HTW-Dresden

Allgemeine Informatik

LoRaWAN Wetterstation

Projektseminar
Wintersemester 21/22

Name:	Reitz	Rezaii-Djafari
Vorname:	Lenny	Raphael
s-Nummer:	s80452	s80455

Tag der Einreichung: 1. März 2022

Professor: Prof. Dr.-Ing. Jörg Vogt

Inhaltsverzeichnis

1 Einleitung

Das Projekt wurde im Laufe des Projektseminars 2021 von Lenny Reitz und Raphael Rezaii-Djafari unter der Führung von Professor Jörg Vogt an der HTW Dresden entwickelt.

1.1 Die Übetragungstechnik: LoRa

LoRa (Abkürzung für Long Range) ist eine patentierte und proprietäre drahtlose Modulationstechnik basierend auf Chirp Spread Spectrum. Seit seiner Veröffentlichung in 2015 durch die Semtech Corporation hat es sich für IoT Geräte bewährt, die außerhalb der konventionellen Reichweiten von W-Lan, Bluetooth und ZigBee operieren.

Anders als Bluetooth und W-Lan können nur geringe Datenmengen mit niedriger Bitrate gesendet werden, dafür aber über eine erheblich größere Reichweite bei geringem Energieverbrauch. LoRa funkt standardmäßig auf dem freien Sub-1-GHz Band, weshalb die nutzbaren Frequenzen, Fair-Use-Policies und Vorschriften am Einsatzort beachtet werden müssen. Für eine höhere Datenrate und keine Beschränkungen gibt es auch eine Variante mit 2.4GHz auf die wir nicht weiter eingehen.

1.2 Das Protokoll & die Architektur: LoRaWAN

LoRaWAN wird Open Source von der LoRa Alliance als Systemarchitektur und Netzwerkprotokoll auf der Vermittlungsschicht entwickelt.

1.2.1 Systemarchitektur

Vermaschte Netze werden genutzt um eine große Abdeckung kosteneffizient zu erzielen, allerdings steht der Stromverbrauch durch die Komplexität und das Weiterleiten der Pakete durch die Geräte, ab hier Endknoten genannt, im Widerspruch zur geforderten Energieeffizienz der Endknoten. Da LoRa onehin eine große Reichweite besitzt, wird eine Stern-Topologie verwendet.

Netzwerkknoten stehen dabei nicht für ein einzelnes, sondern für eine beliebig große Anzahl geographisch naher Gateways. Wenn ein Endknoten Daten sendet, passiert dies durch Broadcasts. Alle Gateways, die das Paket erhalten haben, senden es an den Netzwerk Server, der redundante Pakete

herausfiltert, Sicherheitschecks durchführt und schließlich das Paket an seinen Bestimmungsort weiterleitet.

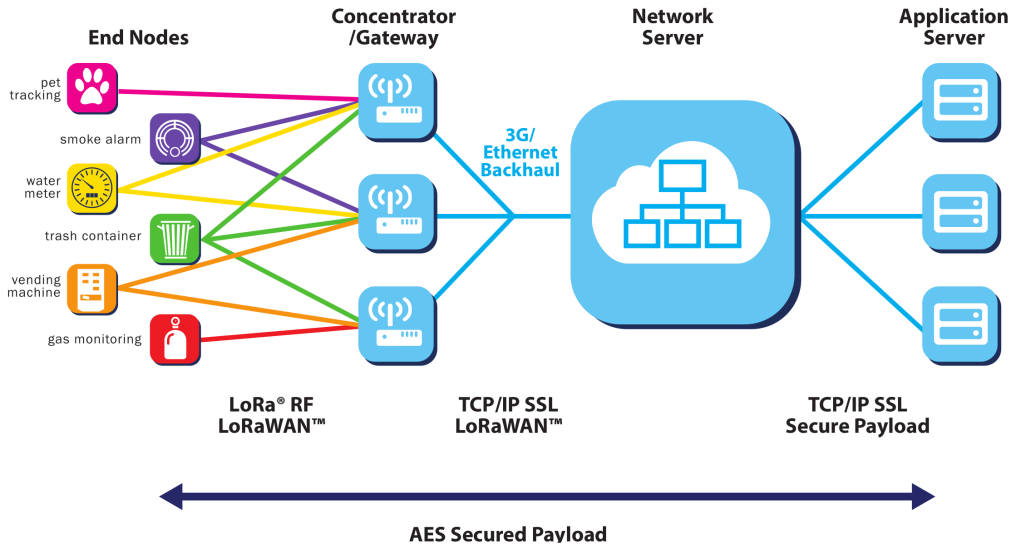

Abbildung 1: LoRaWAN Architektur[1]

Dadurch verlagert sich die Komplexität von den Endknoten und Gateways auf den Netzwerk Server. Außerdem wird die Verfügbarkeit verbessert, wenn ein Gateway ausfällt oder ein Endknoten mobil ist, da kein Handover notwendig ist.[1]

1.2.2 Geräteklassen

Nicht jedes IoT Gerät hat die gleichen Anforderungen. Generell gilt, dass die Kommunikation bidirektional, asynchron und verschlüsselt erfolgt, wobei Endknoten Daten zum Netzwerk senden sobald diese verfügbar sind. Der Unterschied zwischen den Klassen liegt darin, wann ein Endknoten Daten vom Netzwerk empfangen kann.

- **A: batteriebetriebene Sensoren**

  Ein Empffangsfenster wird nur geöffnet nachdem Daten an das Netzwerk übertragen wurden. Das Netzwerk muss Downlink Kommunika-

[1]LoRa Alliance. (2015). A technical overview of LoRa and LoRaWAN [Ebook] (pp. 7-9). https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf

tion für das Gerät solange bereithalten bis dieses Daten sendet. Klasse A ist am energiesparsamsten und muss von allen Geräten unterstützt werden.

- **B: batteriebetriebenes Steuergerät**

  Zusätzlich zu den Eigenschaften der Klasse A, öffnen Geräte der Klasse B das Empfangsfenster zu festgelegten Zeiten. Eine zeitliche Synchronisation mit dem Gateway ist erforderlich.

- **C: Steuergerät mit fester Stromversorgung**

  Das Empfangsfenster der Geräte der Klasse C ist permanent geöffnet. Es wird nur bei Übertragungen geschlossen.

1.2.3 LoRaWAN Netzbetreiber

Aufgrund der Open Source Natur von LoRaWan gibt es die verschiedensten Netzbetreiber. Wir werden uns auf das populäre The Things Network / The Things Stack (TTN) beschränken. Es ist frei nutzbar, allerdings gibt es neben den Frequenzrichtlinien, keine Uptime Verpflichtungen und eine Fair-Use-Policy:

- Die Uplink Airtime (Endknoten → TTN) ist auf 30 Sekunden je 24 Stunden pro Endknoten beschränkt.
- Es dürfen maximal 10 Downlink Nachrichten (TTN → Endknoten) je 24 Stunden pro Endknoten gesendet werden.

Für die nachfolgende Problemstellung eignet sich das TTN optimal. Es sei jedoch erwähnt, dass das kommerzielle The Things Industries (TTI) mit weniger Beschränkungen und SLAs existiert.

1.3 Problemstellung

Wir haben eine analoge Wetterstation und wollen diese an das TTN anbinden. Dafür nutzen wir einen Arduino Uno mit LoRa Shield.

Die Wetterstation muss verbunden und ein Programm geschrieben werden, dass die Daten ausließt und versendet. Die Auflösung und Anzahl der Datensätze, sowie Übertragungsparameter, Authentifizierungsmethoden und Schlüssel müssen einfach konfigurierbar sein.

2 Aufbau/Übersicht

2.1 Hardware

- Wettermessgerät-Kit SEN-15901
  - beinhaltet Regenmesser, Anemometer, Windfahne
- Breadboard
- Jumper-Kabel
- 10 kOhm Widerstand
- 2 Modular-Einbaubuchsen mit Anschlusslitzen, 6P6C
- Arduino Uno
- Dragino Lora Shield v1.4

2.1.1 Aufbau

2.2 Software

2.2.1 Libraries

TODO - LoRa Library

3 Funktionsweise

3.1 TTN

Wie schon in 1.2.3 LoRaWAN Netzbetreiber genannt nutzen wir das The Things Network. Zur Verwaltung der Applications (Verbund von Endknoten) und Gateways gibt es die Console.

In dieser lassen sich Applications und Gateways erstellen, aktuelle Daten anzeigen und Einstellungen konfigurieren. Besonders für Endknoten interessant sind der Payload Formatter, Integrations und ADR.

- **Payload Formatter**

In diesem lassen sich Uplink und Downlink Nachrichten umwandeln. Bei der Übertragung über LoRa sollte die Payload stets so klein wie möglich sein, weshalb die Daten als Bits/Bytes kodiert werden. Siehe dazu auch Working with Bytes. Damit das TTN die Payload definiert in X interpretieren kann müssen wir einen Payload Formatter setzen. Sieh

  - Das Setzen eines Payload Formatter wird in 4.3 beschrieben.

- **Integrations**

  Hier gibt es die Möglichkeit Trigger zu erstellen die Daten verarbeiten und weiterleiten. Beispielsweise Webhooks oder Storage Integrations für Up- und Downlink Nachrichten. Für eine Auflistung siehe Applications & Integrations.

- **Adaptive Data Rate - ADR**

  ADR ist ein Mechanismus für Endknoten der Spreading Factor, Datenübertragungsrate und Sendeleistung optimiert. Dafür werden die letzten 20 Uplink Nachrichten analysiert. Es ist standardmäßig für jeden Endknoten aktiviert, sollte für mobile Endknoten jedoch deaktiviert werden. Siehe 4.4 ADR deaktivieren.

3.2 Authentifizierungsmethoden

Um einen Endknoten im TTN zu Authentifizieren gibt es zwei Möglichkeiten. Beim Anlegen des Endknotens in der TTN Console muss entweder OOTA oder ABP gewählt werden, wobei OOTA vorzuziehen ist.

Einen wichtigen Aspekt spielt dabei im Bezug auf Sicherheit der Frame Counter. Mit jedem gesendeten Paket wird dieser inkrementiert und vom Netzwerk verifiziert um Replay Attacken zu verhindern.

3.2.1 Over-the-Air Activation - OOTA

Bei der bevorzugten Variante OOTA wird ein Join Verfahren beim Einschalten des Endknotens initialisiert, wobei Identifikationsinformationen, inklusive Frame Counter, und Schlüssel ausgetauscht werden. Ebenso werden die verfügbaren Frequenzen und der optimale Spreading Factor ermittelt.

3.2.2 Activation by Personalization - ABP

Bei ABP werden die vordefinierten Identifikationsinformationen, Spreading Factor und Schlüssel auf dem Gerät gespeichert. Falls ADR deaktiviert ist, bleibt der Spreading Factor konstant. Da es kein Join Procedure gibt, muss der Frame Counter, persistent abgespeichert werden. Der Arduino Uno besitzt aber keinen persistenten Speicher. Wenn dieser nun ausgeschaltet wird, resettet sich der Frame Counter auf 0 und beim erneuten Anschalten werden alle Pakete vom TTN verworfen.

Hinweis: Der Arduino Uno besitzt ein EEPROM welchen man nutzen könnte um den Frame Counter zu speichern. Dieser hat jedoch eine begrenzte Lebensdauer von ungefähr 100,000 Schreibzyklen.[2] Der Versuch den Frame Counter im RAM zu behalten und nur beim Ausschalten zu speichern funktioniert nicht, da bei einem Stromausfall die Operation nicht durchgeführt wird. Um das zu umgehen kann mit jedem gesendeten Paket der EEPROM beschrieben werden. Dann ist dieser nach spätestens 100,000 Paketen nicht mehr beschreibbar, was abhängig vom Sendeinterval schnell sein kann.

Daher empfehlen wir unbedingt OTAA. Sollte man dennoch ABP nutzen wollen, so lässt sich in der TTN Console die Frame Counter Überprüfung deaktivieren, wie gezeigt in 4.2 ABP-Konfiguration, wodurch Replay Attacken nicht mehr verhindert werden.

3.3 Sensoren der Wetterstation

3.3.1 Regenmesser

Der Regenmesser besteht aus einem Auffangbehälter und einem darin befindlichen einfachen Kippschalter. Wenn sich genügend Wasser im Behälter gesammelt hat, kippt der Schalter um und das Wasser läuft wieder aus dem Behälter heraus. Ein Kippvorgang entspricht einer Wassermenge von 0,2794 mm. Die folgende Umrechung ergibt die Wassermenge in Milliliter bei einer Auffangfläche von 55 cm^2:

[2] Egger, Alexander. Arduino und der EEPROM. Retrieved 10 February 2022, from https://www.aeq-web.com/arduino-atmega-eeprom-read-write-details/

$$1\,mm = 1\,l/m^2 \quad \text{(Niederschlag)}$$
$$\Rightarrow 0.2794\,mm = 0.2794\,l/m^2 = 0.02794\,ml/cm^2$$
$$\Rightarrow 0.02794\,ml/cm^2 * 55\,cm^2 \approx 1.54\,ml \quad (1)$$

Ein Kippen beträgt also im Idealfall 1,54 ml. Mehrere Testläufe haben aber gezeigt, dass der Regenmesser mit einer relativ großer Ungenauigkeit schaltet. Je nach dem, wie schnell das Wasser einfließt, variiert die Messgenauigkeit stark.

3.3.2 Anemometer

Das Schalenanemometer misst Windgeschwindigkeiten durch Schließen des Kontaktes eines Schalters durch einen Magneten. Eine Windgeschwindigkeit von 2.4 km/h schließt den Schalter einmal pro Sekunde.

Eine Umdrehung entspricht 3 Schaltvorgängen.

3.3.3 Windfahne

Die Windfahne besitzt 8 Schalter, welche jeweils mit unterschiedlich großen Widerständen verbunden sind. Die Schalter könne einzeln oder in Paaren umgelegt werden, wodurch sich 16 mögliche Positionen bestimmen lassen.

Durch einen externen Widerstand wird ein Spannungsteiler implementiert, der eine messbare Spannung ausgibt. Daher lässt sich die Windrichtung nur durch ein analoges Signal bestimmen.

3.4 Beispielablauf

Benötigte Payload Größe (pro Sekunde):

- Windfahne: 4 Bit (16 mögliche Windrichtungen)
- Regenmesser: 2 Bit (maximal 4 Klicks pro Sekunde)
- Anemometer: 7 Bit (maximal 128 Klicks pro Sekunde)

=¿ 13 Bit Payload Größe (pro Sekunde) Wenn wir 60 Messwerte pro Minute aufnehmen würden und dann pro Minute diese Daten an das TTN senden würden, bedeutet das:

- Payload Größe: 13 Bit * 60 = 780 Bit 98 Byte (pro Minute)

Es muss eine Konfiguration für LoRa eingestellt werden, welche optimal für die zu sendende Paketgröße und Sendeinterval abgestimmt ist.

Laut https://www.thethingsnetwork.org/airtime-calculator ist die minimale Airtime für unsere Paketgröße und Sendeinterval:

- Paketgröße: 98 Byte
- Spreading Factor: SF7
- Region: EU868
- Bandbreite: 250 kHz

=¿ Airtime: 94,8 ms

Problem, weil die Fair Use Policy von TTN nur 30s Airtime pro Tag erlaubt. Wir sind allerdings mit 137s pro Tag weit darüber.

=¿ Eine rohe Übermittlung der Daten mit einer Genaugikeit von 1s ist nicht möglich.

Schlussfolgerung: Die Rohdaten müssen gemittelt werden, um eine Übertragung innerhalb der 30s pro Tag zu einzuhalten.

pro 5 Sekunden:

- Regenmesser: 3 Klicks * 5 Sekunden = 15 Klicks =¿ 4 Bits - Anemometer: Wert wird gemittelt =¿ 7 Bits - Windfahne: Wert wird gemittelt =¿ 4 Bits

=¿ Gesamt: 15 Bits pro 5 Sekunden

=¿ Pro Minute: 180 Bits 23 Byte

pro 10 Sekunden:

- Regenmesser: 3 Klicks * 10 Sekunden = 30 Klicks =¿ 5 Bits - Anemometer: Wert wird gemittelt =¿ 7 Bits - Anemometer: Maximalwert =¿ 7 Bits - Windfahne: Wert wird gemittelt =¿ 4 Bits

=¿ Gesamt: 23 Bits pro 10 Sekunden

=¿ Pro Minute: 138 Bits 18 Byte

=¿ pro 5 Minuten: 690 Bits 87 Byte

Das wäre eine mögliche Konfiguration mit einer Airtime von 87.2 ms pro 5 Minuten oder 25.1 s pro Tag

4 Konfiguration & Verbindung der Wetterstation

In dieser Dokumentation wird nicht darauf eingegangen wie ein Gateway eingerichtet wird. Konsultieren Sie dafür die Betriebsanleitung des Herstellers.

Vorbedingung:

1. Installieren Sie die Arduino IDE und die notwendige Library nach `Installation.md`
2. Erstellen Sie einen TTN Account unter www.thethingsnetwork.org
3. Navigieren Sie zur Console
4. Wählen Sie <Go to applications>
5. Fügen Sie durch <Add application> eine Application hinzu
6. Wählen Sie <Add end device> und wechseln in den <Manually> Tab
7. Frequency Plan: Europe 863-870 MHz (SF9 for RX2 - recommended)
8. LoRaWan version: MAC V1.0.2
9. Regional Parameters version: PHY V1.0.2 REV A

Wie bereits in 3.2 Authentifizierungsmethoden erläutert, empfehlen wir OTAA gegenüber ABP.

4.1 OTAA-Konfiguration

10. DevEUI: Generate
11. AppEUI: Fill with zeros
12. AppKey: Generate
13. <Register end device >

14. Wechseln Sie zu `secret.template.h` und geben Sie die generierten Daten ein. Achten Sie ggf. auf das little-endian Format.
15. Benennen Sie `secret.template.h` in `secret.h` um
16. Öffnen Sie `config.h` und aktivieren Sie den OTAA Modus
17. Wählen Sie Ihre gewünschten Datenauflösung und die Größe der Payload. Betrachten Sie dazu das Beispiel, verifizieren Sie die Größe der Payload mit dem LoRaWAN airtime calculator
18. Prüfen Sie, dass die Fair-Use-Policy nicht verletzt wird
19. Beachten Sie, dass bei OTAA der Spreading Factor automatisch gesetzt wird. Wählen Sie am besten Werte die mit SF12 noch Fair-Use sind.
20. Falls gewünscht, können Sie ADR deaktivieren.
21. Installieren Sie das Programm auf dem Arduino Uno.
22. Machen Sie weiter mit Payload Formatter setzen.

4.2 ABP-Konfiguration

10. <Show advanced activation, LoRaWAN class and cluster settings >
11. Activation mode: ABP
12. DevEUI: Generate
13. Device address: Generate
14. AppSKey: Generate
15. NwkSKey: Generate
16. <Register end device >
17. Nachfolgend deaktivieren Sie den Frame Counter, siehe Problematik in 3.2 Authentifizierungsmethoden.
18. Wechseln Sie in den <General settings> Tab
19. Wählen Sie bei Network layer <Expand>
20. Wählen Sie unten <Advanced MAC settings>

21. Setzen Sie das Häkchen in <Resets frame counters>
22. Falls gewünscht, entfernen Sie das Häckchen bei <Use adaptive data rate (ADR)>
23. <Save changes>
24. Wechseln Sie zu `secret.template.h` und geben Sie die generierten Daten ein.
25. Benennen Sie `secret.template.h` in `secret.h` um
26. Öffnen Sie `config.h` und aktivieren Sie den ABP Modus
27. Wählen Sie Ihre gewünschten Datenauflösung und die Größe der Payload. Betrachten Sie dazu das Beispiel, verifizieren Sie die Größe der Payload mit dem LoRaWAN airtime calculator
28. Prüfen Sie, dass die Fair-Use-Policy nicht verletzt wird.
29. Falls gewünscht, können Sie ADR deaktivieren.
30. Installieren Sie das Programm auf dem Arduino Uno.
31. Machen Sie weiter mit Payload Formatter setzen.

4.3 Payload Formatter setzen

1. Wählen Sie den <Payload formatters> Tab
2. Uplink ist standärdmäßig ausgewählt
3. Formatter type: Javascript
4. Kopieren Sie den Javascript Code von `payload_formatter.js` in das vorgesehene Feld
5. <Save changes>
6. Testen Sie den Formatter mit der Payload X
7. Starten Sie den Arduino. Falls ein Gateway in Reichweite ist, sollten die Daten nach einiger Zeit im Tab <Live data> erscheinen

4.4 ADR deaktivieren

1. Wechseln Sie in den <General settings> Tab im TTN
2. Wählen Sie bei Network layer <Expand>
3. Wählen Sie unten <Advanced MAC settings>
4. Entfernen Sie das Häkchen in <Use adaptive data rate (ADR)>
5. <Save changes>

5 Probleme