Lab 07: Some basic applications using LoRaWAN and The Things Network

Description:

In this lab, you will utilize LoRa Technology to connect to The Things Network (TTN), a project aimed at establishing a network for the Internet of Things (IoT). TTN operates using the LoRaWAN protocol, enabling devices to communicate with TTN through Low Power Wide Area Communication Technology. You will use the TTN console to monitor transmitting devices and send data back to them.

Detailed instructions for connecting to The Things Network are provided in this lab. This lab demonstrates how to send data to TTN and program your Arduino to perform specific tasks based on downlink data from TTN. It's essential to thoroughly follow, execute, and comprehend all the showcased functionalities in today's lab.

After completing this lab, please make sure to keep your **modified Arduino script** files and remember the credentials required for creating a **TTN account.** You may find them useful for your future projects or engaging in other intriguing labs.

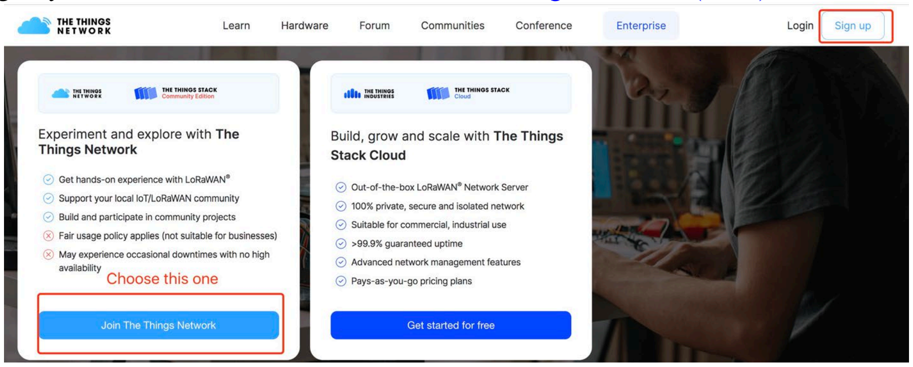
Requirements:

1. The Arduino Starter Kit
2. Dragino LoRa Shield
3. Latest Arduino LMIC library by IBM
4. The Arduino sketch for Lab07_abp and Lab07_otaa (available on LMS and MS Teams)

Setup:

The Things Network configuration

1. To begin, you need to create an account on The Things Network (TTN).



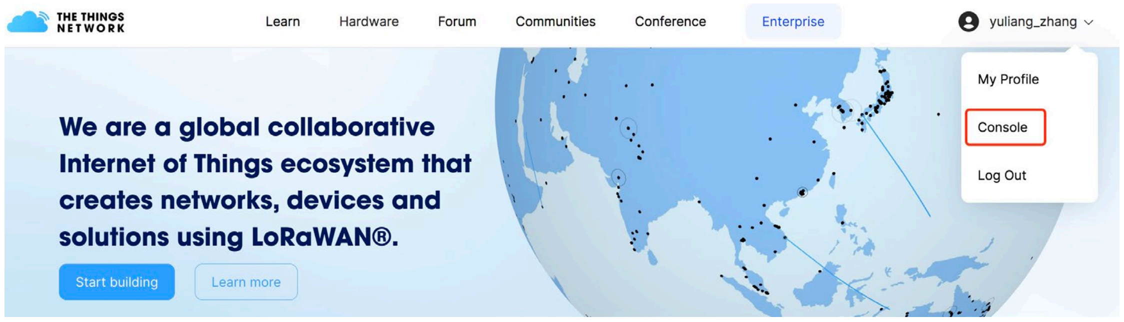
Experiment and explore with The Things Network

- Get hands-on experience with LoRaWAN®
- Support your local IoT/LoRaWAN community
- Build and participate in community projects
- Fair usage policy applies (not suitable for businesses)
- May experience occasional downtimes with no high availability

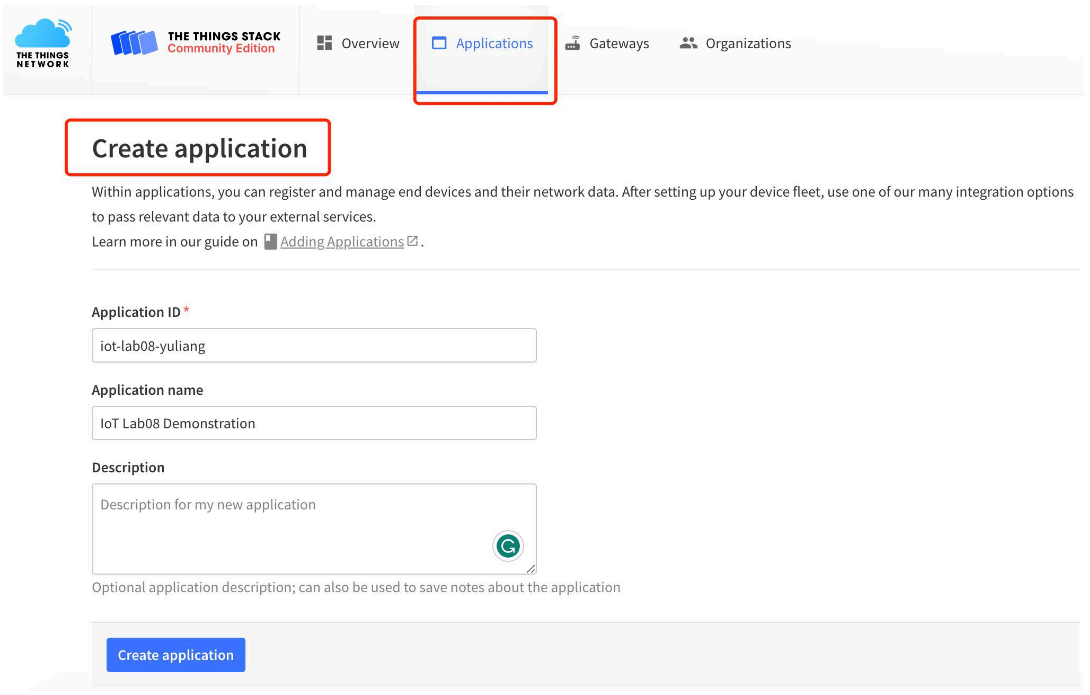
Build, grow and scale with The Things Stack Cloud

- Out-of-the-box LoRaWAN® Network Server
- 100% private, secure and isolated network
- Suitable for commercial, industrial use
- >99.9% guaranteed uptime
- Advanced network management features
- Pays-as-you-go pricing plans

2. Create an Application via TTN. Click on your name at the right up corner and select "***Console***". Select "***Australia 1***" for the cluster. Then, click "***Applications***", and then "***Create application***" to add a new application. You will see the information as shown in the following figure. Provide an application ID and a description. For example, in the following figure, the application is "***iot-lab07-yuliang***". However, you should create your own application ID, application name, and description. Select a description that makes sense to you. To proceed, you need to register a new device under this application.



We are a global collaborative Internet of Things ecosystem that creates networks, devices and solutions using LoRaWAN®.



Create application

Within applications, you can register and manage end devices and their network data. After setting up your device fleet, use one of our many integration options to pass relevant data to your external services.
Learn more in our guide on Adding Applications.

Application ID *
iot-lab08-yuliang

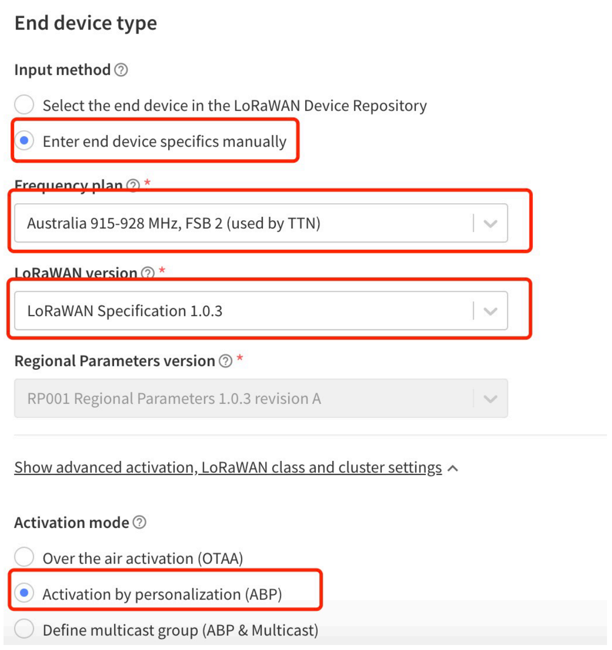
Application name
IoT Lab08 Demonstration

Description
Description for my new application
Optional application description; can also be used to save notes about the application

Create application

3. To register a device, click the "***Register end device***" button. Since we are using an Arduino UNO and LoRa Shield, we should choose the "***Enter end device specifics manually***". Then, select the "***Frequency plan***" of "*Australia 915-928 MHz, FSB 2 (used by TTN)*", and "***LoraWAN version***"

of *"LoRaWAN Specification 1.0.3"*. Under advanced setting, make sure you select the activation mode of **ABP** instead of over the air authentication (OTAA).



End device type

Input method

( ) Select the end device in the LoRaWAN Device Repository
(●) Enter end device specifics manually

Frequency plan *
Australia 915-928 MHz, FSB 2 (used by TTN)

LoRaWAN version *
LoRaWAN Specification 1.0.3

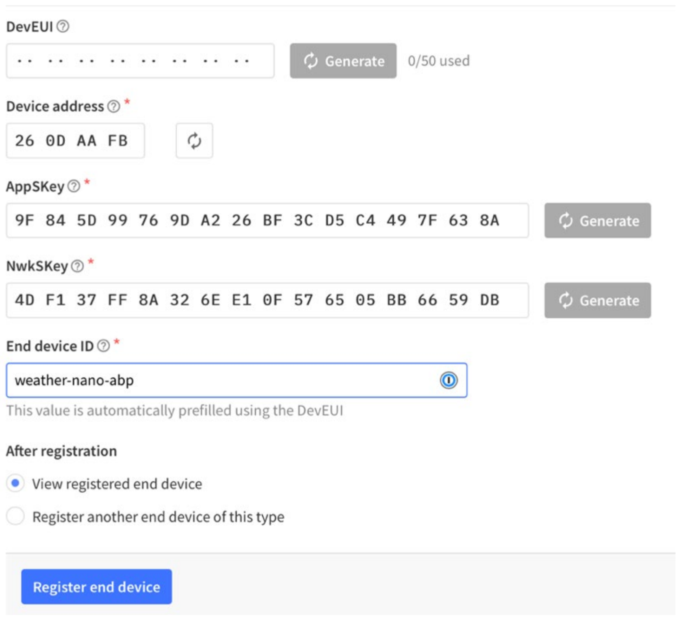
Regional Parameters version *
RP001 Regional Parameters 1.0.3 revision A

Show advanced activation, LoRaWAN class and cluster settings

Activation mode

( ) Over the air activation (OTAA)
(●) Activation by personalization (ABP)
( ) Define multicast group (ABP & Multicast)

4. Next, click the generate button to generate ***"Device address", "App Session Key (AppSKey)"*** and "***Network Session Key (NwkSKey)"***



DevEUI
`.. .. .. .. .. .. .. ..` Generate 0/50 used

Device address *
26 0D AA FB

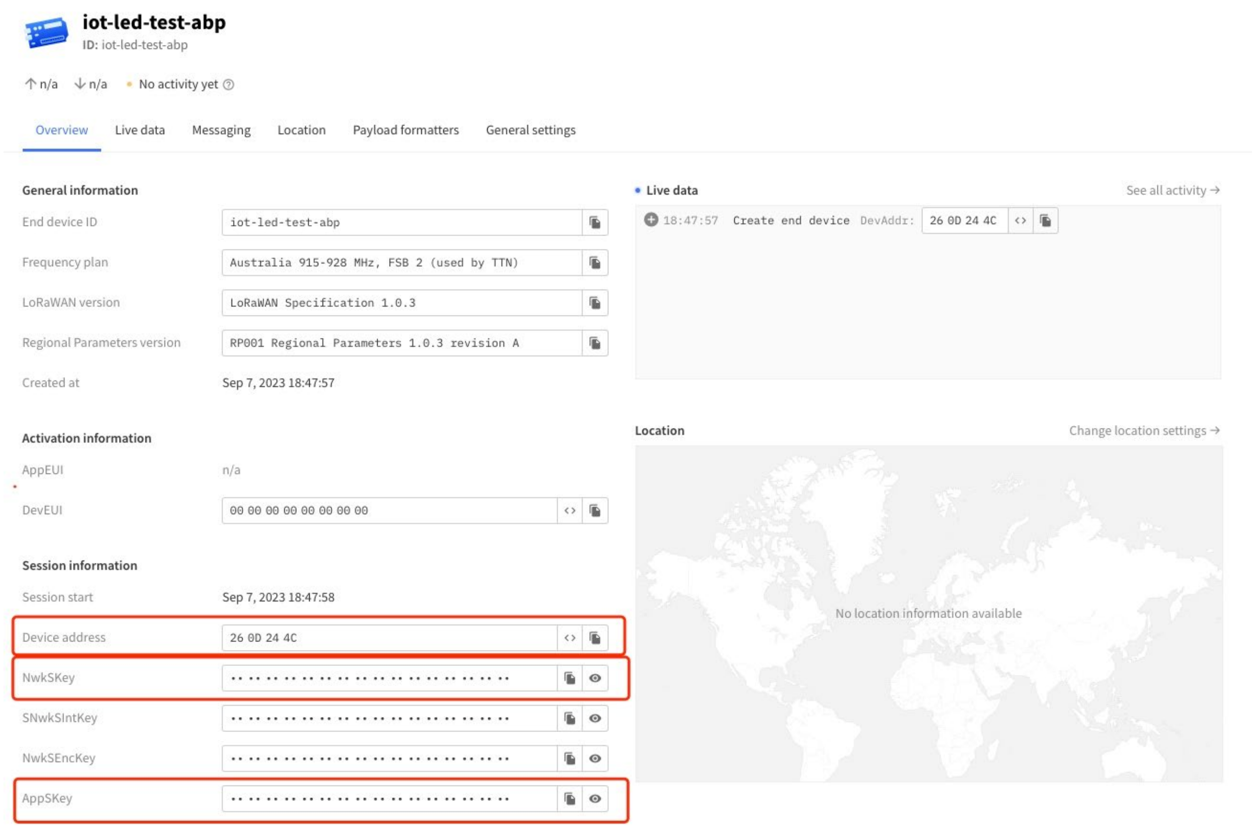
AppSKey *
9F 84 5D 99 76 9D A2 26 BF 3C D5 C4 49 7F 63 8A

NwkSKey *
4D F1 37 FF 8A 32 6E E1 0F 57 65 05 BB 66 59 DB

End device ID *
weather-nano-abp
This value is automatically prefilled using the DevEUI

After registration
(●) View registered end device
( ) Register another end device of this type

5. TTN console will provide you with the ***Network Session Key, App Session Key*** and the ***Device Address*** under "***Overview***" tab. Download and open the Lab07_abp.ino sketch file so we can add these keys to our Arduino Script. Note that the Lab07_abp.ino file should be placed inside a "Lab07_abp" folder.



iot-led-test-abp
ID: iot-led-test-abp

↑ n/a ↓ n/a • No activity yet

General information

End device ID	iot-led-test-abp
Frequency plan	Australia 915-928 MHz, FSB 2 (used by TTN)
LoRaWAN version	LoRaWAN Specification 1.0.3
Regional Parameters version	RP001 Regional Parameters 1.0.3 revision A
Created at	Sep 7, 2023 18:47:57

Activation information

AppEUI	n/a
DevEUI	00 00 00 00 00 00 00 00

Session information

Session start	Sep 7, 2023 18:47:58
Device address	26 0D 24 4C
NwkSKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
SNwkSIntKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
NwkSEncKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••
AppSKey	•• •• •• •• •• •• •• •• •• •• •• •• •• •• •• ••

Live data

18:47:57 Create end device DevAddr: 26 0D 24 4C

Location

6. Use the '*Copy*' button to copy the **Device address** you have generated in C Style from TTN Console, and replace the DEVADDR on line 66 of the ino file with the copied address. Make sure it is in correct format.
static const u4_t DEVADDR = 0x260DAAFB;

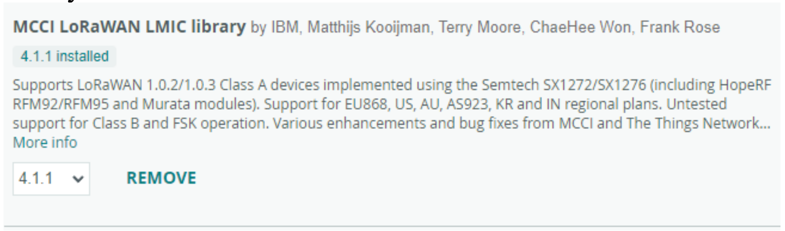
7. Use the '*Show/hide'* button to make **Network Session Key** visible on TTN Console, and use the '*change format button'* to convert it into HEX format (big-endian a.k.a msb order). Copy the key and update the NWKSKEY on line 57 of the script.
static const PROGMEM u1_t NWKSKEY[16] = { 0x4D, 0xF1, 0x37, 0xFF, 0x8A, 0xAC, 0x7D, 0xD4, 0xD5, 0x63, 0x40, 0x0B, 0x34, 0xDD, 0xB7, 0x25 };

8. Again, use the '*Show/hide'* button to make **App Session Key** visible, and use the 'change format button' to convert it into HEX format (big-endian a.k.a msb order). Copy the key and update the APPSKEY on line 61 of the script.
static const u1_t PROGMEM APPSKEY[16] = { 0x9F, 0x84, 0x5D, 0x99, 0x76, 0x1D, 0x81, 0x1E, 0x60, 0x15, 0x22, 0x94, 0x79, 0xD8, 0x7C, 0xFD };

9. Go to "***General settings***" tab, under "***Network layer***", click "***expand***". Under the "***advanced MAC settings***", tag the "***Resets Frame Counters***" and press save**.** The configuring of TTN is complete. Do not close the console window as we will be using it later in the Lab.

Downloading and configuring the LMIC Library

1. We need the latest LMIC library for our sketch to work.
    a. Go to: Sketch -> Include Library -> Manage libraries.
    b. Search for 'MCCI LoRaWAN LMIC library' at the top search bar, and locate the following library



MCCI LoRaWAN LMIC library by IBM, Matthijs Kooijman, Terry Moore, ChaeHee Won, Frank Rose
4.1.1 installed
Supports LoRaWAN 1.0.2/1.0.3 Class A devices implemented using the Semtech SX1272/SX1276 (including HopeRF RFM92/RFM95 and Murata modules). Support for EU868, US, AU, AS923, KR and IN regional plans. Untested support for Class B and FSK operation. Various enhancements and bug fixes from MCCI and The Things Network...
More info

2. Select the version 4.1.1 and install the library.

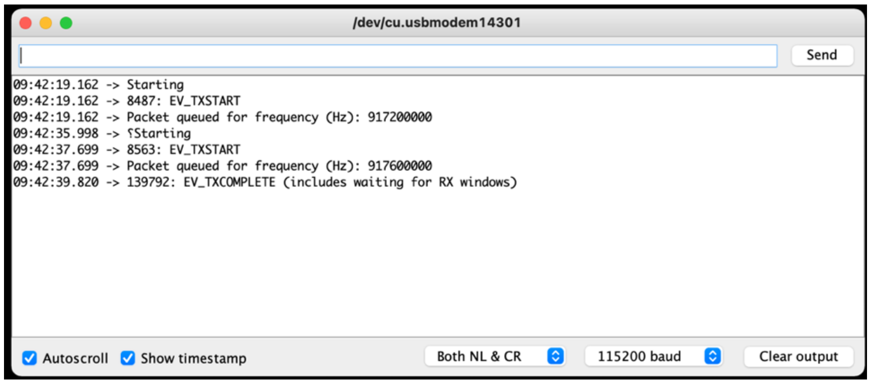
3. This library comes with default configuration set for US. We have to change it to AU.
    a. Go to following location on your hard drive:
    b. Arduino -> libraries -> MCCI_LoRaWAN_LMIC_library-> project_config
    c. Open the 'lmic_project_config.h' file, and modify the 3rd and 4th line and save the file.
    d. //#define CFG_us915 1
    e. #define CFG_au915 1

Hardware setup and demonstration of basic functionality

1. Carefully place Dragino Lora shield on top of Arduino UNO. Make sure that pins map correctly.

2. Place the Antenna on the Dragino shield.

3. Connect the Arduino UNO to your laptop and upload the modified Lab07_abp.ino script. Open the 'Serial Monitor' of Arduino software and observe the LoRa packets being transmitted after 'TX_INTERVAL' set in the script. Make sure the baud rate is 115200.

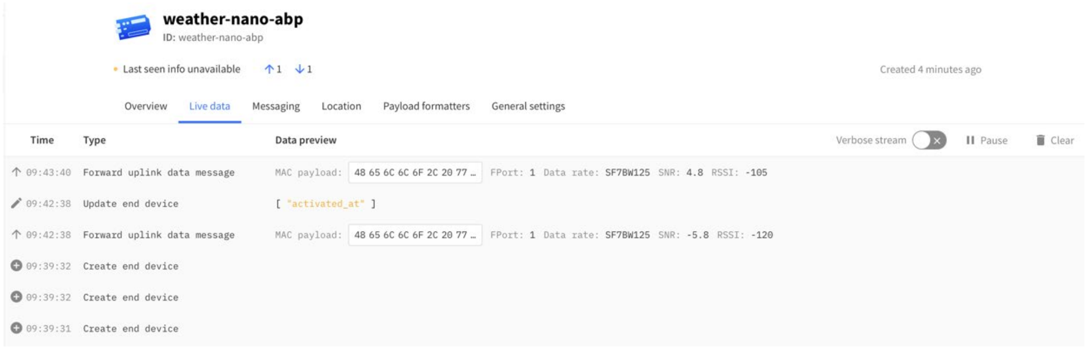


```
09:42:19.162 -> Starting
09:42:19.162 -> 8487: EV_TXSTART
09:42:19.162 -> Packet queued for frequency (Hz): 917200000
09:42:35.998 -> ⸮Starting
09:42:37.699 -> 8563: EV_TXSTART
09:42:37.699 -> Packet queued for frequency (Hz): 917600000
09:42:39.820 -> 139792: EV_TXCOMPLETE (includes waiting for RX windows)
```

4. You should see the 'Status' of device in Overview window of TTN Console change from 'Last seen info unavailable' to a X seconds ago, once a packet has been received by TTN console.

   ● Last seen info unavailable   ● Last seen 29 seconds ago

5. Go to the Live data window and observe the data appear in real time as it is transmitted to TTN via LoRaWAN.



```
Time        Type                          Data preview
↑ 09:43:40  Forward uplink data message   MAC payload: 48 65 6C 6C 6F 2C 20 77 …  FPort: 1  Data rate: SF7BW125  SNR: 4.8  RSSI: -105
  09:42:38  Update end device             [ "activated_at" ]
↑ 09:42:38  Forward uplink data message   MAC payload: 48 65 6C 6C 6F 2C 20 77 …  FPort: 1  Data rate: SF7BW125  SNR: -5.8  RSSI: -120
  09:39:32  Create end device
  09:39:32  Create end device
  09:39:31  Create end device
```

6. We desire for the data to be transmitted every 60 seconds; however, due to processing and transmission delays at Arduino end, this interval is more than 60 seconds. Please check if you can adjust the 'TX_INTERVAL' parameter in Arduino script to bring the actual interval closer to 60 seconds.

7. This data we see at TTN console is quite meaning less as it shows the ASCII only. However, using the '***Payload formatters'***, we can convert it into a readable format. Going to the tab of "***Payload formatters",*** click the ***Uplink*** tab, and select the "***Formatter type***" as "***Custom Javascript formatter***". Paste the following JavaScript into decoder part and press ***"Save changes".***

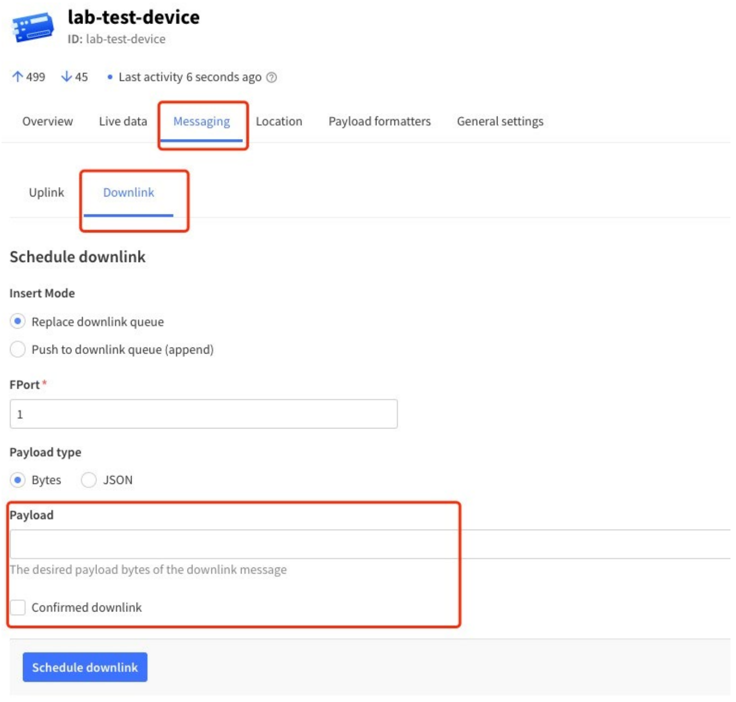
```
function Decoder(bytes, port) {

  // Decode plain text; for testing only
  return {
    myText: String.fromCharCode.apply(null, bytes)
  };
}
```

8. Go to ***"Live data"*** tab and observe the next payload which comes in. This is what the Arduino script is actually transmitting, go and have a look at the mydata[] array in the script.

9. You can use basic JavaScript to do a lot of cool stuff with formatting the payloads, extracting the information etc. Plenty of help and examples are available online if you have never worked with JavaScript before.

10. Now to check another functionality, go to your device and select "***Messaging***", and click "***Downlink***" section. (Not the Simulate Uplink one).



lab-test-device
ID: lab-test-device
↑499 ↓45 • Last activity 6 seconds ago

Schedule downlink

Insert Mode
- [✓ Replace downlink queue]
- Push to downlink queue (append)

FPort*: 1

Payload type: [✓ Bytes] JSON

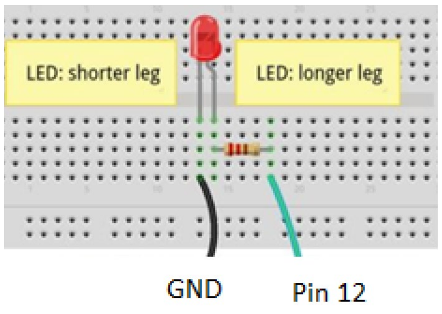
Payload

The desired payload bytes of the downlink message

☐ Confirmed downlink

11. Enter any Hex number of one bytes e.g. A1 in the "***Payload***" text box, check the '***Confirmed downlink'*** tickbox and press '***Schedule downlink'***. The Arduino should receive the byte, and transmit it back after incrementing it by 1 in the next scheduled transmission window. Validate this in "***Live data***" tab.

12. Coming to some basic electronics, the Arduino script does something specific when it receives 'D0' or 'D1' from the console. Make a small circuit with an LED and 330 Ohm resistor in series on the bread board. Connect the Cathode of LED to the ground, and the resistor to pin 12 of Arduino/dragino.



13. See what happens to LED when you send D0 and D1 as downlink. Have a look at the script to see how it is working. (Hex to Int: 0xD0 = 208 and 0xD1 = 209)

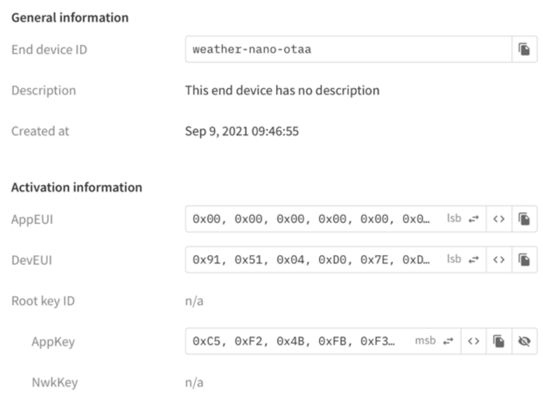
(Alternative) Over the air activation (OTAA) mode device registration

1. To register device with OTAA activation mode, add end device and register it manually.

2. Select the **LoraWAN version MAC V1.0.3** and **Frequency plan of Australia 915-928 MHz, FSB 2 (used by TTN).** Under advanced setting, make sure you select the activation mode of **OTAA** instead of activation by personalization (ABP).

3. Next, you need to generate the **DevEUI**, **AppEUI** and **AppKey** for the device. Give the end device and ID. Click register end device.

4. TTN console will show you the generated **DevEUI**, **AppEUI** and **AppKey** under "**Overview**" tab. Download and open the Lab07_otaa.ino sketch file so we can add these keys to our Arduino Script. Note that the Lab07_otaa.ino file should be placed inside a "Lab07_otaa" folder.



General information

End device ID	weather-nano-otaa
Description	This end device has no description
Created at	Sep 9, 2021 09:46:55

Activation information

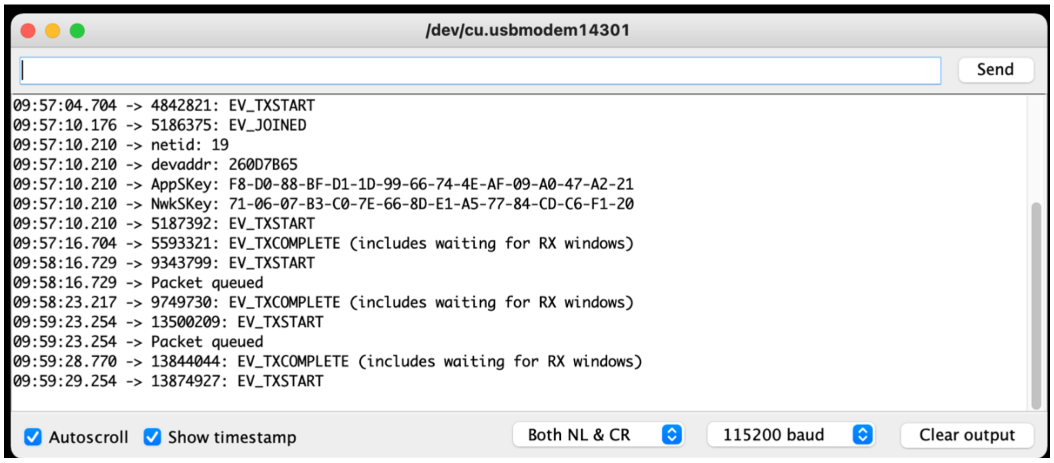
AppEUI	0x00, 0x00, 0x00, 0x00, 0x00, 0x0… lsb
DevEUI	0x91, 0x51, 0x04, 0xD0, 0x7E, 0xD… lsb
Root key ID	n/a
AppKey	0xC5, 0xF2, 0x4B, 0xFB, 0xF3… msb
NwkKey	n/a

5. Use the ‘Copy’ button to copy the **APPEUI** you have generated in HEX format (little-endian a.k.a lsb order). from TTN Console, and replace the **APPEUI** on line 57 of the ino file with the copied address. Make sure it is in correct format.
   static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

6. Use the ‘Copy’ button to copy the **DEVEUI** you have generated in HEX format (little-endian a.k.a lsb order). from TTN Console, and replace the **DEVEUI** on line 61 of the ino file with the copied address. Make sure it is in correct format.
   static const u1_t PROGMEM DEVEUI[8]={ 0x91, 0x51, 0x04, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };

7. Again, use the ‘Show/hide’ button to make **Application Key (APPKEY)** visible, and use the ‘change format button’ to convert it into HEX format (big-endian a.k.a msb order). Copy the key and update the **APPKEY** on line 67 of the script.
   static const u1_t PROGMEM APPKEY[16] = { 0xC5, 0xF2, 0x4B, 0xFB, 0xF3, 0x8A, 0x91, 0x5F, 0x79, 0x08, 0x63, 0xE9, 0x8F, 0xD4, 0xA1, 0x44 };

8. The configuring of TTN is complete. Connect the Arduino UNO to your laptop and upload the modified Lab07_otaa.ino script. Open the ‘Serial Monitor’ of Arduino software and observe the LoRa packets being transmitted after ‘TX_INTERVAL’ set in the script.



```
09:57:04.704 -> 4842821: EV_TXSTART
09:57:10.176 -> 5186375: EV_JOINED
09:57:10.210 -> netid: 19
09:57:10.210 -> devaddr: 260D7B65
09:57:10.210 -> AppSKey: F8-D0-88-BF-D1-1D-99-66-74-4E-AF-09-A0-47-A2-21
09:57:10.210 -> NwkSKey: 71-06-07-B3-C0-7E-66-8D-E1-A5-77-84-CD-C6-F1-20
09:57:10.210 -> 5187392: EV_TXSTART
09:57:16.704 -> 5593321: EV_TXCOMPLETE (includes waiting for RX windows)
09:58:16.729 -> 9343799: EV_TXSTART
09:58:16.729 -> Packet queued
09:58:23.217 -> 9749730: EV_TXCOMPLETE (includes waiting for RX windows)
09:59:23.254 -> 13500209: EV_TXSTART
09:59:23.254 -> Packet queued
09:59:28.770 -> 13844044: EV_TXCOMPLETE (includes waiting for RX windows)
09:59:29.254 -> 13874927: EV_TXSTART
```

✓ Autoscroll ✓ Show timestamp   Both NL & CR   115200 baud   Clear output

Figure 1: OTAA mode Serial Monitor

ABP vs OTAA

To learn more about the difference of ABP and OTAA, please visit: https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/