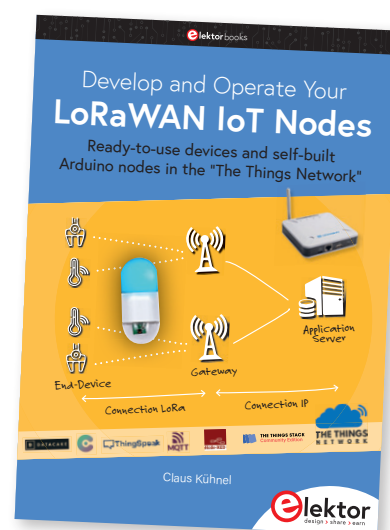


Développer et utiliser vos nœuds LoRaWAN pour l'Internet des Objets (IdO)

Exemple de chapitre :
Modules LoRaWAN Dragino LHT65, LDS01 et LDS02

Claus Kühnel (Suisse)

Considéré comme un circuit, un nœud LoRaWAN associe un microcontrôleur et un émetteur-récepteur LoRa. La disponibilité de nombreux nœuds de capteurs LoRaWAN commerciaux conçus pour un « usage industriel » devrait plutôt encourager les expérimentateurs passionnés. Dans cet article, Claus Kühnel, auteur d'un ouvrage Elektor, montre qu'il est relativement facile de faire « dialoguer » trois modules de capteurs Dragino au sein d'un réseau étendu LoRaWAN.



Note de la rédaction. Cet article est un extrait du livre de 224 pages *Develop and Operate your LoRaWAN IoT Nodes* (Elektor, 2022). L'extrait a été formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. Étant donné qu'il s'agit d'un extrait d'une publication plus importante, certains termes de cet article peuvent faire référence à des discussions se trouvant ailleurs dans le livre. L'auteur et l'éditeur ont fait de leur mieux pour éviter une telle situation et répondront volontiers aux questions. Les coordonnées des personnes à contacter sont indiquées au paragraphe « **Questions ou commentaires** ».

Des dispositifs LoRaWAN (Long-Range Wireless Area Network) extrêmement variés, proposés par de nombreux fabricants, sont disponibles dans le commerce à des prix très différents. Vous aurez une bonne vue d'ensemble de l'offre de dispositifs LoRaWAN chez les distributeurs les plus connus et sur différents sites Internet, dont certains cités ici de manière arbitraire :

- www.thethingsnetwork.org/marketplace/products/devices
- <https://iot-shops.com/product-category/lora/lorawan-devices/>
- www.lora-shop.ch/
- <https://lorawan-webshop.com/shop/10-lorawan-devices/>
- <https://smartmakers.io/iot-sensoren-uebersicht/>

Concernant les dispositifs prêts à l'emploi présentés dans l'ouvrage, j'ai veillé à ce qu'ils se situent dans une fourchette de prix acceptable pour les fabricants afin d'éliminer tout obstacle au développement expérimental de ces capteurs.

Capteur de température et d'humidité Dragino LHT65

Le capteur de température et d'humidité Dragino LHT65 est basé sur la technologie LoRaWAN et permet une transmission à longue portée. Il contient un capteur intégré de température et d'humidité SHT20 de Sensirion et une connexion pour des capteurs externes destinés à mesurer la température, l'humidité du sol, l'inclinaison, etc.

Vous pouvez, par exemple, connecter un capteur DS18B20 étanche au moyen d'un câble de connexion permettant de mesurer la température extérieure par cette liaison externe (**figure 1**).

Vous trouverez toutes les informations nécessaires sur le nœud de capteurs dans le manuel du Dragino LHT65 - référence web [1]. La connexion de capteurs externes supplémentaires y est également décrite.

Le LHT65 est équipé d'une pile intégrée non rechargeable de 2400 mAh. La durée de vie prévue de la pile est de plus de dix ans. Le nœud est entièrement compatible avec le protocole LoRaWAN v1.0.2. Le fonctionnement avec une passerelle LoRaWAN standard arbitraire est possible. Le **tableau 1** résume les caractéristiques



Figure 1. Dragino LHT65-EU868-E1 avec capteur DS18B20 externe.



Figure 2. Identifiants EUI et clés logicielles pour le LHT65.

techniques du capteur Dragino LHT65-EU868-E1. L'emballage du Dragino LHT65 comprend un autocollant indiquant les données nécessaires à l'enregistrement de l'appareil terminal auprès du serveur LoRaWAN (**figure 2**) et un câble de programmation spécial (**figure 3**). Conservez-les en lieu sûr ! Il est préférable de se souvenir des identifiants uniques étendus (EUI) et des clés pour l'enregistrement et les réglages ultérieurs.

Pour configurer le LHT65, il suffit d'utiliser les commandes AT. La mise à jour du micrologiciel et la configuration à l'aide de l'interface série nécessitent le câble de programmation connecté au LHT65 au moyen des zones de contact situées à l'arrière du capteur.

Le manuel d'utilisation du LHT65 contient la description de la mise à jour du micrologiciel à l'aide d'un débogueur/programmeur intégré ST-LINK/V2 pour STM8 et STM32 et précise de nombreux détails [2]. Je ne l'aborderai donc pas ici.

Le point essentiel, c'est la configuration au moyen de commandes AT et du câble de programmation. En raison de l'interface réalisée, il faut un convertisseur FTDI-USB-UART (**figure 4**). Un programme terminal établit une connexion. Seules les lignes Rx, Tx et GND doivent être connectées entre le câble de programmation et le convertisseur FTDI-USB-UART.

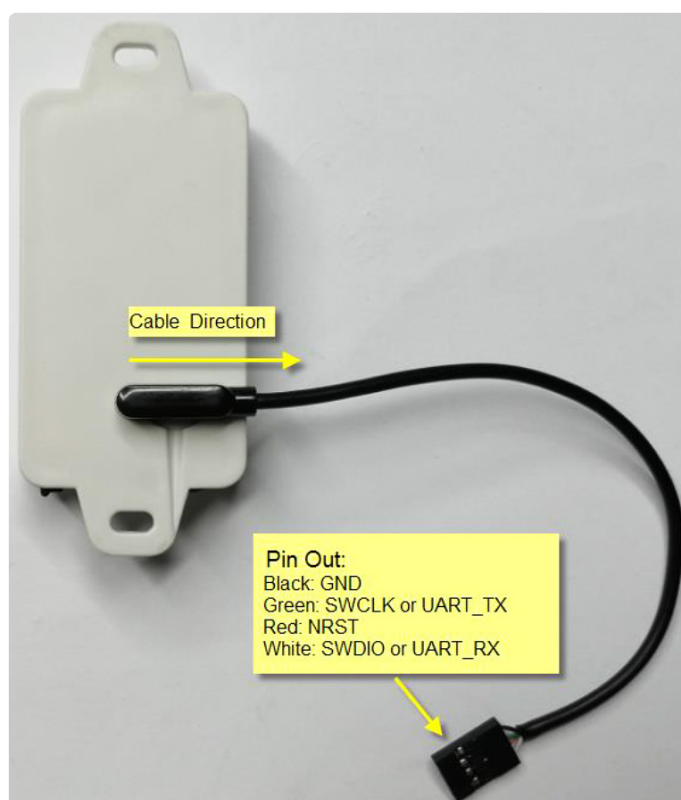


Figure 3. Câble de programmation LHT65.

Tableau 1.
Dragino LHT65-EU868-E1 - Spécifications techniques

Capteur de température (interne)	Résolution	0,01 K
	Précision	± 0,8 K
	Dérive à long terme	< 0,02 K/a
	Plage de mesure	-40 °C - +80 °C
Capteur d'humidité (interne)	Résolution	0,04 % rH
	Précision	±10 % rH
	Plage de mesure	0 - 99,9 % rH
	Temps de réponse	< 5 s
Capteur de température externe DS18B20	Résolution	0,0625 K
	Précision (-10 °C - +85 °C)	±0,5 K
	Accuracy (-55 °C - +125 °C)	±2 K
	Plage de mesure	-55 °C - +125 °C

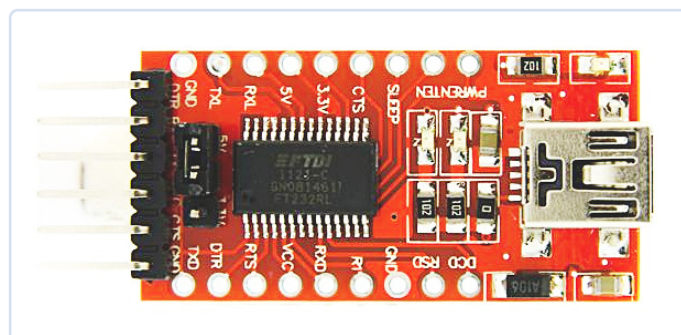


Figure 4. Convertisseur USB-UART de type FTDI.

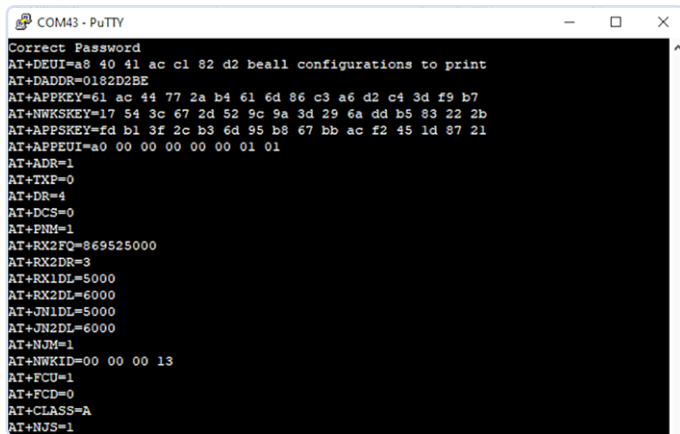


Figure 5. Requête de la configuration du LHT65.

J'utilise PuTTY comme programme terminal et je le connecte à COM43 à 9600 bauds. Vous pouvez voir le port COM utilisé dans le gestionnaire de périphériques lorsque vous connectez le convertisseur FTDI à un port USB.

PuTTY affiche la sortie du LHT65. Pour accéder au mode de configuration, entrez le mot de passe 123456. La commande ATZ suivante réinitialise le microcontrôleur interne. Chaque commande doit se terminer par ↵ (CR/LF).

123456 ↵

ATZ ↵

Vous pouvez interroger la configuration actuelle à l'aide de la commande **AT+CFG** (figure 5).

Les lignes du haut de la configuration contiennent les paramètres importants pour l'enregistrement sur le serveur TTS (CE). S'il n'y a pas de changement, ils doivent correspondre aux données figurant sur l'autocollant fourni.

L'enregistrement sur le serveur TTS (CE) se fait maintenant sous une forme légèrement différente puisque le capteur Dragino LHT65 a déjà une entrée d'enregistrement TTS (CE) et les données correspondantes sont stockées (figure 6).

La saisie des EUI et des clés suit la méthode habituelle (figure 7). Un clic sur le bouton *Register end device* (enregistrer le terminal) met fin à l'enregistrement du nouveau dispositif sur le serveur TTS (CE). S'il n'y a pas d'erreur, le nœud du capteur LHT65 renvoie ses données en direct après un court instant (figure 8).

Pour recevoir des messages lisibles sur la console TTS (CE), j'ai entré le code JavaScript du **listage 1** dans le formateur de charge utile afin de la décoder. Le formateur de charge utile décode les données de tension de la batterie, de température et d'humidité relative mesurées par le SHT20, ainsi que la température mesurée par le DS18B20 externe.

La **figure 9** montre une partie de la fenêtre de la console TTS (CE) contenant les messages décodés du LHT65. Au début, l'intervalle de transmission par défaut est de 20 minutes.

Un message descendant peut configurer le LHT65. Dans le manuel du Dragino LHT65 mentionné ci-dessus, vous trouverez une description de la configuration à l'aide des commandes AT et des messages descendants.

La commande **downlink 01 xx xx xx** configure le temps d'intervalle. Comme vous pouvez le voir sur la **figure 10**, j'ai réglé le temps d'intervalle sur 60 (0x3C = 60₁₆) secondes à l'aide de la commande

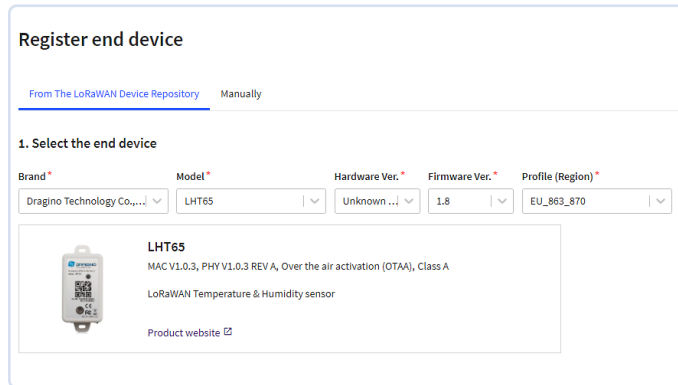


Figure 6. Enregistrement du LHT65 1ère partie.

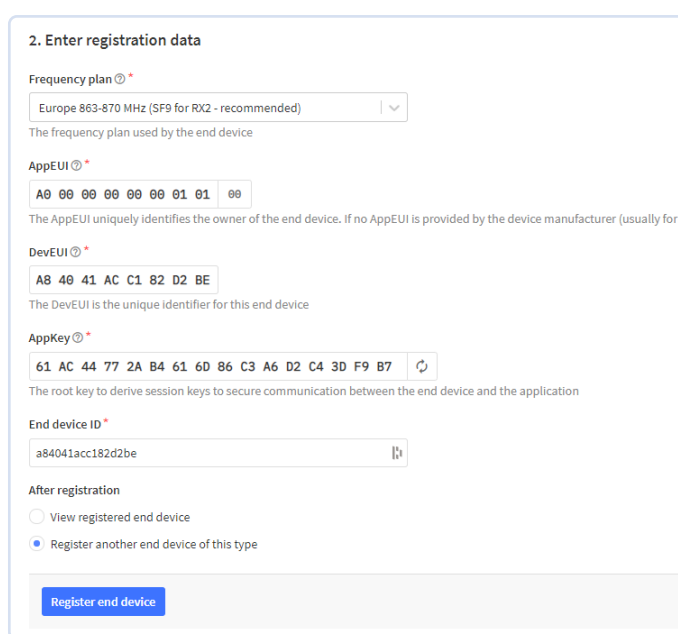


Figure 7. Enregistrement du LHT65 2e partie.

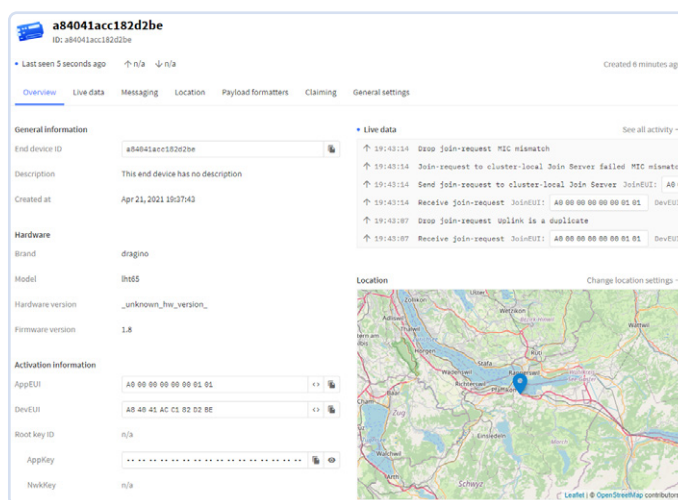


Figure 8. Données en direct produites par le LHT65.

Time	Entity ID	Type	Data preview
↑ 15:02:20	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 28.4, Temp_DS: 31.81, TempC_SHT: 31.43 } CC 18
↑ 15:52:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 27.2, Temp_DS: 31.75, TempC_SHT: 31.21 } CC 18
↓ 15:51:29	a84041acc182d2be	Forward downlink data message	FPort: 1 Payload: 01 00 02 58
↑ 15:51:15	a84041acc182d2be	Decode uplink data message failure	Unknown FPort
↑ 15:51:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 28, Temp_DS: 31.61, TempC_SHT: 31.2 } CC 18
↑ 15:50:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 26.6, Temp_DS: 31.75, TempC_SHT: 31.17 } CC 18
↑ 15:49:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 26.5, Temp_DS: 31.81, TempC_SHT: 31.13 } CC 18
↑ 15:48:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.098, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 27.6, Temp_DS: 31.56, TempC_SHT: 31.08 } CC 1A
↑ 15:47:15	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 24.9, Temp_DS: 31.56, TempC_SHT: 31.05 } CC 18
↑ 15:46:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 25.5, Temp_DS: 31.68, TempC_SHT: 31.01 } CC 18
↑ 15:45:14	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.098, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 27.3, Temp_DS: 31.68, TempC_SHT: 30.98 } CC 1A
⬇ 15:44:14		Console: Stream reconnected	The stream connection has been re-established
↑ 15:44:08	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 27.8, Temp_DS: 31.62, TempC_SHT: 30.94 } CC 18
⬇ 15:44:08		Console: Stream connection closed	The connection was closed by the stream provider
↓ 15:26:04	a84041acc182d2be	Forward downlink data message	FPort: 1 Payload: 01 00 00 3C
↑ 15:24:08	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.098, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 33.3, Temp_DS: 30.87, TempC_SHT: 30.39 } CC 1A
↑ 15:04:13	a84041acc182d2be	Forward uplink data message	Payload: { BatV: 3.096, Bat_status: 3, Ext_sensor: "Temperature Sensor", Hum_SHT: 29.8, Temp_DS: 29.87, TempC_SHT: 30.38 } CC 18

Figure 9. LHT65 — données en direct et configuration descendante.

01 00 00 3C. Les sorties suivantes sont donc également affichées dans cette grille. Avec la commande 01 00 02 58 (0x0258 = 600s) le temps d'intervalle est finalement réglé sur 10 minutes.

Comme on peut le voir sur la **figure 9**, la liaison descendante ne prend pas effet avant la liaison montante suivante. Avec un dispositif LoRaWAN de classe A, une fenêtre de réception s'ouvre dès que la liaison montante a eu lieu, et les messages ne peuvent donc être reçus qu'à ce moment-là. Dans la **figure 10**, *schedule downlink* (programmer une liaison descendante) a une signification littérale. Vous pouvez toujours décider si la liaison descendante doit

remplacer une liaison déjà planifiée ou si elle doit s'y ajouter. La configuration via une liaison descendante est très pratique.

Capteur de porte et de fenêtre Dragino LDS01

Le Dragino LDS01 est un capteur de porte et de fenêtre LoRaWAN. Le capteur détecte l'état ouvert ou fermé, et envoie cette information au serveur LoRaWAN. Grâce à ses dimensions compactes de 64 x 30 x 14 mm, le capteur peut être positionné presque partout. La **figure 11** montre le capteur fermé et la **figure 12**, le capteur ouvert. Un contact reed détecte ces deux états. De plus, certaines



Listage 1. Javascript pour les valeurs Vbatt, RH, et les capteurs de température

```
function decodeUplink(input) {
  var data = {};
  //Battery,units:V
  data.vbat = ((input.bytes[0]<< 8 | input.bytes[1]) & 0x3FFF)/1000;
  //SHT20,temperature,units:
  data.temp = ((input.bytes[2]<< 24 >> 16 | input.bytes[3])/100);
  //SHT20,Humidity,units:%
  data.humi = ((input.bytes[4]<< 8 | input.bytes[5])/10);
  //DS18B20,temperature,units:
  data.extTemp = ((input.bytes[7]<<24>>16 | input.bytes[8])/100);
  return {
    data: data
  };
}
```

LHT65
ID: a84041acc182d2be

Last seen 5 minutes ago ↑ 3,501 ↓ 349

Overview Live data **Messaging** Location Payload formatters Claiming General settings

Uplink **Downlink**

Schedule downlink

Insert Mode
☒ Replace downlink queue
☐ Push to downlink queue (append)

FPort*

Payload

The desired payload bytes of the downlink message

☐ Confirmed downlink

Figure 10. Configuration du LHT65 par liaison descendante.



Figure 11. Dragino LDS01, fermé.



Figure 12. Dragino LDS01, ouvert.

données dérivées de ces deux états sont placées dans la charge utile. Une pile de type CR2032 alimente le LDS01. Avec une bonne couverture réseau (base SF7, 14 dB), il est possible de transmettre jusqu'à 12.000 paquets de liaison montante. Une mauvaise couverture réseau (base SF10, 18,5 dB) réduit ce chiffre à environ 1.300 paquets de liaison montante. Selon les données de conception du fabricant, l'autonomie de la batterie peut atteindre un an. L'utilisateur peut facilement remplacer la pile CR2032.

L'emballage du Dragino LDS01 comprend un autocollant indiquant les données nécessaires à l'enregistrement de l'appareil terminal auprès du serveur LoRaWAN (figure 13). Encore une fois : conservez bien l'autocollant ! Il est sage d'utiliser les EUI et les clés pour l'enregistrement de l'appareil, et d'effectuer les ajustements ultérieurs plus tard.

Le LDS01 utilise une interface série accessible à l'intérieur de l'appareil au moyen d'une bande de connexion pour la configuration. Vous avez besoin d'un convertisseur FTDI-USB-UART pour connecter à nouveau RX, TX et GND (figure 14).

Pour accéder au LDS01, vous devez à nouveau utiliser un programme de terminal (par exemple, PuTTY). Le débit en bauds est de 115200 bps. Le mot de passe requis est 123456. La procédure d'enregistrement du LDS01 est la même que pour le LHT65 (figure 15). Le serveur TTS (CE) « connaît » déjà le LDS01.

Un clic sur le bouton *Register end device* (enregistrer le terminal) enregistre le nouveau dispositif sur le serveur TTS (CE). S'il n'y a pas d'erreur, le nœud du capteur LDS01 renvoie ses données en direct après un court instant (figure 16).

Pour trouver des données lisibles sur la console TTS (CE), j'ai entré



Figure 13. Identifiants EUI et clés logicielles du Dragino LDS01.

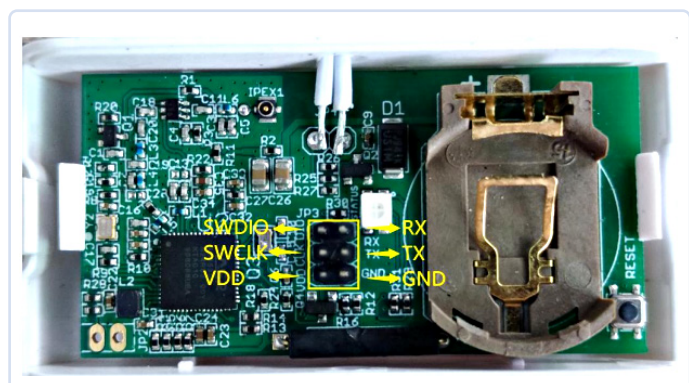


Figure 14. Dragino LDS01, et son boîtier ouvert.

Register end device

From The LoRaWAN Device Repository | Manually

1. Select the end device

Brand: Dragino Technology Co., Ltd. | Model: LDS01 | Hardware Ver.: Unknown... | Firmware Ver.: 1.3.0 | Profile (Region): EU_863_870

LDS01
MAC V1.0.3, PHY V1.0.3 REV A, Over the air activation (OTAA), Class A
LoRaWAN Door Sensor
[Product website](#)

2. Enter registration data

Frequency plan: Europe 863-870 MHz (SF9 for RX2 - recommended)

AppEUI: A8 00 00 00 00 00 01 07 00

DevEUI: A8 40 41 00 01 81 FB 8D

AppKey: 48 7C 93 6D 76 F5 D5 A5 4C 8F 1D 41 EA EA 25 49

End device ID: lds01

After registration
☒ View registered end device
☐ Register another end device of this type

[Register end device](#)

Figure 15. Enregistrement du LDS01.

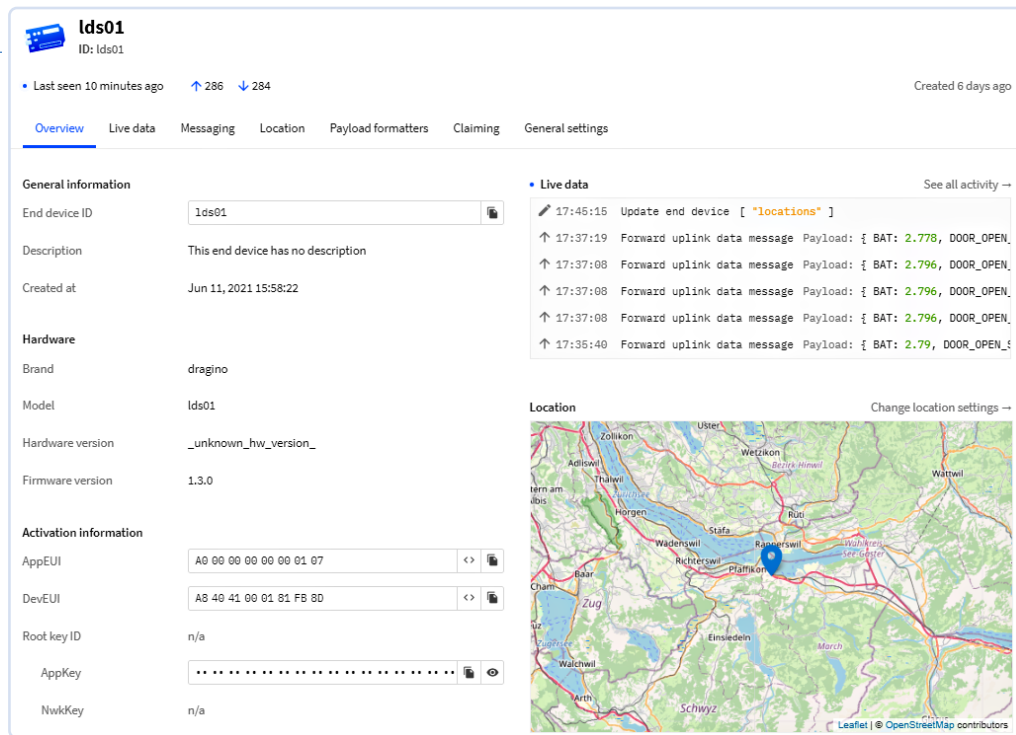


Figure 16. Données en direct produites par le LDS01.

le code JavaScript du **listage 2** dans le formateur de charge utile afin de la décoder. Le formateur de charge utile décode les données relatives à la tension de la pile, à l'état du commutateur, au nombre d'ouvertures, à la durée de la dernière ouverture et à un bit d'alarme (non évalué ici).



Listage 2. Javascript pour LDS01

```
function decodeUplink(input) {
    var state = input.bytes[0] & 0x80 ? 1:0; //
    1:open,0:close
    var voltage = ((input.bytes[0]<< 8 | input.
    bytes[1]) & 0x3FFF)/1000;
    var alarm = input.bytes[9]& 0x01;
    var open_times = input.bytes[3]<<16 | input.
    bytes[4]<<8 | input.bytes[5];
    var open_duration = input.bytes[6]<<16 |
    input.bytes[7]<<8 | input.bytes[8]; //units:min

    return {
        data: {
            BAT: voltage,
            DOOR_OPEN_STATUS: state,
            DOOR_OPEN_TIMES: open_times,
            LAST_DOOR_OPEN_DURATION: open_duration,
        },
        warnings: [],
        errors: []
    };
}
```

En plus de l'utilisation des commandes AT, un message descendant peut également configurer le LDS01. Le manuel du LDS01 décrit les commandes AT et les messages de liaison descendante pour la configuration du Dragino LDS01 - sa référence est [2]. À titre d'exemple, la commande downlink **A8 xx xx** configure le débit de données. Comme vous pouvez le voir sur la **figure 17**, j'ai réglé le débit de données sur DR5 à l'aide de la commande **A8 00 05** pour réduire le temps d'antenne, et donc, la consommation d'énergie. Si la programmation du LDS01 ne fonctionne pas par téléchargement, vous pouvez corriger le micrologiciel à l'aide des commandes AT. La commande **AT+CRX1DELAY=5** pour réduire le temps d'antenne, et donc, la consommation d'énergie.

The screenshot displays the 'lds01' device management page, specifically the 'Messaging' tab. It shows the 'Schedule downlink' section with options for 'Insert Mode' (Replace downlink queue, Push to downlink queue (append)), 'FPort' (1), and 'Payload' (A8 00 05). The 'Confirmed downlink' checkbox is unchecked. A 'Schedule downlink' button is visible at the bottom.

Figure 17. Configuration du LDS01 par liaison descendante.



Figure 18. Le Dragino LDS02.

La bibliothèque LoRaWAN, sur laquelle est basé le code des capteurs LDS01 et LWL01, comporte une erreur fatale dans une partie binaire de la bibliothèque. Un test rapide avec la version 4.4 de la bibliothèque montre que Dragino a corrigé le problème. Il n'y a pas de déclaration concernant les autres versions du logiciel. Il semble que Dragino doive publier une mise à jour du micrologiciel pour ses produits basés sur l'ASR650x afin de les rendre compatibles avec TTS (CE), voir le rapport de problème en [3].

Capteur de porte et de fenêtre Dragino LDS02

Le Dragino LDS02 est aussi un capteur de porte et de fenêtre LoRaWAN. Comme le LDS01, ce capteur détecte l'état ouvert ou fermé et envoie cette information au serveur LoRaWAN.

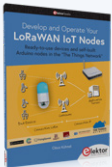
Doté de deux piles AAA, ses dimensions de 69 x 29 x 54 mm sont un peu moins compactes que celles du LDS01 (**figure 18**). Ces deux piles sont suffisantes pour environ 16.000 à 70.000 paquets de liaison montante. Lorsque les piles sont épuisées, l'utilisateur peut les remplacer par deux piles AAA du commerce.

Un jeu de clés uniques pour l'enregistrement LoRaWAN est préchargé sur chaque LDS02. L'enregistrement sur le serveur TTS (CE) est identique à celui du LDS01, et la connexion se fait automatiquement après la mise sous tension. ◀

220296-04



PRODUITS



➤ **C. Kühnel, *Develop and Operate Your LoRaWAN IoT Nodes*, Elektor 2022 (livre en anglais, SKU 20147)**
www.elektor.fr/20147

➤ **C. Kühnel, *Develop and Operate Your LoRaWAN IoT Nodes*, Elektor 2022 (E-Book, SKU 20148)**
www.elektor.fr/20148

➤ **Capteur de porte Dragino LDS02 (EU868, SKU 20004)**
www.elektor.fr/20004

➤ **Autres produits LoRa et LoRaWAN**
www.elektor.fr/catalogsearch/result/?q=LoRA



À propos de l'auteur

Le Dr Claus Kühnel a étudié les technologies de l'information à l'Université technique de Dresde, en Allemagne. Il a développé, entre autres, des systèmes embarqués pour des appareils de diagnostic de laboratoire. Dans ce domaine interdisciplinaire, il a été au contact de l'univers des *makers*. Passionné par les nouvelles technologies autour des microcontrôleurs, il est l'auteur de nombreux articles et ouvrages sur le matériel et les logiciels liés à ces véritables systèmes sur puces, en Allemagne et ailleurs.

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (info@cksript.ch), ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Dragino LHT65 Manual: <https://bit.ly/35cHPiP>

[2] Dragino LHT65 firmware update description: <https://bit.ly/3MWw5oC>

[3] Dragino LDS01 Manual: <https://bit.ly/3a34ovG>

[4] Dragino ARS650x firmware error: www.thethingsnetwork.org/forum/t/new-application-v3-otaanotworking/43338/30