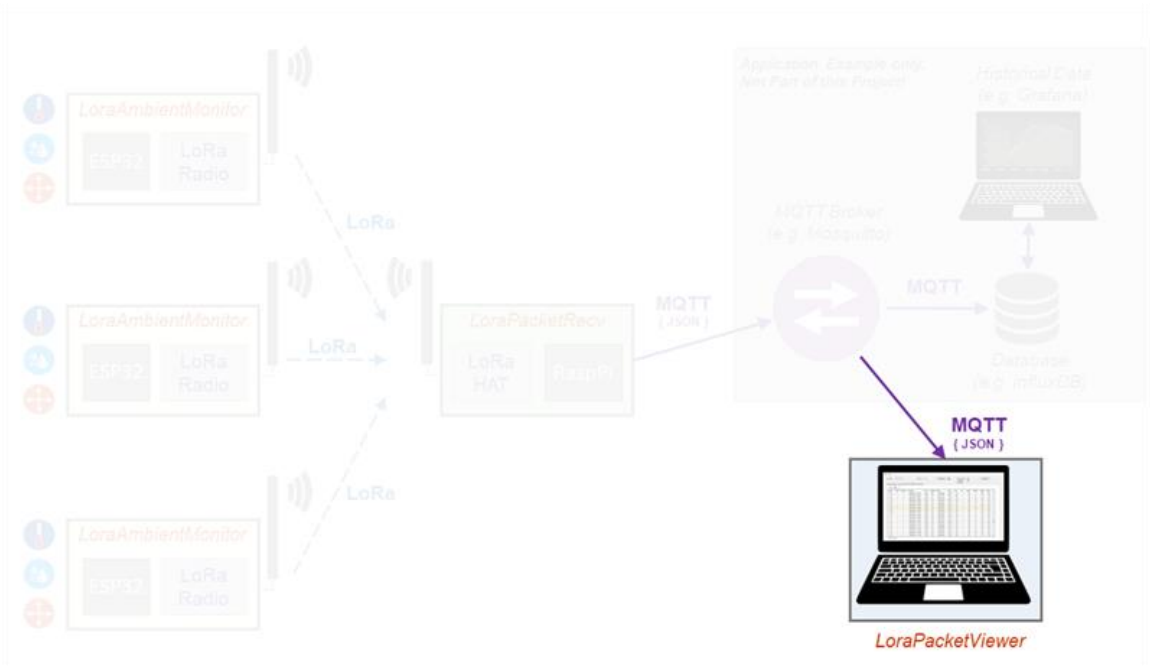
Projekt: https://github.com/ronaldsieber/LoraAmbientMonitor
Teilprojekt: https://github.com/ronaldsieber/LoraAmbientMonitor/LoraPacketViewer
Lizenz: MIT
Autor: Ronald Sieber

LoraPacketViewer

Dieses Visual Studio C#-Projekt ist Teil des Hauptprojektes *LoraAmbientMonitor* und realisiert eine GUI-Applikation zur Anzeige der vom Teilprojekt *<LoraPacketRecv>* entweder online an den MQTT-Broker gesendeten JSON-Records oder der offline erzeugten Logfiles. Dies umfasst sowohl die Bootup- als auch die Sensordaten-Pakete. Diese GUI-Applikation dient in erster Linie der Inbetriebnahme und Diagnose. Für den Produktiveinsatz ist sie nicht erforderlich.



[LoraPacketViewer - *LoraPacketViewer*]

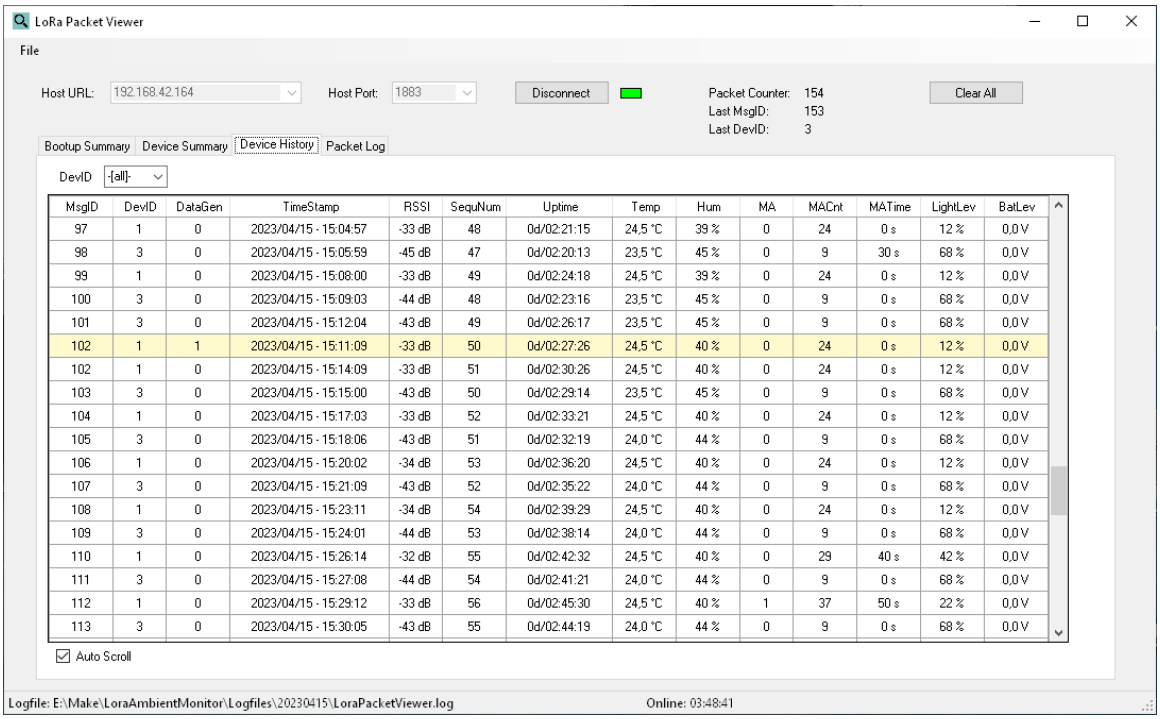
Der *LoraPacketViewer* umfasst in seinem Hauptfenster folgende 4 Tabsheets:

- **Bootup Summary:**
  Zeigt tabellarisch für jede der 16 möglichen DevID's den zuletzt verarbeiteten Bootup-Record an (Gerätekonfiguration)

- **Device Summary:**
  Zeigt tabellarisch für jede der 16 möglichen DevID's den zuletzt verarbeiteten Sensordaten-Record an (aktuelle Umgebungsdaten)

- **Device History:**
  Zeigt tabellarisch die Historie aller Sensordaten-Record (aktuelle Umgebungsdaten) für alle oder ausgewählte DevID's in der Reihenfolge ihrer Verarbeitung an

- **Paket Log:**
  Listet Statusinformationen und alle JSON-Records (sowohl Bootup- als auch Sensordaten-Pakete) in der Reihenfolge ihrer Verarbeitung an



MsgID	DevID	DataGen	TimeStamp	RSSI	SequNum	Uptime	Temp	Hum	MA	MACnt	MATime	LightLev	BatLev
97	1	0	2023/04/15 - 15:04:57	-33 dB	48	0d/02:21:15	24,5 °C	39 %	0	24	0 s	12 %	0,0 V
98	3	0	2023/04/15 - 15:05:59	-45 dB	47	0d/02:20:13	23,5 °C	45 %	0	9	30 s	68 %	0,0 V
99	1	0	2023/04/15 - 15:08:00	-33 dB	49	0d/02:24:18	24,5 °C	39 %	0	24	0 s	12 %	0,0 V
100	3	0	2023/04/15 - 15:09:03	-44 dB	48	0d/02:23:16	23,5 °C	45 %	0	9	0 s	68 %	0,0 V
101	3	0	2023/04/15 - 15:12:04	-43 dB	49	0d/02:26:17	23,5 °C	45 %	0	9	0 s	68 %	0,0 V
102	1	1	2023/04/15 - 15:11:09	-33 dB	50	0d/02:27:26	24,5 °C	40 %	0	24	0 s	12 %	0,0 V
102	1	0	2023/04/15 - 15:14:09	-33 dB	51	0d/02:30:26	24,5 °C	40 %	0	24	0 s	12 %	0,0 V
103	3	0	2023/04/15 - 15:15:00	-43 dB	50	0d/02:29:14	23,5 °C	45 %	0	9	0 s	68 %	0,0 V
104	1	0	2023/04/15 - 15:17:03	-33 dB	52	0d/02:33:21	24,5 °C	40 %	0	24	0 s	12 %	0,0 V
105	3	0	2023/04/15 - 15:18:06	-43 dB	51	0d/02:32:19	24,0 °C	44 %	0	9	0 s	68 %	0,0 V
106	1	0	2023/04/15 - 15:20:02	-34 dB	53	0d/02:36:20	24,5 °C	40 %	0	24	0 s	12 %	0,0 V
107	3	0	2023/04/15 - 15:21:09	-43 dB	52	0d/02:35:22	24,0 °C	44 %	0	9	0 s	68 %	0,0 V
108	1	0	2023/04/15 - 15:23:11	-34 dB	54	0d/02:39:29	24,5 °C	40 %	0	24	0 s	12 %	0,0 V
109	3	0	2023/04/15 - 15:24:01	-44 dB	53	0d/02:38:14	24,0 °C	44 %	0	9	0 s	68 %	0,0 V
110	1	0	2023/04/15 - 15:26:14	-32 dB	55	0d/02:42:32	24,5 °C	40 %	0	29	40 s	42 %	0,0 V
111	3	0	2023/04/15 - 15:27:08	-44 dB	54	0d/02:41:21	24,0 °C	44 %	0	9	0 s	68 %	0,0 V
112	1	0	2023/04/15 - 15:29:12	-33 dB	56	0d/02:45:30	24,5 °C	40 %	1	37	50 s	22 %	0,0 V
113	3	0	2023/04/15 - 15:30:05	-43 dB	55	0d/02:44:19	24,0 °C	44 %	0	9	0 s	68 %	0,0 V

[LoraPacketViewer]

Online-Modus

Im Online-Modus verbindet sich der *LoraPacketViewer* mit dem MQTT-Broker und empfängt von diesem alle vom Teilprojekt *<LoraPacketRecv>* gesendeten JSON-Records. Dazu sind die beiden Felder *"Host URL"* und *"Host Port"* entsprechend IP-Adresse und Portnummer des Brokers zu konfigurieren. Mit dem Button *"Connect"* wird die Verbindung zum Broker hergestellt. Nach erfolgreichem Verbindungsaufbau wechselt der Status-Indikator von rot auf grün.

Durch die Kennzeichnung der MQTT-Pakete als *"Retain"* speichert der Broker jeweils die letzte empfange Nachricht eines Topics zwischen. Das bewirkt, dass der *LoraPacketViewer* bei seiner Anmeldung am Broker von jedem Sensormodul dessen zuletzt gesendete Boot-Record (Gerätekonfiguration) und Sensordaten-Record (aktuelle Umgebungsdaten) erhält. Dadurch kennt der *LoraPacketViewer* unmittelbar nach seiner Anmeldung am Broker den aktuellen Zustand des Gesamtsystems.

Offline-Modus

Im Offline-Modus wird über *"File -> Load…"* eine Datei mit JSON-Records eingelesen und verarbeitet. Die Datei kann entweder zuvor vom *LoraPacketViewer* selbst über *"File -> Save as…"* erstellt wurden sein oder sie wurde vom Teilprojekt *<LoraPacketRecv>* mit dem Kommandozeilenparameter *"-l=<msg_file>"* erzeugt.

MQTT-Kommunikation

Als MQTT Client Implementierung für Kommunikation zwischen Broker und *LoraPacketViewer* wird das Package *"M2Mqtt"* verwendet. Diese wird innerhalb von *LoraPacketViewer* durch das File *LpvMqttClient.cs* gekapselt.

Als Handler für den Empfang abonnierter MQTT-Nachrichten wird die Methode `ClientHandler_MqttMsgPublishReceived` registriert:

```
// register handler to process received messages
m_MqttClientInst.MqttMsgPublishReceived += ClientHandler_MqttMsgPublishReceived;
```

Diese nutzt das Delegate `InvokeDlgt_DispatchLoraNodePackets` für den Aufruf der Methode `DispatchLoraNodePackets` im GUI Thread und übergibt diesem dabei Topic und Payload:

```
m_AppForm.DispatchLoraNodePackets(strTopic_p, strPayload_p);
```

Die weitere Verarbeitung der per MQTT empfangenen JSON-Records erfolgt anschließend im Kontext des GUI Threads.

Parsen und Anzeigen der JSON-Records

Zum Parsen der empfangenen JSON-Records wird das Package *"Newtonsoft.Json"* verwendet. Die darin enthaltene statische Methode `JsonPacketSerializer.Deserialize` unterstützt das Deserialisieren von JSON-Objekten in Custom Types.

Die Methode `DispatchLoraNodePackets` im GUI Thread bildet die zentrale Instanz zur Deserialisierung der JSON-Objekte und deren anschließende Weiterverteilung an die im Hauptfenster angesiedelten 4 Tabsheets. Diese Methode ist sowohl für die Verarbeitung online per MQTT empfangener JSON-Records als auch für die Verarbeitung offline aus einer Datei gelesenen JSON-Records zuständig.

Zunächst wird für jeden zu verarbeitenden JSON-Record ermittelt, ob es sich um einen Boot- oder Sensordaten-Record handelt. Dazu wird die Methode `JsonPacketSerializer.Deserialize` mit dem Custom Type `JsonPacketInfo` aufgerufen:

```
JsonPacketReader = new JsonTextReader(new StringReader(strPayload_p));
PacketInfo = JsonPacketSerializer.Deserialize<JsonPacketInfo>(JsonPacketReader);
PacketType = PacketInfo.GetPacketType();
```

Das Parsen der Boot- und Sensordaten-Records erfolgt anschließend in gleicher Weise unter Anwendung der Custom Typen `JsonStationBootup` und `JsonStationData`. Alle Custom Typen sind in der Datei *LpvJsonPackets.cs* implementiert.

In Abhängigkeit vom Paket-Typ erfolgt dann die Anzeige der Daten in den 4 Tabsheets.

Verwendete Drittanbieter Komponenten

1. **MQTT Client**

   Package: M2Mqtt
   Version: 4.3.0
   Autor: Paolo Patierno

   Das Package wird innerhalb der Visual Studio IDE mit dem NuGet Packet Manager installiert.

2. **JSON Deserialisierung**

   Package: Newtonsoft.Json
   Version: 13.0.2
   Autor: James Newton-King

   Das Package wird innerhalb der Visual Studio IDE mit dem NuGet Packet Manager installiert.