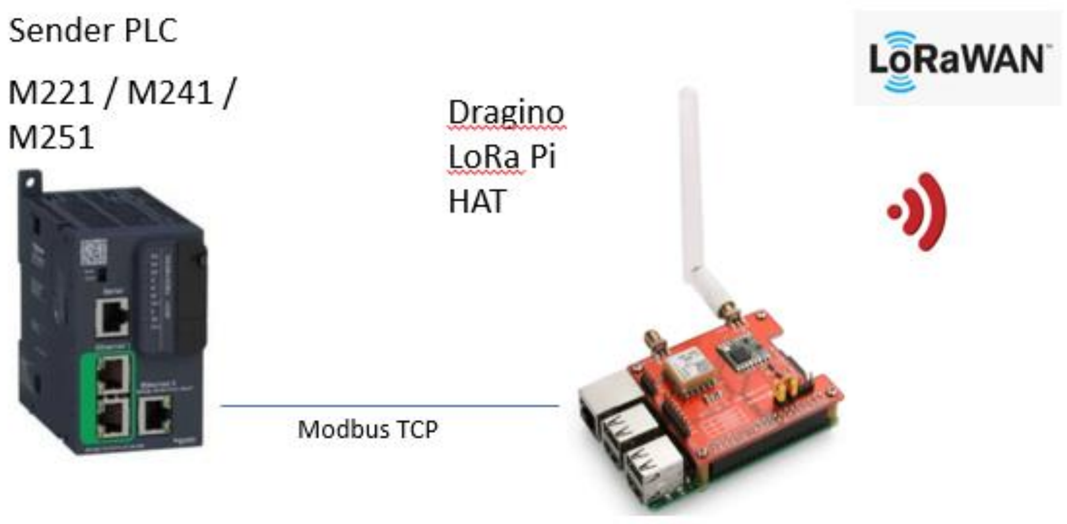
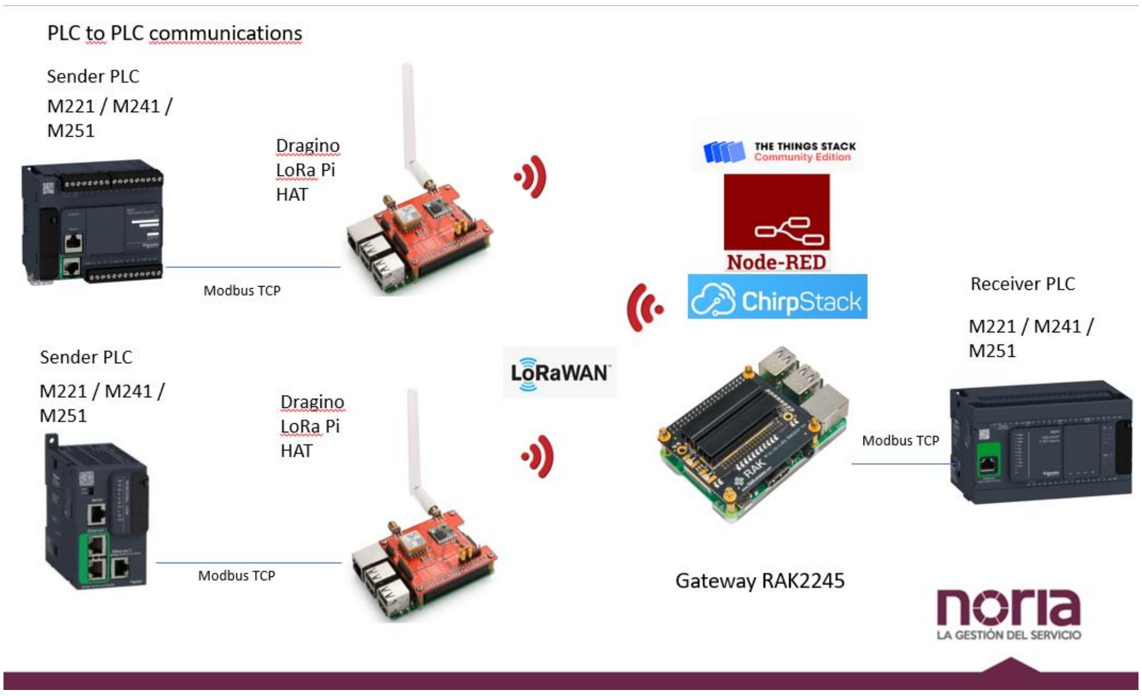
Modbus TCP to LoRaWAN converter



To be able to get Modbus TCP data from any device (Power Meter, Solar inverter, etc) and sending thu LoRaWAN

Or even performing PLC to PLC communications



We have succeed with this one:

https://github.com/pmanzoni/raspi-lmic

and the file as data Exchange method with this one

https://upcommons.upc.edu/bitstream/handle/2117/191014/Sistema+de+comunicaci%F3n+de+largo+alcance+(LoRa),+para+la+gesti%F3n+y+el+monitoreo+del+agua+en+comunidades+rurales.pdf?sequence=1

Other Sources

https://github.com/wklenk/lmic-rpi-lora-gps-hat

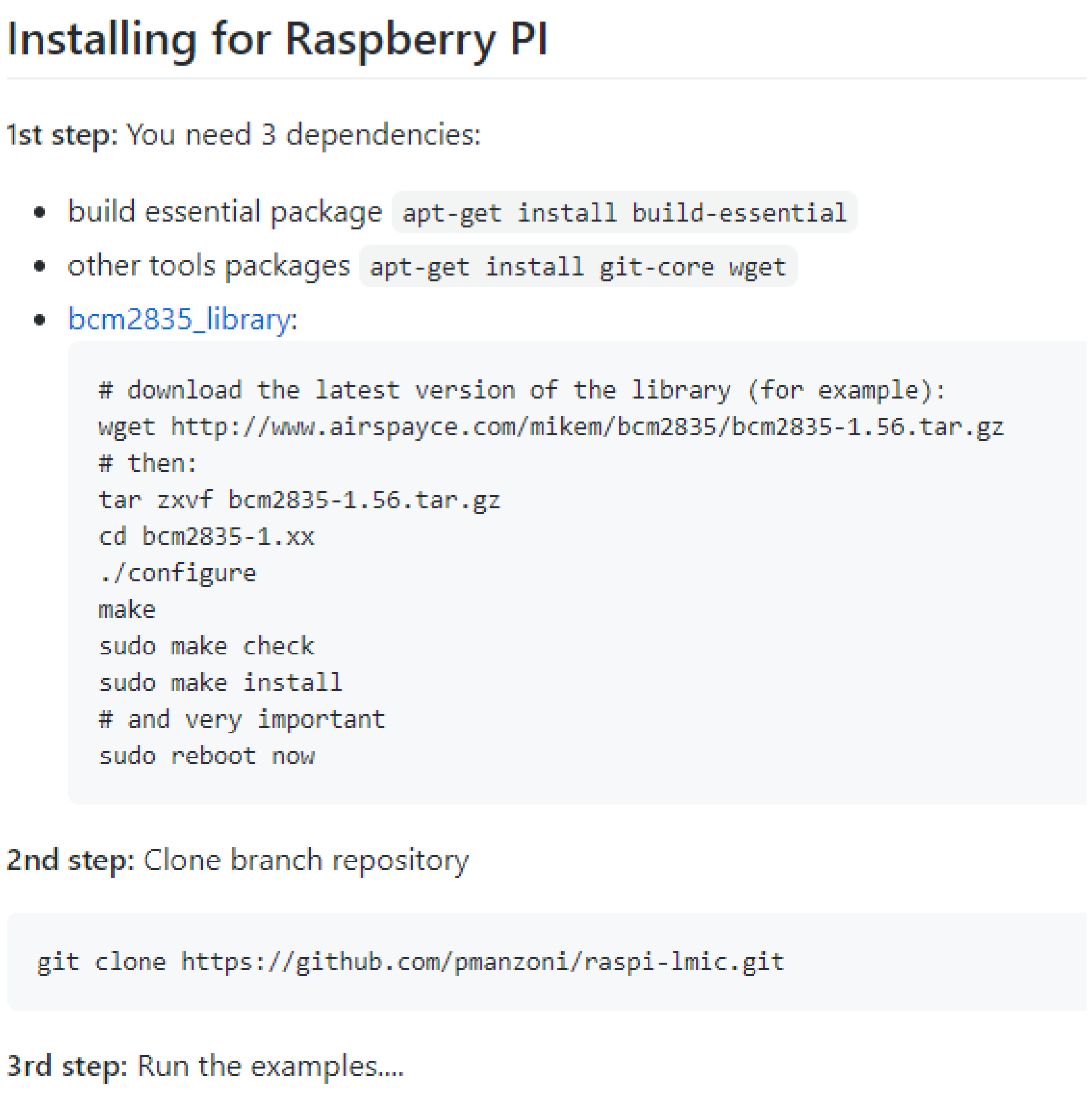
https://github.com/ernstdevreede/lmic_pi

https://github.com/hallard/arduino-lmic/tree/rpi

Step 1

Setting up Dragino Pi HAT to send messages to a near Gateway

Raspberry Pi configuration



Installing for Raspberry PI

1st step: You need 3 dependencies:

- build essential package `apt-get install build-essential`
- other tools packages `apt-get install git-core wget`
- bcm2835_library:

```
# download the latest version of the library (for example):
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.56.tar.gz
# then:
tar zxvf bcm2835-1.56.tar.gz
cd bcm2835-1.xx
./configure
make
sudo make check
sudo make install
# and very important
sudo reboot now
```

2nd step: Clone branch repository

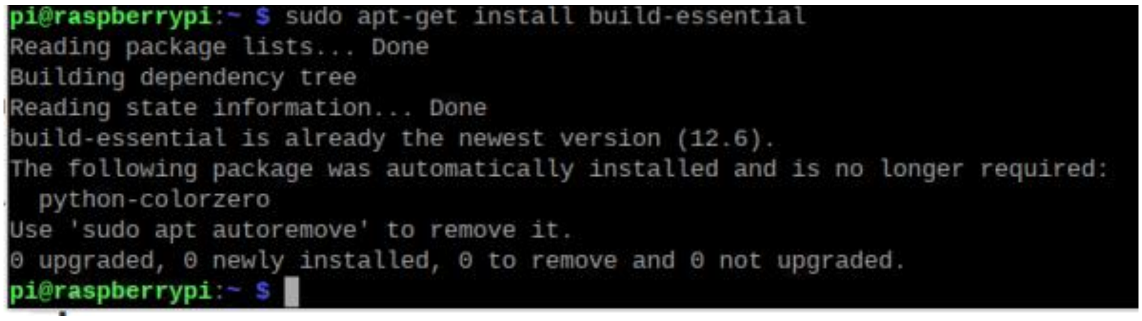
```
git clone https://github.com/pmanzoni/raspi-lmic.git
```

3rd step: Run the examples....

Enable SPI interface

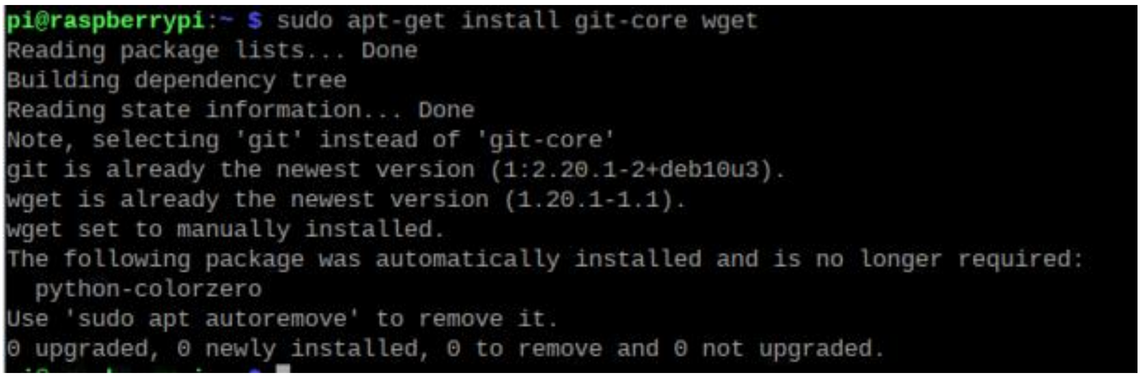
Install wiring Pi

- build essential package `apt-get install build-essential`



```
pi@raspberrypi:~ $ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.6).
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $
```

- other tools packages `apt-get install git-core wget`



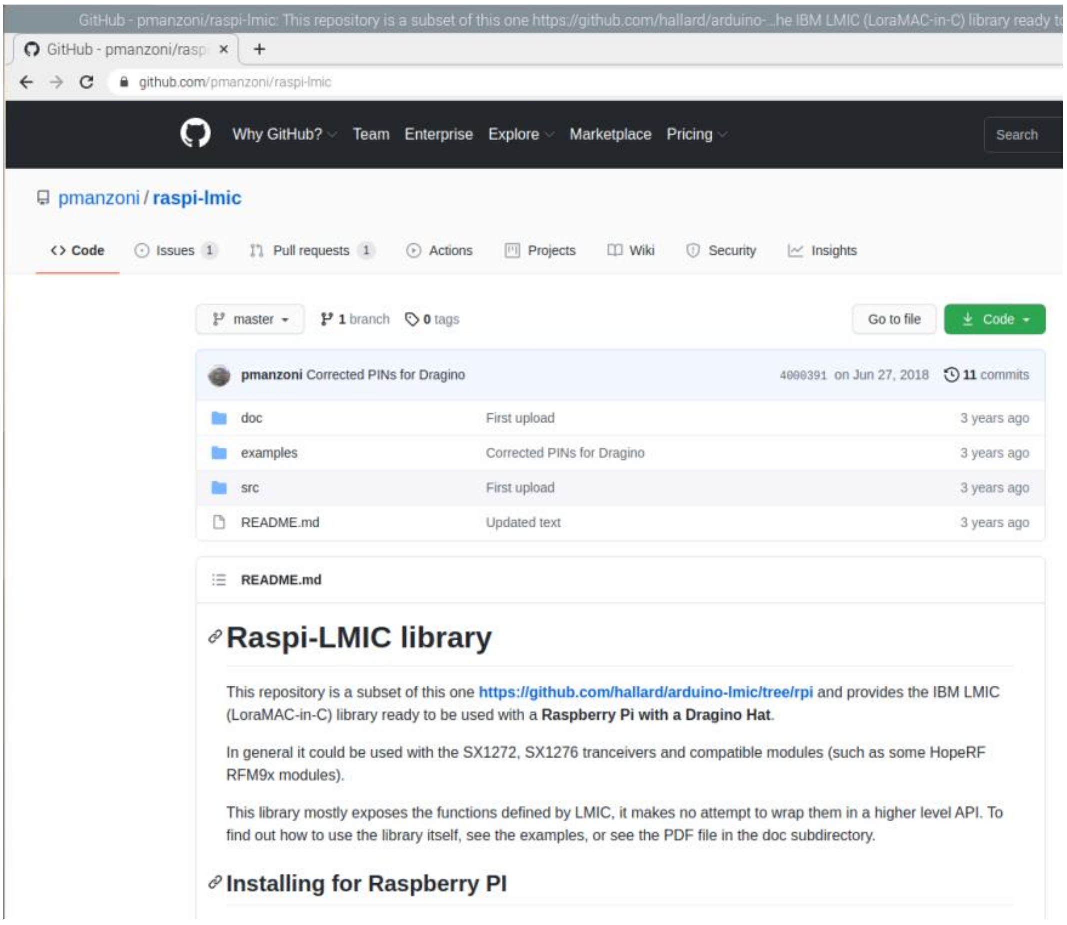
```
pi@raspberrypi:~ $ sudo apt-get install git-core wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
git is already the newest version (1:2.20.1-2+deb10u3).
wget is already the newest version (1.20.1-1.1).
wget set to manually installed.
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Install bcm2835 library

We install the software on the Raspberry PI

We open the following address on an explorer on our Pi

https://github.com/pmanzoni/raspi-lmic



pmanzoni Corrected PINs for Dragino 4000391 on Jun 27, 2018 11 commits

doc	First upload	3 years ago
examples	Corrected PINs for Dragino	3 years ago
src	First upload	3 years ago
README.md	Updated text	3 years ago

README.md

Raspi-LMIC library

This repository is a subset of this one https://github.com/hallard/arduino-lmic/tree/rpi and provides the IBM LMIC (LoraMAC-in-C) library ready to be used with a **Raspberry Pi with a Dragino Hat**.

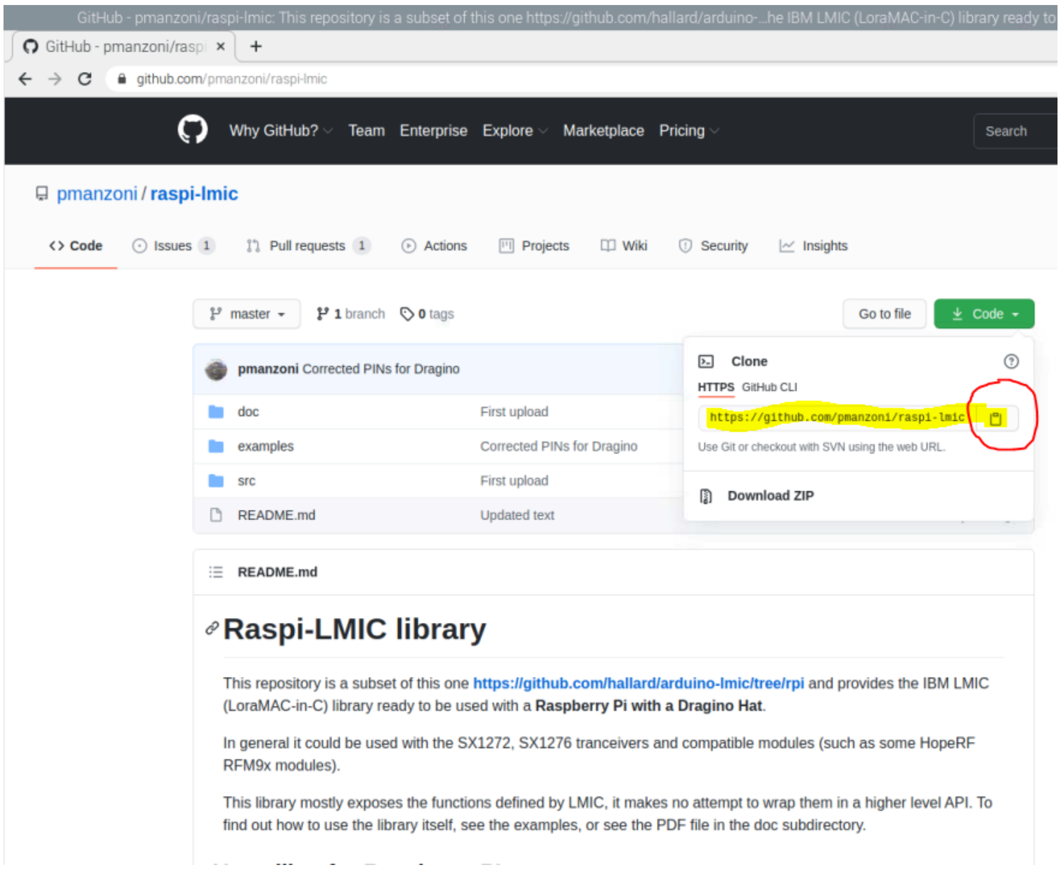
In general it could be used with the SX1272, SX1276 tranceivers and compatible modules (such as some HopeRF RFM9x modules).

This library mostly exposes the functions defined by LMIC, it makes no attempt to wrap them in a higher level API. To find out how to use the library itself, see the examples, or see the PDF file in the doc subdirectory.

Installing for Raspberry PI

Click on the green box called code

Click on the yellow tab on the right to Copy the URL on the clipboard


GitHub - pmanzoni/raspi-lmic: This repository is a subset of this one https://github.com/hallard/arduino-...he IBM LMIC (LoraMAC-in-C) library ready to

github.com/pmanzoni/raspi-lmic

https://github.com/pmanzoni/raspi-lmic

Use Git or checkout with SVN using the web URL.

pmanzoni Corrected PINs for Dragino	
doc	First upload
examples	Corrected PINs for Dragino
src	First upload
README.md	Updated text

README.md

Raspi-LMIC library

This repository is a subset of this one **https://github.com/hallard/arduino-lmic/tree/rpi** and provides the IBM LMIC (LoraMAC-in-C) library ready to be used with a **Raspberry Pi with a Dragino Hat**.

In general it could be used with the SX1272, SX1276 tranceivers and compatible modules (such as some HopeRF RFM9x modules).

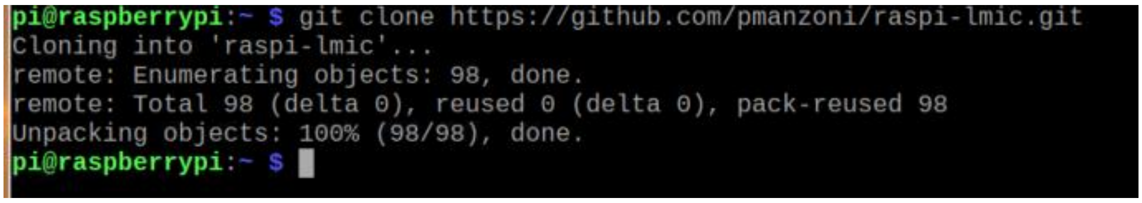
This library mostly exposes the functions defined by LMIC, it makes no attempt to wrap them in a higher level API. To find out how to use the library itself, see the examples, or see the PDF file in the doc subdirectory.

We copy the url

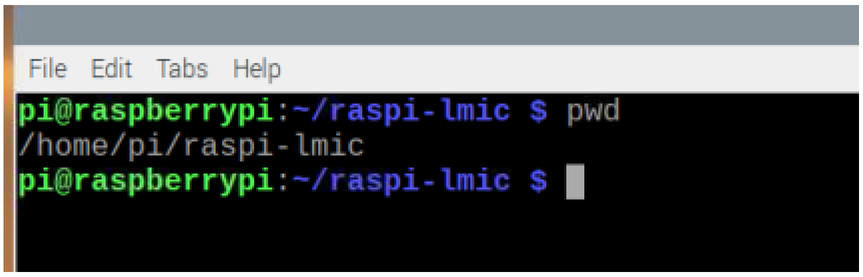
And we use this command

`git clone + the copied url`

`git clone` https://github.com/pmanzoni/raspi-lmic.git

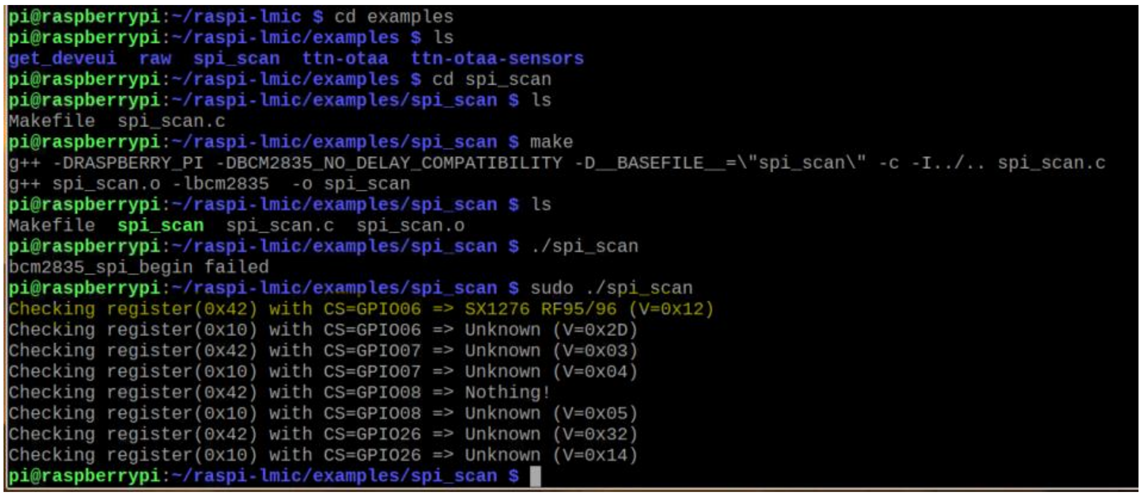

```
pi@raspberrypi:~ $ git clone https://github.com/pmanzoni/raspi-lmic.git
Cloning into 'raspi-lmic'...
remote: Enumerating objects: 98, done.
remote: Total 98 (delta 0), reused 0 (delta 0), pack-reused 98
Unpacking objects: 100% (98/98), done.
pi@raspberrypi:~ $
```

The files are there



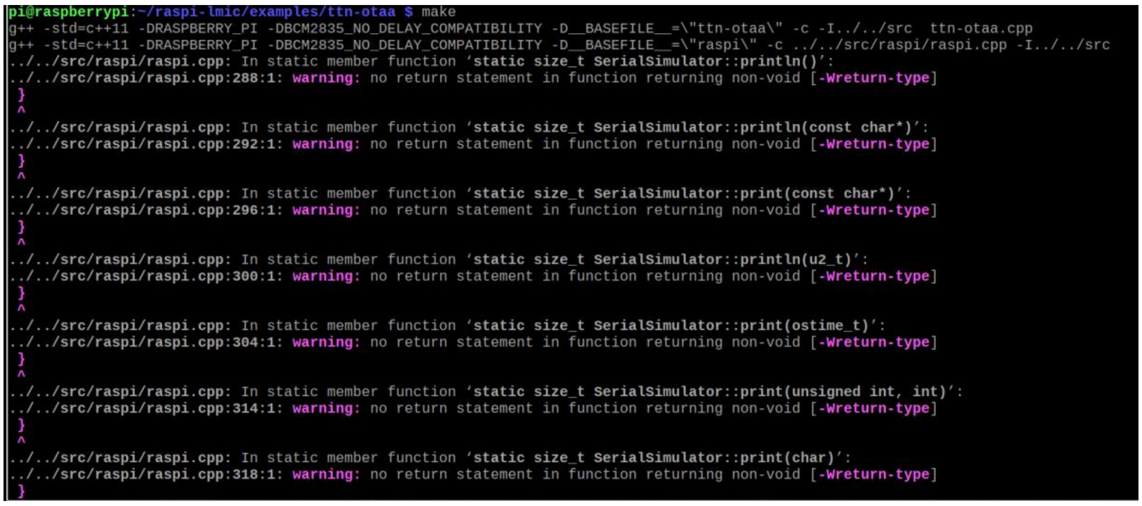
```
pi@raspberrypi:~/raspi-lmic $ pwd
/home/pi/raspi-lmic
pi@raspberrypi:~/raspi-lmic $
```

So the RFM95 is detected!!!

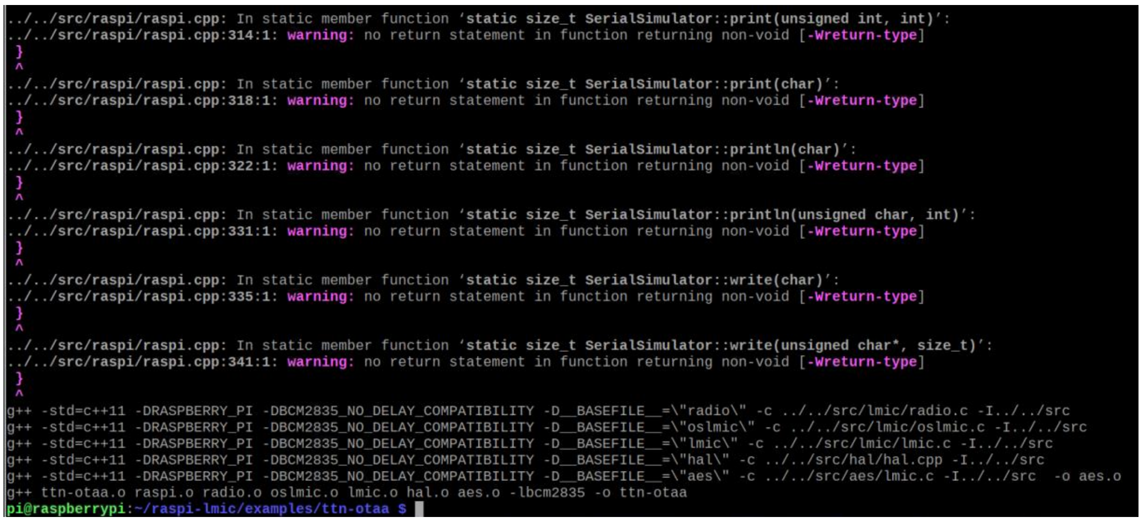


```
pi@raspberrypi:~/raspi-lmic $ cd examples
pi@raspberrypi:~/raspi-lmic/examples $ ls
get_deveui  raw  spi_scan  ttn-otaa  ttn-otaa-sensors
pi@raspberrypi:~/raspi-lmic/examples $ cd spi_scan
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ ls
Makefile  spi_scan.c
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ make
g++ -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"spi_scan\" -c -I../.. spi_scan.c
g++ spi_scan.o -lbcm2835  -o spi_scan
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ ls
Makefile  spi_scan  spi_scan.c  spi_scan.o
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ ./spi_scan
bcm2835_spi_begin failed
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ sudo ./spi_scan
Checking register(0x42) with CS=GPIO06 => SX1276 RF95/96 (V=0x12)
Checking register(0x10) with CS=GPIO06 => Unknown (V=0x2D)
Checking register(0x42) with CS=GPIO07 => Unknown (V=0x03)
Checking register(0x10) with CS=GPIO07 => Unknown (V=0x04)
Checking register(0x42) with CS=GPIO08 => Nothing!
Checking register(0x10) with CS=GPIO08 => Unknown (V=0x05)
Checking register(0x42) with CS=GPIO26 => Unknown (V=0x32)
Checking register(0x10) with CS=GPIO26 => Unknown (V=0x14)
pi@raspberrypi:~/raspi-lmic/examples/spi_scan $ 
```

Now we compile the example file ttn-otaa.cpp



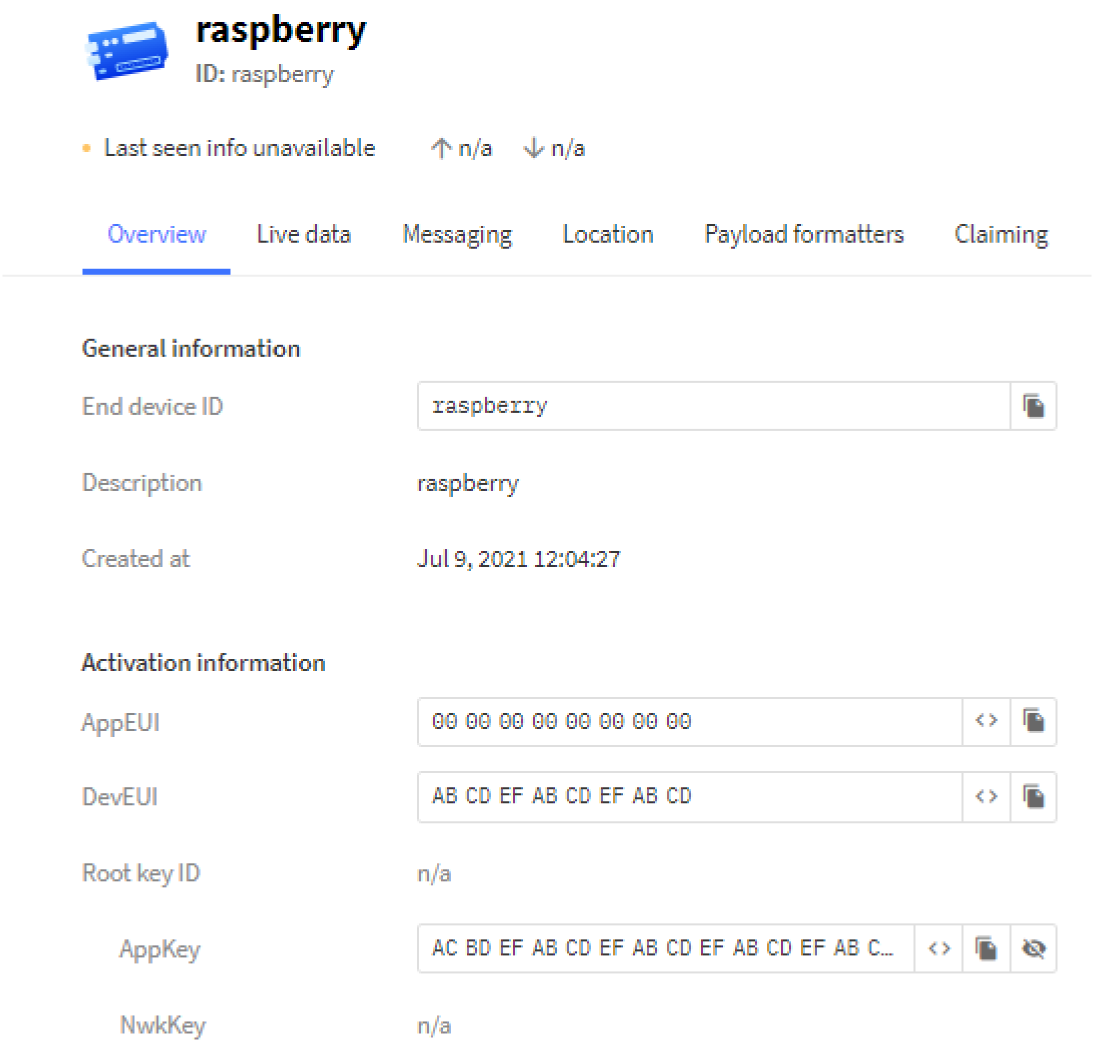
```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ make
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"ttn-otaa\" -c -I../../src  ttn-otaa.cpp
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"raspi\" -c ../../src/raspi/raspi.cpp -I../../src
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::println()’:
../../src/raspi/raspi.cpp:288:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::println(const char*)’:
../../src/raspi/raspi.cpp:292:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(const char*)’:
../../src/raspi/raspi.cpp:296:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::println(u2_t)’:
../../src/raspi/raspi.cpp:300:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(ostime_t)’:
../../src/raspi/raspi.cpp:304:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(unsigned int, int)’:
../../src/raspi/raspi.cpp:314:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(char)’:
../../src/raspi/raspi.cpp:318:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
```



```
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(unsigned int, int)’:
../../src/raspi/raspi.cpp:314:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::print(char)’:
../../src/raspi/raspi.cpp:318:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::println(char)’:
../../src/raspi/raspi.cpp:322:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::println(unsigned char, int)’:
../../src/raspi/raspi.cpp:331:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::write(char)’:
../../src/raspi/raspi.cpp:335:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
../../src/raspi/raspi.cpp: In static member function ‘static size_t SerialSimulator::write(unsigned char*, size_t)’:
../../src/raspi/raspi.cpp:341:1: warning: no return statement in function returning non-void [-Wreturn-type]
 }
 ^
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"radio\" -c ../../src/lmic/radio.c -I../../src
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"oslmic\" -c ../../src/lmic/oslmic.c -I../../src
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"lmic\" -c ../../src/lmic/lmic.c -I../../src
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"hal\" -c ../../src/hal/hal.cpp -I../../src
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"aes\" -c ../../src/aes/lmic.c -I../../src  -o aes.o
g++ ttn-otaa.o raspi.o radio.o oslmic.o lmic.o hal.o aes.o -lbcm2835 -o ttn-otaa
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ 
```

Let’s create a new application on TTN

Manually


raspberry

ID: raspberry

Last seen info unavailable ↑ n/a ↓ n/a

General information

End device ID	raspberry
Description	raspberry
Created at	Jul 9, 2021 12:04:27

Activation information

AppEUI	00 00 00 00 00 00 00 00
DevEUI	AB CD EF AB CD EF AB CD
Root key ID	n/a
AppKey	AC BD EF AB CD EF AB CD EF AB CD EF AB C...
NwkKey	n/a

Let’s modify the credentials accordingly on ttn-otaa.cpp file

`raspi-lmic/examples/ttn-otaa/` : using a Raspi as a TTN node:

- lines to be modified:

  i. `static const u1_t PROGMEM APPEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };`

     This EUI must be in **little-endian format**, ... For TTN issued EUIs the last bytes should be 0xD5, 0xB3,0x70.

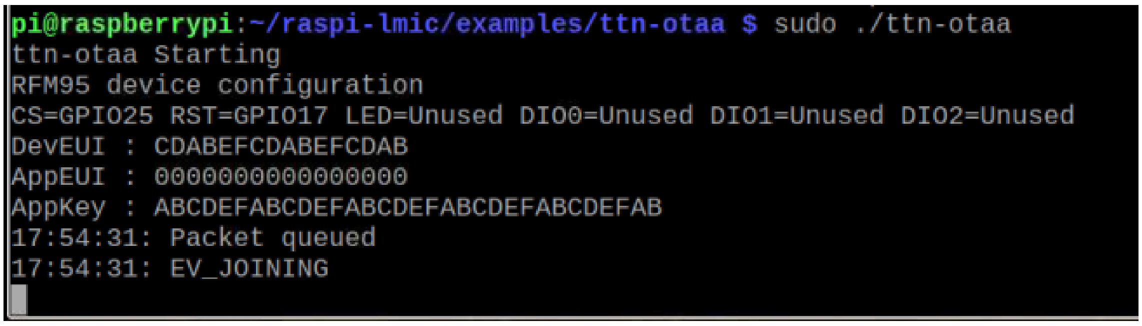
  ii. `static const u1_t PROGMEM DEVEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };`

     This EUI must be in **little-endian format**

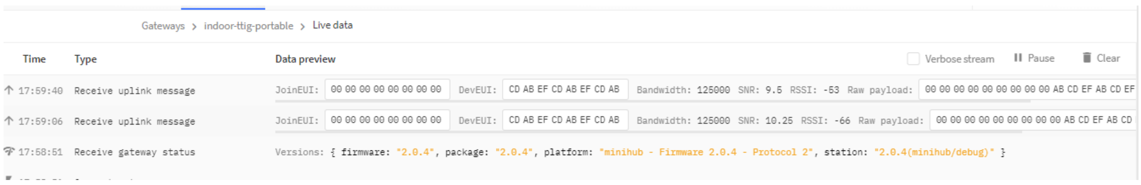
  iii. `static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };`

     This key should be in big endian format

- run using `sudo`

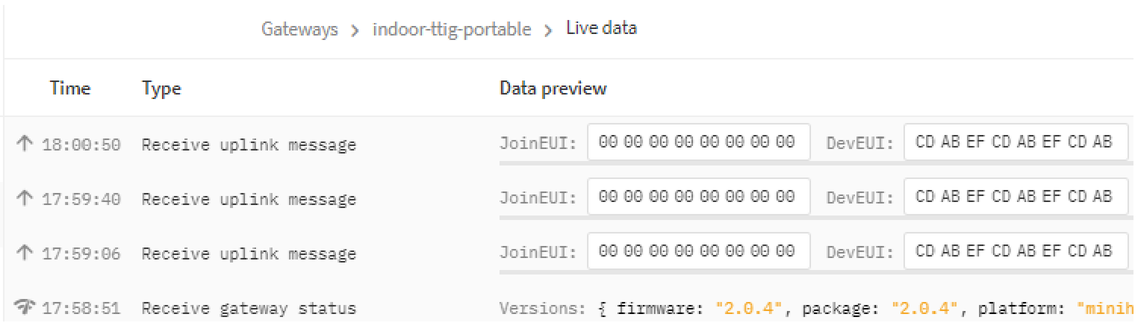


```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : CDABEFCDABEFCDAB
AppEUI : 0000000000000000
AppKey : ABCDEFABCDEFABCDEFABCDEFABCDEFAB
17:54:31: Packet queued
17:54:31: EV_JOINING
```



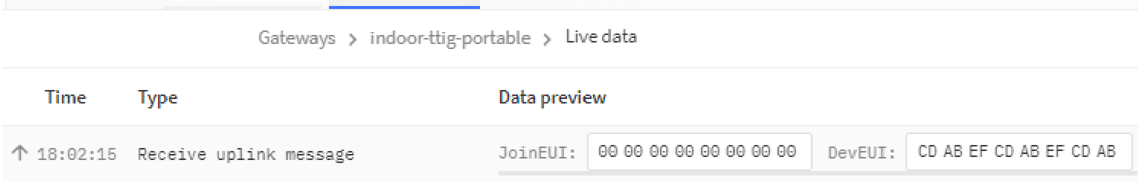
Gateways › indoor-ttig-portable › Live data

```
Time      Type                     Data preview                                                                                                   Verbose stream  Pause  Clear
17:59:40  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB  Bandwidth: 125000 SNR: 9.5 RSSI: -53 Raw payload: 00 00 00 00 00 00 00 00 00 AB CD EF AB CD EF
17:59:06  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB  Bandwidth: 125000 SNR: 10.25 RSSI: -66 Raw payload: 00 00 00 00 00 00 00 00 00 AB CD EF AB CD
17:58:51  Receive gateway status   Versions: { firmware: "2.0.4", package: "2.0.4", platform: "minihub - Firmware 2.0.4 - Protocol 2", station: "2.0.4(minihub/debug)" }
```



Gateways › indoor-ttig-portable › Live data

```
Time      Type                     Data preview
18:00:50  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB
17:59:40  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB
17:59:06  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB
17:58:51  Receive gateway status   Versions: { firmware: "2.0.4", package: "2.0.4", platform: "minih
```

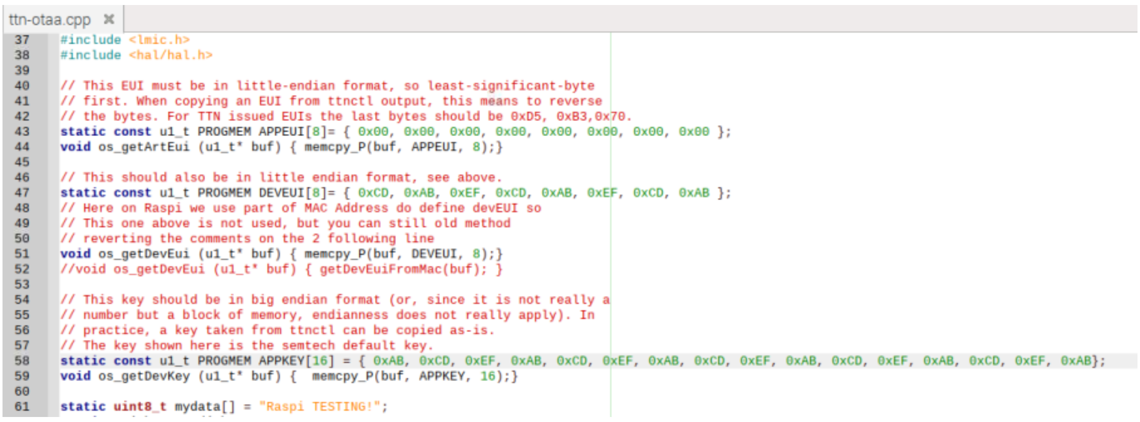


Gateways › indoor-ttig-portable › Live data

```
Time      Type                     Data preview
18:02:15  Receive uplink message   JoinEUI: 00 00 00 00 00 00 00 00  DevEUI: CD AB EF CD AB EF CD AB
```

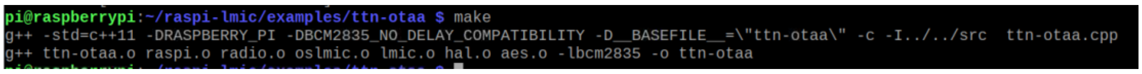
We see that the DevEUI is upside down

Let’s change this

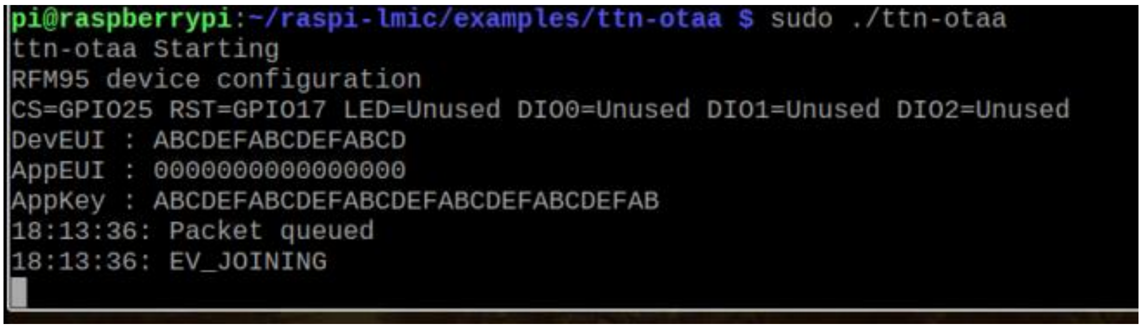


ttn-otaa.cpp

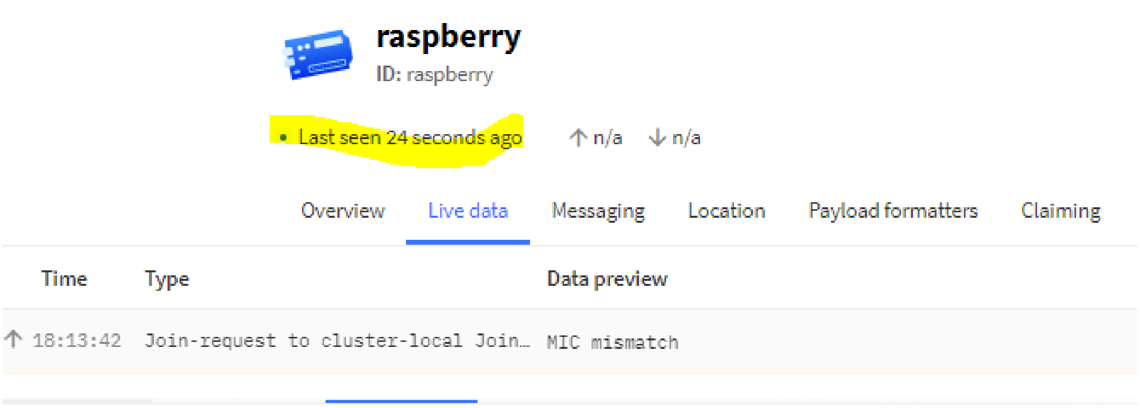
```cpp
#include <lmic.h>
#include <hal/hal.h>

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,0x70.
static const u1_t PROGMEM APPEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]= { 0xCD, 0xAB, 0xEF, 0xCD, 0xAB, 0xEF, 0xCD, 0xAB };
// Here on Raspi we use part of MAC Address do define devEUI so
// This one above is not used, but you can still old method
// reverting the comments on the 2 following line
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
//void os_getDevEui (u1_t* buf) { getDevEuiFromMac(buf); }

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] = { 0xAB, 0xCD, 0xEF, 0xAB, 0xCD, 0xEF, 0xAB, 0xCD, 0xEF, 0xAB, 0xCD, 0xEF, 0xAB, 0xCD, 0xEF, 0xAB};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static uint8_t mydata[] = "Raspi TESTING!";
```



```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ make
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"ttn-otaa\" -c -I../../src  ttn-otaa.cpp
g++ ttn-otaa.o raspi.o radio.o oslmic.o lmic.o hal.o aes.o -lbcm2835 -o ttn-otaa
```



```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : ABCDEFABCDEFABCD
AppEUI : 0000000000000000
AppKey : ABCDEFABCDEFABCDEFABCDEFABCDEFAB
18:13:36: Packet queued
18:13:36: EV_JOINING
```

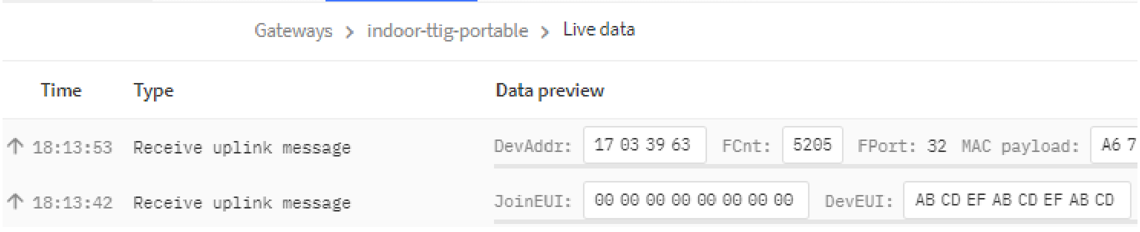


raspberry

ID: raspberry

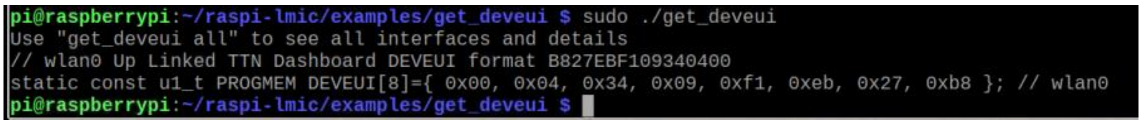
• Last seen 24 seconds ago ↑ n/a ↓ n/a

Time	Type	Data preview
↑ 18:13:42	Join-request to cluster-local Join…	MIC mismatch



Time	Type	Data preview
↑ 18:13:53	Receive uplink message	DevAddr: 17 03 39 63 FCnt: 5205 FPort: 32 MAC payload: A6 7
↑ 18:13:42	Receive uplink message	JoinEUI: 00 00 00 00 00 00 00 00 DevEUI: AB CD EF AB CD EF AB CD

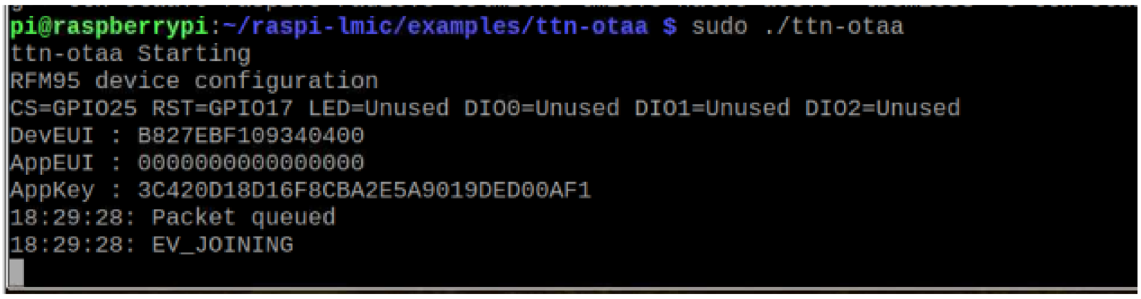
Let’s try with this deveui



```
pi@raspberrypi:~/raspi-lmic/examples/get_deveui $ sudo ./get_deveui
Use "get_deveui all" to see all interfaces and details
// wlan0 Up Linked TTN Dashboard DEVEUI format B827EBF109340400
static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x04, 0x34, 0x09, 0xf1, 0xeb, 0x27, 0xb8 }; // wlan0
pi@raspberrypi:~/raspi-lmic/examples/get_deveui $
```

{ 0x00, 0x04, 0x34, 0x09, 0xf1, 0xeb, 0x27, 0xb8 }

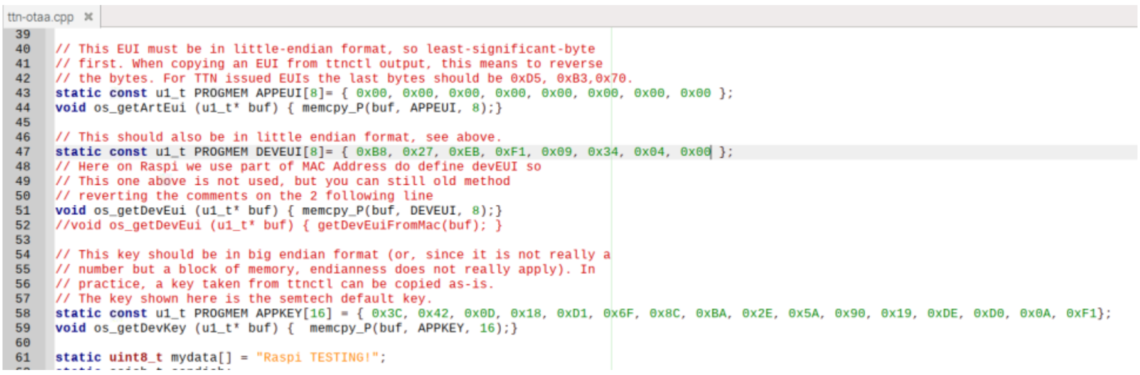
And API Key generated by TTS



```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : B827EBF109340400
AppEUI : 0000000000000000
AppKey : 3C420D18D16F8CBA2E5A9019DED00AF1
18:29:28: Packet queued
18:29:28: EV_JOINING
```

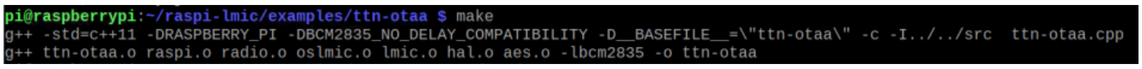
Again it is upside down

Let’s change again on the Raspberry

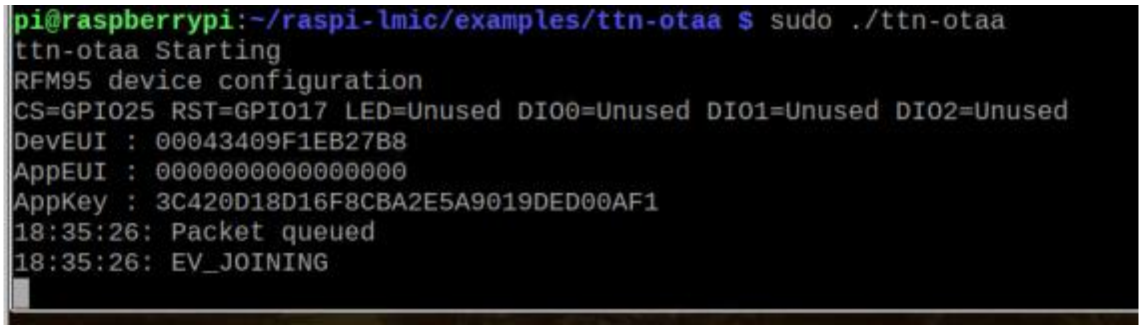


ttn-otaa.cpp

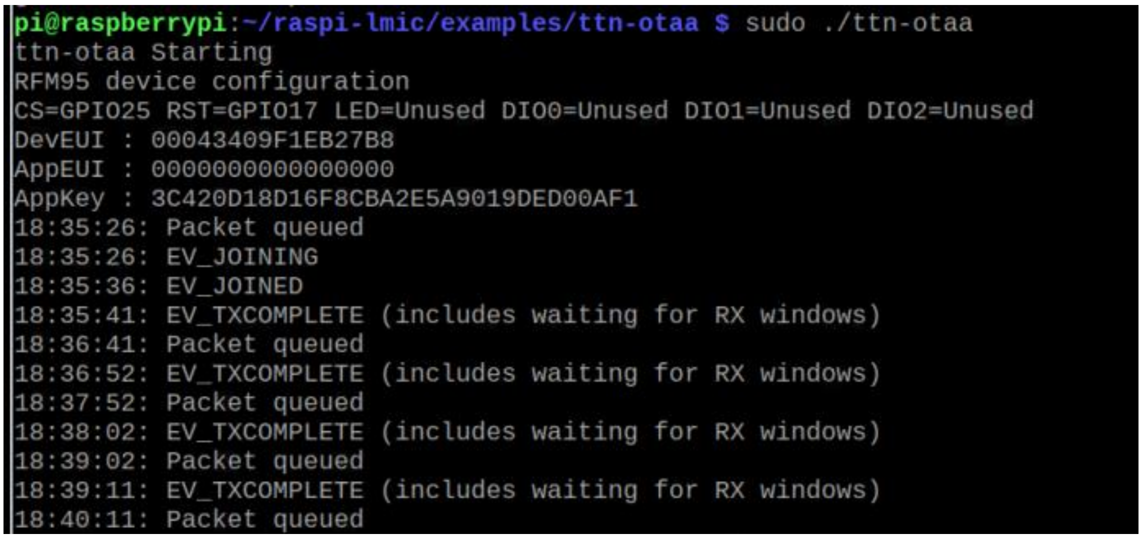
```
// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,0x70.
static const u1_t PROGMEM APPEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]= { 0xB8, 0x27, 0xEB, 0xF1, 0x09, 0x34, 0x04, 0x00 };
// Here on Raspi we use part of MAC Address do define devEUI so
// This one above is not used, but you can still old method
// reverting the comments on the 2 following line
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
//void os_getDevEui (u1_t* buf) { getDevEuiFromMac(buf); }

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] = { 0x3C, 0x42, 0x0D, 0x18, 0xD1, 0x6F, 0x8C, 0xBA, 0x2E, 0x5A, 0x90, 0x19, 0xDE, 0xD0, 0x0A, 0xF1};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static uint8_t mydata[] = "Raspi TESTING!";
```



```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ make
g++ -std=c++11 -DRASPBERRY_PI -DBCM2835_NO_DELAY_COMPATIBILITY -D__BASEFILE__=\"ttn-otaa\" -c -I../../src  ttn-otaa.cpp
g++ ttn-otaa.o raspi.o radio.o oslmic.o lmic.o hal.o aes.o -lbcm2835 -o ttn-otaa
```

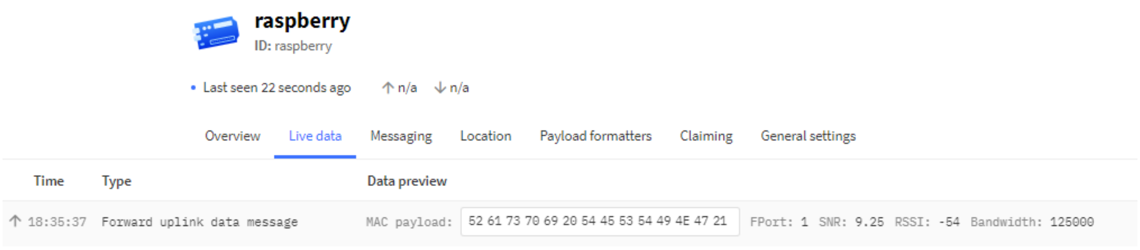


```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : 00043409F1EB27B8
AppEUI : 0000000000000000
AppKey : 3C420D18D16F8CBA2E5A9019DED00AF1
18:35:26: Packet queued
18:35:26: EV_JOINING
```



```
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : 00043409F1EB27B8
AppEUI : 0000000000000000
AppKey : 3C420D18D16F8CBA2E5A9019DED00AF1
18:35:26: Packet queued
18:35:26: EV_JOINING
18:35:36: EV_JOINED
18:35:41: EV_TXCOMPLETE (includes waiting for RX windows)
18:36:41: Packet queued
18:36:52: EV_TXCOMPLETE (includes waiting for RX windows)
18:37:52: Packet queued
18:38:02: EV_TXCOMPLETE (includes waiting for RX windows)
18:39:02: Packet queued
18:39:11: EV_TXCOMPLETE (includes waiting for RX windows)
18:40:11: Packet queued
```

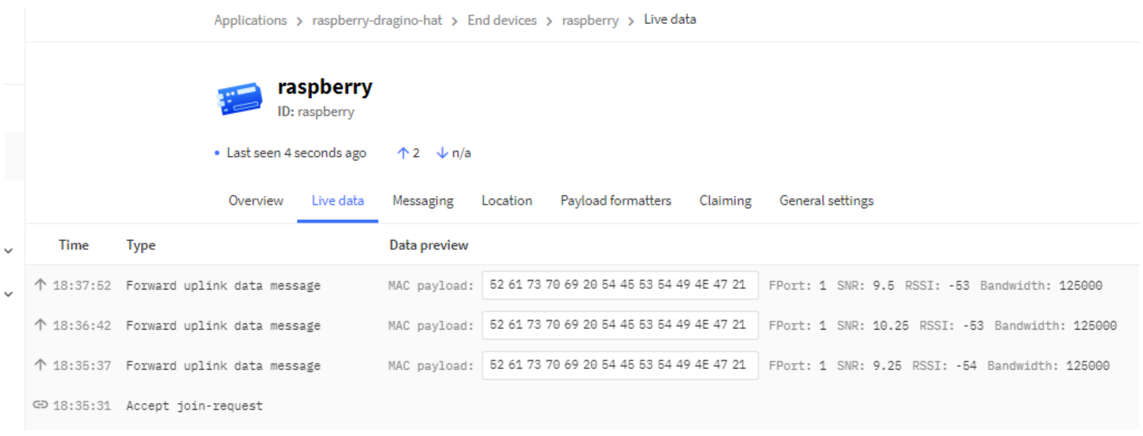
It Works!!



raspberry
ID: raspberry

• Last seen 22 seconds ago ↑ n/a ↓ n/a

Time	Type	Data preview
↑ 18:35:37	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.25 RSSI: -54 Bandwidth: 125000



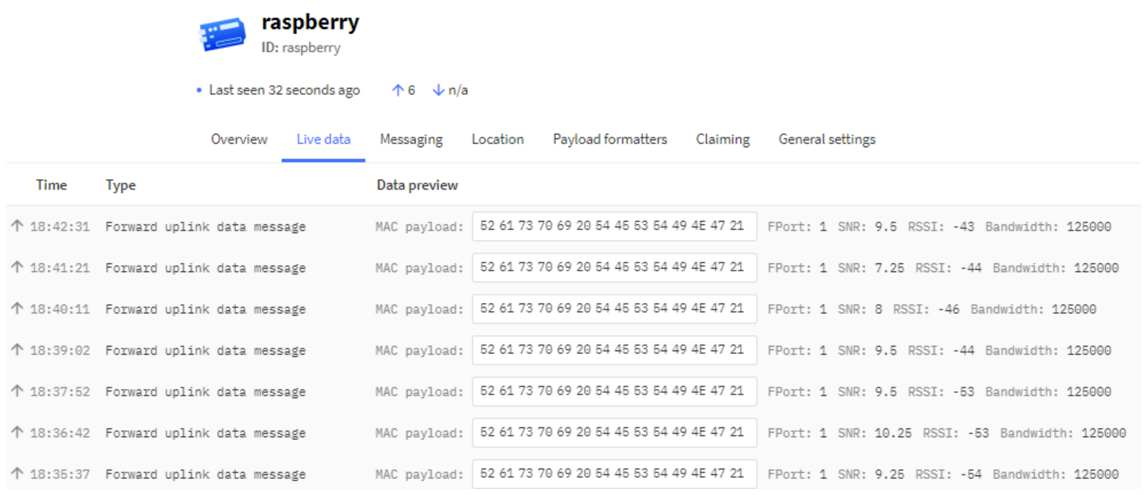
Applications > raspberry-dragino-hat > End devices > raspberry > Live data

raspberry
ID: raspberry

• Last seen 4 seconds ago ↑ 2 ↓ n/a

Overview | Live data | Messaging | Location | Payload formatters | Claiming | General settings

Time	Type	Data preview
↑ 18:37:52	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.5 RSSI: -53 Bandwidth: 125000
↑ 18:36:42	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 10.25 RSSI: -53 Bandwidth: 125000
↑ 18:35:37	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.25 RSSI: -54 Bandwidth: 125000
18:35:31	Accept join-request	

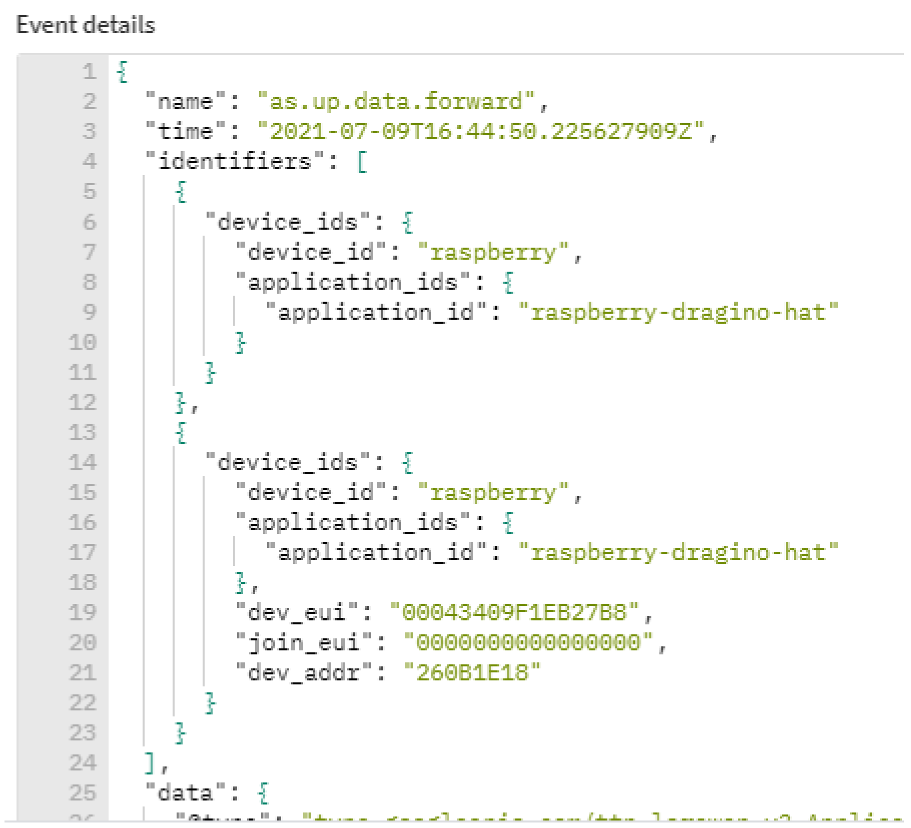


raspberry
ID: raspberry

• Last seen 32 seconds ago ↑ 6 ↓ n/a

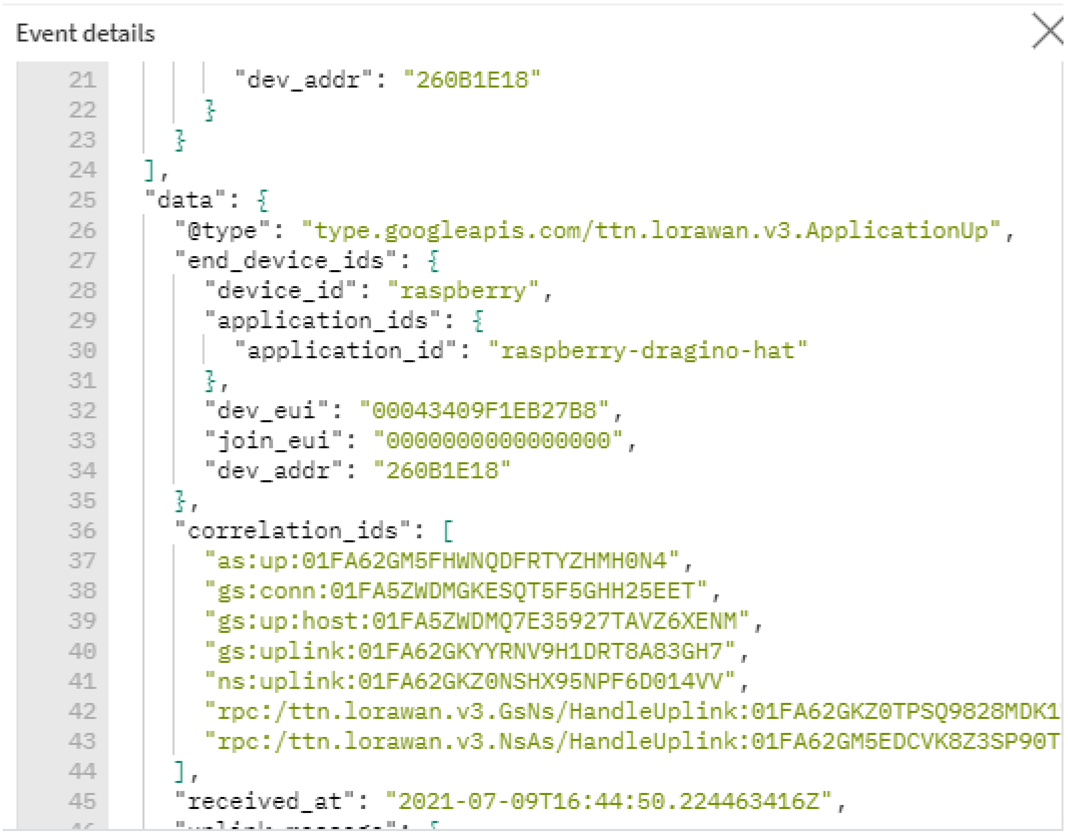
Overview | Live data | Messaging | Location | Payload formatters | Claiming | General settings

Time	Type	Data preview
↑ 18:42:31	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.5 RSSI: -43 Bandwidth: 125000
↑ 18:41:21	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 7.25 RSSI: -44 Bandwidth: 125000
↑ 18:40:11	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 8 RSSI: -46 Bandwidth: 125000
↑ 18:39:02	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.5 RSSI: -44 Bandwidth: 125000
↑ 18:37:52	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.5 RSSI: -53 Bandwidth: 125000
↑ 18:36:42	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 10.25 RSSI: -53 Bandwidth: 125000
↑ 18:35:37	Forward uplink data message	MAC payload: 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 FPort: 1 SNR: 9.25 RSSI: -54 Bandwidth: 125000

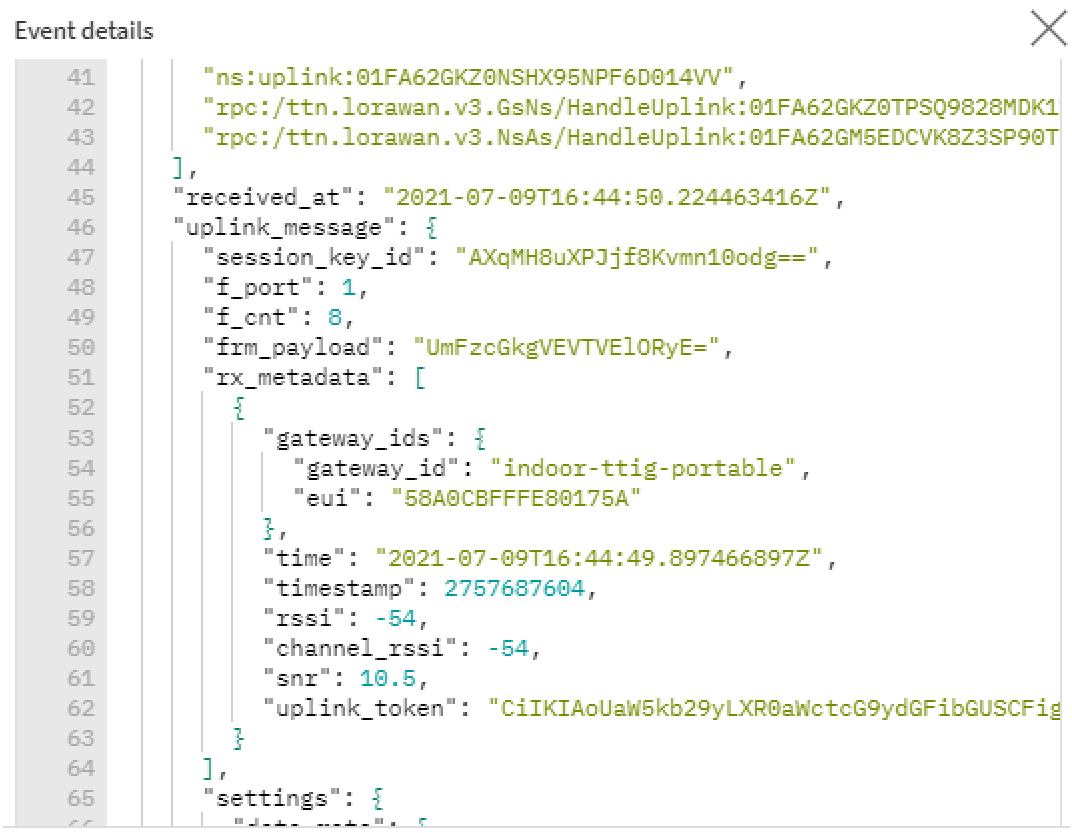


Event details

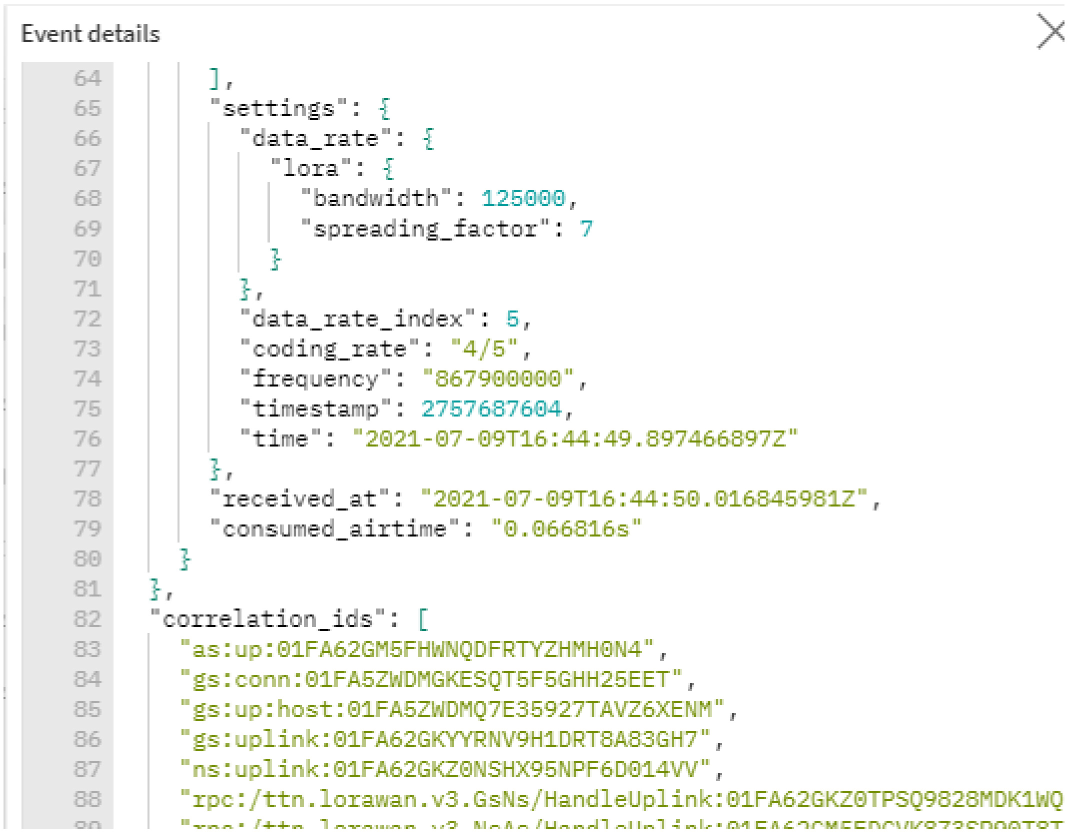
```
{
  "name": "as.up.data.forward",
  "time": "2021-07-09T16:44:50.225627909Z",
  "identifiers": [
    {
      "device_ids": {
        "device_id": "raspberry",
        "application_ids": {
          "application_id": "raspberry-dragino-hat"
        }
      }
    },
    {
      "device_ids": {
        "device_id": "raspberry",
        "application_ids": {
          "application_id": "raspberry-dragino-hat"
        },
        "dev_eui": "00043409F1EB27B8",
        "join_eui": "0000000000000000",
        "dev_addr": "260B1E18"
      }
    }
  ],
  "data": {
```


Event details

```
        "dev_addr": "260B1E18"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.ApplicationUp",
    "end_device_ids": {
      "device_id": "raspberry",
      "application_ids": {
        "application_id": "raspberry-dragino-hat"
      },
      "dev_eui": "00043409F1EB27B8",
      "join_eui": "0000000000000000",
      "dev_addr": "260B1E18"
    },
    "correlation_ids": [
      "as:up:01FA62GM5FHWNQDFRTYZHMH0N4",
      "gs:conn:01FA5ZWDMGKESQT5F5GHH25EET",
      "gs:up:host:01FA5ZWDMQ7E35927TAVZ6XENM",
      "gs:uplink:01FA62GKYYRNV9H1DRT8A83GH7",
      "ns:uplink:01FA62GKZ0NSHX95NPF6D014VV",
      "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FA62GKZ0TPSQ9828MDK1
      "rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FA62GM5EDCVK8Z3SP90T
    ],
    "received_at": "2021-07-09T16:44:50.224463416Z",
```

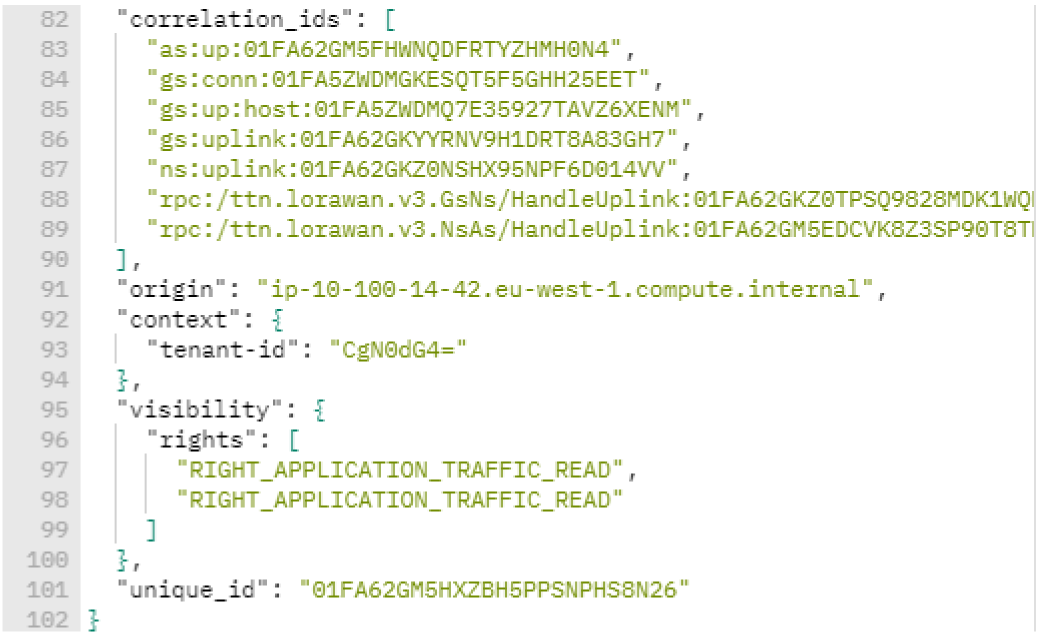

Event details

```
      "ns:uplink:01FA62GKZ0NSHX95NPF6D014VV",
      "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FA62GKZ0TPSQ9828MDK1
      "rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FA62GM5EDCVK8Z3SP90T
    ],
    "received_at": "2021-07-09T16:44:50.224463416Z",
    "uplink_message": {
      "session_key_id": "AXqMH8uXPJjf8Kvmn10odg==",
      "f_port": 1,
      "f_cnt": 8,
      "frm_payload": "UmFzcGkgVEVTVElORyE=",
      "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "indoor-ttig-portable",
            "eui": "58A0CBFFFE80175A"
          },
          "time": "2021-07-09T16:44:49.897466897Z",
          "timestamp": 2757687604,
          "rssi": -54,
          "channel_rssi": -54,
          "snr": 10.5,
          "uplink_token": "CiIKIAoUaW5kb29yLXR0aWctcG9ydGFibGUSCFig
        }
      ],
      "settings": {
```



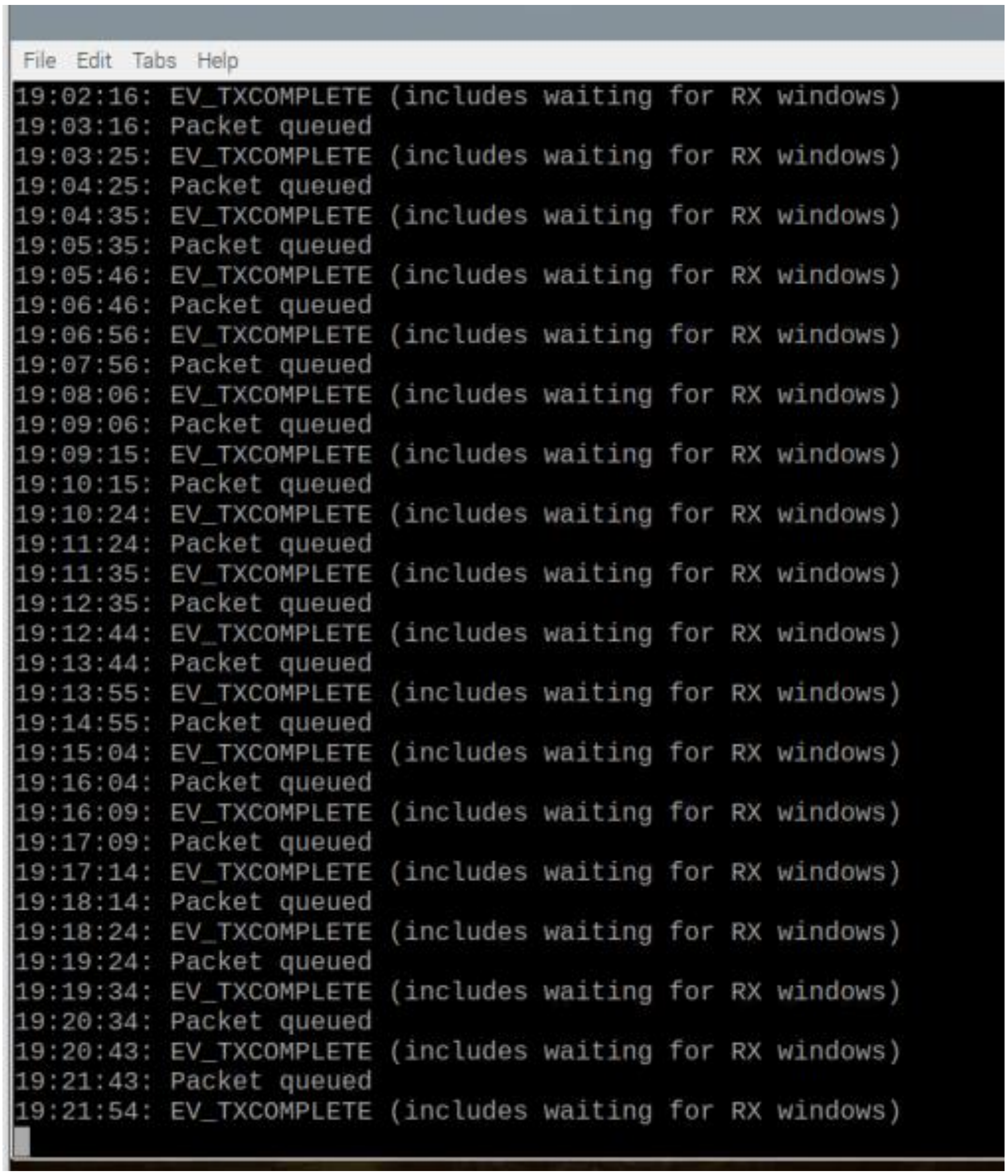
Event details

```
      ],
      "settings": {
        "data_rate": {
          "lora": {
            "bandwidth": 125000,
            "spreading_factor": 7
          }
        },
        "data_rate_index": 5,
        "coding_rate": "4/5",
        "frequency": "867900000",
        "timestamp": 2757687604,
        "time": "2021-07-09T16:44:49.897466897Z"
      },
      "received_at": "2021-07-09T16:44:50.016845981Z",
      "consumed_airtime": "0.066816s"
    }
  },
  "correlation_ids": [
    "as:up:01FA62GM5FHWNQDFRTYZHMH0N4",
    "gs:conn:01FA5ZWDMGKESQT5F5GHH25EET",
    "gs:up:host:01FA5ZWDMQ7E35927TAVZ6XENM",
    "gs:uplink:01FA62GKYYRNV9H1DRT8A83GH7",
    "ns:uplink:01FA62GKZ0NSHX95NPF6D014VV",
    "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FA62GKZ0TPSQ9828MDK1WQ
```



```
  "correlation_ids": [
    "as:up:01FA62GM5FHWNQDFRTYZHMH0N4",
    "gs:conn:01FA5ZWDMGKESQT5F5GHH25EET",
    "gs:up:host:01FA5ZWDMQ7E35927TAVZ6XENM",
    "gs:uplink:01FA62GKYYRNV9H1DRT8A83GH7",
    "ns:uplink:01FA62GKZ0NSHX95NPF6D014VV",
    "rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FA62GKZ0TPSQ9828MDK1WQ
```

```
    "rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FA62GM5EDCVK8Z3SP90T8T
  ],
  "origin": "ip-10-100-14-42.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_APPLICATION_TRAFFIC_READ",
      "RIGHT_APPLICATION_TRAFFIC_READ"
    ]
  },
  "unique_id": "01FA62GM5HXZBH5PPSNPHS8N26"
}
```



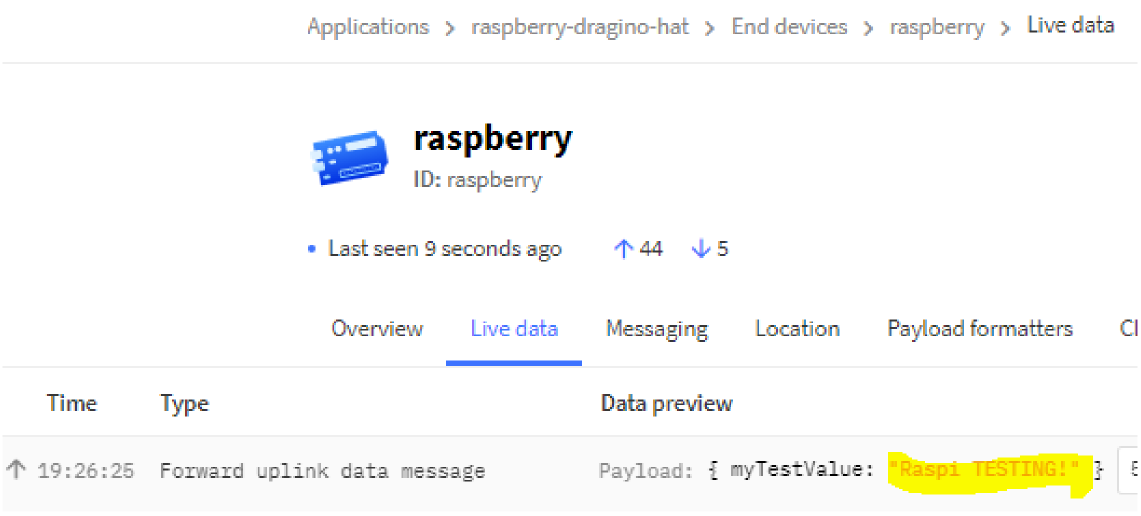
```
19:02:16: EV_TXCOMPLETE (includes waiting for RX windows)
19:03:16: Packet queued
19:03:25: EV_TXCOMPLETE (includes waiting for RX windows)
19:04:25: Packet queued
19:04:35: EV_TXCOMPLETE (includes waiting for RX windows)
19:05:35: Packet queued
19:05:46: EV_TXCOMPLETE (includes waiting for RX windows)
19:06:46: Packet queued
19:06:56: EV_TXCOMPLETE (includes waiting for RX windows)
19:07:56: Packet queued
19:08:06: EV_TXCOMPLETE (includes waiting for RX windows)
19:09:06: Packet queued
19:09:15: EV_TXCOMPLETE (includes waiting for RX windows)
19:10:15: Packet queued
19:10:24: EV_TXCOMPLETE (includes waiting for RX windows)
19:11:24: Packet queued
19:11:35: EV_TXCOMPLETE (includes waiting for RX windows)
19:12:35: Packet queued
19:12:44: EV_TXCOMPLETE (includes waiting for RX windows)
19:13:44: Packet queued
19:13:55: EV_TXCOMPLETE (includes waiting for RX windows)
19:14:55: Packet queued
19:15:04: EV_TXCOMPLETE (includes waiting for RX windows)
19:16:04: Packet queued
19:16:09: EV_TXCOMPLETE (includes waiting for RX windows)
19:17:09: Packet queued
19:17:14: EV_TXCOMPLETE (includes waiting for RX windows)
19:18:14: Packet queued
19:18:24: EV_TXCOMPLETE (includes waiting for RX windows)
19:19:24: Packet queued
19:19:34: EV_TXCOMPLETE (includes waiting for RX windows)
19:20:34: Packet queued
19:20:43: EV_TXCOMPLETE (includes waiting for RX windows)
19:21:43: Packet queued
19:21:54: EV_TXCOMPLETE (includes waiting for RX windows)
```

Now we try to decode the payload with payload formatter given on the repository

```
function Decoder(bytes, port) {
  // Decode plain text; for testing only
  return {
      myTestValue: String.fromCharCode.apply(null, bytes)
  };
}
```

Payload is

frm_payload": "UmFzcGkgVEVTVElORyE=",


Applications > raspberry-dragino-hat > End devices > raspberry > Live data

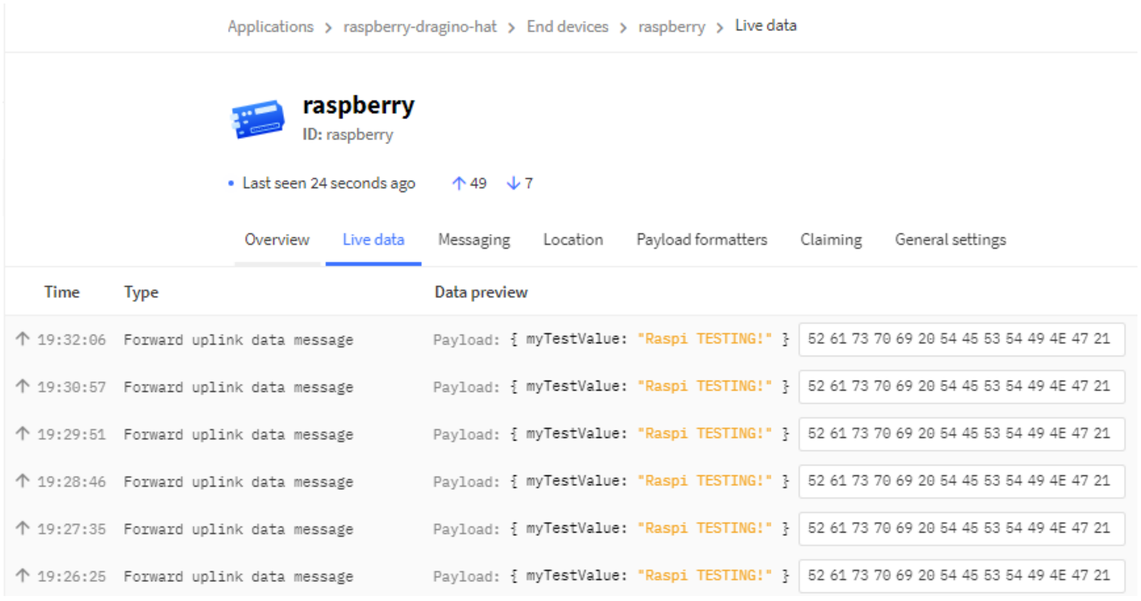
raspberry
ID: raspberry

• Last seen 9 seconds ago ↑ 44 ↓ 5

Overview | Live data | Messaging | Location | Payload formatters | Cl

Time	Type	Data preview
↑ 19:26:25	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" } 5

Voilà


Applications > raspberry-dragino-hat > End devices > raspberry > Live data

raspberry
ID: raspberry

• Last seen 24 seconds ago ↑ 49 ↓ 7

Overview | Live data | Messaging | Location | Payload formatters | Claiming | General settings

Time	Type	Data preview	
↑ 19:32:06	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21
↑ 19:30:57	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21
↑ 19:29:51	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21
↑ 19:28:46	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21
↑ 19:27:35	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21
↑ 19:26:25	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!" }	52 61 73 70 69 20 54 45 53 54 49 4E 47 21

This is the ttn-otaa.cpp used file

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 *
 * Permission is hereby granted, free of charge, to anyone
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.
 *
 * This example sends a valid LoRaWAN packet with payload "Hello,
 * world!", using frequency and encryption settings matching those of
 * the The Things Network.
 *
 * This uses OTAA (Over-the-air activation), where where a DevEUI and
 * application key is configured, which are used in an over-the-air
 * activation procedure where a DevAddr and session keys are
 * assigned/generated for use with all further communication.
 *
 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
 * g1, 0.1% in g2), but not the TTN fair usage policy (which is probably
 * violated by this sketch when left running for longer)!

 * To use this sketch, first register your application and device with
 * the things network, to set or generate an AppEUI, DevEUI and AppKey.
 * Multiple devices can use the same AppEUI, but each device has its own
 * DevEUI and AppKey.
 *
 * Do not forget to define the radio type correctly in config.h.
 *

*****************************************************************************
**/

#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>

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

```
// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,0x70.
static const u1_t PROGMEM APPEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]= { 0xB8, 0x27, 0xEB, 0xF1, 0x09, 0x34, 0x04, 0x00 };
// Here on Raspi we use part of MAC Address do define devEUI so
// This one above is not used, but you can still old method
// reverting the comments on the 2 following line
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
//void os_getDevEui (u1_t* buf) { getDevEuiFromMac(buf); }

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] = { 0x3C, 0x42, 0x0D, 0x18, 0xD1, 0x6F, 0x8C, 0xBA,
0x2E, 0x5A, 0x90, 0x19, 0xDE, 0xD0, 0x0A, 0xF1};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static uint8_t mydata[] = "Raspi TESTING!";
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty)
// cycle limitations).
const unsigned TX_INTERVAL = 60;

//Flag for Ctrl-C
```

```
volatile sig_atomic_t force_exit = 0;

// LoRasPi board
// see https://github.com/hallard/LoRasPI
//#define RF_LED_PIN RPI_V2_GPIO_P1_16 // Led on GPIO23 so P1 connector pin #16
#define RF_CS_PIN  RPI_V2_GPIO_P1_22 // Slave Select on CE0 so P1 connector pin #24
#define RF_IRQ_PIN RPI_V2_GPIO_P1_07 // IRQ on GPIO25 so P1 connector pin #22
#define RF_RST_PIN RPI_V2_GPIO_P1_11 // RST on GPIO22 so P1 connector pin #15

// Raspberri PI Lora Gateway for multiple modules
// see https://github.com/hallard/RPI-Lora-Gateway
// Module 1 on board RFM95 868 MHz (example)
//#define RF_LED_PIN RPI_V2_GPIO_P1_07 // Led on GPIO4 so P1 connector pin #7
//#define RF_CS_PIN  RPI_V2_GPIO_P1_24 // Slave Select on CE0 so P1 connector pin #24
//#define RF_IRQ_PIN RPI_V2_GPIO_P1_22 // IRQ on GPIO25 so P1 connector pin #22
//#define RF_RST_PIN RPI_V2_GPIO_P1_29 // Reset on GPIO5 so P1 connector pin #29



// Dragino Raspberry PI hat (no onboard led)
// see https://github.com/dragino/Lora
#define RF_CS_PIN  RPI_V2_GPIO_P1_22 // Slave Select on GPIO25 so P1 connector pin #22
#define RF_IRQ_PIN RPI_V2_GPIO_P1_07 // IRQ on GPIO4 so P1 connector pin #7
#define RF_RST_PIN RPI_V2_GPIO_P1_11 // Reset on GPIO17 so P1 connector pin #11

// Pin mapping
const lmic_pinmap lmic_pins = {
   .nss  = RF_CS_PIN,
   .rxtx = LMIC_UNUSED_PIN,
   .rst  = RF_RST_PIN,
   .dio  = {LMIC_UNUSED_PIN, LMIC_UNUSED_PIN, LMIC_UNUSED_PIN},
};
```

```
#ifndef RF_LED_PIN
#define RF_LED_PIN NOT_A_PIN
#endif

void do_send(osjob_t* j) {
  char strTime[16];
  getSystemTime(strTime , sizeof(strTime));
  printf("%s: ", strTime);

  // Check if there is not a current TX/RX job running
  if (LMIC.opmode & OP_TXRXPEND) {
    printf("OP_TXRXPEND, not sending\n");
  } else {
    digitalWrite(RF_LED_PIN, HIGH);
    // Prepare upstream data transmission at the next possible time.
    LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
    printf("Packet queued\n");
  }
  // Next TX is scheduled after TX_COMPLETE event.
}

void onEvent (ev_t ev) {
  char strTime[16];
  getSystemTime(strTime , sizeof(strTime));
  printf("%s: ", strTime);

  switch(ev) {
    case EV_SCAN_TIMEOUT:
      printf("EV_SCAN_TIMEOUT\n");
    break;
```

```
case EV_BEACON_FOUND:
   printf("EV_BEACON_FOUND\n");
break;
case EV_BEACON_MISSED:
   printf("EV_BEACON_MISSED\n");
break;
case EV_BEACON_TRACKED:
   printf("EV_BEACON_TRACKED\n");
break;
case EV_JOINING:
   printf("EV_JOINING\n");
break;
case EV_JOINED:
   printf("EV_JOINED\n");
   digitalWrite(RF_LED_PIN, LOW);
   // Disable link check validation (automatically enabled
   // during join, but not supported by TTN at this time).
   LMIC_setLinkCheckMode(0);
break;
case EV_RFU1:
   printf("EV_RFU1\n");
break;
case EV_JOIN_FAILED:
   printf("EV_JOIN_FAILED\n");
break;
case EV_REJOIN_FAILED:
   printf("EV_REJOIN_FAILED\n");
break;
case EV_TXCOMPLETE:
   printf("EV_TXCOMPLETE (includes waiting for RX windows)\n");
   if (LMIC.txrxFlags & TXRX_ACK)
```

```
}

            printf("%s Received ack\n", strTime);
          if (LMIC.dataLen) {
            printf("%s Received %d bytes of payload\n", strTime, LMIC.dataLen);
          }
          digitalWrite(RF_LED_PIN, LOW);
          // Schedule next transmission
          os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
        break;
        case EV_LOST_TSYNC:
          printf("EV_LOST_TSYNC\n");
        break;
        case EV_RESET:
          printf("EV_RESET\n");
        break;
        case EV_RXCOMPLETE:
          // data received in ping slot
          printf("EV_RXCOMPLETE\n");
        break;
        case EV_LINK_DEAD:
          printf("EV_LINK_DEAD\n");
        break;
        case EV_LINK_ALIVE:
          printf("EV_LINK_ALIVE\n");
        break;
        default:
          printf("Unknown event\n");
        break;
    }
}

/* ===================================================================
Function: sig_handler
Purpose : Intercept CTRL-C keyboard to close application
Input   : signal received
Output  : -
Comments: -
====================================================================== */
void sig_handler(int sig)
{
  printf("\nBreak received, exiting!\n");
  force_exit=true;
}

/* ======================================================================
Function: main
Purpose : not sure ;)
Input   : command line parameters
Output  : -
Comments: -
====================================================================== */
int main(void)
{
   // caught CTRL-C to do clean-up
   signal(SIGINT, sig_handler);

   printf("%s Starting\n", __BASEFILE__);

     // Init GPIO bcm
   if (!bcm2835_init()) {
      fprintf( stderr, "bcm2835_init() Failed\n\n" );
      return 1;
   }
```

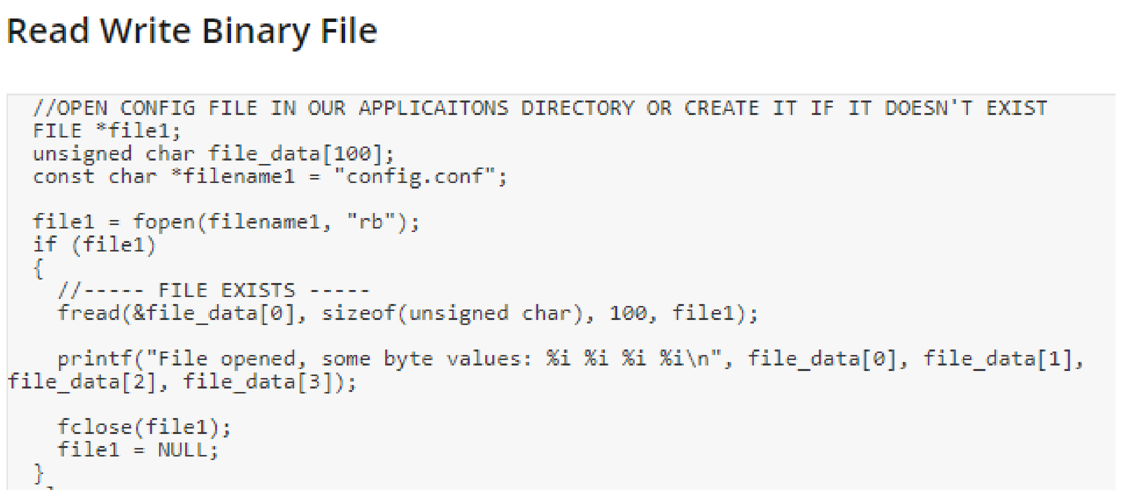
```
        // Show board config
printConfig(RF_LED_PIN);
printKeys();

// Light off on board LED
pinMode(RF_LED_PIN, OUTPUT);
digitalWrite(RF_LED_PIN, HIGH);

// LMIC init
os_init();
// Reset the MAC state. Session and pending data transfers will be discarded.
LMIC_reset();

// Start job (sending automatically starts OTAA too)
do_send(&sendjob);

while(!force_exit) {
  os_runloop_once();

  // We're on a multitasking OS let some time for others
  // Without this one CPU is 99% and with this one just 3%
  // On a Raspberry PI 3
  usleep(1000);
}

// We're here because we need to exit, do it clean

// Light off on board LED
digitalWrite(RF_LED_PIN, LOW);
```

```
    // module CS line High
    digitalWrite(lmic_pins.nss, HIGH);
    printf( "\n%s, done my job!\n", __BASEFILE__ );
    bcm2835_close();
    return 0;
}
```

Now let’s try to modify the programm to get the data with a file from Modbus TCP device

FILE TRANSFER

We will try to use a file like on the example

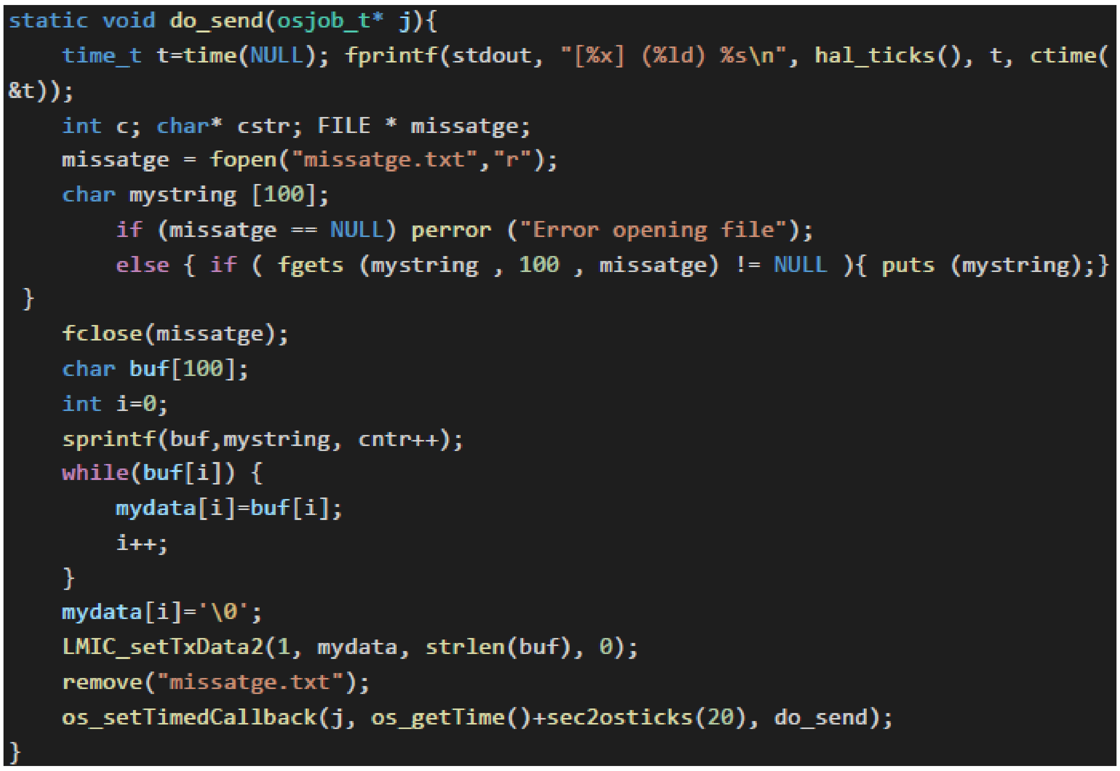


Read Write Binary File

```
 //OPEN CONFIG FILE IN OUR APPLICAITONS DIRECTORY OR CREATE IT IF IT DOESN'T EXIST
 FILE *file1;
 unsigned char file_data[100];
 const char *filename1 = "config.conf";

 file1 = fopen(filename1, "rb");
 if (file1)
 {
   //----- FILE EXISTS -----
   fread(&file_data[0], sizeof(unsigned char), 100, file1);

   printf("File opened, some byte values: %i %i %i %i\n", file_data[0], file_data[1],
file_data[2], file_data[3]);

   fclose(file1);
   file1 = NULL;
 }
```

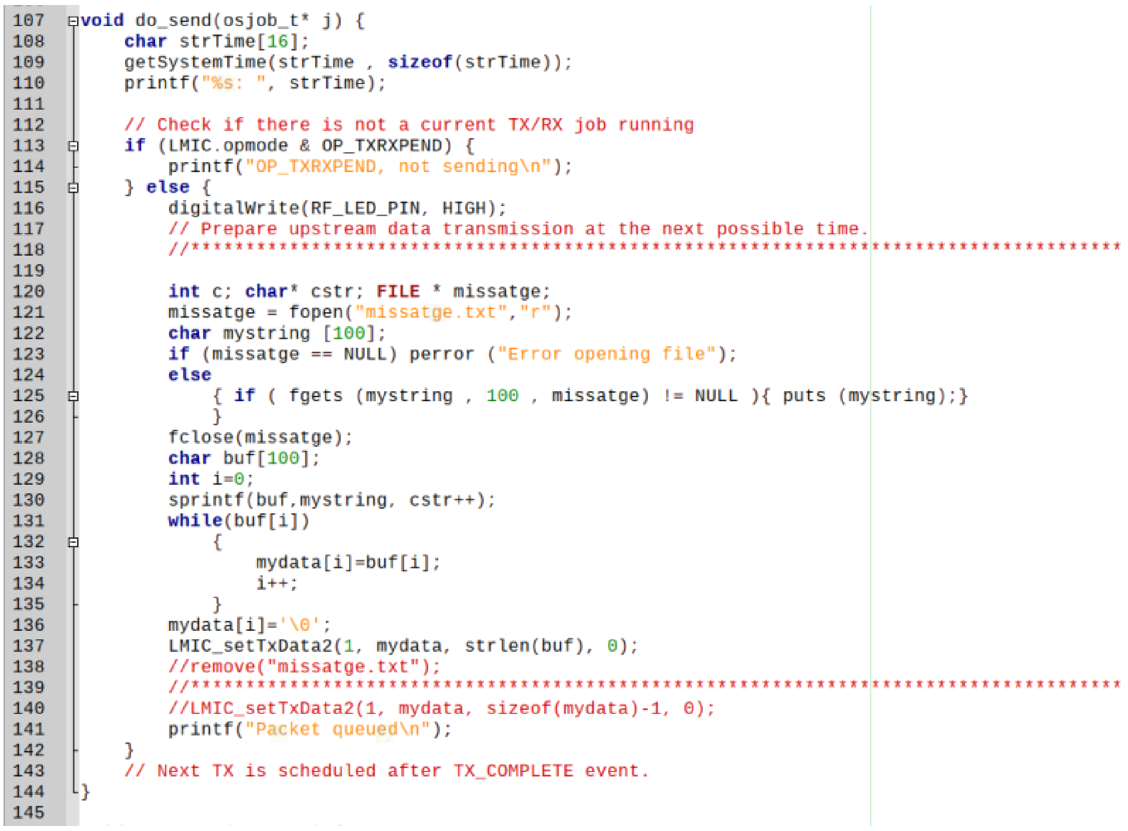
So we modify the ttn-otaa.cpp file like this way

Except the last two lines (we do not erase the file, if you erase the file the program execution breaks)



```
static void do_send(osjob_t* j){
    time_t t=time(NULL); fprintf(stdout, "[%x] (%ld) %s\n", hal_ticks(), t, ctime(
&t));
    int c; char* cstr; FILE * missatge;
    missatge = fopen("missatge.txt","r");
    char mystring [100];
        if (missatge == NULL) perror ("Error opening file");
        else { if ( fgets (mystring , 100 , missatge) != NULL ){ puts (mystring);}
 }
    fclose(missatge);
    char buf[100];
    int i=0;
    sprintf(buf,mystring, cntr++);
    while(buf[i]) {
        mydata[i]=buf[i];
        i++;
    }
    mydata[i]='\0';
    LMIC_setTxData2(1, mydata, strlen(buf), 0);
    remove("missatge.txt");
    os_setTimedCallback(j, os_getTime()+sec2osticks(20), do_send);
}
```

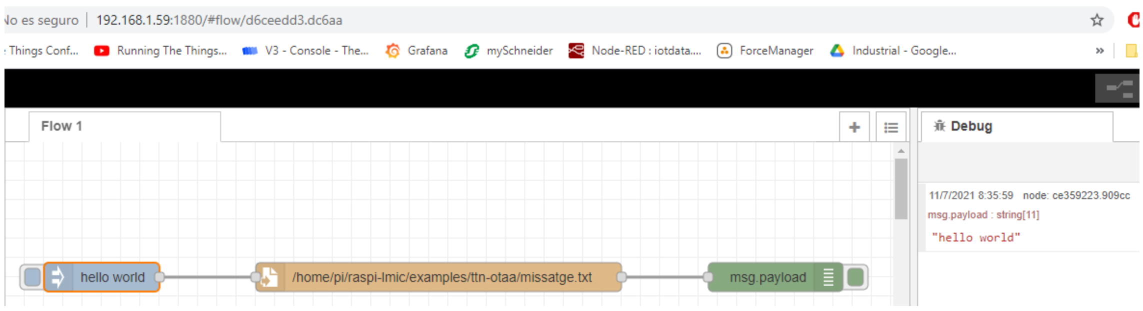
So instead we change the ttn-otaa.cpp like this



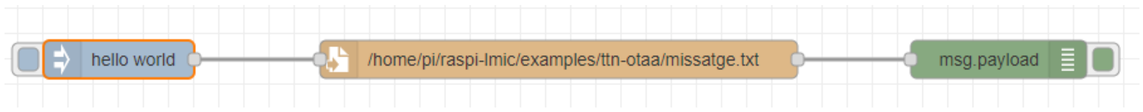
```
void do_send(osjob_t* j) {
    char strTime[16];
    getSystemTime(strTime , sizeof(strTime));
    printf("%s: ", strTime);

    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        printf("OP_TXRXPEND, not sending\n");
    } else {
        digitalWrite(RF_LED_PIN, HIGH);
        // Prepare upstream data transmission at the next possible time.
        //*******************************************************************************************

        int c; char* cstr; FILE * missatge;
        missatge = fopen("missatge.txt","r");
        char mystring [100];
        if (missatge == NULL) perror ("Error opening file");
        else
            { if ( fgets (mystring , 100 , missatge) != NULL ){ puts (mystring);}
            }
        fclose(missatge);
        char buf[100];
        int i=0;
        sprintf(buf,mystring, cstr++);
        while(buf[i])
            {
                mydata[i]=buf[i];
                i++;
            }
        mydata[i]='\0';
        LMIC_setTxData2(1, mydata, strlen(buf), 0);
        //remove("missatge.txt");
        //*******************************************************************************************
        //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        printf("Packet queued\n");
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
```

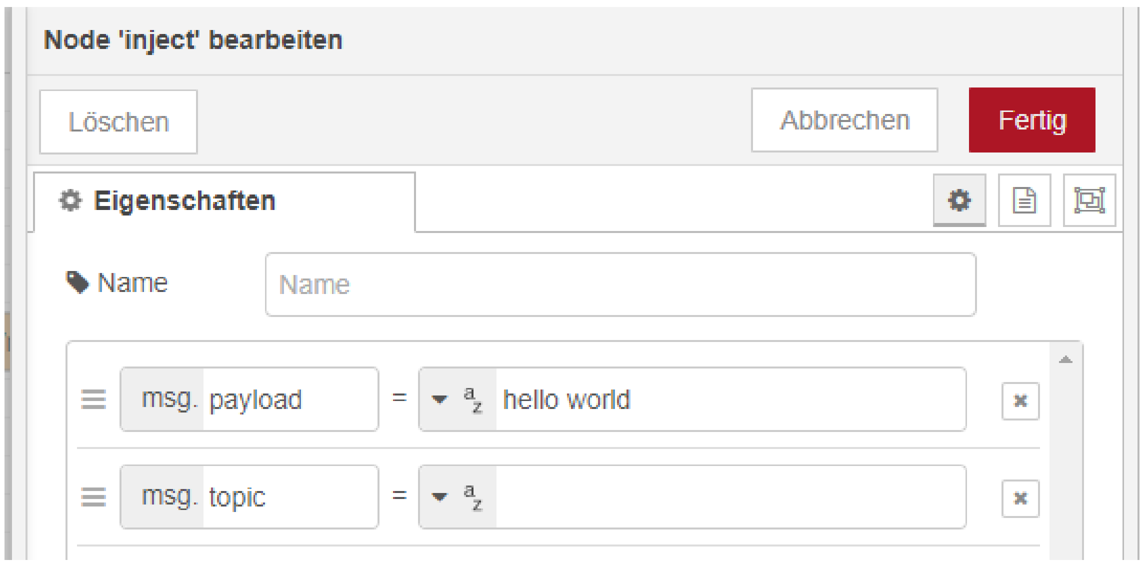
We compile

We inject a new data on the file


No es seguro | 192.168.1.59:1880/#flow/d6ceedd3.dc6aa
Things Conf... Running The Things... V3 - Console - The... Grafana mySchneider Node-RED : iotdata.... ForceManager Industrial - Google...
Flow 1
Debug
hello world
/home/pi/raspi-lmic/examples/ttn-otaa/missatge.txt
msg.payload

```
11/7/2021 8:35:59   node: ce359223.909cc
msg.payload : string[11]
"hello world"
```

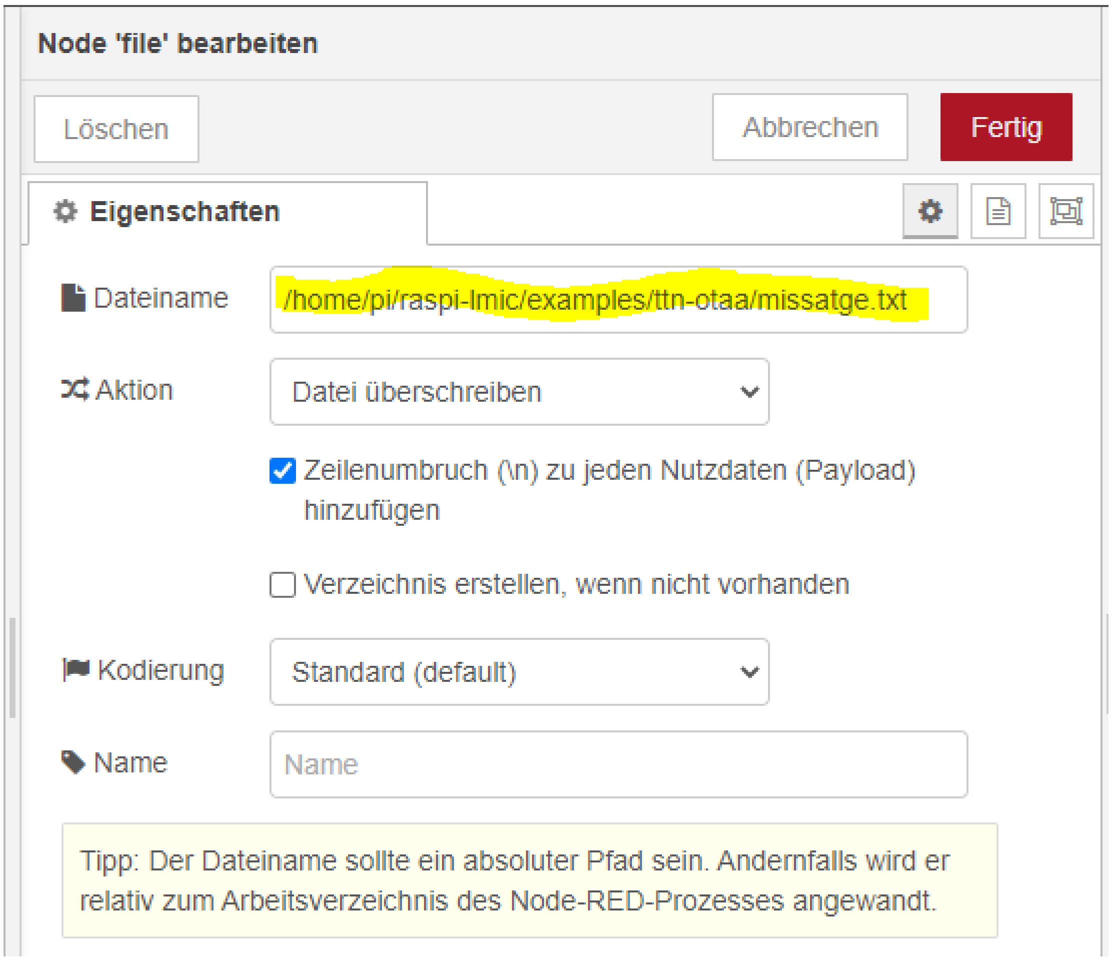




Name: Name

msg. payload = hello world

msg. topic =



Node 'file' bearbeiten

Löschen | Abbrechen | Fertig

Eigenschaften

Dateiname: /home/pi/raspi-lmic/examples/ttn-otaa/missatge.txt

Aktion: Datei überschreiben

[✓] Zeilenumbruch (\n) zu jeden Nutzdaten (Payload) hinzufügen

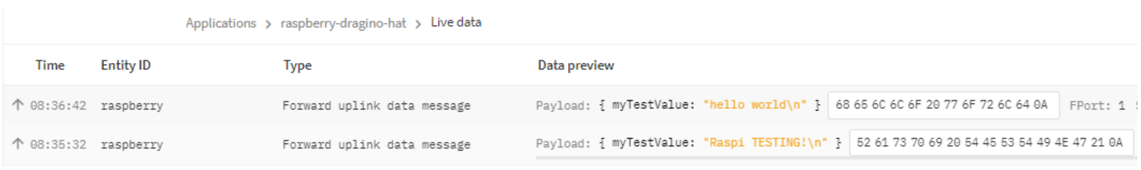
[ ] Verzeichnis erstellen, wenn nicht vorhanden

Kodierung: Standard (default)

Name: Name

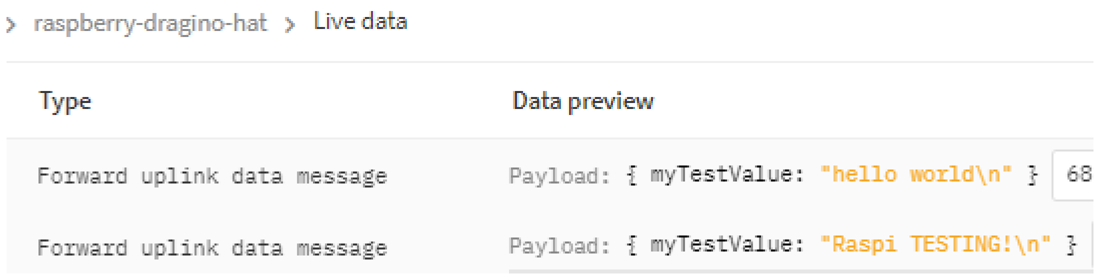
Tipp: Der Dateiname sollte ein absoluter Pfad sein. Andernfalls wird er relativ zum Arbeitsverzeichnis des Node-RED-Prozesses angewandt.

It Works ¡!!



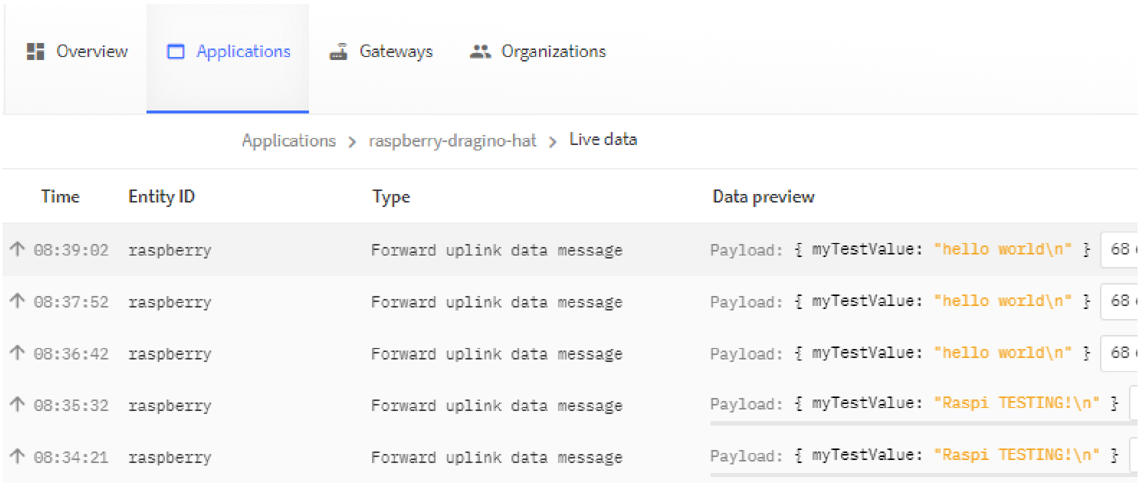
Applications > raspberry-dragino-hat > Live data

Time	Entity ID	Type	Data preview
↑ 08:36:42	raspberry	Forward uplink data message	Payload: { myTestValue: "hello world\n" } 68 65 6C 6C 6F 20 77 6F 72 6C 64 0A FPort: 1
↑ 08:35:32	raspberry	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!\n" } 52 61 73 70 69 20 54 45 53 54 49 4E 47 21 0A



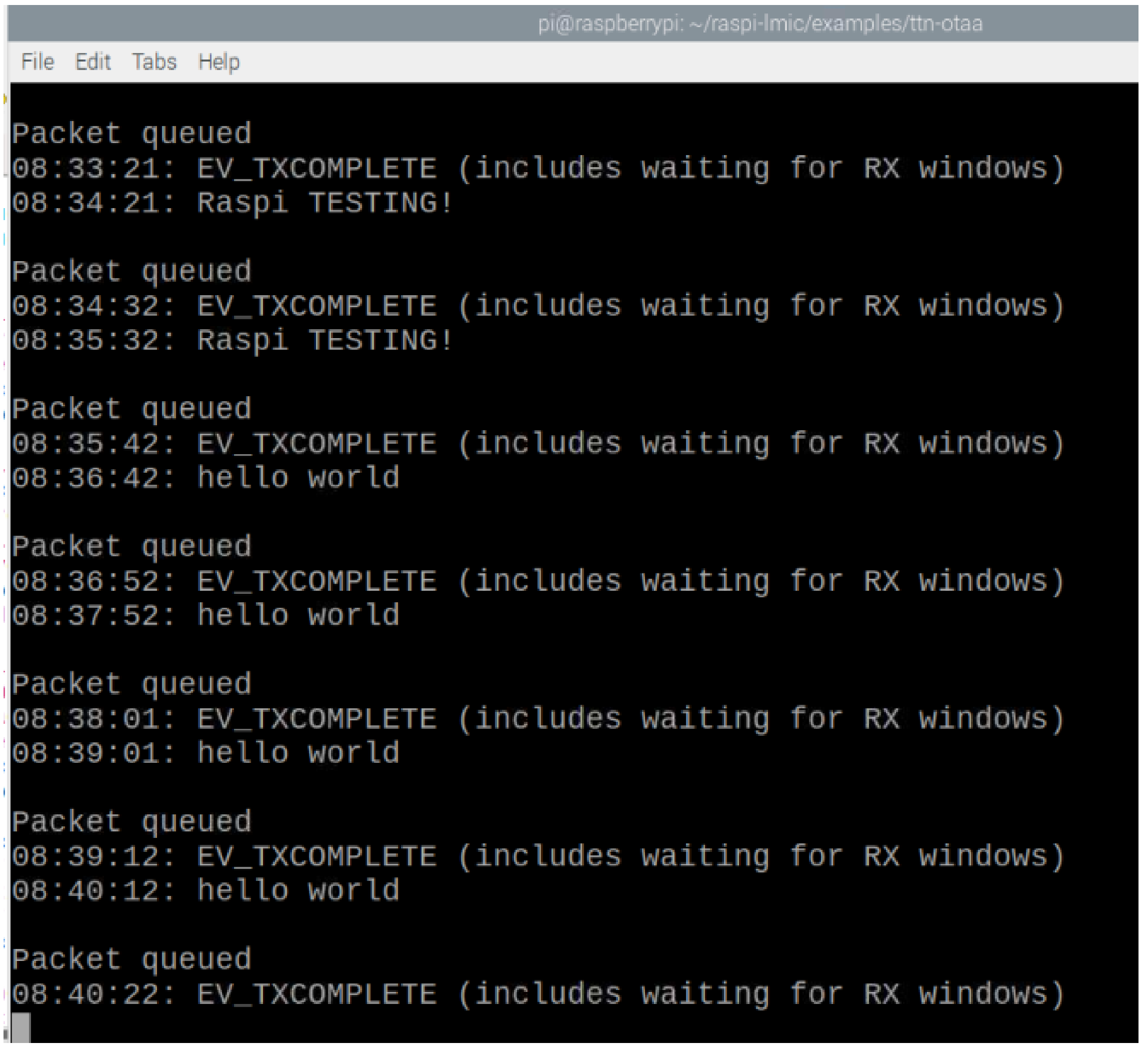
> raspberry-dragino-hat > Live data

Type	Data preview
Forward uplink data message	Payload: { myTestValue: "hello world\n" } 68
Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!\n" }



Applications > raspberry-dragino-hat > Live data

Time	Entity ID	Type	Data preview
↑ 08:39:02	raspberry	Forward uplink data message	Payload: { myTestValue: "hello world\n" } 68
↑ 08:37:52	raspberry	Forward uplink data message	Payload: { myTestValue: "hello world\n" } 68
↑ 08:36:42	raspberry	Forward uplink data message	Payload: { myTestValue: "hello world\n" } 68
↑ 08:35:32	raspberry	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!\n" }
↑ 08:34:21	raspberry	Forward uplink data message	Payload: { myTestValue: "Raspi TESTING!\n" }



```
Packet queued
08:33:21: EV_TXCOMPLETE (includes waiting for RX windows)
08:34:21: Raspi TESTING!

Packet queued
08:34:32: EV_TXCOMPLETE (includes waiting for RX windows)
08:35:32: Raspi TESTING!

Packet queued
08:35:42: EV_TXCOMPLETE (includes waiting for RX windows)
08:36:42: hello world

Packet queued
08:36:52: EV_TXCOMPLETE (includes waiting for RX windows)
08:37:52: hello world

Packet queued
08:38:01: EV_TXCOMPLETE (includes waiting for RX windows)
08:39:01: hello world

Packet queued
08:39:12: EV_TXCOMPLETE (includes waiting for RX windows)
08:40:12: hello world

Packet queued
08:40:22: EV_TXCOMPLETE (includes waiting for RX windows)
```

So this is the code

The complete code for ttn-otaa.cpp

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 *
 * Permission is hereby granted, free of charge, to anyone
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.
 *
 * This example sends a valid LoRaWAN packet with payload "Hello,
 * world!", using frequency and encryption settings matching those of
 * the The Things Network.
 *
 * This uses OTAA (Over-the-air activation), where where a DevEUI and
 * application key is configured, which are used in an over-the-air
 * activation procedure where a DevAddr and session keys are
 * assigned/generated for use with all further communication.
 *
 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
 * g1, 0.1% in g2), but not the TTN fair usage policy (which is probably
 * violated by this sketch when left running for longer)!

 * To use this sketch, first register your application and device with
 * the things network, to set or generate an AppEUI, DevEUI and AppKey.
 * Multiple devices can use the same AppEUI, but each device has its own
 * DevEUI and AppKey.
 *
 * Do not forget to define the radio type correctly in config.h.
 *

*******************************************************************************
**/

#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>

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

```
// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,0x70.
static const u1_t PROGMEM APPEUI[8]= { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]= { 0xB8, 0x27, 0xEB, 0xF1, 0x09, 0x34, 0x04, 0x00 };
// Here on Raspi we use part of MAC Address do define devEUI so
// This one above is not used, but you can still old method
// reverting the comments on the 2 following line
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
//void os_getDevEui (u1_t* buf) { getDevEuiFromMac(buf); }

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] = { 0x3C, 0x42, 0x0D, 0x18, 0xD1, 0x6F, 0x8C, 0xBA,
0x2E, 0x5A, 0x90, 0x19, 0xDE, 0xD0, 0x0A, 0xF1};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static uint8_t mydata[] = "Raspi TESTING!";

static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty)
// cycle limitations).
const unsigned TX_INTERVAL = 60;

//Flag for Ctrl-C
```

```
volatile sig_atomic_t force_exit = 0;

// LoRasPi board
// see https://github.com/hallard/LoRasPI
//#define RF_LED_PIN RPI_V2_GPIO_P1_16 // Led on GPIO23 so P1 connector pin #16
#define RF_CS_PIN  RPI_V2_GPIO_P1_22 // Slave Select on CE0 so P1 connector pin #24
#define RF_IRQ_PIN RPI_V2_GPIO_P1_07 // IRQ on GPIO25 so P1 connector pin #22
#define RF_RST_PIN RPI_V2_GPIO_P1_11 // RST on GPIO22 so P1 connector pin #15

// Raspberri PI Lora Gateway for multiple modules
// see https://github.com/hallard/RPI-Lora-Gateway
// Module 1 on board RFM95 868 MHz (example)
//#define RF_LED_PIN RPI_V2_GPIO_P1_07 // Led on GPIO4 so P1 connector pin #7
//#define RF_CS_PIN  RPI_V2_GPIO_P1_24 // Slave Select on CE0 so P1 connector pin #24
//#define RF_IRQ_PIN RPI_V2_GPIO_P1_22 // IRQ on GPIO25 so P1 connector pin #22
//#define RF_RST_PIN RPI_V2_GPIO_P1_29 // Reset on GPIO5 so P1 connector pin #29


// Dragino Raspberry PI hat (no onboard led)
// see https://github.com/dragino/Lora
#define RF_CS_PIN  RPI_V2_GPIO_P1_22 // Slave Select on GPIO25 so P1 connector pin #22
#define RF_IRQ_PIN RPI_V2_GPIO_P1_07 // IRQ on GPIO4 so P1 connector pin #7
#define RF_RST_PIN RPI_V2_GPIO_P1_11 // Reset on GPIO17 so P1 connector pin #11

// Pin mapping
const lmic_pinmap lmic_pins = {
   .nss  = RF_CS_PIN,
   .rxtx = LMIC_UNUSED_PIN,
   .rst  = RF_RST_PIN,
   .dio  = {LMIC_UNUSED_PIN, LMIC_UNUSED_PIN, LMIC_UNUSED_PIN},
};
```

```
#ifndef RF_LED_PIN
#define RF_LED_PIN NOT_A_PIN
#endif

void do_send(osjob_t* j) {
   char strTime[16];
   getSystemTime(strTime , sizeof(strTime));
   printf("%s: ", strTime);

   // Check if there is not a current TX/RX job running
   if (LMIC.opmode & OP_TXRXPEND) {
      printf("OP_TXRXPEND, not sending\n");
   } else {
      digitalWrite(RF_LED_PIN, HIGH);
      // Prepare upstream data transmission at the next possible time.

//****************************************************************************
*********

      int c; char* cstr; FILE * missatge;
      missatge = fopen("missatge.txt","r");
      char mystring [100];
      if (missatge == NULL) perror ("Error opening file");
      else
         { if ( fgets (mystring , 100 , missatge) != NULL ){ puts (mystring);}
         }
      fclose(missatge);
      char buf[100];
      int i=0;
      sprintf(buf,mystring, cstr++);
      while(buf[i])
```

```
            {
                mydata[i]=buf[i];
                i++;
            }
        mydata[i]='\0';
        LMIC_setTxData2(1, mydata, strlen(buf), 0);
        //remove("missatge.txt");

//**************************************************************************
*********
        //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        printf("Packet queued\n");
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void onEvent (ev_t ev) {
    char strTime[16];
    getSystemTime(strTime , sizeof(strTime));
    printf("%s: ", strTime);

    switch(ev) {
        case EV_SCAN_TIMEOUT:
            printf("EV_SCAN_TIMEOUT\n");
        break;
        case EV_BEACON_FOUND:
            printf("EV_BEACON_FOUND\n");
        break;
        case EV_BEACON_MISSED:
            printf("EV_BEACON_MISSED\n");
        break;
        case EV_BEACON_TRACKED:
```

```
    printf("EV_BEACON_TRACKED\n");
  break;
  case EV_JOINING:
    printf("EV_JOINING\n");
  break;
  case EV_JOINED:
    printf("EV_JOINED\n");
    digitalWrite(RF_LED_PIN, LOW);
    // Disable link check validation (automatically enabled
    // during join, but not supported by TTN at this time).
    LMIC_setLinkCheckMode(0);
  break;
  case EV_RFU1:
    printf("EV_RFU1\n");
  break;
  case EV_JOIN_FAILED:
    printf("EV_JOIN_FAILED\n");
  break;
  case EV_REJOIN_FAILED:
    printf("EV_REJOIN_FAILED\n");
  break;
  case EV_TXCOMPLETE:
    printf("EV_TXCOMPLETE (includes waiting for RX windows)\n");
    if (LMIC.txrxFlags & TXRX_ACK)
      printf("%s Received ack\n", strTime);
    if (LMIC.dataLen) {
      printf("%s Received %d bytes of payload\n", strTime, LMIC.dataLen);
    }
    digitalWrite(RF_LED_PIN, LOW);
    // Schedule next transmission
    os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
```

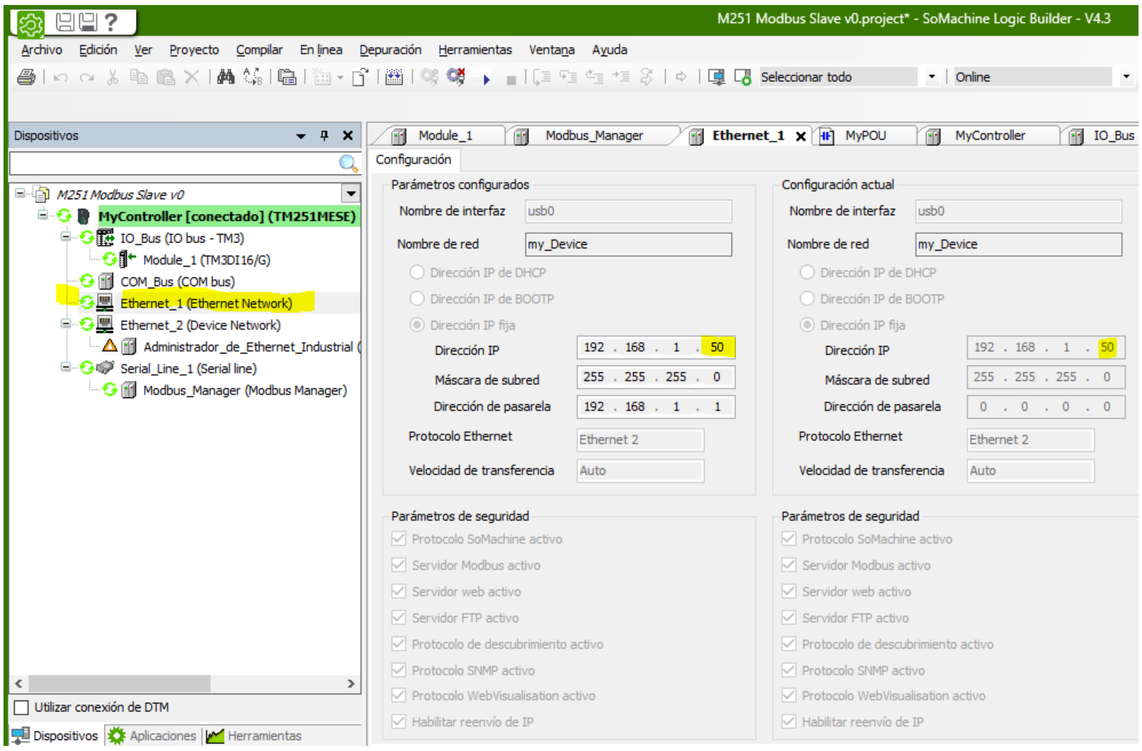
```
        break;
        case EV_LOST_TSYNC:
          printf("EV_LOST_TSYNC\n");
        break;
        case EV_RESET:
          printf("EV_RESET\n");
        break;
        case EV_RXCOMPLETE:
          // data received in ping slot
          printf("EV_RXCOMPLETE\n");
        break;
        case EV_LINK_DEAD:
          printf("EV_LINK_DEAD\n");
        break;
        case EV_LINK_ALIVE:
          printf("EV_LINK_ALIVE\n");
        break;
        default:
          printf("Unknown event\n");
        break;
    }
}

/* ===================================================================
Function: sig_handler
Purpose : Intercept CTRL-C keyboard to close application
Input   : signal received
Output  : -
Comments: -
=================================================================== */
void sig_handler(int sig)
```

```
{
  printf("\nBreak received, exiting!\n");
  force_exit=true;
}

/* ======================================================================
Function: main
Purpose : not sure ;)
Input   : command line parameters
Output  : -
Comments: -
====================================================================== */
int main(void)
{
   // caught CTRL-C to do clean-up
   signal(SIGINT, sig_handler);

   printf("%s Starting\n", __BASEFILE__);

     // Init GPIO bcm
   if (!bcm2835_init()) {
      fprintf( stderr, "bcm2835_init() Failed\n\n" );
      return 1;
   }

           // Show board config
   printConfig(RF_LED_PIN);
   printKeys();

   // Light off on board LED
   pinMode(RF_LED_PIN, OUTPUT);
   digitalWrite(RF_LED_PIN, HIGH);

   // LMIC init
   os_init();
   // Reset the MAC state. Session and pending data transfers will be discarded.
   LMIC_reset();

   // Start job (sending automatically starts OTAA too)
   do_send(&sendjob);

   while(!force_exit) {
     os_runloop_once();

     // We're on a multitasking OS let some time for others
     // Without this one CPU is 99% and with this one just 3%
     // On a Raspberry PI 3
     usleep(1000);
   }

   // We're here because we need to exit, do it clean

   // Light off on board LED
   digitalWrite(RF_LED_PIN, LOW);

   // module CS line High
   digitalWrite(lmic_pins.nss, HIGH);
   printf( "\n%s, done my job!\n", __BASEFILE__ );
   bcm2835_close();
   return 0;
}
```

Now we need to prepare the Modbus read node to extract the data from a device thru Modbus/TCP and send it thru LoRaWAN

Let's prepare the PLC

Ethernet 1 PLC IP: 192.168.1.50

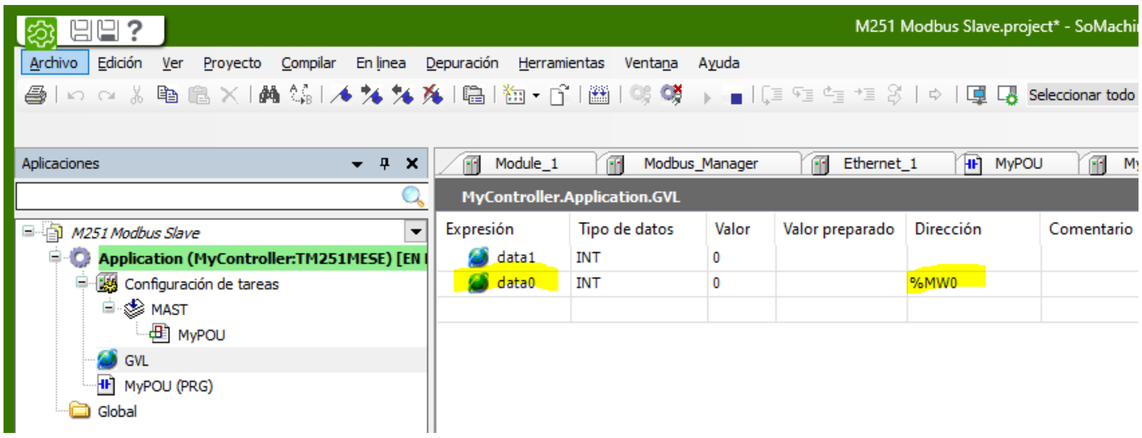


M251 Modbus Slave v0
- MyController [conectado] (TM251MESE)
  - IO_Bus (IO bus - TM3)
    - Module_1 (TM3DI16/G)
  - COM_Bus (COM bus)
  - Ethernet_1 (Ethernet Network)
  - Ethernet_2 (Device Network)
    - Administrador_de_Ethernet_Industrial (
  - Serial_Line_1 (Serial line)
    - Modbus_Manager (Modbus Manager)

	Parámetros configurados	Configuración actual
Nombre de interfaz	usb0	usb0
Nombre de red	my_Device	my_Device
Dirección IP de DHCP	○	○
Dirección IP de BOOTP	○	○
Dirección IP fija	●	●
Dirección IP	192 . 168 . 1 . 50	192 . 168 . 1 . 50
Máscara de subred	255 . 255 . 255 . 0	255 . 255 . 255 . 0
Dirección de pasarela	192 . 168 . 1 . 1	0 . 0 . 0 . 0
Protocolo Ethernet	Ethernet 2	Ethernet 2
Velocidad de transferencia	Auto	Auto

Parámetros de seguridad	Configurados	Actual
Protocolo SoMachine activo	✓	✓
Servidor Modbus activo	✓	✓
Servidor web activo	✓	✓
Servidor FTP activo	✓	✓
Protocolo de descubrimiento activo	✓	✓
Protocolo SNMP activo	✓	✓
Protocolo WebVisualisation activo	✓	✓
Habilitar reenvío de IP	✓	✓

We allocate on PLC memory a variable to store digital inputs

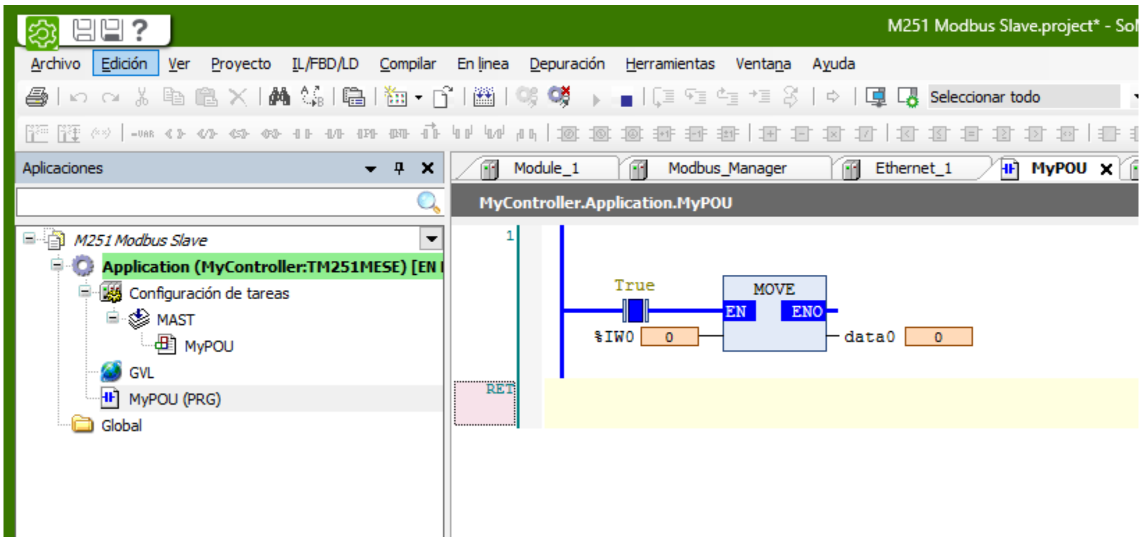


M251 Modbus Slave
- Application (MyController:TM251MESE) [EN
  - Configuración de tareas
    - MAST
      - MyPOU
  - GVL
  - MyPOU (PRG)
- Global

MyController.Application.GVL

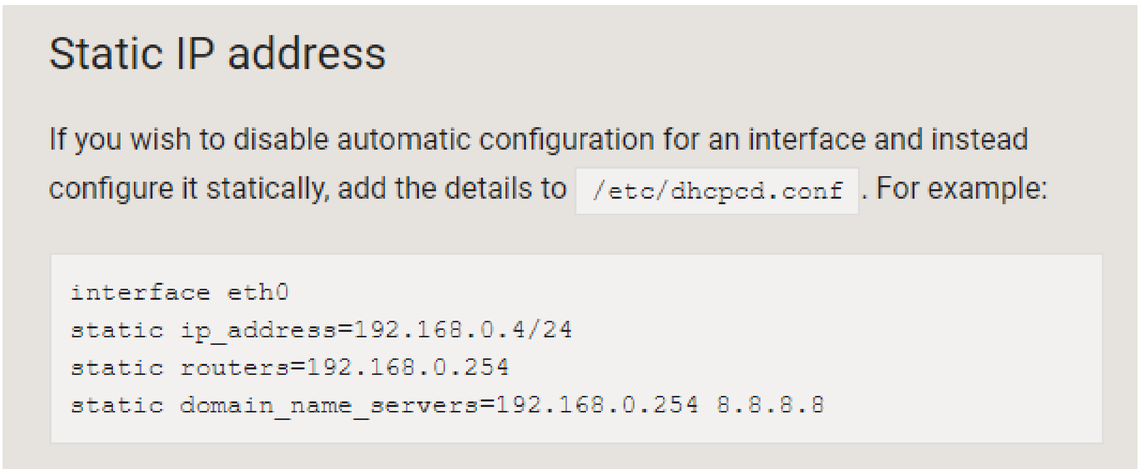
Expresión	Tipo de datos	Valor	Valor preparado	Dirección	Comentario
data1	INT	0			
data0	INT	0		%MW0	

We prepare a programm just to store digital inputs on register %MW0



And then we go to Node-RED an try to get the input status

First we need to assign eth0 as fixed IP on the Raspberry

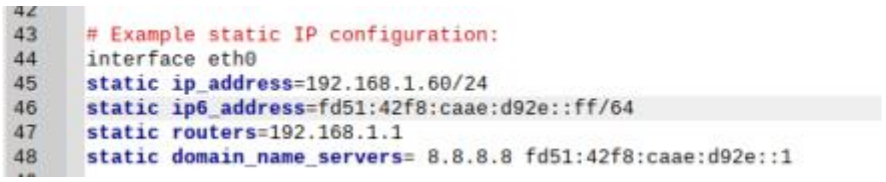


Static IP address

If you wish to disable automatic configuration for an interface and instead configure it statically, add the details to `/etc/dhcpcd.conf`. For example:

```
interface eth0
static ip_address=192.168.0.4/24
static routers=192.168.0.254
static domain_name_servers=192.168.0.254 8.8.8.8
```

So I set this one for the Raspberry PI



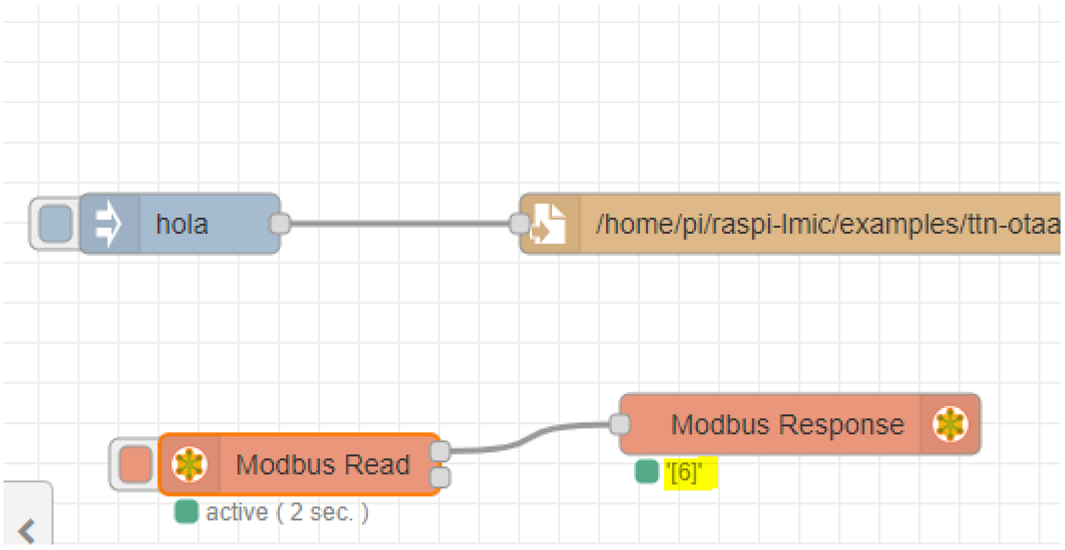
```
# Example static IP configuration:
interface eth0
static ip_address=192.168.1.60/24
static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.1.1
static domain_name_servers= 8.8.8.8 fd51:42f8:caae:d92e::1
```

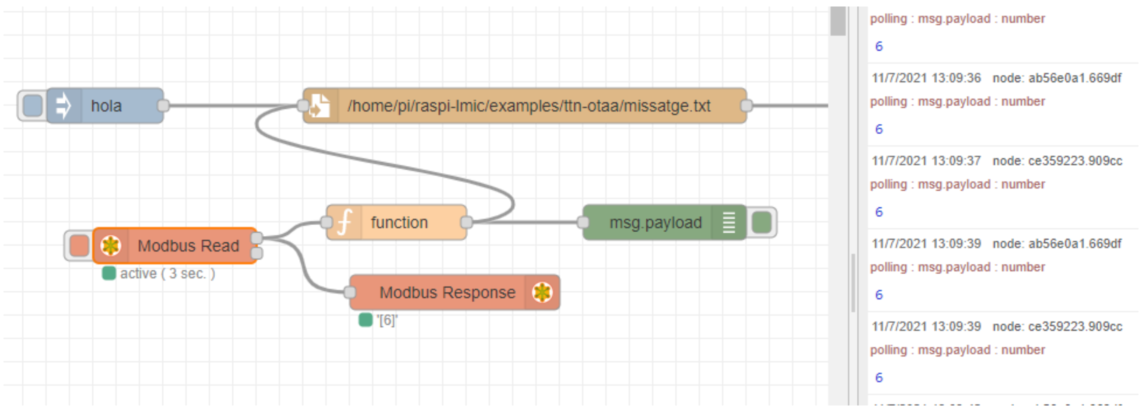
Finally we have used a router in order to have a easy IP configuration

(With a direct connection from Raspberry to PLC I had a mismatch on IP domains between wifi and eth0that are in the same subnet)

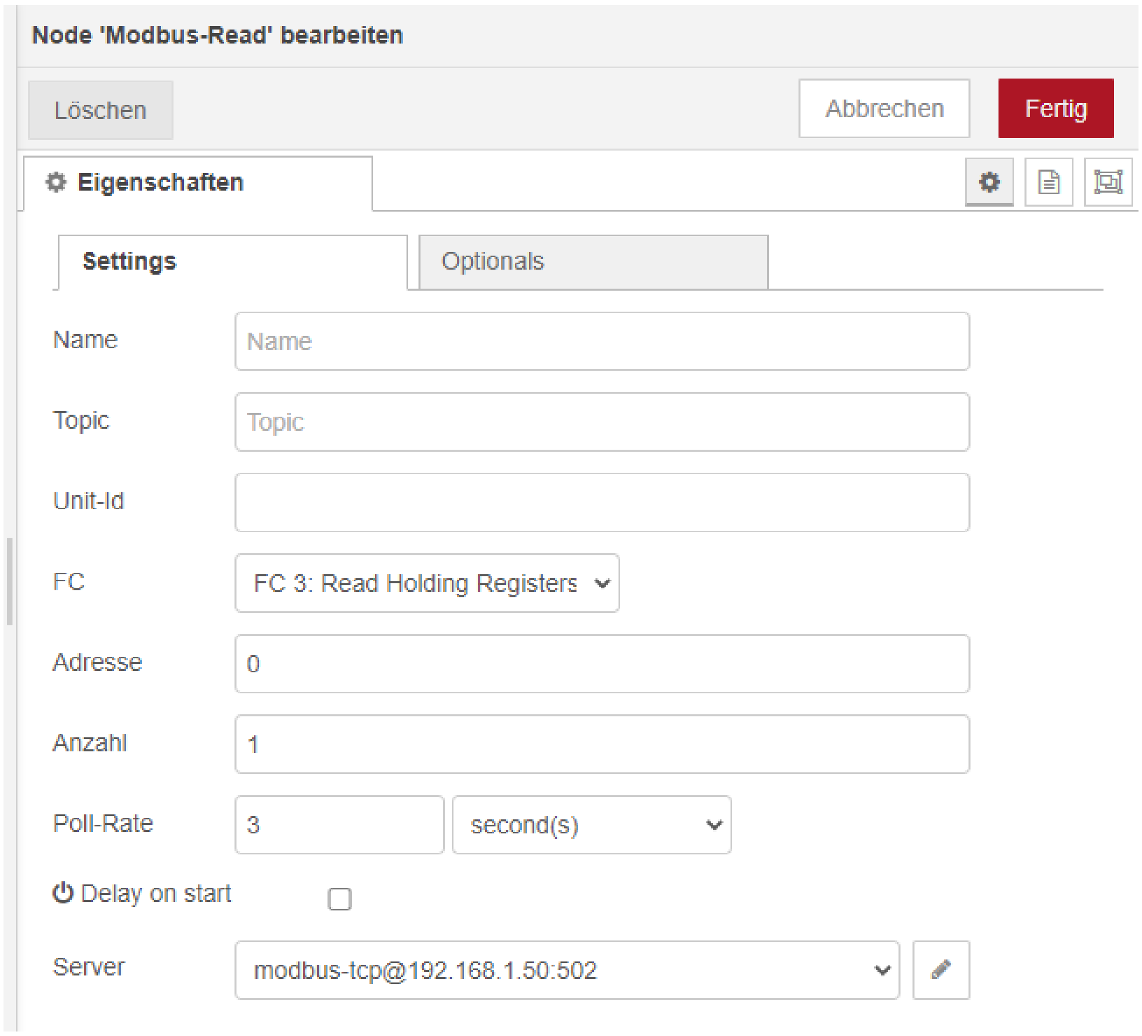
The problem may be solved disabling wifi on the raspberry PI

So now our raspberry is Reading from PLC, with corresponding input status

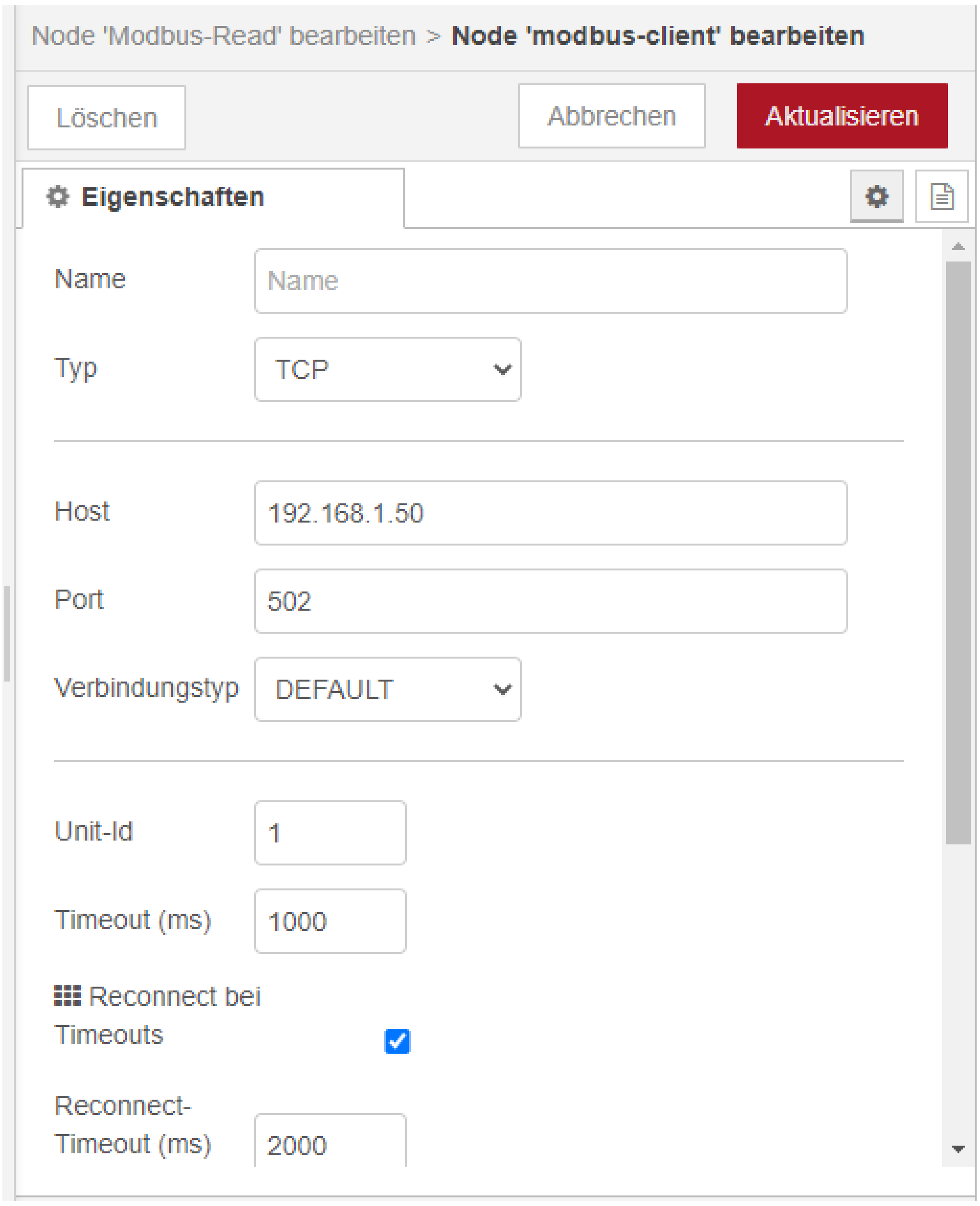




```
polling : msg.payload : number
 6
11/7/2021 13:09:36   node: ab56e0a1.669df
polling : msg.payload : number
 6
11/7/2021 13:09:37   node: ce359223.909cc
polling : msg.payload : number
 6
11/7/2021 13:09:39   node: ab56e0a1.669df
polling : msg.payload : number
 6
11/7/2021 13:09:39   node: ce359223.909cc
polling : msg.payload : number
 6
```



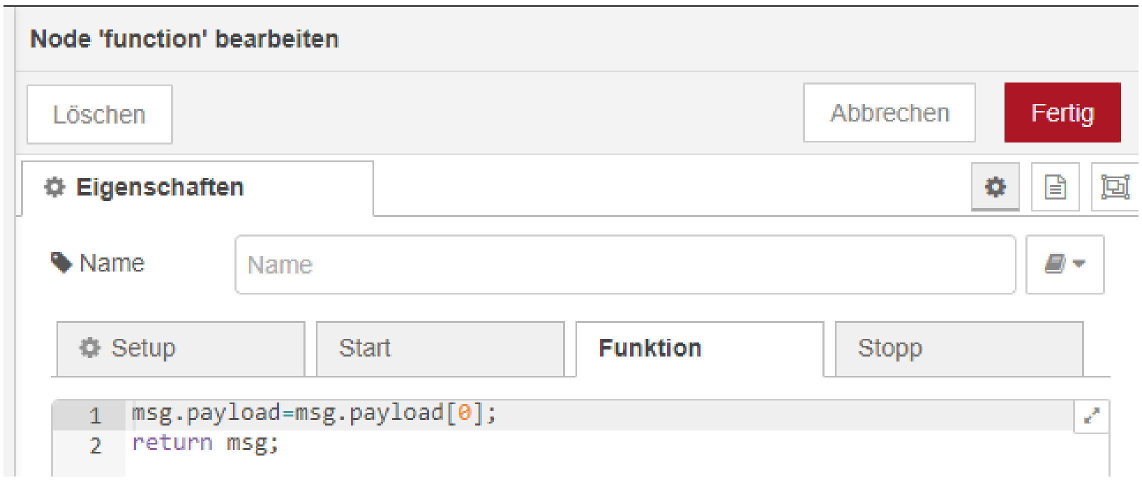
Name	Name
Topic	Topic
Unit-Id	
FC	FC 3: Read Holding Registers
Adresse	0
Anzahl	1
Poll-Rate	3 second(s)
Delay on start	☐
Server	modbus-tcp@192.168.1.50:502


Node 'Modbus-Read' bearbeiten > **Node 'modbus-client' bearbeiten**

Löschen | Abbrechen | Aktualisieren

Eigenschaften

Name	Name
Typ	TCP
Host	192.168.1.50
Port	502
Verbindungstyp	DEFAULT
Unit-Id	1
Timeout (ms)	1000
Reconnect bei Timeouts	✓
Reconnect-Timeout (ms)	2000


Node 'function' bearbeiten

Löschen | Abbrechen | Fertig

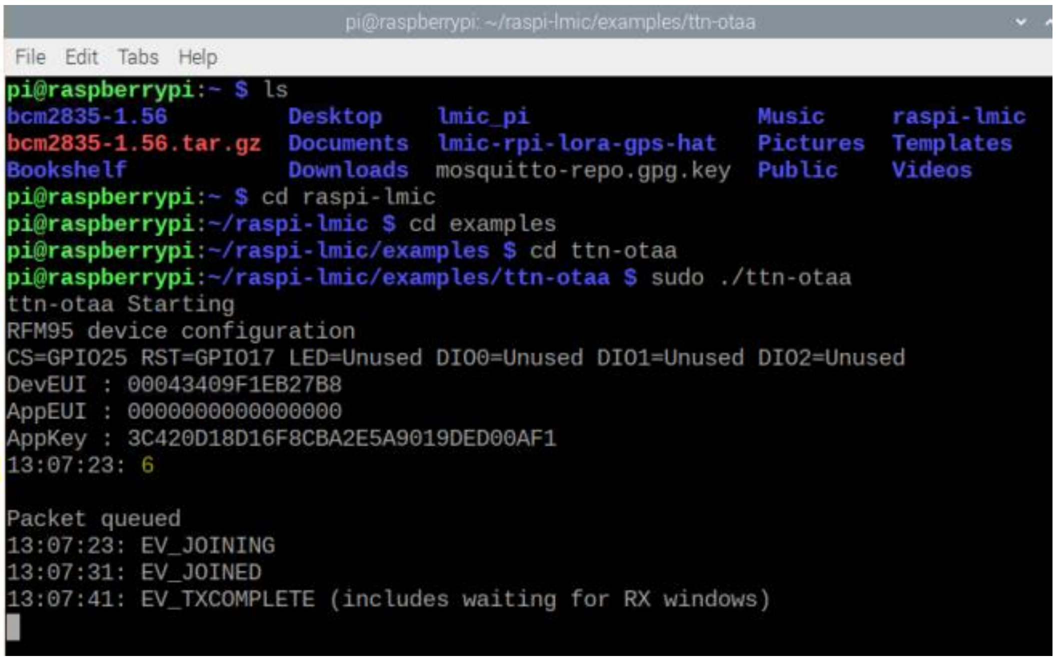
Eigenschaften

Name: Name

Setup | Start | **Funktion** | Stopp

```
msg.payload=msg.payload[0];
return msg;
```

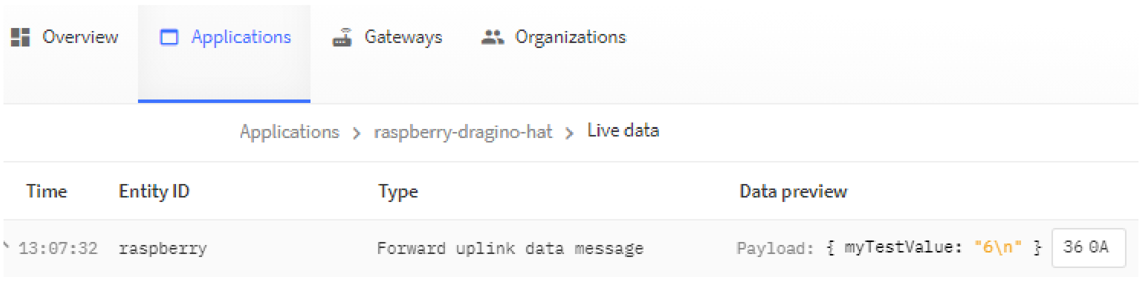
Now let’s write this value on the file



```
pi@raspberrypi: ~/raspi-lmic/examples/ttn-otaa
File  Edit  Tabs  Help
pi@raspberrypi:~ $ ls
bcm2835-1.56          Desktop    lmic_pi                Music     raspi-lmic
bcm2835-1.56.tar.gz   Documents  lmic-rpi-lora-gps-hat  Pictures  Templates
Bookshelf             Downloads  mosquitto-repo.gpg.key Public    Videos
pi@raspberrypi:~ $ cd raspi-lmic
pi@raspberrypi:~/raspi-lmic $ cd examples
pi@raspberrypi:~/raspi-lmic/examples $ cd ttn-otaa
pi@raspberrypi:~/raspi-lmic/examples/ttn-otaa $ sudo ./ttn-otaa
ttn-otaa Starting
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : 00043409F1EB27B8
AppEUI : 0000000000000000
AppKey : 3C420D18D16F8CBA2E5A9019DED00AF1
13:07:23: 6

Packet queued
13:07:23: EV_JOINING
13:07:31: EV_JOINED
13:07:41: EV_TXCOMPLETE (includes waiting for RX windows)
```

It Works!

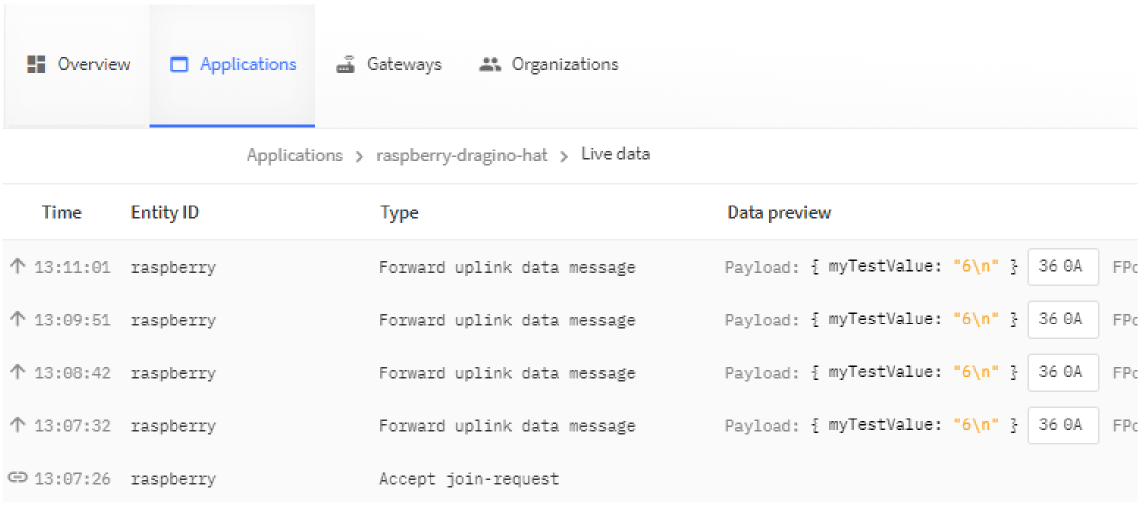
We get the value!



Overview | Applications | Gateways | Organizations

Applications > raspberry-dragino-hat > Live data

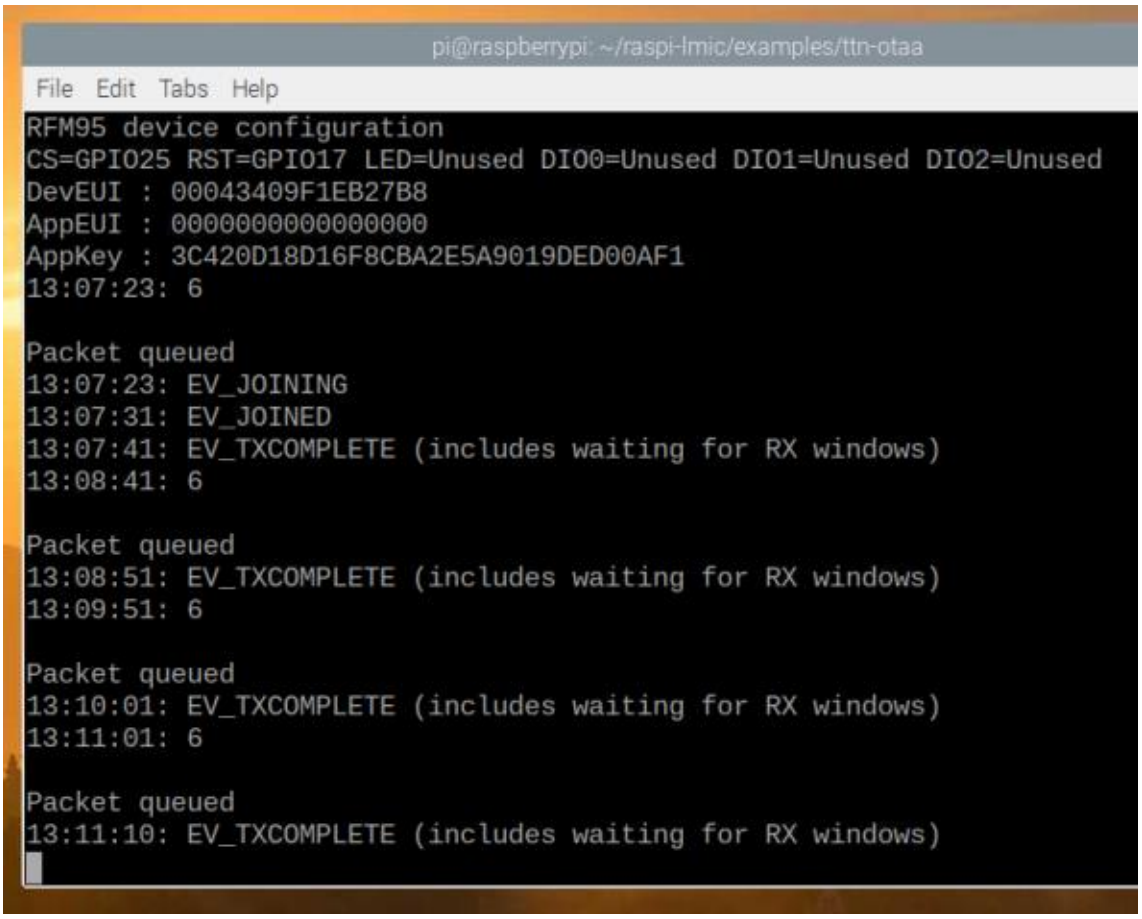
Time	Entity ID	Type	Data preview
13:07:32	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A



Overview | Applications | Gateways | Organizations

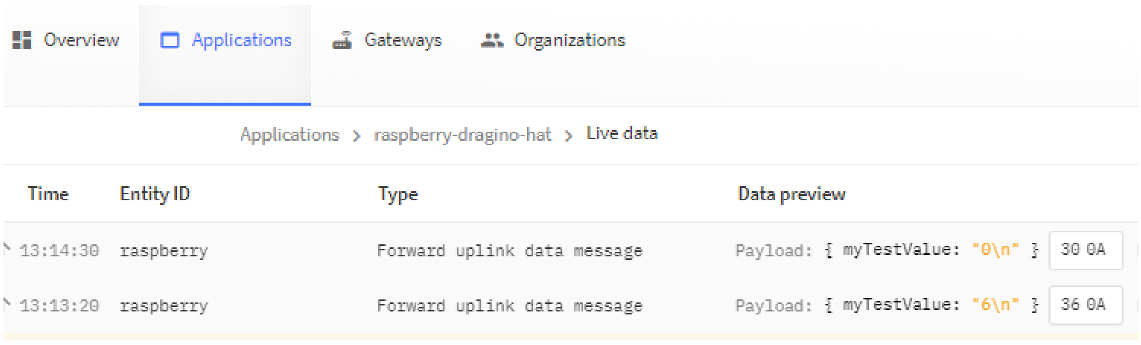
Applications > raspberry-dragino-hat > Live data

Time	Entity ID	Type	Data preview
13:11:01	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A FPo
13:09:51	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A FPo
13:08:42	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A FPo
13:07:32	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A FPo
13:07:26	raspberry	Accept join-request	



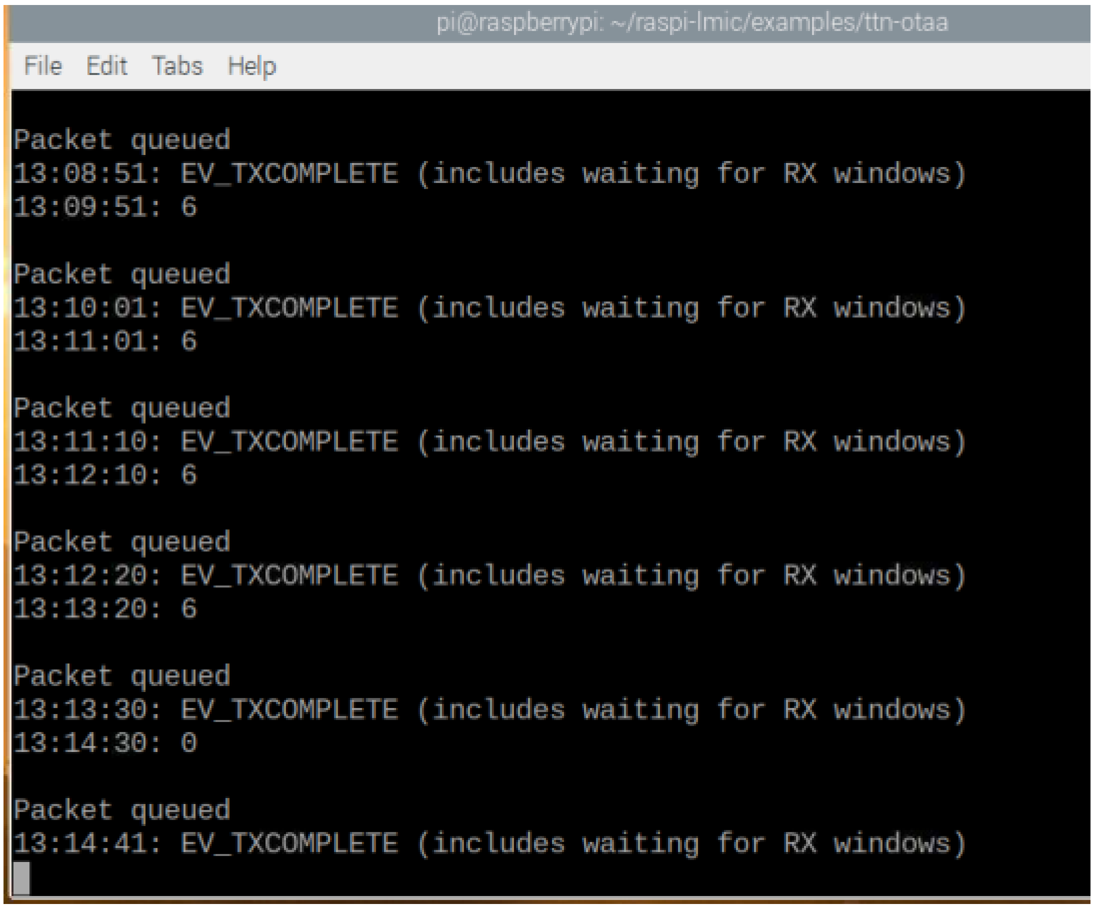
```
RFM95 device configuration
CS=GPIO25 RST=GPIO17 LED=Unused DIO0=Unused DIO1=Unused DIO2=Unused
DevEUI : 00043409F1EB27B8
AppEUI : 0000000000000000
AppKey : 3C420D18D16F8CBA2E5A9019DED00AF1
13:07:23: 6

Packet queued
13:07:23: EV_JOINING
13:07:31: EV_JOINED
13:07:41: EV_TXCOMPLETE (includes waiting for RX windows)
13:08:41: 6

Packet queued
13:08:51: EV_TXCOMPLETE (includes waiting for RX windows)
13:09:51: 6

Packet queued
13:10:01: EV_TXCOMPLETE (includes waiting for RX windows)
13:11:01: 6

Packet queued
13:11:10: EV_TXCOMPLETE (includes waiting for RX windows)
```

If we try now to change the status to 0



Time	Entity ID	Type	Data preview
13:14:30	raspberry	Forward uplink data message	Payload: { myTestValue: "0\n" } 30 0A
13:13:20	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A

Voilà, its working!



```
Packet queued
13:08:51: EV_TXCOMPLETE (includes waiting for RX windows)
13:09:51: 6

Packet queued
13:10:01: EV_TXCOMPLETE (includes waiting for RX windows)
13:11:01: 6

Packet queued
13:11:10: EV_TXCOMPLETE (includes waiting for RX windows)
13:12:10: 6

Packet queued
13:12:20: EV_TXCOMPLETE (includes waiting for RX windows)
13:13:20: 6

Packet queued
13:13:30: EV_TXCOMPLETE (includes waiting for RX windows)
13:14:30: 0

Packet queued
13:14:41: EV_TXCOMPLETE (includes waiting for RX windows)
```

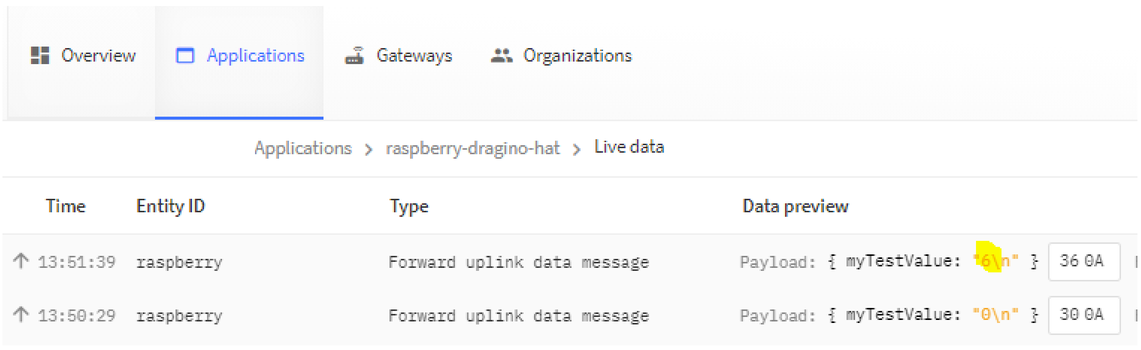
Next step will be to connect the Raspberry without router.

Probably disabling the wifi Access.

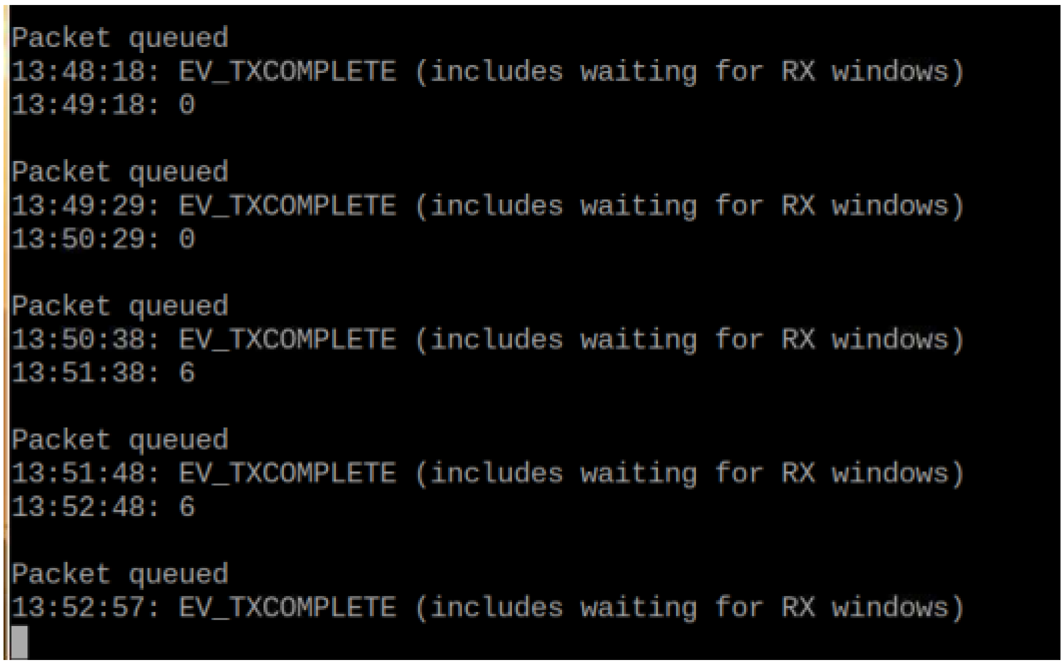
Yes, if we disable the wifi Access on the Raspberry, then we do not need a router.

We just connect the Raspberry directly to the PLC.

And it Works!



Time	Entity ID	Type	Data preview
↑ 13:51:39	raspberry	Forward uplink data message	Payload: { myTestValue: "6\n" } 36 0A
↑ 13:50:29	raspberry	Forward uplink data message	Payload: { myTestValue: "0\n" } 30 0A



```
Packet queued
13:48:18: EV_TXCOMPLETE (includes waiting for RX windows)
13:49:18: 0

Packet queued
13:49:29: EV_TXCOMPLETE (includes waiting for RX windows)
13:50:29: 0

Packet queued
13:50:38: EV_TXCOMPLETE (includes waiting for RX windows)
13:51:38: 6

Packet queued
13:51:48: EV_TXCOMPLETE (includes waiting for RX windows)
13:52:48: 6

Packet queued
13:52:57: EV_TXCOMPLETE (includes waiting for RX windows)
```

You can find the code here

https://github.com/xavierflorensa/Modbus-TCP-to-LoRaWAN-converter-Raspberry/tree/main