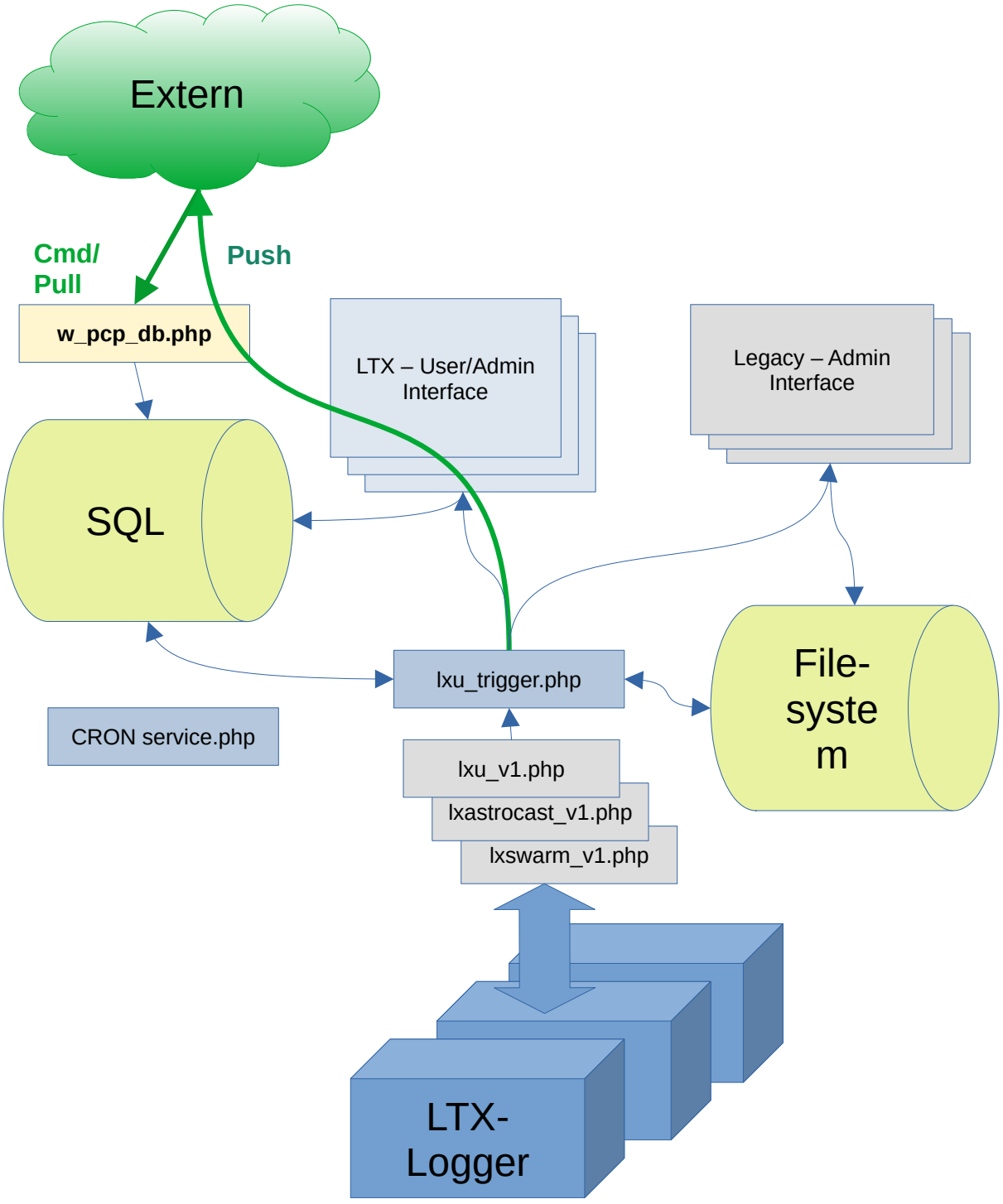
LTX Microcloud Push/Pull



Die Struktur der LTX Microcloud ist recht simpel. Sie wurde so entworfen, dass sie auf nahezu jedem beliebigen Standard-LAMP/WAMP Server läuft. Quasi als „Eingangsfilter“ für ein nachgeschaltetes System.

- Die Logger/Geräte kommunizieren jeweils nur mit dem Treiber `lxu_vX.php` (Logger) oder `lxXXXX_vX` (andere Geräte, z.B. langsame, Datagramm-basierte Satelliten- oder LoRa-Systeme). Die Kommunikation erfolgt möglichst SCHNELL, damit die Verbindung möglichst rasch wieder abgebaut werden kann.
- Jedes Gerät hat ein eigens Verzeichnis mit seiner MAC („`../MAC/..`“) im lokalen Filesystem, was sinngemäß den gespiegelten Speicherinhalt des Gerätes und Zusatzdaten (Logs, ..) enthält.
- Die Treiber triggern das Script `lxu_trigger.php` (was länger laufen darf) und die Daten zuerst ins lokale Filesystem (enthält sinngemäß den gespiegelten Speicher der Logger) und dann in die DB transportieren.
- „Legacy“ ist ein einfaches, älteres Interface auf das Filesystem und hauptsächlich für Admin-Funktionen entwickelt.
- „LTX Microcloud“ ist das neuere, etwas komfortablere Interface für die regulären Aufgaben.
- Per CRON räumt `service.php` regelmäßig auf. Die LTX Microcloud ist als ein sich selbst pflegendes System für die Kurzzeit-Historie und (Fehler-) Analysen gedacht.
- Bei jedem Transfer kann optional eine PUSH-Benachrichtigung ausgelöst werden und per PULL können dann Daten (neue oder alle und auch Einstellungen) abgegriffen/verändert werden.

Push

Sofern aktiviert, wird bei jedem Transfer ein externes Script (vorzugsweise `HTTPS://`) aufgerufen.

Parameter:

s	Die 16-stellige MAC des Gerätes
k	Optional ein Access-Token (pro MAC individuell, daher: Merken!)

Hinweis: Quelle dazu in `lxu_trigger.php`, ca. bei „// 4.th Part PUSH (opt.) Reduce CURL TIMEOUT to 10 sec“

Pull

Mit den beiden URL-Parametern s und (optional) k kann der externe Service nun `w_pcp.php` aufrufen. Quelle: `sw/w_php/w_pcp.php`

Parameter:

s	Die 16-stellige MAC des Gerätes
k	Das Access-Token von oben (Hinweis Aus Datei '`../$mac/quota_days.dat`')

Kommandos mit Beispielaufrufen

Übersicht über die Kommandos

Parameter können per URL / POST übergeben werden:

- Basis-Aufruf / Ausgabe Version: `cmd` leer
- Listet alle Devives zu diesem Key auf: `cmd=list`
- Device Details (‚device'-Eintrag aus DB): `cmd=details`
- Daten Zeilen aus DB ausgeben (optional mit Limits ‚minid'/'maxid'): `cmd=getdata`
- Parameter 'iparam.lxp' zu diesem Device mit Beschreibung ausgeben: `cmd=iparam`
  Vor Ausgabe erfolgt eine Fehlerprüfung
- Parameter 'iparam.lxp' zu diesem Device aendern: `cmd=iparamchange&iparam[...]=...`
  Es können alle (änderbaren Zeilen) geändert werden, vorher und nachher erfolgt eine Fehlerprüfung
- Pending Parameter zu diesem Device entfernen: `cmd=iparamunpend`

Messwerte:

Die Zeilen mit Messwerten bestehen immer aus der Kombination ‚`KanalIndex:Wert`'. Trenner ist ein einzelnes `Leerzeichen`, Wert ist i.d.R: ein Zahlenwert (Fließkommazahl). Optional kann Wert auch ein Alarm oder eine Fehlermeldung sein, siehe „Special" weiter unten.

https://aquatos.net/ltx/sw/w_php/w_pcp.php?
s=F0000000000021F4&k=ABC&cmd=getdata&minid=10&maxid=20

liefert:

```
// 20230131184456
// https://aquatos.net/ltx/sw/w_php/w_pcp.php?
s=F0000000000021F4&k=ABC&cmd=getdata&minid=10&maxid=20

{
  "overview": {
    "mac": "F0000000000021F",
    "db_now": "2023-01-31 17:45:01",
    "min_id": 1,
    "max_id": 917,
    "last_change": "2023-01-31 17:01:27",
    "last_seen": "2023-01-31 17:01:27",
    "name": "TT_RadarFRM20",
    "units": "0:Level_m 1:Dist_m 2:Sig_dB 3:oC 4:Sig_Qual 5:Bat_V 90:V(Bat) 91:oC(int)
                  100:PackInd 101:Travel(sec)",
    "vals": "0:12.92377377 1:2.07622695 2:31.18152618 3:6 4:0 5:5.94999981 90:3.366
                                                          91:4.25 100:1054 101:3678",
    "cookie": 1670531152,
    "transfer_cnt": 513,
    "lines_cnt": 1056,
    "warnings_cnt": 0,
    "alarms_cnt": 0,
    "err_cnt": 0,
    "anz_lines": 917
  },
```

```
  "get_count": 11,
  "get_data": [
    {
      "id": 10,
      "line_ts": "2022-12-04 23:19:52",
      "calc_ts": "2022-12-04 23:00:03",
      "type": "val",
      "line": "0:12.95418644 1:2.04581404 2:32.88109589 3:2.5 4:0 5:5.98999977 90:3.546
                                                          91:-0.5 100:112 101:1189"
    },
    {
      "id": 11,
      "line_ts": "2022-12-05 01:28:34",
      "calc_ts": "2022-12-05 00:00:03",
      "type": "msg",
      "line": "<NT AUTO>"
    },
    {
      "id": 12,
      "line_ts": "2022-12-05 01:28:34",
      "calc_ts": "2022-12-05 00:00:03",
      "type": "val",
      "line": "0:12.95018387 1:2.04981589 2:32.87070084 3:2 4:0 5:5.96000004 90:3.525
                                                          91:-0.25 100:113 101:5311"
    },
    {
      "id": 13,
      "line_ts": "2022-12-05 01:28:35",
      "calc_ts": "2022-12-05 01:00:03",
      "type": "msg",
      "line": "<NT AUTO>"
    },
    {
      "id": 14,
      "line_ts": "2022-12-05 01:28:35",
      "calc_ts": "2022-12-05 01:00:03",
      "type": "val",
      "line": "0:12.94942188 1:2.05057788 2:33.32662201 3:2 4:0 5:5.98000002 90:3.515
                                                          91:-0.25 100:114 101:1712"
    },
…

    {
      "id": 20,
      "line_ts": "2022-12-05 06:03:49",
      "calc_ts": "2022-12-05 04:00:03",
      "type": "val",
      "line": "0:12.95699501 1:2.0430069 2:32.46614075 3:1 4:0 5:5.98999977 90:3.508
                          91:-1.5 100:117 101:7426"
    }
  ],
  "status": "0 OK (11.0469 msec)"
}
```


Special: Auflösung von Messwerten

Messwerte sind zwar intern immer Fließkommazahlen, aber um den Speicherverbrauch zu optimieren, kann die Anzahl der Nachkommastellen zwischen 0 und 8 variieren.

Special: Alarme oder Fehlermeldungen in Messwerten

Ein Kanal wird zwar i.d.R. einen Messwert als Zahl enthalten, aber es ist auch möglich, Alarme oder Fehlermeldungen einzubauen, daher gibt es exakt drei Möglichkeiten. Eine Kombi (z.B. Alarm plus Error ist nicht möglich):

- „Normal“: Das ist der Normalfall: KanalIndex:Messwert - Beispiel: `2:32.46614`
- „Alarm“: Hat vor dem Messwert ein ‚*‘ vorangestellt - Beispiel: `2:*32.46614`
- „Errors“: Nach dem KanalIndex kommt ein Fehlertext - Beispiel: `2:ErrorNoReply`
  (Der Fehlertext enthält keine Leerzeichen)

Parameter des Gerätes holen

http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparam

liefert:


```
// 20230131184857
// http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparam

{
  "overview": {
    "mac": "26FEA299F444F836",
    "db_now": "2023-01-31 17:48:57",
    "min_id": 371,
    "max_id": 1824,
    "last_change": "2023-01-31 15:47:47",
    "last_seen": "2023-01-31 14:58:24",
    "name": "Meriva_F836",
    "units": "0:mcnt 90:V(Bat) 91:°C(int) 92:%rH(int) 93:mAh(Bat) 94:mBar",
    "vals": "0:Test 90:5.47 91:0.75 92:67.2 93:94.54 94:1016.0 100:602 101:7139",
    "cookie": 1666462431,
    "transfer_cnt": 657,
    "lines_cnt": 1800,
    "warnings_cnt": 13,
    "alarms_cnt": 0,
    "err_cnt": 1,
    "anz_lines": 1454
  },
  "par_pending": false,
  "iparam": [
    {
      "line": "@100",
      "info": "*@100_System (Common, Line 0)"
    },
    {
      "line": "1400",
      "info": "*DEVICE_TYP (Common, Line 1)"
    },
    {
      "line": "10",
      "info": "*MAX_CHANNELS (Common, Line 2)"
    },
    {
      "line": "31",
      "info": "*HK_FLAGS (Common, Line 3)"
    },
    {
      "line": "1666462431",
      "info": "*NewCookie [Parameter 10-digit Timestamp.32] (Common, Line 4)"
    },
    {
      "line": "Meriva_F836",
      "info": "Device_Name[BLE:$11/total:$41] (Common, Line 5)"
    },
    {
```

```
    "line": "21600",
    "info": "Period_sec[10..86400] (Common, Line 6)"
  },
  {
    "line": "0",
    "info": "Period_Offset_sec[0..(Period_sec-1)] (Common, Line 7)"
  },
  {
    "line": "1800",
    "info": "Period_Alarm_sec[0..Period_sec] (Common, Line 8)"
  },
  {
    "line": "0",
    "info": "Period_Internet_sec[0..604799] (Common, Line 9)"
  },
  {
    "line": "0",
    "info": "Period_Internet_Alarm_sec[0..Period_Internet_sec] (Common, Line 10)"
  },
  {
    "line": "3600",
    "info": "UTC_Offset_sec[-43200..43200] (Common, Line 11)"
  },
  {
    "line": "3",
    "info": "Flags (B0:Rec B1:Ring) (0: RecOff) (Common, Line 12)"
  },
  {
    "line": "31",
    "info": "HK_flags (B0:Bat B1:Temp B2.Hum B3.Perc) (Common, Line 13)"
  },
  {
    "line": "6",
    "info": "HK_reload[0..255] (Common, Line 14)"
  },
  {
    "line": "1",
    "info": "Net_Mode (0:Off 1:OnOff 2:On_5min 3:Online) (Common, Line 15)"
  },
  {
    "line": "0",
    "info": "ErrorPolicy (O:None 1:RetriesForAlarms, 2:RetriesForAll) (Common, Line
         16)"
  },
  {
    "line": "-10",
    "info": "MinTemp_oC[-40..10] (Common, Line 17)"
  },
  {
    "line": "0",
    "info": "Period_Internet_Offset[0..Period_Internet_sec] (Common, Line 18)"
  },
  {
    "line": "@0",
    "info": "*@ChanNo (Chan 0, Line 19)"
  },
  {
    "line": "1",
    "info": "Action[0..65535] (B0:Meas B1:Cache B2:Alarms) (Chan 0, Line 20)"
  },
  {
    "line": "1024",
    "info": "Physkan_no[0..65535] (Chan 0, Line 21)"
  },
  {
    "line": "",
    "info": "Kan_caps_str[$8] (Chan 0, Line 22)"
  },
```

```
    {
      "line": "0",
      "info": "Src_index[0..255] (Chan 0, Line 23)"
    },
    {
      "line": "mcnt",
      "info": "Unit[$8] (Chan 0, Line 24)"
    },
    {
      "line": "0",
      "info": "Mem_format[0..255] (Chan 0, Line 25)"
    },
    {
      "line": "0",
      "info": "DB_id[0..2e31] (Chan 0, Line 26)"
    },
    {
      "line": "0.0",
      "info": "Offset[float] (Chan 0, Line 27)"
    },
    {
      "line": "1.0",
      "info": "Factor[float] (Chan 0, Line 28)"
    },
    {
      "line": "0.0",
      "info": "Alarm_hi[float] (Chan 0, Line 29)"
    },
    {
      "line": "0.0",
      "info": "Alarm_lo[float] (Chan 0, Line 30)"
    },
    {
      "line": "0",
      "info": "Messbits[0..65535] (Chan 0, Line 31)"
    },
    {
      "line": "06000003",
      "info": "Xbytes[$32] (Chan 0, Line 32)"
    }
  ],
  "status": "0 OK (27.5369 msec)"
}
```


Parameter am Gerät ändern

Wird beim nächsten Transfer übertragen. Bis dahin sind Änderungen wieder zurücknehmbar. Die geänderten Parameter werden auf Plausibilität geprüft. Solange die Parameter nicht übertragen worden sind, gelten sie als „pending“ (‚par_pending: bool‘).

Hier wird ein neuer Name gesetzt und die Periode auf 3601 Sekunden erhöht.

http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparamchange&iparam[5]=NeuMeriva&iparam[6]=3601

liefert:


```
// 20230221164719
// http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparamchange&iparam[5]=NeuMeriva&iparam[6]=3601

{
  "overview": {
    "mac": "26FEA299F444F836",
    "db_now": "2023-02-21 15:47:19",
    "min_id": 371,
```

```
    "max_id": 1822,
    "last_change": "2023-02-11 13:28:46",
    "last_seen": "2023-01-29 17:00:29",
    "name": "NeuMeriva",
    "units": "0:mcnt 90:V(Bat) 91:°C(int) 92:%rH(int) 93:mAh(Bat) 94:mBar",
    "vals": "0:0 90:5.628 91:13.9 92:67.2 93:94.54 94:1016.0",
    "cookie": 1666462431,
    "transfer_cnt": 656,
    "lines_cnt": 1799,
    "warnings_cnt": 13,
    "alarms_cnt": 0,
    "err_cnt": 0,
    "anz_lines": 1452
  },
  "par_pending": true,
  "status": "0 OK (13.7999 msec)"
}
```


Bei Bedarf können auch neue Kanäle erzeugt werden, dazu einfach mindestens einen Wert des Kanals belegen, fehlende Werte werden aus dem letzten verfügbaren Kanal kopiert, aber ‚action' des neuen Kanals ist per Default 0. Die letzten Kanäle ohne ‚action' (also ‚action' ist 0) werden automatisch entfernt (Parameterdatei hat also immer minimale Größe).

Geänderte Parameter zurücknehmen

‚par_pending' wird dadurch wieder auf false gesetzt (genauso wie nach einer erfolgreichen Übertragung).

http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparamunpend

liefert:


```
/ 20230221165104
// http://localhost/ltx/sw/w_php/w_pcp.php?s=26FEA299F444F836&k=ABC&cmd=iparamunpend

{
  "overview": {
    "mac": "26FEA299F444F836",
    "db_now": "2023-02-21 15:51:04",
    "min_id": 371,
    "max_id": 1822,
    "last_change": "2023-02-11 13:28:46",
    "last_seen": "2023-01-29 17:00:29",
    "name": "Meriva_F836",
    "units": "0:mcnt 90:V(Bat) 91:°C(int) 92:%rH(int) 93:mAh(Bat) 94:mBar",
    "vals": "0:0 90:5.628 91:13.9 92:67.2 93:94.54 94:1016.0",
    "cookie": 1666462431,
    "transfer_cnt": 656,
    "lines_cnt": 1799,
    "warnings_cnt": 13,
    "alarms_cnt": 0,
    "err_cnt": 0,
    "anz_lines": 1452
  },
  "par_pending": false,
  "status": "0 OK (30.6509 msec)"
}
```

Positionsdaten

Es gibt viele verschiedene Möglichkeiten "wie" Positionsdaten entstehen. können. Manche haben z.B. zusätzlich Höhe, Geschwindigkeit, Richtung, Signalqualität und/oder sind genauer, manche weniger genau. In der Evaluierung wir uns nur auf "Zellenlokalisierung via Google" beschränkt.

Vorgesehen sind aber z. Bsp.:

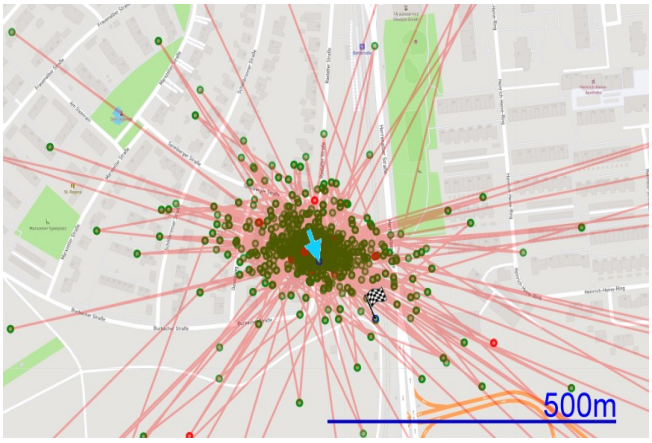
- echtes GPS via GPS/GNSS-Empfänger
- „pseudo-“ GPS von Sensoren (z.B. LoRa, Satelliten-Modems, Beacons, …)
- Zellenlokalisierung nachträglich z.B. via Google-API
- Zellenlokalisierung über das Gerät z.B. mittels QUECLOCATOR (in mehreren „Qualitäten“ möglich, siehe Quectel-Application-Notes: Einfach, Triangulation, per BLE oder WiFi-Scan, …)
- und noch einige mehr...

Diese Daten sind alle in den Zeilen der Messwerte als „Meta-Daten“ eingepflegt (Metadaten sind immer in "<" / ">" - Paaren).

Genauigkeit und Energieverbrauch von Positionsdaten

GPS (GNSS)

Die Genauigkeit von GPS-Daten hängt sehr von der Signalqualität ab! Ein Standort mit sehr gutem Empfang und längerer Messzeit kommt tatsächlich auf Genauigkeiten im Meter-Bereich. Bei schlechten Signalen, Abschattungen durch Gebäude, atmosphärischen Störungen, etc. sind auch bei GPS größere Abweichungen möglich. Hier z.B. ein Standort zwischen 2 Gebäuden und schlechtem Empfang:



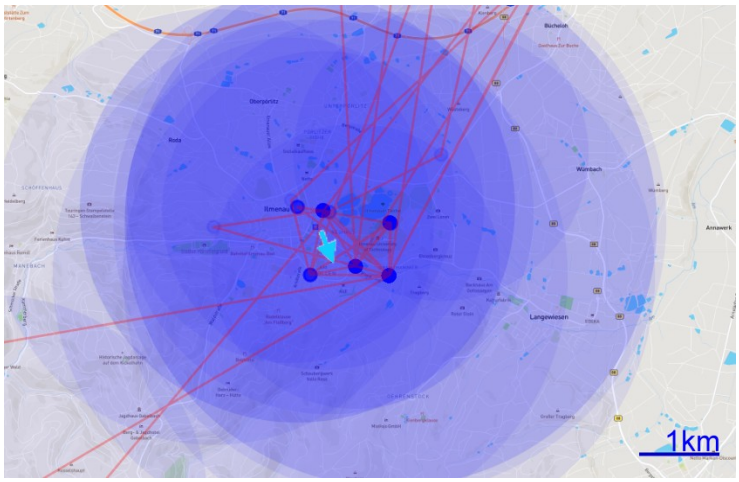
Interessant ist: Der arithmetische Mittelwert mehreren Positionen in denen das Gerät nicht bewegt wurde (es enthält standardmäßig einen Bewegungssensor (Unit: „mcnt“)), liegt sehr schnell (trotz einzelner hoher Abweichungen) auf wenige Meter genau (Pfeil).

Der Energieverbrauch von GPS-Positionen ist vergleichsweise hoch, denn bei schlechtem Empfang kann es unter Umständen mehrere Minuten dauern, bis, wenn überhaupt, gültige Koordinaten gefunden wurden. Um Energie zu sparen ist daher die maximale Suchzeit auf Geräten i.d.R: auf

maximal 2 Minuten beschränkt (dieser Wert ist aber einstellbar). Ganz grob kann man pro GPS-Position einen Energieverbrauch von ca. 0.01 mAh (bei gutem Empfang) bis zu 0.05 mAh (je nach Modul deutlich unterschiedlich) überschlagen.

Pseudo-GPS-Zellenlokalisierung per Google-API

Im urbanen Gebiet sind die Funkzellen recht nahe beieinander (im Stadtgebiete teilweise alle 100 Meter). „Meistens“ bucht sich das Gerät in eine der stärksten Zellen in der Umgebung ein. Auch so erhält man durch Mittelung oft recht gute Genauigkeiten im Bereich < 100 Metern. Im ländlichen Gebiet sind die Zellen meist weniger dicht. Im „völlig freien Feld“ sind allerdings Abweichungen von 2-5 km durchaus möglich.



Hier befand sich das Gerät in einer Kleinstadt, nur mit Zellenortung. Die Zentren sind die kleinen blauen Kreise, die leicht blauen, äußeren Kreisflächen geben den ungefähren Radius an, in dem die jeweilige Funkzelle verfügbar war. Fazit: auch ohne „echtes“ GPS lassen sich Positionen erstaunlich gut bestimmen.

Wichtig: Wie unten erläutert wird, kann es vorkommen, dass unter Umständen die Position einer Funkzelle der Google-API unbekannt ist. Es kann aber auch sein, dass beispielsweise bei Großveranstaltungen temporäre Funkzellen aufgebaut werden. Die Google-API lernt zwar recht schnell und korrigiert gegebenenfalls auch die gelieferten Positionen, dies dauert aber meist eine gewisse Zeit. Ergo: es kann durchaus, wenn auch selten, vorkommen, dass z.B. eine Position in als Ausreißer in einer völlig anderen Stadt liegt.

Zellenlokalisierung per Google-API verbraucht keinerlei Batteriekapazität auf dem Gerät, daher sollte dies immer die präferierte Methode sein!

Pseudo-GPS-Positionen vom Gerät erzeugt

Hier ist keine allgemeine Aussage zu Genauigkeit oder Energieverbrauch möglich. Speziell für den QUECLOCATOR (siehe unten) gibt es aber Erfahrungswerte für die Betriebsarten „Einfach“ und „Triangulation“:

- Einfach: Die gelieferten Koordinaten entsprechen in etwa denen der Google-API, allerdings benötigt das Gerät dazu eine – wenn auch geringe – Menge

Batterie-Kapazität (ganz grob im Bereich von ca. 0.005 mAh).

- Triangulation: Dabei bucht sich das Gerät reihum in die benachbarten Zellen ein. Dies kann durchaus 1 Minute dauern. Die gelieferten Koordinaten sind (meist) deutlich besser als „Einfach“ oder „Google-API“, der Gesamt-Energie-Verbrauch ist aber deutlich höher und liegt ganz grob im Bereich ca. 0.03 mAh)

Wichtig: Die Verwendung des QUECLOCATORS erfordert eine Registrierung bei Quectel.

Speziell zu Positionsdaten:

Positionen aus echten GPS-Modulen

Werden in den Daten wie Messwerte abgespeichert (Units ‚Lat‘ und ‚Lng‘).

Z.B. vom "alten Tracker im schwarzen Rundgehäuse“:

```
!U 0:Lat 1:Lng 2:mcnt 3:sec_on 4:cnt_fix 90:V(Bat) 91:oC(int) 92:rH 93:mAh(Bat)!
1640991600 0:48.963371 1:8.404253 2:0 3:60 4:20 90:5.890 91:11.0 92:69.5 93:687.833
<NT AUTO OK(2)>
1641013200 0:48.963356 1:8.403930 2:0 3:59 4:20 90:5.901 91:10.0 92:69.6 93:688.712
```

Pseudo-GPS-Daten aus Zellenortung

Sind z.B. Google-API erzeugt worden. Meistens kennt Google die Zellen, es kann aber auch vorkommen, dass Zellen z.B. zu neu sind. Deren Daten werden dann als Fehler weitergereicht, vielleicht kann ja der Cloud Provider das besser auflösen?Die Metadaten sind nach dem Schema "<CELLOC Lat Lng RadiusInMetern >" aufgebaut, z.B. wie von einem aktuellen LTRACK (bei diesem Gerät sind sowohl Google-Zelldaten (gelb), als auch echte GPS-Daten (grün) dabei, man beachte den Unterschied!):

```
<NT AUTO OK(2)>
!1654556400 0:48.963314 1:8.403700 2:0 3:89 4:20 90:5.937 91:20.6 92:57.5 93:50.1431853
<CELLOC 48.9589 8.38657 1393>
<NT AUTO OK(4)>
!1654578000 0:48.963497 1:8.408330 2:0 3:63 4:30 90:5.926 91:17.7 92:58.0 93:51.033
```

Nicht von Google auflösbare Zellen werden als Fehlermeldung in den Metadaten weitergegeben im Format "<ERROR: CELLOC: $einfo>" abgespeichert, wobei $info den Fehlercode und die Zellendaten enthält.

Als Test: diese Zelle z.B. `mcc=262, net=1 ,lac=54055, cid=38222853` sollte ganz grob bei „48.9,8.40“ (Karlsruhe) liegen, wurde aber von Google (Anf. 2023) nicht gefunden:

https://flexgate.org/ltx_api/gcells/gcells.php?k=TESTTOKEN&s=22727759176F8D69&mcc=262&net=1&lac=54055&cid=38222853

Pseudo-GPS-Daten des Geräte, z.B. QUECLOCATOR

Die LTRACK mit Quectel-Modem haben die Möglichkeit einen Service “QUECLOCATOR“ zu nutzen. Dieser kann auf verschiedene Arten Positionen (je nach Art mehr oder weniger genau)

erzeugen. Aktuell (Anf. 2023) ist QUECLOCATOR implementiert, wird aber (noch) nicht verwendet. Hier nur zur Vollständigkeit beschrieben:

Generierte Positionsdaten werden als "User-Content" "<UCMD: $info>" abgespeichert. Hier für den QUECLOCATOR wäre der Aufbau: "<UCMD: q%u: %d,%f,%f>" mit den Parametern: Art (0:Einfach, 1:Triangulation), Ergebnis als Zahlenwert (0: für OK,<>0: Siehe Quectel-App-Note), Lat, Lng.

Bsp: `"<UCMD: q1: 0,48.9921,8.523>"` (Interessant: QUECLOATOR liefert keine Radius-Infos)

(Andere, zukünftige Quellen haben dann gegebenenfalls einen anderen Aufbau des "<UCMD: $info>").

*** Weitere Kommandos/Infos nach Bedarf / Absprache ***