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

LoraAmbientMonitor

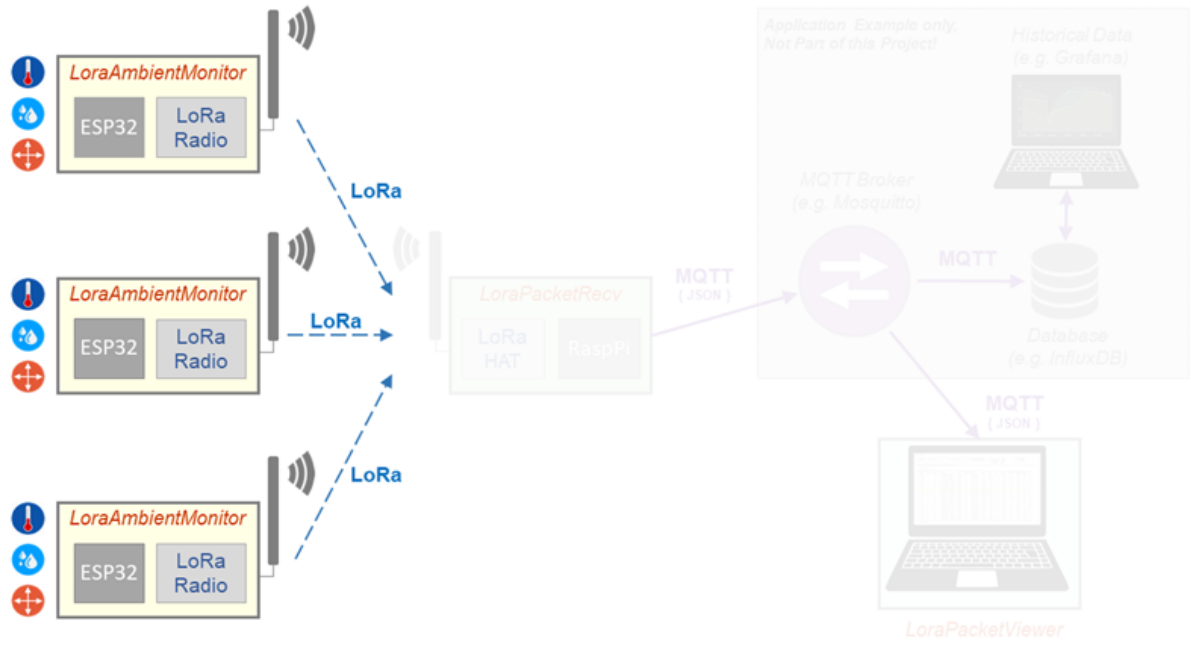
Dieses Arduino-Projekt ist Teil des Hauptprojektes *LoraAmbientMonitor* und umfasst die ESP32-Firmware für die Sensorbaugruppe basierend auf dem Hardware Projekt <*LoRaAmbientMonitor_PCB*>. Die Sensorbaugruppe erfasst folgende Umgebungsdaten:

- Temperatur (DHT22)
- Luftfeuchtigkeit (DHT22)
- Bewegungen im Raum (PIR-Sensor HC-SR501)
- Umgebungshelligkeit (ALS-PDIC243 @ ADS1115)
- optional: Spannung der KFZ-Startbatterie (ADS1115)

Alle von der Sensorbaugruppe erfassten Umgebungsdaten werden mittels LoRa-Datenpaketen an die im Teilprojekt <*LoraPacketRecv*> beschriebene Empfängerbaugruppe übertragen.

Das für den *LoraAmbientMonitor* verwendete ESP32-Modul vom Typ *"Heltec ESP32 WiFi LoRa OLED"* beinhaltet bereits den LoRa-Chipsatz SX1276 sowie ein 0.96" OLED-Display, das zur lokalen Anzeige der Sensorwerte genutzt wird.

Die dem *Heltec ESP32 Kit* beiliegende Stick Antenne eignet sich aufgrund ihrer kompakten und platzsparenden Ausführung gut für Entwicklung und kurze Entfernungen. Zur Erhöhung der Reichweite sollte im Produktiveinsatz besser eine mit Koaxialkabel abgesetzte externe Antenne (6dBi oder besser) verwendet werden.



[Project Overview - LoraAmbientMonitor]

Geräte-Laufzeitkonfiguration

Die Geräte-Konfigurationseinstellungen zur Laufzeit erfolgt mit Hilfe des 4-fach DIP-Schalter SW2:

- **DIP3/4 = DevID:** Die Node- bzw. DevID wird in die LoRa Datenpakte eingebettet und ermöglicht so dem Empfänger die Zuordnung der empfangenen Pakete zur jeweiligen Quelle. Damit ist es möglich, dieselbe Firmware unverändert für mehrere Boards gleichzeitig zu verwenden. Die jeweilige DevID des Boards wird im OLED-Display angezeigt.

- **DIP2 = Pause SEN-HC-SR501 (IR Motion Sensor):** Je nach eingestellter Empfindlichkeit löst der Bewegungsmelder durch Störbeeinflussung vom LoRa-Modul auch fälschlicher weise während des Sendes von LoRa-Paketen aus. Um Fehlmeldungen zu verhindern, wird der Signalausgang des Bewegungsmelders während der LoRa-Übertragung für das mittels `SEN_HC_SR501_PAUSE_PERIOD` konfigurierte Zeitintervall ignoriert.

- **DIP1 = Generieren asynchroner LoRa Sende-Events:** Im Normalfall erfolgt die Übertragung der Sensordaten zyklisch, typisch im Stundentakt. Bei Freigabe asynchroner LoRa Sende-Events werden beim Auslösen des Bewegungsmelders sowie beim An- oder Abklemmen KFZ-Startbatterie asynchrone LoRa-Pakete gesendet, so dass die aktuellen Informationen auf Empfängerseite unmittelbar verfügbar sind. Asynchrone LoRa-Pakete werden jedoch frühestens nach Ablauf des mittels `LORA_PACKET_INHIBIT_TIME` konfigurierte Zeitintervalls versendet. Ausführliche Details zum LoRa-Zeitregime beschreibt der Abschnitt *"LoRa-Konfiguration Sektion"* weiter unten.

Insbesondere für Inbetriebnahme und Diagnose erlaubt der CFG-Taster die Verkürzung der LoRa-Sendeintervalle (*"Commissioning Mode"*):

- **CFG-Taster:** Ist beim Booten (Power-on oder Reset) der CFG-Taster gedrückt, nutzt die Sensorbaugruppe verkürzte Sendeintervalle. Das damit verbundene häufigere Senden von Datenpaketen erleichtert Diagnosen im Fehlerfall. Zudem können Maßnahmen wie beispielsweise das Verändern bzw. Ausrichten der Antennen-Position schneller bewertet werden.

Bei aktivierten *"Commissioning Mode"* blinkt die grüne LED permanent um auf verkürzten LoRa-Sendeintervalle hinzuweisen.

Statusanzeigen

Die *LoraAmbientMonitor* Sensorbaugruppe nutzt 3 on-board LED's mit folgenden Bedeutungen sowie das OLED-Display:

- **LED Grün:** Dauerlicht von mehreren Sekunden = Senden eines LoRa-Paketes, permanentes Blinken = Baugruppe im *"Commissioning Mode"* mit verkürzten LoRa-Sendeintervallen

- **LED Rot:** Dauerlicht von mehreren Sekunden = Auslösen des SEN-HC-SR501 (IR Motion Sensor)

- **LED Gelb:** Angeschlossene KFZ-Startbatterie erkannt

- **OLED-Display:** Anzeige von Geräteparametern und aktuellen Sensormesswerten

LoRa Bootup Paket

Das Bootup Paket wird vom *LoraAmbientMonitor* einmalig am Ende der Sketch `setup()` Funktion versendet. Es beinhaltet verschiedene Informationen zur Geräte-Firmware (Versionsnummer, Konfiguration), sowie die mittels 4-fach DIP-Schalter vorgenommene Laufzeitkonfiguration. Den detaillierten Aufbau beschreibt die Struktur `tLoraBootupHeader` in der Headerdatei *<LoraPacket.h>*. Der Bootup-Header wird durch eine zusätzliche 16Bit CRC vom Typ `uint16_t` gesichert. Somit ergibt sich eine Gesamtgröße von 80Bit (= 10 Bytes).

Jedes LoRa-Datenpaket ist durch die Struktur `tLoraDataPacket` definiert. Im Fall des Bootup-Paketes wird jedoch nur der Header mit der Struktur `tLoraBootupHeader` genutzt. Die nachfolgenden drei für Sensordaten vorgesehenen Elemente vom Typ `tLoraDataRec` sind komplett mit Null gefüllt. Insgesamt besitzt das Bootup-Paket eine Payload-Länge von 40 Bytes.

Die Kodierung der Konfigurationsdaten in das over-the-air Format erfolgt in der Methode `LoraPayloadEncoder:: EncodeTxBootupPacket()`.

LoRa Sensordaten Paket

Die Sensordaten Pakete werden vom *LoraAmbientMonitor* periodisch innerhalb der Sketch `loop()` Funktion versendet. Dazu werden die Sensordaten in einem kompakten 64Bit-Feld (`uint64_t`), das als Struktur vom Typ `tLoraDataRec` definiert ist, übertragen. Dieser Sensordatensatz wird durch eine zusätzliche 16Bit CRC vom Typ `uint16_t` gesichert. Somit ergibt sich für einen Datensatz inclusive CRC eine Gesamtgröße von 80Bit (= 10 Bytes).

Jedes LoRa-Datenpaket ist durch die Struktur `tLoraDataPacket` definiert. Es beinhaltet einen Header vom Typ `tLoraDataHeader` gefolgt von drei Generationen Sensordaten jeweils vom Typ `tLoraDataRec` (als `kLoraPacketDataGen0`, `kLoraPacketDataGen1` und `kLoraPacketDataGen2`). Header und Sensordaten umfassen jeweils 64Bit zuzüglich einer 16Bit CRC. Die Gesamtgröße eines LoRa-Paketes hat damit eine Payload-Länge von 40 Bytes (4 * (64Bit + 16Bit) = 4 * 80Bit = 320Bit). Die Headerdatei *<LoraPacket.h>* definiert den Aufbau eines LoRa-Paketes im Detail, das Dokument *<LoRa_Payload_Design.pdf>* verdeutlicht dessen schematischen Aufbau.

Die Kodierung der Sensordaten in das over-the-air Format erfolgt in der Methode `LoraPayloadEncoder:: EncodeTxDataPacket()`.

Sensordaten Mittelwert

Aufgrund der regulatorischen Vorgaben an den Duty Cycle zur Nutzung des 868 MHz-Bandes (max. 1% Kanalbelegung) werden die Sensordatenpakete nur in größeren zeitlichen Intervallen gesendet (typischerweise im Stundentakt). Um dabei auch die Trendentwicklung zwischen den Sendezeitpunkten mit zu berücksichtigen, wird über die Sensordaten (Temperatur, Luftfeuchtigkeit, CarBattLevel) jeweils ein gleitender Mittelwert gebildet. Hierzu wird der Simple Moving Average Filter aus dem Projekt *<SimpleMovingAverage>* genutzt. Der in einem LoRa Datenpaket übertragene Wert ist also nicht der aktuelle Sensorwert zum Sendezeitpunkt, sondern der Mittelwert über die durch `SMA_DHT_SAMPLE_WINDOW_SIZE` und `SMA_CARBATT_SAMPLE_WINDOW_SIZE` definierte Datenreihe des jeweiligen Sensors.

Applikations-Konfiguration Sektion

Am Anfang des Sketches befindet sich folgender Konfigurations-Abschnitt:

```
const int  CFG_ENABLE_OLED_DISPLAY          = 1;
const int  CFG_ENABLE_DHT_SENSOR            = 1;
const int  CFG_ENABLE_SEN_HC_SR501_SENSOR   = 1;
const int  CFG_ENABLE_ADS1115_LIGHT_SENSOR  = 1;
const int  CFG_ENABLE_ADS1115_CAR_BATT_AIN  = 1;
```

Hiermit lässt sich die Laufzeit-Ausführung der zugehörigen Code Abschnitte freischalten (=1) bzw. sperren (=0). Das vermeidet das Auftreten von Laufzeitfehlern bei Boards, auf denen nicht alle Komponenten vorhanden bzw. bestückt sind.

Die aktuelle Geräte-Konfiguration wird im LoRa Bootup Paket übertragen.

LoRa-Konfiguration Sektion

Am Anfang des Sketches befindet sich ein Konfigurations-Abschnitt mit folgenden Parametern zur Definition der LoRa-Sendeeigenschaften:

```
const int   LORA_TX_POWER                 = 20;
const int   LORA_SPREADING_FACTOR         = 12;
const long  LORA_SIGNAL_BANDWIDTH         = 125E3;
const int   LORA_CODING_RATE_DENOMINATOR  = 5;
```

Ein weiterer Parametersatz definiert das Sendezeitregime, das insbesondere zur Wahrung des Duty Cycle für die Nutzung des 868 MHz-Bandes (max. 1% Kanalbelegung) von Bedeutung ist:

`LORA_PACKET_FIRST_TIME`
Definiert die Zeit zwischen dem Bootup Paket und ersten Sensordaten Paket
(bzw. `LORA_PACKET_FIRST_TIME_COMM_MODE` im *"Commissioning Mode"* bei aktivem CFG-Taster)

`LORA_PACKET_CYCLE_TIME`
Definiert die Zeit zwischen zwei aufeinanderfolgenden Sensordaten Paketen (ab dem 2. Paket)
(bzw. `LORA_PACKET_CYCLE_TIME_COMM_MODE` im *"Commissioning Mode"* bei aktivem CFG-Taster)

`LORA_PACKET_INHIBIT_TIME`
Definiert die Mindestzeit zwischen zwei aufeinanderfolgenden Paketen, ist nur bei Aktivierung asynchroner LoRa Sende-Events (DIP1) relevant und gibt die Sperrzeit an, um die ein asynchrones Event-Paket nach dem Senden des letztes zyklischen Sensordaten Paketes zur Einhaltung des Duty Cycles verzögert wird

Um gegenseitige Störung mehrerer Geräte zu minimieren, wird das Sendeintervall zwischen zwei aufeinanderfolgenden Paketen um einen zufälligen Wert im Bereich +/- 5% variiert (`LoraTransmitter::CalcNextTransmitCycleTime()`).

Laufzeitausgaben im seriellen Terminal-Fenster

Zur Laufzeit werden alle relevanten Informationen im seriellen Terminal-Fenster (115200Bd) ausgegeben. Insbesondere während des Systemstarts (Sketch Funktion *setup()*) werden hier auch Fehlermeldungen angezeigt, die ggf. auf eine fehlerhafte Softwarekonfiguration zurückzuführen sind. Diese Meldungen sollten insbesondere während der Erstinbetriebnahme auf jeden Fall beachtet werden.

In der Hauptschleife des Programms (Sketch Funktion *main()*) werden zyklisch die Werte aller Sensoren angezeigt. Darüber hinaus informieren auch hier wieder Meldungen über evtl. Probleme beim Zugriff auf die Sensoren.

Durch aktivieren der Zeile *#define DEBUG* am Anfang des Sketches werden weitere, sehr detaillierte Laufzeitmeldungen angezeigt. Diese sind insbesondere während der Programmentwicklung bzw. zur Fehlersuche sehr hilfreich. Durch auskommentieren der Zeile *#define DEBUG* werden die Ausgaben wieder unterdrückt.

Verwendete Drittanbieter Komponenten

1. **LoRa Radio**

   Für den on-board LoRa-Chipsatz SX1276 des "Heltec ESP32 WiFi LoRa OLED" Moduls wird folgende Treiberbibliothek verwendet:
   https://github.com/sandeepmistry/arduino-LoRa

   Die Installation erfolgt mit dem Library Manager der Arduino IDE.

2. **OLED-Display**

   Für das on-board OLED-Display des "Heltec ESP32 WiFi LoRa OLED" Moduls wird folgende Treiberbibliothek verwendet (U8x8lib):
   https://github.com/olikraus/u8g2

   Die Installation erfolgt mit dem Library Manager der Arduino IDE.

3. **ADC ADS1115**

   Für den ADC vom Typ ADS1115 wird die Treiberbibliothek von Adafruit verwendet:
   https://github.com/adafruit/Adafruit_ADS1X15

   Die Installation erfolgt mit dem Library Manager der Arduino IDE.

4. **DHT Sensor**

   Für den DHT Sensor (Temperatur, Luftfeuchtigkeit) wird die Treiberbibliothek von Adafruit verwendet. Die Installation erfolgt mit dem Library Manager der Arduino IDE.