RNLI-Projekt Dokumentation

Übersicht

Diese Dokumentation umfasst zwei Hauptkomponenten des RNLI-Projekts:

1. **GNSS-Sensor** - Ein GPS/GNSS-basiertes System zur Positionserfassung und -protokollierung
2. **RPM-Sensor** - Ein System zur Messung und Protokollierung von Drehzahlen

Beide Komponenten wurden entwickelt, um in maritimen Umgebungen zuverlässig zu funktionieren und wichtige Betriebsdaten zu erfassen.

Teil 1: GNSS-Sensor

1. Systembeschreibung

Der GNSS-Sensor ist ein auf dem ESP32-Mikrocontroller basierendes System, das kontinuierlich GNSS-Daten (Global Navigation Satellite System) erfasst, verarbeitet und auf einer SD-Karte speichert. Das System ist für den Einsatz auf Seenotrettungsfahrzeugen konzipiert, um präzise Positionsdaten zu erfassen.

1.1 Hauptfunktionen:

- Kontinuierliche Erfassung von GNSS-Positionsdaten
- Protokollierung von Koordinaten, Zeit, Datum und Satelliteninformationen
- Speicherung der Daten im CSV-Format auf einer SD-Karte
- Statusanzeige über LEDs

1.2 Aktuelle Version:

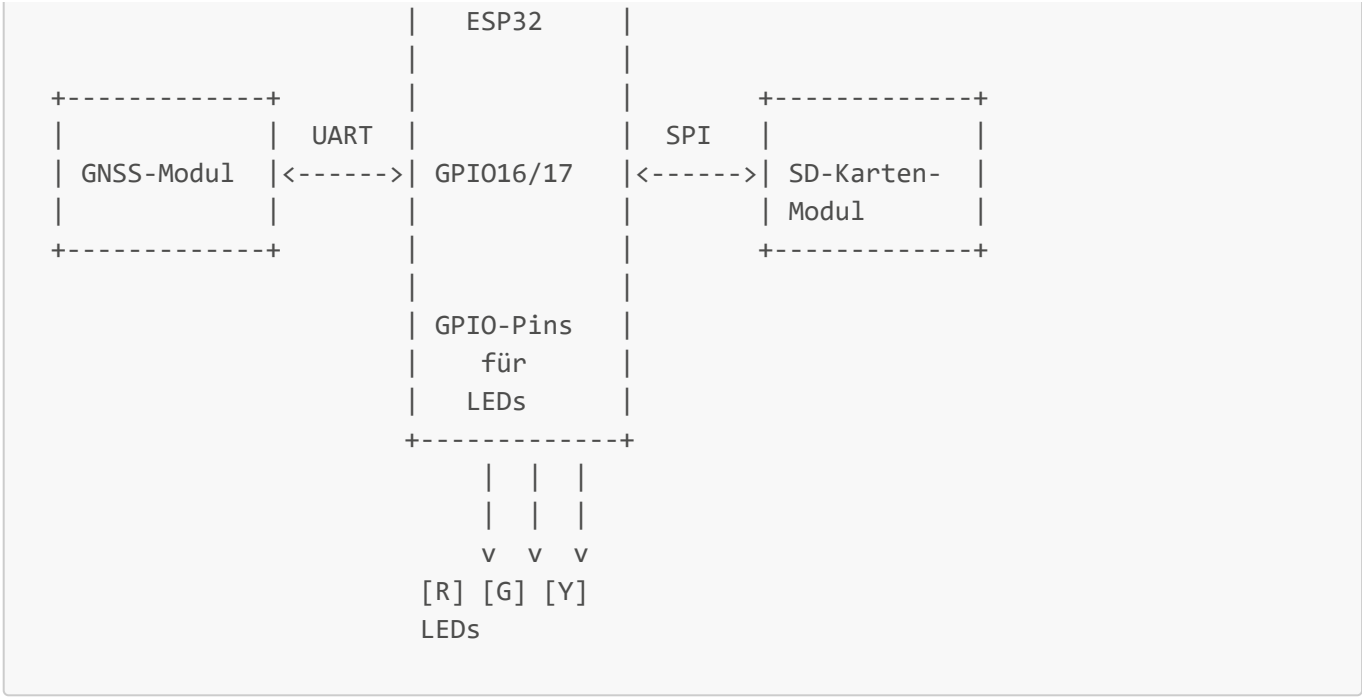
Release 1.2.5 (2025-07-27)

2. Hardware-Komponenten

2.1 Erforderliche Komponenten:

- ESP32 Entwicklungsboard (z.B. ESP32-WROOM-32)
- GNSS/GPS-Modul (mit UART-Schnittstelle)
- SD-Kartenmodul (mit SPI-Schnittstelle)
- Status-LEDs (rot, grün, gelb)
- Stromversorgung (5V)
- Gehäuse mit ausreichendem Schutz gegen Wasser und Erschütterungen

2.2 Schaltplan:



2.3 Pin-Belegung:

Komponente	ESP32 Pin	Funktion
GNSS-Modul RX	GPIO 16	Datenempfang
GNSS-Modul TX	GPIO 17	Datenübertragung
SD-Karte MISO	GPIO 19	SPI MISO
SD-Karte MOSI	GPIO 23	SPI MOSI
SD-Karte SCLK	GPIO 18	SPI SCLK
SD-Karte CS	GPIO 5	Chip Select
Rote LED	gemäß pins.h	Fehleranzeige
Grüne LED	gemäß pins.h	Betriebsanzeige
Gelbe LED	gemäß pins.h	SD-Kartenzugriff

3. Software-Architektur

3.1 Verwendete Bibliotheken:

- Arduino-Framework
- TinyGPS++ für die GNSS-Datenverarbeitung
- SPI-Bibliothek für die SD-Kartenanbindung
- SD-Bibliothek für Dateisystemoperationen

3.2 Hauptkomponenten:

- **main.cpp**: Hauptprogramm mit Setup- und Loop-Funktionen
- **GNSS_module.h/cpp**: Funktionen zur GNSS-Datenverarbeitung

- **SD_module.h/cpp**: Funktionen für SD-Kartenoperationen
- **pins.h**: Definition der Pin-Belegungen

3.3 Datenfluss:

1. Das GNSS-Modul empfängt Satellitensignale und sendet NMEA-Daten über UART an den ESP32
2. TinyGPS++ interpretiert diese Daten und extrahiert Position, Zeit, Datum und Satellitenanzahl
3. Die verarbeiteten Daten werden in einem Puffer zwischengespeichert
4. In regelmäßigen Abständen oder wenn der Puffer voll ist, werden die Daten auf die SD-Karte geschrieben
5. LEDs zeigen den aktuellen Betriebszustand an

4. Installation und Inbetriebnahme

4.1 Hardware-Installation:

1. Verbinden Sie das GNSS-Modul mit den entsprechenden UART-Pins des ESP32
2. Verbinden Sie das SD-Kartenmodul mit den entsprechenden SPI-Pins des ESP32
3. Schließen Sie die LEDs an die definierten GPIO-Pins an
4. Formatieren Sie die SD-Karte im FAT32-Format
5. Stellen Sie eine stabile 5V-Stromversorgung sicher

4.2 Software-Installation:

1. Installieren Sie PlatformIO als Erweiterung in VS Code
2. Öffnen Sie das Projekt in PlatformIO
3. Installieren Sie die erforderlichen Bibliotheken über die Abhängigkeiten in platformio.ini
4. Kompilieren Sie den Code und laden Sie ihn auf den ESP32 hoch

4.3 Konfiguration:

- Passen Sie bei Bedarf die Pin-Definitionen in `pins.h` an
- Überprüfen Sie die Buffer-Größe und das Schreibintervall in der Hauptdatei

5. Bedienung und LED-Statusanzeigen

5.1 Normaler Betrieb:

- Nach dem Einschalten durchläuft das System die Initialisierungsroutine
- Die grüne LED leuchtet, wenn das System nach einer Position sucht
- Die grüne LED blinkt, wenn eine Position gefunden und auf die SD-Karte geschrieben wurde

5.2 LED-Indikatoren:

- Rotes Licht (hell): Setup nach dem Einschalten
- Grün ein: Suche nach Position
- Grün blinkend: Position auf SD-Karte geschrieben
- Grün blinkend und Rot ein: Position berechnet, aber Schreibfehler auf SD-Karte aufgetreten
- Rot ein (gedimmt): Keine SD-Karte gefunden

6. Datenformat und -verarbeitung

6.1 CSV-Dateiformat:

Die Daten werden in CSV-Dateien mit dem Namen `backup_YYYYMMDD.csv` gespeichert. Jede Zeile enthält:

- Datum (JJJJMMTT)
- Zeit (HH:MM:SS)
- Breitengrad (Dezimalgrad, 6 Nachkommastellen)
- Längengrad (Dezimalgrad, 6 Nachkommastellen)
- Anzahl der sichtbaren Satelliten

6.2 Datenpufferung:

- Puffergröße: Definiert durch `BUFFER_SIZE` (Standard: 15 Einträge)
- Schreibintervall: Definiert durch `WRITE_INTERVAL` (Standard: 15000ms)
- Der Puffer wird auf die SD-Karte geschrieben, wenn entweder:
  - Der Puffer voll ist
  - Das Schreibintervall seit dem letzten Schreiben verstrichen ist

7. Fehlerbehebung

7.1 Häufige Probleme:

1. **Keine GNSS-Daten empfangen**:
   - Überprüfen Sie die Verbindungen zum GNSS-Modul
   - Stellen Sie sicher, dass das GNSS-Modul freie Sicht zum Himmel hat
   - Überprüfen Sie die Baudrate in der Software und am GNSS-Modul
2. **Fehler beim Schreiben auf die SD-Karte**:
   - Überprüfen Sie, ob die SD-Karte richtig eingesteckt ist
   - Formatieren Sie die SD-Karte im FAT32-Format
   - Versuchen Sie eine andere SD-Karte
   - Überprüfen Sie die SPI-Verbindungen
3. **LEDs zeigen Fehlerzustand**:
   - Rote LED an: Überprüfen Sie die SD-Karte und deren Verbindungen
   - Keine LEDs leuchten: Überprüfen Sie die Stromversorgung

7.2 Diagnosemöglichkeiten:

- Verbinden Sie den ESP32 über USB und nutzen Sie den seriellen Monitor mit 115200 Baud
- Aktivieren Sie zusätzliche Debug-Ausgaben im Code, falls erforderlich

---

Teil 2: RPM-Sensor

---

1. Systembeschreibung

Der RPM-Sensor ist ein auf dem ESP32-Mikrocontroller basierendes System zur Messung von Drehzahlen (Umdrehungen pro Minute) mittels eines Hall-Sensors. Die gemessenen Daten werden auf einem OLED-Display angezeigt und sowohl im EEPROM als auch auf einer SD-Karte gespeichert, wobei ein RTC-Modul für die Zeitstempelung verwendet wird.

1.1 Hauptfunktionen:

- Messung der Drehzahl mittels Hall-Sensor
- Anzeige der aktuellen Drehzahl auf OLED-Display(s)
- Protokollierung von Drehzahl, Zeit, Datum und Temperatur
- Speicherung im EEPROM und auf SD-Karte
- Einstellung von Datum und Uhrzeit über Tasten

1.2 Aktuelle Version:

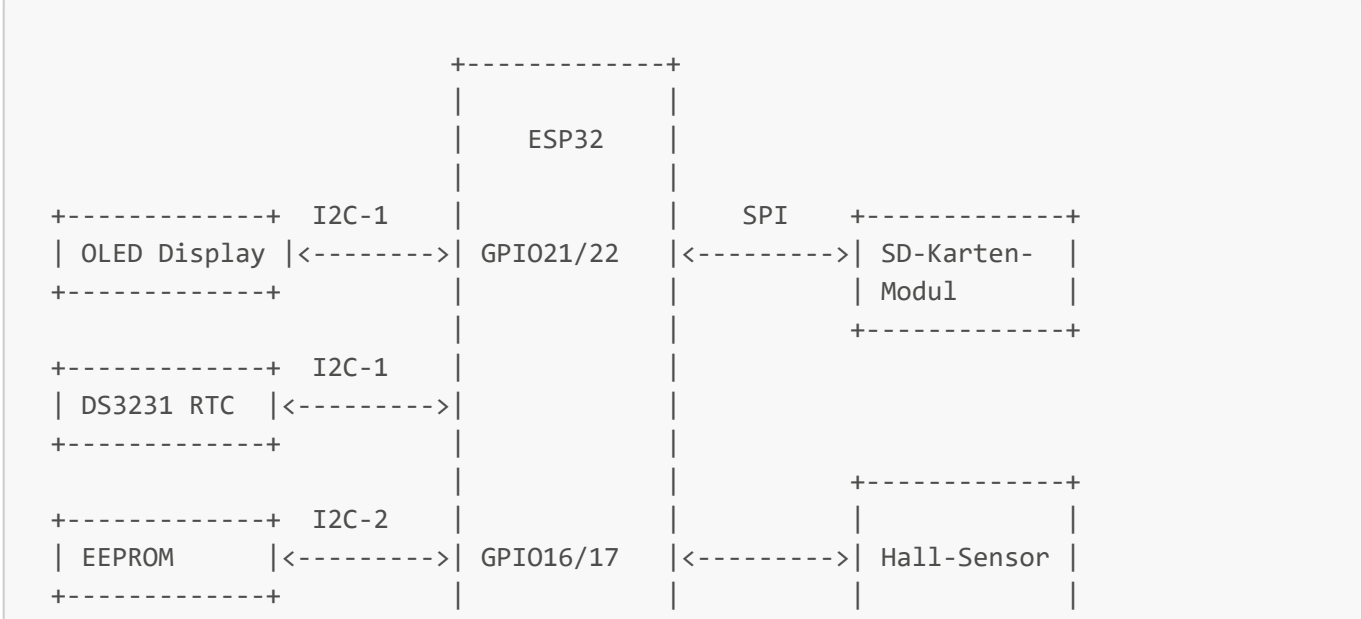
Release 2.3 (2025-07-18)

2. Hardware-Komponenten

2.1 Erforderliche Komponenten:

- ESP32 Entwicklungsboard
- DS3231 RTC-Modul (I2C)
- OLED-Display(s) (I2C)
- Hall-Sensor für die Drehzahlmessung
- SD-Kartenmodul (SPI)
- 24AA512-MIC EEPROM (I2C)
- 3 Bedientasten
- Status-LED
- Stromversorgung (5V)

2.2 Schaltplan:



```
                          |             |          +-------------+
                          | GPIO-Pins   |
                          |    für      |
                          |  Tasten &   |
                          |    LED      |
                          +-------------+
                               | | | |
                               v v v v
                           [+] [-] [S] [LED]
                           Tasten & LED
```

2.3 Pin-Belegung:

Komponente	ESP32 Pin	Funktion
OLED/RTC SDA	GPIO 21	I2C-1 Datenleitung
OLED/RTC SCL	GPIO 22	I2C-1 Taktleitung
EEPROM SDA	GPIO 16	I2C-2 Datenleitung
EEPROM SCL	GPIO 17	I2C-2 Taktleitung
SD-Karte CS	GPIO 5	Chip Select
Hall-Sensor	GPIO 15	Signaleingang
PLUS-Taste	GPIO 25	Werterhöhung (blau)
MINUS-Taste	GPIO 26	Wertverringerung (weiß)
SET-Taste	GPIO 27	Bestätigung (gelb)
Status-LED	GPIO 12	Betriebsanzeige

3. Software-Architektur

3.1 Verwendete Bibliotheken:

- Arduino-Framework
- U8g2lib für OLED-Display
- RTClib für DS3231 RTC
- SPI-Bibliothek für SD-Karte
- Wire-Bibliothek für I2C-Kommunikation

3.2 Hauptkomponenten:

- **RpmSensorRtcOled.cpp**: Hauptprogrammdatei mit allen Funktionen
- Funktionen für:
  - Drehzahlmessung und -berechnung
  - OLED-Display-Anzeige
  - RTC-Zeitabfrage und -einstellung
  - EEPROM-Datenspeicherung

- SD-Karten-Datenspeicherung

3.3 Datenfluss:

1. Der Hall-Sensor erkennt Magnetimpulse und sendet Signale an den ESP32
2. Der ESP32 berechnet die Drehzahl basierend auf der Frequenz der Impulse (4 pro Umdrehung)
3. Die aktuelle Zeit und das Datum werden vom RTC-Modul gelesen
4. Drehzahl, Zeit, Datum und Temperatur werden auf dem OLED-Display angezeigt
5. Die Daten werden regelmäßig im EEPROM gespeichert
6. Die Daten werden in einer CSV-Datei auf der SD-Karte protokolliert

4. Installation und Inbetriebnahme

4.1 Hardware-Installation:

1. Verbinden Sie alle Komponenten gemäß dem Schaltplan und der Pin-Belegung
2. Montieren Sie den Hall-Sensor so, dass er die Magnete am rotierenden Teil erkennen kann
3. Formatieren Sie die SD-Karte im FAT32-Format (max. 16 GB, vorzugsweise 8 GB)
4. Stellen Sie eine stabile 5V-Stromversorgung sicher

4.2 Software-Installation:

1. Installieren Sie PlatformIO als Erweiterung in VS Code
2. Öffnen Sie das Projekt in PlatformIO
3. Installieren Sie die erforderlichen Bibliotheken über die Abhängigkeiten in platformio.ini
4. Kompilieren Sie den Code und laden Sie ihn auf den ESP32 hoch

4.3 Ersteinrichtung:

1. Nach dem ersten Start müssen Datum und Uhrzeit eingestellt werden:
   - Drücken Sie die SET-Taste, um den Einstellungsmodus zu aktivieren
   - Verwenden Sie die PLUS- und MINUS-Tasten, um Werte zu ändern
   - Drücken Sie erneut SET, um zum nächsten Wert zu wechseln
   - Nach dem letzten Wert wird die Einstellung gespeichert

5. Bedienung

5.1 Normaler Betrieb:

- Nach dem Einschalten zeigt das Display die aktuelle Drehzahl, Zeit, Datum und Temperatur an
- Die gemessenen Daten werden automatisch im EEPROM und auf der SD-Karte gespeichert

5.2 Einstellung von Zeit und Datum:

1. SET-Taste drücken, um in den Einstellungsmodus zu gelangen
2. PLUS-/MINUS-Tasten zum Ändern des aktuellen Werts verwenden
3. SET-Taste zur Bestätigung und zum Wechsel zum nächsten Wert drücken
4. Nach dem letzten Wert werden die Einstellungen gespeichert

6. Datenformat und -verarbeitung

6.1 CSV-Dateiformat:

Die Daten werden in CSV-Dateien mit folgendem Format gespeichert:

- Datum (JJJJ-MM-TT)
- Uhrzeit (HH:MM:SS)
- Drehzahl (RPM)
- Temperatur (°C)

6.2 EEPROM-Speicherung:

- Die Daten werden im EEPROM in einem effizienten Format gespeichert
- Bei Stromausfall bleiben die letzten Messwerte erhalten
- Das EEPROM bietet eine zuverlässige Sicherung auch bei Problemen mit der SD-Karte

7. Fehlerbehebung

7.1 Häufige Probleme:

1. **Unregelmäßige oder falsche Drehzahlmessung**:
    - Überprüfen Sie die Position und den Abstand des Hall-Sensors zu den Magneten
    - Stellen Sie sicher, dass die Magnete gleichmäßig angeordnet sind
    - Überprüfen Sie die Verkabelung des Sensors
2. **Probleme mit der SD-Karte**:
    - Verwenden Sie nur SD-Karten mit max. 16 GB (bevorzugt 8 GB)
    - Formatieren Sie die SD-Karte im FAT32-Format
    - Überprüfen Sie die SPI-Verbindungen
3. **Falsche Zeit- oder Datumsanzeige**:
    - Stellen Sie die RTC neu ein
    - Überprüfen Sie die Backup-Batterie der RTC
    - Überprüfen Sie die I2C-Verbindungen zur RTC

7.2 Diagnosemöglichkeiten:

- Verbinden Sie den ESP32 über USB und nutzen Sie den seriellen Monitor mit 115200 Baud
- Überprüfen Sie die Sensorsignale mit einem Oszilloskop
- Aktivieren Sie zusätzliche Debug-Ausgaben im Code, falls erforderlich

---

Anhang A: Versionshistorie

GNSS-Sensor:

- **Release 1.2.0**: Basisversion mit GNSS-Funktionalität und SD-Kartenspeicherung
- **Release 1.2.1** (2025-03-09): LED-Funktionen hinzugefügt, Codeoptimierung
- **Release 1.2.2** (2025-03-18): Zeitstempel- und Datumsstempelfunktionen hinzugefügt

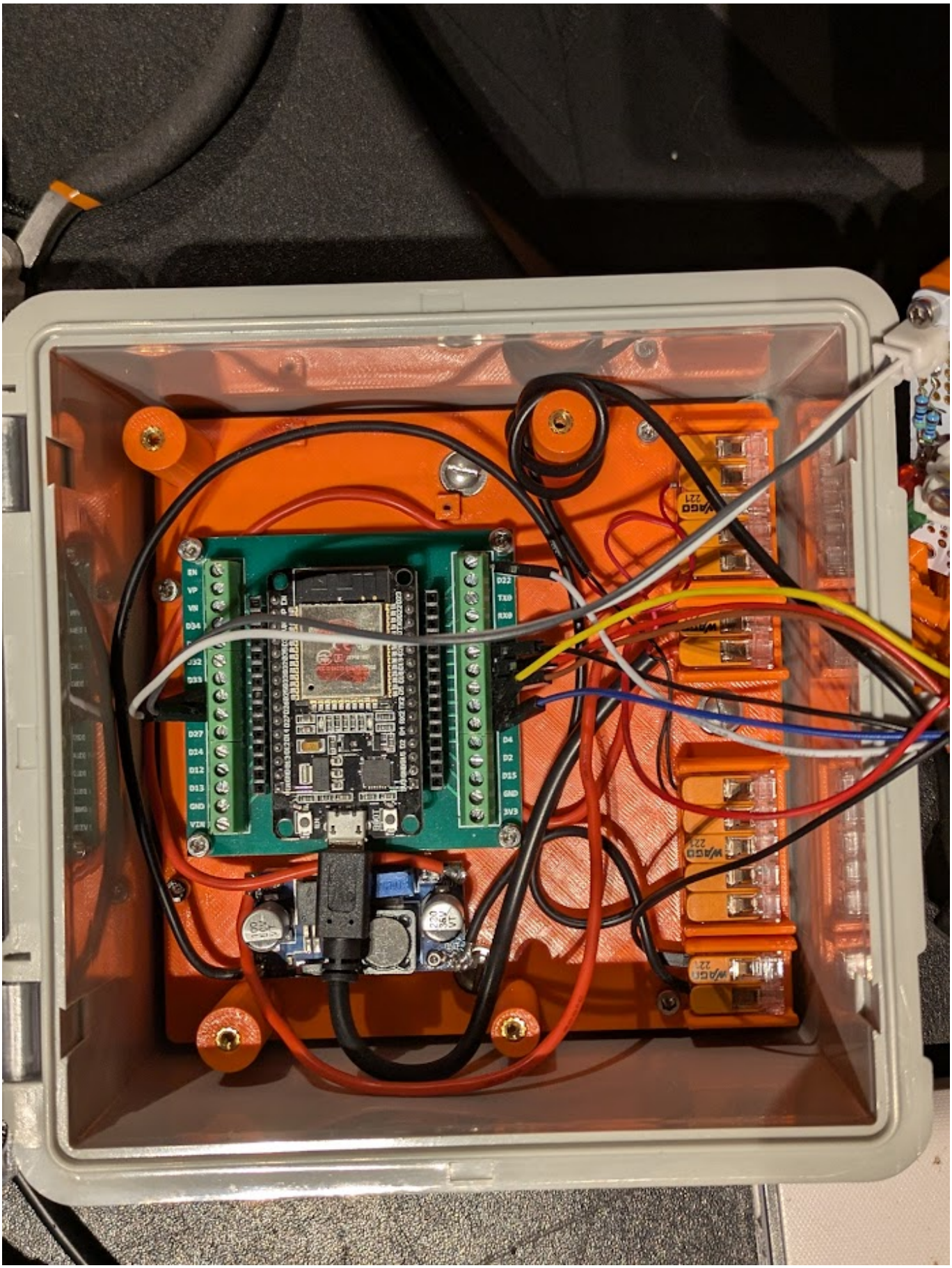
- **Release 1.2.3** (2025-03-19): CSV-Backup-Funktion hinzugefügt
- **Release 1.2.4** (2025-03-20): Verbesserte Formatierung für CSV, Dateinamenerstellung und SD-Karteninitialisierung
- **Release 1.2.5** (2025-07-27): Nicht verwendete Funktionen entfernt, Codebereinigung

RPM-Sensor:

- **Release 2.0** (2025-04-22): Umstellung von TFT auf OLED-Display
- **Release 2.1** (2025-04-23): Verbesserungen der OLED-Anzeige
- **Release 2.2** (2025-04-24): Doppelte OLED-Displays, verbesserte RPM-Berechnung, mit Oszilloskop überprüfte Ergebnisse
- **Release 2.3** (2025-07-18): EEPROM hinzugefügt, Datenprotokollierung auf EEPROM, SD-Karte und Display

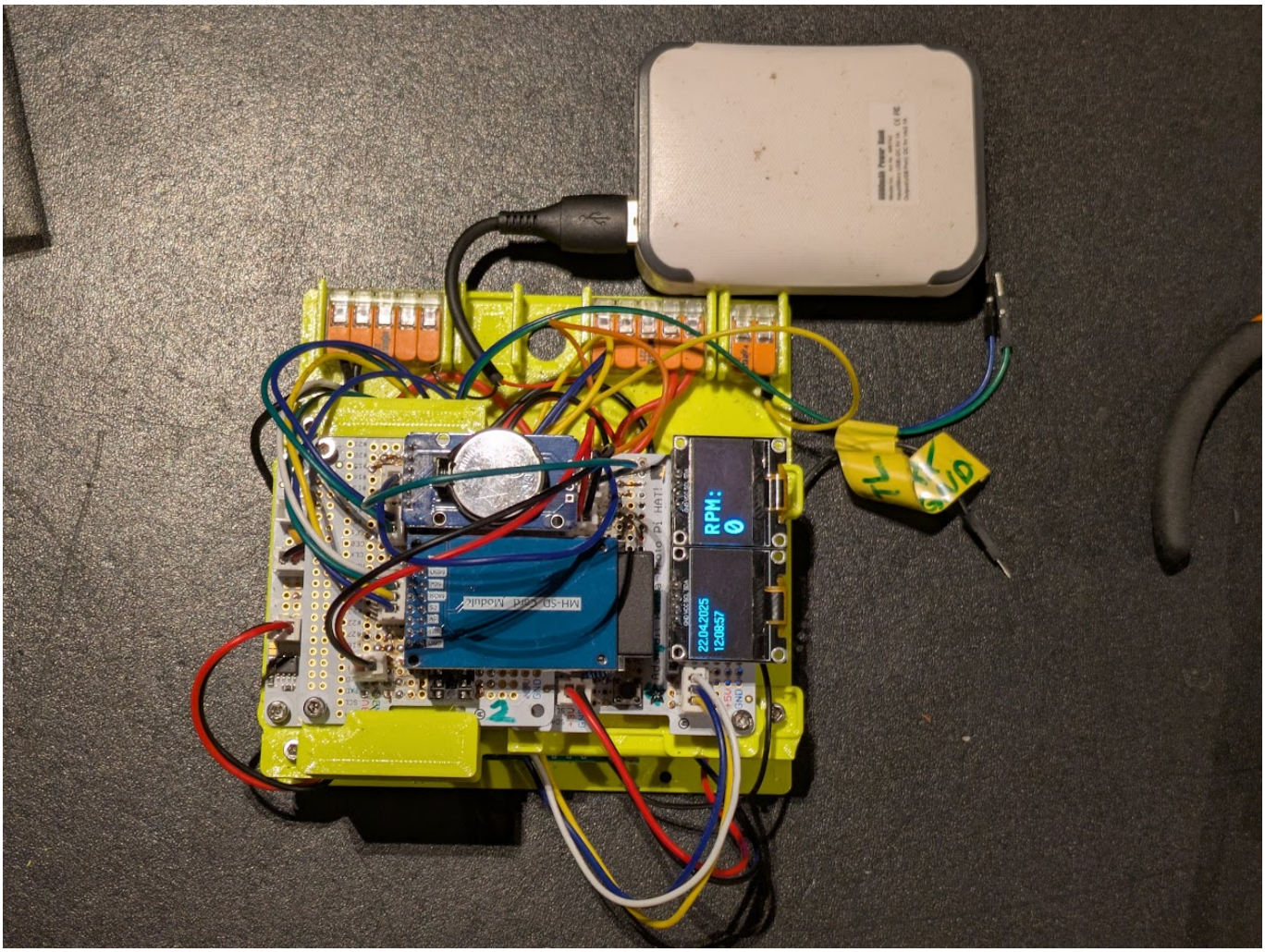
Anhang B: Bildmaterial

GNSS-Sensor

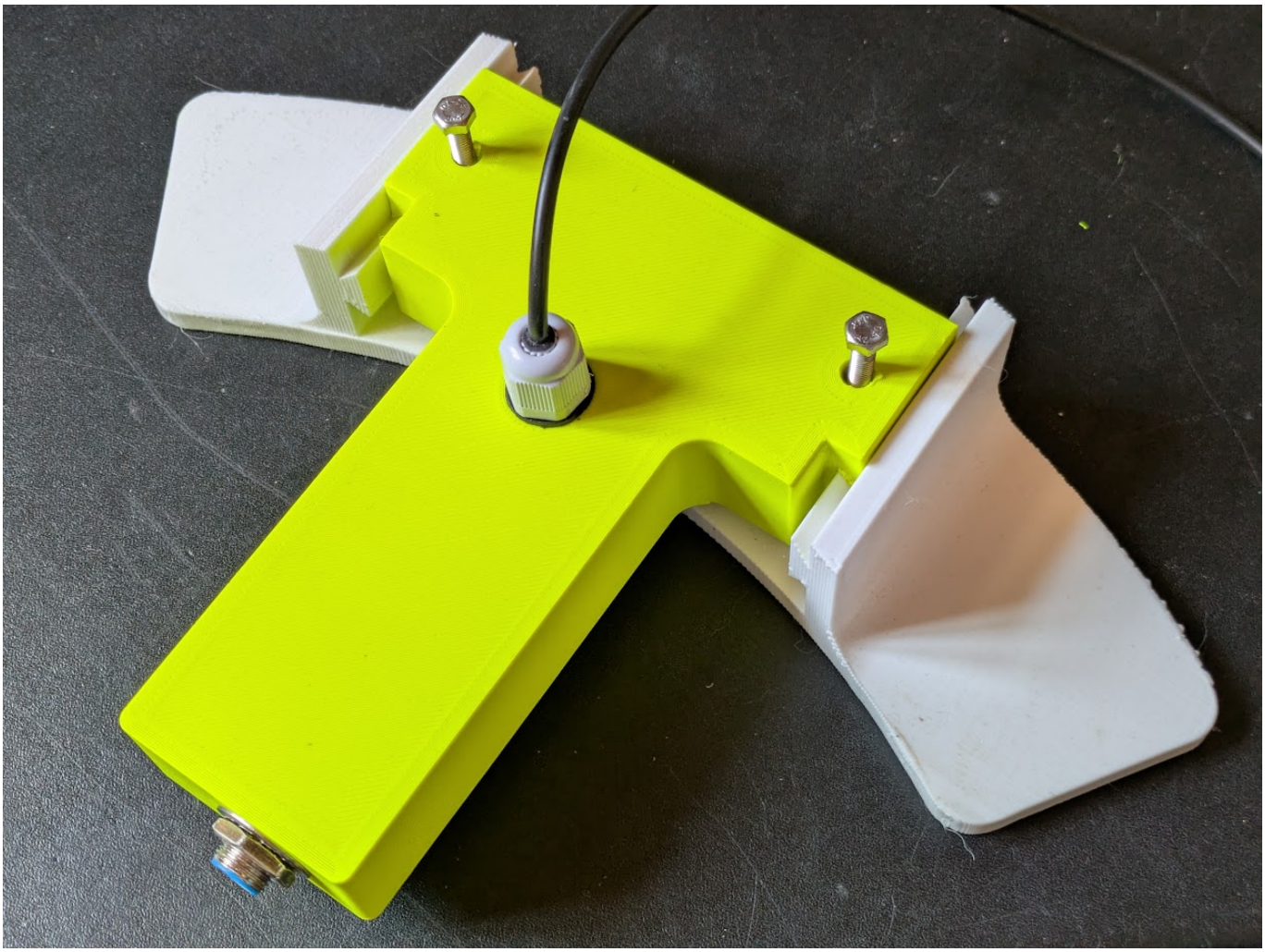


GNSS-Sensor Grundplatine mit Hauptkomponenten

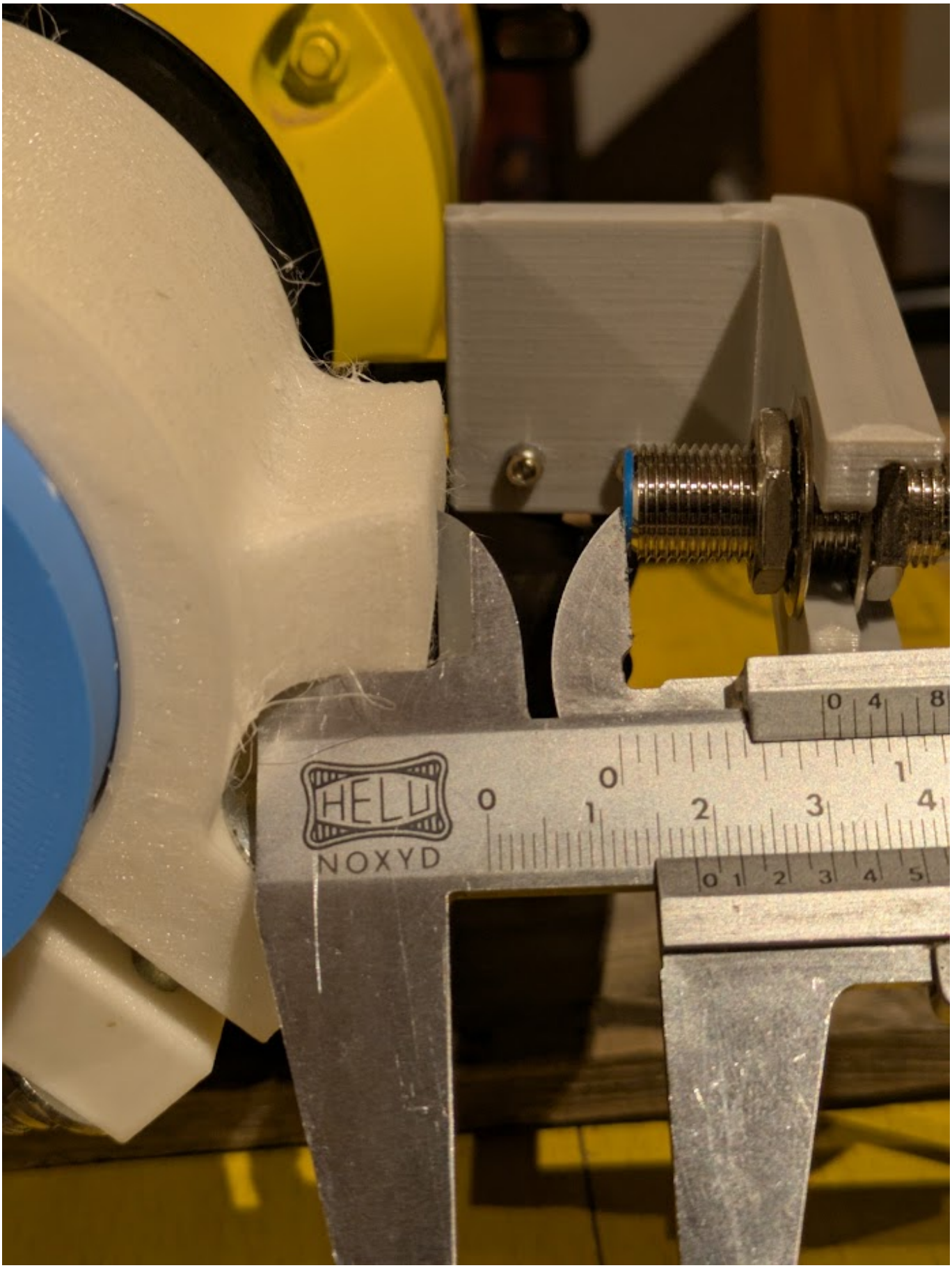
RPM-Sensor



RPM-Sensor im Betrieb



RPM-Sensorbox mit Befestigungshalterung



Hall-Sensor Montage

Anhang C: Datenblätter und Referenzen

- ESP32-WROOM-32 Datenblatt

- DS3231 RTC Datenblatt
- 24AA512-MIC EEPROM Datenblatt (siehe `RpmSensor/EPROM data sheet 24AA512-MIC.pdf`)
- TinyGPS++ Bibliotheksdokumentation
- U8g2lib Bibliotheksdokumentation
- RTClib Bibliotheksdokumentation

Anhang D: Quellcode und 3D-Druckvorlagen

Der vollständige Quellcode und alle zugehörigen Dateien für die RNLI-Projekte sind in den folgenden GitHub-Repositories verfügbar:

GNSS-Sensor

Repository: github.com/hansratzinger/GnssSensor

RPM-Sensor

Repository: github.com/hansratzinger/RpmSensor

Beide Repositories enthalten:

1. Den vollständigen Quellcode des jeweiligen Projekts
2. Konfigurationsdateien und Bibliotheken
3. 3D-Druckvorlagen (.3mf Dateien) im Ordner `/3d` für alle benötigten Komponenten:
    - Gehäuseteile
    - Halterungen
    - TPU-Magnet-Manschetten (für RPM-Sensor)
    - Spritzwassergeschützte Sensorboxen

Die 3D-Druckdateien können mit gängiger 3D-Druck-Software geöffnet und bei Bedarf angepasst werden.