Building a TTN node

Identifying the components

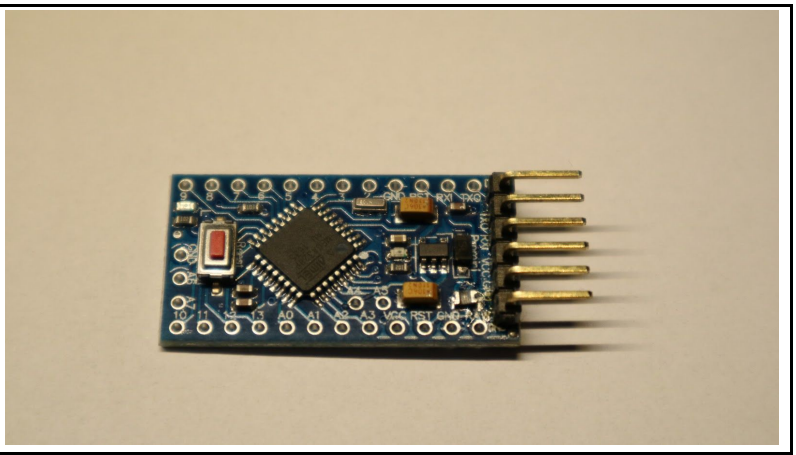
Our bill of materials (set available at TinyTronics)

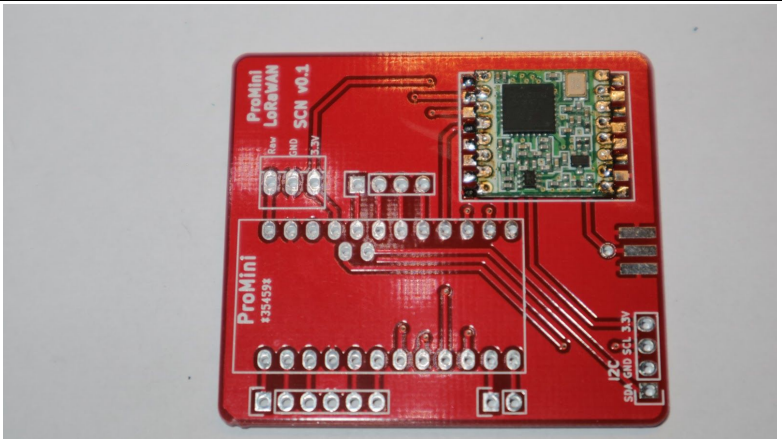
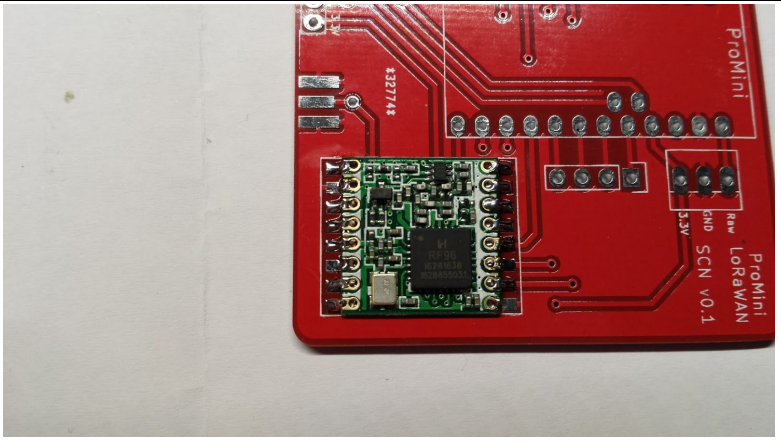
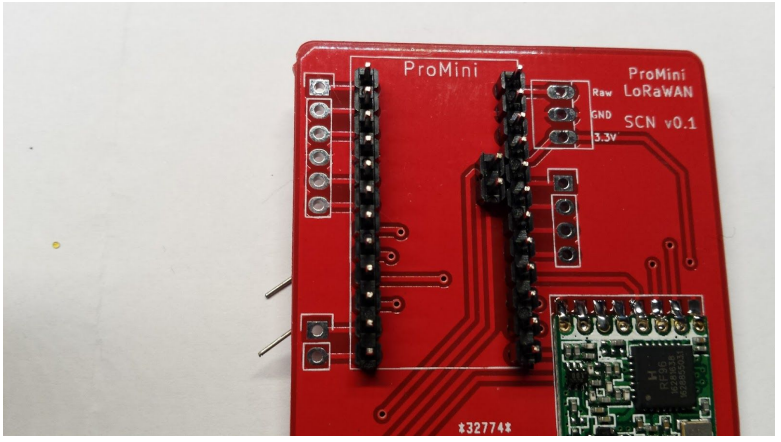
- RFM Module RFM95W (868 Mhz)
- Pro Mini 3.3V 8Mhz (5V will NOT work!!)
- Wemos CH340 3.3v-5v TTL USB Serial Port Adapter (has to have a 3.3V option!)
- USB cable for FTDI
- Led 5mm red
- resistor 1k (brown-black-red)
- Header 40p divided in: 2x 4p, 1x 6p, 1x 3p and 1x 2p
- PCB 'DougLarue'
- Sensor BMP280 (Temperature and air pressure)
- Battery holder 2x AA
- 2 AA batteries
- 2x female headers 2p (battery and led connection)

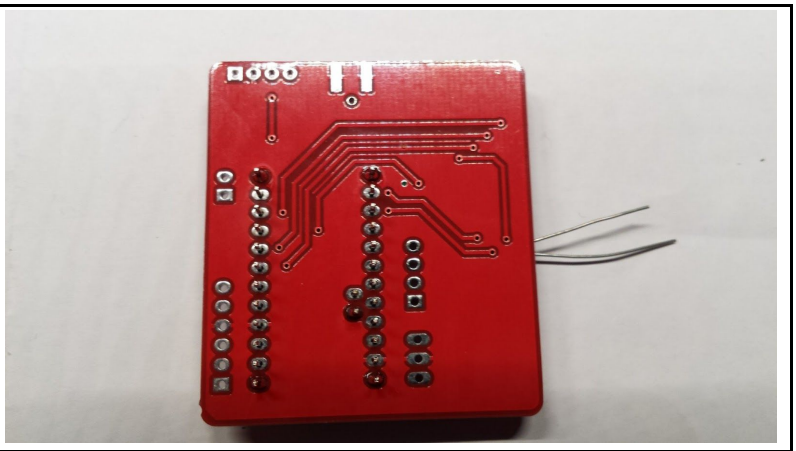
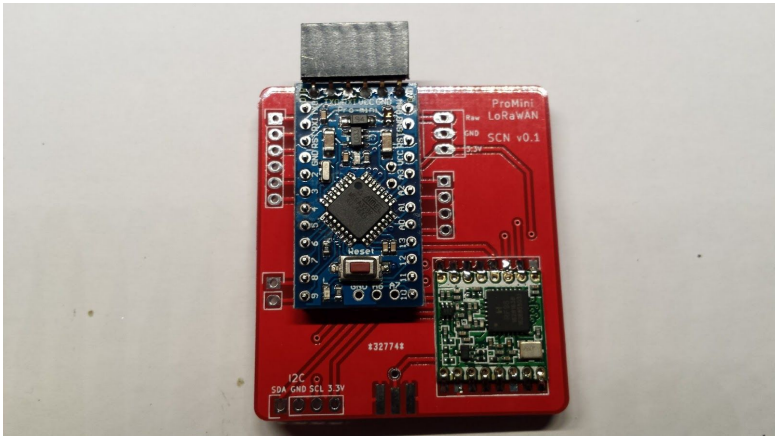
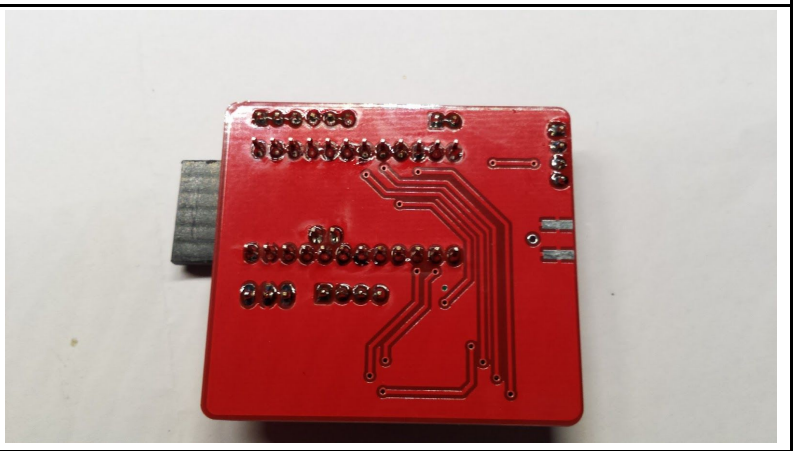
Low Power the Arduino?

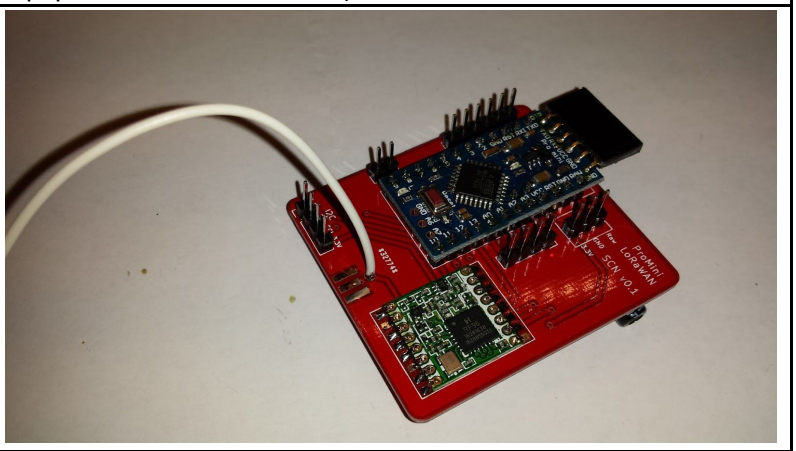
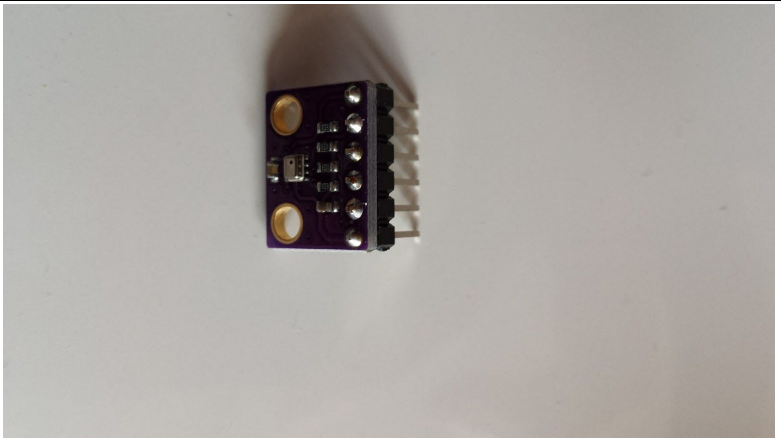
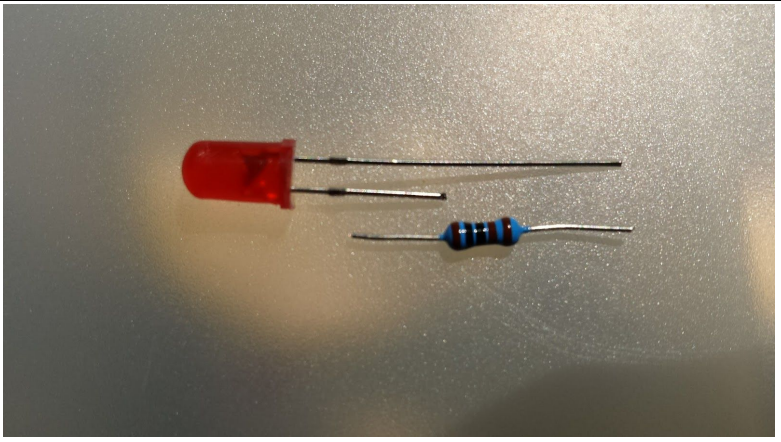
There is an option to 'low power' the Arduino. BUT this requires some precise soldering! The setup will work without removing the components. If you want to low power your node, look at https://andreasrohner.at/posts/Electronics/How-to-modify-an-Arduino-Pro-Mini-clone-for-low-power-consumption/ for more information.

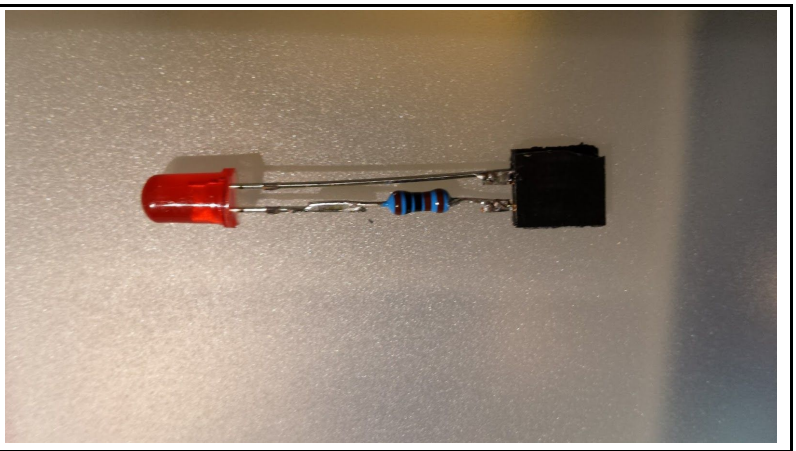
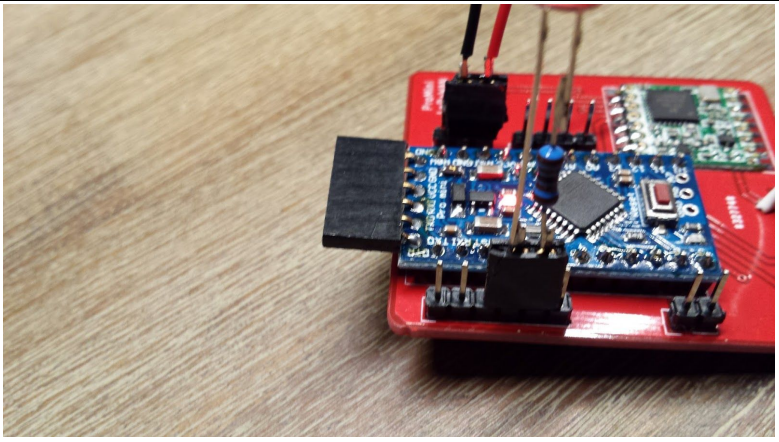
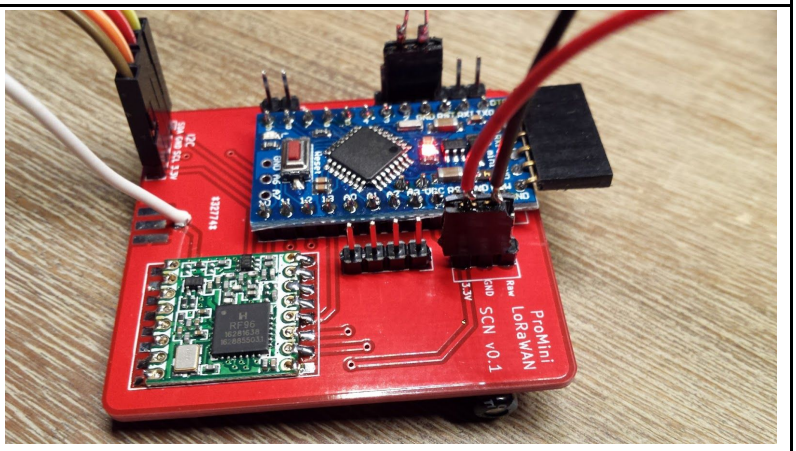
Building the Node

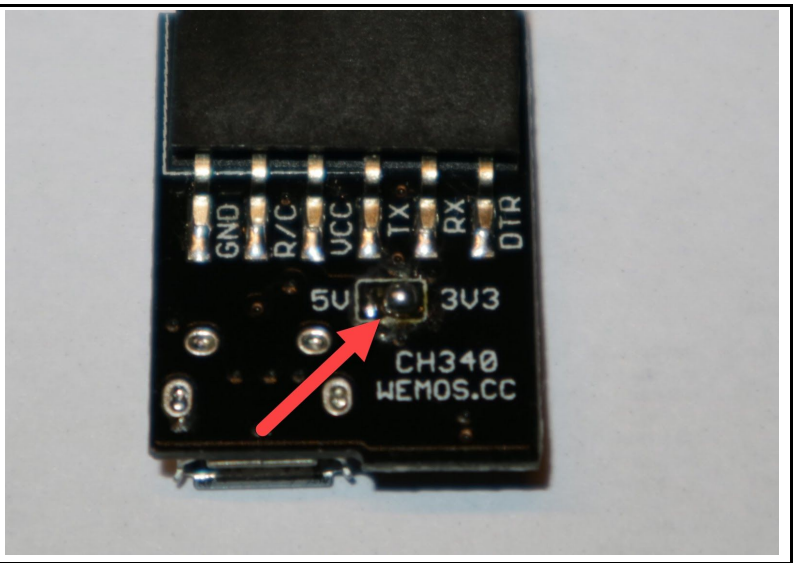
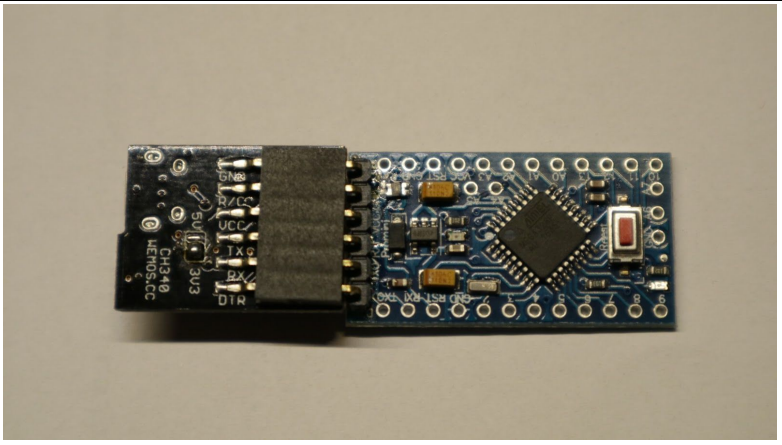
Solder the 6p male header onto the Arduino PCB	

Position the RFM95 radio chip, you might use some Tesla tape to fix it. The crystal should be in the right lower corner (as shown in the picture). Start with just a little solder in two opposite corners.	
Align the solder pads and if the module is in place, solder the other pads. Do use as little as possible solder! Check the connections carefully!	
Put two 12p headers into the ProMini socket. Also put the 2p header inside the ProMini area. By holding them with a piece of paper you can turn around the PCB. (You can also use the ProMini to keep the pins in place by using an elastic band.)	

First solder the end pins, align the headers straight and solder all the other pins.	
Turn around the PCB and solder the ProMini in place. Do not forget the two pin header!	
Now solder the 6p, 2p, 3p and 4p header.	
	

Cut a piece of wire, length 82.2 mm. This is a ¼ wave length antenna. You may add some short soldering end.	Wavelength =300/868 = 0.3456m Wavelength /4 = 0,0864 m correction 0.95*0.0864=0.0822m (to determine the precise length you can do some experiments, or use proper test equipment like a SWR meter)
Solder the Antenna into the hole near the connector pins. It is also possible to solder a SMA edge connector.	
Solder a header on the BMP280 barometric sensor	
Prepare the LED, this requires a 'helping hand' for soldering. Cut the longest wire of the LED ('+'), and the 1K resistor as shown	

Now solder the resistor, you may use a spare female header to create a ‘connector’ or some female jumper wires	
Put the LED ‘connector’ to pins 4 and 5 of the 6 pin jumper, as shown (Pin 4: GND, PIN 5 - D2), the resistor pin (‘+’) goes to pin 5 / D2. You may even solder the pins directly	
You can solder the battery holder on the pins GND (black) and 3.3V (red), you may use a female header here as well	

Ensure the soldering bridge on the FTDI is on the 3v3 position	
Connect the FTDI board to the Arduino as shown. DTR should match DTR, GND to GND, Vcc to Vcc, Tx To Rx and Rx to Tx. Check your wiring and connect the FTDI board to your PC.	

Starting up the Arduino Environment

https://www.arduino.cc/en/Guide/HomePage

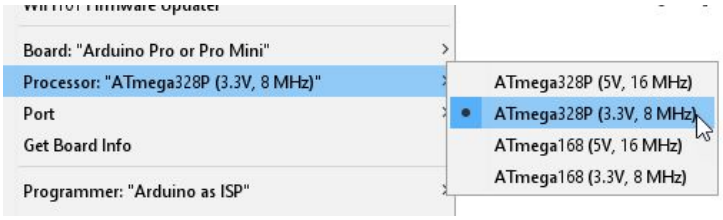
Install the Arduino IDE on your favourite platform (Mac, Linux or Windows). These examples are tested with Arduino IDE (tested on 1.8.0)!

https://www.arduino.cc/en/Guide/ArduinoProMini

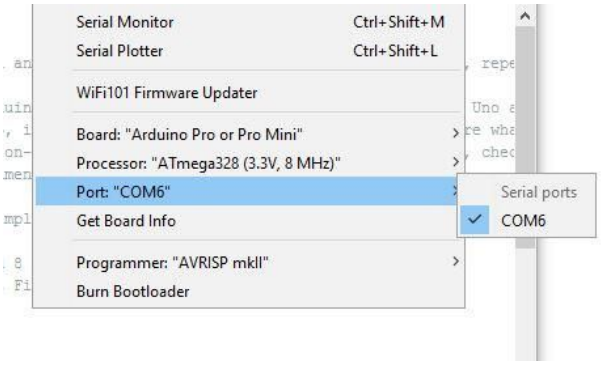
Use the FTDI, ensure that the jumper or bridge is on the '3V' side. Battery power turned off (check the switch)

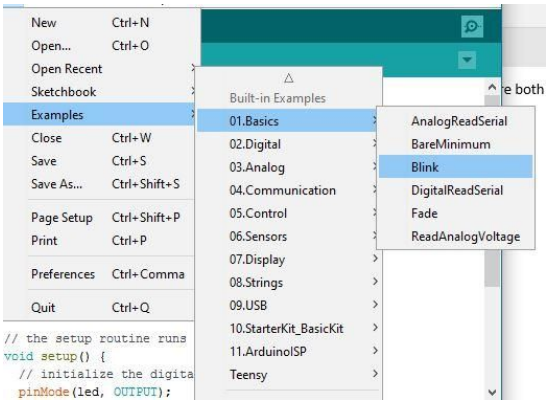
Connect the FTDI and the Pro Mini as shown in the pictures, align the signals: DTR should match DTR, GND to GND, Vcc to Vcc, Tx To Rx and Rx to Tx.

- Start the Arduino IDE
- Select the board: 'Arduino Pro or Pro mini'
- Select the Processor 'ATmega328 (3.3V, 8Mhz) -> IMPORTANT TO SELECT THE RIGHT CLOCK FREQUENCY!!

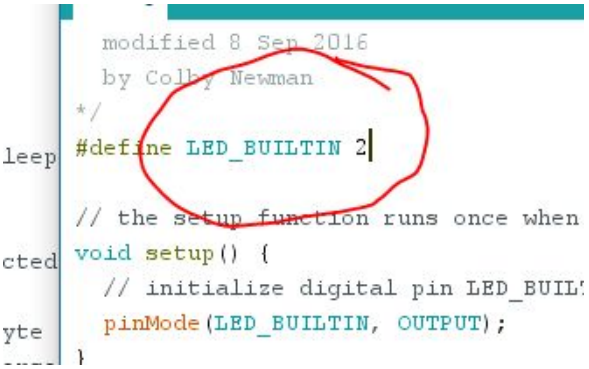


- Select the COM port given





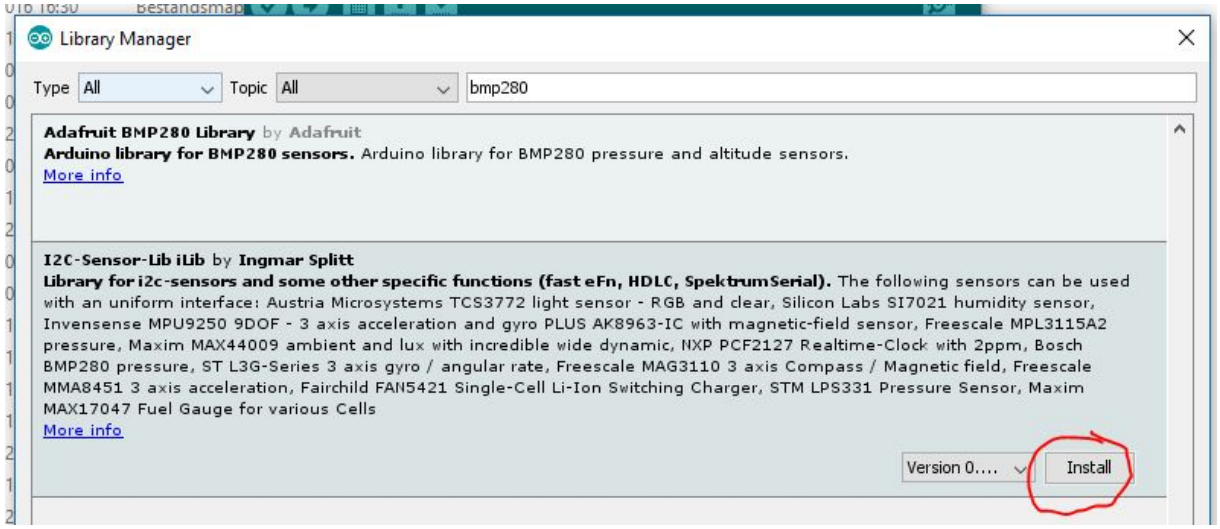
Select the default ‘BLINK’ example, change the LED pin from 13 to 2, in the newest blink example you may change ‘LED_BUILTIN’ to 2.



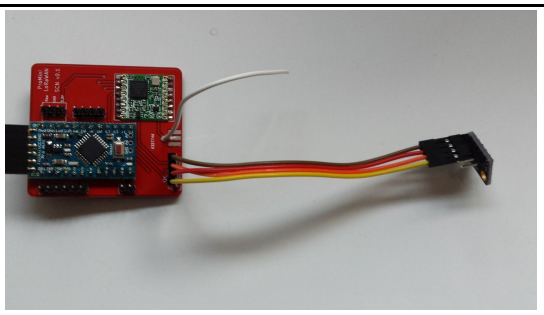
Load the code with CTRL-U to the Pro Mini. After compilation and uploading the led on the board should blink at 1 Hz. Great: we have a working programming environment!

Testing the sensors

1. Add the BMP280 library: Sketch-> include library -> manage libraries:
2. Search for BMP280:



3. Install the library 'I2C-Sensor-Lib'
4. Connect the BMP280 sensor:

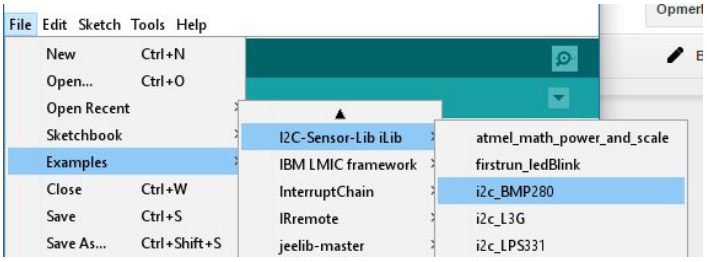


Connect the BMP280 sensor with female-female jumper wires to the I2C connector:

- GND-> GND
- SDA->SDA
- SCL->SCL
- VCC->+3.3V

The CSB en SDO are left open (only used in SPI connection). You can not connect straight forward, but you should twist the two wires in the middle.

5. Load the the i2c_BMP280 example (File->Examples->I2C-Sensor-Lib->i2c_BMP280). By

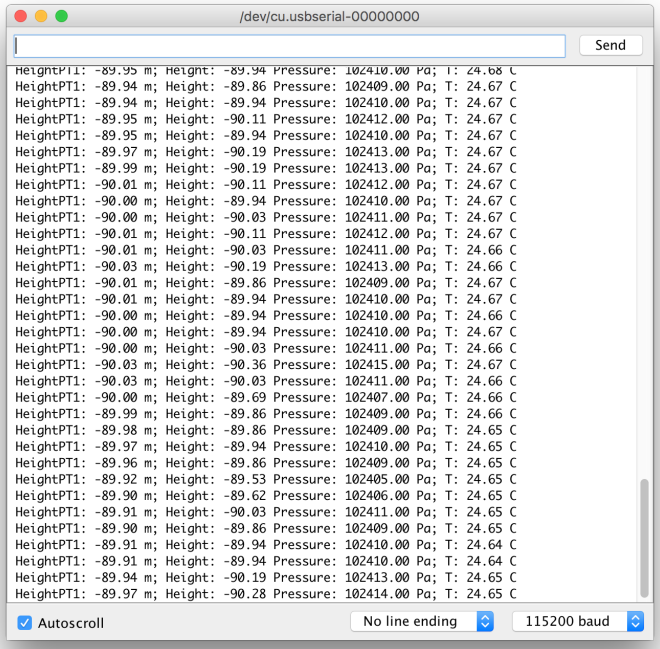


opening the Serial Monitor (115200 bps) you will see the height, pressure and temperature of the sensor. Put your finger on the sensor (the small metal unit) and see the temperature rise. The sensor's height is relative and this library has a 'fixed' sea level pressure. (Beyond the scope of this workshop, we do not use the height, but if you want to experiment with it): To change this you have to change the i2c_bmp280.h file. In line 306, 102025 is the actual

pressure:

```
meter = 44330.0*(1-pow(float(iPascal)/102025.0,1.0/5.255));
```

The Serial Monitor should display something like this:
if the Serial display shows ‘Chinese’ characters, check your baudrate and check your choice of ‘ Arduino Pro Mini 3,3V 8Mhz in the board selection.



Getting things started on 'The Things Network'

Download the LMIC library from Github:

1. Goto https://github.com/matthijskooijman/arduino-lmic
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
8. Goto https://github.com/rocketscream/Low-Power
9. Choose 'Clone or Download'
10. Download ZIP
11. Mark the location
12. Go to Arduino IDE
13. Select 'Sketch->Include Library->Add .ZIP Library'
14. Select the downloaded low-power-master.zip file from your download location

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.

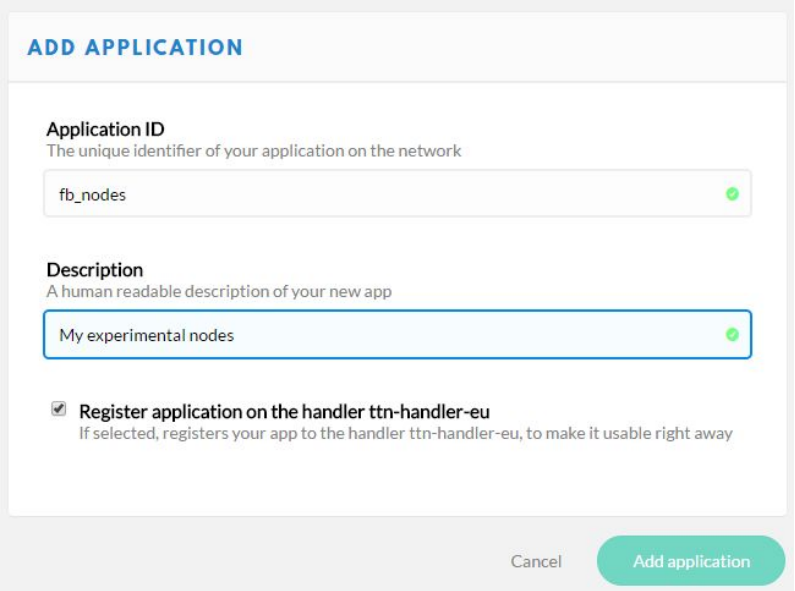
Create an Application

https://console.thethingsnetwork.org/applications

Choose 'add application'

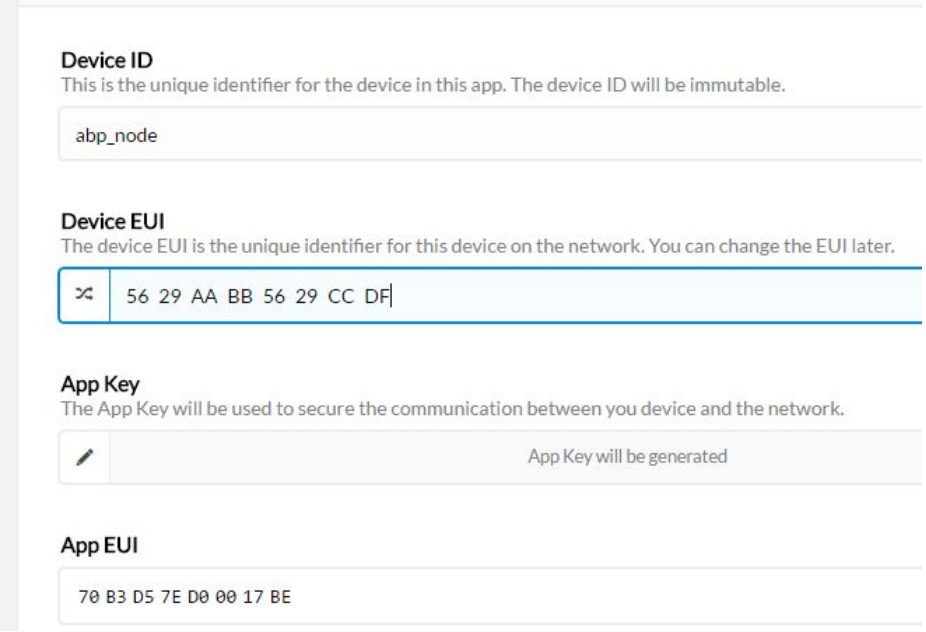
Give your Application a unique ID. You can use ONLY lowercase! You can add an unique number to get uniqueness over the ttn network (this is a global ID)

Your description can be any description you like.



ABP

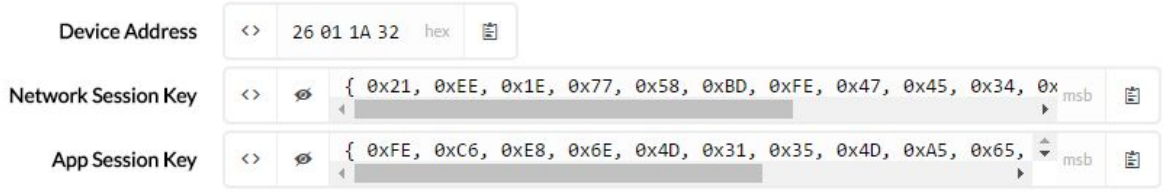
Activation by Personalization (ABP) is a method where the security keys are stored in the device. Not as safe as the OTAA method, but for experiments it works OK. There is no join procedure, nodes will work right away.



Choose *Register the device.*

Now edit the settings of the device and choose ABP (OTAA will be selected by default)

Select ‘save’. Now some values are system generated and we have to copy them to our code. Get the code from https://github.com/galagaking/ttn_bmp_280_abp We use the ttn_bmp280_abp_mini.ino example here.

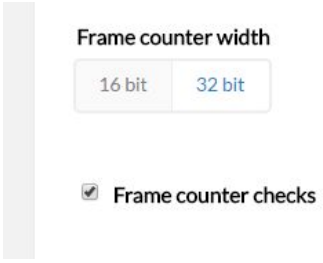


- Copy the Device Address as a HEX value to DEVADDR in the example, so 26 01 1A 32 will be 0x26011A32.
- Copy the Network Session Key as MSB to NWSKEY.
- Copy the App Session Key as MSB to APPSKEY.

```
// LoRaWAN NwkSKey, network session key, AppSKey, application session key, end-device address
static const PROGMEM u1_t NWKSKEY[16] = { 0x21, 0xEE, 0x1E, 0x77, 0x58, 0xBD, 0xFE, 0x47, 0x4
static const PROGMEM u1_t APPSKEY[16] = { 0xFE, 0xC6, 0xE8, 0x6E, 0x4D, 0x31, 0x35, 0x4D, 0xA
// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x260[illegible]2 ; // <-- Change this address for every node!
```

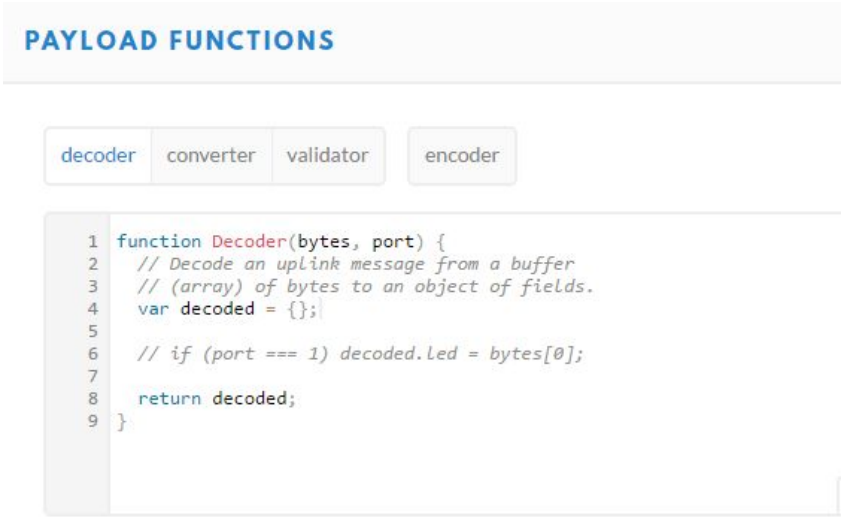
Compile and upload the code. Check in the dashboard the working.

You might want to uncheck the frame counter check



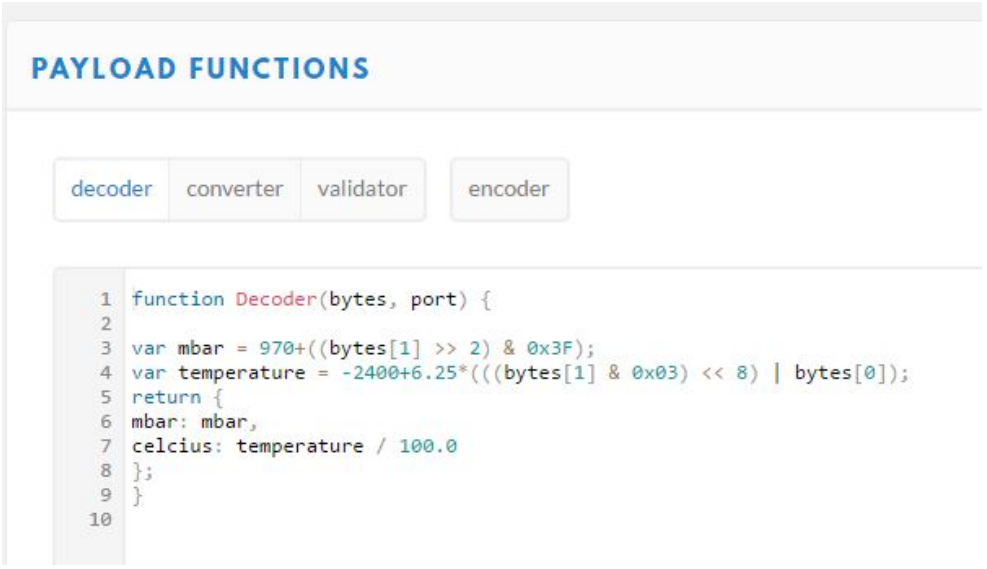
If the frame counter is checked, you must respect the sequence number, and probably copied packages are refused on the network. Though restarting you node, and thereby resetting the frame counter, will disable your node. So to get this working, you have to disable this check by unchecking this box.

To get the right display format, we can create a decoder in our application. Select your application and open the 'Payload Functions':



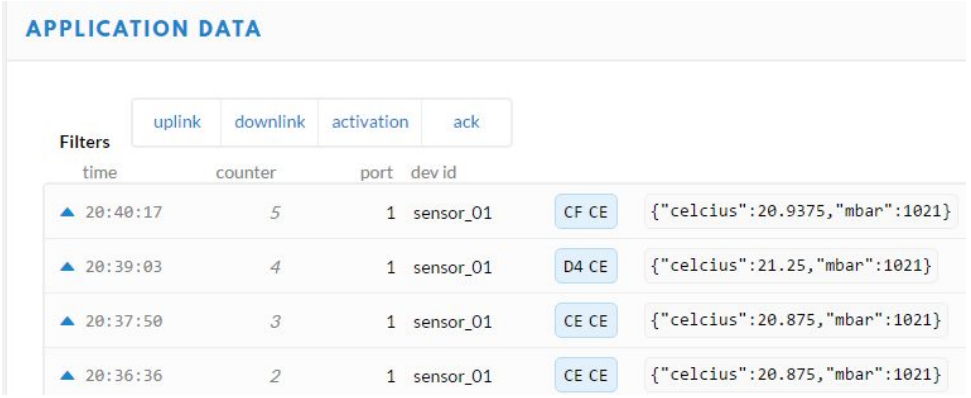
Enter the function below, overwriting the standard function

```
function Decoder(bytes, port) {
var mbar = 970+((bytes[1] >> 2) & 0x3F);
var temperature = -2400+6.25*(((bytes[1] & 0x03) << 8) | bytes[0]);
return {
mbar: mbar,
celcius: temperature / 100.0
};
}
```

First test the function with two dummy bytes CE D0, And then Save the function.

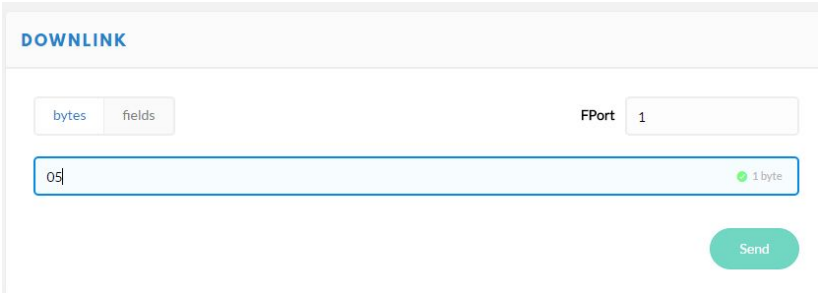
Return to your data and look what happens:



This way you can put several values into a byte string. Sending in clear ASCII is possible but due to bandwidth limitations not preferable in production environments. To make this even more scalable take a look at https://github.com/thesolarnomad/lora-serialization .

DOWNLINK

To send bytes to our node we can use the 'Downlink' section: in the console, click on your device and scroll down in the Overview page . Enter one byte (two hex digits, 05 for example) and click Send.



The information will be send to the node after the next time the node contacts the gateway. With a sample rate of approx. 60 seconds, that is the maximum time this data will be received. The Led will blink this amount of times, with a maximum of 10 (0x0A).

🎉 **Congratulations! You successfully completed the workshop.**

The Moment of truth

Now, it's time for a live demonstration. It's important to gather a small audience which you can impress with your node. Also, don't forget to make a picture and post it on Twitter using @thethingsntwrk

OTAA

The 'Over the Air Activation' is more secure than ABP because the key is not stored in the device, but exchanged at first contact with the network. A far more safer approach, but less successful in workshops due to a lot of 'authentication traffic'. You might want to test it yourself.

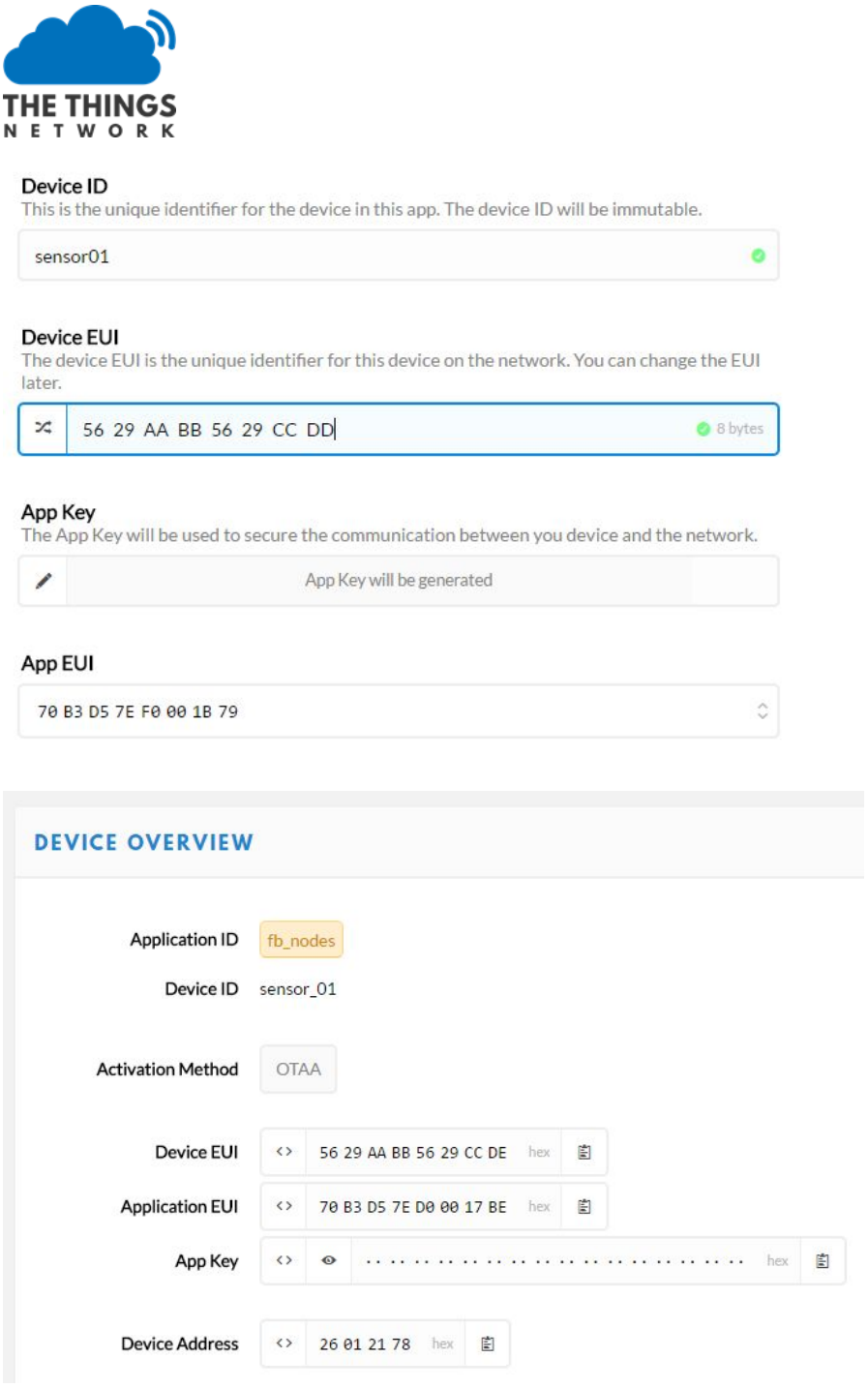
First download the code for ttn_bmp280i.ino from
https://github.com/galagaking/ttn_nodeworkshop

You can download the ZIP and unzip the file to get the examples:

1. Goto https://github.com/galagaking/ttn_nodeworkshop
2. Choose 'Clone or download'
3. Download ZIP
4. Open the ZIP in explorer
5. Open the example **ttn_bmp280.ino**
6. Create the sketch folder for the application

Register the device

Now register your device. We will use OTAA in this example. This is 'Over the Air Authentication'. You have to define an address for yourself. This is a kind of MAC address, but because there is no registration yet of these, define some RANDOM 8 byte value, using your postcode and some values in between.

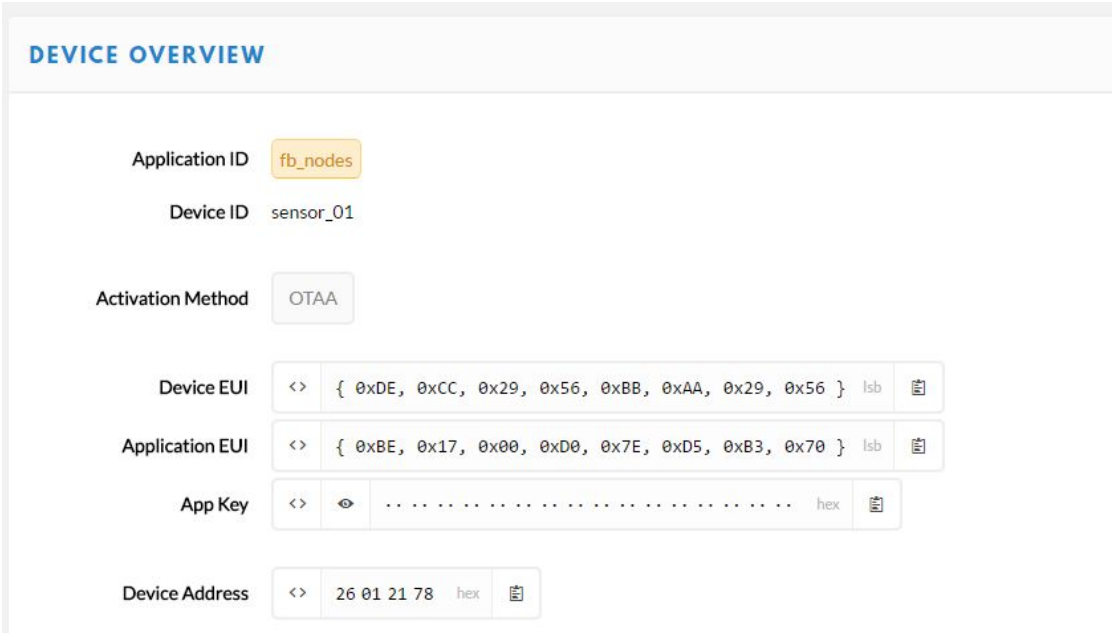


There are three values you should copy into your Arduino Code. Dev EUI (the device identifier), App EUI (The application identifier) and the App key.

Dev EUI

With the <> sign you can select different 'views'. We need the **'LSB'** view, also called little Endian or Least Significant Byte First.

NOTE: For the App Key we need MSB, not LSB.



Both Dev EUI and App EUI will be used in LSB or little Endian format.

First copy the Dev EUI with the clip board sign on the right. Paste the string into your code at the static const u1_t DEVEUI. Remember to respect the semicolon at the end of the line.

```
// 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 DEVEUI[8]  = { 0xDD, 0xCC, 0x29, 0x56, 0xBB, 0xAA, 0x29, 0x56 };
static const u1_t APPEUI[8] = { 0xBE, 0x17, 0x00, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };

// 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.
```

APP EUI

Next copy APP EUI, select **LSB** for this one as well. Copy to APPEUI in your code.

APP KEY

The last key is a secret hash, so this one just shows when you click the 'EYE' icon. After that you can copy this to the clipboard. This one can be copied in **MSB** format.

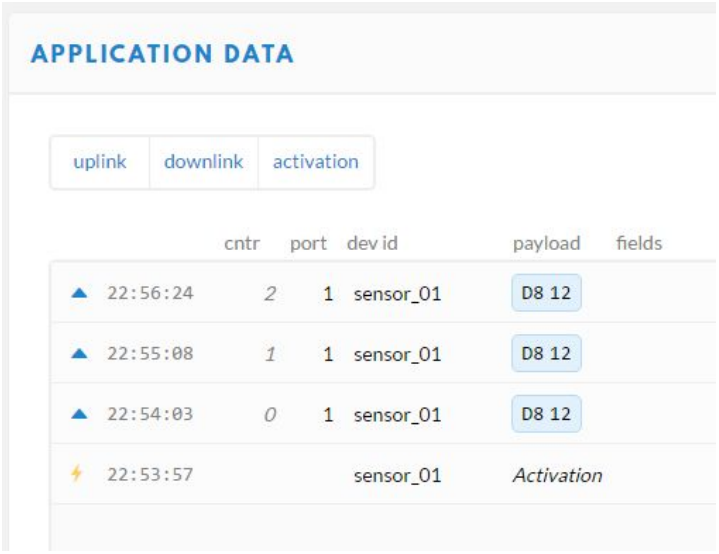
Copy the 16 bytes APPKEY after static const u1_t APPKEY[16] = .

The BMP280 sensor should be connected!

Upload your code to the Arduino.

The LED will blink during the JOIN process. The Arduino gets his keys from TTN, thereby all information sent will be encrypted with these keys.

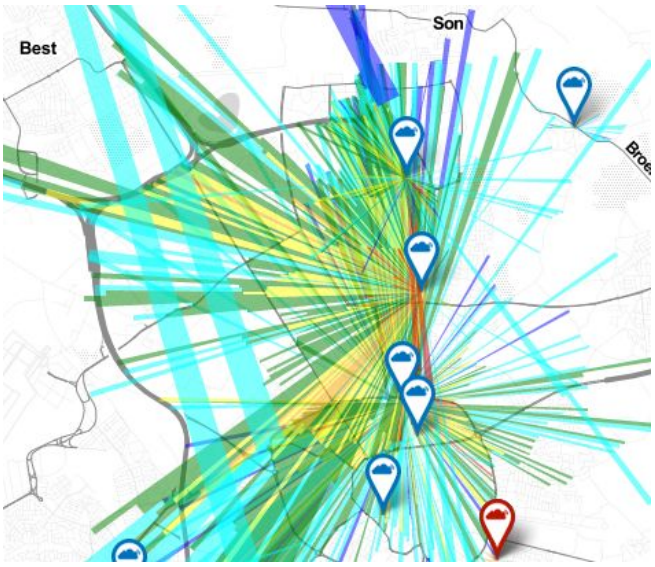
You can also follow this process by activating the ‘data’ view:



After a few seconds (hopefully, due to clock frequency drift this can last as long as 15 minutes or more), the LED will stop blinking. You will see the information coming in into the dashboard. This is coded information, because some algorithm is used to put temperature and humidity in two bytes. You can use the same payload function as in the ABP example.

TTN Mapper

On www.ttnmapper.org you can look at the coverage of The Things Network in your neighbourhood.



You can even contribute to this map by using the ttnmapper app on your Android Smartphone. You can use your TTN credentials to select your sensor

LMIC Pin Mapping

If you are using other 'LMIC' or TTN examples, there is ONE part to take care of, the pin setting of the RFM95 module. Most times you have to adjust this to the pinout of the board:

```
// Pin mapping is hardware specific
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    .dio = {4, 5, 7}, //DIO0, DIO1 and DIO2 connected
};
```

And even more…

- You can add professional Antenna's on the PCB by using the appropriate connector.
- Support: https://www.thethingsnetwork.org/forum/

Thanks to Doug Larue for his PCB design and sharing his manuals.

Frank Beks

December 2017

