Transponder connection to The Things Network

Author: Marco Rainone
Document Version: 1.0
Date: 2019-08-28

index

This tutorial describes the steps necessary to allow the transponder to transmit the Weather Station data to The Things Network.

1 If you don't have a TTN account ...

1. Go to TTN and create a TTN account.
2. After Login, Choose “Console” to enter the TTN console page.

2 Prerequisite for the next steps …

In order to transfer data to the application registered on TTN, it is necessary to have a LoraWAN Gateway registered on TTN in the vicinity of the transponder, able to receive its messages.

LoraWAN gateways are in fact a bridge between LoraWAN end-devices and the TTN servers.

If necessary, the introductions for the TTN network structure can be found on this link:

If you don't have a TTN account ...
Version 1.0

https://www.thethingsnetwork.org/docs/

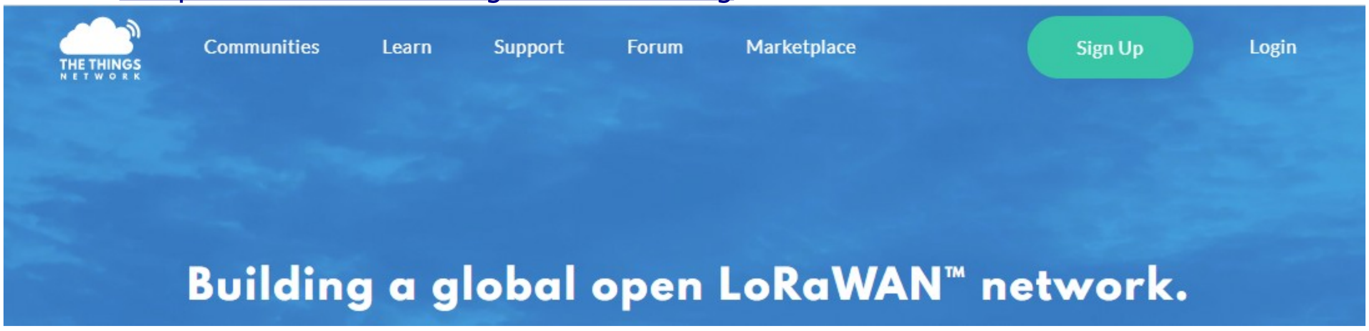
If we are not near any TTN registered gateway, you need to get a gateway (buy it or build it) and proceed with its registration on TTN.

For this operation follow the tutorial:

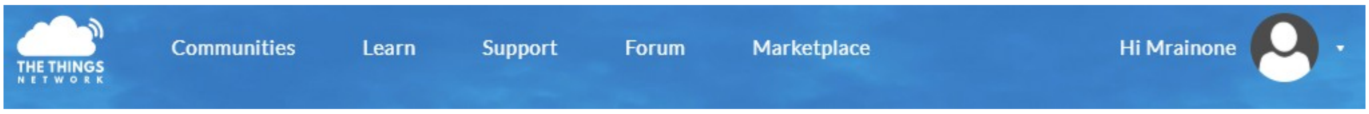
https://www.thethingsnetwork.org/docs/gateways/registration.html

3 Creating an application

Go to https://www.thethingsnetwork.org

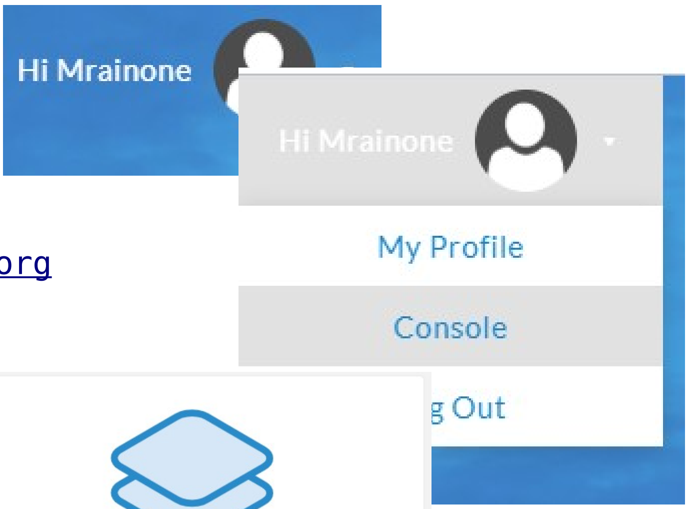


Login to TTN: https://account.thethingsnetwork.org/users/login



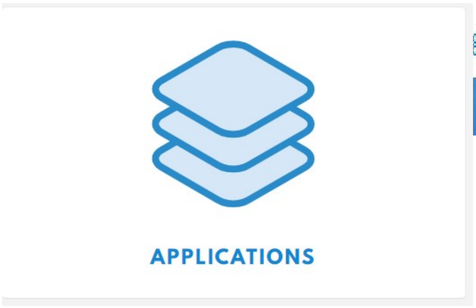
Click on the right icon.

Select “Console”

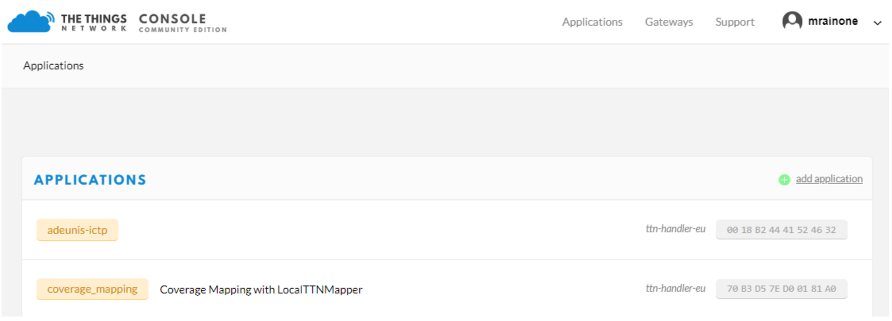


https://console.thethingsnetwork.org

Select “Applications”



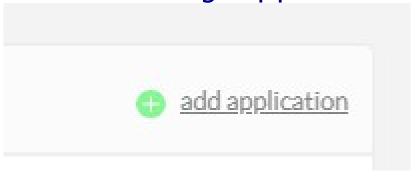
https://console.thethingsnetwork.org/applications



adeunis-ictp		ttn-handler-eu	00 18 B2 44 41 52 46 32
coverage_mapping	Coverage Mapping with LocalTTNMapper	ttn-handler-eu	70 B3 D5 7E D0 01 81 A0

On right, press “Add application”

https://console.thethingsnetwork.org/applications/add



Set these parameters:

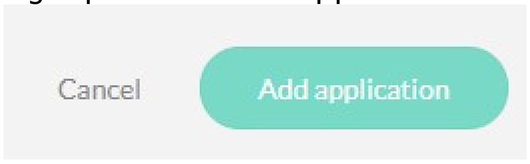
1. Application ID (The unique identifier of your application on the network)

   Example: **weather_station_data**

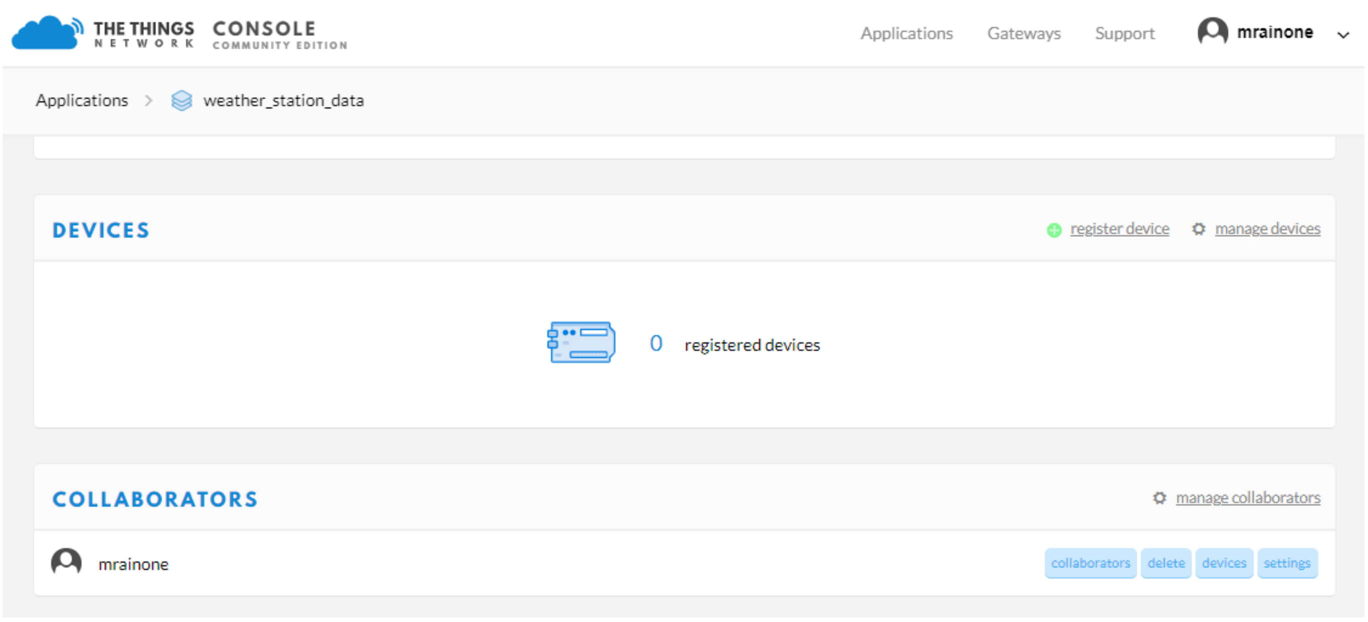
2. Description (A human readable description of your new app)

   Example: **WS data acquisition through Lorawan transponder**

At the bottom of the page press "Add application"



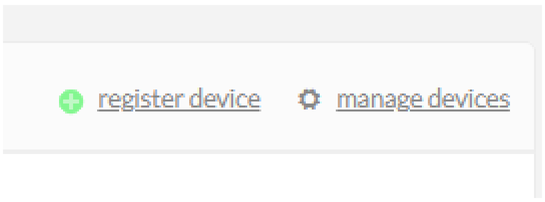
https://console.thethingsnetwork.org/applications/weather_station_data



4 Registration of transponder in the TTN application

On the top right of web page, press “register device”

https://console.thethingsnetwork.org/applications/weather_station_data/devices/register



Set the “Device ID”: this is the unique identifier for the device in this app. The device ID will be immutable.

Example:

a possible name could be:

tr_5in1_01

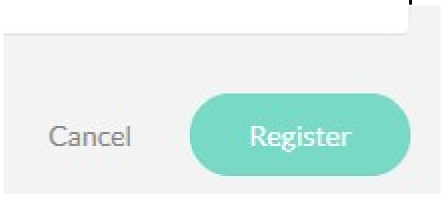
where:

1. tr: device is transponder
2. 5in1: the type of WS to which the transponder is connected
3. 01: a number that identifies the transponder within the application

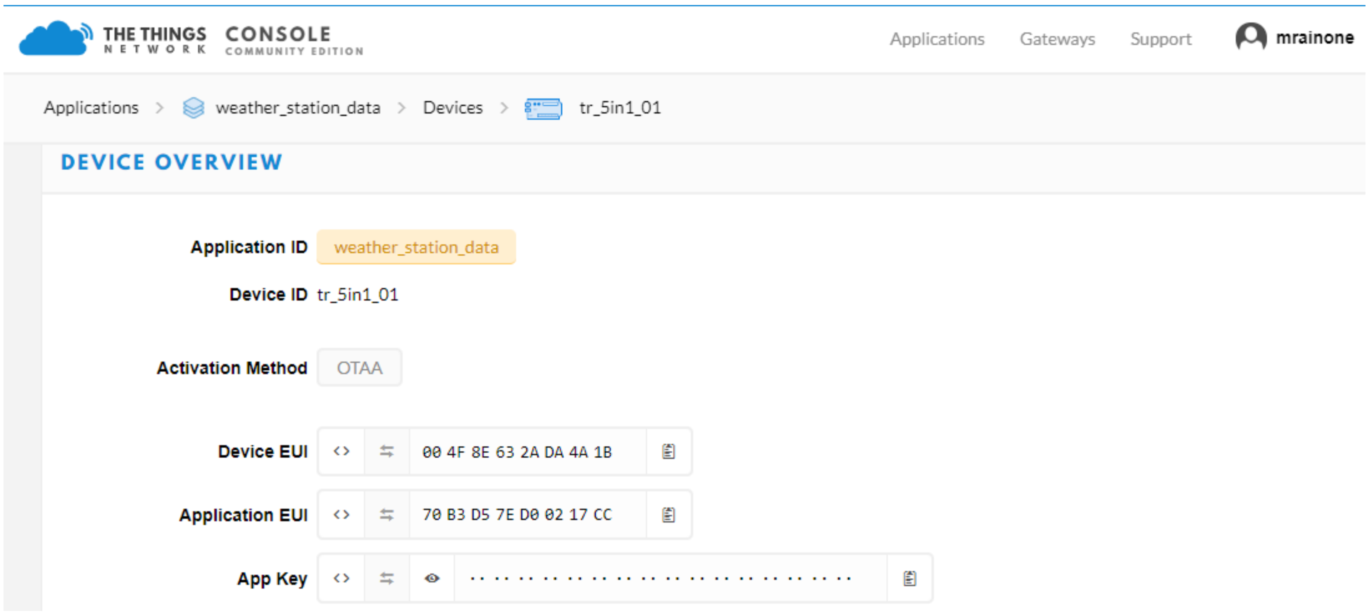
Device EUI: The device EUI is the unique identifier for this device on the network. You can change the EUI later.

Click on the symbol to the left of the box to generate the Device EUI automatically.

Finally, press Register at the bottom of the page on the right.



https://console.thethingsnetwork.org/applications/weather_station_data/devices/tr_5in1_01


Applications > weather_station_data > Devices > tr_5in1_01

DEVICE OVERVIEW

Application ID weather_station_data

Device ID tr_5in1_01

Activation Method OTAA

Device EUI 00 4F 8E 63 2A DA 4A 1B

Application EUI 70 B3 D5 7E D0 02 17 CC

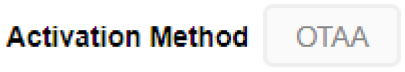
App Key ································

The activation method actually used is "**Over-the-Air Activation**" (OTAA).

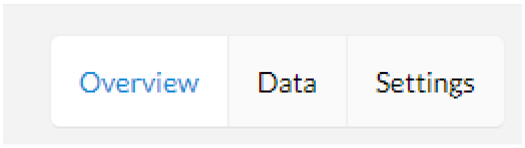
All the transponder projects described here use the "**Activation by Personalization**" (ABP)

For info about the activation methods, see:

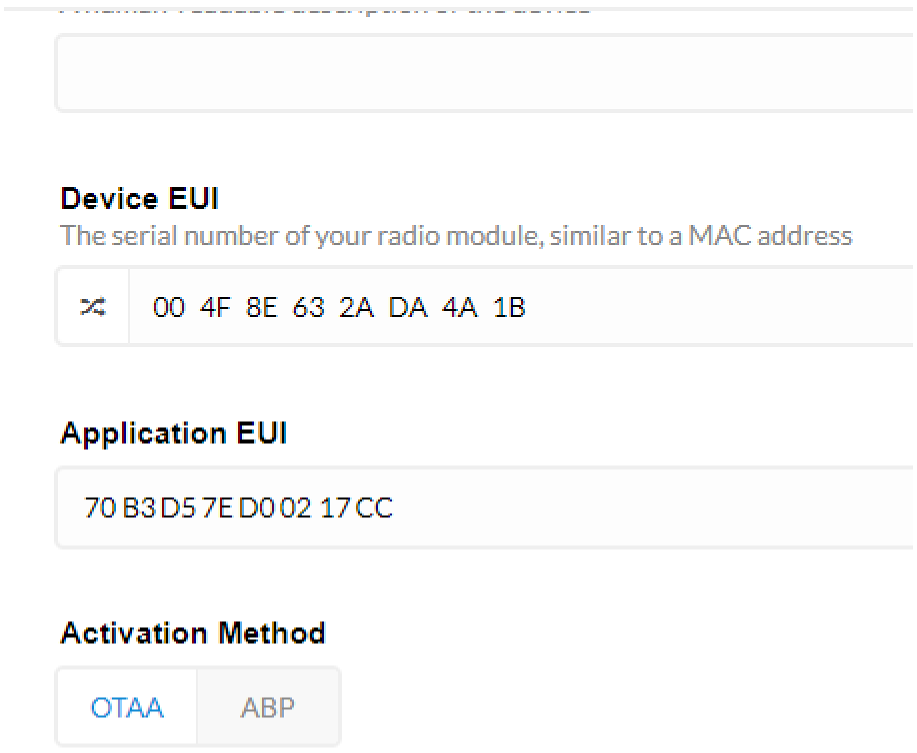
https://www.thethingsnetwork.org/forum/t/what-is-the-diffe-rence-between-otaa-and-abp-devices/2723/2


Activation Method OTAA

To change the type of activation, select **Settings** at the top right.



https://console.thethingsnetwork.org/applications/weather_station_data/devices/tr_5in1_01/settings

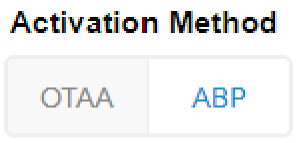


Device EUI
The serial number of your radio module, similar to a MAC address
00 4F 8E 63 2A DA 4A 1B

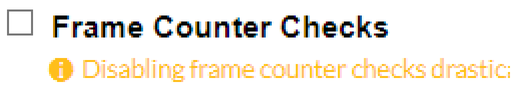
Application EUI
70 B3 D5 7E D0 02 17 CC

Activation Method

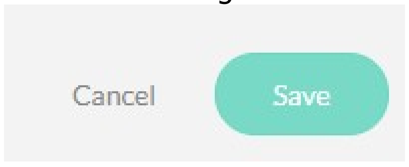
Click on **ABP**



Uncheck **Frame Counter Checks**



Finally, press **Save** at the bottom right.



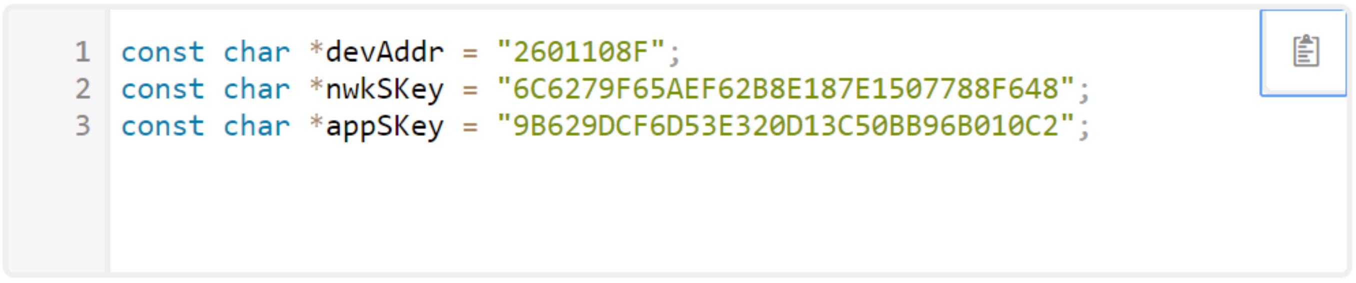
Return to the main page of the device.

https://console.thethingsnetwork.org/applications/weather_station_data/devices/tr_5in1_01

4.1 get the ABP activation keys generated by TTN application

At the bottom of the main page of the device, you will find the ABP activation keys.

https://console.thethingsnetwork.org/applications/weather_station_data/devices/tr_5in1_01



```
const char *devAddr = "2601108F";
const char *nwkSKey = "6C6279F65AEF62B8E187E1507788F648";
const char *appSKey = "9B629DCF6D53E320D13C50BB96B010C2";
```

Press the icon

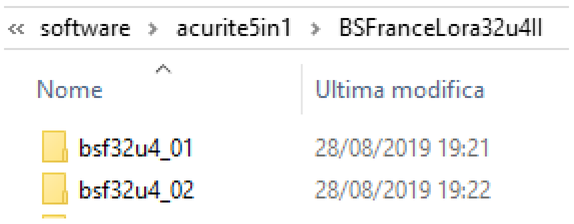
to copy the ABP keys to clipboard:

```
const char *devAddr = "2601108F";
const char *nwkSKey = "6C6279F65AEF62B8E187E1507788F648";
const char *appSKey = "9B629DCF6D53E320D13C50BB96B010C2";
```

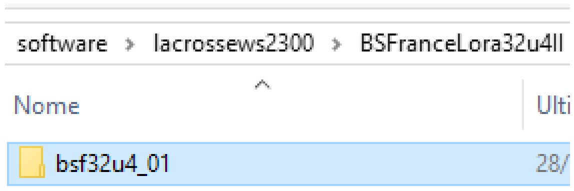
5 Transponder configuration with the ABP activation keys

Transponder project list actually in repository:

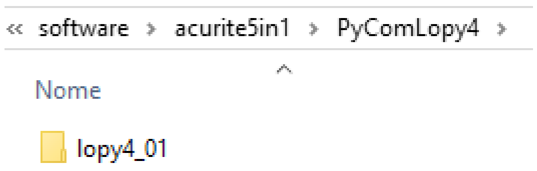
Relative path	**Project Name**	Weather Station	HW Module
software/acurite5in1/BSFranceLora32u4II	**bsf32u4_01**	Acurite 5in1	Lora32u4II
software/acurite5in1/BSFranceLora32u4II	**bsf32u4_02**	Acurite 5in1	Lora32u4II
software/acurite5in1/PyComLopy4	**lopy4_01**	Acurite 5in1	Lopy4
software/lacrossews2300/BSFranceLora32u4II	**bsf32u4_01**	LaCrosse WS2300	Lora32u4II



Nome	Ultima modifica
bsf32u4_01	28/08/2019 19:21
bsf32u4_02	28/08/2019 19:22



bsf32u4_01 28/



lopy4_01

5.1 WS Acurite projects bsf32u4_01, lopy4_01 and LaCrosse bsf32u4_01 project

These three projects are configured in the same way.

For example, in case of **bsf32u4_01** project:

```
If you don't have a TTN account ...
               Version 1.0
```

```
software/acurite5in1/BSFranceLora32u4II
                         ├───bsf32u4_01
                         │   │   bsf32u4_01.h
                         │   │   bsf32u4_01.ino
```

With Arduino ide (or another text editor) open bsf32u4_01.ino file.

Look for the following code section:

```
// -------------------------------------------------------------
// KEYS FOR ABP ACCESS

// Application ID: demo_ttn
// LoRaWAN NwkSKey, network session key
static u1_t NWKSKEY[16] = { 0x00, .. , 0x00 };
// LoRaWAN AppSKey, application session key
static u1_t APPSKEY[16] = { 0x00, .. , 0x00 };
// LoRaWAN end-device address (DevAddr)
static u4_t DEVADDR = 0x00000000;   // Put here the device id in hexadecimal form.
```

convert the ABP application keys previously generated:

```
const char *devAddr = "2601108F";
const char *nwkSKey = "6C6279F65AEF62B8E187E1507788F648";
const char *appSKey = "9B629DCF6D53E320D13C50BB96B010C2";
```

to the format provided by the code c:

```
static u1_t NWKSKEY[16] = {   0x6C, 0x62, 0x79, 0xF6,
                             0x5A, 0xEF, 0x62, 0xB8,
                             0xE1, 0x87, 0xE1, 0x50,
                             0x77, 0x88, 0xF6, 0x48};
static u1_t APPSKEY[16] = {   0x9B, 0x62, 0x9D, 0xCF,
                             0x6D, 0x53, 0xE3, 0x20,
                             0xD1, 0x3C, 0x50, 0xBB,
                             0x96, 0xB0, 0x10, 0xC2};
static u4_t DEVADDR = 0x2601108F;
```

Recompile the project with Arduino IDE and finally transfer the new firmware version to the board.

5.2 WS Acurite project bsf32u4_02 for Lora32u4II module

The firmware of this project, derived from bsf32u4_01, is configured using the **bsfcfg** program executed in linux environment (the operation was verified with the distribution ubuntu 18:04).

The sources are located in the following repository directory:

software/acurite5in1/BSFranceLora32u4II/cfglinux_02

To compile the **bsfcfg** project:

```
~/cfglinux_02$ make clean
rm -f bsfcfg bsf32u4.ini
~/cfglinux_02$ make
gcc -g -I../src -o bsfcfg bsfcfg.c dictionary.c iniparser.c -I./
~/cfglinux_02$
```

To get help, run the command without parameters:

```
~/cfglinux_02$ ./bsfcfg
=====================================================================================
Linux configuration tool for Acurite transponder with BsFrance Lora32u4II.
Version:1.00
Use:
./bsfcfg <serial port name> <configuration filename>
       OR
./bsfcfg <serial port name>
       in this case a default configuration file bsf32u4.ini is automatically
generated.
Example:
./bsfcfg ttyUSB0 ./bsf32u4.ini
where:
ttyUSB0:     USB serial port where the transponder Lora32u4II is connected
bsf32u4.ini: configuration file name

Run this tool in Windows 10 (using WSL, Windows Subsystem for Linux compatibility
layer):
If the program run in a bash shell of windows10, use the WSL name
of the serial to which the Lora32u4II module is connected.
In WSL, the serial port COM<n> port is available at /dev/ttyS<n>
For example, if the board is connected to the windows serial port COM3,
use ttyS3 as serial port name
In bash shell launch the program with these parameters:
./bsfcfg ttyS3 ./bsf32u4.ini
For info, see:
https://blogs.msdn.microsoft.com/wsl/2017/04/14/serial-support-on-the-windows-
subsystem-for-linux/
=====================================================================================
~/cfglinux_02$
```

The parameters are saved in the EEprom memory of the Lora32u4II devi-ce.

An external configuration program for the firmware has the advantage of disconnecting the code from the configuration parameters (it is no longer necessary to recompile the firmware to change important system parameters) and to have a single firmware equal for all units of the same kind.

Of course the main disadvantage is that it increases the code occupation for the addition of configuration management functions.

For example, the starting project bsf32u4_01 generates these messages at the end of the compilation:

```
Sketch uses 25282 bytes (88%) of program storage space. Maximum is 28672 bytes.
Global variables use 1350 bytes of dynamic memory.
```

For the firmware bsf32u4_02, managed by the external configuration program:

```
Sketch uses 28210 bytes (98%) of program storage space. Maximum is 28672 bytes.
Global variables use 1457 bytes of dynamic memory.
```

5.2.1 Configuration variables for bsf32u4_02 firmware

NAME	USE	Note
NWKSKEY	LoRaWAN Network Session Keys	ABP application key1
APPSKEY	LoRaWAN APP Key	ABP application key2
DEVADDR	LoRaWAN end-device address	ABP application key3
TX_INTERVAL	LoRaWAN tx data interval	corresponds to the duration of the operating cycle time in seconds of the transponder (Tattivo + Tsleep).
IdWs	Id Acurite 5in1	identification code of the Acurite 5in1 weather control unit to which the transponder is associated (12bit)
PVarFWdog	+/- max tolerance percentage of the watchdog frequency	Configuration variable used during debugging. Do not change.

5.2.2 Example of configuration file for bsf32u4_02 firmware

```
# LoRaWAN Network Session Keys
NWKSKEY      = "A5EC135CBA37A2C55266E77B0B948982" ;
# LoRaWAN Network Session Keys
APPSKEY      = "7D5821DB815148C2A3139BE3BB0678FA" ;
# LoRaWAN end-device 4 byte hex address (DevAddr)
DEVADDR      = "26011824" ;

#
# This is the base ini file for bsf32u4 lora controller
#
[Lorawan]
```

If you don't have a TTN account ...
Version 1.0

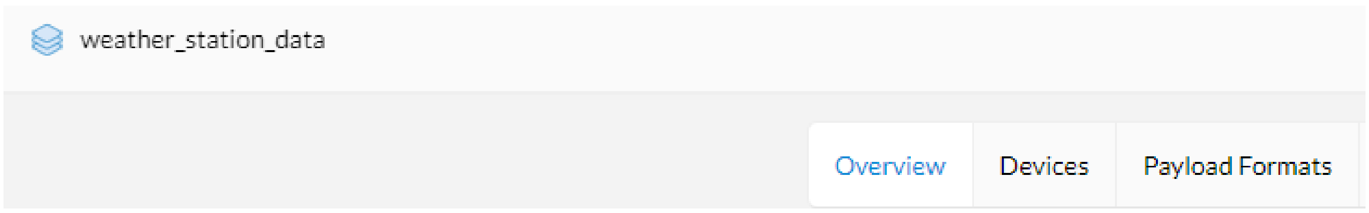
```
# ========================== bsf32u4 lorawan parameters

# LoRaWAN Network Session Keys
NWKSKEY      = "A5EC135CBA37A2C55266E77B0B948982" ;
# LoRaWAN Network Session Keys
APPSKEY      = "7D5821DB815148C2A3139BE3BB0678FA" ;
# LoRaWAN end-device 4 byte hex address (DevAddr)
DEVADDR      = "26011824" ;
# LoRaWAN tx data interval (sec)
TX_INTERVAL  = 300 ;

[Ws]
# ========================== weather station parameters

# Id weather station
IdWs       = "0C24" ;

[Micro]
# ========================== bsf32u4 micro parameters

# +/- max tolerance percentage of the watchdog frequency
PVarFWdog   = 15 ;
```

6 Setting the payload decoding function

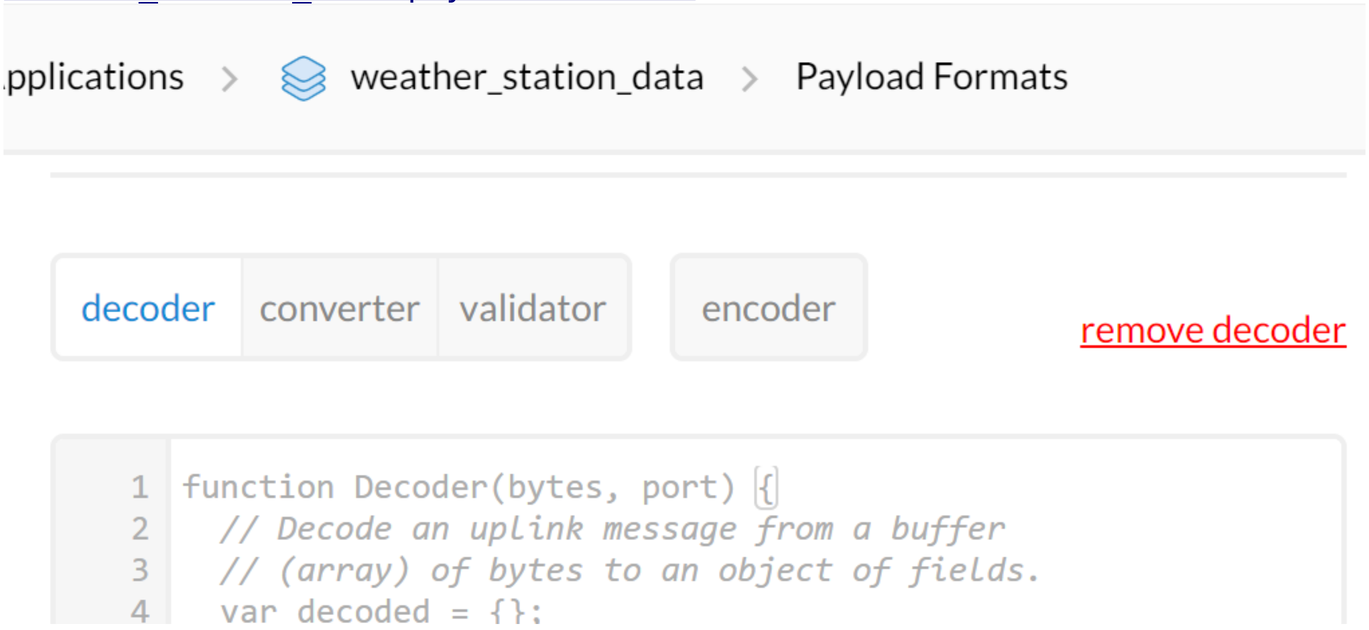
The data sent via LoraWAN by transponder to The Things Network are all in bytes. We need to decode it into a usable values.

Go to the main page of the TTN application.

On the top bar press “Payload Formats”



https://console.thethingsnetwork.org/applications/weather_station_data/payload-formats



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

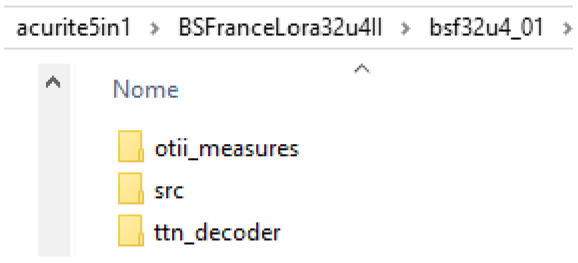
Open the transponder project folder.

Inside it there is the ttn_decoder subfolder:

it contains a text file with the correct decoder function.

Replace it on the web page.

Finally, at the bottom of the page press "Save".


acurite5in1 › BSFranceLora32u4II › bsf32u4_01 ›

Nome

otii_measures
src
ttn_decoder

6.1 Decoder of the transponder messages for Acurite 5in1

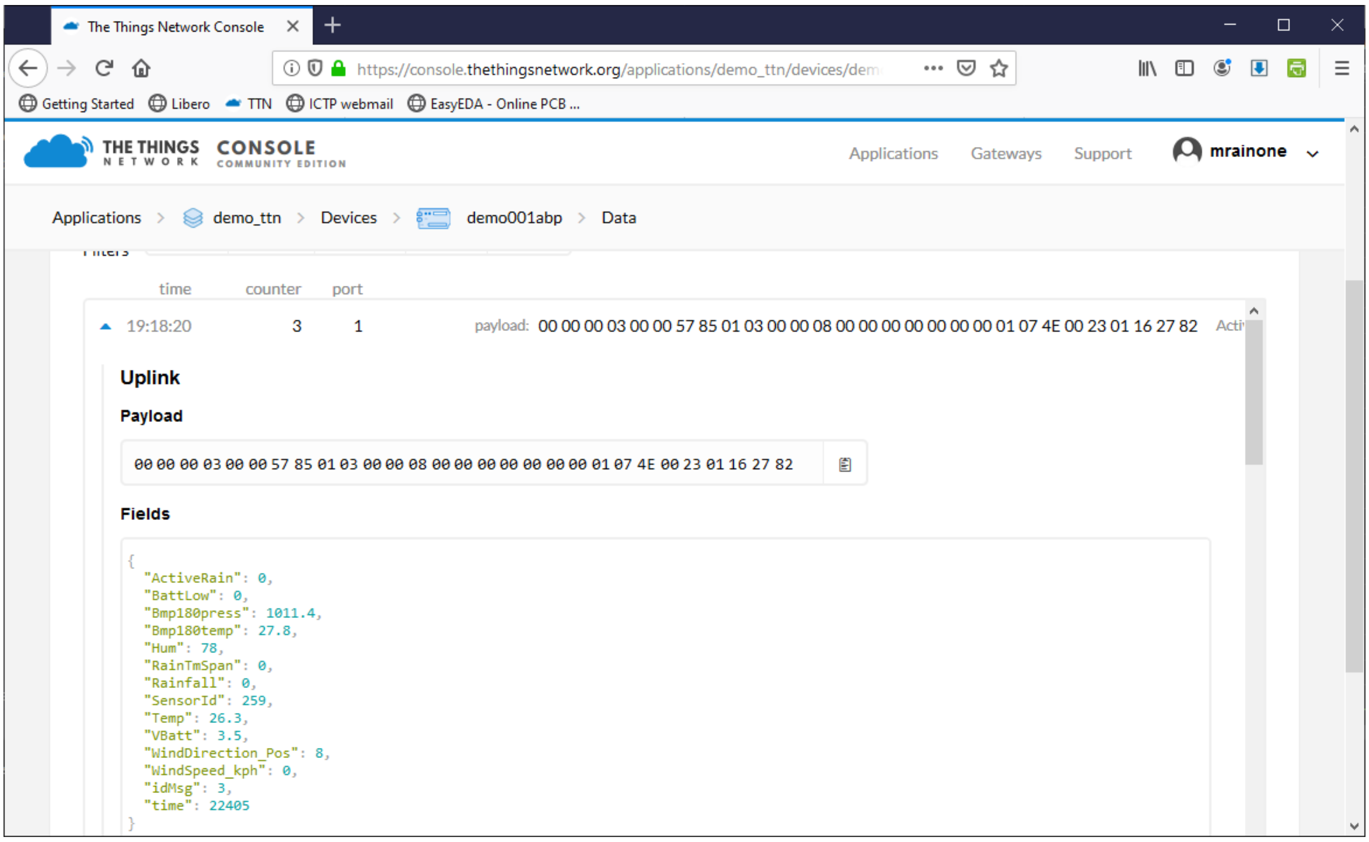
```
function Decoder(bytes, port) {
      var deco = {};
      var wordValue;
      deco.idMsg = (bytes[0]<<24) + (bytes[1]<<16) + (bytes[2]<<8) + (bytes[3]);
      deco.time = (bytes[4]<<24) + (bytes[5]<<16) + (bytes[6]<<8) + (bytes[7]);
      deco.SensorId = (bytes[8]<<8) + (bytes[9]);
      deco.WindSpeed_kph = (bytes[10]<<8) + (bytes[11]);
      deco.WindSpeed_kph = deco.WindSpeed_kph / 10.0;
      deco.WindDirection_Pos = bytes[12] & 0x0F;
      deco.Rainfall = (bytes[13]<<8) + (bytes[14]);
      deco.Rainfall = deco.Rainfall / 100.0;
      deco.ActiveRain = bytes[15] & 0x01;
      deco.RainTmSpan = (bytes[16]<<24) + (bytes[17]<<16) + (bytes[18]<<8) +
(bytes[19]);
      deco.Temp = (bytes[20]<<8) + (bytes[21]);
      var sign = bytes[20] & (1 << 7);
      if(sign)
      {
            deco.Temp = 0xFFFF0000 | deco.Temp;
      }
      deco.Temp = deco.Temp / 10.0;
      deco.Hum = bytes[22];
      if(deco.Hum > 99)
            deco.Hum = 99;
      deco.BattLow = bytes[23] & 0x01;
      deco.VBatt = bytes[24];
      deco.VBatt = deco.VBatt / 10.0;
      deco.Bmp180temp = (bytes[25]<<8) + (bytes[26]);
      deco.Bmp180temp = deco.Bmp180temp / 10.0;
      deco.Bmp180press = (bytes[27]<<8) + (bytes[28]);
      deco.Bmp180press = deco.Bmp180press / 10.0;
      return deco;
}
```

6.2 Decoder of the transponder messages for LaCrosse WS2300

```
function Decoder(bytes, port) {
      var deco = {};
      var wordValue;
      deco.idMsg = (bytes[0]<<24) + (bytes[1]<<16) + (bytes[2]<<8) + (bytes[3]);
      deco.time = (bytes[4]<<24) + (bytes[5]<<16) + (bytes[6]<<8) + (bytes[7]);
      deco.SensorId = (bytes[8]<<8) + (bytes[9]);
      deco.WindSpeed = (bytes[10]<<8) + (bytes[11]);
      deco.WindSpeed = deco.WindSpeed / 10.0;
      deco.WindDirection = bytes[12] & 0x0F;
      deco.GustSpeed = (bytes[13]<<8) + (bytes[14]);
      deco.GustSpeed = deco.GustSpeed / 10.0;
      deco.GustDirection = bytes[15] & 0x0F;
      deco.Rainfall = (bytes[16]<<8) + (bytes[17]);
      deco.Rainfall = deco.Rainfall * 0.5180;
      deco.Temp = (bytes[18]<<8) + (bytes[19]);
      var sign = bytes[18] & (1 << 7);
      if(sign)
      {
            deco.Temp = 0xFFFF0000 | deco.Temp;
      }
      deco.Temp = deco.Temp / 10.0;
      deco.Hum = (bytes[20]<<8) + (bytes[21]);
      if(deco.Hum > 99)
            deco.Hum = 99;
      deco.VBatt = bytes[22];
      deco.VBatt = deco.VBatt / 10.0;
      deco.Bmp180temp = (bytes[23]<<8) + (bytes[24]);
      deco.Bmp180temp = deco.Bmp180temp / 10.0;
      deco.Bmp180press = (bytes[25]<<8) + (bytes[26]);
      deco.Bmp180press = deco.Bmp180press / 10.0;
      return deco;
}
```

7 Example of messages sent on TTN

7.1 Data page transponder for Acurite 5in1



```
time        counter   port
19:18:20       3        1     payload: 00 00 00 03 00 00 57 85 01 03 00 00 08 00 00 00 00 00 00 00 00 01 07 4E 00 23 01 16 27 82   Acti
```

Uplink

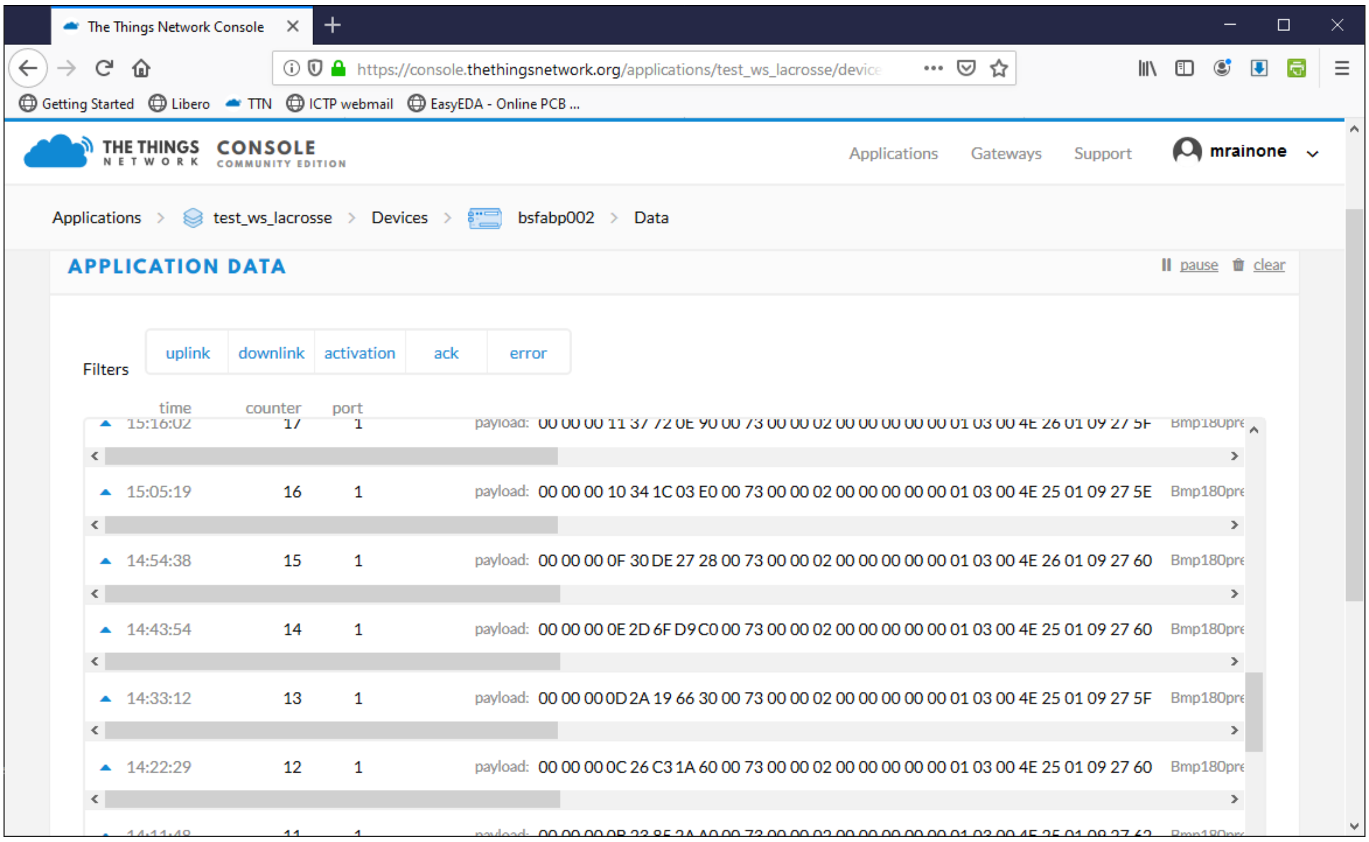
Payload

```
00 00 00 03 00 00 57 85 01 03 00 00 08 00 00 00 00 00 00 00 00 01 07 4E 00 23 01 16 27 82
```

Fields

```
{
  "ActiveRain": 0,
  "BattLow": 0,
  "Bmp180press": 1011.4,
  "Bmp180temp": 27.8,
  "Hum": 78,
  "RainTmSpan": 0,
  "Rainfall": 0,
  "SensorId": 259,
  "Temp": 26.3,
  "VBatt": 3.5,
  "WindDirection_Pos": 8,
  "WindSpeed_kph": 0,
  "idMsg": 3,
  "time": 22405
}
```

7.2 Data page transponder for LaCrosse WS2300


The Things Network Console
https://console.thethingsnetwork.org/applications/test_ws_lacrosse/device
Getting Started · Libero · TTN · ICTP webmail · EasyEDA - Online PCB ...
THE THINGS NETWORK CONSOLE COMMUNITY EDITION · Applications · Gateways · Support · mrainone
Applications > test_ws_lacrosse > Devices > bsfabp002 > Data

APPLICATION DATA · pause · clear

Filters: uplink · downlink · activation · ack · error

```
time      counter  port
15:16:02  17       1     payload: 00 00 00 11 37 72 0E 90 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 26 01 09 27 5F  Bmp180pre
15:05:19  16       1     payload: 00 00 00 10 34 1C 03 E0 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 25 01 09 27 5E  Bmp180pre
14:54:38  15       1     payload: 00 00 00 0F 30 DE 27 28 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 26 01 09 27 60  Bmp180pre
14:43:54  14       1     payload: 00 00 00 0E 2D 6F D9 C0 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 25 01 09 27 60  Bmp180pre
14:33:12  13       1     payload: 00 00 00 0D 2A 19 66 30 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 25 01 09 27 5F  Bmp180pre
14:22:29  12       1     payload: 00 00 00 0C 26 C3 1A 60 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 25 01 09 27 60  Bmp180pre
14:11:48  11       1     payload: 00 00 00 0B 23 8F 2A A0 00 73 00 00 02 00 00 00 00 00 01 03 00 4E 25 01 09 27 62  Bmp180pre
```

15:05:19161

Payload

00000010341C03E00073000002000000000001030004E250109275E

Fields

```
{
  "Bmp180press": 1007.8,
  "Bmp180temp": 26.5,
  "GustDirection": 0,
  "GustSpeed": 0,
  "Hum": 78,
  "Rainfall": 0,
  "SensorId": 115,
  "Temp": 25.9,
  "VBatt": 3.7,
  "WindDirection": 2,
  "WindSpeed": 0,
  "idMsg": 16,
  "time": 874251232
}
```