

Manual Nexus LoRaWAN low power

Date: 9-10-2017
Version: 1.0
Title: Manual Nexus LoRaWAN low power

1 Revision history

Version	Date	Changes
1.0	16-02-2017	First release

1.1 Contact

Ideetron B.V.
Dropsstraat 81
3941 JL Doorn
Nederland

Tel: +31 (0)343 769094
E-mail: info@ideetron.nl

2 Table of contents

1	REVISION HISTORY	2
1.1	Contact	2
2	TABLE OF CONTENTS	3
3	INTRODUCTION	4
4	ARDUINO SKETCHES	4
4.1	Demoboard	5
4.2	Keyfob	7
4.3	Pinger sketch	8
4.4	TH06	9
5	NEXUS MODIFICATION	10
6	CODE	11
6.1	LoRaWAN structure	11
6.2	Supply voltage measurement	13
6.3	IDE	14
7	THE THINGS NETWORK	14
8	CAYENNE	16
8.1	TTN configuration	16
8.2	Cayenne	17

3 Introduction

This document describes the LoRaWAN sketches for the nexus board from IdeeTron for low power LoRaWAN, which is derived from the original Nexus LoRaWAN sketch which was provided when the Nexus LoRaWAN board was launched. The Nexus LoRaWAN development board was designed to provide an LoRaWAN enabled board which is compatible with the Arduino environment to enable hobbyist and makers to create their own LoRaWAN devices.

The original sketch wasn't low power as the atmega328p was always on and the RFM95 was always receiving. The low power sketches provide examples for low power operation of the RFM95 and the atmega328p in Class A mote type with sensor examples for the Nexus baseboard, which is an add-on board with a few low power sensors and pushbuttons. Power consumption is reduced from ~10 milliamps to 1.73 – 3,0 micro amps, depending on which Arduino sketch and sensors are used.

The sketches use The Things Network (TTN) for middle ware and Cayenne MyDevices for visualisation. Cayenne MyDevices' Low Power Protocol (LPP) is implemented, which enables the use of MyDevices to visualise the data in a dashboard.

4 Arduino Sketches

There are four sketches available for the Nexus LoRaWAN low power:

1. Demoboard sketch.
2. Keyfob sketch.
3. Pinger sketch.
4. TH06 sketch.

All sketches except the Pinger make use of the Nexus demo board, but can easily be adapted to add your own sensor to either sketch. All sketches use OTAA to connect with the TTN, which will provide the following output on the serial port when OTAA completes:

```
Port closed
Opening port
Port open
DS2401 DEV EUI: 8 bytes; 01 22 DC 0C 19 00 00 4C
Sending Join Request: 0
Sending Join Request: 1
Join Accept Message: 29 bytes; 20 A7 B3 1D 13 00 00 A0 29 01 26 03 01 18 4F 84 E8 56 84 B8 5E 84 88
66 84 58 6E 84 00
CFlist
Freq: 867100000      14206603
Freq: 867300000      14209879
Freq: 867500000      14213157
Freq: 867700000      14216433
Freq: 867900000      14219711
Device Address: 4 bytes; 26 01 29 A0
Network Session Key: 16 bytes; DE 22 F9 B0 12 5A 6E BE 87 F1 62 A8 71 C6 E7 7E
Application Session Key: 16 bytes; 81 A1 89 D6 75 18 4B 5A 71 8A 01 FE 2D B7 18 96
```

4.1 Demoboard

This sketch demonstrates all features of the Nexus board and demo board:

- Uses Over The Air Activation (OTAA) with The Thing Network. An unique ID is retrieved from the DS2401p to be used as a Device EUI number for OTAA.
- Flash memory read, write and low power configuration of the w25x40cl SPI flash chip on the Nexus Board.
- MCP7940 Real Time Clock which periodically wakes the Nexus board each minute or on a configurable time interval in minutes from the lowest possible sleep mode of the atmega328p.
- Movement detection with a vibration sensor, which will wake the Nexus when in sleep mode and increments the movement count.
- Measures the LDR value with analogRead. Measuring the supply voltage is also possible with a little editing of the Arduino analogRead function in the Arduino enviroment.
- An unconfirmed message is send when one or both pushbutton on the demo board are pressed.
- All data messages are transmitted as confirmed up, which will be replied with an unconfirmed message down by The Things Network. This provides the option to change the message interval with the back-end by configuring the reply message in TTN.
- All LoRaWAN messages use the MyDevices LPP, which makes visualisation in MyDevice Cayenne dashboard possible.

TTN data:

time	counter	port						
15:07:43		0						
15:07:43	40	1	confirmed	payload: 00 67 00 E3 01 68 78 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 60	temperature_0: 22.7	
15:06:43		0						
15:06:43	39	1	confirmed	payload: 00 67 00 E3 01 68 79 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 60.5	temperature_0: 22.7	
15:05:43		0						
15:05:43	38	1	confirmed	payload: 00 67 00 E3 01 68 78 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 60	temperature_0: 22.7	
15:04:43		0						
15:04:43	37	1	confirmed	payload: 00 67 00 E3 01 68 78 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 60	temperature_0: 22.7	
15:03:43		0						
15:03:43	36	1	confirmed	payload: 00 67 00 E3 01 68 78 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 60	temperature_0: 22.7	
15:02:43		0						
15:02:43	35	1	confirmed	payload: 00 67 00 E3 01 68 77 02 03 01 40	analog_out_2: 3.2	relative_humidity_1: 59.5	temperature_0: 22.7	
15:01:43		0						

Serial sketch output:

11:32:0 day:1 9/10/17
Temperature: 22.7558822631, Humidity: 60.1430358886
Movement count: 0
Supply Voltage: 3.2091169357
LDR: 730 POT: 562
Received data:0 bytes;

11:33:0 day:1 9/10/17
Temperature: 22.7451629638, Humidity: 60.0743713378
Movement count: 0
Supply Voltage: 3.2091169357
LDR: 735 POT: 562
Received data:0 bytes;

11:34:0 day:1 9/10/17
Temperature: 22.7558822631, Humidity: 60.0743713378
Movement count: 0
Supply Voltage: 3.2091169357
LDR: 731 POT: 562
Received data:0 bytes;

11:35:0 day:1 9/10/17
Temperature: 22.7773361206, Humidity: 59.9980773925
Movement count: 0
Supply Voltage: 3.2091169357
LDR: 733 POT: 562
Received data:0 bytes;

4.2 Keyfob

The Keyfob sketch use OTAA to connect to the Thing Network and transmits an LPP message when a button is pressed for 1 second. An unconfirmed message is send when one or both pushbutton on the demo board are pressed. A LoRaWAN message will be send with a digitalOutput message. The digitalOutput value will be 1 when the left button is pressed, it will be 2 when the right button is pressend and shall be 3 when both buttons are pressed.

Power consumption was 1.73µA in sleep mode.

Data example in TTN:

Filters				
<input type="button" value="uplink"/> <input type="button" value="downlink"/> <input type="button" value="activation"/> <input type="button" value="ack"/> <input type="button" value="error"/>				
	time	counter	port	
▲	13:57:10	9	1	payload: 03 01 03 digital_out_3: 3
▲	13:57:07	8	1	payload: 03 01 01 digital_out_3: 1
▲	13:57:04	7	1	payload: 03 01 02 digital_out_3: 2
▲	13:57:01	6	1	payload: 03 01 03 digital_out_3: 3
▲	13:56:58	5	1	payload: 03 01 03 digital_out_3: 3
▲	13:56:54	4	1	payload: 03 01 02 digital_out_3: 2
▲	13:56:51	3	1	payload: 03 01 01 digital_out_3: 1
▲	13:56:47	2	1	payload: 03 01 02 digital_out_3: 2

4.3 Pinger sketch

The Pinger sketch uses the RTC to wake-up each minute and transmit an 8-bit counter value as an digitalOutput LPP type. This sketch disables the flash chip to reduce power consumption and would be a good starting point for creating your own low power Nexus sketch.

Serial output:

```

Opening port
Port open
DS2401 DEV EUI: 8 bytes; 01 22 DC 0C 19 00 00 4C
Sending Join Request: 0
Sending Join Request: 1
Join Accept Message: 29 bytes; 20 2C 05 52 13 00 00 16 21 01 26 03 01 18 4F 84 E8 56 84 B8 5E 84 88 66 84 58 6E
84 00
CFlist
Freq: 867100000 14206603
Freq: 867300000 14209879
Freq: 867500000 14213157
Freq: 867700000 14216433
Freq: 867900000 14219711
Device Address: 4 bytes; 26 01 21 16
Network Session Key: □ Application Session Key: 16 bytes; CD 98 00 34 ED A7 94 AF FA 11 A9 C0 4A 1E B2 86
Application Session Key: 16 bytes; CD 08 45 22 3B 6D C9 E9 0A 7B 74 0A 82 57 B4 16
10:6:7 day:1    9/10/17

10:6:7 day:1    9/10/17
Ping: 0

10:7:0 day:1    9/10/17
Ping: 1

10:8:0 day:1    9/10/17
Ping: 2

10:9:0 day:1    9/10/17
Ping: 3

10:10:0 day:1   9/10/17
Ping: 4
    
```

TTN data:

time	counter	port	dev id: nexusredevelopment	payload: 00 01 05	digital_out_0: 5
▲ 15:54:14	5	1	dev id: nexusredevelopment	payload: 00 01 05	digital_out_0: 5
▲ 15:53:14	4	1	dev id: nexusredevelopment	payload: 00 01 04	digital_out_0: 4
▲ 15:52:14	3	1	dev id: nexusredevelopment	payload: 00 01 03	digital_out_0: 3
▲ 15:51:14	2	1	dev id: nexusredevelopment	payload: 00 01 02	digital_out_0: 2
▲ 15:50:14	1	1	dev id: nexusredevelopment	payload: 00 01 01	digital_out_0: 1
▲ 15:49:20	0	1	dev id: nexusredevelopment	payload: 00 01 00	digital_out_0: 0

4.4 TH06

This sketch wakes-up each minute, measures the temperature, humidity, supply voltage and transmits the results in LPP format to the Things Network. Measuring the supply voltage does require some adjustments in the Arduino analogRead() code to function properly.

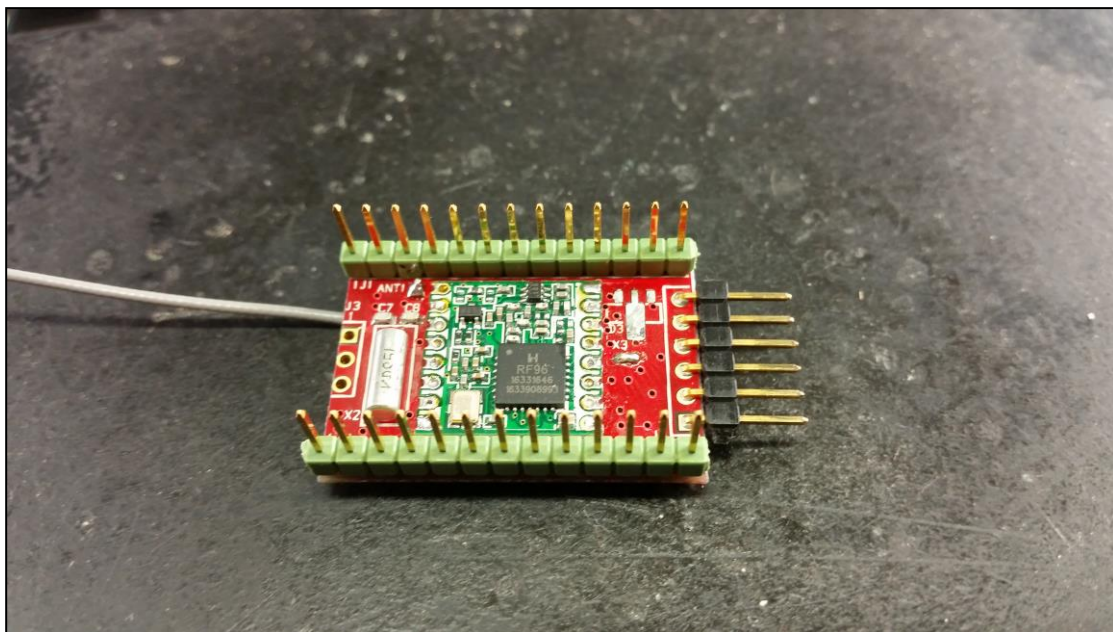
The LoRaWAN messages will be as following:

time	counter	port						
11:31:24	16	1	confirmed	payload: 00 67 00 EA 01 68 75 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 58.5	temperature_0: 23.4	
11:30:24	15	1	confirmed	payload: 00 67 00 EA 01 68 76 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 59	temperature_0: 23.4	
11:29:24	14	1	confirmed	payload: 00 67 00 EB 01 68 75 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 58.5	temperature_0: 23.5	
11:28:24	13	1	confirmed	payload: 00 67 00 EB 01 68 76 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 59	temperature_0: 23.5	
11:27:24	12	1	confirmed	payload: 00 67 00 EB 01 68 75 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 58.5	temperature_0: 23.5	
11:26:24	11	1	confirmed	payload: 00 67 00 EB 01 68 75 02 03 01 23	analog_out_2: 2.91	relative_humidity_1: 58.5	temperature_0: 23.5	
11:25:24	10	1	confirmed	payload: 00 67 00 EC 01 68 7C 02 03 01 25	analog_out_2: 2.93	relative_humidity_1: 62	temperature_0: 23.6	

5 Nexus modification

In order to achieve low power, the Nexus LoRaWAN board must be modified if the nexus board has been bought before the release of this document, the low power version of the nexus was not bought or when a different board is used. On the default Nexus boards is a voltage regulator (D3) that can be used when a power source higher than 3.6V is used, which would otherwise damage the flash, RTC and TH06 Chips. For the Nexus low power version we assume a battery cell, battery pack or the FTDI programming cable is used to power the Nexus, so the regulator isn't needed. (be careful with Lipo packs, which have an output voltage of 4,2 Volt when fully charged and thus will result in possible permanent damage.)

When the regulator doesn't supply the voltage, but another supply does, it consumes about 2.2 milliamps in reverse leakage current. Therefore the regulator next to the RFM95 module (see the picture below) must be removed with a soldering iron if low power consumption needs to be achieved.



6 Code

6.1 LoRaWAN structure

The Nexus LoRaWAN Low Power code is created from the original Nexus LoRaWAN sketch, but now uses a lot less variables and instead uses a single structure and a few class for the LoraMac and LPP for the LoRaWAN functionality. In order to change transmission or receive parameters the lora structure must be used to access the necessary variables or the variables can be configured in the lorawan_def.h file.

The sLoRaWAN structure is as following:

```
typedef struct
{
    eLoRaWAN_MOTE_CLASS Mote_Class           = CLASS_A;           //
    Mote Class, only CLASS A or CLASS C are supported.
    eMOTE_NETWORK_JOIN  activation_method     = OVER_THE_AIR_ACTIVATION;
    eBACKENDS           back_end             = THE_THINGS_NETWORK;
    bool                Channel_Hopping_enabled = false;
    volatile uint16_t   timeslot;
    // List of additional channels in hexadecimal format pre-calculated for the RFM
    frequency register
    eCHANNEL_LIST CH_list =
    {
        .channel = {0},
        .index   = 0,
        .rx1_dr_offset = 0,
        .rx2_dr   = 0,
        .rx_delay = 0,
        .channel_hopping_on = true
    };

    /* Session parameters for the current session */
    sLoRa_Session Session =
    {
        .NwkSKey = {0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6,
                    0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C},
        .AppSKey = {0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6,
                    0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C},
        .DevAddr = {0x3A, 0x12, 0x34, 0x56},
        .frame_counter_down = 0,
        .frame_counter_up   = 0,
        .Receive_delay      = RECEIVE_DELAY2,
        .Transmit_Power     = 0x0F,
        .TxChDr = {SF09_BW125kHz, CH00_868_100},
        .RxChDr = {SF09_BW125kHz, CH10_869_525}
    };

    /* OTAA configuration and encryption parameters */
    sLoRa_OTAA OTAA =
    {
        .DevEUI = {0}, // If the DS2401 is used, set the DEV EUI to zero.

        .AppEUI = {0x70, 0xB3, 0xD5, 0x7E, 0xF0, 0x00, 0x3E, 0x0C}, //
        TTN Ideetron Application EUI, change this to your own APP EUI code
        .AppKey = {0xB3, 0x23, 0xC2, 0xFD, 0xDF, 0x34, 0x7B, 0x3D,
                  0xA2, 0x04, 0xF2, 0x0E, 0x01, 0x3A, 0xA4, 0x59},
    };
};
```

```
.DevNonce = {0},
.AppNonce = {0},
.NetID     = {0},
.OTAAdone = false,
.JoinDelay = JOIN_DELAY_2,
.Transmit_Power = 0x0F,
.TxChDr = {SF10_BW125kHz, CH00_868_100},
.RxChDr = {SF09_BW125kHz, CH10_869_525}
};

/* Message structure for both receive and transmitting */
sLoRa_Message TX =
{
    .MAC_Header    = INIT_VAL,
    .DevAddr       = {0},
    .Frame_Control = 0,
    .Frame_Counter = 0,
    .Frame_Port    = 0,
    .Frame_Options = {0},
    .MIC           = {0},
    .Confirmation = CONFIRMED,
    .retVal        = INIT,
    .Count         = 0,
    .Data          = {0}
};
sLoRa_Message RX =
{
    .MAC_Header    = INIT_VAL,
    .DevAddr       = {0},
    .Frame_Control = 0,
    .Frame_Counter = 0,
    .Frame_Port    = 0,
    .Frame_Options = {0},
    .MIC           = {0},
    .Confirmation = CONFIRMED,
    .retVal        = INIT,
    .Count         = 0,
    .Data          = {0}
};
}sLoRaWAN;
```

In order to change the spreadingfactor in the code you would need to access the structures element before being able to change the datarate. For example, in order to change the transmission data rate to SF12 and the transmit channel to 868_300:

```
lora.Session.TxChDr.datarate = SF12_BW125kHz;
lora.Session.TxChDr.channel = CH01_868_300;
```

The `sLoRa_Message TX` and `sLoRa_Message RX` structure elements are used to hold and construct the lorawan messages. For example to create a message to transmit:

```
lora.TX.Data[0] = 0x01;
lora.TX.Data[1] = 0x02;
lora.TX.Data[2] = 0x03;
lora.TX.Data[3] = 0x04;
lora.TX.Count = 4; // Four bytes to transmit

// Transmit the created message as a confirmed up message type and then receive the back-
// end reply.
lorawan.LORA_send_and_receive();
```

If for example the LPP format is used:

```
// Form a payload according to the LPP standard to transmit the Temperature, Humidity and
supply Voltage.
LPP.clearBuffer();
LPP.addTemperature(0x00, TH06.Temperature);
LPP.addRelativeHumidity(0x01, TH06.Humidity);
LPP.addAnalogOutput(0x02, app.SupplyVoltage);

// Transmit the created message as a confirmed up message type and then receive the back-
end reply.
lorawan.LORA_send_and_receive();
```

The use of a structure makes the code a lot tidier than the original sketch, but you'll need to access the right elements to select the variable that needs to be changed.

6.2 Supply voltage measurement

A popular question is the ability to measure the supply voltage of the Nexus board when using a battery. The default analogRead function uses its supply voltage as reference, so it's not possible to read the supply voltage directly or with a divider. To make it possible, the following changes need to be made to the analog Read code under your Arduino installation.

Open the wiring_analog.c file under C:\Program Files (x86)\Arduino\hardware\arduino\avr\cores\arduino\wiring_analog.c with a text editor which has administrator rights to change the files contents and add or change the green highlighted text:

```
int analogRead(uint8_t pin)
{
    uint8_t low, high;
    if((pin & 0x80) == 0x00)
    {
        #if defined(analogPinToChannel)
        #if defined(__AVR_ATmega32U4__)
            if (pin >= 18) pin -= 18; // allow for channel or pin numbers
        #endif
            pin = analogPinToChannel(pin);
        #elif defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
            if (pin >= 54) pin -= 54; // allow for channel or pin numbers
        #elif defined(__AVR_ATmega32U4__)
            if (pin >= 18) pin -= 18; // allow for channel or pin numbers
        #elif defined(__AVR_ATmega1284__) || defined(__AVR_ATmega1284P__) || defined(__AVR_ATmega644__) ||
            defined(__AVR_ATmega644A__) || defined(__AVR_ATmega644P__) || defined(__AVR_ATmega644PA__)
            if (pin >= 24) pin -= 24; // allow for channel or pin numbers
        #else
            if (pin >= 14) pin -= 14; // allow for channel or pin numbers
        #endif
    }

    #if defined(ADCSRB) && defined(MUX5)
        // the MUX5 bit of ADCSRB selects whether we're reading from channels
        // 0 to 7 (MUX5 low) or 8 to 15 (MUX5 high).
        ADCSRB = (ADCSRB & ~(1 << MUX5)) | (((pin >> 3) & 0x01) << MUX5);
    #endif

    // set the analog reference (high two bits of ADMUX) and select the
    // channel (low 4 bits). this also sets ADLAR (left-adjust result)
    // to 0 (the default).
    #if defined(ADMUX)
    #if defined(__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
        ADMUX = (analog_reference << 4) | (pin & 0x0F);
    #else
        ADMUX = (analog_reference << 6) | (pin & 0x0F);
    #endif
}
```

```
#endif  
#endif
```

This code modification will bypass the pin to channel conversion when bit 7 of the pin variable is set, allowing to directly modify the channel register to select the ADC input to other inputs such as the 1.1V reference. In the function `double read_supply_voltage(void)` from `baseboard.h` the internal 1.1V reference is measured with the supply voltage as reference. The supply voltage is then calculated and returned as a double.

6.3 IDE

The low power Nexus code is written in Atmel Studio with the vMicro add-on (free version), as the Arduino IDE can be quite cumbersome in formatting, file handling, structure name autocompletion and other features that make code development easier.

For a tutorial and a walkthrough see: <http://www.visualmicro.com/>

7 The Things Network

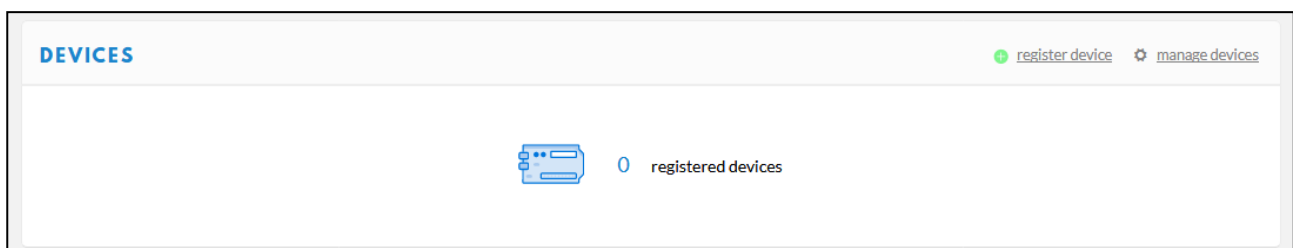
As of writing has Semtech closed it's back-end for testing for users without any form of subscription or payment, so only TTN is available for testing. Therefore only TTN is explained on how to add a mote.

To add a mote whit OTAA to TTN follow these steps:

1. Sing in or create an account on the TTN site.
2. Go to Applications.
3. Click "add application".



4. Fill in the fields and press "Add application".
5. After registering the application you will see a screen with all the information regarding the application. Now click on "register device".



6. Fill in the Device ID with a description without whitespaces or uppercase letters. Set the Device EUI to be generated or fill in the DEV EUI from the DS2401, which is printed to the serial port on start-up. Enter this hexadecimal value: `0xB323C2FDDF347B3DA204F20E013AA459` for

the Application key for the example sketches or a custom key and add that key to the lorawan_def.h file.

REGISTER DEVICE

[bulk import devices](#)

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

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

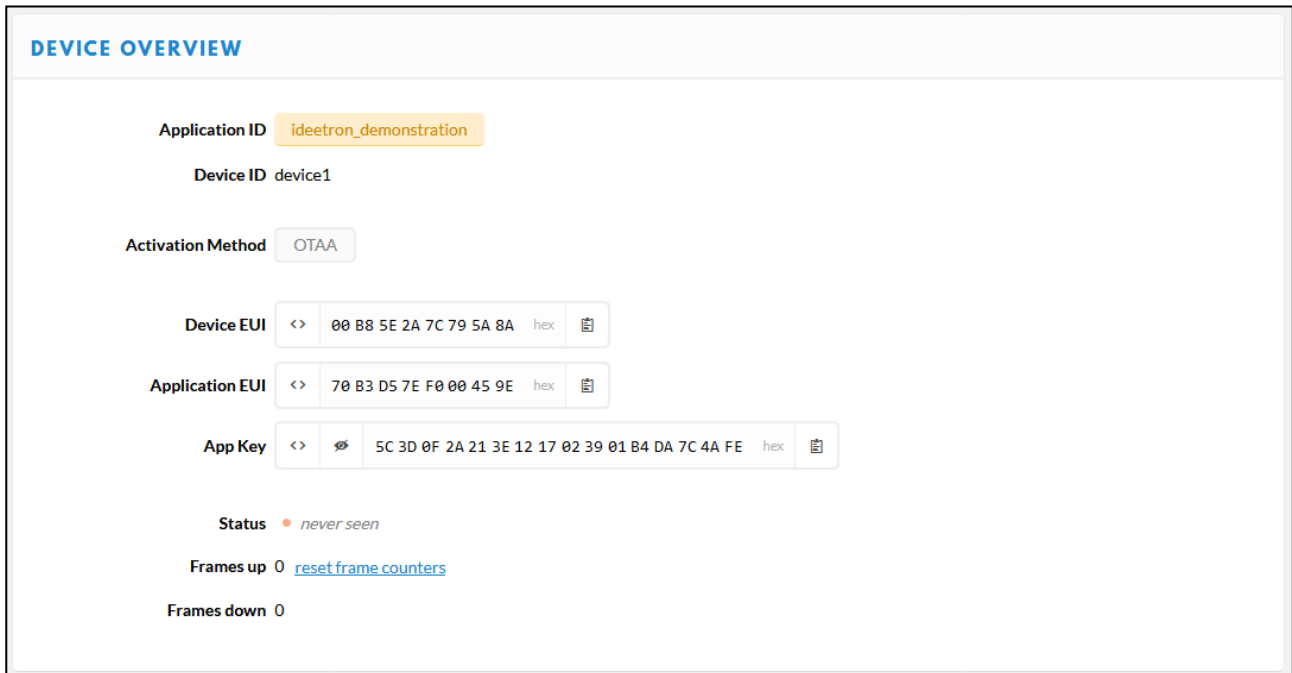
App Key
The App Key will be used to secure the communication between you device and the network.

App EUI

Cancel Register

- After registering the device, you will see an overview of the device settings. You will see the Device EUI, Application EUI and Application Key. Copy the App EUI to the Nexus code in the lorawan_def.h file. Change the code shown below on line 275 to your application EUI.

```
.AppEUI = {0x70, 0xB3, 0xD5, 0x7E, 0xF0, 0x00, 0x3E, 0x0C},
```



DEVICE OVERVIEW

Application ID `ideetron_demonstration`

Device ID `device1`

Activation Method `OTAA`

Device EUI `<> 00 B8 5E 2A 7C 79 5A 8A hex`

Application EUI `<> 70 B3 D5 7E F0 00 45 9E hex`

App Key `<> 5C 3D 0F 2A 21 3E 12 17 02 39 01 B4 DA 7C 4A FE hex`

Status • *never seen*

Frames up 0 [reset frame counters](#)

Frames down 0

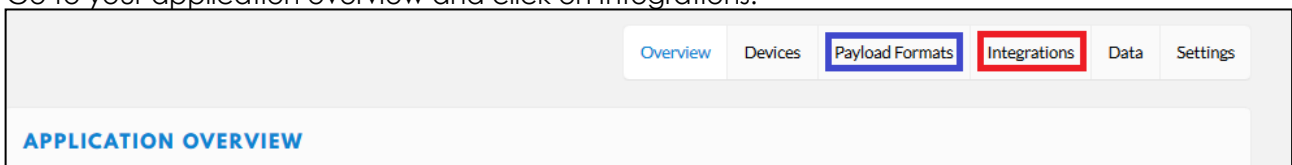
8 Cayenne

In order to add your mote to Cayenne you must do the following steps:

- Configure your application in your TTN application to transmit the data to Cayenne
- Register on Cayenne
- Add your mote to your dashboard. (which must use the LPP format)

8.1 TTN configuration

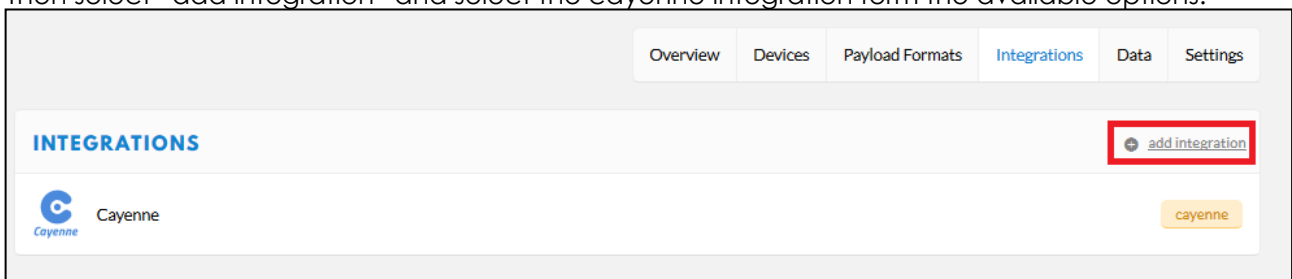
Go to your application overview and click on Integrations:



Overview Devices **Payload Formats** **Integrations** Data Settings


APPLICATION OVERVIEW

Then select "add integration" and select the cayenne integration from the available options.



Overview Devices Payload Formats **Integrations** Data Settings

INTEGRATIONS [+ add integration](#)

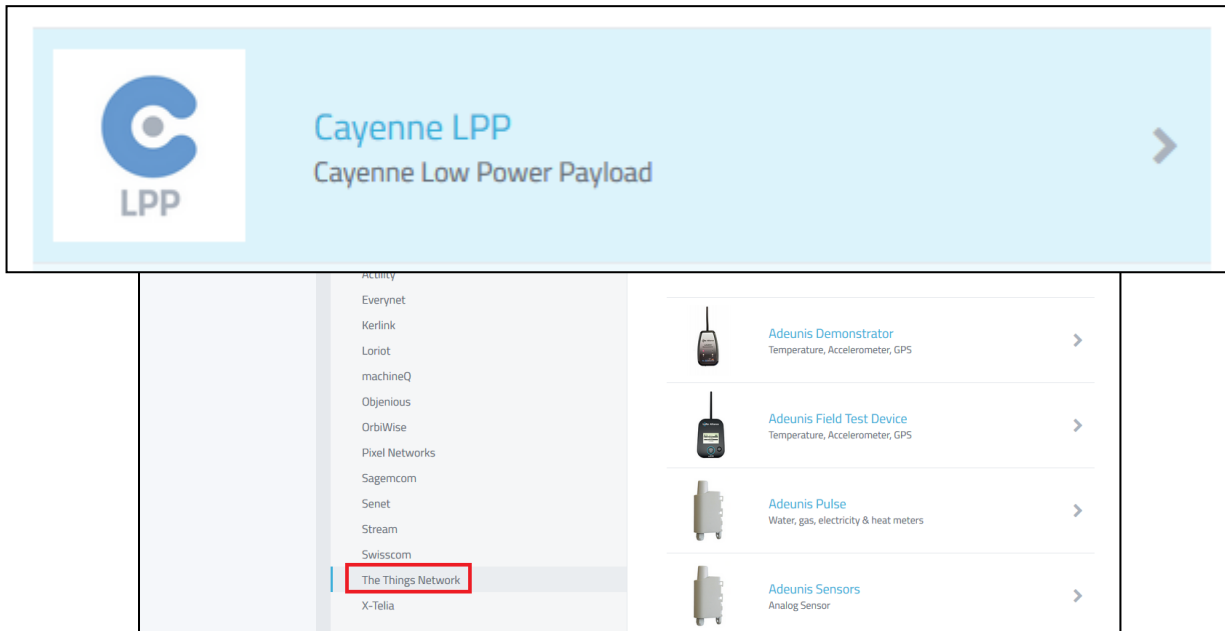
 Cayenne [cayenne](#)

In order to display the LPP format in TTN, click on "Payload formats" and select the Cayenne LPP format, which enables the value display in TTN as shown by the data output of the sketches.

8.2 Cayenne

Go to <https://mydevices.com/cayenne/signin/> and register or sign in. when in your dashboard click on the green button "Add new" and select Device/widget. Then click on LoRa (Beta) and click on The Things Network.

From the list of devices find the Cayenne LPP mote type as shown below



Fill in the Device EUI number of your mote (without any spaces and only capital letters) and add your address/ location of your mote if your mote is stationary. If there's data received by TTN from your mote, Cayenne will automatically add the widgets to display the LPP formats. For example the demo board sketch will have the following widgets:

