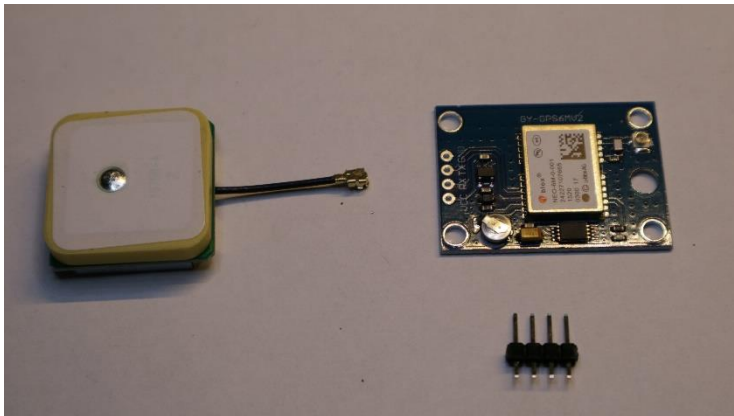


Building the Node

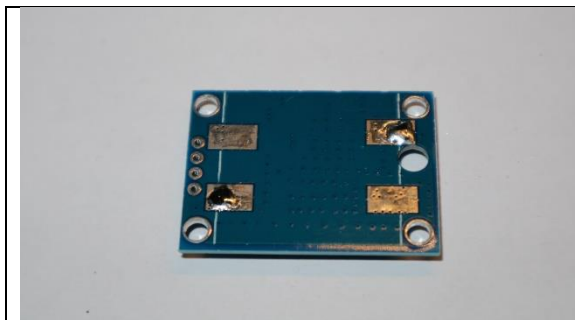
Follow the instructions on <https://www.thethingsnetwork.org/labs/story/creating-a-ttn-node> . We will start these instructions with a working node on the TTN network.

Attach the GPS module

Most of the GPS modules can be used. Important: 3v3 power and signals, 9600 baud, Rx and Tx connection. Instructions describe the gy-neo6mv2 module.



This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.



The ceramic antenna can be soldered on the back side of the PCB. Add some solder on two opposite corners and melt it while pressing the module on the PCB.



This is giving a compact design



Gently press the antenna onto the connector, and add the 4-pin header to the module

Connect the GPS module

Use 4 female-female Dupont wires to connect the GPS module to your Node.

GPS Module	TTN Node
Vcc	3v3
Rx	Pin 9
Tx	Pin 8
GND	GND

Testing the GPS Module

1. First install the needed TinyGPS library. It can be downloaded from <https://github.com/mikalhart/TinyGPSPlus/releases> , download the latest version (0.95) as a ZIP file. Download the LMIC library from Github:
2. Choose 'Clone or download'
3. Download ZIP
4. Mark the location
5. Go to Arduino IDE
6. Select 'Sketch->Include Library->Add .ZIP Library'
7. Select the downloaded Arduino-lmic-master.zip file from your download location

If you did not download the LMIC library in the labs story 'Creating a TTN node', you should do this here as well.

1. Download the LMIC library from Github:
2. Goto <https://github.com/matthijskooijman/arduino-lmic>
3. Choose 'Clone or download'
4. Download ZIP
5. Mark the location
6. Go to Arduino IDE
7. Select 'Sketch->Include Library->Add .ZIP Library'
8. Select the downloaded Arduino-lmic-master.zip file from your download location

Upload the sketch

To test the GPS go to File->Examples->TinyGPSPlus-Master->FullExample

Change the following lines:

```
static const int RXPin = 8, TXPin = 9;
static const uint32_t GPSBaud = 9600;
```

RX is connected to TX and TX is connected to RX of the GPS module. The GPS node is sending on TX, the node is receiving on RX.

Now upload the sketch.

Check the monitor to see if it is working. Remember that the GPS module should have some 'line of sight' to four satellites. After a (very) cold start, it could last several minutes before a 'lock'. It might help to move the GPS module outside or close to a window. After a 'lock' the position will be less critical.

```
4 307 51.4 19 655 11/21/2017 15:39:28 765 1.60 0.00 3.89 N 388 272.54 W 12852 26 0
4 307 51.4 39 676 11/21/2017 15:39:29 788 1.00 0.00 4.39 N 388 272.54 W 12900 28 0
4 307 51.4 29 690 11/21/2017 15:39:30 803 1.80 0.00 3.04 N 388 272.54 W 12948 30 0
4 307 51.4 34 709 11/21/2017 15:39:31 821 3.20 0.00 3.37 N 388 272.54 W 12996 32 0
4 307 51.4 22 723 11/21/2017 15:39:32 834 4.00 0.00 2.26 N 388 272.54 W 13045 34 0
4 307 51.4 32 745 11/21/2017 15:39:33 858 5.10 0.00 5.48 N 388 272.54 W 13093 36 0
4 307 51.4 37 762 11/21/2017 15:39:34 874 5.30 0.00 4.78 N 388 272.54 W 13141 38 0
4 307 51.4 16 782 11/21/2017 15:39:35 895 6.30 0.00 2.52 N 388 272.54 W 13189 40 0
4 307 51.4 89 795 11/21/2017 15:39:36 907 5.60 0.00 1.81 N 388 272.54 W 13237 42 0
4 307 51.4 75 826 11/21/2017 15:39:37 975 4.20 0.00 0.87 N 388 272.54 W 13334 46 0
4 307 51.4 29 326 11/21/2017 15:39:39 438 2.70 0.00 2.30 N 388 272.54 W 13382 48 0
4 307 51.4 96 346 11/21/2017 15:39:40 459 2.50 0.00 4.02 N 388 272.54 W 13430 50 0
4 307 51.4 05 364 11/21/2017 15:39:41 477 0.20 0.00 2.09 N 388 272.54 W 13478 52 0
4 307 51.4 98 381 11/21/2017 15:39:42 493 -1.50 0.00 3.11 N 388 272.54 W 13527 54 0
4 307 51.4 07 399 11/21/2017 15:39:43 512 -2.90 0.00 1.50 N 388 272.54 W 13575 56 0
4 306 51.4 05 414 11/21/2017 15:39:44 524 -4.20 0.00 1.22 N 388 272.54 W 13623 58 0
4 306 51.4 01 433 11/21/2017 15:39:45 545 -1.30 0.00 1.88 N 388 272.54 W 13670 60 0
```

The Things Network Dashboard

Your applications and devices can be managed by The Things Network Dashboard.

Create an Account

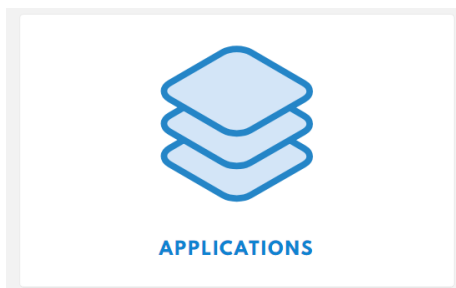
To use the dashboard, you need a The Things Network account. You can create an account here:

<https://account.thethingsnetwork.org/users/login> .

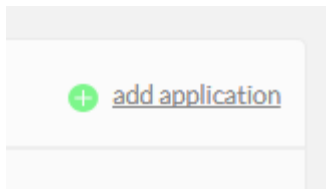
After registering and validating your email address, you will be able to log in to The Things Network Dashboard.

Application

We log in in the console and create a new node. Because we will use a different payload format, we will create a new application as well.



And create an application



Give your application a name: 'gps_nodes' (our choose your own):

Applications > Add Application

ADD APPLICATION

Application ID
The unique identifier of your application on the network

gps_nodes

Description
A human readable description of your new app

My GPS nodes

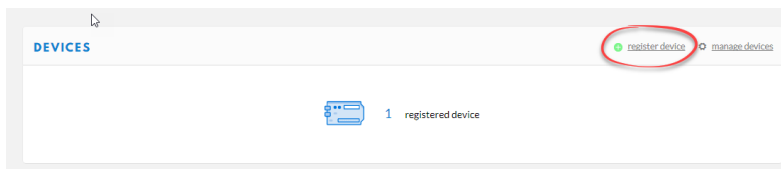
Application EUI
An application EUI will be issued for The Things Network block for convenience, you can add your own ir

EUI issued by The Things Network

Handler registration
Select the handler you want to register this application to

ttn-handler-eu

Now we can register our device:



Overview

Devices

Payload Formats

Integrations

Data

Settings

REGISTER DEVICE

[bulk import devices](#)

Device ID

This is the unique identifier for the device in this app. The device ID will be immutable.

gps_02

Device EUI

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

this field will be generated

App Key

The App Key will be used to secure the communication between you device and the network.

this field will be generated

App EUI

70 B3 D5 7E F0

Cancel

Register

Device ID is a lower-case identifier for your device (gps_01, gps_02), choose to generate the Device ID by choosing the pencil. The field will show: 'this field will be generated'. Choose 'Register' to register your device.

Now the parameters of the device are shown:

DEVICE OVERVIEW

Application ID

gps_nodes

Device ID

gps_02

Activation Method

OTAA

Device EUI

<> 00 59 B0 D9 77

Application EUI

<> 70 B3 D5 7E F0

App Key

<>

Status

never seen

Frames up

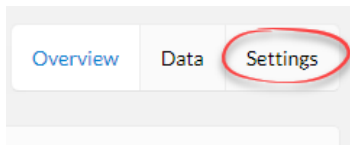
0

reset frame counters

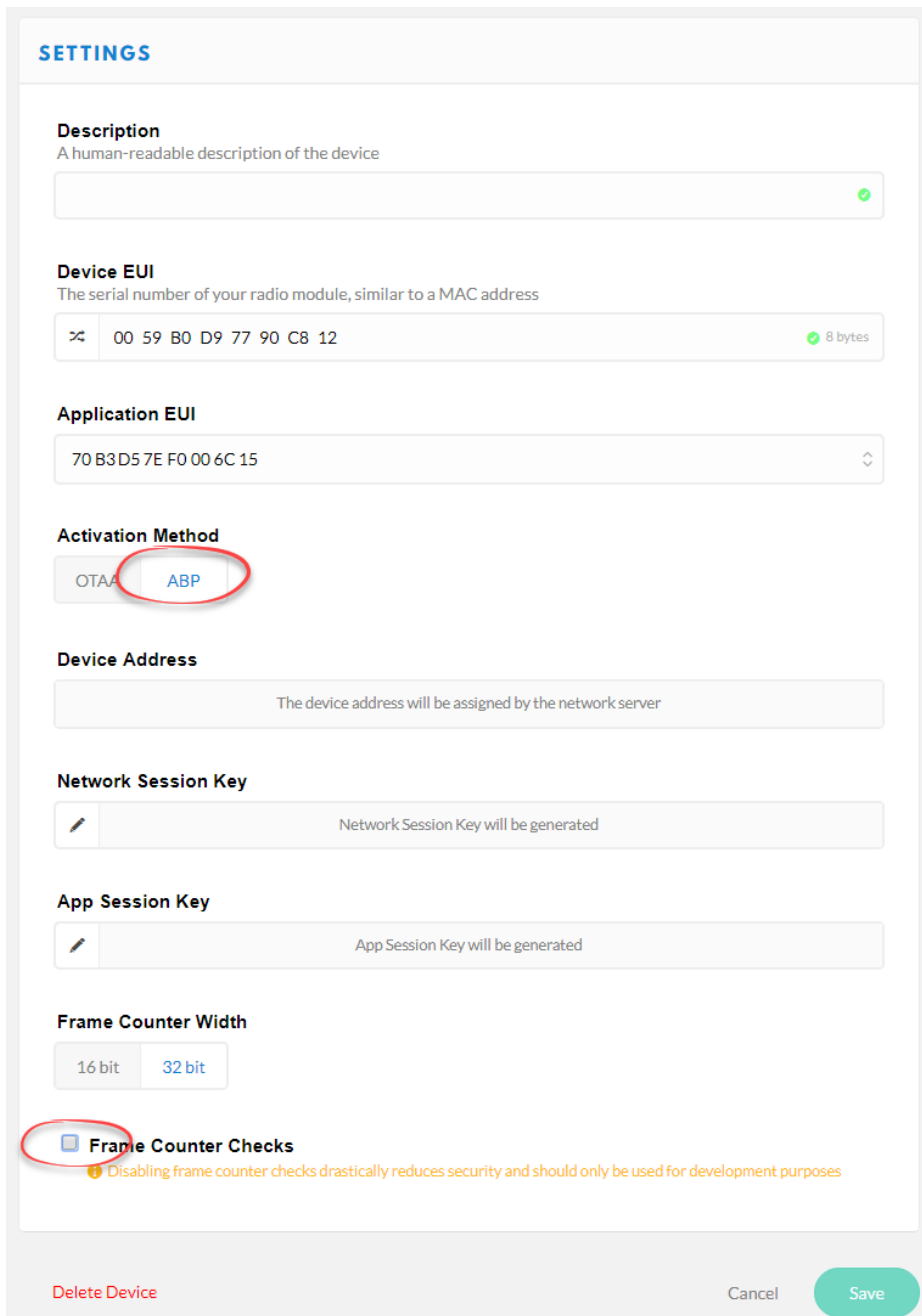
Frames down

0

The standard activation method will be OTAA. This is the preferable setting. But in experimental setups (like workshops), ABP authentication will be more 'hassle free'. You might change these settings later on if you are familiar with them. Now we will choose 'settings':



Here we have the possibility to change the authentication method:



SETTINGS

Description
A human-readable description of the device

Device EUI
The serial number of your radio module, similar to a MAC address

00 59 B0 D9 77 90 C8 12 8 bytes

Application EUI
70 B3D57E F0 00 6C 15

Activation Method
OTAA ABP

Device Address
The device address will be assigned by the network server

Network Session Key
Network Session Key will be generated

App Session Key
App Session Key will be generated

Frame Counter Width
16 bit 32 bit

☒ **Frame Counter Checks**
Disabling frame counter checks drastically reduces security and should only be used for development purposes

Delete Device Cancel Save

Choose ABP and disable 'Frame Counter Checks' as stated, this reduces security. So only for development purposes! Save your settings.

Setup the Arduino Sketch

Download the 'ttn_gps_workshop' sketch from https://github.com/galagaking/GPS_node and open it in your Arduino Console

Copy the keys and ID to your Arduino Code:

DEVICE OVERVIEW

Application ID `gps_nodes`

Device ID `gps_02`

Activation Method `ABP`

Device EUI `<> 00 59 B0 D9`

Application EUI `<> lsb { 0x00, 0xF0, 0x7E, 0xD5, 0xB3, 0x70 }`

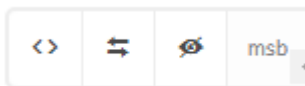
Device Address `<> 26 01`

Network Session Key `<> msb { 0xD5, 0x3C, 0xEB, 0x28, 0xFD, 0xB4, 0xB6, 0xD6, 0xA0, 0xDD, 0x95, 0xB6, 0x...`

App Session Key `<> msb { 0xA4, 0xD8, 0x31, 0xA4, 0xE6, 0x0A, 0x5C, 0x37, 0xF2, 0xA6, 0x03, 0x62, 0xF2, 0x...`

Status ● never seen

You can use these symbols to change: LSB/MSB (<>), hex or array code (arrows) and view (the eye).



The keys should be in MSB format. You can use the copy symbol to copy a key to the clip board:



Add the keys in the example:

```
static const PROGMEM u1_t NWKSKY[16] = { 0xD5, 0x3C, 0xEB, 0x28, 0xFD, 0xB4, 0xB6, 0xD6, 0xA0, 0xDD, 0x95, 0xB6, 0x...  
static const PROGMEM u1_t APPSKY[16] = { 0xA4, 0xD8, 0x31, 0xA4, 0xE6, 0x0A, 0x5C, 0x37, 0xF2, 0xA6, 0x03, 0x62, 0xF2, 0x...  
static const u4_t DEVADDR = 0x26011...; // <-- Change this address for
```

The DEVADDR should be starting with 0x.

LMIC Config

Using the BMP280 together with LMIC/TTN and a GPS results in a too large program to fit into the small (32k) Arduino Pro Memory. We can set some options in LMIC to compress the code a little (28K). Locate the `\lmic\config.h` file and change the following defines:

```
//#define LMIC_DEBUG_LEVEL 2  
//#define LMIC_PRINTF_TO Serial
```

```
//#define LMIC_FAILURE_TO Serial
#define DISABLE_PING
#define DISABLE_BEACONS
```

If you only use 'ABP' (and not 'OTAA') you can even #define DISABLE_JOIN.

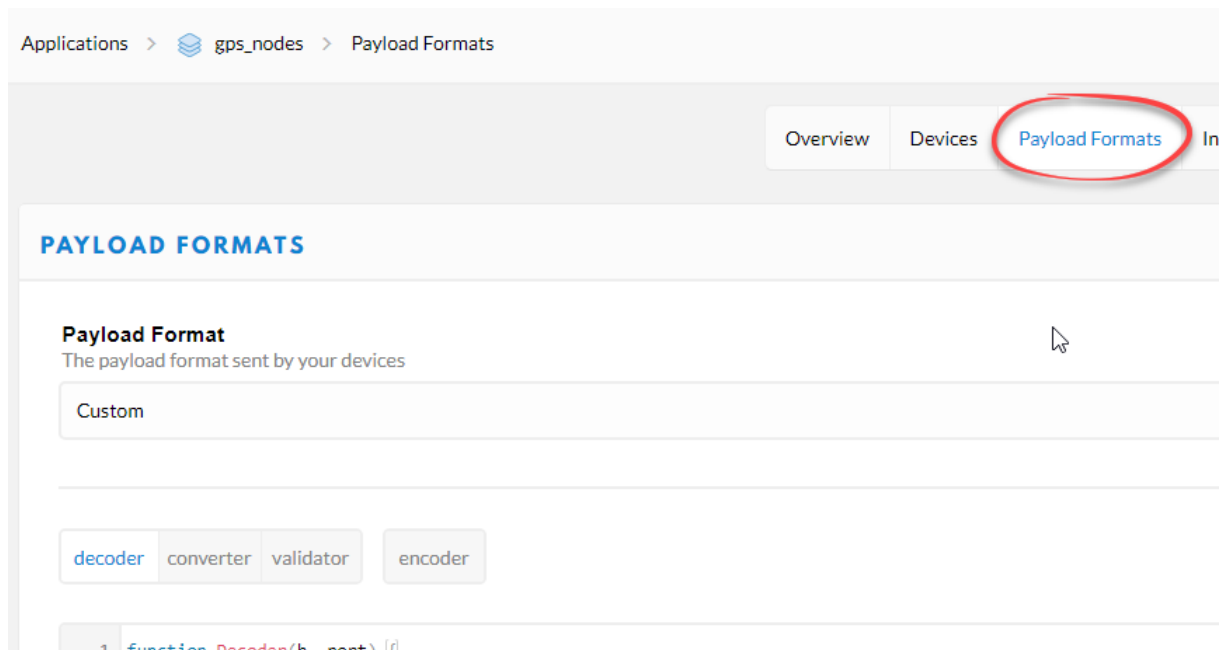
```
23 // Set this to 1 to enable some basic debug output (using printf) about
24 // RF settings used during transmission and reception. Set to 2 to
25 // enable more verbose output. Make sure that printf is actually
26 // configured (e.g. on AVR it is not by default), otherwise using it can
27 // cause crashing.
28 // #define LMIC_DEBUG_LEVEL 2
29
30 // Enable this to allow using printf() to print to the given serial port
31 // (or any other Print object). This can be easy for debugging. The
32 // current implementation only works on AVR, though.
33 // #define LMIC_PRINTF_TO Serial
34
35 // Any runtime assertion failures are printed to this serial port (or
36 // any other Print object). If this is unset, any failures just silently
37 // halt execution.
38 // #define LMIC_FAILURE_TO Serial
39
40 // Uncomment this to disable all code related to joining
41 // #define DISABLE_JOIN
42 // Uncomment this to disable all code related to ping
43 #define DISABLE_PING
44 // Uncomment this to disable all code related to beacon tracking.
45 // Requires ping to be disabled too
46 #define DISABLE_BEACONS
--
```

Save the file. If you compiled your code before, you might have to restart Arduino IDE to effectuate these settings.

Payload function

We have to send the data in encrypted form, two bytes for the temperature and pressure of the sensor, and 6 bytes for the location. Refer to [this](#) link for more info on sending GPS data. Each coordinate is converted to 3 bytes in the code, so 6 bytes in total. To decode this, we use the 'payload' function of the application.

Select your application and select 'payload formats', 'custom':



Paste the following code over the default decoder function:

```
function Decoder(b, port) {  
  // Decode an uplink message from a buffer  
  // (array) of bytes to an object of fields.  
  var lat = ((b[0]<<16)>>>0) + ((b[1]<<8)>>>0) + b[2];  
  lat = (lat / 16777215.0 * 180) - 90;  
  var lon = ((b[3]<<16)>>>0) + ((b[4]<<8)>>>0) + b[5];  
  lon = (lon / 16777215.0 * 360) - 180;  
  var altValue = ((b[6]<<8)>>>0) + b[7];  
  var sign = b[6] & (1 << 7);  
  var alt  
  if(sign)  
  {  
    alt = 0xFFFF0000 | altValue;  
  }  
  else  
  {  
    alt = altValue;  
  }  
  var hdop = b[8] / 10.0;  
  var airpressure = 970+((b[9] >> 2) & 0x3F);  
  var temperature = (-2400+6.25*((b[10] & 0x03) << 8) | b[9]))/100.0;  
  return {  
    lat: lat,  
    lon: lon,  
    alt: alt,  
    hdop:hdop,  
    airpressure: airpressure,  
    temperature: temperature  
  };  
}
```

If we power on our node, we will see values coming in to the console:

```

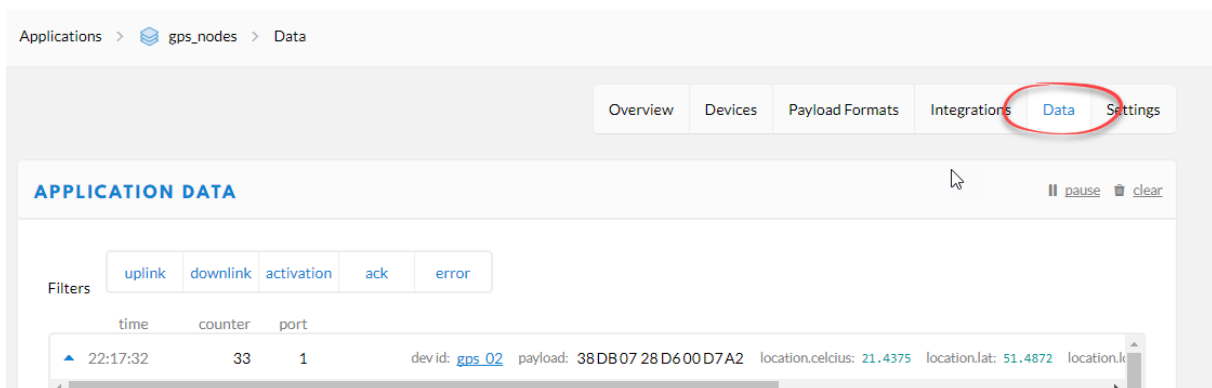
Starting
Probe BMP280: Sensor found
Location: █.487159,█.482266 Date/Time: 11-11-2017 22:44:20.00
Pressure: 1009.79 Pa; T: 19.01 C
Packet queued
411407: EV_TXCOMPLETE (includes waiting for RX windows)
Location: █.487178,█.482280 Date/Time: 11-11-2017 22:45:27.00
Pressure: 1009.79 Pa; T: 18.95 C
Packet queued
4636924: EV_TXCOMPLETE (includes waiting for RX windows)
Location: █.487213,█.482261 Date/Time: 11-11-2017 22:46:34.00
Pressure: 1009.87 Pa; T: 19.01 C
Packet queued
8851630: EV_TXCOMPLETE (includes waiting for RX windows)
Location: █.487197,█.482276 Date/Time: 11-11-2017 22:47:42.00
Pressure: 1009.82 Pa; T: 19.02 C
Packet queued
13076715: EV_TXCOMPLETE (includes waiting for RX windows)
Location: █.487171,█.482244 Date/Time: 11-11-2017 22:48:50.00
Pressure: 1009.85 Pa; T: 19.01 C

```

If the BME280 sensor is not connected, pressure and temperature values are not shown and two zero fields are send to TTN.

Showing data in the TTN Console

The node is sending every minute it's position to TTN. We can show the data in the console. In the application 'gps_nodes' select 'Data'. You can see the original payload (8 bytes) and the converted parameters.



If it does not work...

- Arduino running? Start with the 'Blink' sketch: get the LED blinking.
- Upload not possible? Check the right node in the Arduino IDE (Pro Mini, 3.3v 8Mhz), check the COM port, check the FTDI voltage (should be 3.3v), check the FTDI connection
- GPS running? Remove the 'comment' slashes to display the serial output of the GPS module. You can even directly connect the GPS to the FTDI, GND-GND, VCC-3v3, Rx-Tx, Tx-Rx. If you open the serial monitor at 9600 bps data should be appear.
- GPS not getting a lock? The LED should blink in a 1 sec frequency. If not, put it near a window or outside to get a lock.
- Voltage correct? Check the voltage on the '3.3v' power pin. It should be 3.3v
- Sensor not working? Check your I2C wiring, A4 and A5 should be connected (these are the two inner pins of the Arduino Pro Mini).

- TTN data not coming in? Check your NWSKEY, APSKEY and Device Address. Check the log files (if possible) of your gateway. Check your antenna. For extra debug options, you have to start with the basic ABP sketch
https://github.com/galagaking/GPS_node/blob/master/ttn_gps_no_sensors.ino . This sketch will sent some coordinates every minute, so the decode function will work. (Change 'config.h in the LMIC directory of the LMIC library to get more debug output) (LMIC_DEBUG_LEVEL=2, LMIC_PRINTF_TO Serial):

```

23 // Set this to 1 to enable some basic debug output (using printf) about
24 // RF settings used during transmission and reception. Set to 2 to
25 // enable more verbose output. Make sure that printf is actually
26 // configured (e.g. on AVR it is not by default), otherwise using it can
27 // cause crashing.
28 #define LMIC_DEBUG_LEVEL 2
29
30 // Enable this to allow using printf() to print to the given serial port
31 // (or any other Print object). This can be easy for debugging. The
32 // current implementation only works on AVR, though.
33 #define LMIC_PRINTF_TO Serial
34
35 // Any runtime assertion failures are printed to this serial port (or
36 // any other Print object). If this is unset, any failures just silently
37 // halt execution.
38 #define LMIC_FAILURE_TO Serial
39

```

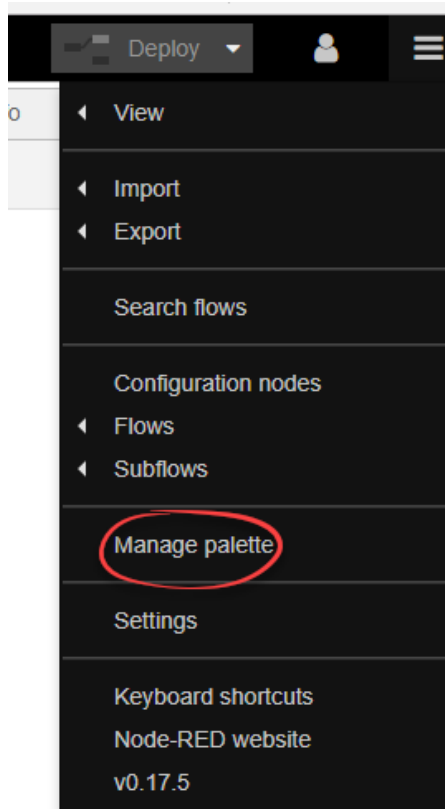
- If there are any 'LMIC' errors there is something wrong with the RFM95 connection, check your soldering.

Showing your data on a 'map'

We will use Node-Red to show the data on a map. You can use a 'local' node-red instance or a Cloud based version (like IBM Cloud or 'FRED').

- IBM Cloud (formerly known as IBM Bluemix) Node-Red instance installed follow the instructions on: <https://nodered.org/docs/platforms/bluemix>

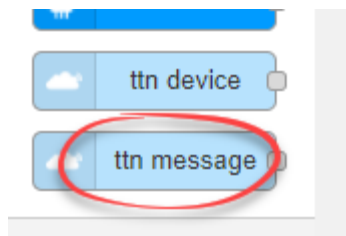
We have to install two modules, Worldmap (node-red-contrib-web-worldmap) and TTN (node-red-contrib-ttn). You can install these modules by choosing 'manage palette' in the main menu:



Here you can add the two modules by looking them up in the library.

Add a TTN node

Add a 'TTN message' to your canvas.



Edit the properties of your node:

Edit ttn message node

Delete Cancel Done

▼ node properties

Name GPS

App gps_nodes

Device ID gps_02

Field

Choose the 'pencil':

ttn message > **Edit ttn app node**

Delete Cancel Update

App ID gps_nodes

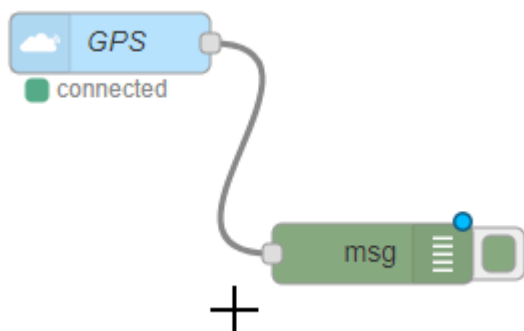
Region or Broker eu

Access Key

Fill in your previous defined 'App ID', select 'EU' as your region (**this is already stated in gray, but you have to fill in the field!**), and copy-paste your access key from your application. Choose 'Update'.

Now you can fill in your Device ID ('gps_01').

To check your settings, connect a debug node to your Node.



The GPS node should say 'connected', as there is a connection with TTN. If not: check the settings above. Your debug screen should show the values after you choose 'Deploy' to run your sketch.

We will add a 'JSON' function to covert the JSON values to a NodeRed Payload Message.

After this JSON node we will add a 'Convert Payload' function:

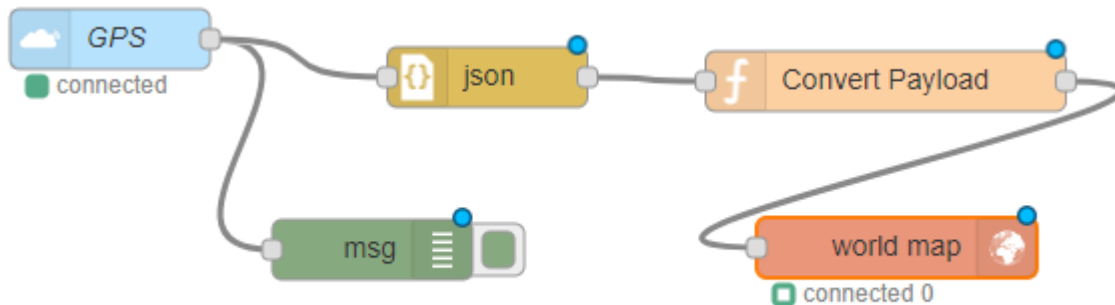
```

var msg1 = {};
msg1.payload = JSON.parse(msg.payload);
msg1.payload.icon = "map-pin";
msg1.payload.name = msg.dev_id;
msg1.payload.iconColor = "orange";
msg1.payload.date=Date().toString();
return msg1;

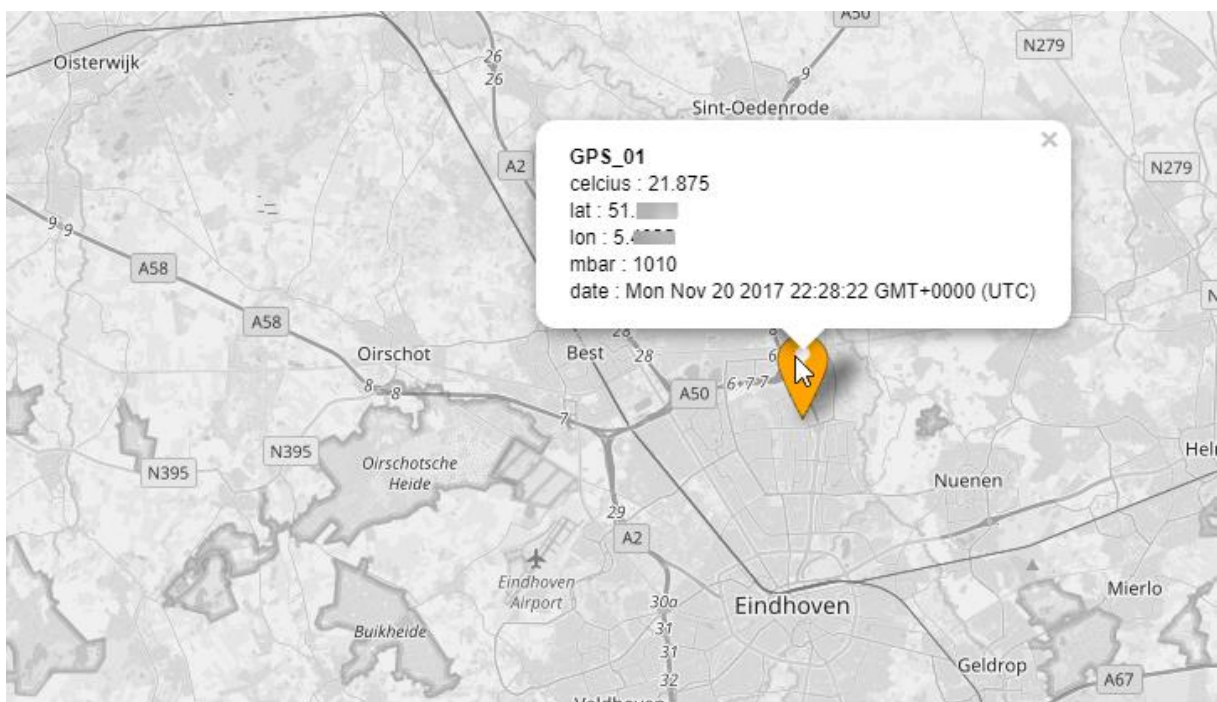
```

This function will put some additional values (name, date, and type/colour of the marker) in to the object.

At the end we will add the 'Worldmap' node to display the information. Your sketch will look like this:



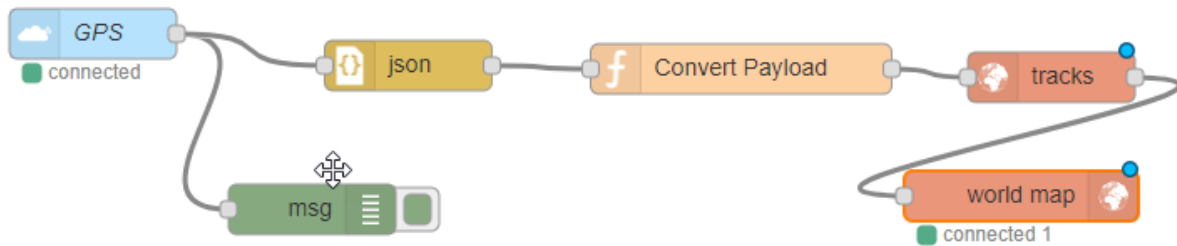
Now we can deploy the sketch. Wait to get the data coming in and with CTRL-SHIFT-M we will go to the MAP:



If we go to the 'orange' point we will see the data from our node (temperature and pressure).

Tracking

If you want to keep track of your nodes, you can add the 'track' node before 'worldmap'



TTN Mapper

TTN Mapper is a place where you can view the coverage of The Things Network. You can add your own measurements with a smartphone and an app, or you can use a GPS node like the one we build.



The code of the GPS is 'TTN mapper' compatible if you remove the temperature sensor. Use the following steps (from www.ttnmapper.org):

Using a node with a GPS, transmitting GPS coordinates (like a Sodaq One)

I will listen for your node on MQTT and import measurement as I receive packets from your device. Your node needs to at least transmit latitude, longitude, altitude and hdop. Then send me (blog@jpmeijers.com) your:

- Application ID (MQTT username)
- Access Key (MQTT password)
- Device ID (or tell me if I should listen to all devices in your application)
- Region (e.g. eu). Region is the last part of the handler you registered your application to.
- A description of your payload format. You can follow the same payload format as the Sodaq One universal tracker, or use my even more compressed format which is used [in this example](#).

-> The code of your gps node is already 'ttnmapper compatible'!

Credits

- <https://www.thethingsnetwork.org/u/BjoernA>
- <http://jpmeijers.com/>