

# Smart Drone Hangar System

Assignment #02 - Embedded Systems & IoT

Academic Year 2025-2026

**Justin Carideo**

ID: 0001115610

31 gennaio 2026

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Componenti usate . . . . .	3
<b>2</b>	<b>Architettura di sistema</b>	<b>3</b>
2.1	Software Design Pattern . . . . .	3
2.2	Task trattate . . . . .	3
<b>3</b>	<b>Finite State Machines</b>	<b>4</b>
<b>4</b>	<b>Implementazione Hardware</b>	<b>10</b>
4.1	Schematic / Wiring . . . . .	10
4.2	Configurazione Pin . . . . .	10
<b>5</b>	<b>Drone Remote Unit (Java)</b>	<b>11</b>
5.1	GUI . . . . .	11
5.2	Protocollo di comunicazione seriale tra Arduino e PC . . . . .	11

# 1 Introduzione

## 1.1 Overview

L'obiettivo è quello di creare un sistema embedded che rappresenta un "Smart Drone Hangar". Il sistema è composto da due sottosistemi:

1. **Drone Hangar (Arduino):** l'hangar intelligente che gestisce l'apertura/chiusura della porta, monitora la presenza del drone, la distanza e la temperatura interna, e visualizza lo stato su un display LCD.
2. **Drone Remote Unit (PC):** un'unità remota che consente di monitorare e controllare l'hangar tramite una GUI Java. Esegue due operazioni in particolare: Takeoff(apertura) e Land(chiusura).

## 1.2 Componenti usate

Il sistema utilizza le seguenti componenti hardware:

- **Sensors:** HC-SR04 (Sonar), PIR, TMP36 (Temperature Sensor), Button
- **Actuators:** Servo Motor, LEDs (Red, Green), I2C LCD Display

# 2 Architettura di sistema

## 2.1 Software Design Pattern

Il software si basa su una architettura **task-based**, nella quale uno scheduler cooperativo gestisce le principali task del sistema. In particolare possiamo trovare i seguenti layer che compongono il sistema:

- **HWPlatform:** incapsula tutte le componenti hardware, fornendo un livello di astrazione che gestisce interazioni a basso livello.
- **Context:** una struttura dati condivisa tra le varie task del sistema.
- **Task Logic:** Il comportamento del sistema è distribuito su diverse task specializzate pianificate dallo Scheduler.

## 2.2 Task trattate

La FSM e le funzioni ausiliarie sono divise in task specifiche:

**HangarTask** Adotta il pattern **Multi-state task**. Incorpora gli stati principali e coordina le altre task aggiornando il **Context** condiviso.

**DoorTask** Responsabile della gestione degli stati della porta dell'hangar (Servo Motore) in base ai comandi presenti nel Context.

**AlarmTask** Monitora continuamente i limiti di temperatura e gestisce gli stati di pre-allarme e allarme, sovrascrivendo le operazioni normali quando necessario.

**DistanceDetectorTask** Gestisce il Rilevatore di Distanza del Drone (DDD/Sonar). Si occupa dell'attivazione (triggering) e del filtraggio delle letture ultrasoniche per aggiornare la distanza nel Context.

**PresenceDetectorTask** Gestisce il Rilevatore di Presenza del Drone (DPD/PIR). Si occupa della calibrazione e della sincronizzazione del rilevamento di presenza.

**DisplayTask** Gestisce il monitor LCD I2C. Legge lo stato del sistema dal Context e aggiorna i messaggi visualizzati in modo asincrono.

**SerialCommTask** Gestisce la comunicazione bidirezionale con la Drone Remote Unit (DRU) su PC, effettuando il parsing dei comandi in ingresso e inviando aggiornamenti di stato.

**TestHWTTask** Una task dedicata utilizzata durante la fase di sviluppo per verificare e calibrare le singole componenti hardware in isolamento.

### 3 Finite State Machines

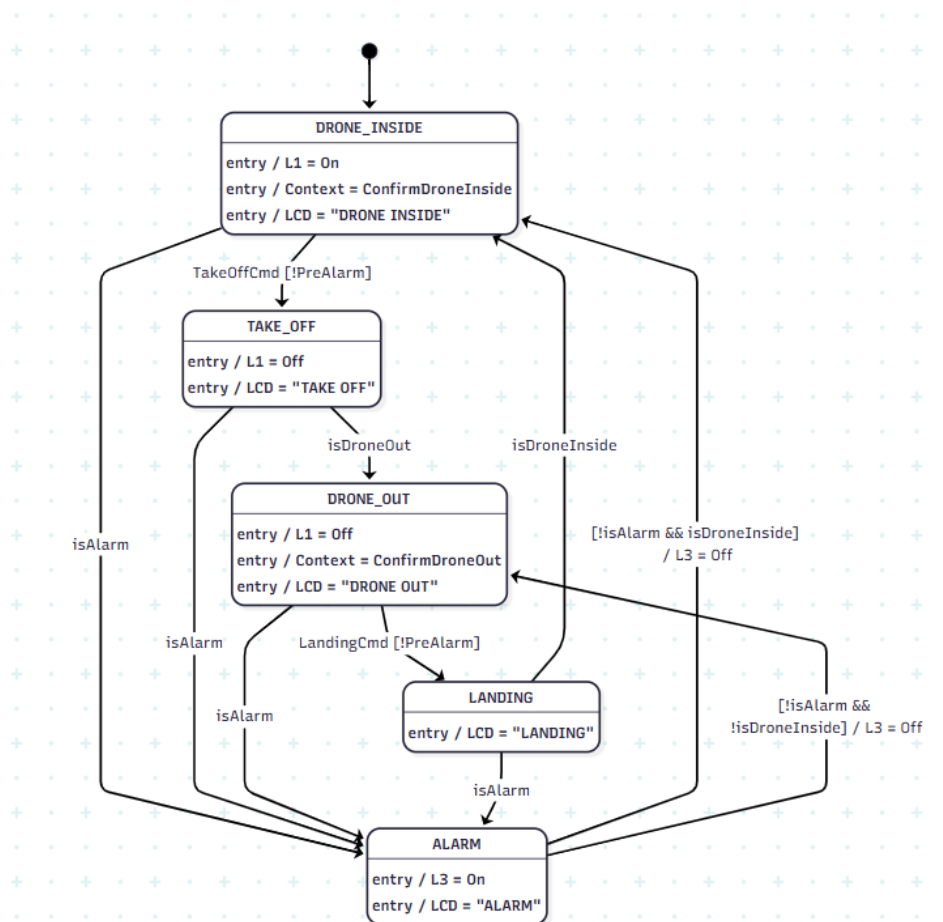


Figura 1: FSM di HangarTask

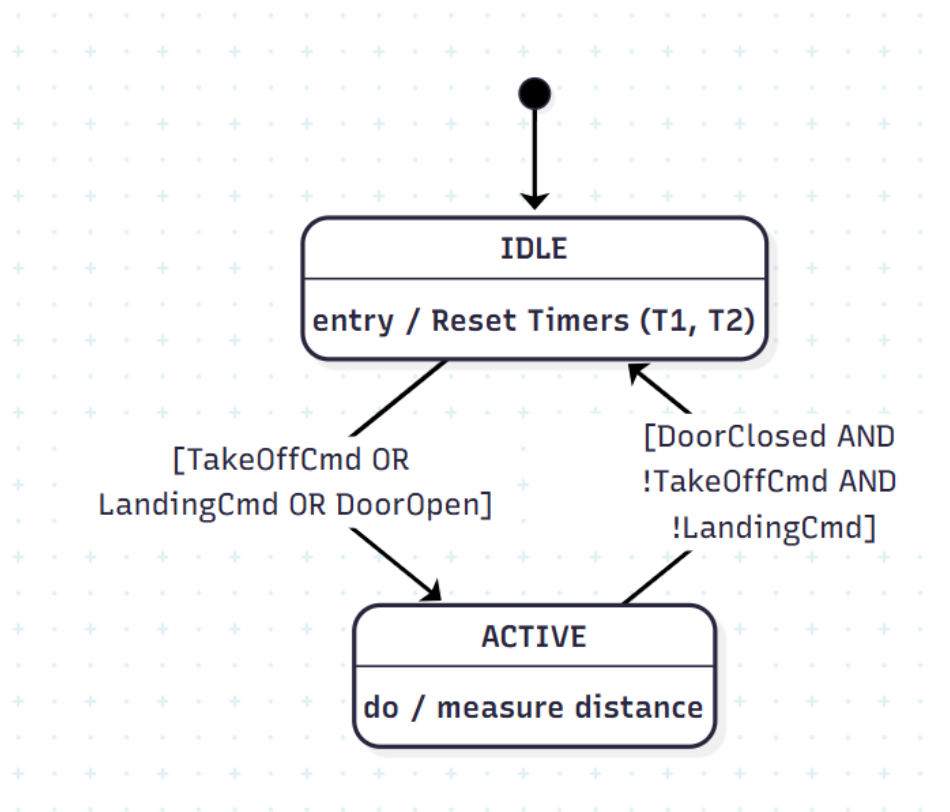


Figura 2: FSM di DistanceDetectorTask

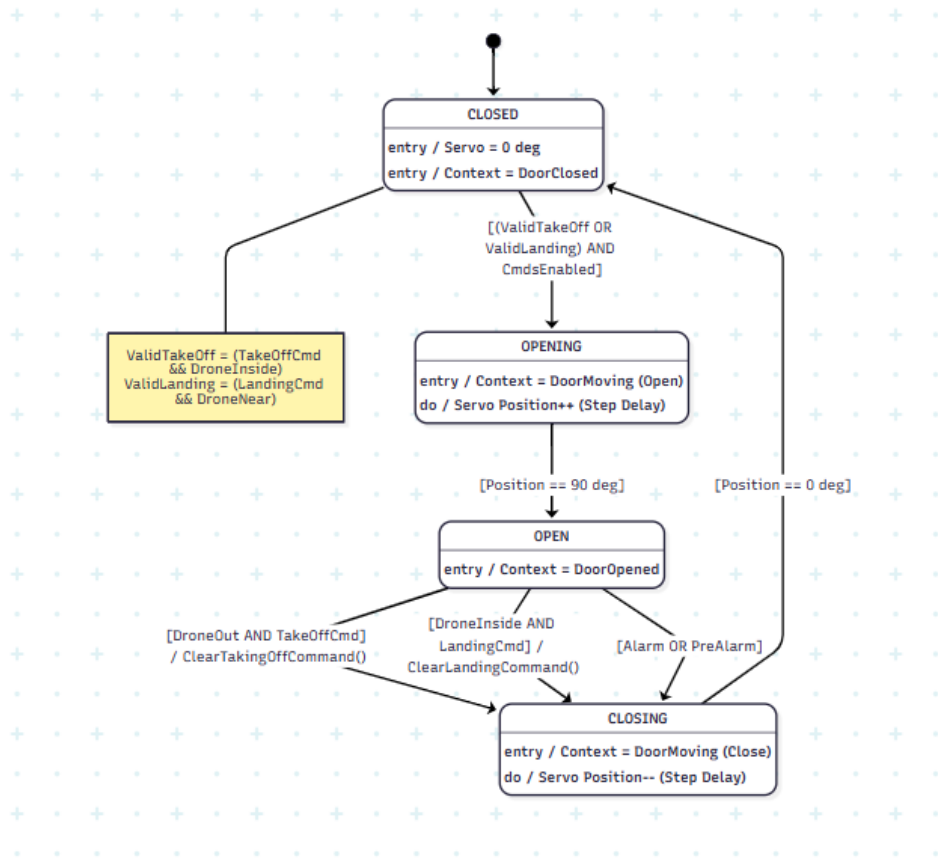


Figura 3: FSM di DoorTask

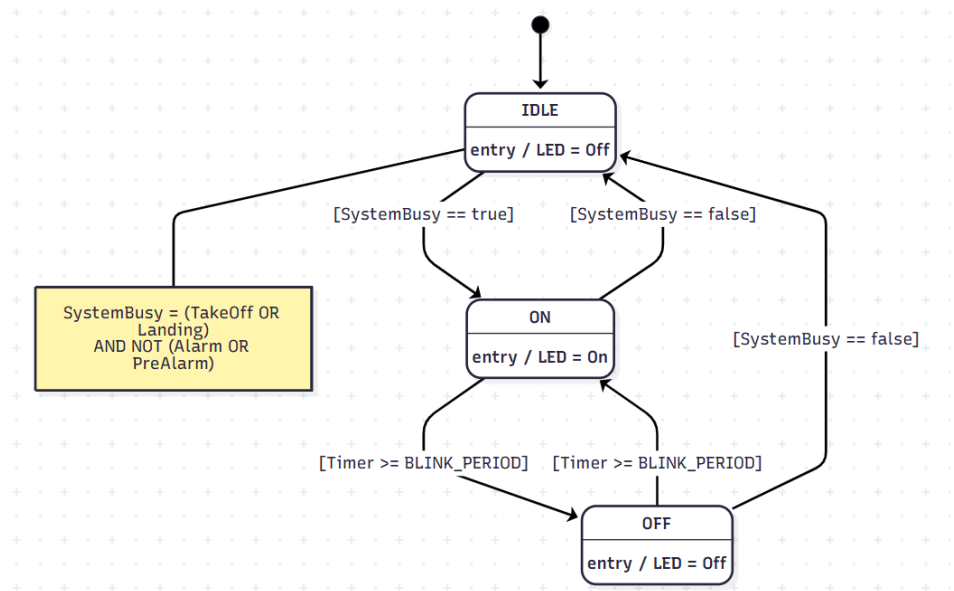


Figura 4: FSM di BlinkingTask

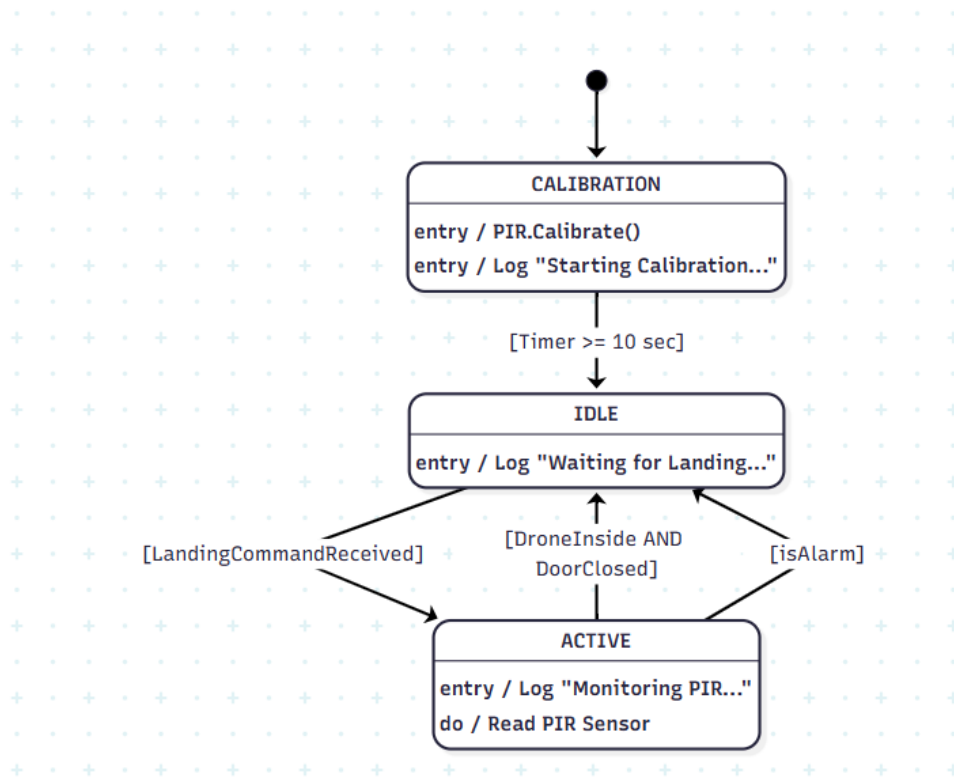


Figura 5: FSM di PresenceDetectorTask

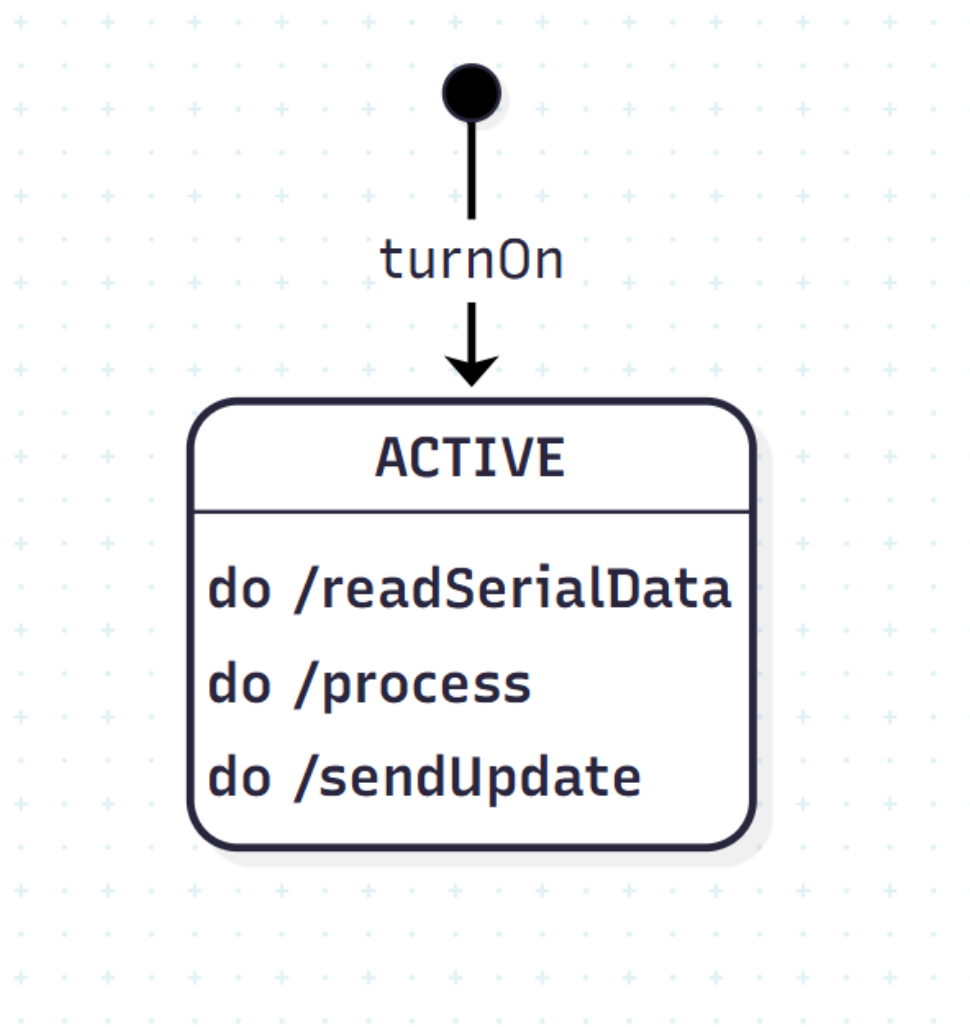


Figura 6: FSM di `SerialCommTask`



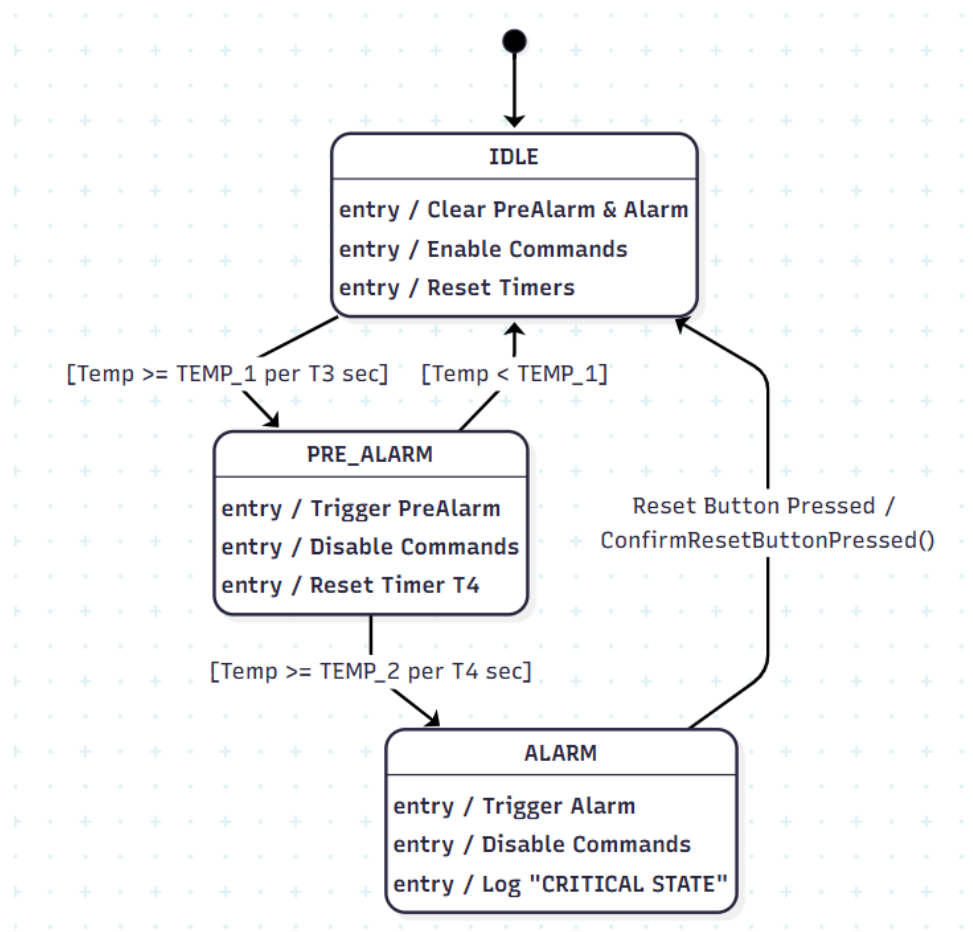


Figura 7: FSM di AlarmTask



## 5 Drone Remote Unit (Java)

### 5.1 GUI

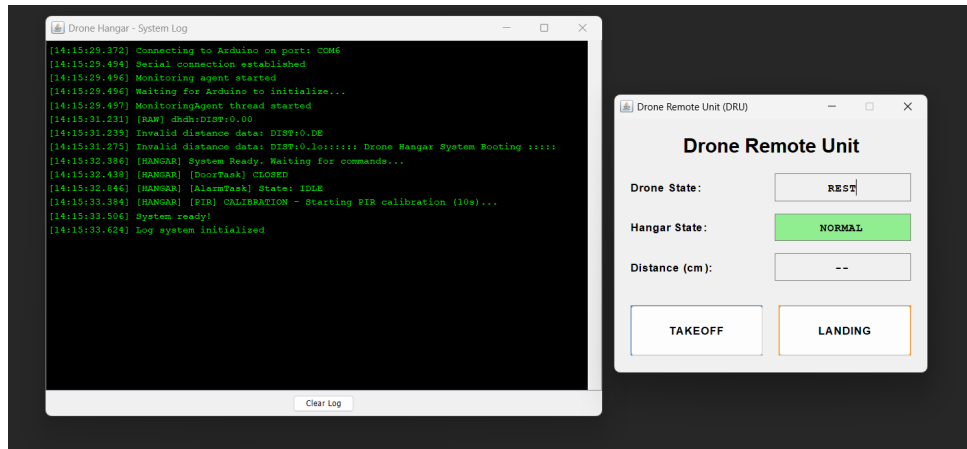


Figura 9: A sinistra: Log View, a destra: Dashboard View

### 5.2 Protocollo di comunicazione seriale tra Arduino e PC

Il sistema utilizza un protocollo testuale basato su prefissi per distinguere la tipologia di informazione scambiata sulla porta seriale:

- **dh:** (Drone Hangar): Messaggi di stato applicativi inviati da Arduino verso la GUI.
- **lo:** (Log): Messaggi di debug e informazioni di sistema per il monitoraggio interno.
- **ru:** (Remote Unit): Comandi inviati dalla GUI Java verso Arduino.

Tabella 2: Dettaglio del Protocollo di Messaggistica

Prefisso	Messaggio / Comando	Descrizione
dh:	STATUS:DRONE_INSIDE	Il drone è all'interno dell'hangar.
dh:	STATUS:TAKING_OFF	Fase di decollo (apertura porta).
dh:	STATUS:DRONE_OUT	Il drone è fuori dall'hangar.
dh:	STATUS:LANDING	Fase di atterraggio (chiusura porta).
dh:	STATUS:ALARM	Stato di allarme critico (sovratemperatura).
dh:	DIST:<valore>	Invio della distanza rilevata dal Sonar (cm).
lo:	System initialized	Notifica di avvio completato.
lo:	CMD TAKEOFF	Log di ricezione comando di decollo.
lo:	[TaskName] State:X	Debug dello stato interno di una specifica task.
lo:	ERR CMD:UNKNOWN	Errore nel parsing del comando ricevuto.
ru:	TAKEOFF	Richiesta di apertura porta per decollo.
ru:	LANDING	Richiesta di chiusura porta per atterraggio.

Prefisso	Messaggio / Comando	Descrizione
ru:	RESET	Ripristino del sistema dopo un allarme.