Building a TTN node

Identifying the components

Our bill of materials (set available at TinyTronics)

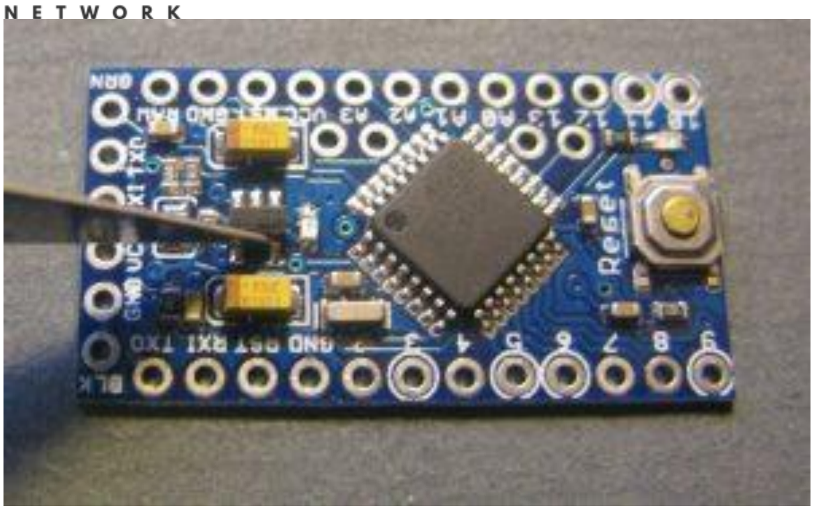
- RFM Module RFM95W (868 Mhz)
- Pro Mini 3.3V 8Mhz (5V will NOT work!!)
- FT232RL-3.3v-5v-TTL-USB-Serial-Port-Adapter (has to have a 3.3V option!)
- USB cable for FTDI
- 1x Led 3mm red
- 1x resistor 1k (brown-black-red)
- 2x resistor 4k7 (yellow-violet-red)
- 1x resistor 10k (brown-black-orange)
- 1x resistor 100k (brown-black-yellow)
- Header 6p female (Arduino programming)
- Header 4p female (sensor)
- Header 4p male 90 degrees (gps) (cut 2p from the header included with the Pro Mini)
- 2x header 8p 2mm (RFM95W)
- 2x header 12p 2.54mm male (Arduino)
- 1x header 2p 2.54mm male (Arduino)
- PCB ‘Loratracker RFM98
- Sensor TH06 (si7102) (Temperature and Humidity)
- Sensor BH1750 (light sensor)
- Ball sensor

We use a very compact ‘sandwich’ construction. So you have to follow the steps in the manual in the right order! You cannot reach some soldering points when you mess up the order!

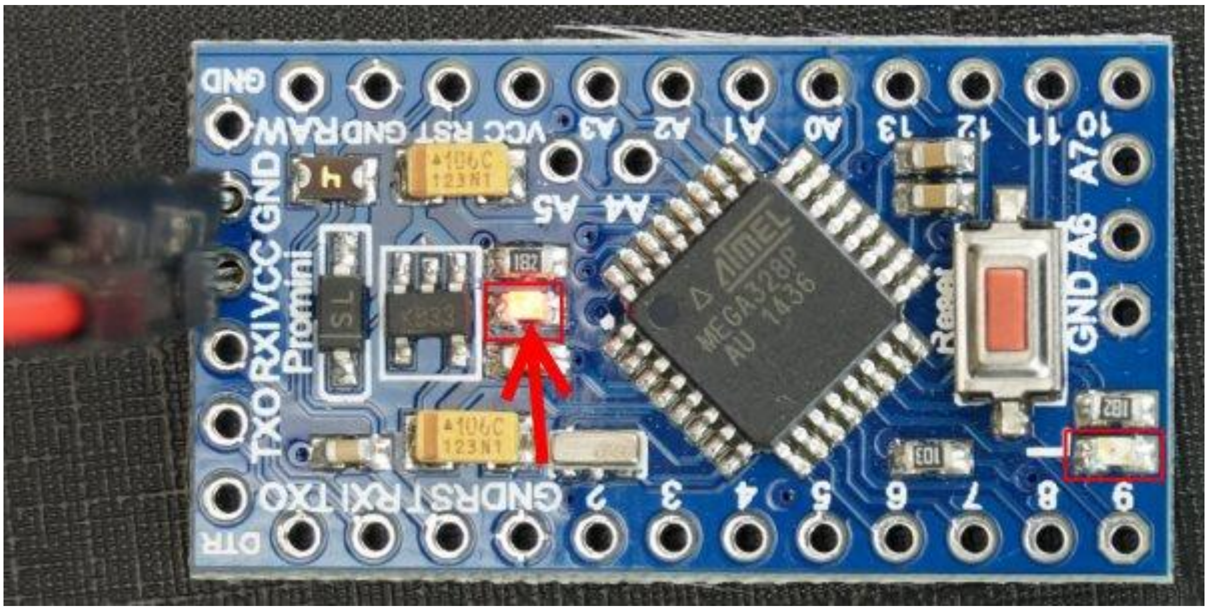
There is an option to ‘low power’ the Arduino. It requires a sharp knife to remove the power converter, and to cut the connections for the two on board leds. If you want to battery power your node for a longer time (weeks or more) this is the option for you. Current in sleep mode will be reduced to <100uA.

Prepare the Arduino

The easiest way to remove the regulator and not damage the board is to use a very sharp scalpel to cut through the regulator leads at the point they join the regulator body.

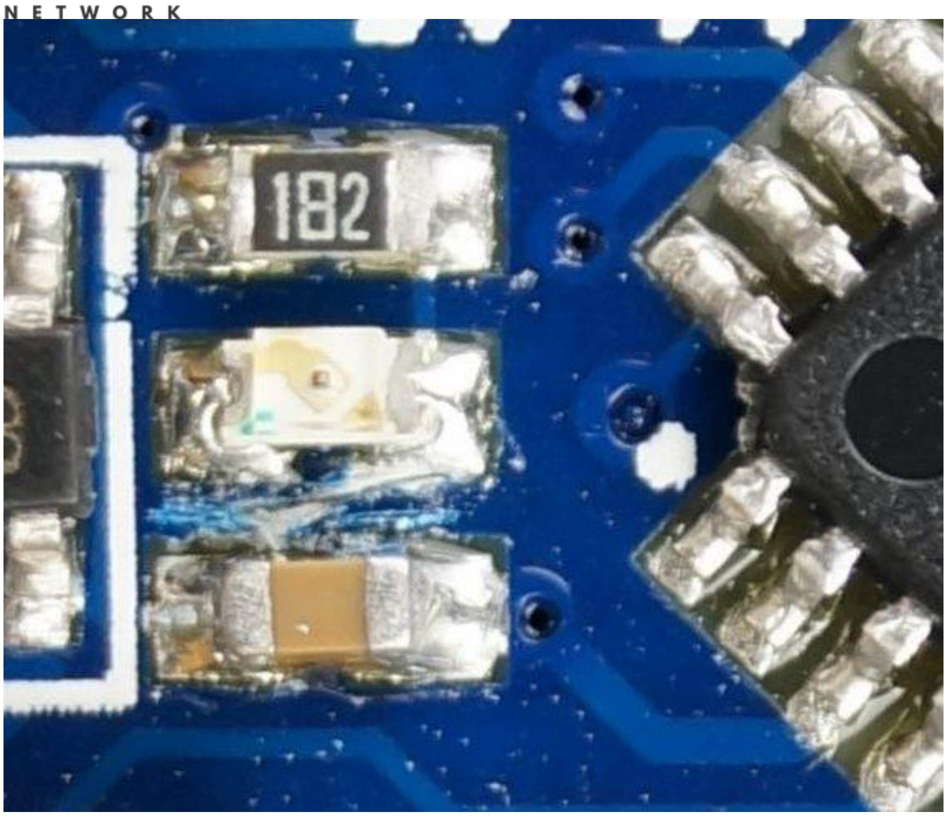


There are two LEDs on the Pro Mini board. The two LEDs are marked with a red square in the following picture. The power LED is marked with an arrow. If you are not sure where the power LED is on your board, then you can just power it and you will see the LED.



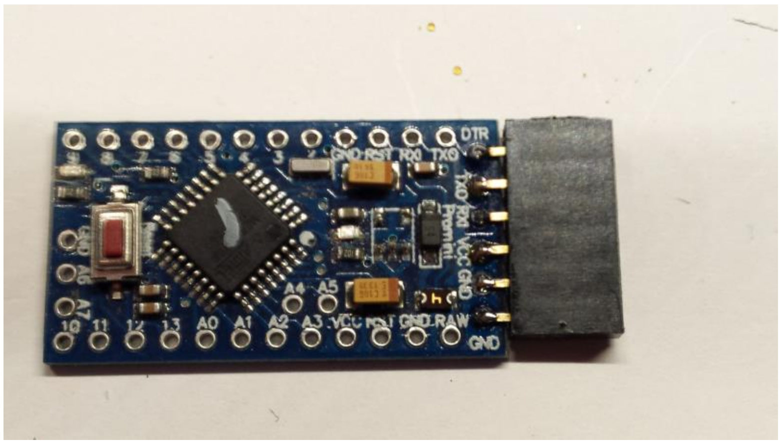
When you found the power LED, then try to locate at least one trace that leads to the LED. In the second picture below, I marked the traces on my board. A high-resolution picture with a lot of light helps to find the traces.

When you found a trace to the power LED, then you take a knife and break the trace, so that it will not conduct any more. You can see my result in the third picture below.



Removing the LEDs can be tricky, so it's easier to remove the series resistors for the LEDs instead. This version of Pro Mini also has a resistor feedback network for the regulator across VCC, these consume power so should be removed. Just push the resistors aside with a soldering iron. The picture shows the Pro Mini with unwanted parts removed, locations of the removed components are circled in red.

Connect the 6p female header to the Arduino. To get it low profile bow it so it is straight with the PCB.

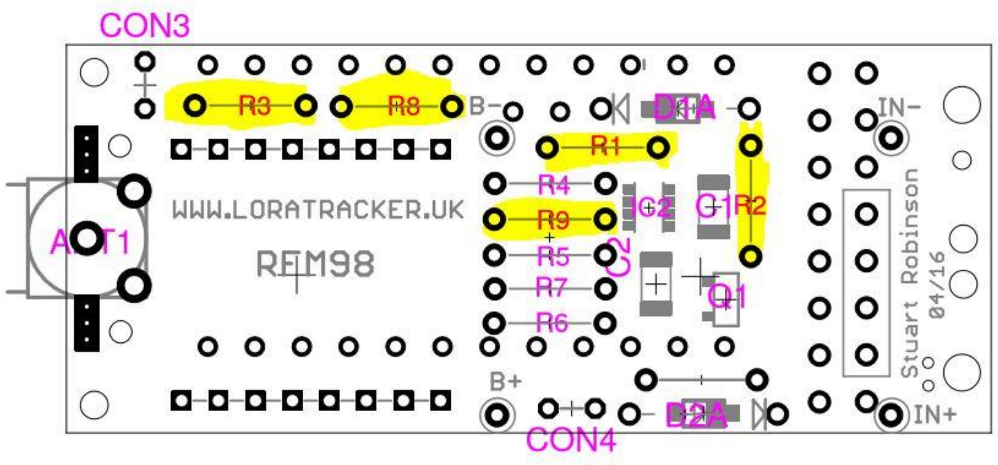


Mount the resistors on the PCB.

There is no identification in the Silk screen, but the resistors have to be mounted on the silk screen side (with the RFM98 text on top).

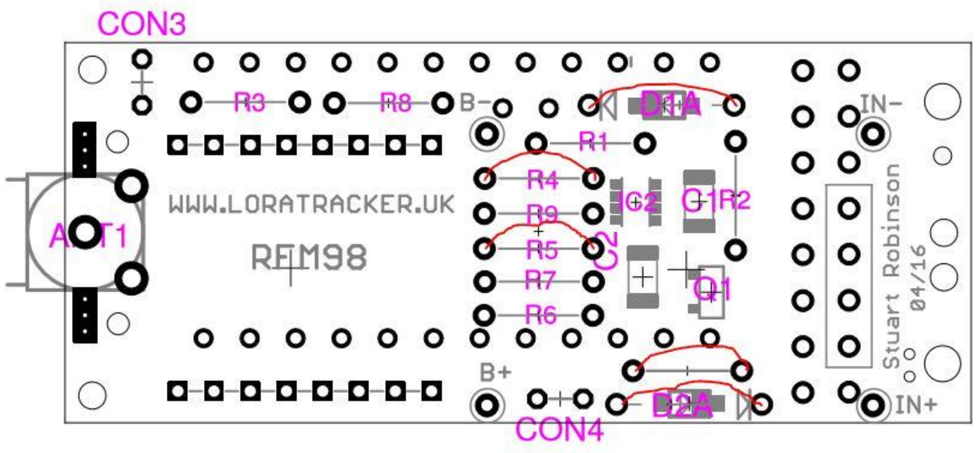
- R3: 1k resistor (brown-black-red)
- R8: 4k7 (violet, yellow, red).

- R1: 100k (brown, black, yellow).
- R9, 10k (brown, black, orange).
- R2 is vertical mounted on the right, 4k7 as well (violet, yellow, red).

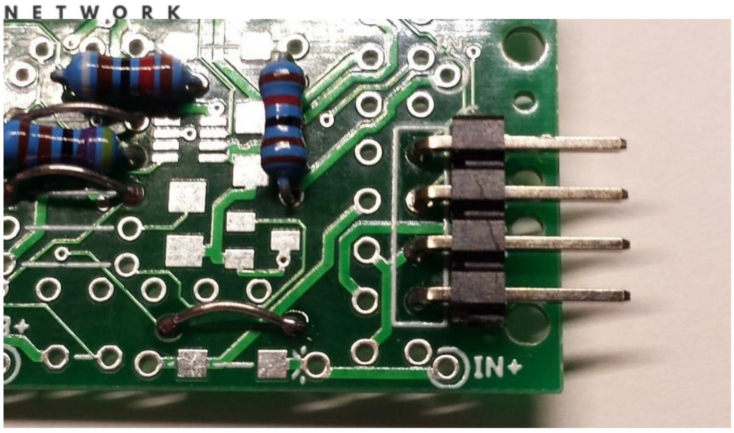


Cut the wires on the bottom, and create some bridges. These Bridges have to be true bridges, so not too close to the PCB.

- D1A will be a bridge
- R4 will be a bridge
- R5 will be a bridge
- C3 will be a bridge
- D2A will be a bridge

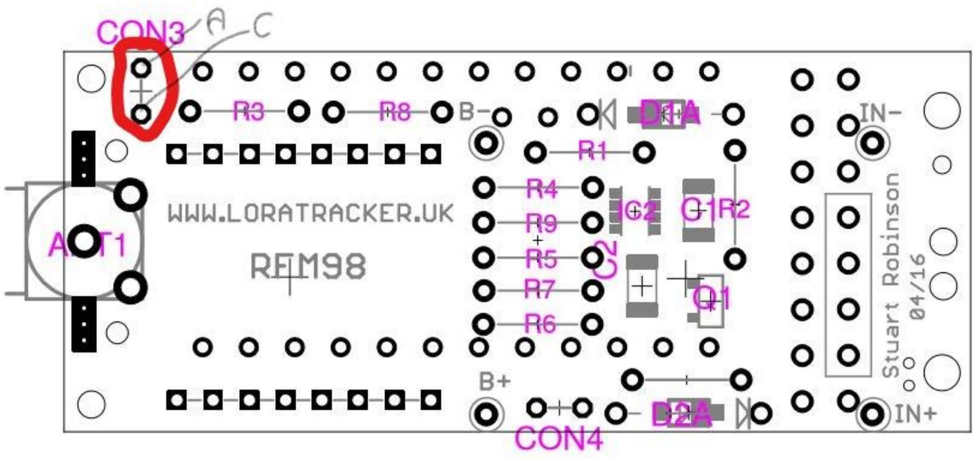


Mount the 4p header 90 degrees at the marked four pins (one above the side). The pins should point to the side of the board. You can use the left over program connector from your Pro Mini board.

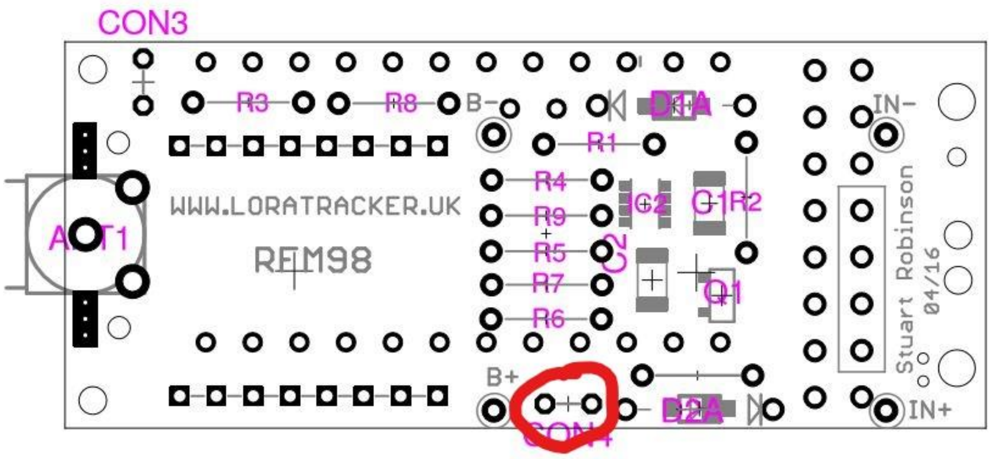


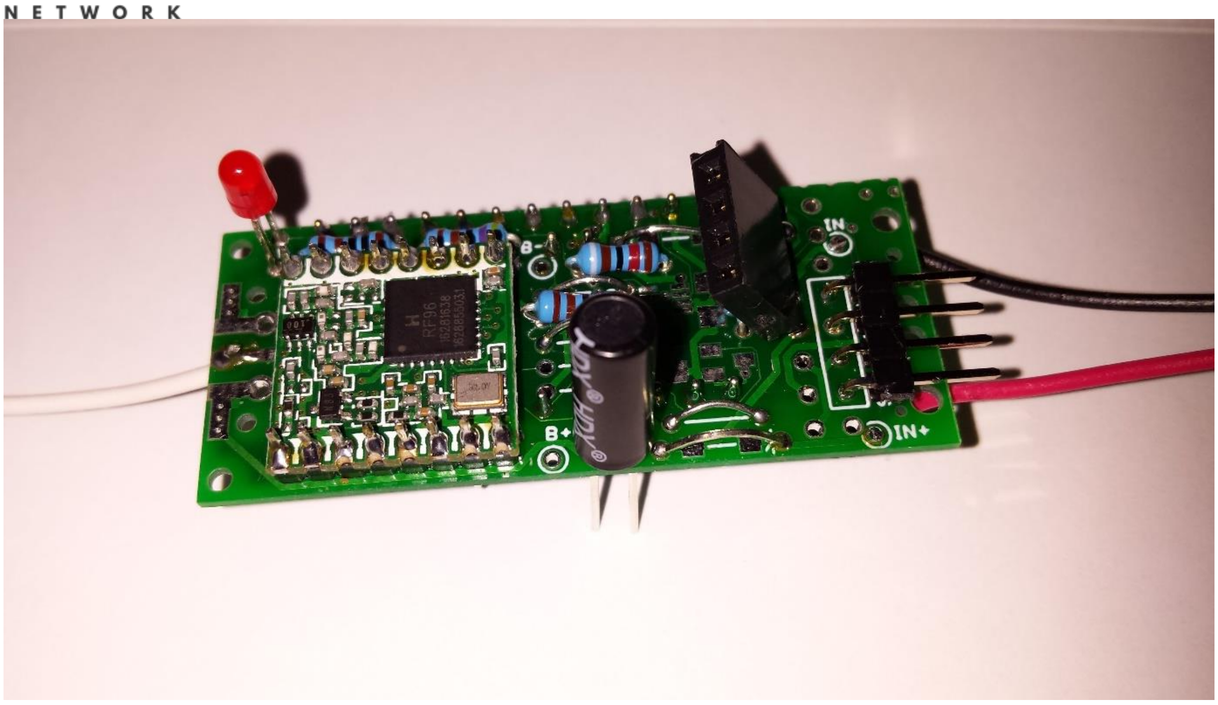
The battery wires connect to VCC and GND

Solder the LED on the CON3 position. The longest wire (Anode) should be on the outside of the board.



Mount the ball switch 'looks like a Condensator' into the CON4 position. The ball switch has no polarity, so you can place it in any direction. REMARK: if you do not want to use this you can connect any digital signal or switch on CON4!



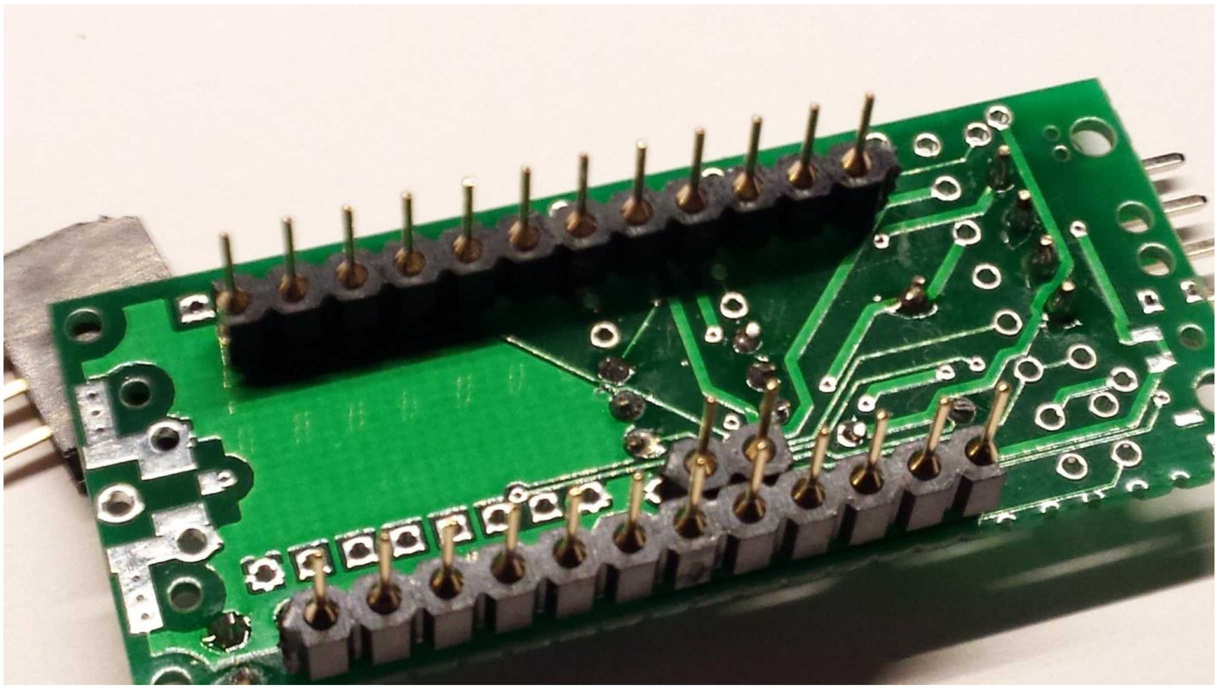


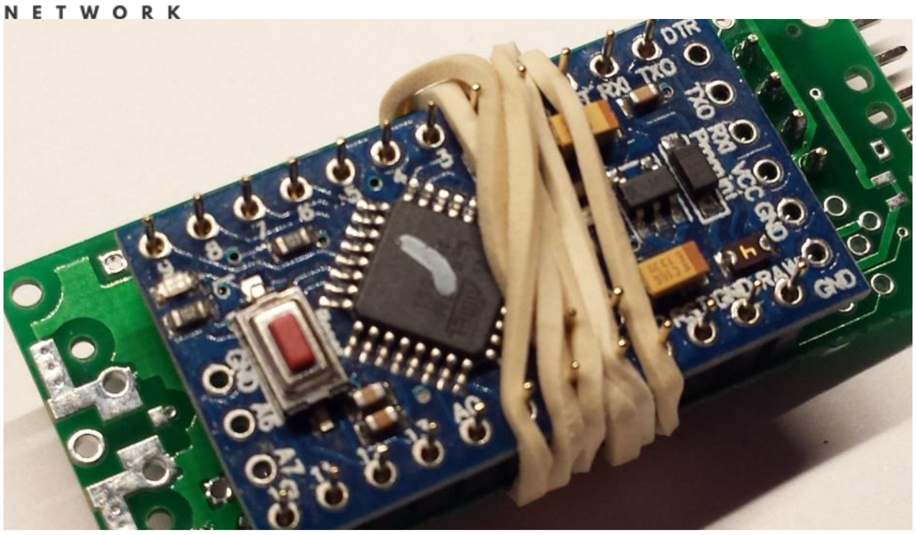
Mount the two headers of the Arduino Board, they are mounted on the other side of the board.

Put the small 2 pin header as shown on the picture.

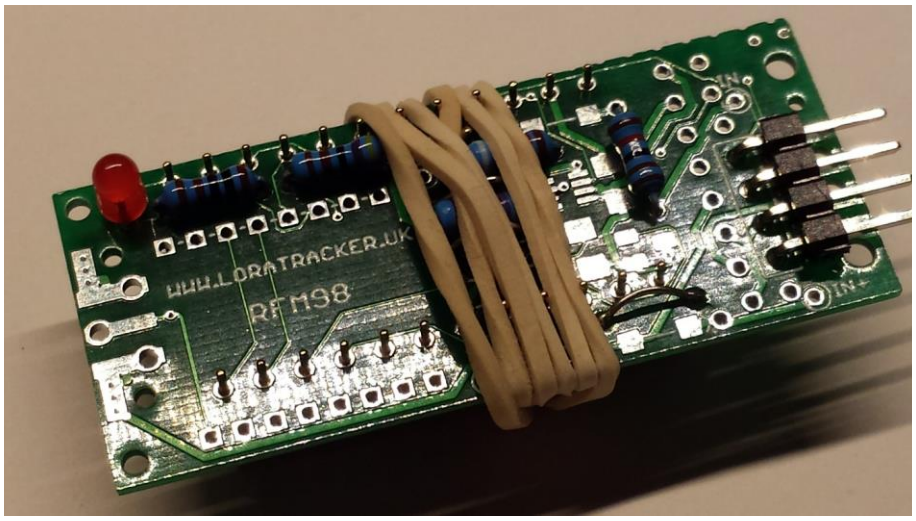
You can use the Arduino and an elastic band to hold the pins while soldering the pins to the board. Take note of the right place of the A4 and A5 pins!

DO NOT SOLDER THE ARDUINO AT THIS MOMENT!!!



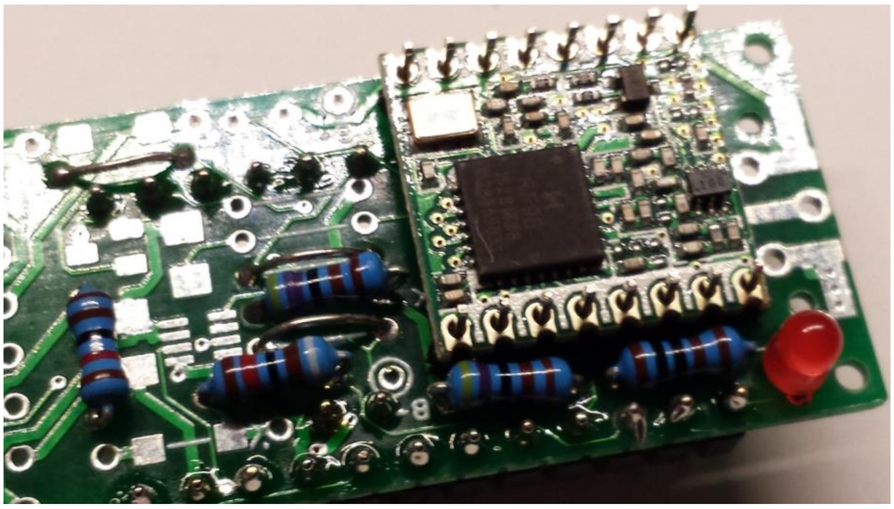


So only solder the pins on the LED side!

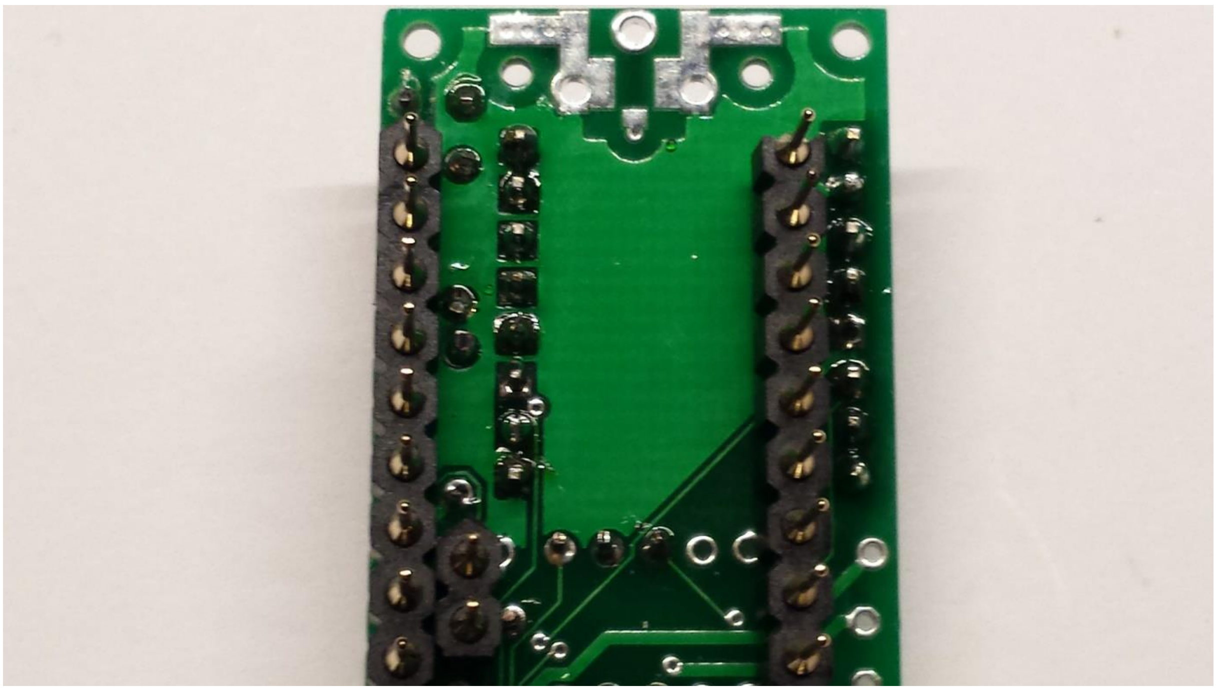


After soldering the 12+12+2 pins remove the elastic band and the Arduino.

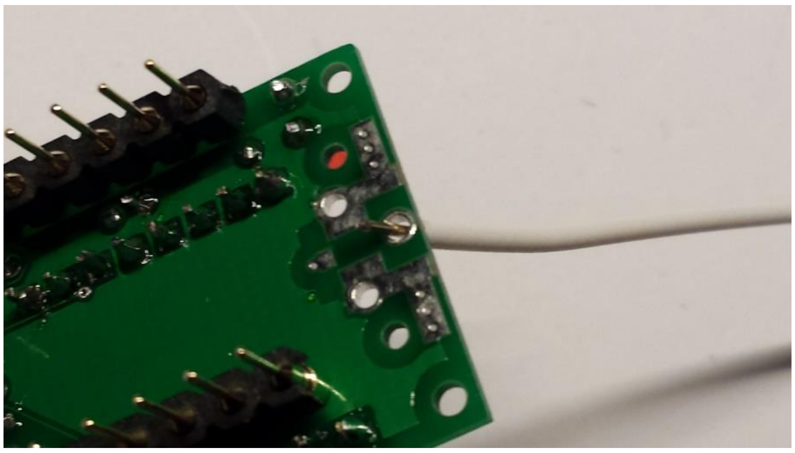
Now follow the same procedure for the RFM95 board. Put the 8pin 2mm headers on the ‘resistor’ side of the board with the RFM95 on top.



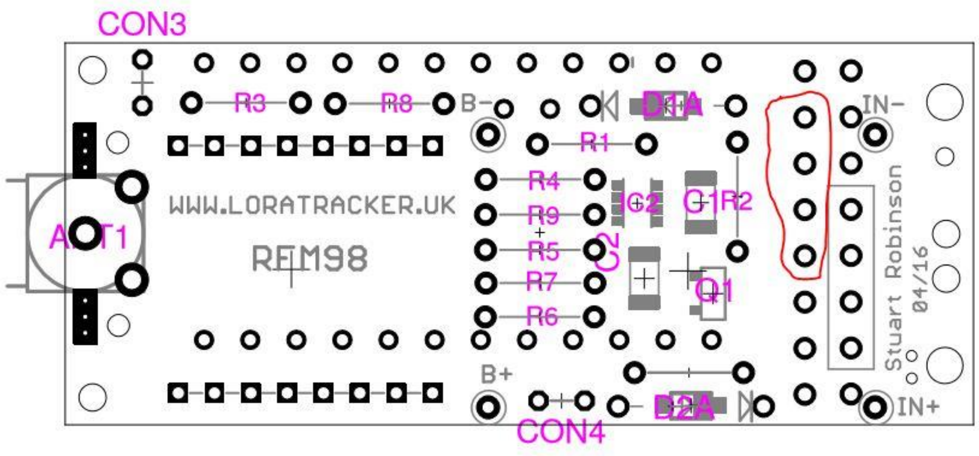
Now solder the other side, besides the two Arduino connectors.



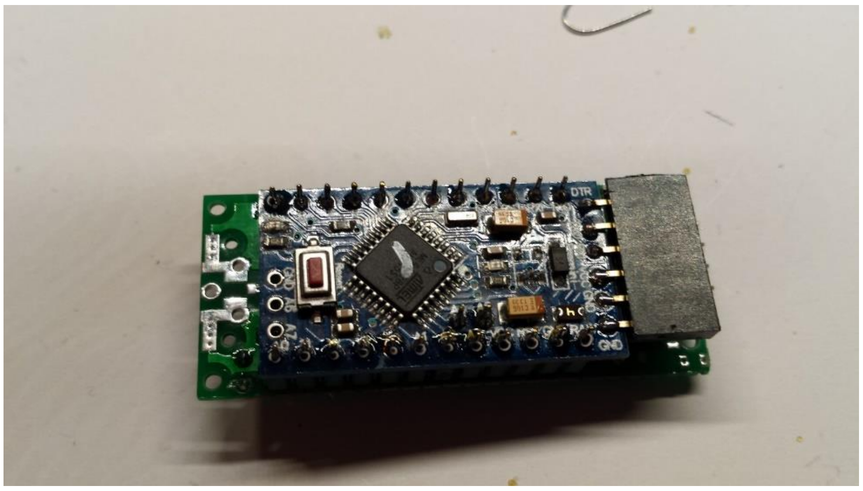
Cut a wire for the antenna. It should be 82mm long, measured from the PCB. So add some 5 mm to solder it. We use the mid hole of the antenna connector. You can as well use a SMA or uC connector on these pads for later experiments.



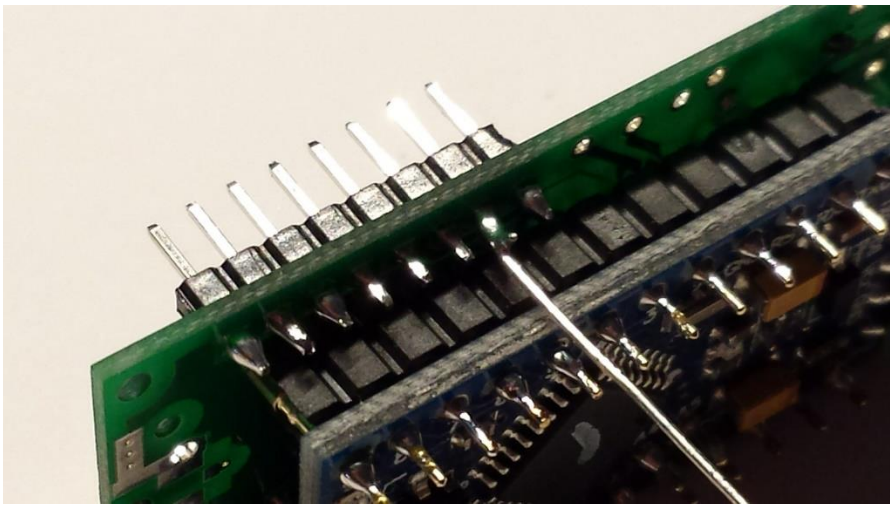
Put the 4 pin female header on position 2, 3, 4 and 5 of the left row as shown in the red circle.



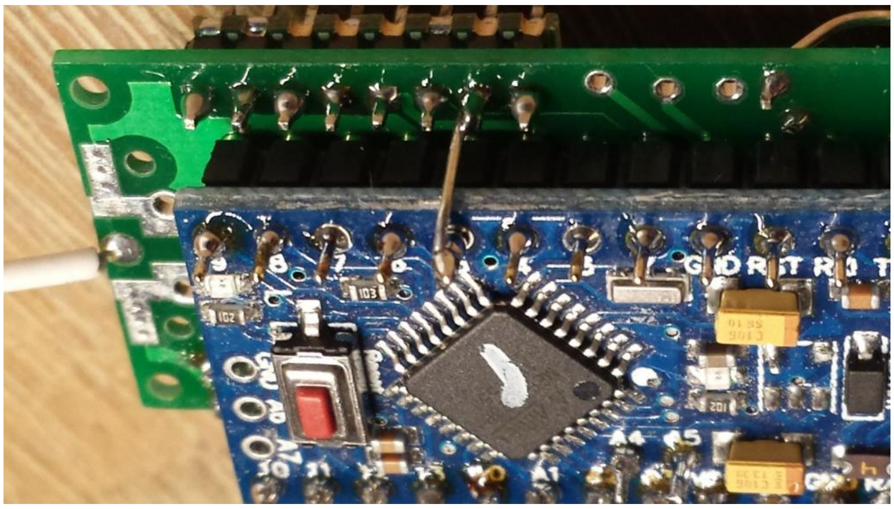
Put the Arduino in place. The reset button goes on the Antenna side. Solder the Arduino, do not forget A4 and A5.



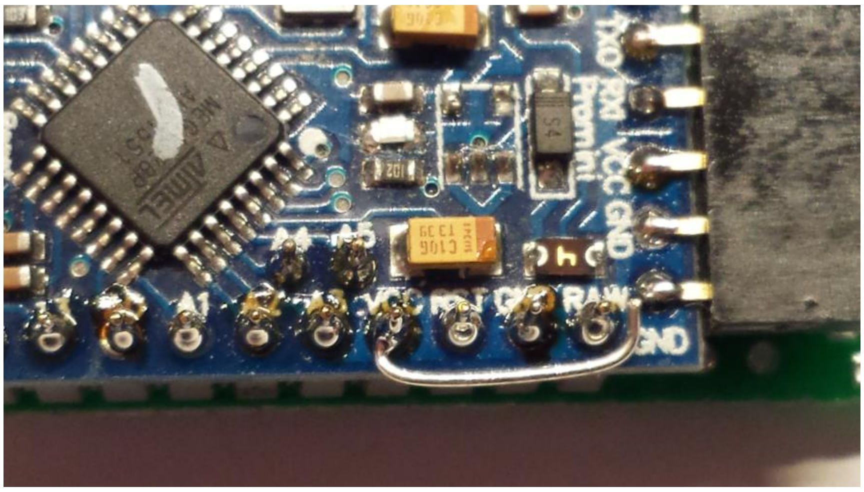
We have to modify two points here. The RFM95 is used in 'LoRa' mode and therefore pin Dio1 should be connected. The easiest way is to connect the unused D5 pin from the Arduino to Dio1.



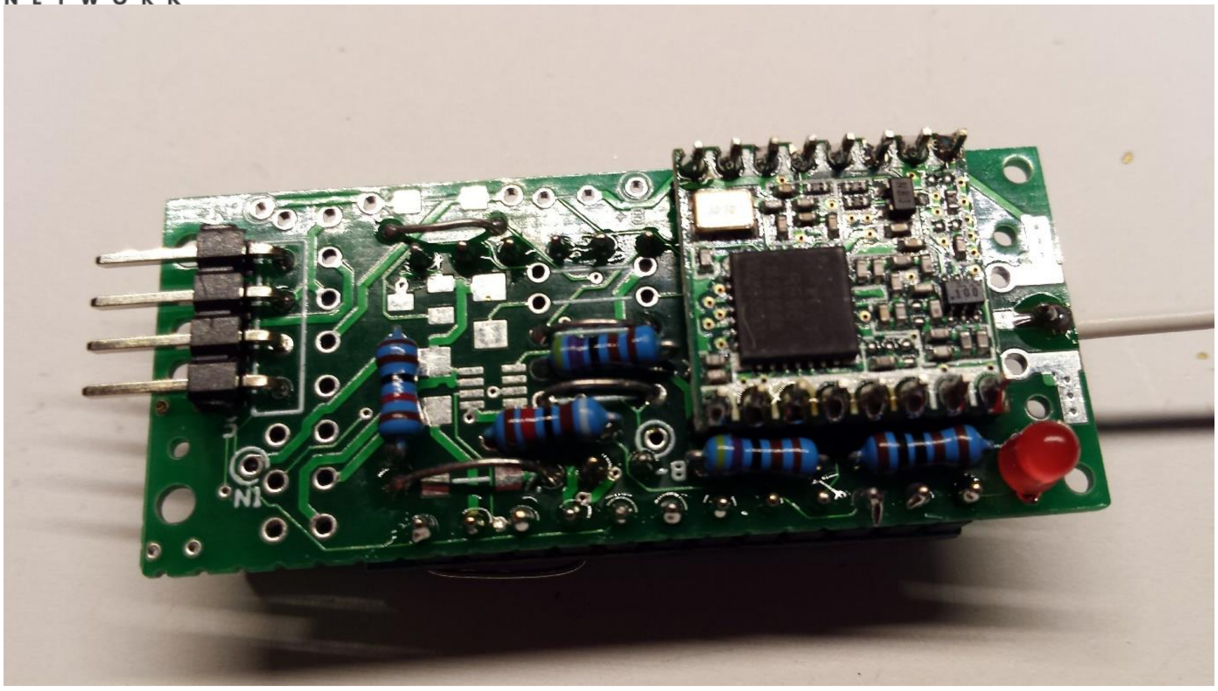
DIO1 is the second pin in line of pin 5 of the Arduino. Solder a short piece of wire on it. Cut the wire in the right length, bend it to pin 5 and connect it (without connecting the other pins).



Now connect with a bridge pen 'RAW' and 'VCC' on the Arduino.

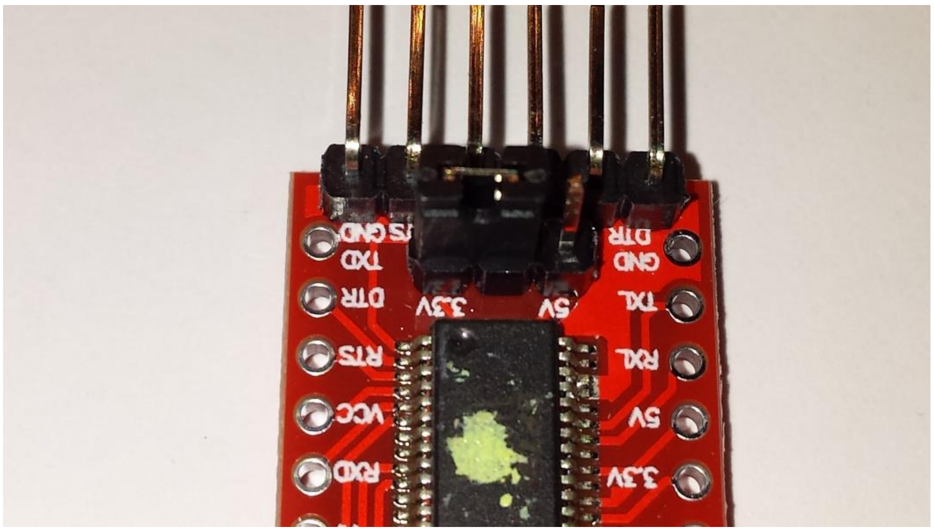


Solder the RFM95 Board. Look at the picture for the orientation. The big black chip should be on the inner side of the PCB.

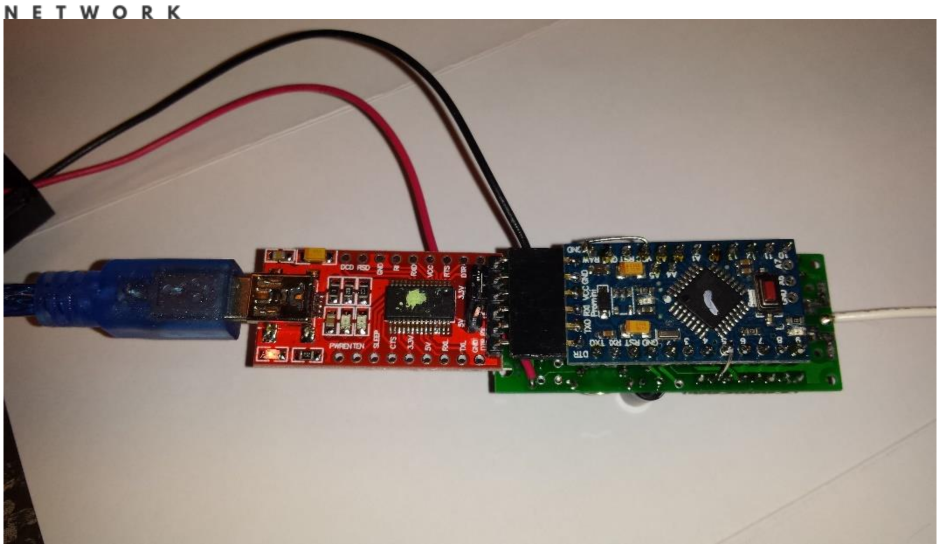


Solder a header on the si7021 sensor (marked as TH06). You can connect the TH06 sensor direct on the board. First solder a 4p header on the sensor.

Put the Jumper on the FTDI board on 3.3V position!



Connect the USB cable to the FTDI board, the FTDI board to the Arduino as shown.



Starting up the Arduino Environment

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

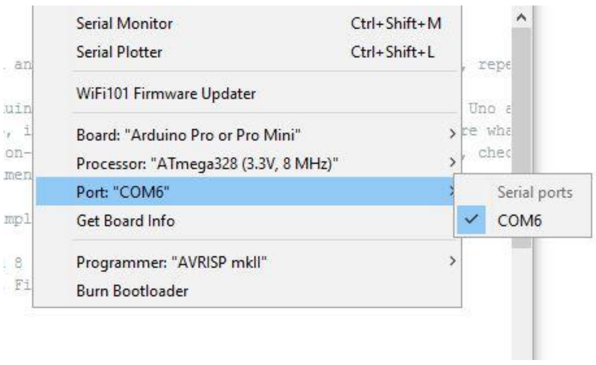
Install the Arduino IDE on your favourite platform (Mac, Linux or Windows).

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

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

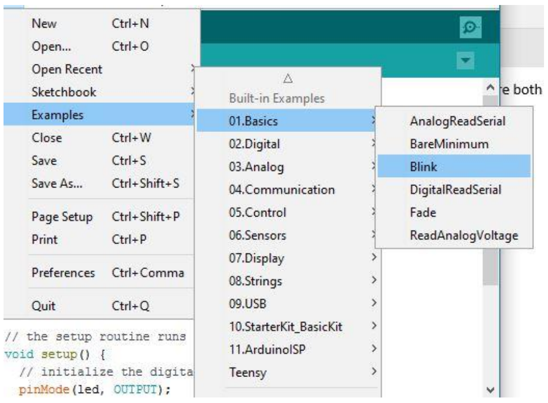
Connect the FTDI and the Pro Mini so that the components are both facing up.

- Start the Arduino IDE
- Select the board: 'Arduino Pro or Pro mini'
- Select the Processor 'ATmega328 (3.3V, 8Mhz)
- Select the COM port given



Board: "Arduino Pro or Pro Mini"
Processor: "ATmega328 (3.3V, 8 MHz)"
Port: "COM6"
Programmer: "AVRISP mkII"

Select the default 'BLINK' example, change the LED pin from 13 to 10:



```
// the setup routine runs
void setup() {
  // initialize the digita
  pinMode(led, OUTPUT);
```

Load the code with CTRL-U to the Pro Mini

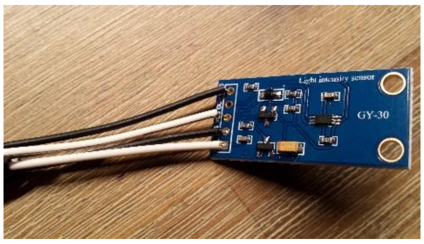
After compilation and uploading the led on the board should blink at 1 Hz.

Great: we have a working programming environment!

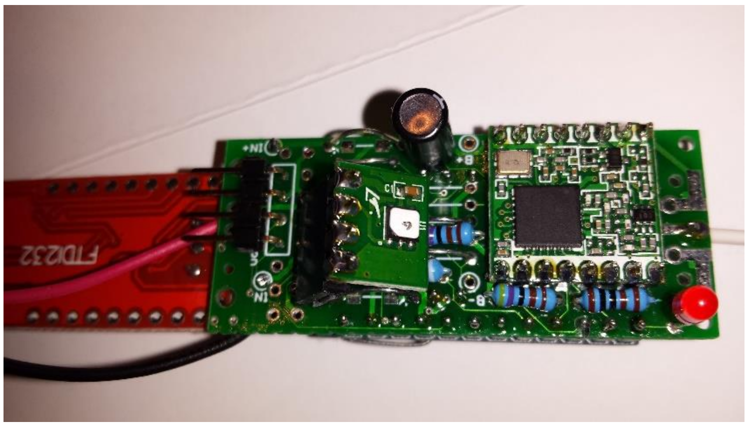
We will first test the light and the temperature sensor

1. Download the sensor libraries: https://github.com/claws/BH1750
2. Import the library in the Arduino environment: include library -> Add .ZIP library and select the downloaded ZIP file.
3. Select 'manage libraries' and search for 'si7021'. You will find the universal i2c sensor library. Install this one.

4. Connect the GY-30 / BH1750 light sensor: GND-> GND, SDA->A4, SCL->A5, VCC->+5V. You can cut two jump wires to create four and solder them on the specified sensor points



5. Run the BH1750test example.
6. By opening the Serial Monitor (9600 bps) you will see the LUX value of the sensor. If you cover the sensor, the value will change.
7. Connect the TH06 (temperature) sensor. It will fit in the header, pointed to the RFM95 radio.



8. Run the i2c_si7021 test example. You will see the temperature and humidity in the monitor.

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. Find the LMIC.C
9. Go to line 685
10. Change setDrJoin(DRCHG_SET, DR_SF7); to: setDrJoin(DRCHG_SET, DR_SF9);
11. Save LMIC.C

The last change is known as a 'dirty fix', it will help joining on OTAA faster (seconds in stat of 7 minutes).

The Things Network Dashboard

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

Create an Account

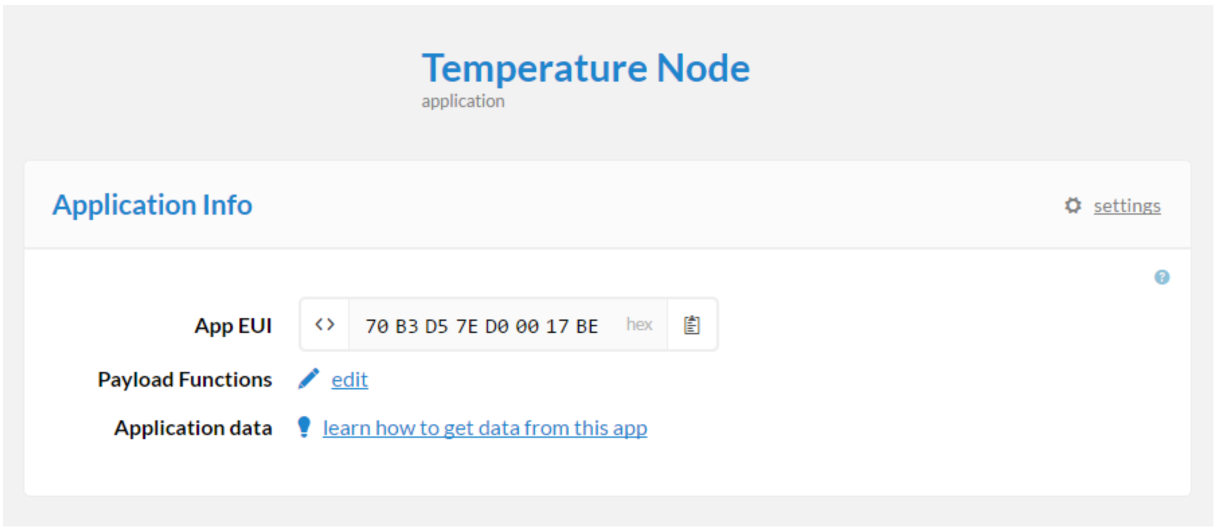
To use the dashboard, you need a The Things Network account. You can create an account here.

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

Create an Application

https://staging.thethingsnetwork.org/applications/create

Name your application 'Temperature Node'



Temperature Node
application

Application Info — settings

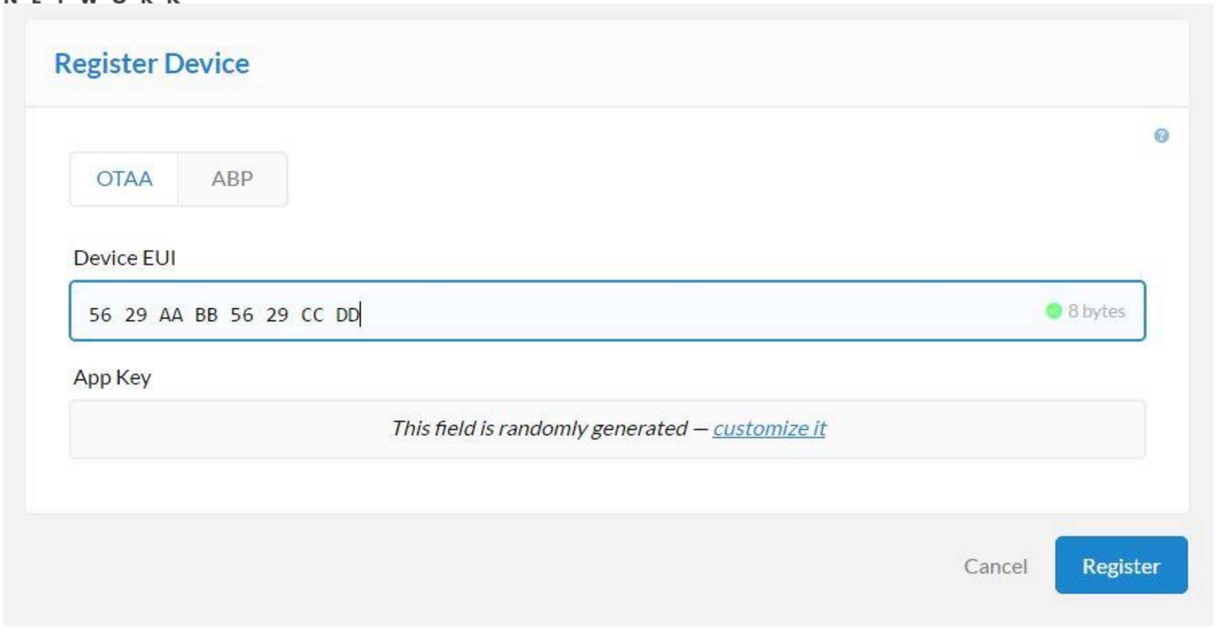
App EUI: 70 B3 D5 7E D0 00 17 BE hex

Payload Functions: edit

Application data: learn how to get data from this app

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

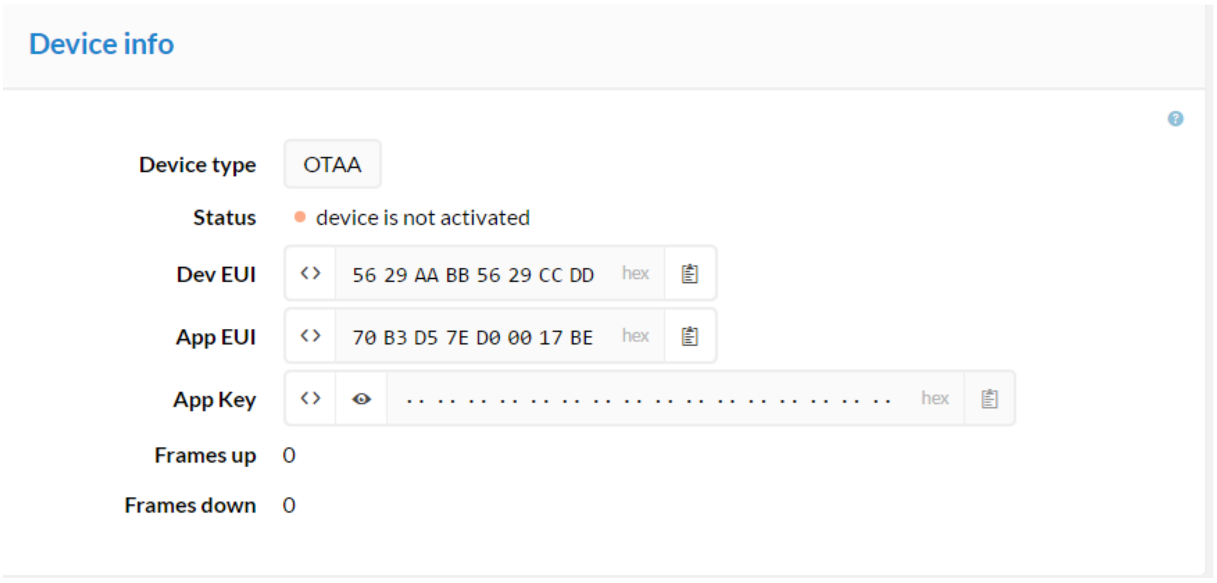
Now register your device. We will use OTAA in this example. This is 'Over the Air Authentication'. You have to define an address for yourself. This is a kind of MAC address, but because there is no registration yet of these, define some RANDOM 8 byte value, using your postcode and some values in between.


Register Device

Device EUI: 56 29 AA BB 56 29 CC DD (8 bytes)

App Key: *This field is randomly generated — customize it*

Register the device

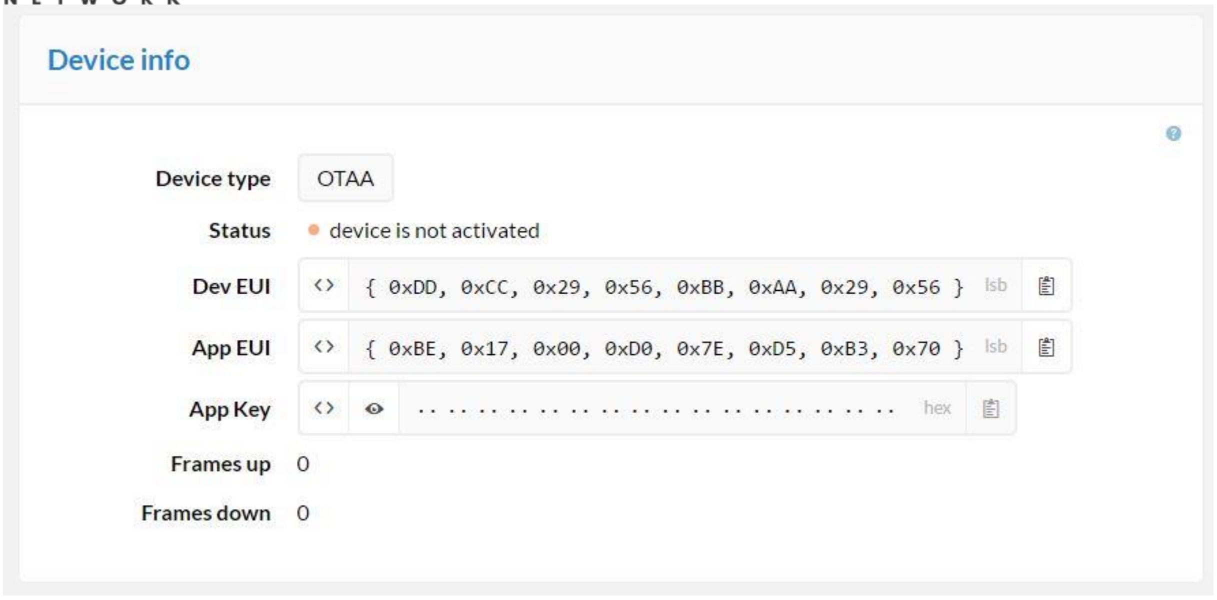

Device info

Device type	OTAA
Status	device is not activated
Dev EUI	56 29 AA BB 56 29 CC DD hex
App EUI	70 B3 D5 7E D0 00 17 BE hex
App Key	.. .. hex
Frames up	0
Frames down	0

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.

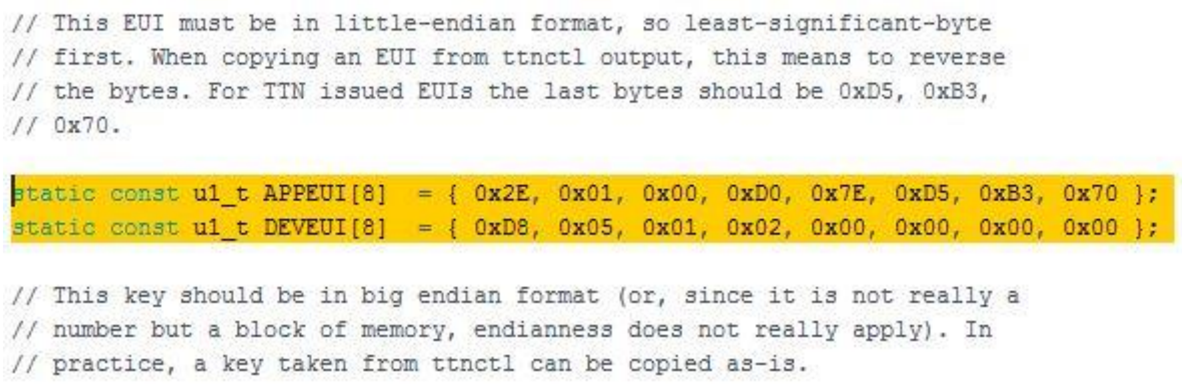


Device info

Device type	OTAA
Status	● device is not activated
Dev EUI	{ 0xDD, 0xCC, 0x29, 0x56, 0xBB, 0xAA, 0x29, 0x56 } lsb
App EUI	{ 0xBE, 0x17, 0x00, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 } lsb
App Key	.... hex
Frames up	0
Frames down	0

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 APPEUI. 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 APPEUI[8]  = { 0x2E, 0x01, 0x00, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
static const u1_t DEVEUI[8]  = { 0xD8, 0x05, 0x01, 0x02, 0x00, 0x00, 0x00, 0x00 };

// 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.
```

Next copy DEVEUI, select MSB for this one as well.

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 LSB format.

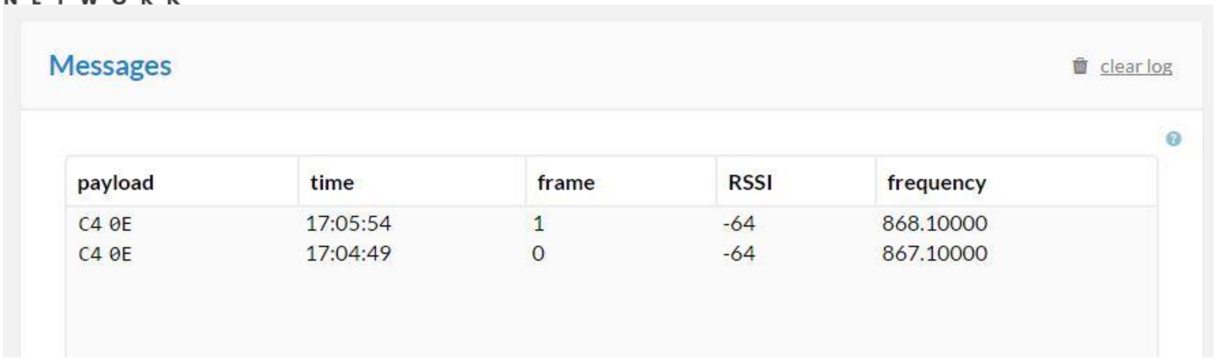
Copy the 16 bytes APPKEY after static const u1_t APPKEY[16] = .

The Temperature 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.

After a few seconds, the LED will stop blinking. You will see the information coming in into the dashboard:

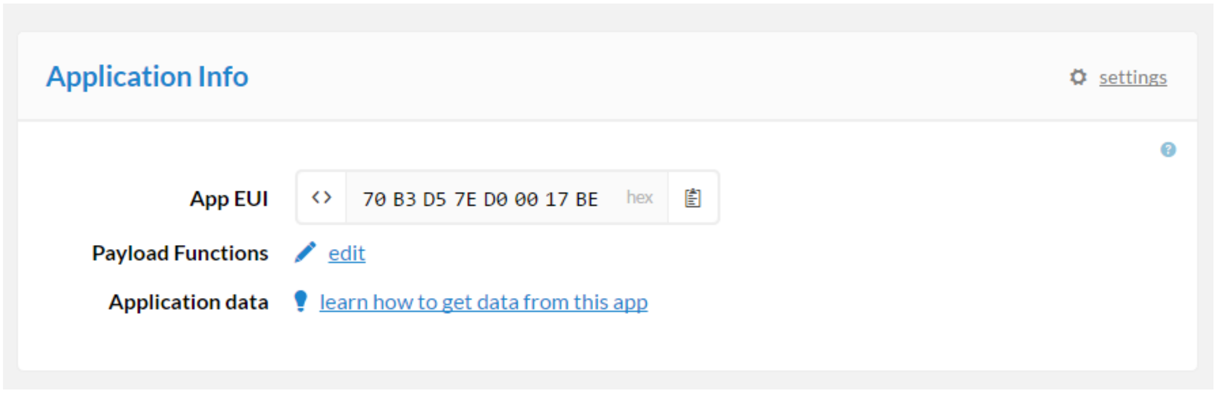

Messages

clear log

payload	time	frame	RSSI	frequency
C4 0E	17:05:54	1	-64	868.10000
C4 0E	17:04:49	0	-64	867.10000

This is coded information, because some algorithm is used to put temperature and humidity in two bytes.

To get the right display format, we can create a decoder in our application. Select the application 'Temperature Node' and click 'Edit' on the right of the Payload Function:


Application Info

settings

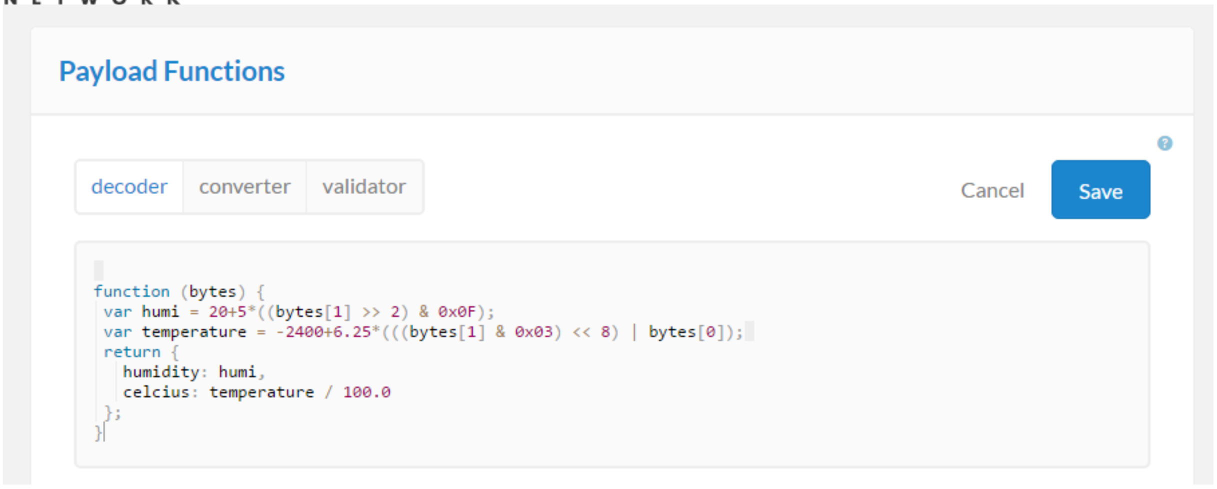
App EUI: 70 B3 D5 7E D0 00 17 BE hex

Payload Functions: edit

Application data: learn how to get data from this app

Enter the function below, overwriting the standard function

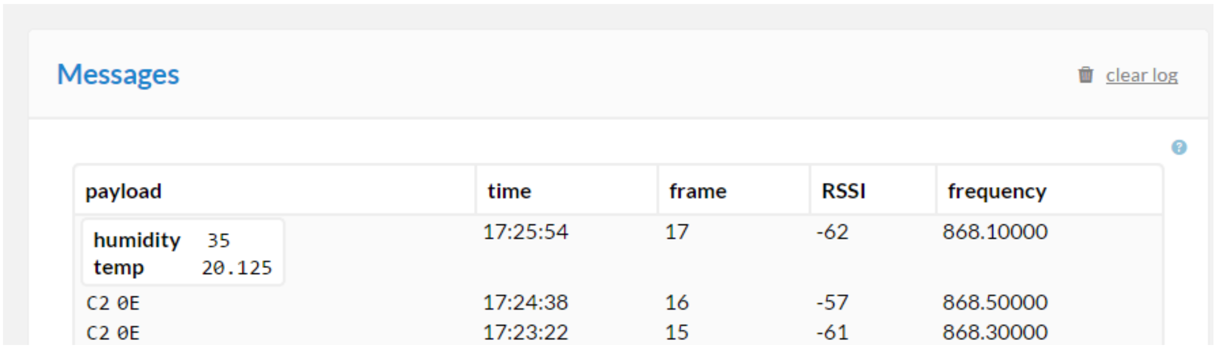
```
function (bytes) {
var humi = 20+5*((bytes[1] >> 2) & 0x0F);
var temperature = -2400+6.25*(((bytes[1] & 0x03) << 8) | bytes[0]);
return {
humidity: humi,
temp: temperature / 100.0
};
}
```



```
function (bytes) {
 var humi = 20+5*((bytes[1] >> 2) & 0x0F);
 var temperature = -2400+6.25*(((bytes[1] & 0x03) << 8) | bytes[0]);
 return {
   humidity: humi,
   celcius: temperature / 100.0
 };
}
```

And Save the function.

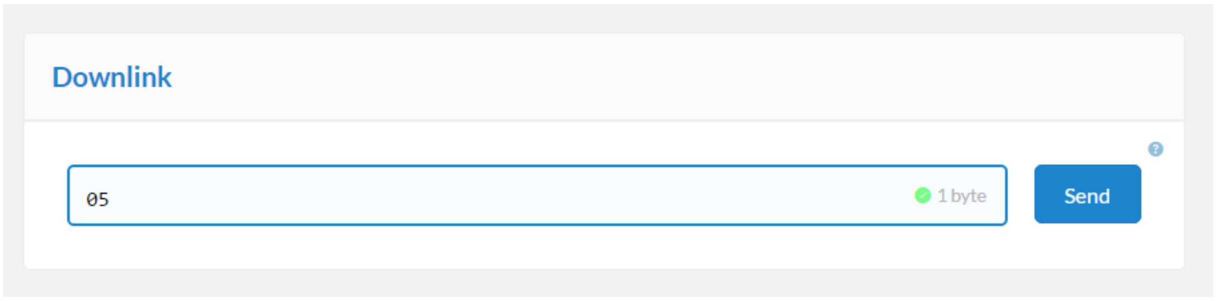
Return to your sensor and look what happens:



payload	time	frame	RSSI	frequency
humidity 35 temp 20.125	17:25:54	17	-62	868.10000
C2 0E	17:24:38	16	-57	868.50000
C2 0E	17:23:22	15	-61	868.30000

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 .

To send bytes to our node we can use the 'Download' section. 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).

Light Sensor

The light sensor acts like a temperature sensor. It is also under i2C control. Connect it like described in the sensor test section and load the ttn_lux example. You can create another sensor to follow the OTAA procedure once again.

To get the right values we have to change the decoder function:

```
function (bytes)
{
var lux = (((bytes[1] ) << 8) | bytes[0]);
return { lux: lux
};
}
```

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 = 8,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2,5, LMIC_UNUSED_PIN}, //DIO0 and DIO1 connected
};
```

And even more…

- On A0 you can measure the battery voltage. It is between a divider network 100k-10K.
- You can add professional Antenna’s
- On the 4 pin connector you can add a GPS or other general purpose I/O
- Support: https://www.thethingsnetwork.org/forum/

Thanks to Stuart Robinson for his PCB design and sharing his manuals: www.loratracker.uk and http://www.loratracker.uk/?p=30 (we use the RFM98 version of the board, replacing the RFM98 for RFM95).

Frank Beks

November 2016