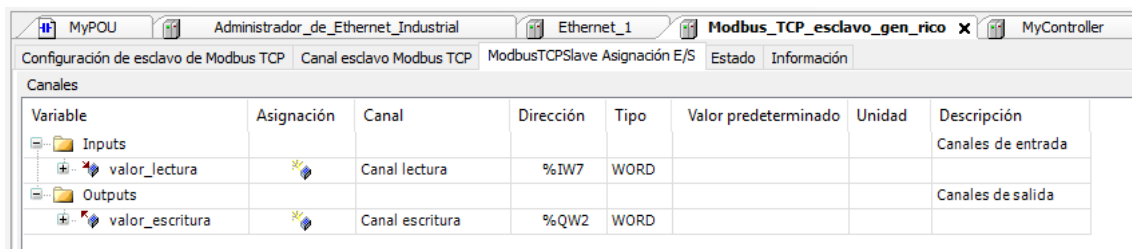
También asignamos 1 word.



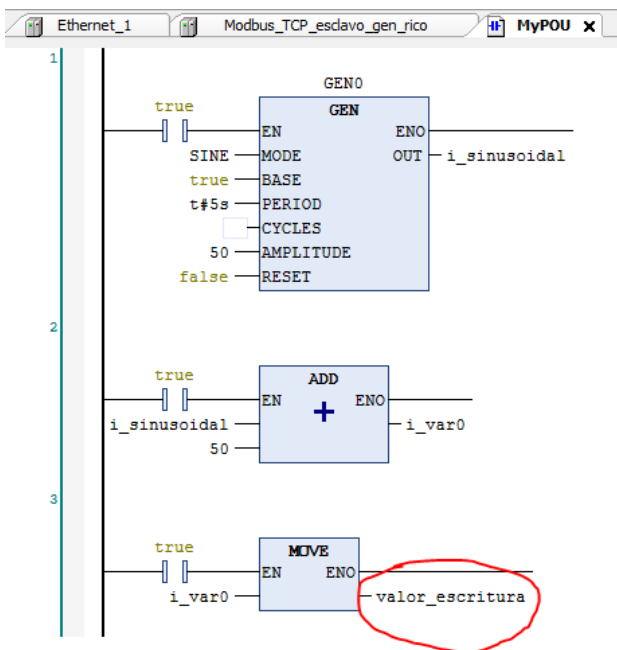
Podemos a su vez declarar variables para usar en nuestro programa de PLC mediante esta pestaña



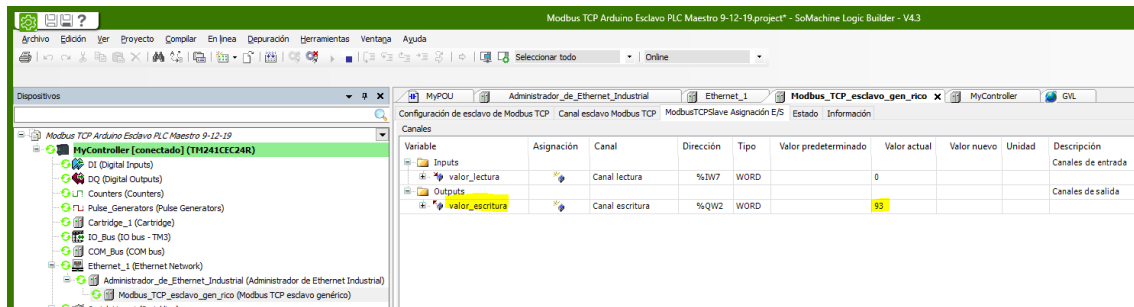
Lo vemos en detalle



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



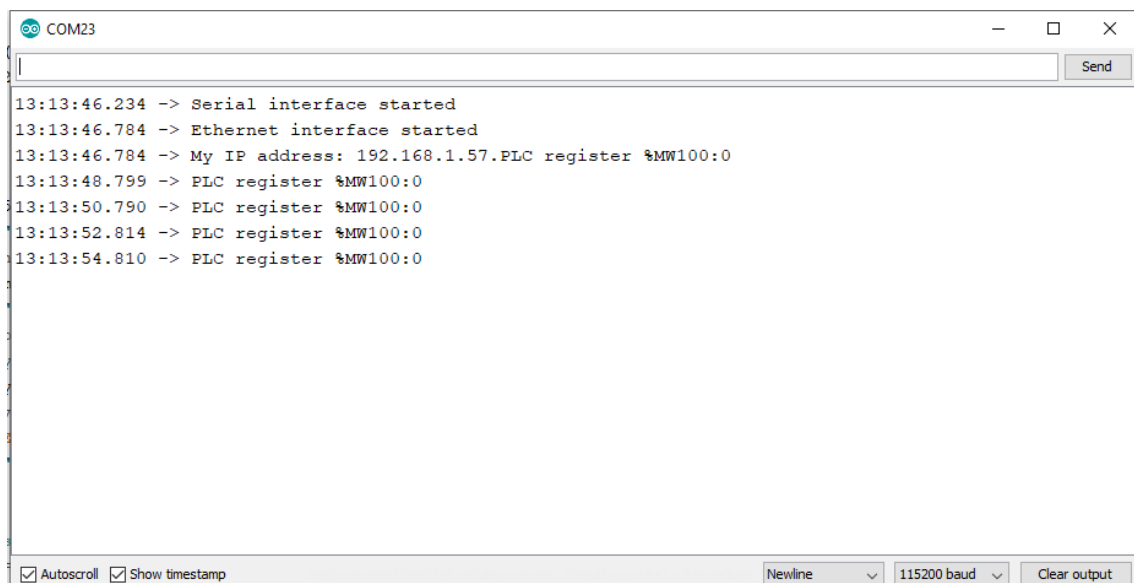
Con esto ya tendríamos todo preparado en el lado del PLC

PLC TEST on MODBUS TCP

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



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



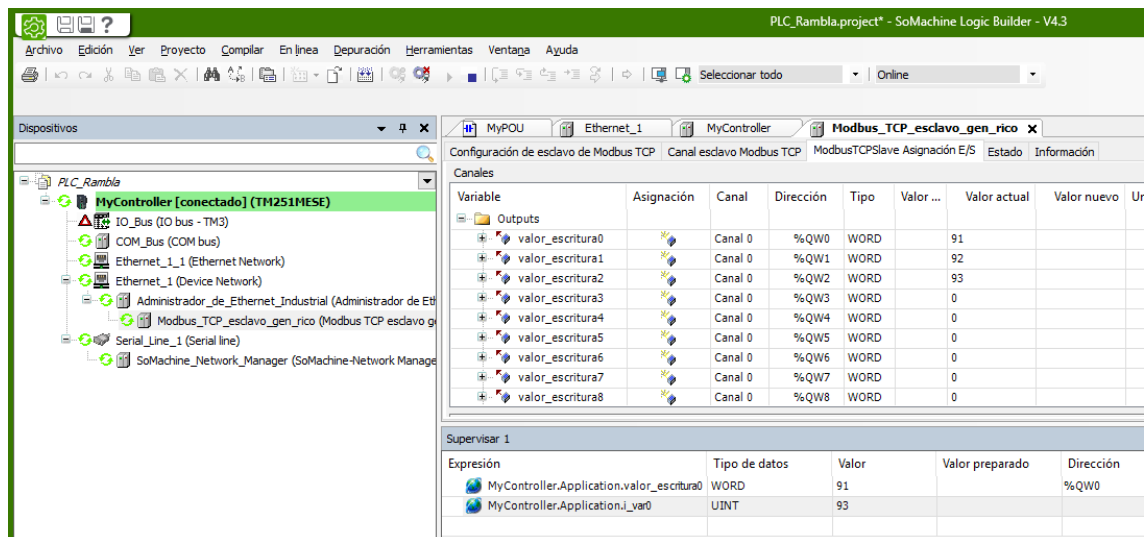
```
Modbus_Rambla | Arduino 1.8.10
File Edit Sketch Tools Help

#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);

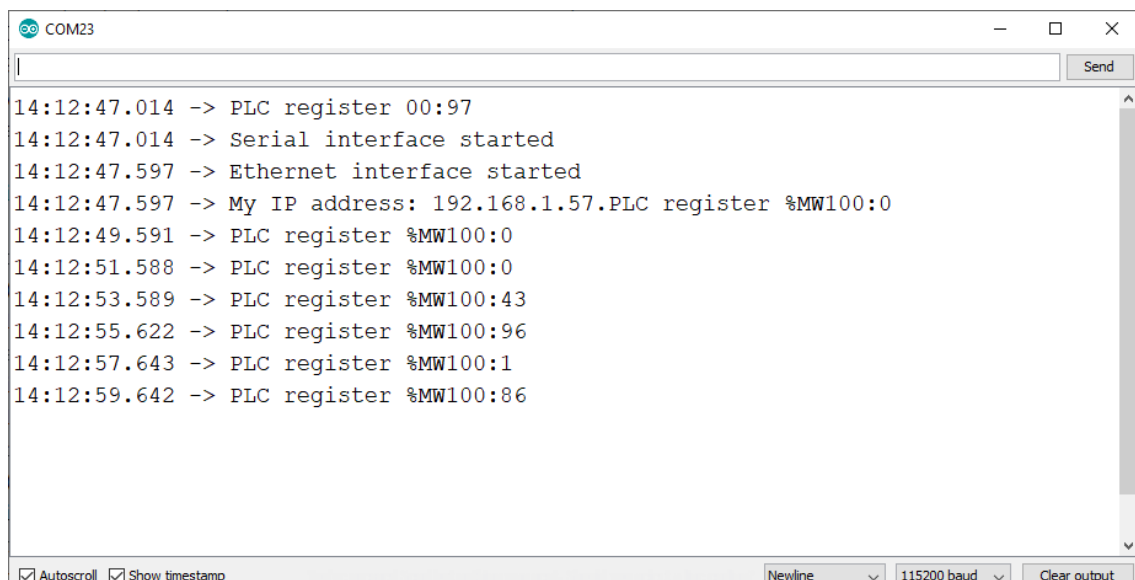
void setup(void)
{
  // serial setup
  Serial.begin(115200);
  Serial.println("Serial interface started");
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet); // start ethernet interface
  Serial.println("Ethernet interface started");
  // print your local IP address:
  Serial.print("My 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 program 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)

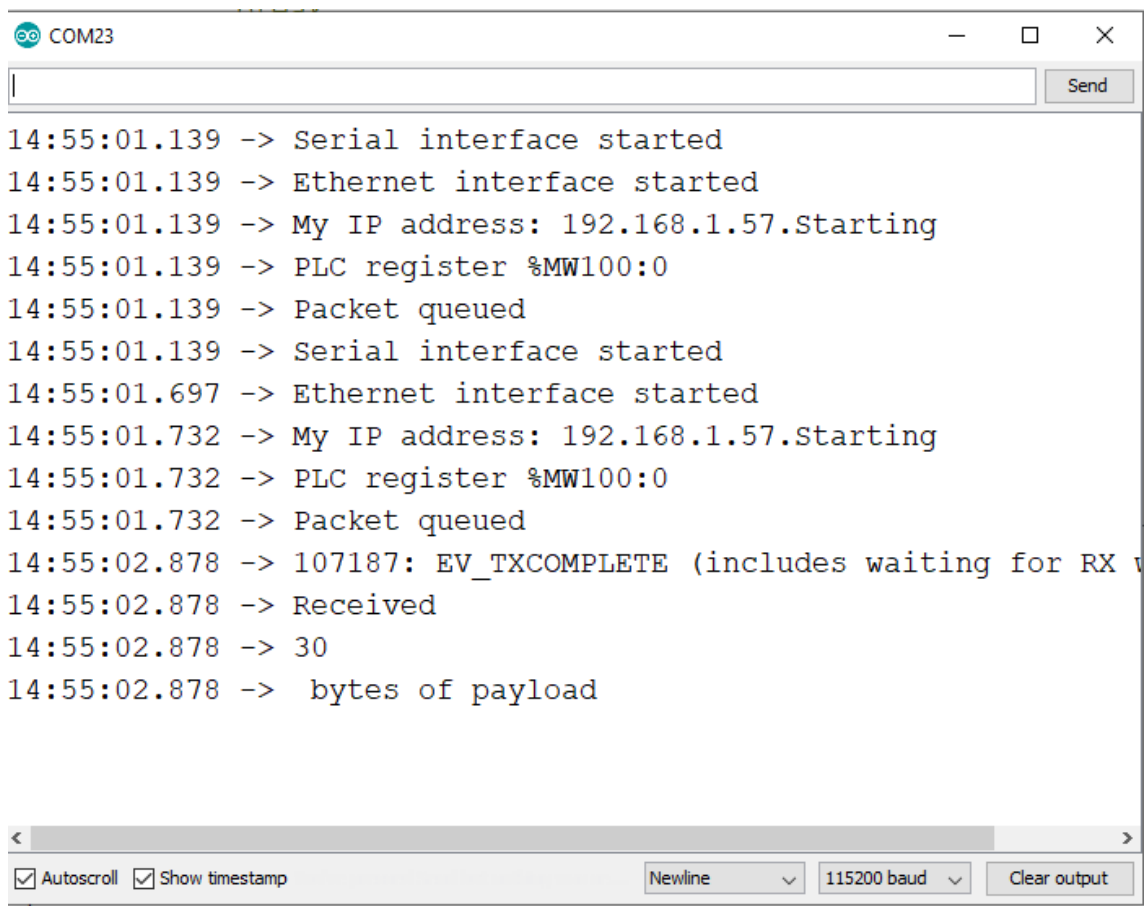
If I try to comment this line

```

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 possible
        // moment
        Serial.print("PLC register %MW100:");
        Serial.println(Mb.MbData[100]);
        sprintf(mydata, format, 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



```

14:55:01.139 -> Serial interface started
14:55:01.139 -> Ethernet interface started
14:55:01.139 -> My IP address: 192.168.1.57.Starting
14:55:01.139 -> PLC register %MW100:0
14:55:01.139 -> Packet queued
14:55:01.139 -> Serial interface started
14:55:01.697 -> Ethernet interface started
14:55:01.732 -> My IP address: 192.168.1.57.Starting
14:55:01.732 -> PLC register %MW100:0
14:55:01.732 -> Packet queued
14:55:02.878 -> 107187: EV_TXCOMPLETE (includes waiting for RX v
14:55:02.878 -> Received
14:55:02.878 -> 30
14:55:02.878 -> bytes of payload

```

< [Progress Bar] >

☒ Autoscroll ☒ Show timestamp Newline 115200 baud Clear output

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 PMuplink

2:57:30 PMuplink

2:57:09 PMuplink

applicationID: "4"

applicationName: "RAMBLAAPP"

deviceName: "Rambla_node"

devEUI: "3e786a68f51c2757"

rxInfo: 1 item

0: 14 keys

gatewayID: "b827ebfffe1bd4e1"

time: null

timeSinceGPSEPOCH: null

rsst: -13

loRaSNR: 9.5

channel: 0

rfChain: 1

board: 0

antenna: 0

BASE64

Decode and Encode

Decode

Encode

Have to deal with Base64 format? Then this site is made for you! Use our super handy online

Decode from Base64 format

Simply enter your data then push the decode button.

MAA

For encoded binaries (like images, documents, etc.) use the file upload form a bit further down on the

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.

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

Send

17:21:48.126 -> PLC register %MW100:56

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

Clear output

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 1

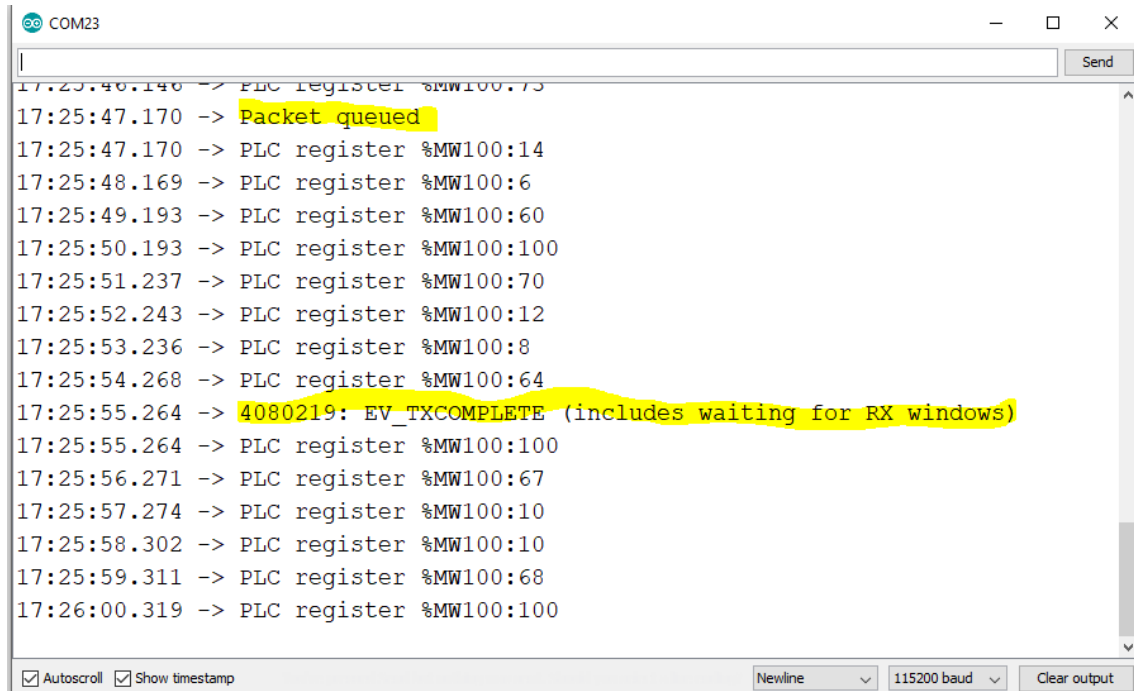
< DECODE >

Decodes your data into the textarea below.

87

Delay (1000)

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



```
COM23
17:25:46.146 -> PLC register %MW100:73
17:25:47.170 -> Packet queued
17:25:47.170 -> PLC register %MW100:14
17:25:48.169 -> PLC register %MW100:6
17:25:49.193 -> PLC register %MW100:60
17:25:50.193 -> PLC register %MW100:100
17:25:51.237 -> PLC register %MW100:70
17:25:52.243 -> PLC register %MW100:12
17:25:53.236 -> PLC register %MW100:8
17:25:54.268 -> PLC register %MW100:64
17:25:55.264 -> 4080219: EV_TXCOMPLETE (includes waiting for RX windows)
17:25:55.264 -> PLC register %MW100:100
17:25:56.271 -> PLC register %MW100:67
17:25:57.274 -> PLC register %MW100:10
17:25:58.302 -> PLC register %MW100:10
17:25:59.311 -> PLC register %MW100:68
17:26:00.319 -> PLC register %MW100:100

☒ Autoscroll ☒ Show timestamp Newline 115200 baud Clear output
```

BASE64

Decode and Encode

 Decode

 Encode

Have to deal with **Base64** format? Then this site is made for you! Use our super handy online

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

BASE64

Decode and Encode

Decode

Encode

Have to deal with **Base64** format? Then this site is made for you! Use our super ha

Decode from Base64 format

Simply enter your data then push the decode button.

Nzg=

For encoded binaries (like images, documents, etc.) use the file upload form a bit further do

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)

< DECODE >

Decodes your data into the textarea below.

78

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 program 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
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
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

BASE64

Decode and Encode

Decode

Encode

Have to deal with **Base64** format? Then this site is made for you.

Decode from Base64 format

Simply enter your data then push the decode button.

JQA=

For encoded binaries (like images, documents, etc.) use the

UTF-8

Source character set.

☐ Decode each line separately (useful for multiple entries).

Live mode OFF

Decodes in real-time when you type

< DECODE >

Decodes your data into the textarea

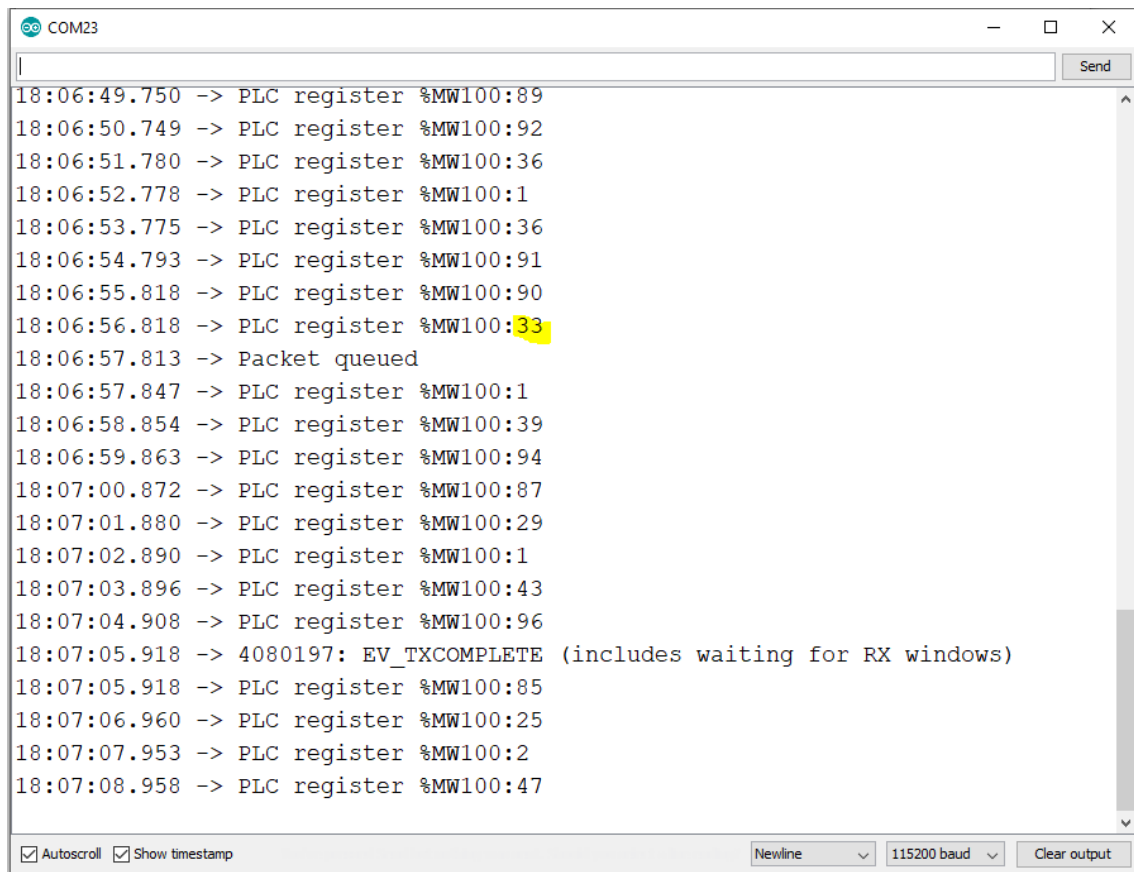
%

37

25

%

So now after decoding, we get the ascii carácter



COM23

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

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.

IQA=

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 I

< DECODE >

Decodes your data into the textarea below.

!

Decimal	Hex	Char
32	20	[SPACE]
33	21	!

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

```
C:\ OpenSSH SSH client

C:\Users\francisco.florensa>ssh admin@192.168.1.214
admin@192.168.1.214's password:

Chirpstack

Documentation and copyright information:
> www.chirpstack.io

Commands:
> sudo gateway-config - configure the gateway
> sudo monit summary - display service monitor

raspberrypi3:~$
```

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

The screenshot shows the configuration of the LoRa Server Gateway in the ThingsBoard interface. The configuration is as follows:

- LoRa Server Gateway** (connected) → **msg.payload**
- LoRa Server Gateway** → **json** (parser) → **data** (filter) → **only data not pings** (filter) → **base 64 decode** (filter) → **msg.payload**

```
9/5/2020 19:40:28 node: fd63ab60.c01d68
application/4/device/3e786a8b51c2757f/rx : msg.payload : buffer[2]
▶ [ 25, 0 ]

9/5/2020 19:40:56 node: fd63ab60.c01d68
application/4/device/3e786a8b51c2757f/rx : msg.payload : buffer[2]
▶ [ 100, 0 ]

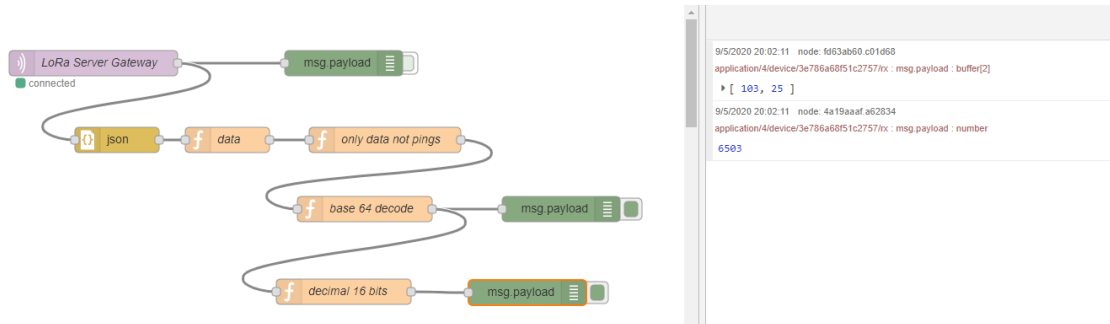
9/5/2020 19:41:24 node: fd63ab60.c01d68
application/4/device/3e786a8b51c2757f/rx : msg.payload : buffer[2]
▶ [ 25, 0 ]

9/5/2020 19:41:53 node: fd63ab60.c01d68
application/4/device/3e786a8b51c2757f/rx : msg.payload : buffer[2]
▶ [ 25, 0 ]
```

```
19:42:44.550 -> PLC register %MW100:82
19:42:45.584 -> PLC register %MW100:97
19:42:46.576 -> PLC register %MW100:47
19:42:47.607 -> PLC register %MW100:2
19:42:48.604 -> PLC register %MW100:25
19:42:49.614 -> Packet queued
19:42:49.614 -> PLC register %MW100:85
19:42:50.641 -> PLC register %MW100:95
19:42:51.664 -> PLC register %MW100:42
19:42:52.670 -> PLC register %MW100:1
19:42:53.685 -> PLC register %MW100:30
19:42:54.688 -> PLC register %MW100:87
19:42:55.685 -> PLC register %MW100:93
19:42:56.717 -> PLC register %MW100:39
19:42:57.727 -> 363531716: EV_TXCOMPLETE (includes waiting for RX windows)
19:42:57.727 -> PLC register %MW100:1
19:42:58.737 -> PLC register %MW100:33
19:42:59.743 -> PLC register %MW100:90
19:43:00.753 -> PLC register %MW100:91
19:43:01.760 -> PLC register %MW100:34
19:43:02.777 -> PLC register %MW100:1
19:43:03.782 -> PLC register %MW100:37
```

```
9/5/2020 19:42:49 node: fd63ab60.c01d688
application/4/device/3e786a68f5c2757f/x: msg payload : buffer[2]
▶ [ 25, 0 ]
```

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



Yes

The screenshot shows the COM23 terminal window with the following data:

```

20:02:51.790 -> PLC register %MW100:22251
20:02:52.803 -> PLC register %MW100:59032
20:02:53.802 -> PLC register %MW100:59033
20:02:54.804 -> PLC register %MW100:22251
20:02:55.814 -> PLC register %MW100:24
20:02:56.858 -> PLC register %MW100:24618
20:02:57.867 -> PLC register %MW100:60434
20:02:58.875 -> PLC register %MW100:57482
20:02:59.880 -> PLC register %MW100:19944
20:03:00.889 -> PLC register %MW100:210
20:03:01.898 -> PLC register %MW100:26223
20:03:02.897 -> PLC register %MW100:61677
20:03:03.926 -> PLC register %MW100:55791
20:03:04.935 -> PLC register %MW100:17709
20:03:05.933 -> PLC register %MW100:581
20:03:06.966 -> PLC register %MW100:28660
20:03:07.954 -> Packet queued
20:03:07.988 -> PLC register %MW100:62757
20:03:08.983 -> PLC register %MW100:53970
20:03:09.993 -> PLC register %MW100:14864
20:03:11.000 -> PLC register %MW100:1135
20:03:12.034 -> PLC register %MW100:31943
20:03:13.047 -> PLC register %MW100:63666
20:03:14.056 -> PLC register %MW100:52028
20:03:15.069 -> PLC register %MW100:12848

```

The debug window on the right shows the following data:

```

9/5/2020 20:02:11 node: f063ab60.c01d68
application/4/device/3e786a68f51c2757/rx: msg.payload: buffer[2]
▶ [ 103, 25 ]

9/5/2020 20:02:11 node: 4a19aaaf.a62834
application/4/device/3e786a68f51c2757/rx: msg.payload: number
6503

9/5/2020 20:02:39 node: f063ab60.c01d68
application/4/device/3e786a68f51c2757/rx: msg.payload: buffer[2]
▶ [ 37, 245 ]

9/5/2020 20:02:40 node: 4a19aaaf.a62834
application/4/device/3e786a68f51c2757/rx: msg.payload: number
62757

9/5/2020 20:03:08 node: f063ab60.c01d68
application/4/device/3e786a68f51c2757/rx: msg.payload: buffer[2]
▶ [ 244, 111 ]

9/5/2020 20:03:08 node: 4a19aaaf.a62834
application/4/device/3e786a68f51c2757/rx: msg.payload: number
28660

```

The screenshot shows the COM23 terminal window with the following data:

```

20:05:26.632 -> PLC register %MW100:16265
20:05:27.627 -> PLC register %MW100:54591
20:05:28.651 -> PLC register %MW100:62416
20:05:29.644 -> Packet queued
20:05:29.679 -> PLC register %MW100:27845
20:05:30.666 -> PLC register %MW100:437
20:05:31.705 -> PLC register %MW100:18444
20:05:32.714 -> PLC register %MW100:56934
20:05:33.725 -> PLC register %MW100:61281
20:05:34.736 -> PLC register %MW100:25419
20:05:35.744 -> PLC register %MW100:127
20:05:36.756 -> PLC register %MW100:20705
20:05:37.766 -> 448524828: EV_TXCOMPLETE (includes waiting for RX windows)
20:05:37.766 -> PLC register %MW100:58532
20:05:38.777 -> PLC register %MW100:58532

```

The debug window on the right shows the following data:

```

9/5/2020 20:05:01 node: f063ab60.c01d68
application/4/device/3e786a68f51c2757/rx: msg.payload: buffer[2]
▶ [ 131, 23 ]

9/5/2020 20:05:01 node: 4a19aaaf.a62834
application/4/device/3e786a68f51c2757/rx: msg.payload: number
6019

9/5/2020 20:05:29 node: f063ab60.c01d68
application/4/device/3e786a68f51c2757/rx: msg.payload: buffer[2]
▶ [ 208, 243 ]

9/5/2020 20:05:30 node: 4a19aaaf.a62834
application/4/device/3e786a68f51c2757/rx: msg.payload: number
62416

```

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

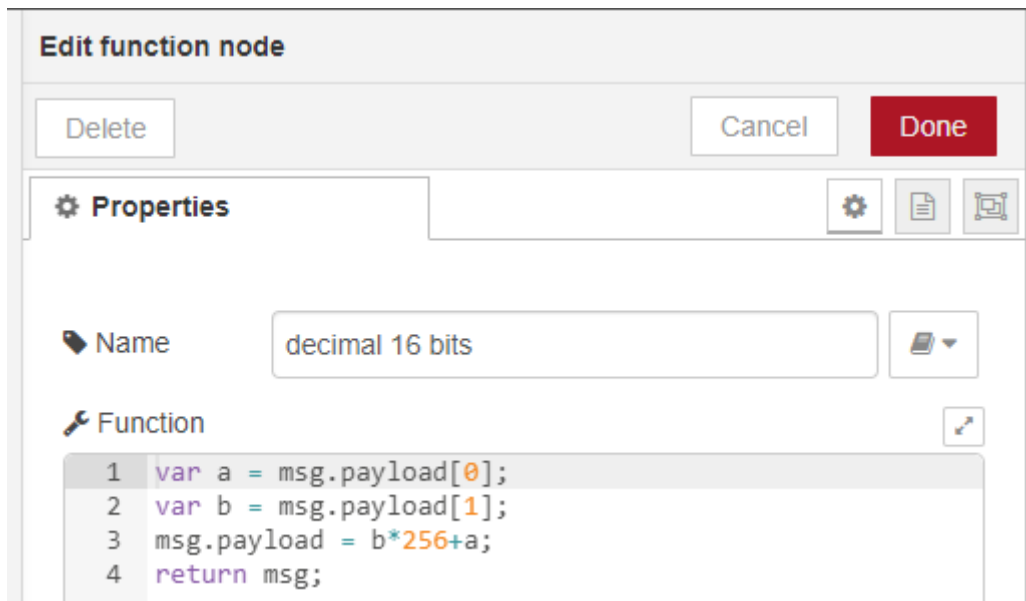
var b = new Buffer (msg.payload, 'base64');

2

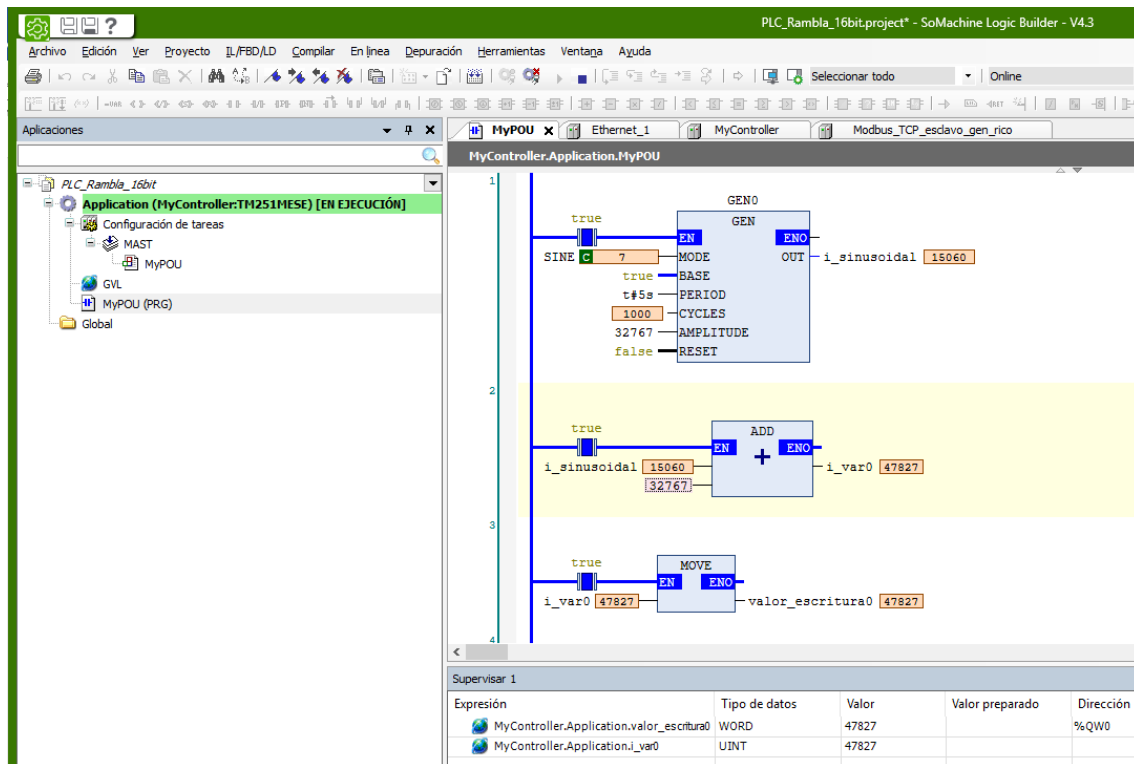
msg.payload = b;

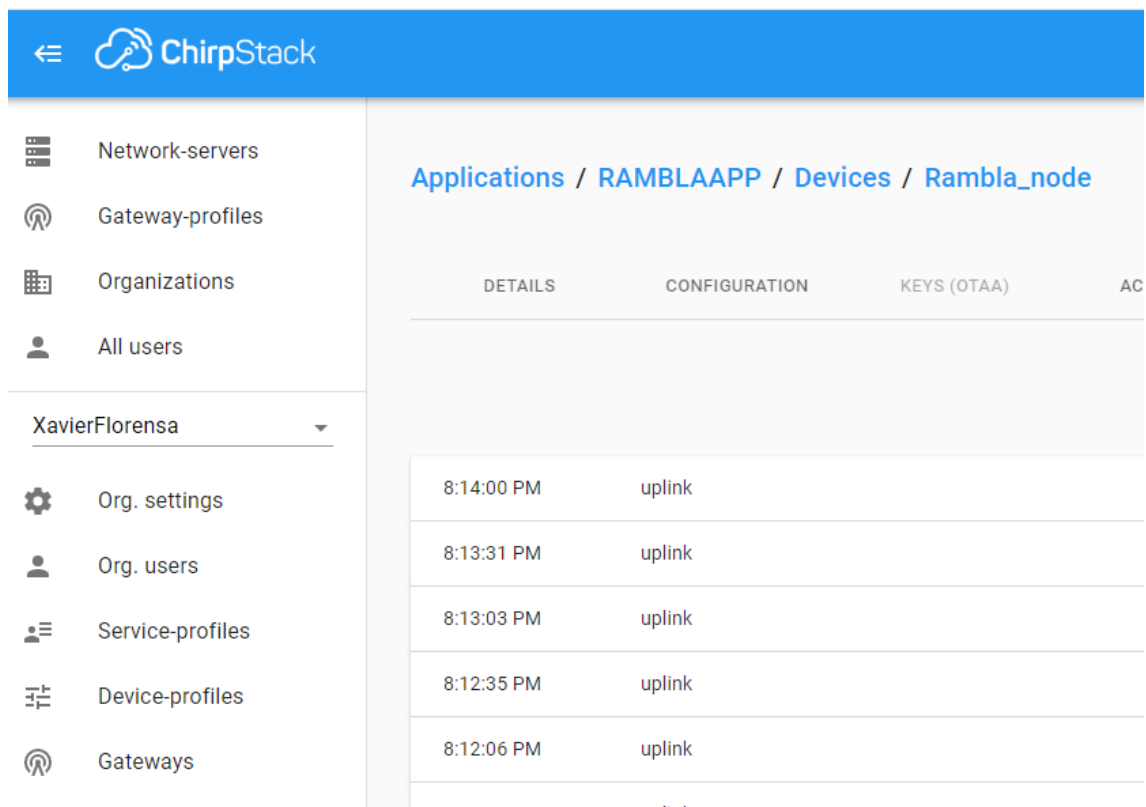
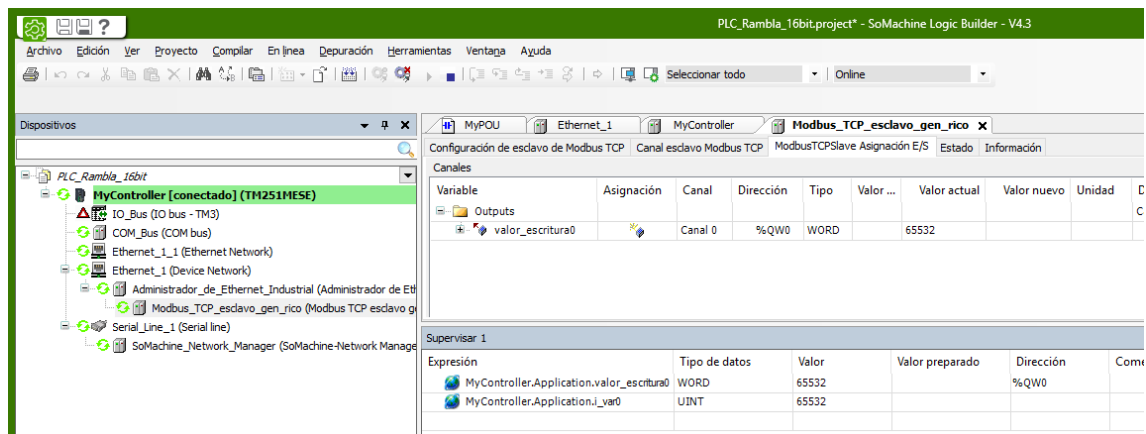
3

return msg;



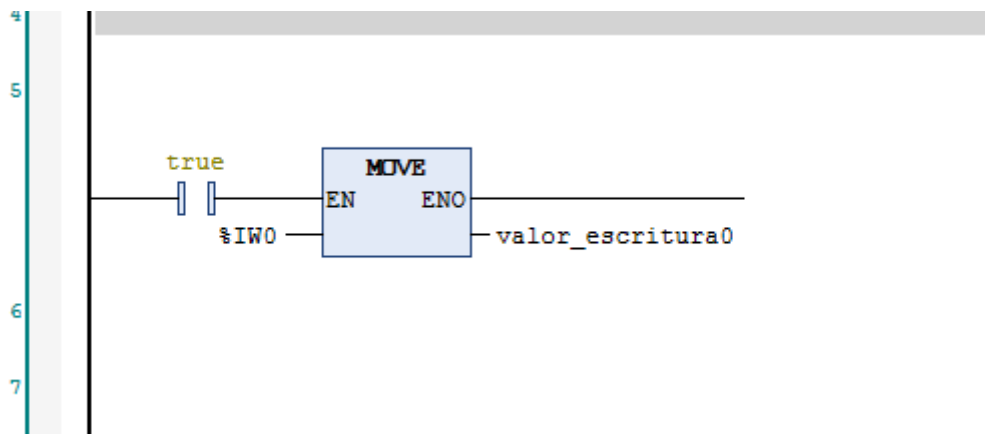
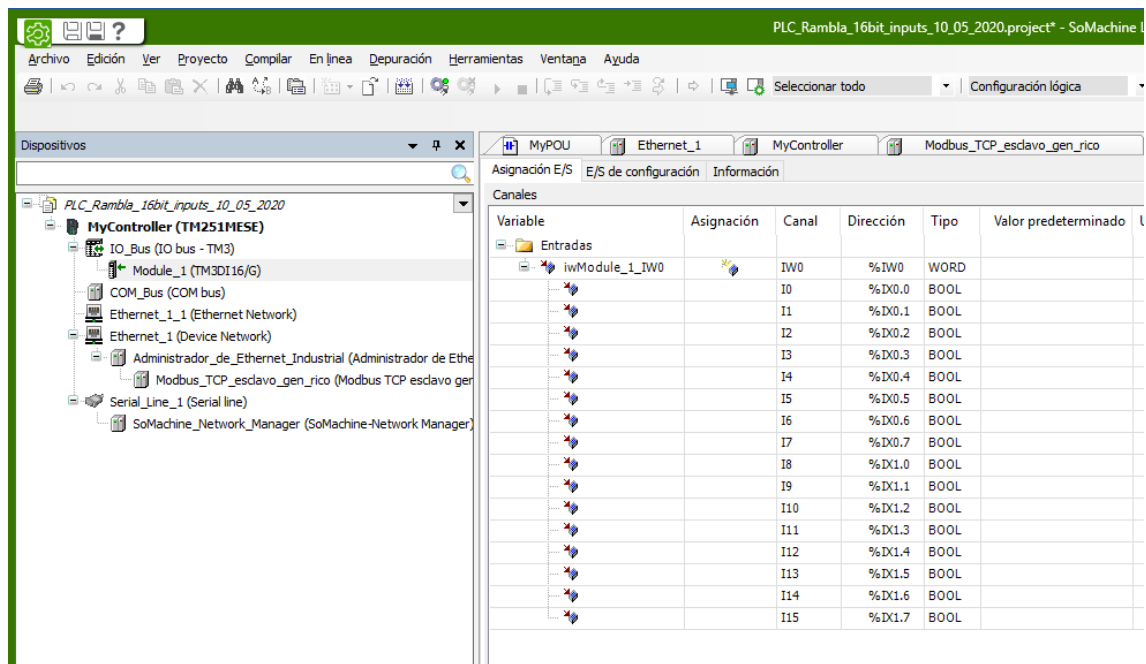
This is the PLC program



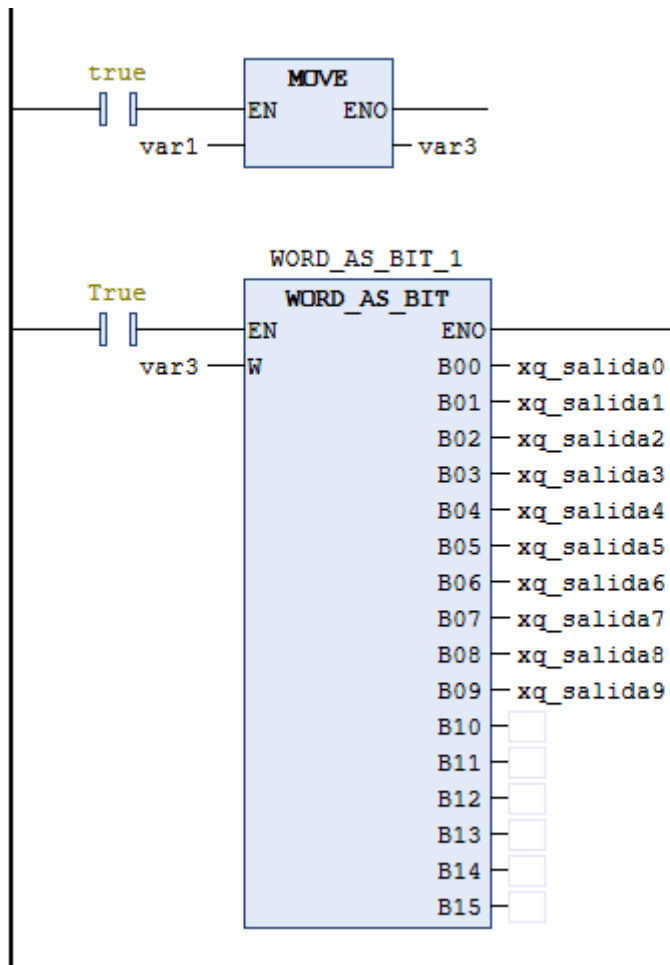


Next step is to use the digital inputs and translate to the digital outputs of a master PLC

LoRa PLC programm



This is the PLC program 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

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

Login

Username *

Password *

LOGIN

And it is still working!

The screenshot shows the ChirpStack web interface. On the left is a sidebar menu with options: Network-servers, Gateway-profiles, Organizations, All users, XavierFlorensa (selected), Org. settings, Org. users, Service-profiles, Device-profiles, Gateways, Applications, and Multicast-groups. The main content area shows the breadcrumb 'Applications / RAMBLAAPP / Devices / Rambla_node' and a tabbed interface with 'DETAILS', 'CONFIGURATION', 'KEYS (OTAA)', 'ACTIVATION', and 'DEVICE DATA' (selected). Under the 'DEVICE DATA' tab, there is a table with two rows of data:

Time	Action
8:16:44 PM	uplink
8:16:25 PM	uplink

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/dhcpd.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

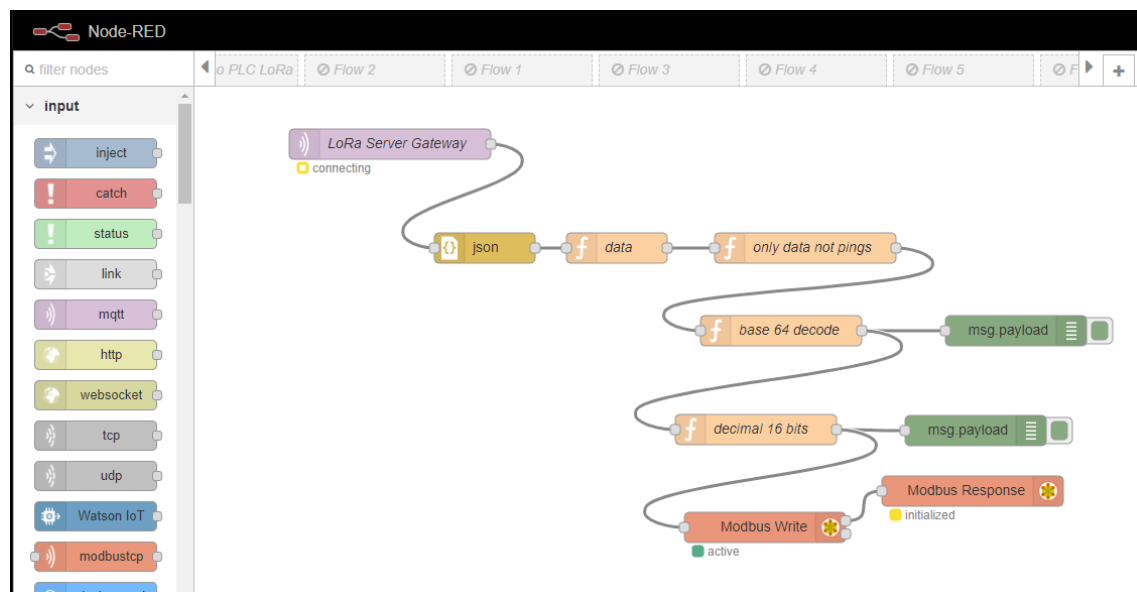
<https://www.raspberrypi.org/forums/viewtopic.php?t=221060#p1357512>

And stop node red

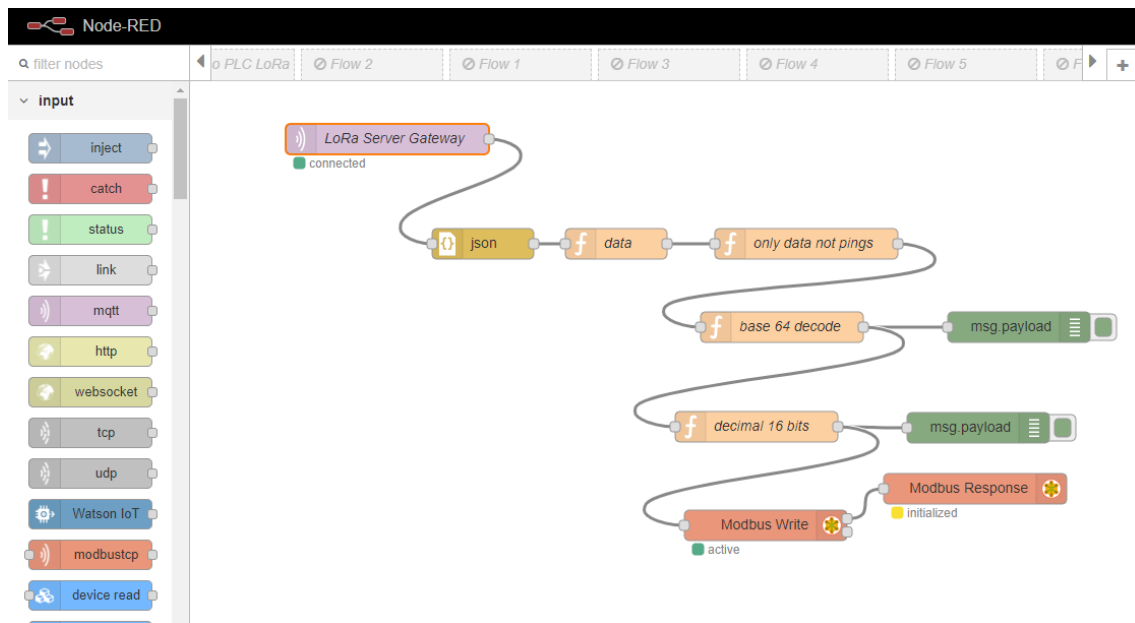
systemctl stop nodered

Start node red

Systemctl start nodered



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



Voila connected

And running

```

11/5/2020 20:49:13 node: f94a62d1.76f38
application/4/device/3e786a68f51c2757/rx : msg.payload : buffer[2]
▶ [ 2, 0 ]

11/5/2020 20:49:13 node: dbff2c29.5bcff
application/4/device/3e786a68f51c2757/rx : msg.payload : number
2

11/5/2020 20:49:31 node: f94a62d1.76f38
application/4/device/3e786a68f51c2757/rx : msg.payload : buffer[2]
▶ [ 2, 0 ]

11/5/2020 20:49:31 node: dbff2c29.5bcff
application/4/device/3e786a68f51c2757/rx : msg.payload : number
2

```

i!!

But let's make it all in one Raspberry

INSTALLATION OF CHIRPSTACK OVER RASPBIAN

```
pi@loraserver: ~
Microsoft Windows [Versión 10.0.17763.805]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\francisco.florensa>ssh pi@loraserver.local
pi@loraserver.local's password:
Linux loraserver 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 12 22:53:13 2020 from fe80::ddec:ca4:3152:1e82%eth0
pi@loraserver:~ $
```

sudo apt update

sudo apt upgrade

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

Also explained here

<https://www.youtube.com/watch?v=FnTP7t47DI>

jwt

SHCktXRUSLIySEl/ikZilf6UIW/pBcXFD88zJnKtluA=

```
File Edit Tabs Help

pi@loraserver:~ $ openssl rand -base64 32
SHCktXRUSLIySEl/ikZilf6UIW/pBcXFD88zJnKtluA=
pi@loraserver:~ $
```

Missing (optional)

Optional: install ChirpStack Gateway Bridge on the gateway


It is advised to run ChirpStack Gateway Bridge on each gateway itself, to enable a secure connection between your gateways and your server.

As there are many types of gateways available, please refer to the ChirpStack Gateway Bridge instructions for [installing ChirpStack Gateway Bridge on the gateway](#).

Create Gateway and device

MAC address of Gateway

Advanced IP scanner

	192.168.1.101	192.168.1.101	Raspberry Pi Foundation	B8:27:EB:1B:D4:E1
---	---------------	---------------	-------------------------	-------------------

<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
```

```
May 13 23:57:39 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:39+02:00" level=error msg="schedule next device-queue item error" ctx_id=23080020-811a-4c88-800d-11fd703e141 dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"  
May 13 23:57:40 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:40+02:00" level=error msg="finished unary call with code Unknown" ctx_id=002e978d-9351-4ebc-bfaa-9554ee08a54 error="rpc error: code = Unknown desc =  
lorawan/band: invalid data-rate" grpc.code=Unknown grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService grpc.start_time="2020-05-13T23:57:40+02:00" grpc.time_ms=436 peer.address="[::1]:37028" span.kind=server  
system=grpc  
May 13 23:57:40 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:40+02:00" level=error msg="schedule next device-queue item error" ctx_id=1a134bd4-d0d0-4319-80be-acbbe013acaa dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"  
May 13 23:57:41 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:41+02:00" level=error msg="finished unary call with code Unknown" ctx_id=82772d93-552a-40ef-9afb-be4abedbc45e error="rpc error: code = Unknown desc =  
lorawan/band: invalid data-rate" grpc.code=Unknown grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService grpc.start_time="2020-05-13T23:57:41+02:00" grpc.time_ms=389 peer.address="127.0.0.1:58906" span.kind=server  
system=grpc  
May 13 23:57:41 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:41+02:00" level=error msg="schedule next device-queue item error" ctx_id=1d05160b-0a7f-43b8-8845-d2f4e2aa2c6 dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"  
May 13 23:57:42 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:42+02:00" level=error msg="finished unary call with code Unknown" ctx_id=a30a1043-ba60-44a1-af6b-7a61eb5406f9 error="rpc error: code = Unknown desc =  
lorawan/band: invalid data-rate" grpc.code=Unknown grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService grpc.start_time="2020-05-13T23:57:42+02:00" grpc.time_ms=403 peer.address="[::1]:37028" span.kind=server  
system=grpc  
May 13 23:57:42 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:42+02:00" level=error msg="schedule next device-queue item error" ctx_id=5e5d9325-f106-41cf-9c23-b674f7d1f65 dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"  
May 13 23:57:43 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:43+02:00" level=error msg="finished unary call with code Unknown" ctx_id=d5a2aa0b-a480-4ebc-82a0-bdaicdfc800 error="rpc error: code = Unknown desc =  
lorawan/band: invalid data-rate" grpc.code=Unknown grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService grpc.start_time="2020-05-13T23:57:43+02:00" grpc.time_ms=385 peer.address="127.0.0.1:58906" span.kind=server  
system=grpc  
May 13 23:57:43 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:43+02:00" level=error msg="schedule next device-queue item error" ctx_id=3cd3ec9c-40f0-4f0b-bbd5-50f1ae8de0e3 dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"  
May 13 23:57:44 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:44+02:00" level=error msg="finished unary call with code Unknown" ctx_id=00774bbe-0b01-47ef-a10b-0ef5abdbcf70b error="rpc error: code = Unknown desc =  
lorawan/band: invalid data-rate" grpc.code=Unknown grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService grpc.start_time="2020-05-13T23:57:44+02:00" grpc.time_ms=426 peer.address="[::1]:37028" span.kind=server  
system=grpc  
May 13 23:57:44 lorasever chirpstack-network-server[415]: time="2020-05-13T23:57:44+02:00" level=error msg="schedule next device-queue item error" ctx_id=bf72322a-03ac-4631-be37-e0b727290f73 dev_eui=3e786a60f51c2757 error="get dev  
ice gateway XXXInfoSet error: object does not exist"
```

Print the ChirpStack Application Server log-output:

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

```
pi@raspberrypi:~$ sudo journalctl -f -n 100 -u chirpstack-application-server
May 14 00:02:00 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:00+02:00" level=error msg="finished client unary call" ctx_id="" error="rpc error: code = Unknown desc = lorawan/band: invalid data-rate"  
grpc.code=Unknown grpc.ctx_id=ab217b9a-dc27-40ae-b20f-c3431ceab754 grpc.duration=3.01151ms grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService span.kind=client system=grpc  
May 14 00:02:00 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:00+02:00" level=error msg="send gateway ping error: send ping error: send proprietary payload error: rpc error: code = Unknown desc = lorawan/  
band: invalid data-rate"  
May 14 00:02:01 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:01+02:00" level=info msg="gateway ping created" ctx_id=a48768e2-d04d-4b5d-9b42-4d5a2b46773f dr=60 frequency=60 gateway_mac=0bf47994f5ddf36e id  
=4356  
May 14 00:02:01 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:01+02:00" level=error msg="finished client unary call" ctx_id="" error="rpc error: code = Unknown desc = lorawan/band: invalid data-rate"  
grpc.code=Unknown grpc.ctx_id=a80b2a19-cf48-470c-826a-2a6d9f80b3e1 grpc.duration=2.68590ms grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService span.kind=client system=grpc  
May 14 00:02:01 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:01+02:00" level=error msg="send gateway ping error: send ping error: send proprietary payload error: rpc error: code = Unknown desc = lorawan/  
band: invalid data-rate"  
May 14 00:02:02 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:02+02:00" level=info msg="gateway ping created" ctx_id=d3d584ab-6a1b-4e03-9be1-8bea51463bb3 dr=60 frequency=60 gateway_mac=0bf47994f5ddf36e id  
=4357  
May 14 00:02:02 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:02+02:00" level=error msg="finished client unary call" ctx_id="" error="rpc error: code = Unknown desc = lorawan/band: invalid data-rate"  
grpc.code=Unknown grpc.ctx_id=a390bcb3-9a0b-4079-9300-2a61d1a2a515 grpc.duration=3.93015ms grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService span.kind=client system=grpc  
May 14 00:02:02 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:02+02:00" level=error msg="send gateway ping error: send ping error: send proprietary payload error: rpc error: code = Unknown desc = lorawan/  
band: invalid data-rate"  
May 14 00:02:03 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:03+02:00" level=info msg="gateway ping created" ctx_id=e1350f5f-1e0f-4c4a-a03d-203c00f09c3b dr=60 frequency=60 gateway_mac=0bf47994f5ddf36e id  
=4358  
May 14 00:02:03 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:03+02:00" level=error msg="finished client unary call" ctx_id="" error="rpc error: code = Unknown desc = lorawan/band: invalid data-rate"  
grpc.code=Unknown grpc.ctx_id=391e8326-0587-4911-954e-5abcc8c8022b grpc.duration=2.63723ms grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService span.kind=client system=grpc  
May 14 00:02:03 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:03+02:00" level=error msg="send gateway ping error: send ping error: send proprietary payload error: rpc error: code = Unknown desc = lorawan/  
band: invalid data-rate"  
May 14 00:02:04 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:04+02:00" level=info msg="gateway ping created" ctx_id=07432508-7cd3-46e7-0b04-4c1c5231423e dr=60 frequency=60 gateway_mac=0bf47994f5ddf36e id  
=4359  
May 14 00:02:04 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:04+02:00" level=error msg="finished client unary call" ctx_id="" error="rpc error: code = Unknown desc = lorawan/band: invalid data-rate"  
grpc.code=Unknown grpc.ctx_id=cedd2b40-70ea-4157-8098-827a0adac177 grpc.duration=2.9475ms grpc.method=SendProprietaryPayload grpc.service=ns.NetworkServerService span.kind=client system=grpc  
May 14 00:02:04 lorasever chirpstack-application-server[414]: time="2020-05-14T00:02:04+02:00" level=error msg="send gateway ping error: send ping error: send proprietary payload error: rpc error: code = Unknown desc = lorawan/  
band: invalid data-rate"
```