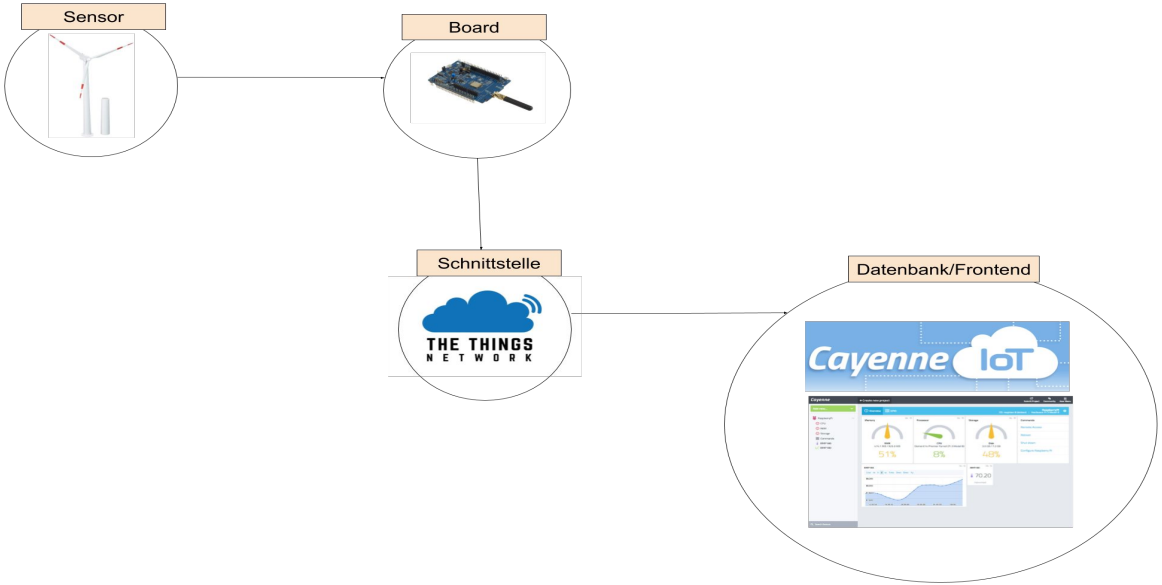
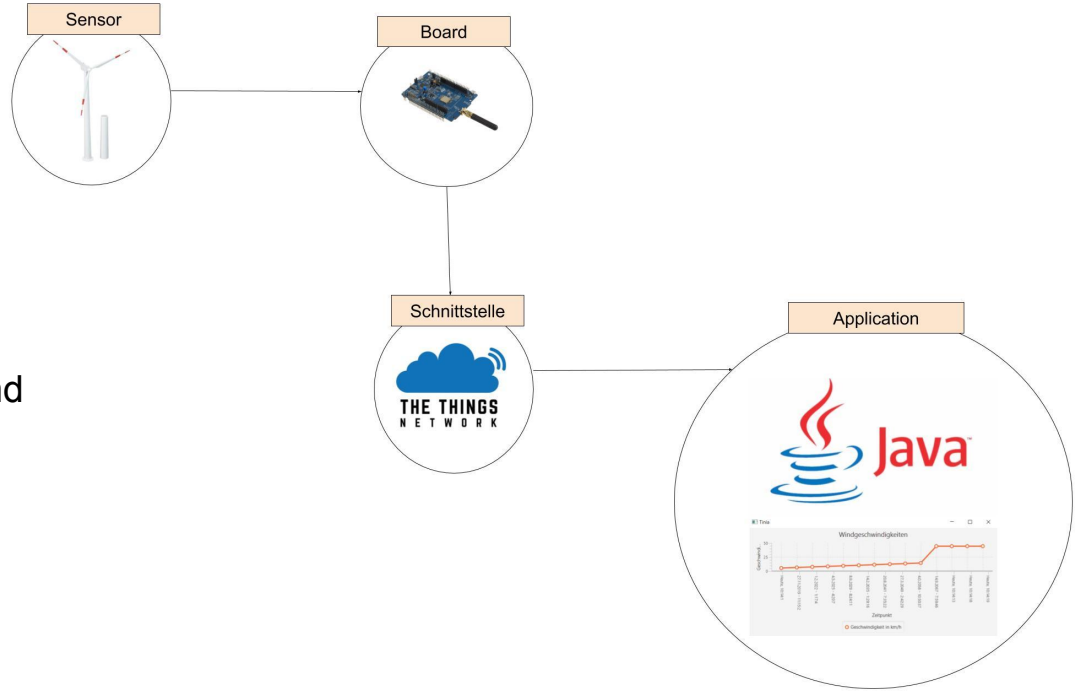
Release Candidate 1

Tinia

Rückblick: Was wollten wir erreichen?



Was hat sich geändert?



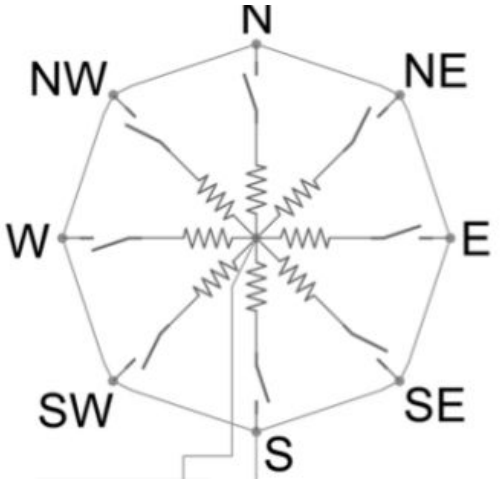
- Sensor
- Frontend

Wetterstation

- Bisheriger Windsensor nur für geringe Geschwindigkeiten ausgelegt/nicht für reale Anwendungen geeignet
- Deswegen -> Wetterstation!

Windrichtungs-Sensor

- Acht interne Widerstände erzeugen in jedem Zustand eine andere Spannung



Windstärke-Sensor

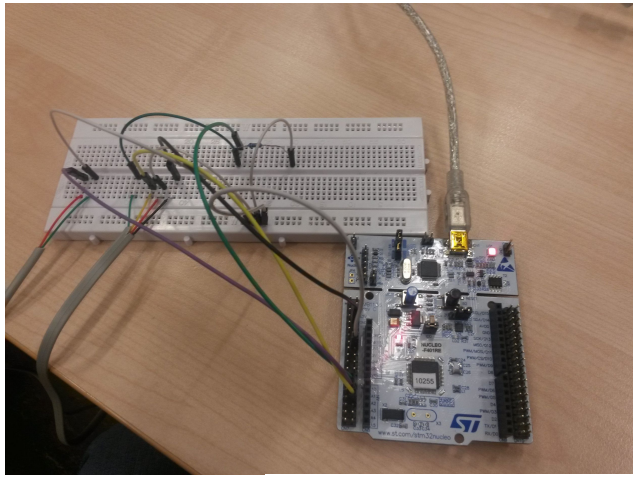
- Bei einer Windgeschwindigkeit von 2.4 km/h wird der Kontakt einmal pro Sekunde geschlossen

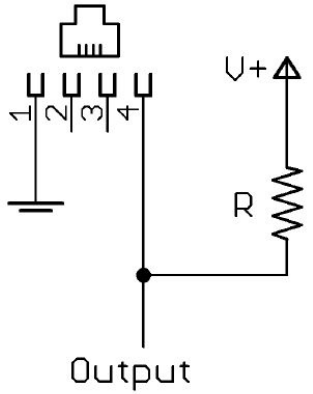
Regenfall-Sensor

- Jede 0.2794 mm Regenfall wird ein Kontakt erzeugt

Anbindung

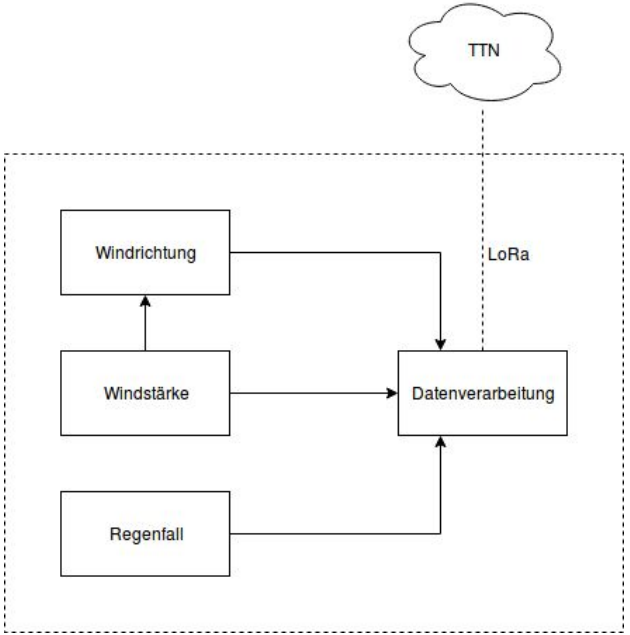
- Windstärke- und Regenfallsensoren werden über IO-Interrupts getrackt
- Windstärke über einen konkreten ADC-Messwert





Board-Software

- Programm wird aufgeteilt in vier Threads
- Drei Threads für jeden einzelnen Sensor
- Ein Thread für die Datenverarbeitung und -übertragung über LoRa
- Windrichtung und Windstärke werden gekoppelt gemessen



Funkverbindung

- LongRangeWideAreaNetwork
- max. 30 Sekunden Uplink
- 10 Downlink Messages

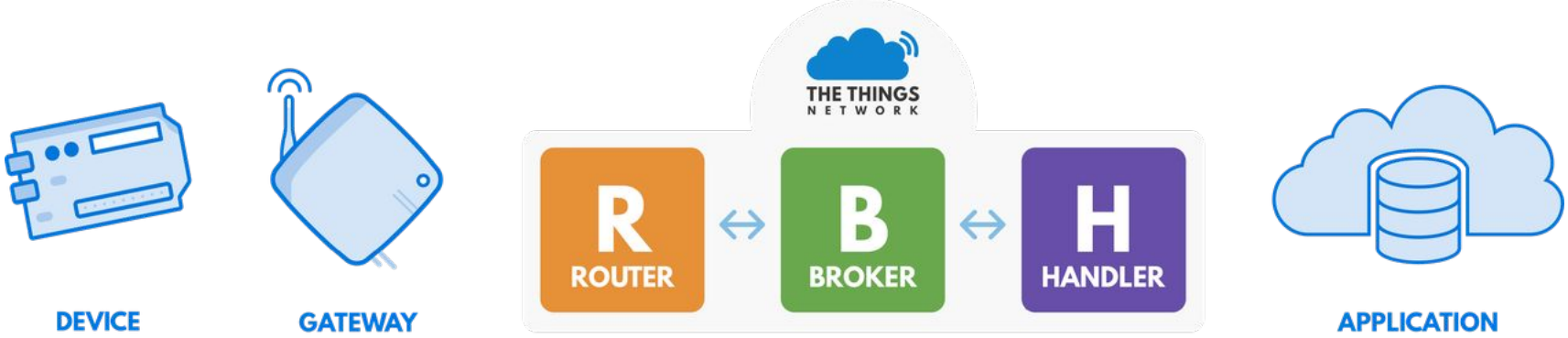
Data-Uplink

- Datentransfer im CayenneLPP (LowPowerPayload) Format
- Mehrere Payloads zu einem Buffer zusammengefasst

1 Byte	1 Byte	N Bytes
Data1 Ch.	Data1 Type	Data1

- 3*4Bytes = 12 Byte Payload
- ~10 Messages pro Stunde (geplant Message alle 10 Minuten)

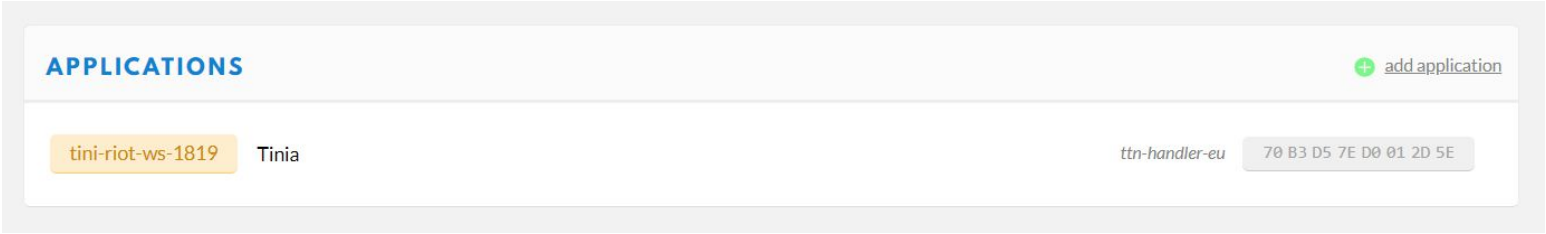
The Things Network



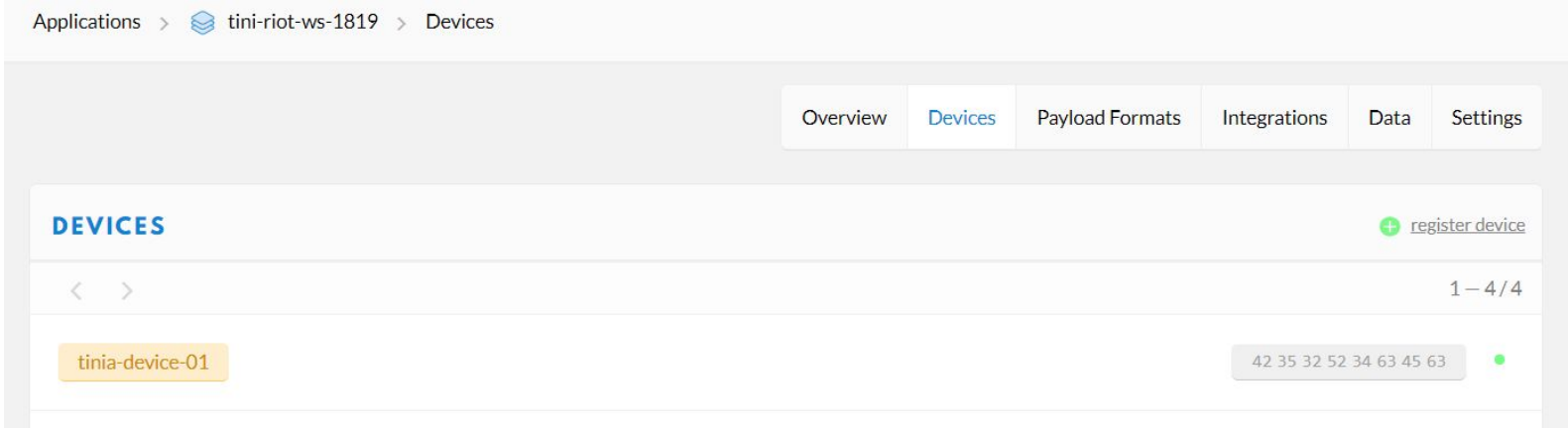
- **Device** - Ein LoRaWAN-fähiges Gerät
- **Gateway** - Empfängt LoRa Nachrichten und leitet diese an einen Router weiter
- **Router** - Ein Microservice, der Nachrichten vom Gateway empfängt und einen Broker findet, an den die Nachricht weitergeleitet wird.
- **Broker** - Ein Microservice, der das Device identifiziert, den Datenverkehr dedupliziert und das Paket an den Handler weiterleitet, bei dem die Application registriert ist.
- **Handler** - Ein Microservice, der den Payload ver- und entschlüsselt und diesen bei Message Brokern der registrierten Applications veröffentlicht
- **Application** - Software für den Enduser

Ein Gerät im TTN registrieren

- Man erzeugt eine Application über die TTN Console
- Diese bekommt mindestens einen AppEUI (Application End-device Unique Identifier) und eine Application ID



- Der Application wird ein Device hinzugefügt
- Dieses besitzt einen DevEUI (Device End-device Unique Identifier)
- Application ID, AppEUI und DevEUI werden zur Identifikation vom Device in der Nachricht mitversendet

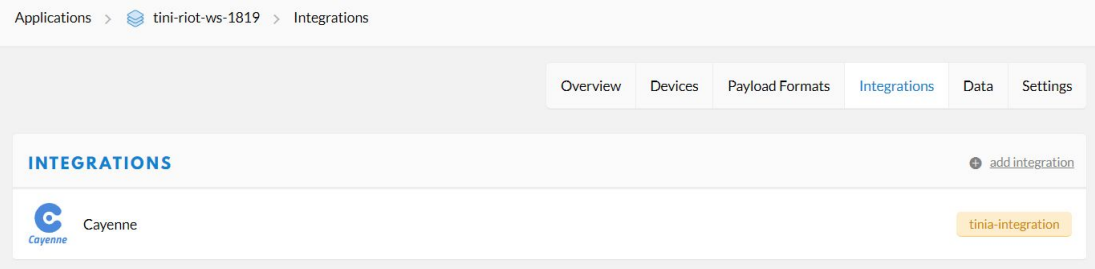


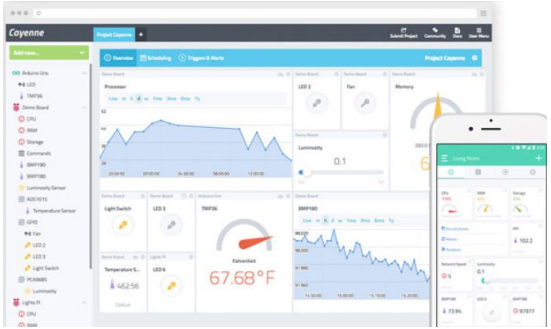
Application

- Unser erster Ansatz: Über eine Integration mit einer Plattform verbinden
  - Cayenne Integration im TTN
  - In der Cayenne Cloud ein Projekt anlegen und über die DevEUI ein Device registrieren
  - Payload muss im LPP (Low Power Payload) Format geschrieben sein

1 Byte	1 Byte	N Bytes
Data1 Ch.	Data1 Type	Data1

  - Cayenne Cloud erkennt Daten und zeigt diese direkt mit geeigneten Diagrammen an
  - Vorteil: Schnelle und einfache Visualisierung der Daten





Warum nicht Cayenne Cloud?

- Kein direkter Zugriff auf die Daten
- Widgets unflexibel
- Trigger konnten zwar Webhooks aufrufen, allerdings keine Werte mitliefern

The Things Network Java SDK

- Unser zweiter Ansatz: The Things Network Java MQTT Client
- Einbinden als Maven Dependency
- Empfangen und Senden von Nachrichten über das TTN
- Zugriff über Application ID und Access Key im TTN
- Vorteile:
  - Flexibler Umgang mit den Daten
  - Flexible Visualisierung der Daten
  - Bessere Auswertung der Daten und Möglichkeit komplexerer Logik
- Nachteile:
  - Mehraufwand
  - Keine Datenbank Integration

Zusammenfassung

- Technologischer Durchstich 
- Schnittstelle zwischen TTN und Board definiert
- Analyse eingehender Daten durch Trigger

Was noch fehlt

- Frontend: Visualisierung aller Sensoren
- Board: Threadprogrammierung
- Frontend: Downlink senden
- Board: Downlink für Konfiguration des Sensors
- Application: Persistente Speicherung der Daten