Übersicht

Dieses Dokument beschreibt, wie von extern auf die Daten der LTX Microcloud zugegriffen werden können. Für den einfachsten Fall, reine Abfrage, gibt es diese kleine und sehr einfache API, die z.B. auch der Grafik-Viewer G-Draw der LTX Microcloud verwendet.

Das System kurz erklärt

LTX Logger – Mobilfunk oder WLAN - Bidirektional

Primär „spiegeln“ die LTX Logger den relevanten Teil ihres Dateisystems bei jeder Übertragung auf einen Server (ein einfacher LA(M)P oder WA(M)P Low-Cost-Server reicht aus, PHP Versionen 7 – 8 sind getestet, eine SQL-Datenbank ist optional). Die Komminikation Logger – Server erfolgt bidirektional und mit einem propritären Protokoll per HTTP(S) und wurde extrem auf Low-Power und Redunanz optimiert und benötigt weitaus weniger Energie als die oft im Zusammenhang mit IoT genannten Protokolle, wie etwa MQTT oder CoAP. Das Übertragungssystem hat sich langjährig bewährt und arbeitet absolut stabil.

Eine Übertragung wird immer vom Logger initialisiert.

Wichtig: Dieses Protokoll ist redundant und Fehlertolerant!

Open-SDI12-Blue-Sensoren und deren Gateways - Unidirektional

Diese Geräte verfügen i.d.R. über keinen Speicher für die Daten und senden Messwerte „ad hoc“. Der Vorteil hier sind die meist sehr kleinen Datenpakete, die sich sehr gut über Kanäle wie LoRa, NB-IoT oder IoT-Satelliten versenden lassen. Diese Pakete werden dann einzeln (unidirektional) auf die Reise geschickt.

Allerdings gilt es zu beachten, dass bei Paket-basierten Datenversand die Pakete nicht immer in der richtigen Reihenfolgen eintreffen: Speziell bei IoT-Satelliten können hier Laufzeiten im Stunden-Bereich auftreten. Sinngemäß bedeutet das einfach, dass z.B. die Pakte mit den Nummern 3,4,5,6 dann in der Reihenfolge von z.B. 4,6,5,3 eintreffen können.

Dem Server ist dies bekannt (der Index wird als HK-Wert , Kanal „100: PackInd“) jedem Datensatz automatisch angefügt) und diese Daten werden vor Ausgabe automatisch auf die richtige Reihenfolge sortiert. Allerdings gilt dies nur für den „LTX_Server“ (siehe folgend), für „LTX-Legacy“ müssen die Daten gegebenenfalls selbst vor weiterer Verarbeitung sortiert werden.

Wichtig: Bei diesem System können verlorengegangene Pakete nicht wiederholt werden!

(Anmerkung: LTX-Logger benötigen keine Packet-Nummern, hier ist die Reihenfolge immer korrekt, da der Logger direkt und bidirektional mit dem Server kommuniziert).

Auf dem Server

Die Software für den Server liegt in zwei Versionen vor „LTX_Legacy“ und „LTX_Server“.

Für „LTX_Legacy“ reicht ein einfacher Webserver (z.B. „Apache“ oder „Nginx“ sind perfekt) zusammen mit PHP (Version 7 – 8).

Für „LTX_Server“ wird zusätzlich eine einfache Standard SQL-Datenbank (z.B. MySQL oder MariaDB) benötigt.

Im einfachsten Fall („LTX_Legacy“) liegen die Daten dann auf dem Server in Dateien und der Anwender kann mit eigenen Scripten auf Übertragungen reagieren.

Die etwas komfortablere Variante („LTX_Server“) verwendet zusätzlich eine kleine SQL-Datenbank zur Zwischenspeicherung der eingehenden Daten. Diese SQL-Datenbank speichert nur eine begrenzbare Historie der einghenden Daten, z.B. die letzten 30 Tage oder letzten 10000 Messungen.

Zur Performance: Bei „LTX_Legacy“ ist die Menge der möglichen Geräte nur durch den Speicherplatz des Servers begrenzt. Selbst auf sehr kleinen Servern sind damit problemlos (theoretisch) Zig-Tausende von Geräten möglich! Bei „LTX_Server“ dürfte die Maximalgröße der Datenbank das Maximum festlegen. Da die Historie aber einstelbar ist, sind damit ebenfalls Tausende von Geräten möglich.

Beide Installationsmöglichkeiten sind im Quellcode (PHP, HTML) auf einem Git vorhanden und können vom Anwender gerne auch kostenfrei selbst auf eigenen Servern installiert werden. Gerne geben wir Zugriff auf das ensprechende Git.

LTX_Legacy

Die notwendigen Docs dazu befinden sich im Git.

LTX_Server

Die notwendigen Docs dazu befinden sich im Git. Dies hier ist eine kleine Zusammenfassung zum externen Zugriff auf die Daten.

// V1.7x: Als Beispiel wurden die Daten eines Loggers mit 3 TensioMark upgedate zum „schnellen Test“ //

API: Get-Zugriff

Hier befindet sich das Basis-Script:

https://joembedded.de/ltx/sw/w_php/w_gdraw_db.php

Unbedingt benötigte Parameter:

- s: MAC hier `MAC=5C9DA0D493EA84B3` (immer 16 Digits)

- k: Token #x (vergebbar durch Owner) oder Owner-Token (alle immer 16 Digits) hier:
  Owner-Token=F818CC75A53C34F8 (aus Werks-Setup)
  Token #0 =2E20BCE8BB980C8D (vergebbar pro Gerät unter "Server" -> "Token")
  // Es wird empfohlen das (jederzeit änderbare) vergebbare Token #x zu verwenden

Optionale Empfohlene Parameter:

- lim: Anzahl der Zeilen (Default ist 0), maximal werden alle verfügbaren Daten ausgegeben
- m: Letztes abgerufenes Datum (UTC-Sekunden)

Optionaler GPS Parameter (bitte nur bei Bedarf und nach Rücksprache verwenden):

- mk: Wenn gesetzt wird der hinterlegte Map-Key mit augegeben und die Daten der letzten Funkzelle ermittelt:

  ```
  #MK: pk.eyJ1Ijoiam9lxxxxxxxn4s_MTgg6ctAQZvXg
  #LAT: 48.946489
  #LNG: 8.407911
  #RAD: 1074
  ```

  (Also hier im Umkreis von 1074m um GPS:48.946489, 8.407911 (Ettlingen))

Beispiel:

Letztes abgerufenes Datum war 1617012000:

```
https://joembedded.de/ltx/sw/w_php/w_gdraw_db.php?
s=5C9DA0D493EA84B3&k=2E20BCE8BB980C8D&lim=1000&m=1617012000

#MDATE: 1617012070
#NOW: 1617017360
<MAC: 5C9DA0D493EA84B3>
<NAME: LTX93EA84B3>
!U 0:°C 1:pF 2:°C 3:pF 4:°C 5:pF 90:V(Bat) 91:°C(int) 92:%rH(int) 93:mAh(Bat)
199 !1616990400 0:8.4 1:5.780 2:8.6 3:5.919 4:8.5 5:5.997
200 <NT AUTO OK(19)>
201 !1616994000 0:14.8 1:6.419 2:13.1 3:6.277 4:13.1 5:6.323 90:7.342 91:11.7 92:37.4 93:34.55
202 !1616997600 0:15.9 1:6.514 2:15.1 3:6.448 4:14.9 5:6.454
203 <NT AUTO OK(11)>
204 !1617001200 0:15.9 1:6.506 2:15.9 3:6.518 4:15.6 5:6.495 90:7.339 91:15.0 92:38.3 93:34.92
205 !1617004800 0:15.0 1:6.432 2:15.4 3:6.484 4:15.2 5:6.492
206 <NT AUTO ERROR -2303(17)>
207 !1617008400 0:16.0 1:6.512 2:15.5 3:6.479 4:15.8 5:6.515 90:7.344 91:15.6 92:38.3 93:35.60
208 !1617012000 0:18.0 1:6.710 2:16.8 3:6.585 4:17.7 5:6.668
<10 Lines (5.816 msec)>
```

Es werden die Zeilen 199-208 aus der DB ausgegeben. Das Format nennt sich „*.EDT (Easy Data)“ und ist sehr einfach:
- Die Messkanäle sind „0:“ bis „89:“
- Die HK-Kanäle (seltener erfassten „Hausmeisterdaten“) für LTX-Logger von „90:“ bis „99:“ und für Open-SDI12-Blue basierte Geräte von „90:“ bis „127:“
- Auf Zeilen mit „!(UTC)“). Headers jeweils nach Zeile „!U…“ (mind. 1 mal am Anfang vorh.)
- Kommentare in Zeilen „<...“.

Ein kleines Konvertierungs-Script in PHP ist im Git enthalten („..ltx/legacy/edt_view.php“ , bzw. auch in JS in „..ltx/sw/gdraw.html“) und liefert dazu:

```
NO, TIME, °C(0), pF(1), °C(2), pF(3), °C(4), pF(5), V(Bat)(90), °C(int)(91), %rH(int)(92), mAh(Bat)
(93)

199, 29.03.21 04:00:00, 8.4, 5.780, 8.6, 5.919, 8.5, 5.997
<NT AUTO OK(19)>
201, 29.03.21 05:00:00, 14.8, 6.419, 13.1, 6.277, 13.1, 6.323, 7.342, 11.7, 37.4, 34.55
202, 29.03.21 06:00:00, 15.9, 6.514, 15.1, 6.448, 14.9, 6.454
<NT AUTO OK(11)>
204, 29.03.21 07:00:00, 15.9, 6.506, 15.9, 6.518, 15.6, 6.495, 7.339, 15.0, 38.3, 34.92
205, 29.03.21 08:00:00, 15.0, 6.432, 15.4, 6.484, 15.2, 6.492
<NT AUTO ERROR -2303(17)>
207, 29.03.21 09:00:00, 16.0, 6.512, 15.5, 6.479, 15.8, 6.515, 7.344, 15.6, 38.3, 35.60
208, 29.03.21 10:00:00, 18.0, 6.710, 16.8, 6.585, 17.7, 6.668
<RUNTIME: 11.6801 msec>
```

Es ist wichtig, sich hier den Eintrag

```
#MDATE: 1617012070
```

Zu merken, beim folgenden Aufruf (== keine neuen Daten vorhanden) wäre die Ausgabe lediglich:

```
https://joembedded.de/ltx/sw/w_php/w_gdraw_db.php?
s=5C9DA0D493EA84B3&k=2E20BCE8BB980C8D&lim=10&m=1617012070

#MDATE: 1617012070
#NOW: 1617017419
```

Kontakt

Für technische Rückfragen: JoEmbedded@gmail.com

***