

# **Dezibot Schwarm Logging & Monitoring**

**Vom Einzelbot-Debugging zur Schwarmplattform**

Projektpräsentation, Februar 2026  
Projektteam Dezibot: Niclas Jost,  
Marius Busalt

Fakultät Informatik und Medien  
HTWK Leipzig

# Gliederung

1. Motivation
2. Architektur
3. Funktionen
4. Ausblick

## Ausgangslage vor diesem Projekt

- Debug-Server zeigte nur die Sensordaten des Webserver-Host Dezibots
- Es gab nur Diagnose eines einzelnen Dezibots
- Keine Kommunikation zu anderen Dezibots implementiert
- Großes Webbundle

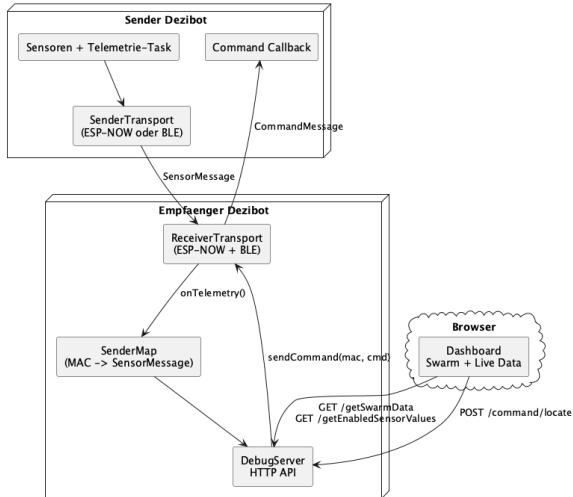
## Ursprüngliche Verbesserungsideen

- Überwachung einer Dezibotgruppe bzw. Schwarm
- Schnelle Übersicht über Schwarm: Wer ist online, wer sendet, Logs einsehen
- Fokus auf Diagnostik, ausgelesene Sensordaten einsehbar und spezifische Logs verfügbar machen
- Identifizierung von Dezibots ermöglichen, um Daten leichter zuordnen zu können

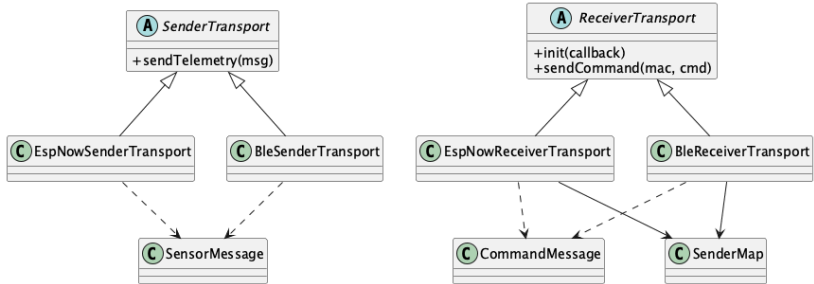
# Projektziele

- Zentrale Schwarm-Übersicht aller verbundenen Dezibots über Masterdezibot
- Telemetrie und Logs einzelner Dezibots verfügbar machen
- Bidirektionale Kommunikation: Masterdezibot soll Telemetrie empfangen und Kommandos zurücksenden
- Kommunikationsprotokoll-Agnostisch
- Echtzeitverfügbarkeit durch entkoppeltes Telemetriemanagement
- Verbesserte UI passend zu neuen Funktionen, Speicherverbrauch reduzieren
- Logging für größere Schwärme ermöglichen

# PlantUML Grobarchitektur (Systemkontext)



# PlantUML Klassendiagramm (Transport-Layer)



## Build- und Flash-Workflow

- Einheitliche Build-Umgebung für Sender und Empfänger, gesamte Config in platform.ini einsehbar
- Frontend-Artefakte können optional in data/ gebaut und per SPIFFS hochgeladen werden.

```
cd web  
npm install  
npm run build
```

```
pio run -e esp32s3_receiver -t uploadfs  
pio run -e esp32s3_receiver -t upload  
pio run -e esp32s3_sender -t upload
```



## Schwarm-Übersicht im Dashboard

- Je Bot sichtbar: Online/Offline, MAC, Message Counter, Uptime, Last Seen
- Klick auf einen Bot führt direkt in die Live-Sensoransicht für diesen Dezibot
- Senden von Commands durch verschiedene Buttons

## Charts mit Live-Sensordaten

- Pro Sensorwert ein eigenes Echtzeit-Diagramm
- Anzeige umfasst Sensordaten und Systemmetriken wie Heapinfo oder Chip-Temperatur
- Anzahl der Datenpunkte ist konfigurierbar

# Logging

- Eigene Logging-Seite mit Filterung nach Level (ALL, INFO, WARNING, ERROR, DEBUG, TRACE)
- Polling-basierter Nachlade-Mechanismus für neue Einträge in nahezu Echtzeit
- Ringpuffer im Receiver-Dezibot verhindert unkontrolliertes Speicherwachstum
- Ermöglicht wichtigen Debugging Einstieg

## Kommunikationsprotokolle

- ESP-NOW und BLE GATT implementiert
- Transport-Adapter Pattern Prozess und Kommunikationslogik getrennt
- -> Flexibilität um verschieden Usecases abzudecken, erweiterbar
- ESP-Now: 200-500m Range, bis zu 1 MBPS
- BLE: 10-50m Range, bis zu 2 MBPS

## ESP-NOW: Telemetrie und Kommandos

- Sender broadcastet `SensorMessage` im Sekundentakt, Empfänger sammelt zentral
- Kommandos gehen vom Dezibot-Receiver an einen spezifischen Dezibot-Sender, z.B. `Locate Dezibot`
- `Magic Numbers` trennen Sensor- und Kommando-Pakete robust

## Bluetooth (BLE GATT)

- Sender als GATT-Server, Empfänger als GATT-Client
- Sensordaten kommen per Notification, Kommandos per Write
- Ermöglicht gemischte Setups mit ESP-NOW und BLE innerhalb derselben Plattform

## Web-Server

- UI basiert auf SolidJS/Vite als Single-Page-Application
- Monitoring nutzerfreundlicher gestalten
- Bundle Größe wurde deutlich reduziert (530kb -> 135kb)
- Receiver-Bot öffnet Soft-AP
- Webserver aus Standardbibliothek
- SPIFFS (SPI Flash File System)

# Demo



## Roadmap

- Sensor-Export als CSV/JSON für Analyse in Python oder Tabellenkalkulation
- Device Naming für besser lesbare Bot-Identifikation anstatt reiner MAC-Adressen
- Erweiterte Remote-Commands für Motoren, LEDs und gruppierte Aktionen
- Alerting bei Schwellwerten, z. B. Heap, Temperatur oder Verbindungsverlust
- Funktionen wie OTA-Update, Fernsteuerung, RSSI-basierte Entfernungsschätzung oder Ladestandanzeige des Dezibots

## Fazit

- Aus einem Einzelbot-Debugging wurde eine zentrale Schwarmüberwachung entwickelt
- Architektur ist so aufgebaut, dass weitere Protokolle und Features anschlussfähig sind
- Mehrwert unserer Erweiterung ist unter anderem das Live-Monitoring anderer Bots, Daten- und Kommandokommunikation sowie Protokollauswahl

Fragen?

## Quellen und Referenzen

- Projekt-Repository und README:  
<https://github.com/niclas-j/dezibot-swarm-logging>
- Dezibot Logging Vorgängerprojekt:  
<https://github.com/Tim-Dietrich/dezibot-logging>
- Dezibot4 Bibliothek: <https://github.com/dezibot/dezibot>
- ESP-NOW Dokumentation (Espressif):  
<https://docs.espressif.com/>