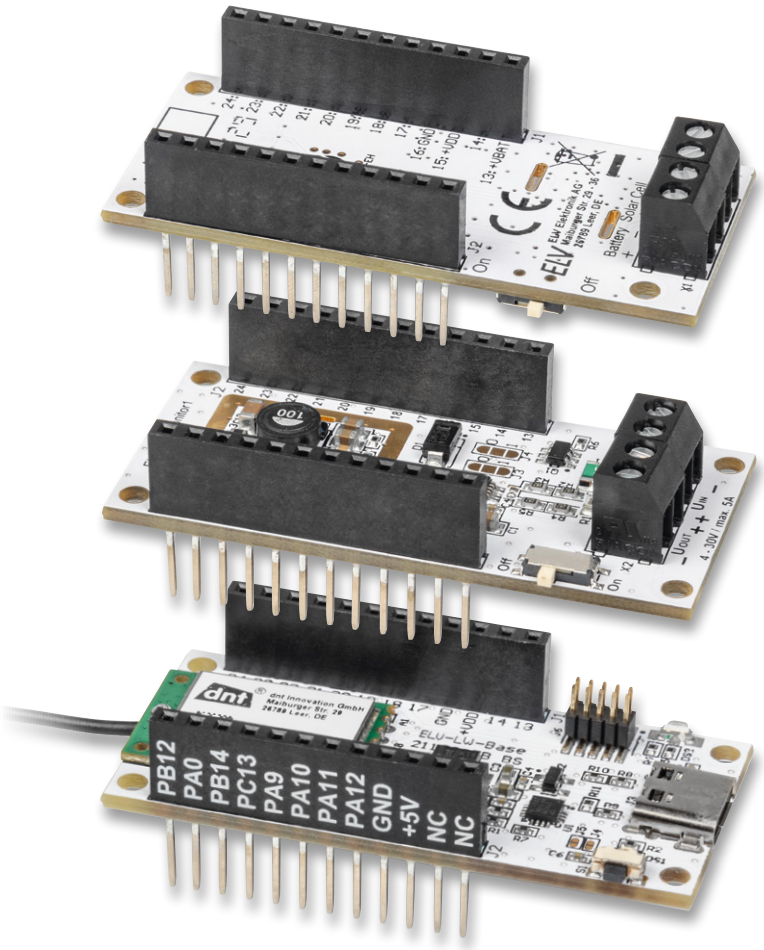
Schnell integrieren

Einfache Einbindung von LoRaWAN®-Geräten in das The Things Network

Mit einer stetig wachsenden Anzahl von Applikations- und Powermodulen deckt das ELV-Modulsystem bereits jetzt viele spannende Einsatzgebiete im LoRaWAN® ab. Für die bisherige Inbetriebnahme der ELV-LW-Base war eine manuelle Registrierung in einem LoRaWAN®-Netzwerkserver wie The Things Stack, Helium oder ChirpStack notwendig. Um diesen Prozess zu vereinfachen, hat das The Things Network mit dem „Device Repository for LoRaWAN®“ eine interessante Möglichkeit geschaffen, die Integration von Geräten zu beschleunigen. Wie die Registrierung dadurch vereinfacht wird und welche Schritte notwendig sind, um ein eigenes Gerät dem Repository hinzuzufügen, erfahren Sie in diesem Beitrag.



THE THINGS NETWORK

Logo: https://account.thethingsnetwork.org

Bisherige Registrierung im The-Things-Netzwerkserver

Bisher war die Registrierung einer ELV-LW-Base [1], eines Bausatzes wie dem ELV-LW-GPS1 [2] oder dem ELV-LW-ESI [3] ausschließlich über die manuelle Angabe aller LoRaWAN®-spezifischen Informationen möglich. Bild 1 zeigt exemplarisch die Registrierung in einer TTS-Applikation. Mit der Nutzung des Device Repositorys von „The Things Network" (TTN) [4] kann nun ein großer Teil der Gerätekonfiguration hinterlegt und per Drop-down-Menü im TTS-Netzwerkserver ausgewählt werden.

Neben dem Frequenzplan werden dort auch die LoRaWAN®-Version sowie die DevEUI, JoinEUI und der AppKey angegeben. Abgeschlossen wird der Prozess über die Vergabe einer End device ID und der anschließenden Bestätigung über den Button „Register end device" (Bild 2).

Von der ELV-LW-Base bzw. dem ELV-LW-ESI oder ELV-LW-GPS1 werden die Daten in einem Byte-Format gesendet. Um diese in einem lesbaren Format darzustellen, muss ein Payload-Decoder in der Netzwerk-Infrastruktur bei TTN hinterlegt werden. Dieser kann im im ELVshop jeweils im Downloadbereich der ELV-LW-Base, der Applikationsmodule und der Bausätze heruntergeladen und im TTN-Bereich „Payload formatters ⇨ Uplink" eingefügt werden.

Vereinfachte Registrierung mithilfe des Device Repositorys

Für die Nutzung der Geräte aus dem ELV-Modulsystem für LoRaWAN® oder entsprechender Bausätze wird nun statt der manuellen Registrierung die Option „Select the end device in the LoRaWAN® Device Repository" ausgewählt. Daraufhin erscheint ein Drop-down-Menü, aus dem der Hersteller (End device brand) und ein Gerät (Model) ausgewählt werden können. Bild 3 zeigt in diesem Zusammenhang die Auswahl der ELV-LW-Base [5].

Neben dem Bild des ausgewählten LoRaWAN®-Geräts befindet sich eine Kurzbeschreibung mit den LoRaWAN®-Spezifikationen und Geräteeigenschaften. Eingestellt werden muss in diesem Fall nur noch der Frequenzplan. Die Geräte-Keys wie DevEUI, JoinEUI und AppKey müssen weiterhin entweder manuell eingegeben werden oder werden durch Einscannen des QR-Codes automatisch eingefügt.

Dennoch vereinfacht das Device Repository die Geräteintegration erheblich, da die LoRaWAN®-Parameter bis auf den Frequenzplan bereits korrekt konfiguriert sind. Dadurch können Fehler und ungewollte Effekte während der Inbetriebnahme minimiert werden.

Ein weiterer Vorteil des Device Repositorys ist der bereits für das ausgewählte Gerät passende und installierte Payload-Decoder. Dieser muss in der am Anfang beschriebenen manuellen Installation separat hinzugefügt werden.

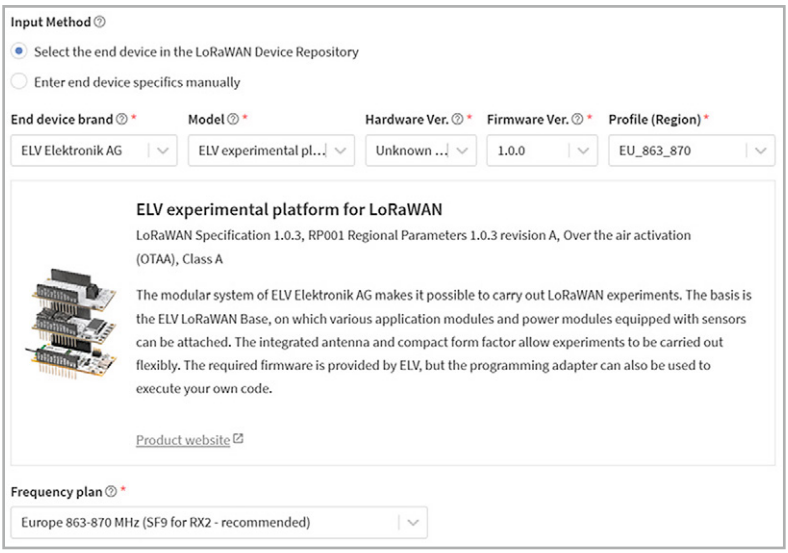


Bild 3: Registrierung der ELV-LW-Base mithilfe des Device Repositorys

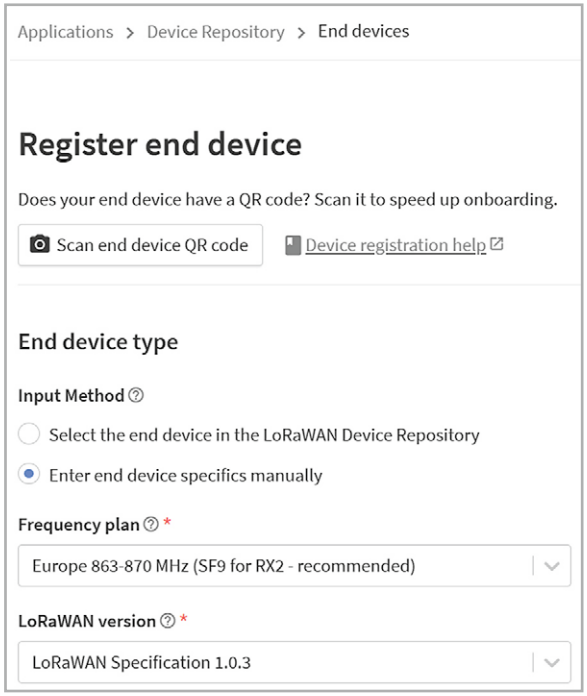


Bild 1: Manuelle Registrierung eines LoRaWAN®-Geräts im TTS

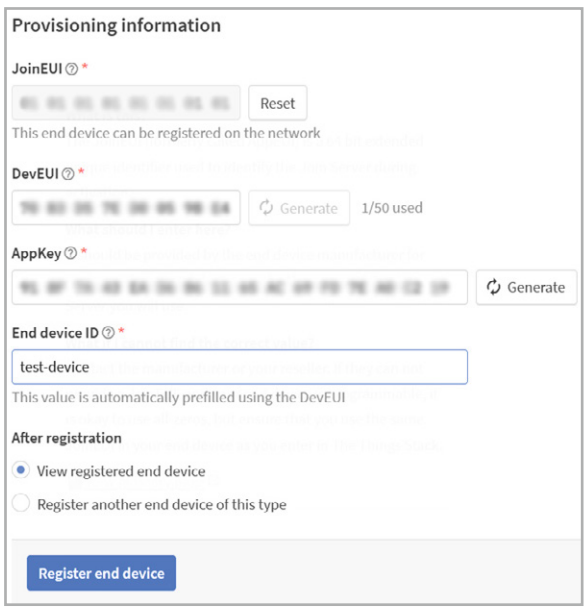


Bild 2: Eingabe der Geräte-Keys im TTS

Das Device Repository gewährleistet die sichere Zuordnung von Gerät und Decoder und eliminiert somit eine weitere potenzielle Fehlerquelle.

Neben der vereinfachten Registrierung bietet das Device Repository noch weitere Vorteile. Auf einer Website sind alle hinterlegten Geräte aufgeführt, und können eingesehen werden. Dies erhöht die Attraktivität des Produkts unter interessierten Kunden [6]. Außerdem können Produkte aus dem Device Repository im Rahmen der The Things Conference auf der „Wall of Fame" ausgestellt werden [7]. Das Hinzufügen des Geräts ist somit auch aus Unternehmenssicht lohnend.

Falls Sie neugierig geworden sind, wie neue Geräte in das Device Repository gelangen, oder sogar selbst ein Gerät integrieren möchten, geben Ihnen die folgenden Abschnitte eine detaillierte Anleitung dazu.

Das eigene Device Repository

Will man eigene oder modifizierte LoRaWAN®-Geräte in das Device Repository einbinden, so ist das für jedermann, ob als privater Nutzer oder als Unternehmen, möglich: „You can definitely add a device as a private person. Any fields like Vendor ID which you don't have, you can just remove from the yaml file" [8]. Das bietet sich vor allem für den Fall an, bei dem man eine größere Menge an gleichen Sensoren bzw. Geräten in die LoRaWAN®-Netzwerkstruktur einbinden möchte. Es entsteht zwar ein einmaliger Aufwand, zukünftige Geräte können aber deutlich schneller in das TTN integriert werden. Wir beschreiben daher diesen interessanten Weg, bei dem gleichzeitig auch die Interaktion mit GitHub an einem Beispiel gezeigt wird.

Voraussetzungen für das Editieren des Device Repositorys

Um ein eigenes Gerät dem „Device Repository for LoRaWAN®" hinzuzufügen, stellt TTN eine Anleitung in der Readme-Datei des GitHub-Repositorys bereit [9]. Die folgenden Abschnitte beschreiben die wesentlichen Schritte. Als weitere Informationsquelle kann ein Video auf dem TTN-YouTube-Kanal dienen [10].

Benötigt wird ein Linux-, MacOS oder Windows-Betriebssystem. Da Windows eines der am häufigsten verwendeten Betriebssysteme ist und die Installation einige zusätzliche Schritte beinhaltet, wird hierauf näher eingegangen. TTN empfiehlt für die einfache Ausführung von Konsolenbefehlen die Verwendung der Umgebung Windows for Linux (WSL) [11]. Dazu ist die Aktivierung von zwei Windows-Funktionen notwendig.

Das Menü kann über die Suchleiste durch den Suchbegriff „Windows Features aktivieren oder deaktivieren" gesucht werden. Innerhalb des Fensters werden dann die Punkte „Hyper-V-Plattform" und „Windows-Subsystem für Linux" aktiviert (Bild 4). Damit diese Änderungen wirksam werden, ist ein Neustart erforderlich.

Im nächsten Schritt erfolgt die Auswahl einer Linux-Distribution aus dem Windows Store. Im Rahmen dieses Artikels wird Ubuntu ausgewählt. Weiterhin sind jedoch auch u. a. die in Bild 5 dargestellten Distributionen verfügbar.

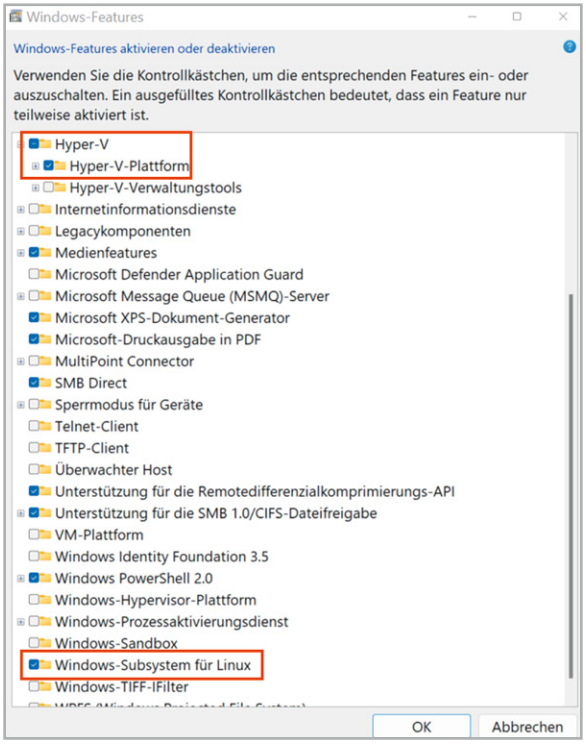


Bild 4: Aktivierung der notwendigen Windows-Funktionen

Nach dem Herunterladen der Ubuntu-App kann diese über die Suchleiste oder im Bereich „Empfohlen" des Windows-Startmenüs geöffnet werden. Innerhalb der Konsole werden ein Nutzername und ein Passwort vergeben (Bild 6).

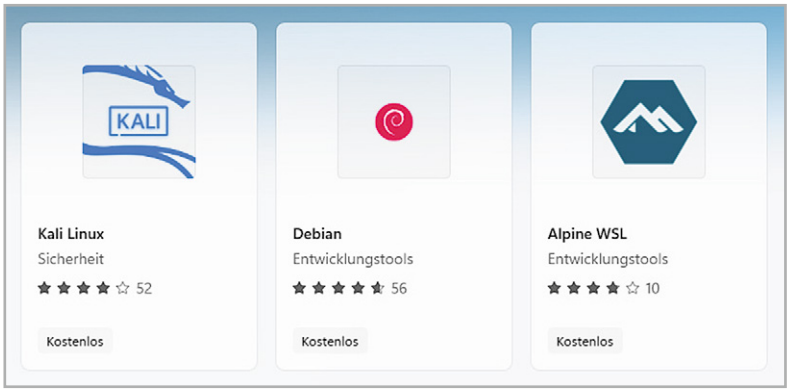


Bild 5: Weitere Linux-Distributionen für die WSL-Umgebung

Bild 6: Eingabe eines Benutzernamens und Passworts in Ubuntu

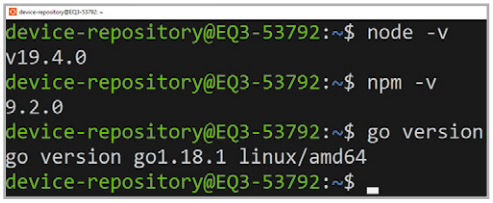


Bild 7: Erfolgreiche Installation der benötigten Pakete

Neben der Installation von WSL müssen noch die Pakete Node.js, npm und go installiert werden.
Die folgenden Befehle stellen sicher, dass die aktuellen Pakete auf dem neuesten Stand sind:

```
sudo apt-get update
sudo apt-get upgrade
```

Die Installation der aktuellen Version von go erfolgt durch den Befehl:

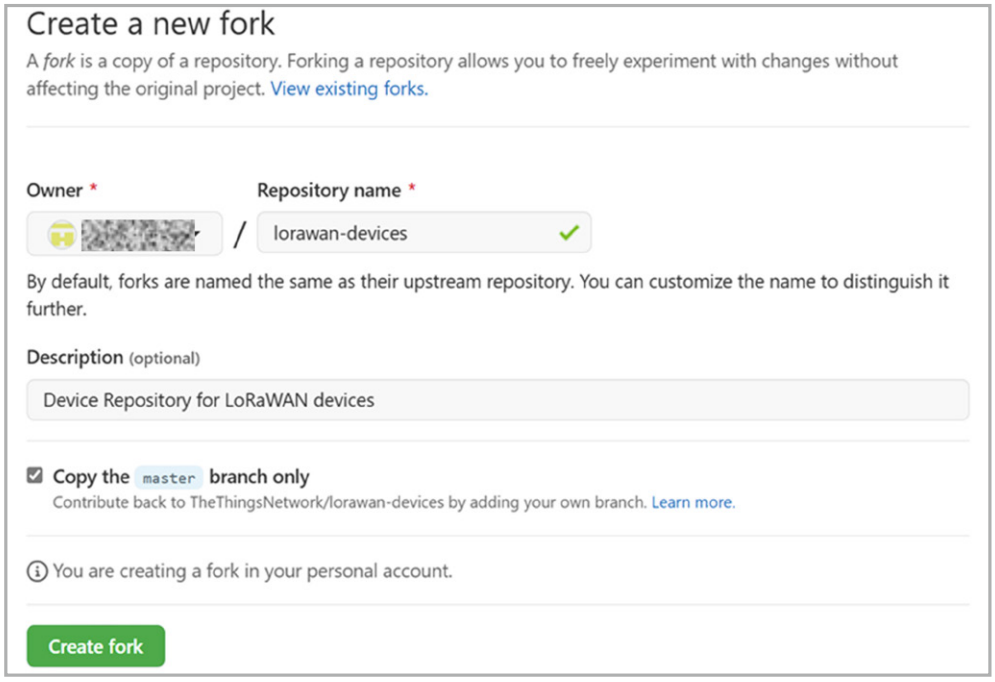
```
sudo apt-get install golang-go
```

Die Installation von Node.js und npm geschieht über die folgende Befehlssequenz:

```
sudo apt install curl
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
source ~/.profile
nvm install node
```

Zur Überprüfung der korrekten Installation können die Befehle „nmp -v", „node -v" und „go version" eingegeben werden. Die Ausgabe sollte ähnlich zu Bild 7 aussehen.

Bild 8: Erstellung eines Forks des Device Repositorys



Klonen des GitHub Repositorys

Nach der Installation aller benötigten Abhängigkeiten kann das GitHub Repository nun geklont werden. Dazu wird ein GitHub-Account benötigt. Da normale Nutzer keine Schreibberechtigung für das Repository haben, wird ein sogenannter „Fork" des Device Repositorys erstellt. Nur innerhalb dieser Kopie hat der Nutzer auch die nötigen Schreibrechte (git commit + git pull). Die Zusammenführung erfolgt am Ende in Form eines Pull Requests.

Pull Request
Wunsch an den Inhaber bzw. Administrator eines Repositorys, lokale Änderungen in das ursprüngliche Repository einzubeziehen. Die Mitteilung sollte eine Kurzbeschreibung der Änderungen beinhalten. Die Anfrage wird dann überprüft und bei Richtigkeit und Vollständigkeit mit dem ursprünglichen Repository zusammengeführt.

Ein Link zum Device Repository ist unter [6] zu finden. Zur Erstellung des Forks wird der Button „Fork" auf der Website betätigt. Daraufhin erscheint die Ansicht, wie in Bild 8 dargestellt.

In dem Formular ist keine weitere Modifikation der Felder erforderlich. Durch den Button „Create Fork" wird der Fork erstellt und der Liste von eigenen Repositorys hinzugefügt. Diese Liste ist über Profil ⇨ Your Repositorys einsehbar. Das Repository kann nun wahlweise über HTTPS oder SSH geclont werden. Die entsprechende URL erscheint nach Klicken des Buttons „Code" (Bild 9).

Innerhalb der konfigurierten WSL-/Ubuntu-Umgebung kann das Repository nun geklont werden. Dazu sollte zuvor an gewünschter Stelle ein neues Verzeichnis angelegt werden. Auf die Laufwerke z. B. C: oder D: kann aus dem Home-Verzeichnis über den Befehl

cd /mnt/c/pfad-zum-repo bzw. cd /mnt/d/pfad-zum-repo

zugegriffen werden. Der folgende Befehl zeigt den auszuführenden Befehl für das Klonen des Repositorys über HTTPS und SSH. Das Feld „user-name" entspricht dabei dem eigenen Nutzernamen. Für SSH muss zudem ein entsprechender Key auf GitHub hinterlegt sein [12]:

git clone https://github.com/user-name/lorawan-devices.git
git@github.com:user-name/lorawan-devices.git

Beschreibung der Ordnerstruktur

Die Struktur des Repositorys „lorawan-devices" beinhaltet mehrere Ordner und Dateien (Bild 10).

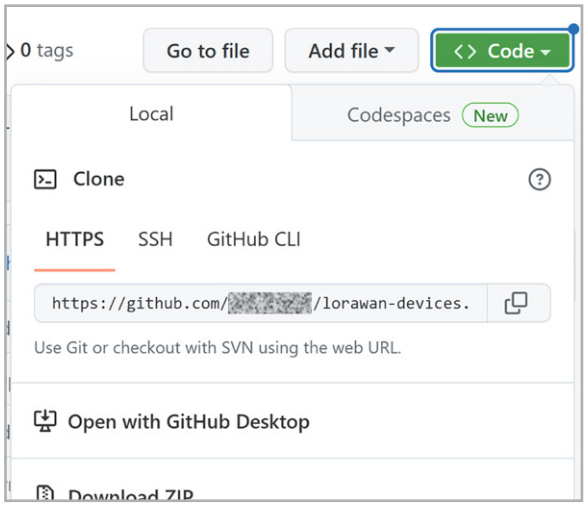


Bild 9: Auswahl des Links zum Klonen des Repositorys

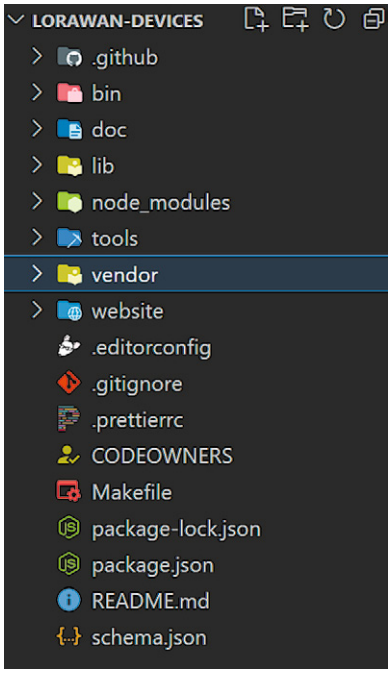


Bild 10: Ordnerstruktur des Device Repositorys

```
1929    - id: elv
1930      name: ELV Elektronik AG
1931      website: https://de.elv.com/
1932      logo: elv-logo.svg
```

Bild 11: Eintrag der ELV Elektronik AG in der Datei „index.yaml"

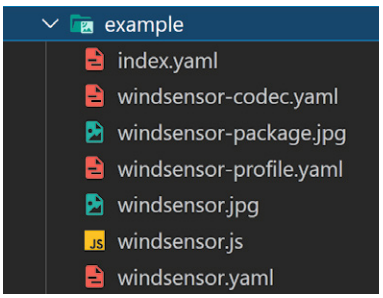


Bild 12: Ordnerstruktur des Beispiels

```
1  # This example contains just one end device: windsensor.
2
3  endDevices:
4    # Unique identifier of the end device (lowercase, alph
5    - windsensor # look in windsensor.yaml for the end dev
```

Bild 13: Auszug aus der Datei „index.yaml" aus dem Ordner „example"

```
1  name: Wind Sensor
2  description: Wind Sensor with controllable LED
3
4  # Hardware versions (optional, use when you have revisions)
5  hardwareVersions:
6    - version: '1.0'
7      numeric: 1
8    - version: '1.0-rev-A'
9      numeric: 2
10
11 # Firmware versions (at least one is mandatory)
12 firmwareVersions:
13   - # Firmware version
14     version: '1.0'
15     numeric: 1
```

Bild 14: Ausschnitt aus der Datei „windsensor.yaml"

```
1  # Vendor profile ID, can be freely issued by t
2  # This vendor profile ID is also used on the C
3  # https://lora-alliance.org/sites/default/file
4  vendorProfileID: 0
5
6  # LoRaWAN MAC version: 1.0, 1.0.1, 1.0.2, 1.0.
7  macVersion: '1.0.3'
8  # LoRaWAN Regional Parameters version. Values
9  #   1.0:   TS001-1.0
10 #   1.0.1: TS001-1.0.1
11 #   1.0.2: RP001-1.0.2 or RP001-1.0.2-RevB
12 #   1.0.3: RP001-1.0.3-RevA
13 #   1.0.4: RP002-1.0.0 or RP002-1.0.1
14 #   1.1:   RP001-1.1-RevA or RP001-1.1-RevB
15 regionalParametersVersion: 'RP001-1.0.3-RevA'
```

Bild 15: Ausschnitt aus der Datei „windsensor-profile.yaml"

```
10 function decodeUplink(input) {
11   switch (input.fPort) {
12     case 1:
13       return {
14         // Decoded data
15         data: {
16           direction: directions[input.bytes[0]],
17           speed: input.bytes[1],
18         },
19       };
20     default:
21       return {
22         errors: ['unknown FPort'],
23       };
24   }
25 }
```

Bild 16: Ausschnitt aus der Datei „windsensor.js" (Payload-Decoder)

Für das Hinzufügen eines eigenen Geräts werden jedoch nur Dateien im Ordner „vendor" hinzugefügt und bearbeitet. Dies geschieht in einem beliebigen Editor, empfohlen wird die Verwendung von Visual Studio Code. Auch die folgenden Bilder stammen aus VS Code.

Im Ordner „vendor" befindet sich die Datei „index.yaml", in der pro Hersteller ein Eintrag vorhanden ist. Jeder dieser Einträge beinhaltet mindestens die Felder „id" und „name", optional ist jedoch auch die Angabe weiterer Informationen in den folgenden Feldern möglich:

- vendorID: optionaler Parameter zum Laden eines definierten LoRaWAN®-Profils
- ouis: organizationally unique identifier = einzigartiger Organisationsbezeichner bestehend aus 24 Bit, IEEE-genormt
- website: Link zum Hersteller des Produkts
- logo: Logo des Unternehmens (.png oder .svg Datei)
- description: Kurzbeschreibung des Herstellers
- social: Angabe von Links zu Facebook-, LinkedIn- oder Twitter-Profilen

Für das Unternehmen ELV Elektronik AG sieht der Eintrag derzeit wie in Bild 11 aus.

Für die neu erstellte ID muss nun, ebenfalls im Verzeichnis „vendor", ein gleichnamiger Ordner erstellt werden. Dieser beinhaltet gekapselt alle Informationen zu den Geräten eines Herstellers. Die weiterführende Ordnerstruktur wird anhand des von TTN bereitgestellten Beispielordners „example" beschrieben.

Bild 12 zeigt die Dateien des Ordners „example". Bild 13 zeigt einen Ausschnitt aus der Datei „index.yaml". Dort wird unter dem Key „endDevices" jedes Gerät mit einer eindeutigen ID referenziert. Dieser Bezeichner dient gleichzeitig als Präfix für alle zugehörigen Dateien des Geräts.

Die eigentliche Beschreibung eines Geräts erfolgt in der Datei „windsenor.yaml". Dort werden die Spezifikationen des Geräts, wie z. B. der Name, eine Kurzbeschreibung, die Hard- und Firmwareversion, die verbauten Sensoren, weitere Funkstandards sowie die Abmessungen angegeben. Bild 14 zeigt einen Ausschnitt aus der Datei.

Innerhalb der Datei „windsensor-profile.yaml" erfolgt die Angabe der LoRaWAN®-spezifischen Parameter, wie die MAC-Version und die unterstützten Geräteklassen (A, B oder C) (Bild 15). Durch diese Angaben wird eine potenzielle Fehlerquelle während des manuellen Registrierens eines Geräts auf TTS ausgeschaltet, da sich Applikationen mit einer falschen MAC-Version nicht wie gewünscht verhalten können.

Ein weiterer essenzieller Bestandteil des Device-Repositorys sind die hinterlegten Payload-Decoder. Im Beispiel befindet sich dieser in der Datei „windsensor.js" (Bild 16). Dort werden Funktionen zur Dekodierung des Up- und Downlinks definiert. Folgende Signaturen der Funktionen sind möglich:

- decodeUplink(input) ⇨ dekodierter/lesbarer Payload
- normalizeUplink(input) ⇨ SI-normalisierter Payload
- decodeDownlink(input) ⇨ dekodierter/lesbarer Payload

In der Datei „windsensor-codec.yaml" befinden sich Beispiele für die Dekodierung von Up- und Downlinks. Als Eingabe dient ein Byte-Array (Bild 17), dem dann die entsprechenden Ausgaben des Decoders zugeordnet werden . Diese Beispiele sind später auch auf der Geräteübersicht des Device Repositorys sichtbar (Bild 18).

Bei den beiden Dateien „windsensor-package.jpg" und „windsensor.jpg" handelt es sich um Produktbilder des Windsensors. Allgemein können beliebig viele Bilder im JPG- oder PNG-Format hinzugefügt werden, die maximale Auflösung ist jedoch auf 2000 x 2000 Pixel beschränkt.

Die Bilder sind sowohl auf der Website als auch bei der Registrierung in TTS zu sehen. Wichtig dabei ist, dass unter dem Tag „photos ⇨ main" in der Datei „windsensor.yaml" möglichst eine Gesamtansicht des Geräts zu sehen sein sollte. Unter dem Tag „other" können weitere Perspektiven oder Detailaufnahmen hinzugefügt werden.

```yaml
# Uplink decoder decodes binary data uplink
# For documentation on writing encoders and
uplinkDecoder:
  fileName: windsensor.js
  # Examples (optional)
  examples:
    - description: 32 knots from the North
      input:
        fPort: 1
        bytes: [0, 32]
      output:
        data:
          direction: 'N'
          speed: 32
```

Bild 17: Ausschnitt aus der Datei „windsensor-codec.yaml"

Bild 18: Ausschnitt aus dem Data-Previewbereich des ELV-LW-Base-Eintrags auf der Device-Repository-Website

Zusammenfassend müssen also für jedes neue Gerät die folgenden Dateien erstellt werden:

<Gerätename>.yaml: Spezifikationen des Geräts + Verweise auf Payload-Decoder und Bilder

<Gerätename>.js: Implementierung des Payload-Decoders (mindestens Uplink, Normalisierung und Downlink optional)

<Gerätename>-codec.yaml: Beispiele des Payload-Decoders

<Gerätename>-profile.yaml: LoRaWAN®-Parameter (kann auch für mehrere Geräte genutzt werden)

index.yaml: zentrale Übersicht aller Geräte

Validierung der Änderungen und Erstellung des Pull Requests

Nach dem Hinzufügen eines neuen Geräts müssen die Änderungen überprüft werden. Innerhalb der Ubuntu-Umgebung wird dazu der folgende Befehl eingegeben, um das Repository vorzubereiten:

make deps

Falls das Paket „make" nicht vorhanden ist, kann es über den Befehl

sudo apt install make

nachinstalliert werden. Die Validierung wird dann mit dem Befehl

make validate

gestartet. Nach dem Durchlauf des Skripts dürfen keine Fehlermeldungen erscheinen. Zur automatischen Formatierung sollte ebenfalls der Befehl

make validte fmt

ausgeführt werden. Auch hier sollten keine weiteren Fehler auftreten.

Die Ausgabe des erfolgreichen Durchlaufs ist in Bild 19 zu sehen.

Im nächsten Schritt können die Änderungen in der eigenen Kopie des Repositorys über die Befehle „git add", „git commit" und „git push" hochgeladen werden. Wichtig ist, der Commit-Nachricht die E-Mail-Adresse des GitHub-Accounts hinzuzufügen, da es sonst in einem der folgenden Schritte (Unterzeichnung der CLA) zu Problemen kommen kann.

Die Befehlsfolge könnte wie folgt aussehen:

git config user.email <E-Mail-Adresse des GitHub Accounts>
git add .
git commit -m „Beschreibung der Änderungen"
git push

Damit diese Änderungen auch im globalen Repository sichtbar werden, wird ein Pull Request gestellt. Diese Ansicht befindet sich im gleichnamigen Tab des Forks (Bild 20).

Über den Button „New pull request" gelangt man in eine weitere Ansicht, die in Bild 21 dargestellt ist. Dort kann optional der Branch angepasst werden, dies ist jedoch hier nicht erforderlich.

```
vendor/yosensi/yo-refrigerant-monitor.yaml 5ms
vendor/yosensi/yo-temp-profile-au.yaml 3ms
vendor/yosensi/yo-temp-profile-us.yaml 4ms
vendor/yosensi/yo-temp-profile.yaml 4ms
vendor/yosensi/yo-temp.yaml 6ms
bin/uplink-fields.js 36ms
bin/validate.js 49ms
device-repository@EQ3-53792:/mnt/d/Journalbeitrag-Device-Repo/lorawan-devices$
```

Bild 19: Ausgabe bei erfolgreicher Ausführung des Befehls „make validate fmt"

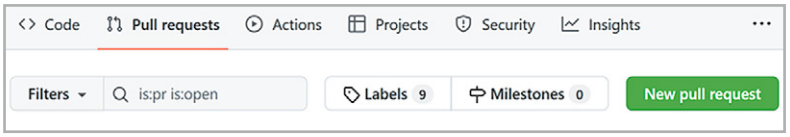


Bild 20: Erstellung eines neuen Pull Requests

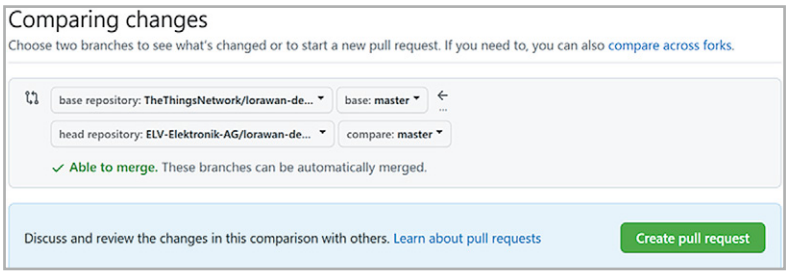


Bild 21: Auswahl des Branches für den Pull Request

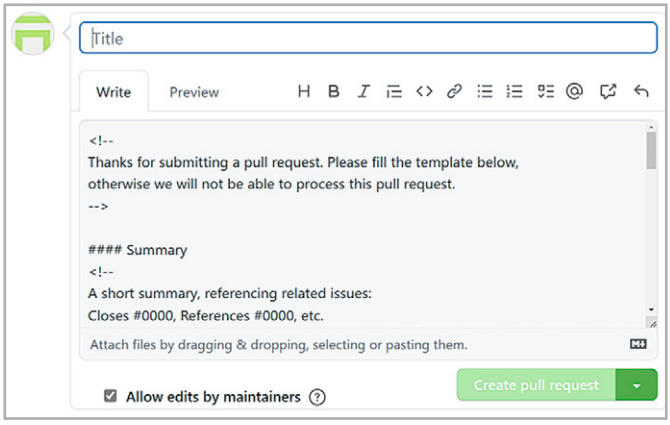


Bild 22: Angabe des Titels und der Änderungen des Pull Requests

Bild 23: Auszug aus dem Contributor License Agreement des Device Repositorys

Über den Button „Create pull request" kann dann das finale Fenster aus Bild 22 erreicht werden. Nach Eingabe eines aussagekräftigen Titels, der die unternommenen Änderungen zusammenfasst, sowie der Beschreibung der Änderungen kann der Pull Request durch Betätigen des Buttons „Create pull request" abgeschickt werden.

Der Status des Pull Requests ist im globalen Device Repository im Tab „Pull Requests" einsehbar. Abschließend muss dort auch innerhalb des eigenen Pull Requests das sogenannte Contributor License Agreement (CLA) unterzeichnet werden (Bild 23).

Nach Abschluss dieses Schritts beschäftigt sich ein Reviewer mit den Änderungen des Pull Requests und kommentiert diese, falls formale oder inhaltliche Probleme auftreten. Ansonsten werden die Änderungen zusammengeführt und der Pull Request wird geschlossen (Bild 24).

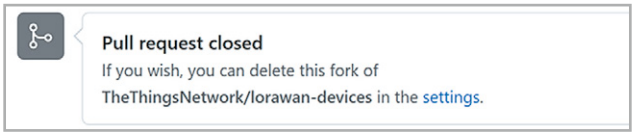


Bild 24: Erfolgreich abgeschlossener Pull Request

Weitere Infos

[1] ELV-LW-Base Experimentierplattform für LoRaWAN®, ELV-BM-TRX1: Artikel-Nr. 158052
[2] ELV LoRaWAN® GPS Tracker ELV-LW-GPS1: Artikel-Nr. 157519
[3] ELV Bausatz LoRaWAN® Energiezähler-Sensorschnittstelle ELV-LW-ESI: Artikel-Nr. 157439
[4] The Things Network: https://www.thethingsnetwork.com
[5] Website des ELV-LW-Base im Device Repository:
https://www.thethingsnetwork.org/device-repository/devices/elv/elv-bm-trx1/
[6] TTN Device Repository for LoRaWAN®: https://www.thethingsnetwork.org/device-repository/
[7] The Things Conference: https://www.thethingsnetwork.org/conference/
[8] The Things Network – Support
[9] TTN Device Repository for LoRaWAN® auf GitHub: https://github.com/TheThingsNetwork/lorawan-devices
[10] Anleitung Device Repository (TTN YouTube-Kanal): https://www.youtube.com/watch?v=pnwtEgw4f-c
[11] WSL-Distributionen im Microsoft Store: https://aka.ms/wslstore
[12] Hinzufügen eines SSH-Keys auf GitHub:
https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

Alle Links finden Sie auch online unter: de.elv.com/elvjournal-links