POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Meccanica



Politecnico di Torino

Airsat: sviluppo di un free-flyer per la simulazione planare di operazioni in orbita

Relatori:

Prof. Stefano Mauro

Correlatori:

Dott. Davide Sorli

Dott. Giulia Calvo

Dott. Ernesto Russo

Candidati:

Antonio De Rosa

A.a. 2025/2026

Indice

Lista delle figure **III**

Lista delle tabelle **V**

Lista degli acronimi **VI**

Nomenclatura **VII**

Abstract **1**

1 Introduzione **3**
1.1 Obiettivi 4

2 Stato dell'arte dei simulatori di satelliti in laboratorio **5**
2.1 POSEIDYN testbed 7
2.2 PINOCCHIO testbed 8
2.3 DISCOWER ATMOS 9
2.4 CubeTAS 11
2.5 Considerazioni sulla dotazione sensoristica 12
2.6 Algoritmi di sensor fusion 13
2.7 Collocazione di AirSat 13

3 Design e componenti di Airsat **15**
3.1 Panoramica dei componenti del prototipo Airsat 15
3.1.1 Implementazione del sistema di visione artificiale 18
3.2 Design completo di Airsat 19

4 Aggiornamento del Digital Twin **21**
4.1 Panoramica del Digital Twin 21
4.2 Blocco Sensori 22
4.2.1 IMU 23

4.2.2 Camera 24
4.2.3 Analisi dei contenuti in frequenza dei sensori 26
4.3 Blocco Filtro di Kalman 30
4.3.1 Performance del filtro di Kalman 31

5 Extended Kalman Filter per la sensor fusion visual-inertial 41
5.1 Richiami teorici sull'EKF 42
5.2 Struttura del filtro EKF nell'AirSat 44
5.2.1 Definizione dello spazio degli stati 44
5.2.2 Fase di prediction: modello di processo e covarianza del filtro . 46
5.2.3 Fase di update: modello di misura 49
5.2.4 Matrici di covarianza dei rumori di processo e di misura 50
5.2.5 Fase di update: calcolo del guadagno di Kalman e covarianza del filtro 52
5.3 L'Extended Kalman Filter nell'Airsat 53

6 Architettura ROS2 54
6.1 Panoramica Generale 55
6.2 Nodo microROS e interfaccia con il microcontrollore 57
6.2.1 Configurazione dei dispositivi hardware 57
6.2.2 Inizializzazione e protocollo di calibrazione del firmware 61
6.2.3 Struttura del ciclo principale del firmware 61
6.3 Nodo Bringup 63
6.4 Nodi Camera e Perception 64
6.5 Nodo Sensor Fusion con filtro di Kalman 67
6.6 Nodo Guidance 72
6.7 Nodo di controllo LQI 73
6.8 Nodo microROS per l'attuazione 76

7 Test e Risultati 78
7.1 Setup del testbench 78
7.2 Validazione filtro di Kalman 80
7.3 Risultati sperimentali 81

8 Conclusioni 84
8.1 Lavori futuri 85

Bibliografia 86

Elenco delle figure

2.1 Schema dei sistemi di simulatori di satelliti esistenti. 6
2.2 POSEIDYN testbed. 7
2.3 Architettura software POSEIDYN. 8
2.4 (a) Piattaforma PINOCCHIO (b) schema dell'architettura GNC di PINOCCHIO. 9
2.5 FSS DISCOWER ATMOS. 10
2.6 Piattaforma CubeTAS 11
2.7 Schema dell'architettura software del CubeTAS 12

3.1 Profilato di alluminio e struttura assemblata 15
3.2 Thruster e cuscinetti ad aria 16
3.3 Componenti del sistema pneumatico 16
3.4 Componenti del sistema reaction wheel 17
3.5 IMU MPU9250 17
3.6 Camera Intel RealSense D435 [1] 18
3.7 Supporti aggiornati 20
3.8 Modello CAD piattaforma AIRSAT 20

4.1 Schema a blocchi del digital twin in MATLAB Simulink 21
4.2 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per l'angolo di assetto 27
4.3 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato del giroscopio per la velocità angolare 28
4.4 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per la posizione x 28
4.5 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato dell'accelerometro lungo l'asse x 29
4.6 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per la posizione y 29

4.7 Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato dell'accelerometro lungo l'asse y 30
4.8 Performance del filtro di Kalman sulla stima dell'angolo di yaw 32
4.9 Performance del filtro di Kalman sulla stima della velocità angolare yawrate 33
4.10 Performance del filtro di Kalman sulla stima della posizione x 34
4.11 Performance del filtro di Kalman sulla stima della posizione y 34
4.12 Performance del filtro di Kalman sulla stima della velocità v_x 35
4.13 Performance del filtro di Kalman sulla stima della velocità v_y 36
4.14 Stima dei bias dell'accelerometro del filtro di Kalman 37
4.15 Stima del bias del giroscopio del filtro di Kalman 37
4.16 Valori RMSE del filtro di Kalman rispetto ai dati di riferimento ideali 38
4.17 Valori RMSE del filtro di Kalman rispetto ai dati di riferimento ideali dell'angolo di yaw 39
4.18 Drift di ψ integrando dal giroscopio, rispetto al valore ideale e filtrato 40
4.19 Valori RMSE del filtro di Kalman rispetto al valore di ψ integrando nel tempo senza sensor fusion 40

5.1 Schema dei sistemi di riferimento del sistema Airsat 45

6.1 Schema delle connessioni delle periferiche di Airsat. 55
6.2 Rappresentazione del grafo di comunicazione ROS2. 56
6.3 Schema logico dei pin del microcontrollore Arduino Nano ESP32. . . 58
6.4 Rappresentazione schematica dei sistemi di riferimento 66
6.5 Schema a blocchi del flusso di controllo LQI e attuazione Reaction Wheel. 76

7.1 Setup sperimentale per le prove di laboratorio 79
7.2 Marker ArUco utilizzato nel test 79
7.3 Performance Kalman sulla stima dell'angolo 81
7.4 Performance Kalman sulla stima della velocità angolare 82
7.5 Performance Kalman: confronto RMSE 83

Elenco delle tabelle

3.1 Principali caratteristiche della Intel RealSense D435 19

4.1 Parametri caratteristici del modello dell'accelerometro lungo gli assi x e y 24
4.2 Parametri caratteristici del modello del giroscopio attorno all'asse z . 24
4.3 Parametri del modello di camera virtuale per la rotazione 25
4.4 Parametri del modello di camera virtuale per la traslazione lungo x . 25
4.5 Parametri del modello di camera virtuale per la traslazione lungo y . 26
4.6 Parametri delle matrici di covarianza filtro di Kalman sul digital twin 31

5.1 Variabili di stato nell'EKF dell'AirSat. 45
5.2 Variabili di input del modello di processo nell'EKF dell'AirSat. 47
5.3 Variabili del modello di misura nell'EKF dell'AirSat. 50
5.4 Parametri di taratura dell'EKF utilizzati nell'implementazione AirSat. 51
5.5 Elenco delle matrici utilizzate nell'EKF di AirSat. 52

6.1 Connessioni tra microcontrollore e DEC Module 50/5. 59
6.2 Configurazioni del range di velocità del motore. 59
6.3 Parametri del Motore Maxon EC45 60
6.4 Connessioni tra microcontrollore e MPU9250. 60
6.5 Parametri del nodo Camera. 64
6.6 Parametri del Nodo Perception. 65
6.7 Parametri di Process Noise Q e Measurement Noise R 72
6.8 Parametri del file `guidance_params.yaml`. 73
6.9 Parametri del file `control_params.yaml`. 74

Lista degli acronimi

ADCS	Attitude Determination and Control System
CAD	Computer Aided Design
DLPF	Digital Low Pass Filter
DOF	Degree Of Freedom
EKF	Extended Kalman Filter
FSS	Floating Spacecraft Simulator
GNC	Guidance, Navigation and Control
HIL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
IR	Infra Red
LQI	Linear Quadratic Integral
LSB	Less Significant Bit
OOS	On Orbit Services
PSD	Power Spectral Density
PWM	Pulse Width Modulation
RMSE	Root Mean Square Error
ROS	Robot Operating System
SdR	Sistema di Riferimento

Nomenclatura

x	Coordinata laterale nel SdR inerziale
y	Coordinata longitudinale nel SdR inerziale
$\dot{x}$	Velocità laterale nel SdR inerziale
$\dot{y}$	Velocità longitudinale nel SdR inerziale
$\ddot{x}^I$	Accelerazione laterale nel SdR inerziale
$\ddot{y}^I$	Accelerazione longitudinale nel SdR inerziale
ψ	Angolo di assetto nel SdR inerziale
$\dot{\psi}^I$	Velocità angolare nel SdR inerziale
x^m	Coordinata laterale acquisita dalla camera
y^m	Coordinata longitudinale acquisita dalla camera
ψ^m	Angolo di assetto acquisito dalla camera
a_x^m	Accelerazione laterale acquisita dall'IMU
a_y^m	Accelerazione longitudinale acquisita dall'IMU
g_z^m	Velocità angolare acquisita dall'IMU
b_{ax}	Bias stimato dal filtro sull'acquisizione di a_x
b_{ay}	Bias stimato dal filtro sull'acquisizione di a_y
b_{gz}	Bias stimato dal filtro sull'acquisizione di g_z
a_x^c	Accelerazione laterale acquisita dall'IMU corretta con il bias b_{ax}
a_y^c	Accelerazione longitudinale acquisita dall'IMU corretta con il bias b_{ay}
g_z^c	Velocità angolare acquisita dall'IMU corretta con il bias b_{gz}
$\mathbf{SS_t}$	Vettore dello spazio degli stati del EKF
$\mathbf{u_t}$	Vettore degli input del EKF
$f(SS_t, u_t)$	Funzione dello stato del EKF
w_t	Rumore di processo del EKF
$\mathbf{z_t}$	Vettore delle misure del EKF

$h(SS_t)$	Funzione delle misure del EKF
v_t	Rumore di misura del EKF
$\mathbf{F_t}$	Matrice Jacobiana del modello di processo del EKF
$\mathbf{V_t}$	Matrice Jacobiana di propagazione del rumore di processo del EKF
$\mathbf{P_t}$	Matrice di covarianza degli stati del EKF
$\mathbf{Q}$	Matrice di covarianza del rumore di processo del EKF
$\mathbf{R}$	Matrice di covarianza del rumore di misura del EKF
$\mathbf{K_t}$	Matrice del guadagno di Kalman del EKF
$\mathbf{i_t}$	Vettore dei residui di innovazione del EKF
σ_{ax}	Deviazione standard di a_x^m
σ_{ay}	Deviazione standard di a_y^m
σ_{g_z}	Deviazione standard di g_z^m
$\sigma_{b_{ax}}$	Deviazione standard di b_{ax}
$\sigma_{b_{ay}}$	Deviazione standard di b_{ay}
$\sigma_{b_{gz}}$	Deviazione standard di b_{gz}
σ_x	Deviazione standard di x^m
σ_y	Deviazione standard di y^m
σ_ψ	Deviazione standard di ψ^m

Abstract

Italiano

La simulazione in laboratorio di condizioni di microgravità riveste un ruolo centrale nello sviluppo e nella validazione di tecnologie spaziali destinate agli *On Orbit Services* e alle manovre di *rendezvous*. Questo lavoro di tesi descrive lo sviluppo e la sperimentazione di *AirSat*, un free-flyer planare a tre gradi di libertà, progettato per simulare le dinamiche di navigazione di un satellite in microgravità mediante l'impiego di cuscinetti ad aria compressa, propulsori pneumatici e ruota di reazione. L'obiettivo è l'aggiornamento del sistema di sensori e di controllo con lo scopo di permettere manovre basate su sistemi di visione. Sulla base del prototipo esistente, completo di sistemi pneumatici ed elettrici, il presente lavoro estende il comparto sensoristico, precedentemente basato su un solo sensore inerziale, integrando un sistema di visione artificiale tramite camera Intel RealSense D435, e implementa un algoritmo di sensor fusion per la stima dello stato odometrico basato su Extended Kalman Filter (EKF). In questo modo è possibile compensare il drift del sensore inerziale e difetti di campionamento qualora si verifichino disturbi esterni. L'efficacia dei sensori scelti e del filtro implementato è stata valutata tramite simulazione di manovre in ambiente Matlab/Simulink-Simscape.

L'intera architettura software è sviluppata in ambiente ROS 2, con una struttura modulare che comprende nodi dedicati alla percezione, alla stima dello stato con EKF, al controllo LQI e all'attuazione tramite reaction wheel. Il microcontrollore ESP32, interfacciato via microROS, gestisce in tempo reale l'acquisizione IMU e l'azionamento del driver motore della reaction wheel.

Tramite l'esecuzione di test sperimentali sulla piattaforma free-flyer è stato verificato l'efficace funzionamento del filtro di Kalman. I dati sperimentali raccolti hanno inoltre permesso un'accurata modellazione dei fenomeni di rumore e drift caratteristici dei sensori utilizzati.

English

Ground-based simulation of microgravity conditions plays a fundamental role in the development and validation of technologies intended for *On Orbit Services* and *rendezvous* maneuvers. This thesis describes the development and experimental validation of *AirSat*, a planar free-flyer with three degrees of freedom, designed to reproduce satellite microgravity dynamics in a laboratory environment using air bearing pads, pneumatic thrusters, and a reaction wheel. The ultimate objective is the update of the sensor and control system in order to achieve vision-based maneuvers.

Building upon the existing prototype, already equipped with pneumatic and electrical systems, this work extends the sensor unit — previously based on a single inertial sensor — by integrating a computer vision system using an Intel RealSense D435 camera. Furthermore, a sensor fusion algorithm for odometric state estimation based on an Extended Kalman Filter (EKF) is implemented. In this way, inertial sensor drift and sampling failure are mitigated in the event of external disturbances. The sensors and filter effectiveness have been evaluated through maneuvers' simulation in Matlab/Simulink-Simscape environment.

The software architecture is developed within the ROS 2 framework, featuring a modular pipeline with dedicated nodes for perception, EKF-based state estimation, LQI control, and reaction wheel actuation. An ESP32 microcontroller, interfaced via microROS, handles real time IMU acquisition and reaction wheel motor driver actuation.

The functioning effectiveness of the Kalman filter has been verified through the execution of experimental tests on the free-flyer. Furthermore, the experimental data collected also allowed for an accurate modelling of the characteristic noise and drift phenomena of the implemented sensors.

Capitolo 1

Introduzione

Alla base di questo progetto di tesi vi è la l'esigenza di sviluppare, presso il Politecnico di Torino, un Floating Spacecraft Simulator (FSS). A questo scopo, in lavori precedenti è stato concepito e assemblato il prototipo Airsat. Airsat è un simulatore dinamico planare a tre gradi di libertà, due per la traslazione e uno per la rotazione; la finalità dell'iniziativa è la realizzazione di una piattaforma destinata alla sperimentazione in microgravità di on orbit servicing attraverso l'uso di algoritmi di Guidance, Navigation e Control (GNC) e, più in generale, nel contributo all'avanzamento della ricerca nel settore della robotica spaziale e dei sistemi autonomi. Nello specifico, lo scopo di questo lavoro di tesi consiste nel continuare lo sviluppo dell'Airsat con l'implementazione di nuovi sensori e algoritmi per l'affinamento delle strategie di GNC.

Nei prossimi due capitoli è presente una panoramica sullo stato dell'arte di piattaforme sperimentali simili impiegate nella simulazione della dinamica di veicoli spaziali, e l'illustrazione generale del prototipo AIRSAT, evidenziando gli elementi caratteristici adottati nel progetto. Successivamente verrà presentato il lavoro di aggiornamento del prototipo: nel capitolo quattro viene approfondita l'integrazione di modelli di sensori realistici e del filtro di Kalman nel digital twin; il capitolo cinque è dedicato alla trattazione matematica della teoria del filtro di Kalman applicata per la Sensor Fusion; nel capitolo sei è illustrata la realizzazione del workspace ROS2 che starà alla base della gestione software del prototipo; infine, nel capitolo sette sono presentati i risultati ottenuti e il banco di prova di laboratorio utilizzato per condurre i test.

1.1 Obiettivi

Il primo obiettivo che si pone questo lavoro è l'aggiornamento dell'apparato sensoristico di Airsat, attualmente basato sulla sola unità inerziale (IMU), il cui impiego come sensore unico presenta limiti significativi, in particolare nella stima della posizione e dell'assetto planare a lungo termine. In tale contesto, si intende ampliare le capacità del prototipo mediante l'integrazione di un sistema di visione artificiale basato su camera, al fine di rendere più robusta e completa la percezione dello stato odometrico ed ampliare gli scenari di utilizzo del Floating Spacecraft Simulator. La coesistenza di più sensori rende necessaria l'adozione di tecniche di sensor fusion, per combinare in modo coerente i contributi del sistema inerziale e del sistema di visione. Per valutare i risultati della sensor fusion, è stato aggiornato il digital twin di Airsat. La prima modifica risiede nel sottosistema sensoristico, nel quale i relativi modelli sono stati ricostruiti sulla base delle caratteristiche determinate sperimentalmente dei due sensori; mentre la seconda modifica consiste nell'implementazione del nuovo sottosistema di sensor fusion che integra un filtro di Kalman.

Il secondo obiettivo consiste nello sviluppo dell'architettura software e dell'implementazione hardware necessarie a trasferire sul prototipo Airsat quanto sviluppato nel digital twin, mediante un workspace in ambiente Linux/ROS2 orientato alla percezione, alla stima e al controllo del sistema.

L'obiettivo ultimo consiste nella valutazione, con prove sperimentali, delle prestazioni dei nuovi sensori implementati. Tali test saranno volti ad analizzarne il comportamento in termini di affidabilità e qualità delle misure fornite, nonché a valutare il processo di stima dello stato. In questo modo, sarà possibile verificare la validità delle soluzioni adottate sia all'interno del digital twin sia nell'integrazione nel prototipo reale.

Capitolo 2

Stato dell'arte dei simulatori di satelliti in laboratorio

La complessità delle missioni spaziali moderne, in scenari operativi quali il rendezvous, il docking e l'on-orbit servicing (OOS), rende necessario lo sviluppo di piattaforme sperimentali terrestri in grado di simulare in modo controllato le condizioni dinamiche caratteristiche dell'ambiente orbitale. In tal senso, i Floating Spacecraft Simulator (FSS) forniscono supporto per la validazione di architetture GNC, poiché consentono di testare prototipi reali, sensori e algoritmi di stima e controllo in un contesto ripetibile, sicuro e a costi inferiori rispetto alla sperimentazione in volo. In letteratura si distinguono diverse classi di simulatori di FSS, in funzione di diversi elementi che caratterizzano il modo in cui si conduce la simulazione, come illustrato nella figura 2.1 [2].

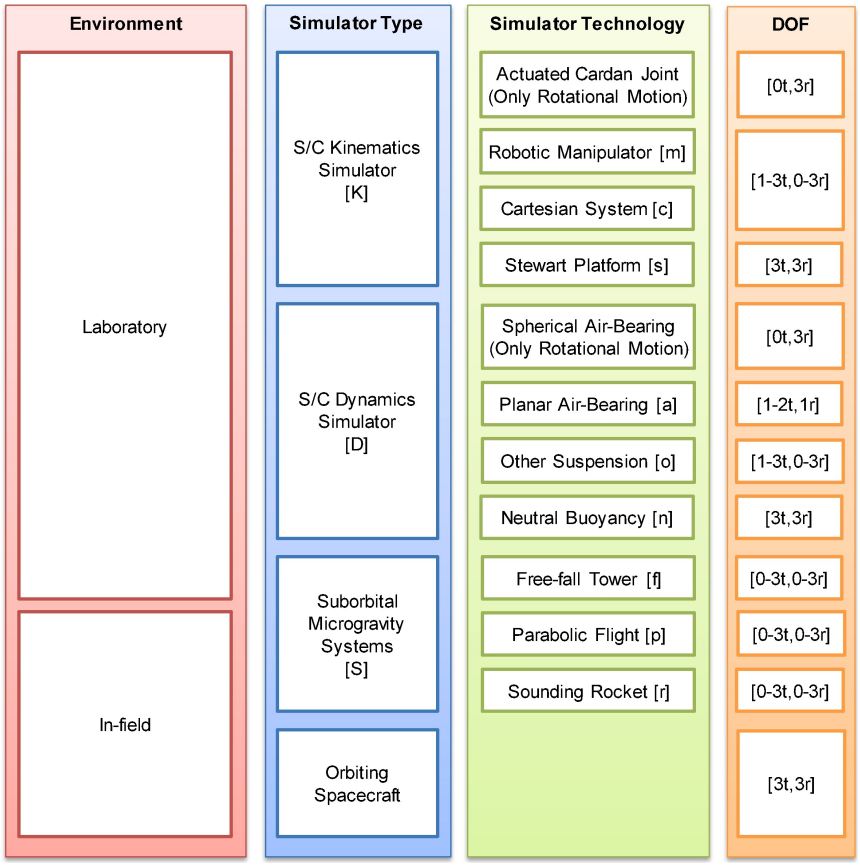


Figura 2.1: Schema dei sistemi di simulatori di satelliti esistenti.

Secondo questa classificazione è possibile collocare il presente progetto all'interno dei Dynamic Simulator in Laboratory Environment. Questo tipo di simulatori riproducono in modo diretto le forze e le coppie mediante attuatori fisici a bordo. Tale configurazione consente di valutare il comportamento del sistema in un contesto più rappresentativo della realtà operativa, includendo gli effetti della sensoristica e della catena di attuazione. La condizione di microgravità è realizzata mediante Planar Air-Bearings che riducono quasi totalmente l'attrito e consentono il moto su un piano con tre gradi di libertà, due traslazionali e uno rotazionale. Questa soluzione, pur non riproducendo il sistema completo a sei gradi di libertà, è una delle più efficaci in contesti di missioni di prossimità orbitale, poiché consente di validare con buona fedeltà algoritmi di navigazione relativa, controllo di free-flyer e logiche di interazione tra chaser e target [2].

2.1 POSEIDYN testbed

Tra i riferimenti più significativi nel panorama dei simulatori planari vi è il testbed POSEIDYN, sviluppato presso il Naval Postgraduate School. Considerata la stretta affinità concettuale con il presente progetto AIRSAT, si procede con una breve analisi della sua architettura e dei principi operativi su cui si basa.

I FSS sviluppati nel progetto POSEIDYN si basano su un sistema a cuscinetti d'aria che permette il moto planare con attrito trascurabile, replicando le condizioni dinamiche orbitali. La traslazione è ottenuta tramite l'utilizzo di propulsori, mentre la rotazione tramite una ruota di reazione. Infine, i FSS vengono equipaggiati con dei sensori onboard tra cui Unità di Misura Inerziali (IMU) e camere. La più recente evoluzione, presenta una struttura alleggerita grazie alla sostituzione dell'alluminio con componenti in policarbonato realizzati con l'additive manufacturing [3][4].

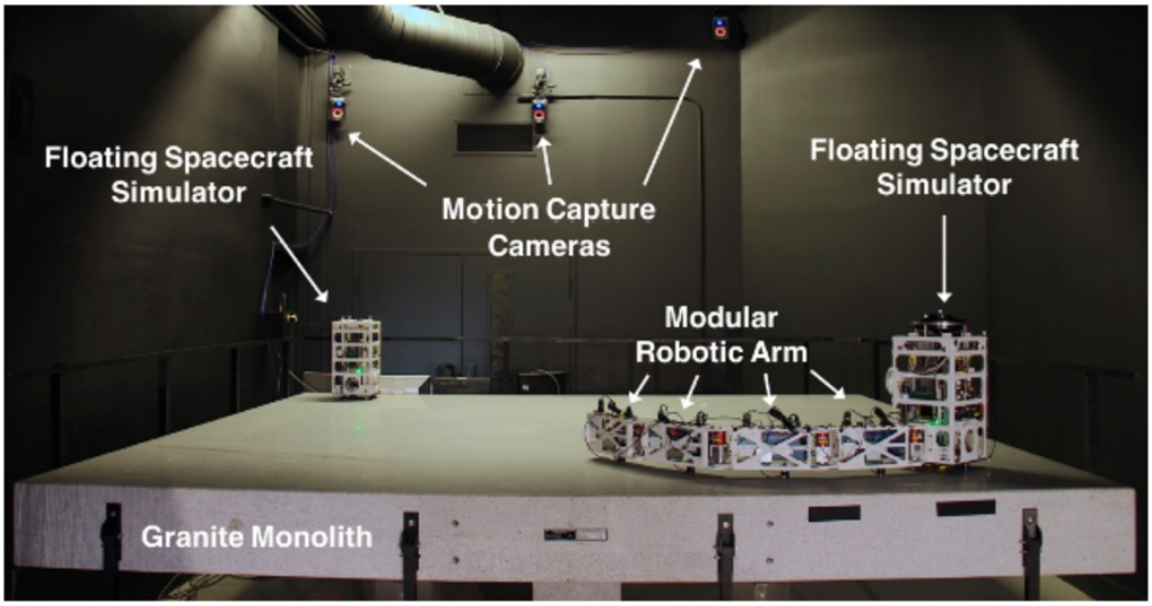


Figura 2.2: POSEIDYN testbed.

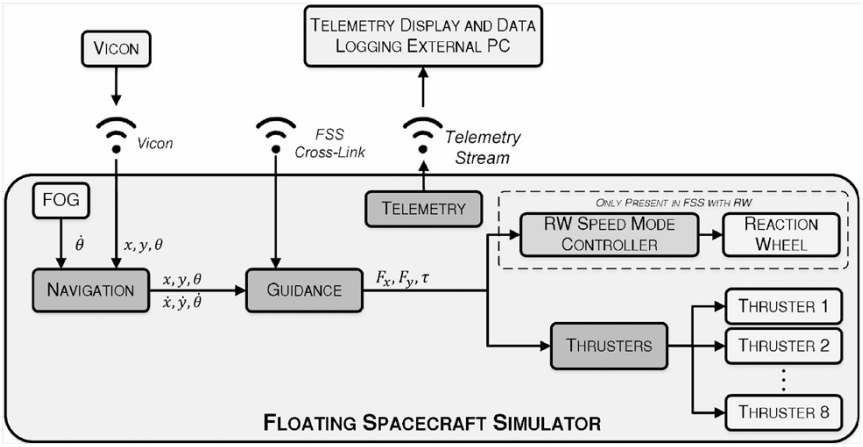


Figura 2.3: Architettura software POSEIDYN.

La figura 2.3 illustra l'architettura software implementata nel progetto POSEIDYN, che adottando una struttura modulare, permette l'integrazione tra simulazione di dinamiche orbitali, lettura dai sensori e algoritmi di controllo in un contesto di HIL in tempo reale. In particolare, le acquisizioni dei sensori vengono processate da moduli dedicati, ad esempio nel sottosistema Navigation agisce il filtro di Kalman, per ottenere una Sensor Fusion tra le acquisizioni inerziali e visive. Successivamente, le informazioni vengono rielaborate dai moduli di Guidance e Control e convertite in comandi per gli attuatori di movimento: propulsori (thruster) o ruota di reazione (reaction wheel). Questa configurazione rende il FSS flessibile e riconfigurabile, per futuri miglioramenti e implementazione di nuova componentistica. [3]

Il progetto si pone come obiettivi, quelli già citati di replicare la dinamica in microgravità di operazioni di prossimità e la validazione di strategie e algoritmi per la navigazione. [3]

2.2 PINOCCHIO testbed

Un secondo riferimento in questo ambito è rappresentato dalla testbed PINOCCHIO, sviluppata presso l'Università Sapienza di Roma. Il progetto della piattaforma nasce per riprodurre la microgravità planare al fine di condurre dei test su strategie di Guidance, leggi di controllo e sensori di navigazione. Il FSS è dotato di sensori inerziali, optical sensor device e filtro di Kalman per la fusione delle misure, e anche in questo caso il moto avviene su cuscinetti ad aria (air bearing) e l'attuazione è affidata a propulsori (cold-gas thrusters) [2][5].

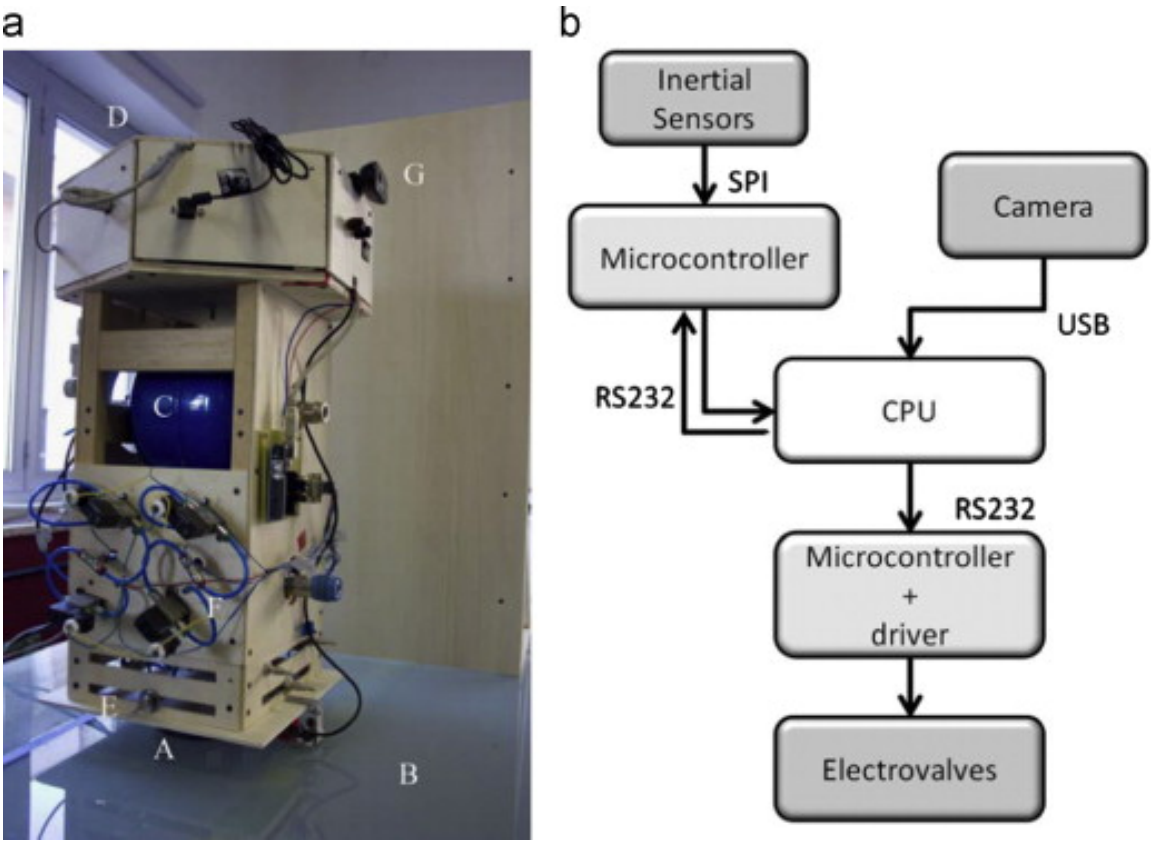


Figura 2.4: (a) Piattaforma PINOCCHIO (b) schema dell'architettura GNC di PINOCCHIO.

PINOCCHIO mostra quindi un passaggio significativo dello stato dell'arte: l'abbandono di una dipendenza esclusiva dalle infrastrutture esterne di laboratorio a favore di una navigazione più strettamente onboard, già concettualmente vicina a quella di uno spacecraft [5].

2.3 DISCOWER ATMOS

Una evoluzione ulteriore è rappresentata dal testbed DISCOWER ATMOS (Autonomy Testbed for Multi-purpose Orbiting Systems), sviluppato presso il KTH Space Robotics Laboratory come piattaforma modulare e open-source per la robotica spaziale sperimentale [6].

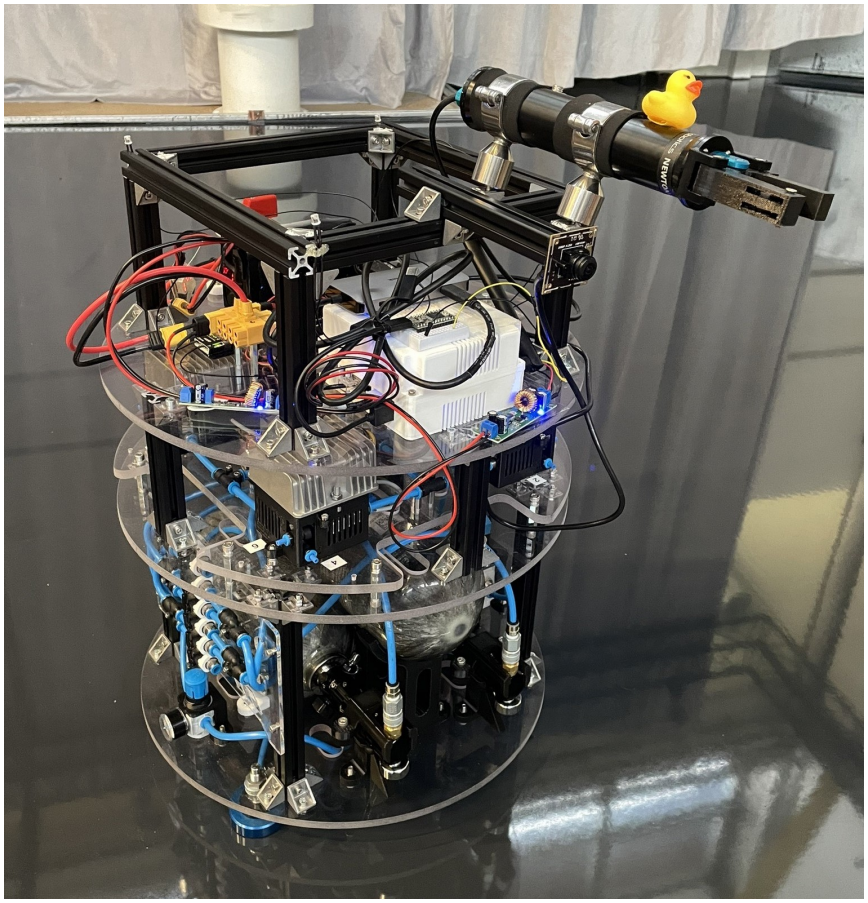


Figura 2.5: FSS DISCOWER ATMOS.

ATMOS è progettato come free-flyer planare su air bearing, integra propulsori attuati da solenoidi e presenta un sistema di supporto che consente la sperimentazione di diverse configurazioni e strumentazioni. La natura open-source del progetto ne amplia ulteriormente il valore, rendendo documentazione, distinta base e istruzioni di assemblaggio liberamente accessibili, così da favorire la replicabilità e la standardizzazione delle metodologie sperimentali nella comunità della robotica spaziale. Il nucleo dell'architettura Hardware è l'Avionic Plate, che ospita un controllore Pixhawk e una scheda NVIDIA Jetson Orin NX. Il Pixhawk svolge la funzione di flight-controller a basso livello, gestendo la fusione delle misure inerziali con gli altri sensori e stabilizzando assetto e moto del free-flyer. Inoltre, si interfaccia con il sistema di attuazione eseguendo i comandi di controllo e stabilizzando il moto del FSS in tempo reale durante le manovre. La scheda Jetson, invece, si occupa dell'elaborazione ad alte prestazioni, supportando algoritmi di intelligenza artificiale per la percezione autonoma del contesto ambientale e il processo decisionale in tempo reale su task complesse. La combinazione di questi due sistemi consente ad ATMOS di coniugare un controllo robusto con capacità di elaborazione onboard intelligenti [6].

2.4 CubeTAS

Oltre ai simulatori planari per la navigazione di prossimità, esistono banchi prova specificamente orientati alla validazione di Attitude Determination and Control Systems (ADCS) [7][8]. In questo ambito, uno dei riferimenti principali è CubeTAS, cioè un simulatore rotazionale a tre assi basato su una struttura emisferica flottante che sfrutta spherical air bearing, per riprodurre con elevata fedeltà la dinamica d'assetto di un piccolo satellite. CubeTAS ospita a bordo componenti tipici di un ADCS, tra cui tre reaction wheel, tre magnetorquer, un computer di bordo, un digital sun sensor e una IMU triassiale. A questa dotazione onboard si affianca un sistema di metrologia esterna ad elevata accuratezza basato su quattro stereo-camere, che misurano la posa della piattaforma tramite LED attivi montati sulla struttura [7].

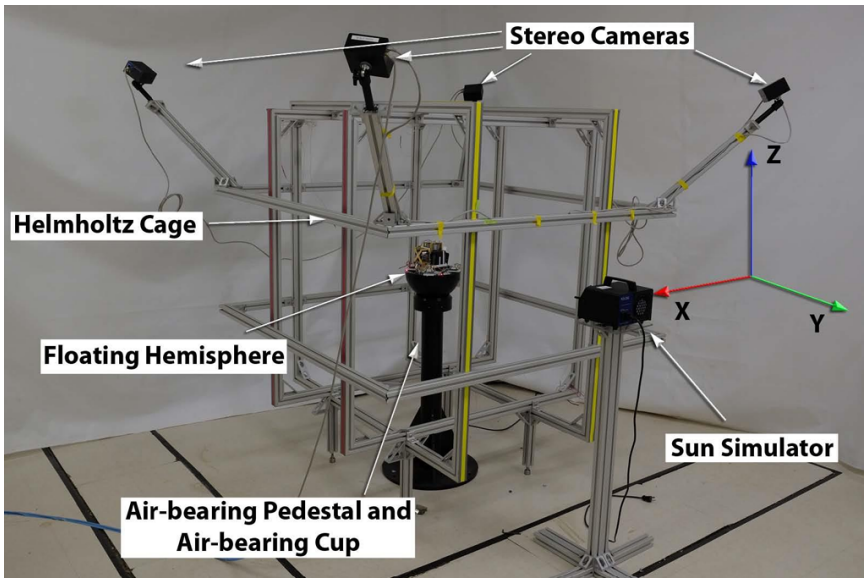


Figura 2.6: Piattaforma CubeTAS

Come illustrato nella figura 2.7 l'architettura sensoristica di CubeTAS risulta molto articolata in quanto combina in un unico banco prova: sensori inerziali, sensore solare e metrologia ottica esterna; pertanto, anche qui viene implementato un Kalman Filter per la stima dell'assetto [7].

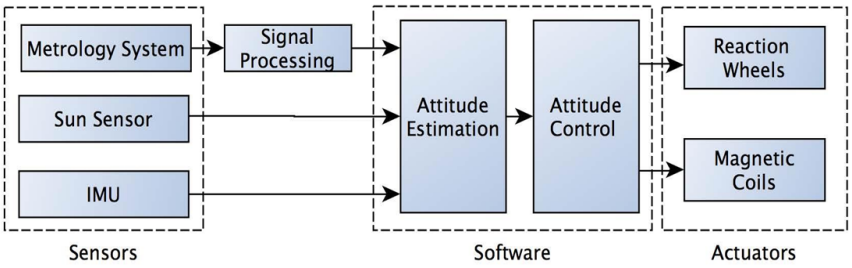


Figura 2.7: Schema dell'architettura software del CubeTAS

Anche se il focus principale di CubeTas si differenzia da quella del presente progetto, esso conferma dunque la centralità del problema della sensor fusion nell'intero ciclo di validazione sperimentale di un ADCS [8].

2.5 Considerazioni sulla dotazione sensoristica

Come illustrato negli esempi precedenti un fattore determinante che influisce sulla qualità di un FSS risiede nel suo ecosistema sensoriale. In particolare, i testbed più evoluti sono caratterizzati dalla presenza di architetture multi-sensore, nelle quali i dati provenienti da dispositivi eterogenei vengono integrati in tempo reale allo scopo di ottenere una stima affidabile dello stato del veicolo. Dal punto di vista funzionale, la sensoristica impiegata nei simulatori può essere distinta in due categorie: sensori off-board e sensori onboard. Alla prima categoria appartengono i sistemi esterni di motion capture, le camere fisse per tracking assoluto e in generale le infrastrutture ottiche; alla seconda appartengono invece le IMU, i magnetometri, le camere RGB o depth, i dispositivi di optical flow, i Lidar e altri sensori di prossimità o percezione tridimensionale installati direttamente sul veicolo [2][7].

I sensori off-board offrono in genere misure ad alta accuratezza e sono particolarmente utili per fornire ground truth sperimentali o riferimenti assoluti durante la validazione degli algoritmi. Tuttavia, il loro impiego introduce una forte dipendenza dall'infrastruttura di laboratorio e non riproduce in modo completo il comportamento di un vero sistema di bordo autonomo. Per questo motivo, una tendenza consolidata nella ricerca è orientata verso la sensoristica onboard, demandando a opportuni algoritmi di sensor fusion la ricostruzione robusta dell'odometria del FSS [2][7]. In questo contesto, la Inertial Measurement Unit rappresenta il nucleo sensoriale minimo più diffuso e trasversale tra i diversi testbed esaminati [7][8]. L'IMU fornisce misure di accelerazione lineare e velocità angolare ad alta frequenza, risultando quindi particolarmente adatta alla propagazione temporale dello stato; tuttavia, l'integrazione di tali misure porta inevitabilmente all'accumulo di errori dovuti a rumore, bias e deriva

[8]. Proprio questa limitazione ha reso centrale la combinazione dell'IMU con sensori ottici capaci di correggere la deriva inerziale mediante osservazioni assolute o relative a più lungo termine [2].

2.6 Algoritmi di sensor fusion

Nel panorama attuale della ricerca sui FSS emerge una tendenza consolidata verso architetture multisensore nelle quali la Sensor fusion è un elemento essenziale per una piattaforma sperimentale avanzata [2][7]. Nessun sensore, infatti, è in grado da solo di fornire una stima di stato completa e robusta in tutte le condizioni operative: le misure inerziali sono rapide ma soggette a drift, quelle ottiche e geometriche sono più stabili ma meno frequenti e più sensibili al contesto ambientale. La sensor fusion si afferma come soluzione a questa complementarità [2].

La famiglia di algoritmi più diffusa nelle fonti esaminate è quella dei filtri di Kalman [7][8]. Nei casi in cui il problema viene modellato in forma sufficientemente lineare, il Kalman filter standard costituisce una scelta naturale per combinare misure provenienti da IMU, camere e dispositivi di optical flow. Quando invece la dinamica del sistema o il modello di misura risultano marcatamente non lineari, la letteratura ricorre a estensioni quali Extended Kalman Filter e Unscented Kalman Filter, come mostrano rispettivamente il lavoro sul MACS testbed e quello su CubeTAS [8][7].

2.7 Collocazione di AirSat

Per concludere, si può notare come lo stato dell'arte evidenzi delle caratteristiche ricorrenti: la presenza della IMU come requisito tra i sensori on-board; il ricorso a sensori ottici e vision-based, utilizzati sia come riferimenti assoluti sia come sensori di navigazione relativa; e l'implementazione di algoritmi di Sensor fusion come i filtri di Kalman per la stima robusta dello stato [2][7][8].

Pertanto, il progetto AirSat si colloca in modo coerente all'interno delle linee più attuali della ricerca sui FSS. La combinazione di IMU e camera, insieme all'impiego di un filtro di Kalman per la stima dello stato e di una architettura di controllo avanzata, risponde infatti ai principi più consolidati emersi dall'analisi della letteratura: complementarità tra misure inerziali e visive, necessità di integrazione multi-sensore e stretta interdipendenza tra stima, percezione e controllo [2][7].

AIRSAT, in definitiva, si avvicina ai simulatori planari come POSEIDYN e PINOCCHIO per l'impiego di un free-flyer con navigazione inerziale e visiva, ma si

pone anche in continuità con piattaforme più recenti come DISCOWER ATMOS per l'attenzione alla modularità e all'autonomia [2].

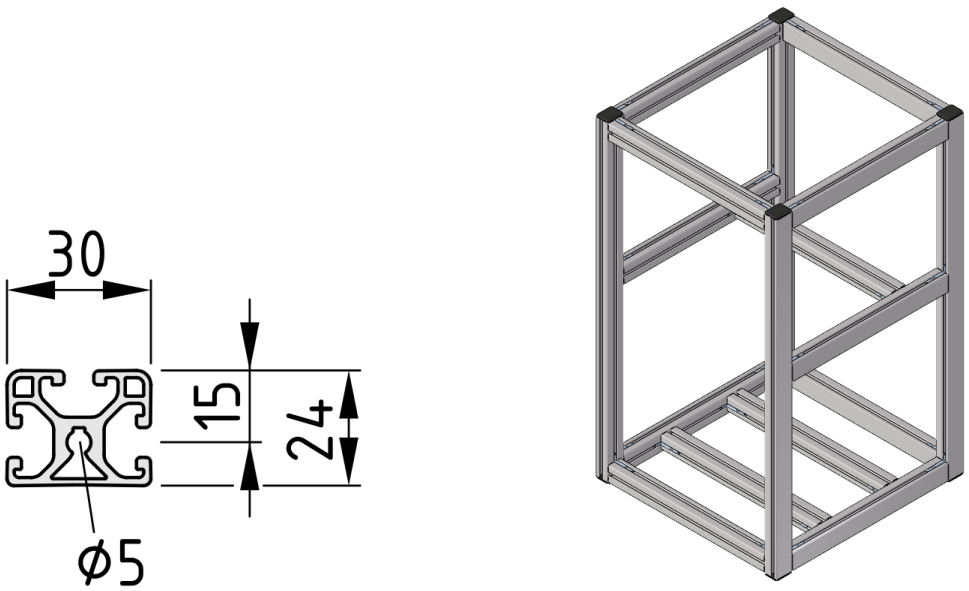
Capitolo 3

Design e componenti di Airsat

Il seguente capitolo è dedicato alla presentazione dei principali componenti che costituiscono il prototipo Airsat.

3.1 Panoramica dei componenti del prototipo Airsat

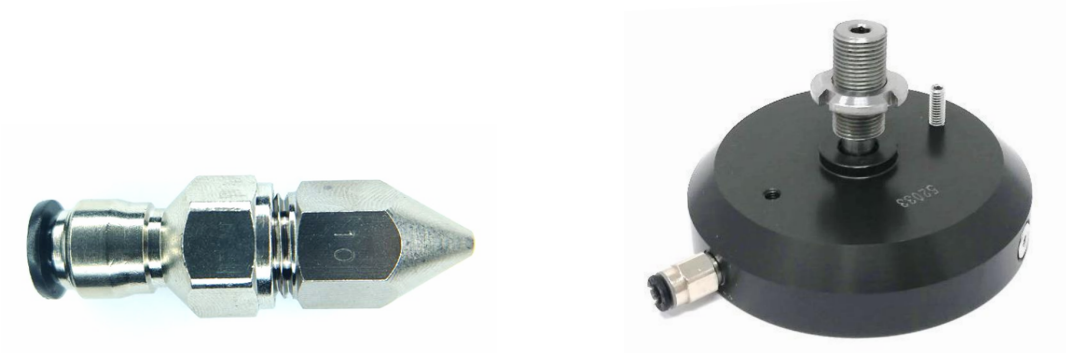
La struttura del prototipo si serve di profili sagomati di alluminio come visibile in Figura 3.1a. La struttura complessiva è ottenuta con 4 profilati di lunghezza $600mm$, e ulteriori tredici di $300mm$. L'assemblaggio finale è visibile in Figura 3.1b.



(a) Sezione del profilato di alluminio [9] (b) Struttura assemblata

Figura 3.1: Profilato di alluminio e struttura assemblata

La traslazione del free-flyer è ottenuta mediante l'uso di otto thrusters (Figura 3.2a), che consistono in ugelli semplicemente convergenti con un diametro critico di $1mm$. Nella parte posteriore sono equipaggiati con degli agganci rapidi per l'ingresso dei tubi del sistema pneumatico del diametro di $6mm$. Lo stesso aggancio è presente nei quattro cuscinetti ad aria utilizzati per la simulazione della microgravità Figura 3.2b.



(a) Thruster (b) Cuscinetti ad aria [10]

Figura 3.2: Thruster e cuscinetti ad aria

L'alimentazione di thrusters e cuscinetti è garantita per mezzo di un serbatoio di aria compressa con capacità pari a 1 L e pressione massima di esercizio pari a 20 MPa. Per garantire una pressione di esercizio adeguata sono implementati due regolatori di pressione e cinque valvole elettromagnetiche: quattro comandate tramite segnale PWM per la gestione dei thruster, e una per il controllo dei quattro cuscinetti ad aria.



(a) Valvola elettromagnetica [11] (b) Serbatoio d'aria compressa [12]

Figura 3.3: Componenti del sistema pneumatico

Il moto rotazionale è ottenuto tramite una reaction wheel che è composta da un motore elettrico brushless, il suo driver di azionamento e un disco cilindrico di 400 g con la funzione di massa inerziale. L'assieme di questi componenti è il responsabile della generazione della coppia necessaria alla rotazione del free-flyer attorno al proprio asse verticale, sfruttando il principio della conservazione del momento angolare. Il tutto viene alimentato tramite due batterie NiCd da 12 V e 0.65 Ah [13].



(a) Reaction wheel (b) Motore brushless [14] (c) Driver del motore [15]

Figura 3.4: Componenti del sistema reaction wheel

A bordo sono presenti due unità di calcolo: un microcontrollore Arduino Nano ESP32 per l'interfaccia fisica con il sensore IMU e gli attuatori; e un PC Linux per l'esecuzione degli algoritmi di controllo e l'acquisizione dei dati della camera.

L'acquisizione dei dati di accelerazione e velocità angolare è affidata al sensore IMU MPU9250, che integra un accelerometro e un giroscopio triassiali. Ulteriori dettagli sull'implementazione sono discussi nel Capitolo 6. Il componente è illustrato in Figura 3.5.

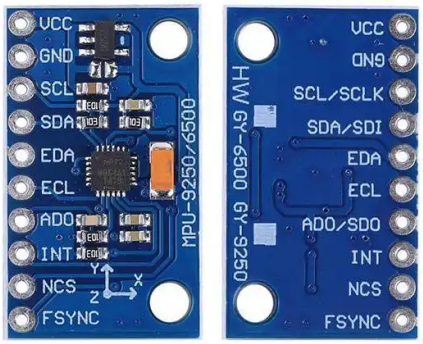


Figura 3.5: IMU MPU9250

3.1.1 Implementazione del sistema di visione artificiale

Come anticipato in precedenza, viene implementato un sistema di visione artificiale basato su camera stereoscopica Intel RealSense D435. La camera RealSense è un sensore di profondità della famiglia D400, progettato per fornire in tempo reale una stream IR, RGB e depth mediante un'architettura stereo, assistita da un proiettore infrarosso.

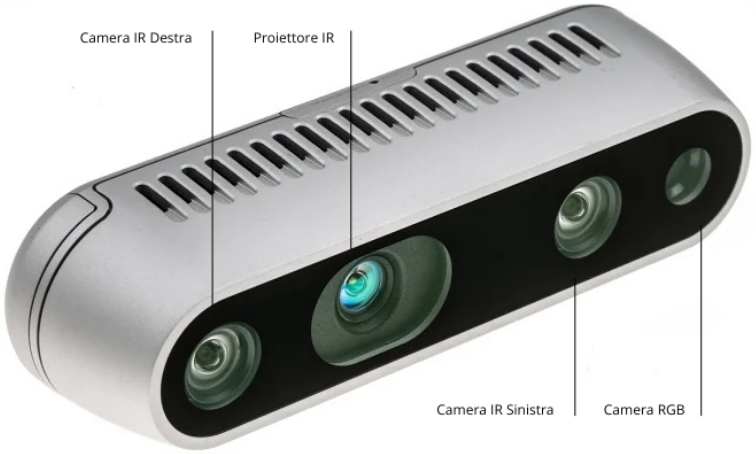


Figura 3.6: Camera Intel RealSense D435 [1]

Il modulo integra due sensori imagers IR wide, un sensore RGB e un proiettore infrarosso, consentendo la stima della profondità anche in presenza di superfici poco strutturate. I sensori imager IR, presentano un global shutter che permette una modalità di acquisizione in cui tutti i pixel del sensore vengono letti simultaneamente, evitando la scansione riga per riga come nel rolling shutter. Tale caratteristica riduce le distorsioni da movimento e risulta particolarmente vantaggiosa in contesti con rapidi movimenti della camera o della scena, nonché in applicazioni di robotica e visione artificiale in cui è richiesta un'elevata accuratezza geometrica [1].

Caratteristica	**Valore**
Risoluzione Sensori stereo	fino a 1280x800
Risoluzione Sensore RGB	fino a 1920x1080
Field of view stereo	91.2° H, 65.5° V, 100.6° D
Field of view RGB	69.4° H, 42.5° V, 77° D
IR Field of projection	100.4° H, 69° V, 110.4° D
Frame Rate	30 fps, full stream
Dimensioni	90 × 25 × 25 mm
Massa	72 g

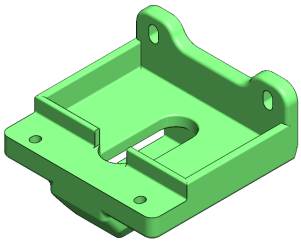
Tabella 3.1: Principali caratteristiche della Intel RealSense D435

Nel progetto AirSat, la RealSense D435 viene impiegata nel sottosistema di percezione per l'acquisizione delle immagini del marker ArUco e per la stima della posa del target, contribuendo alla catena di navigazione, stima e controllo del sistema che verrà approfondita più avanti.

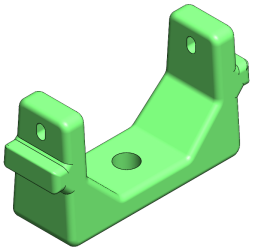
3.2 Design completo di Airsat

Illustrati i componenti che costituiscono l'AIRSAT, si procede ad una breve illustrazione del modello CAD tridimensionale. La presenza dei componenti precedentemente elencati e la necessità di contenere la massa hanno richiesto la progettazione di supporti dedicati in grado di ospitare in modo integrato e robusto l'intera strumentazione. Per realizzare tali elementi si è ricorso alla produzione additiva con stampa in 3D. Tra gli esempi significativi si citano gli alloggiamenti del motore e della reaction wheel, la cui ubicazione deve essere il più vicino possibile all'asse centrale del free-flyer, per preservare il corretto moto rotazionale. Analogamente, il serbatoio dell'aria compressa, essendo il componente più pesante dell'insieme, deve essere allineato rispetto allo stesso asse per minimizzare gli effetti di sbilanciamento. Ulteriori esempi sono i supporti per le valvole elettromagnetiche e quello per l'alloggiamento del circuito elettrico.

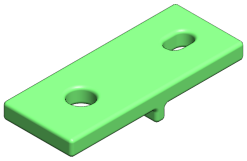
Con l'obiettivo di ottenere più libertà di manovra sulla testbench accorciando il passo tra i cuscinetti, è stato aggiornato il sistema di supporti rispetto alla precedente versione, incorporando l'alloggiamento delle batterie ove necessario (Fig. 3.7a - 3.7b). Inoltre, si è realizzato il supporto dedicato alla camera. Le immagini dei componenti sono riportate in Figura 3.7.



(a) Supporto con alloggiamento batteria



(b) Supporto cuscinetto semplice



(c) Supporto camera

Figura 3.7: Supporti aggiornati

Combinando i modelli CAD tridimensionali dei componenti commerciali con quelli delle parti progettate su misura, si ottiene l'assemblaggio finale del FSS, visibile in Figura 3.8.

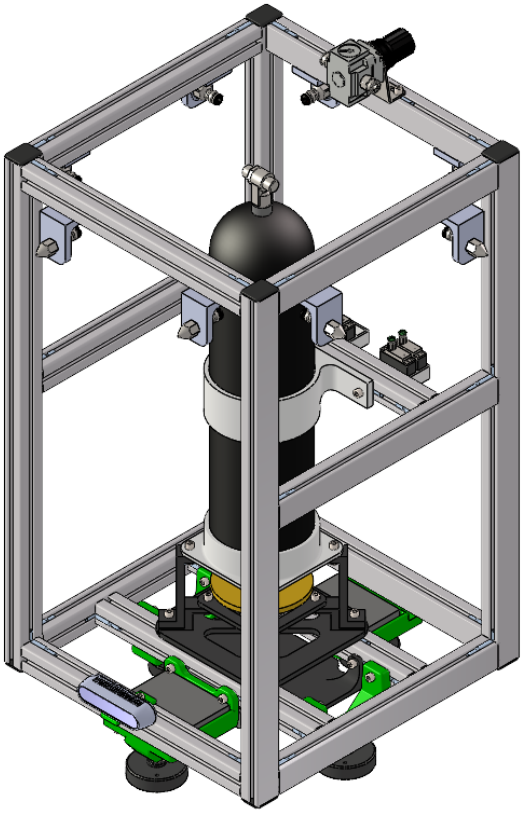


Figura 3.8: Modello CAD piattaforma AIRSAT

Capitolo 4

Aggiornamento del Digital Twin

Il digital twin è la rappresentazione virtuale di un sistema fisico, e ne riproduce il comportamento dinamico, ne supporta lo sviluppo attraverso la simulazione di test e consente attività di validazione in un ambiente controllato e ripetibile. Nel caso del prototipo Airsat, il modello virtuale costituisce già uno strumento consolidato per le strategie di GNC, fornendo un supporto essenziale nella fase di sviluppo del sistema. Per affinare ulteriormente il modello virtuale, in questo lavoro di tesi, viene aggiornato il comparto sensoristico, modellando i sensori in modo realistico, e integrando un sistema di Sensor Fusion come il Kalman Filter. Infine, vengono analizzate le prestazioni di modellazione dei sensori e del filtro di Kalman implementato. Segue una breve spiegazione del modello attuale del digital twin per fornire una migliore comprensione delle evoluzioni successive apportate nello svolgimento di questo studio.

4.1 Panoramica del Digital Twin

Il modello del digital twin è sviluppato sul software MATLAB Simulink, con l'utilizzo dell'estensione Simscape Multibody. Come anticipato l'obiettivo del modello è replicare la dinamica del sistema fisico reale, per fare ciò viene strutturato in blocchi, ognuno contentente la replica del corrispettivo sottosistema reale. Lo schema è rappresentato in Figura 4.1.

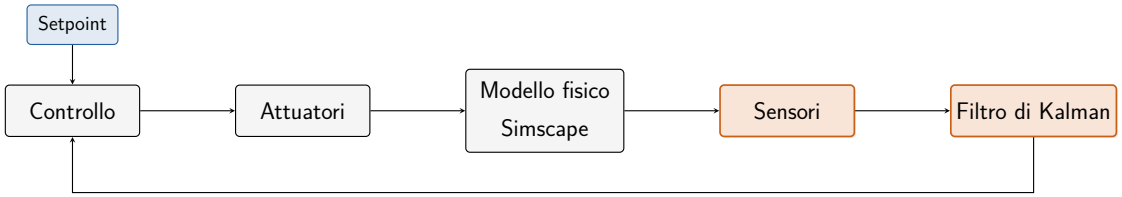


Figura 4.1: Schema a blocchi del digital twin in MATLAB Simulink

Il sottosistema Simscape costituisce la riproduzione del sistema fisico, al suo interno è presente il modello del corpo rigido del free-flyer, schematizzato mediante un parallelepipedo vincolato con un giunto planare sul piano XY, le cui proprietà inerziali, dimensionali e di massa sono allineate a quelle del CAD del prototipo. Allo stesso modo la reaction wheel è modellata come disco rigido, coassiale al corpo principale e vincolato mediante un giunto rotoidale, in modo da preservare il suo contributo inerziale alla dinamica rotazionale. L'obiettivo di questa modellazione semplificata è la riduzione del costo computazionale pur mantenendo le caratteristiche dinamiche originali. Sui rispettivi giunti verranno successivamente applicate le forze e le coppie degli azionamenti. L'attuazione è divisa in due parti: traslazione e rotazione. La traslazione è generata dai thruster pneumatici, modellati come ugelli convergenti usando le equazioni del manuale NASA [16]; azionati dalle valvole elettromagnetiche di cui è stata omessa la dinamica se non per i limiti reali di apertura delle valvole. Il comando dei thruster passa attraverso un blocco PWM che modula la forza proporzionalmente con il segnale di comando. La rotazione è affidata alla reaction wheel tramite il modello di un motore DC, il comando di coppia viene prima diviso per l'inerzia della ruota e poi integrato per ottenere la velocità di rotazione che viene convertita in comando di tensione. Infine, per garantire un inseguimento accurato del setpoint e la riduzione dell'errore a regime, il sottosistema di controllo implementa una strategia LQI, che in base all'errore calcolato e ai guadagni impostati, restituisce le componenti della forza e della coppia per gli attuatori.

Nel proseguo del capitolo verranno approfonditi i blocchi evidenziati in rosso in Figura 4.1, corrispondenti ai sottosistemi dei sensori e della sensor Fusion con filtro di Kalman.

4.2 Blocco Sensori

Introdurre un modello realistico dei sensori virtuali consente di disporre di un riferimento per la validazione preliminare degli algoritmi di sensor fusion. Per raggiungere questo obiettivo, è necessario verificare la correlazione dei contenuti in frequenza delle rilevazioni dei sensori reali con quelli nel digital twin. Inoltre, avendo a disposizione i riferimenti di misura ideali delle grandezze fisiche all'interno dell'ambiente di simulazione virtuale, è possibile sfruttarle come benchmark e confrontarle con i risultati ottenuti dagli algoritmi di sensor fusion per valutarne le performance.

Nella prossima sezione si procede alla modellazione del blocco sensori con l'ipotesi unificata di sensori affetti da rumore bianco. Successivamente si procede alla verifica

della correlazione con i corrispettivi sensori reali attraverso l'analisi spettrale del rumore.

4.2.1 IMU

Il sensore IMU MPU9250 è composto da un accelerometro e un giroscopio, perciò si procede alla modellazione di entrambi all'interno del digital twin mediante il blocco IMU all'interno di MATLAB Simulink, grazie al quale con l'individuazione di parametri caratteristici è possibile riprodurre le dinamica di acquisizione di sensori inerziali come quelli di interesse. In particolare, il blocco richiede alcuni parametri disponibili dal datasheet o facilmente individuabili attraverso rilevazioni statiche dei sensori reali. I parametri richiesti, che verranno in seguito definiti per ognuno dei modelli dei due sensori, sono:

- **Risoluzione**: (m/s^2)/LSB per l'accelerometro e (rad/s)/LSB per il giroscopio;
- **Constant bias offset**: m/s^2 per l'accelerometro e rad/s per il giroscopio;
- **Max range**: m/s^2 per l'accelerometro e rad/s per il giroscopio;
- **Velocity/angle random walk**: $(m/s^2)/\sqrt{Hz}$ per l'accelerometro e $(rad/s)/\sqrt{Hz}$ per il giroscopio.

La risoluzione rappresenta la minima variazione apprezzabile dal sensore, espressa per LSB ovvero Less Significant Bit. Il max range indica l'intervallo massimo di misura del sensore, quindi il limite oltre il quale il sensore non è in grado di misurare. Il velocity/angle random walk esprime come il contributo del rumore bianco si accumula nel tempo, descrivendo quanto il rumore tende a propagarsi quando viene integrato; risulta un parametro fondamentale per considerare la degradazione della stima nella navigazione inerziale del modello. Infine, il constant bias offset rappresenta un errore sistematico considerabile costante, cioè uno scostamento medio della misura rispetto al valore reale, questo parametro non è fornito esplicitamente nel datasheet, perciò deve essere stimato sperimentalmente tramite calibrazione. Di seguito sono riportati i valori per l'accelerometro, in Tabella 4.1, e per il giroscopio, in Tabella 4.2 [17] [18].

Variabile	Valore	Unità di misura	Origine
Resolution	$[6.23 \cdot 10^{-3}, 6.23 \cdot 10^{-3}]$	$(m/s^2)/LSB$	datasheet
Constant bias offset	$[3.37 \cdot 10^{-2}, 3.17 \cdot 10^{-2}]$	m/s^2	sperimentale
Max range	$[\pm 19.60, \pm 19.6]$	m/s^2	datasheet
Velocity random walk	$[2.94 \cdot 10^{-3}, 2.94 \cdot 10^{-3}]$	$(m/s^2)/\sqrt{Hz}$	datasheet

Tabella 4.1: Parametri caratteristici del modello dell'accelerometro lungo gli assi x e y

Variabile	Valore	Unità di misura	Origine
Resolution	$1.33 \cdot 10^{-4}$	$(rad/s)/LSB$	datasheet
Constant bias offset	$5.6 \cdot 10^{-5}$	rad/s	sperimentale
Max range	± 4.36	rad/s	datasheet
Angle random walk	$1.74 \cdot 10^{-4}$	$(rad/s)/\sqrt{Hz}$	datasheet

Tabella 4.2: Parametri caratteristici del modello del giroscopio attorno all'asse z

Per la rilevazione dei parametri del constant bias offset sperimentale, sono state effettuate 1000 misurazioni statiche sul piano per ciascun sensore e ne è stato calcolato l'RMSE rispetto al valore nullo atteso.

4.2.2 Camera

Poiché non è presente un blocco camera adeguato per la rappresentazione di tale sensore nel contesto semplificato di questo simulatore, si procede con una modellazione di un segnale statisticamente equivalente a quello reale, caratterizzato da un rumore con varianza e tempo di campionamento noti, attraverso l'uso del blocco Band-Limited White Noise di Simulink. Questo blocco richiede due parametri, la Noise power e il Sample time come riporta la documentazione ufficiale del blocco [19].

Il blocco Band-Limited White Noise, a ogni passo di campionamento, genera un numero casuale gaussiano a media nulla che viene mantenuto costante per un intervallo pari al tempo di campionamento t_c, coincidente con il parametro Sample time del blocco. Il parametro Noise power q è legato alla varianza dalla relazione:

$$\sigma^2 = \frac{q}{t_c}, \tag{4.1}$$

dove σ^2 è la varianza del segnale generato in uscita e t_c è il tempo di campionamento. Per replicare il comportamento statistico di un sensore reale di cui sono note la varianza

$\tilde{\sigma}^2$ e il tempo di campionamento $\tilde{t_c}$, dunque è sufficiente invertire l'Equazione (4.1) e ricavare il valore da inserire nel parametro *Noise Power* come:

$$q = \tilde{\sigma}^2 \cdot \tilde{t_c}. \tag{4.2}$$

In questo modo il generatore restituisce, a ogni istante di campionamento, un rumore gaussiano $n_k \sim \mathcal{N}(0, \sigma_s^2)$ statisticamente equivalente al rumore del sensore fisico. Il segnale del sensore virtuale può quindi essere espresso come

$$y_k = x_k + n_k, \qquad n_k \sim \mathcal{N}(0, \sigma_s^2), \tag{4.3}$$

dove x_k è il valore vero della grandezza misurata e y_k è la misura simulata affetta da rumore, analoga alla formulazione della misura rumorosa impiegata nei filtri di stima quali il filtro di Kalman. Nel modello virtuale la misura che rappresenta il valore vero è ottenuta tramite un Trasform sensor collegato direttamente al Simscape del modello fisico. Il valore di varianza è acquisito tramite una rilevazione di 1000 misurazioni di un obiettivo fisso e successivamente viene utilizzato per il calcolo dei parametri del blocco. Nella Tabella 4.3 sono riportati i valori dei parametri.

Variabile	Simbolo	Valore	Unità di misura	Origine
Varianza	σ^2	$3.43 \cdot 10^{-3}$	$(\text{rad/s})^2$	sperimentale
Tempo di campionamento	t_c	0.035	s	sperimentale
Noise Power	q	$1.22 \cdot 10^{-3}$	$(\text{rad/s})^2/Hz$	Formula (4.2)

Tabella 4.3: Parametri del modello di camera virtuale per la rotazione

Variabile	Simbolo	Valore	Unità di misura	Origine
Varianza	σ^2	$5.19 \cdot 10^{-7}$	$(\text{rad/s})^2$	sperimentale
Tempo di campionamento	t_c	0.035	s	sperimentale
Noise Power	q	$1.82 \cdot 10^{-8}$	$(\text{rad/s})^2/Hz$	Formula (4.2)

Tabella 4.4: Parametri del modello di camera virtuale per la traslazione lungo x

Variabile	Simbolo	Valore	Unità di misura	Origine
Varianza	σ^2	$3.63 \cdot 10^{-5}$	$(\text{rad/s})^2$	sperimentale
Tempo di campionamento	t_c	0.035	s	datasheet camera
Noise Power	q	$1.27 \cdot 10^{-6}$	$(\text{rad/s})^2/Hz$	Formula (4.2)

Tabella 4.5: Parametri del modello di camera virtuale per la traslazione lungo y

A differenza dei sensori inerziali, il cui segnale deve essere integrato nel tempo per fornire posizione o velocità e che, pertanto, accumulano un errore di drift sistematico dovuto alla propagazione del rumore, un sensore di visione artificiale fornisce una misura diretta e assoluta della grandezza osservata a ogni istante di acquisizione, senza alcuna integrazione numerica dello stato. Ne consegue che il modello di rumore associato non necessita dell'introduzione di ulteriori termini quali il random walk tipicamente richiesto per caratterizzare accelerometri e giroscopi. Il semplice rumore bianco a banda limitata descritto dalle equazioni (4.1) e (4.3) è quindi sufficiente a rappresentare l'incertezza di misura di un sensore di visione artificiale, il cui errore non presenta componenti correlate nel tempo.

Il modello proposto, basato unicamente sul blocco Band-Limited White Noise, costituisce una rappresentazione volutamente semplificata del rumore del sensore di visione, poiché non tiene conto degli effetti sistematici introdotti dalla distorsione ottica di acquisizione della camera reale. Analogamente, non è considerato l'effetto dello shutter, in particolare del rolling shutter che introduce una distorsione temporale dell'immagine dovuta all'acquisizione sequenziale delle righe del sensore durante il moto relativo tra camera e scena. L'esclusione di questi fenomeni è accettabile in una fase preliminare di validazione dell'algoritmo di stima, ma va tenuta in considerazione qualora si intenda estendere il modello a condizioni operative ancora più realistiche.

4.2.3 Analisi dei contenuti in frequenza dei sensori

Come anticipato a inizio capitolo, si procede a verificare la correlazione tra le acquisizioni dei sensori reali e quelle dei sensori simulati, per stabilire se siano riprodotti in modo fedele all'interno del digital twin. Per questo scopo si effettua l'analisi spettrale dei due segnali.

L'analisi del contenuto spettrale di un segnale consiste nella sua scomposizione nelle diverse frequenze per identificarne bande dominanti o caratteristiche di rumore; questa scomposizione si può ottenere tramite la Trasformata di Fourier e fornisce il grafico della densità spettrale di potenza (PSD). Per un processo stocastico stazionario

a media nulla, questa potenza coincide con la varianza del segnale, perciò il grafico della PSD descrive come la varianza del segnale è distribuita sulle frequenze. Se lo spettro del segnale virtuale produce lo stesso andamento osservato sullo spettro del sensore reale, allora il modello virtuale sta replicando correttamente rumori e dinamiche fisiche del sistema nelle medesime bande di frequenza.

Per verificare la correlazione tra reale e virtuale si è applicato il metodo della stima della PSD di Welch, tramite il comando matlab `pwelch`. Al fine di operare un confronto rigoroso, dalle acquisizioni viene sottratta la propria media mobile, in modo da isolare completamente il contenuto di rumore ad alta frequenza dall'offset delle oscillazioni delle misurazioni dell'angolo ψ; oltre a effettuare l'acquisizione con la stessa frequenza di campionamento. L'accordo nel dominio del tempo e delle frequenze garantisce che il digital twin riproduca fedelmente il comportamento del sensore reale. Seguono i grafici di confronto della PSD dei diversi sensori.

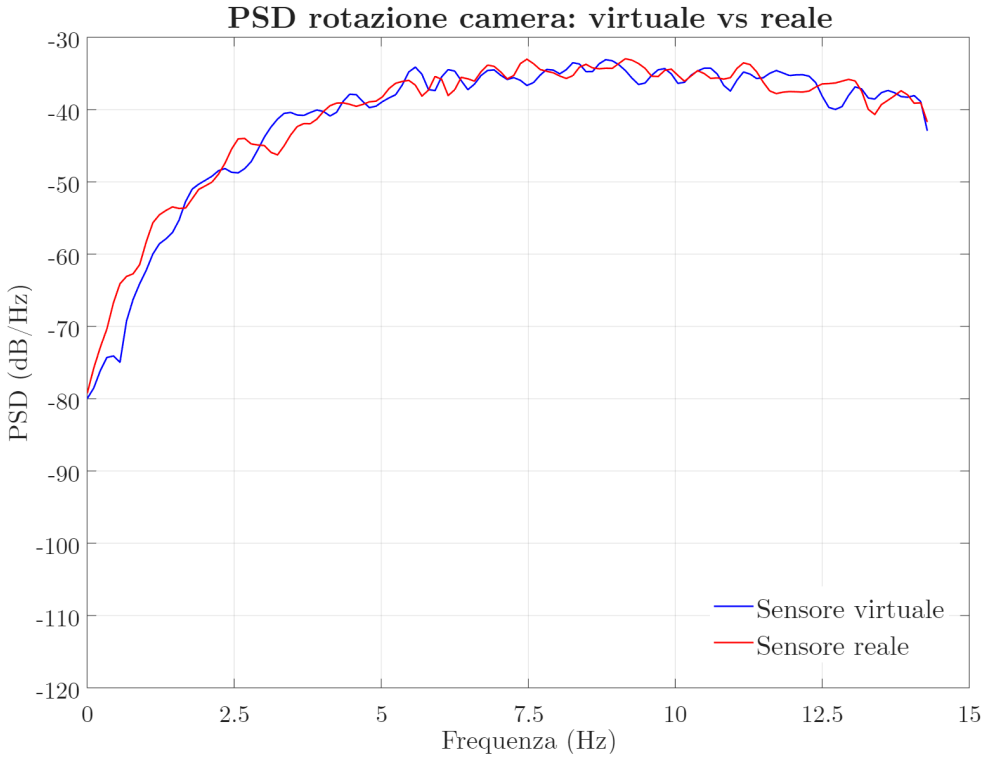


Figura 4.2: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per l'angolo di assetto

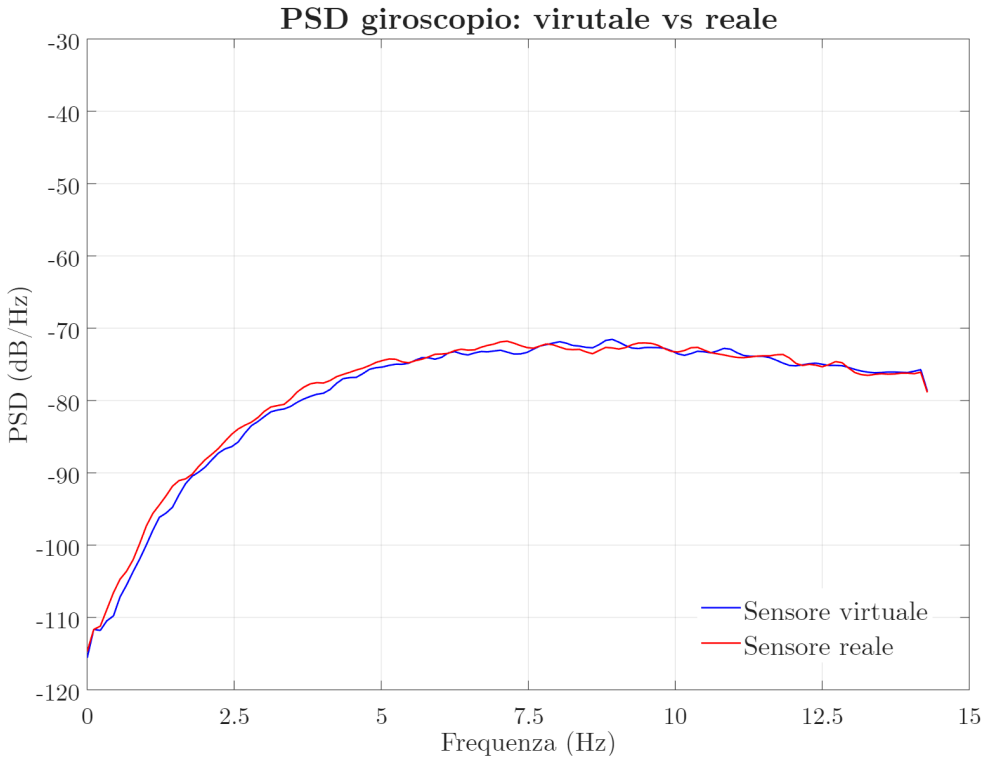


Figura 4.3: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato del giroscopio per la velocità angolare

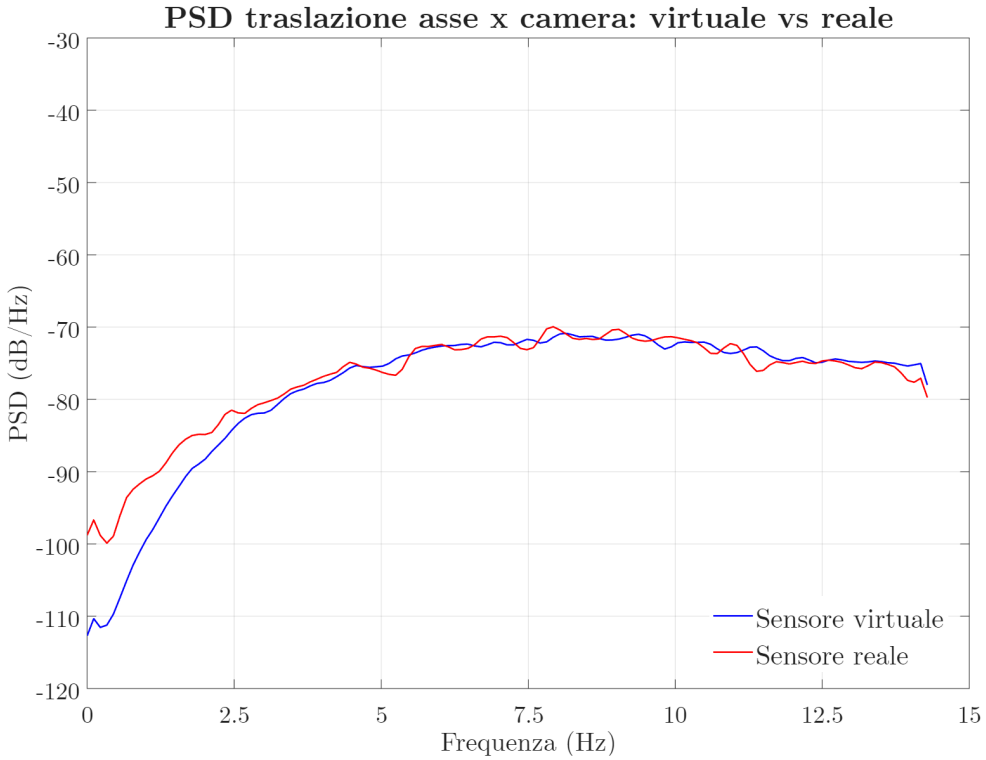


Figura 4.4: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per la posizione x

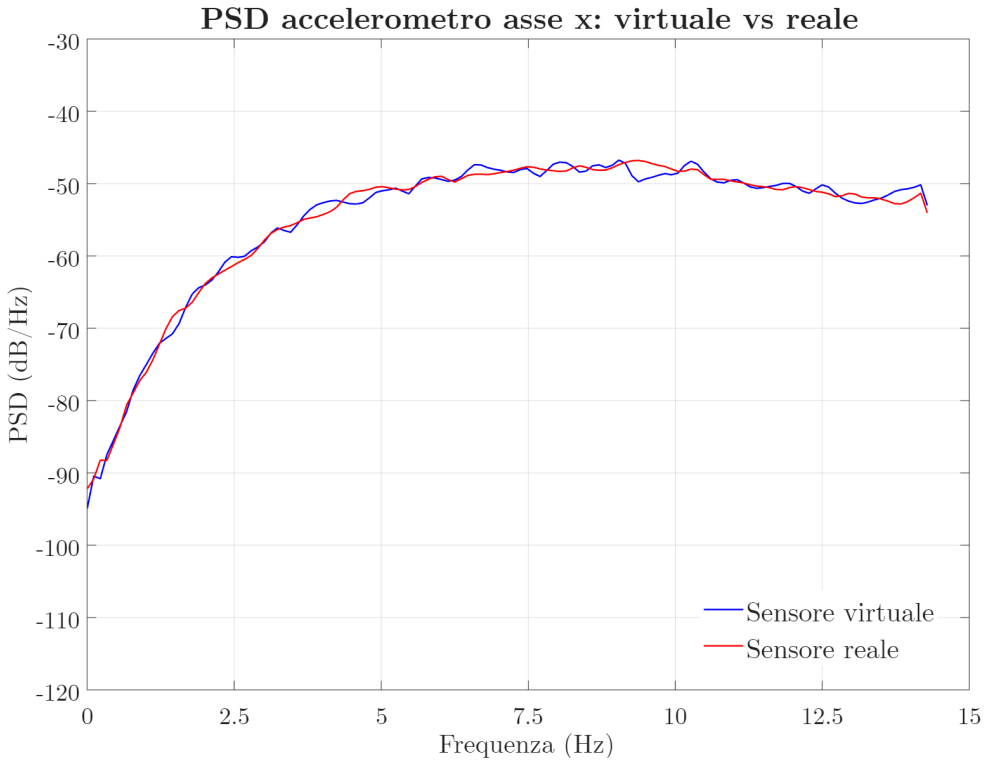


Figura 4.5: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato dell'accelerometro lungo l'asse x

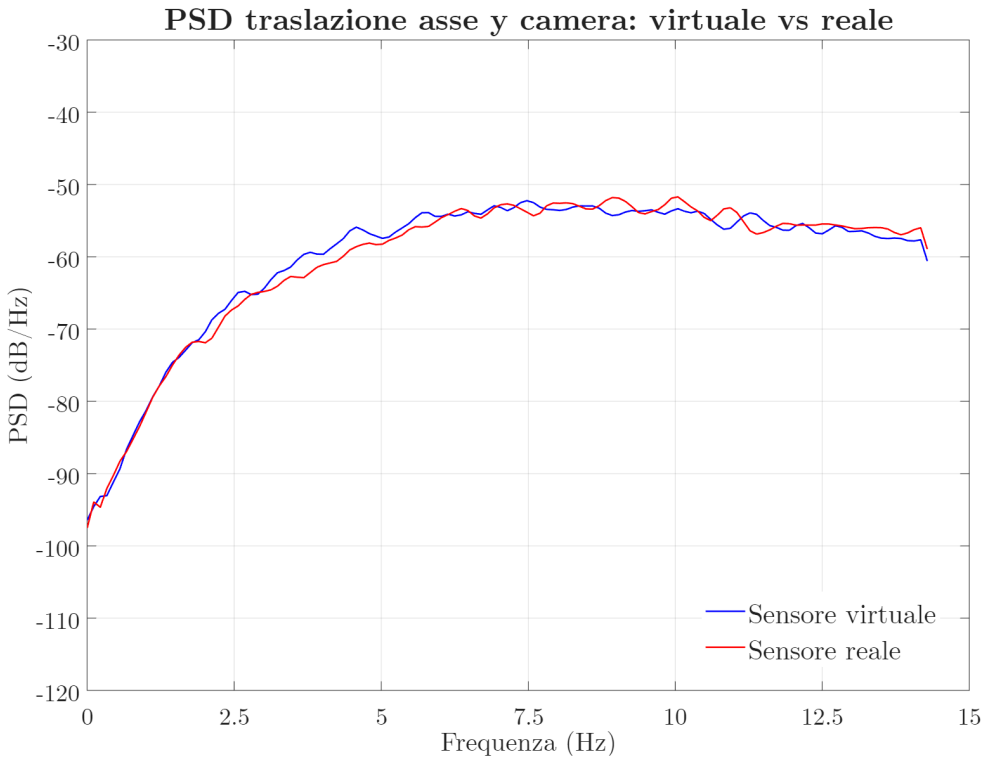


Figura 4.6: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato della camera per la posizione y

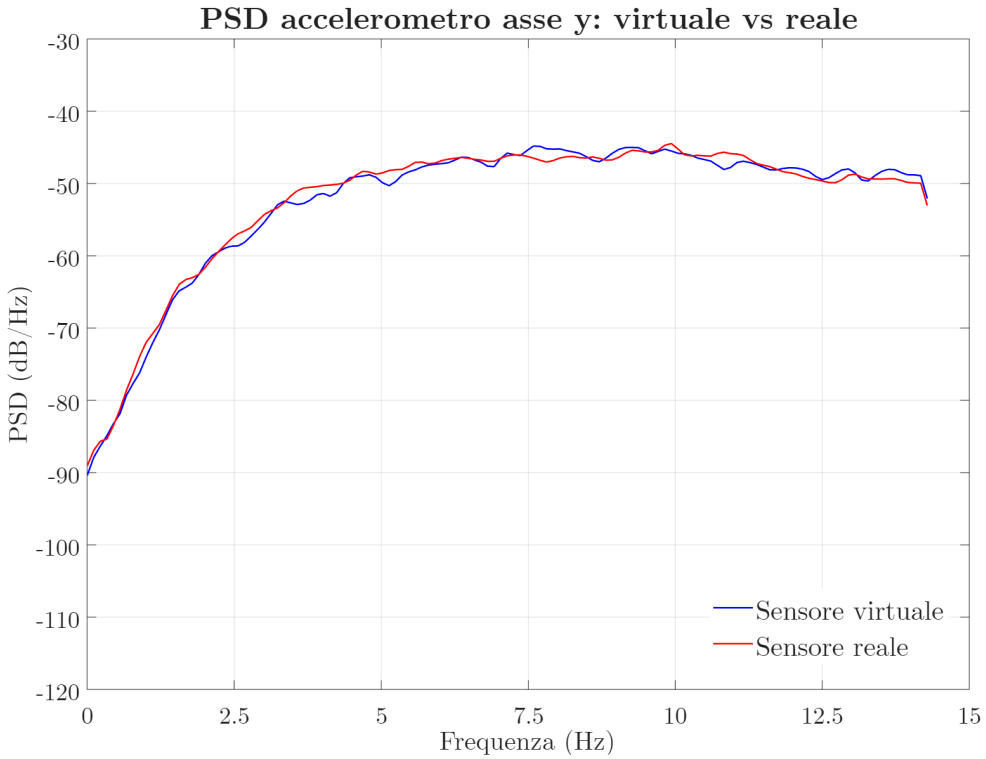


Figura 4.7: Confronto della densità spettrale di potenza tra il segnale reale e il modello simulato dell'accelerometro lungo l'asse y

Si osservi come gli andamenti nei grafici si appiattiscano verso le alte frequenze tipiche del rumore di questi sensori, confermando l'ipotesi iniziale di rumore bianco, che per definizione presenta una distribuzione in frequenza della PSD costante [20, pp. 73, 82].

4.3 Blocco Filtro di Kalman

Riguardo al blocco del filtro di Kalman si è scelto di non utilizzare i blocchi presenti nell'ambiente MATLAB Simulink per avere maggiore controllo sul codice e adattare il filtro al caso specifico dell'Airsat.

Per simulare la discretizzazione del filtro di Kalman si sfrutta un *enabled subsystem* collegato ad una matlab function che all'arrivo di ogni nuovo campione dai sensori attiva il subsystem contentente il filtro di Kalman, replicando la dinamica di attivazione presente nel software implementato sul free-flyer fisico, descritta nel Capitolo 6. Il subsystem riceve in ingresso le misurazioni grezze in arrivo dai sensori virtuali $x^m, y^m, a_x^m, a_y^m, \psi^m, g_z^m$. All'interno del blocco, due MATLAB function elaborano l'algoritmo di Kalman seguendo la versione approfondita nel Capitolo 5: la prima

svolge la fase di prediction, la seconda la fase di update. Nella Tabella 4.6 sono elencati i parametri delle matrici di covarianza utilizzati nelle simulazioni.

Matrice di Rumore di Processo Q		
Simbolo	**Valore**	**Descrizione**
q_{ax}	$5.0 \cdot 10^{-4}$	Varianza del rumore dell'accelerometro su x
q_{ay}	$5.0 \cdot 10^{-4}$	Varianza del rumore dell'accelerometro su y
q_{gz}	$5.0 \cdot 10^{-7}$	Varianza del rumore del giroscopio
$q_{b_{a_x}}$	$1.0 \cdot 10^{-8}$	Varianza del bias dell'accelerometro su x
$q_{b_{a_y}}$	$1.0 \cdot 10^{-8}$	Varianza del bias dell'accelerometro su y
$q_{b_{gz}}$	$1.0 \cdot 10^{-10}$	Varianza del bias del giroscopio
Matrice di Rumore di Misura R		
Simbolo	**Valore**	**Descrizione**
r_x	$5.0 \cdot 10^{-7}$	Varianza di misura ArUco su x^m
r_y	$5.0 \cdot 10^{-5}$	Varianza di misura ArUco y^m
r_ψ	$5.0 \cdot 10^{-3}$	Varianza di misura ArUco ψ^m

Tabella 4.6: Parametri delle matrici di covarianza filtro di Kalman sul digital twin

Si osserva un'asimmetria dei parametri per quanto riguarda la covarianza del rumore di misura tra l'asse x e y, che non si riscontra nella covarianza del rumore di processo. Questo sottolinea la differenza di incertezza degli algoritmi di stima per la determinazione della posizione laterale, letta sulla superficie del sensore ottico, rispetto alla stima di profondità. Lo stesso comportamento, dunque, non si riscontra nella covarianza del rumore di processo in quanto il sensore coinvolto ha un tipo di natura differente che non introduce asimmetrie nelle rilevazioni.

4.3.1 Performance del filtro di Kalman

Per testare le performance del filtro, si effettuano diverse simulazioni con traiettorie preimpostate, per avere un riferimento di *ground truth*. Si riportano i grafici della più rappresentativa, in quanto racchiude i principali scenari operativi dell'Airsat. Le traiettorie hanno queste caratteristiche:

1. per ogni grado di libertà non sono previsti transitori eccessivamente rapidi,

2. sono previste situazioni di oscillazione intorno alla posizione di zero,
3. sono previste situazioni di inversione di andamento delle velocità e delle accelerazioni

Vengono riportati i grafici di ogni grandezza dello *state space* e successivamente valutati i RMSE rispetto al valore ideale di ground truth.

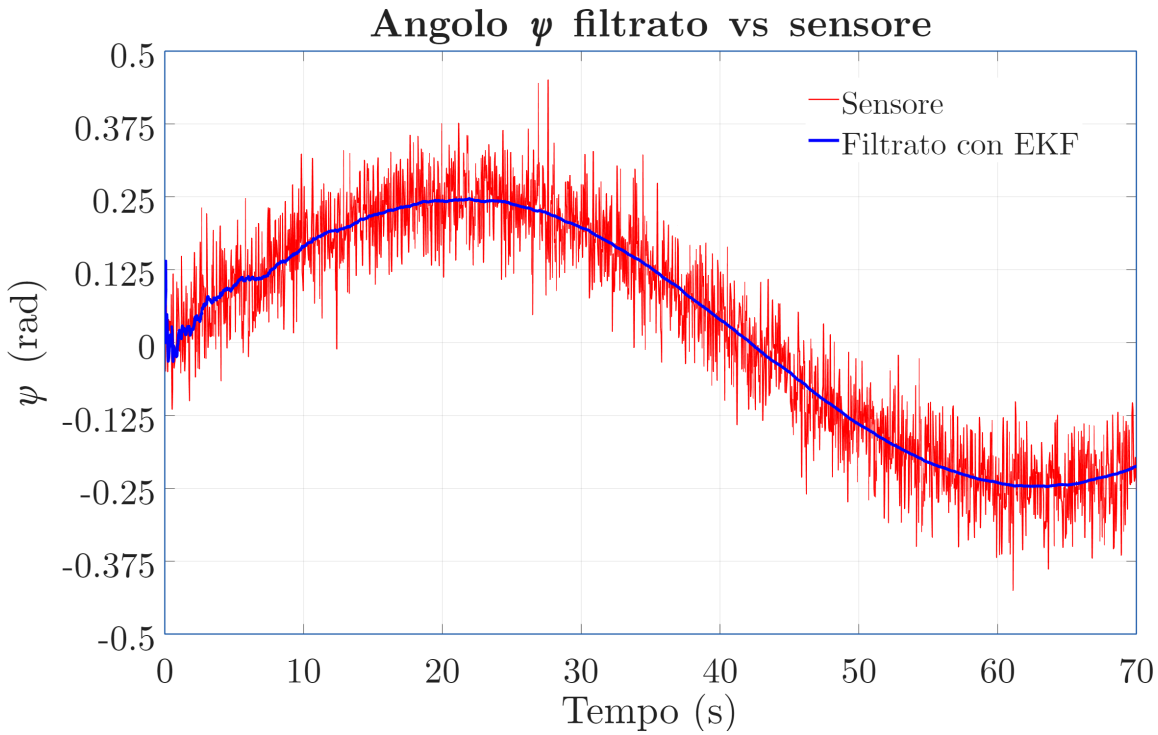


Figura 4.8: Performance del filtro di Kalman sulla stima dell'angolo di yaw

L'angolo misurato dal sensore risulta il più rumoroso, per le caratteristiche di distorsione illustrate in precedenza riguardo la modellazione del sensore camera. Come si può vedere in Figura 4.8, dopo un breve periodo di assestamento il segnale di misura filtrata riproduce l'andamento dell'angolo di assetto azzerando quasi totalmente tutto il rumore introdotto dalla camera.

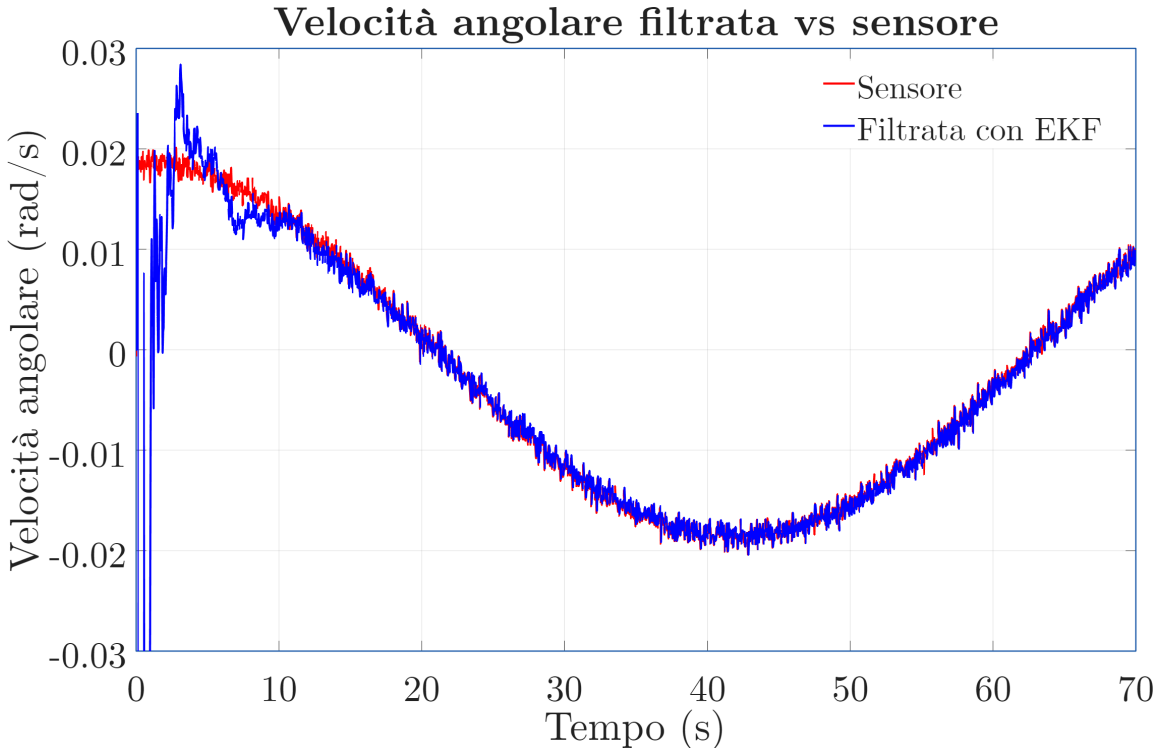


Figura 4.9: Performance del filtro di Kalman sulla stima della velocità angolare yawrate

Anche in Figura 4.9 è visibile un periodo di assestamento dopo il quale il filtro converge verso il valore di misura del giroscopio. Il fenomeno osservato durante la fase di transitorio iniziale evidenzia il verificarsi di un *overshoot*. In corrispondenza dell'inizializzazione dell'algoritmo, infatti, avviene a una fase di stabilizzazione poiché i valori di covarianza del filtro e di residuo di innovazione elevati devono ancora trovare coerenza reciproca nella stima dello stato. Di conseguenza, il guadagno di Kalman che serve a correggere il residuo di innovazione in base al valore della covarianza assume a sua volta un valore elevato e porta a una correzione rapida della stima. Il processo continua per un breve periodo fino alla convergenza tra la fase di propagazione e la fase di update. Questo comportamento è tipico di algoritmi ricorsivi come il filtro di Kalman, ed è visibile in altri esempi tra cui nel paper di Jetto et al. [21].

Nelle fasi successive i due segnali sono sovrapponibili per via del constant bias offset molto basso come visto in Tabella 4.2.

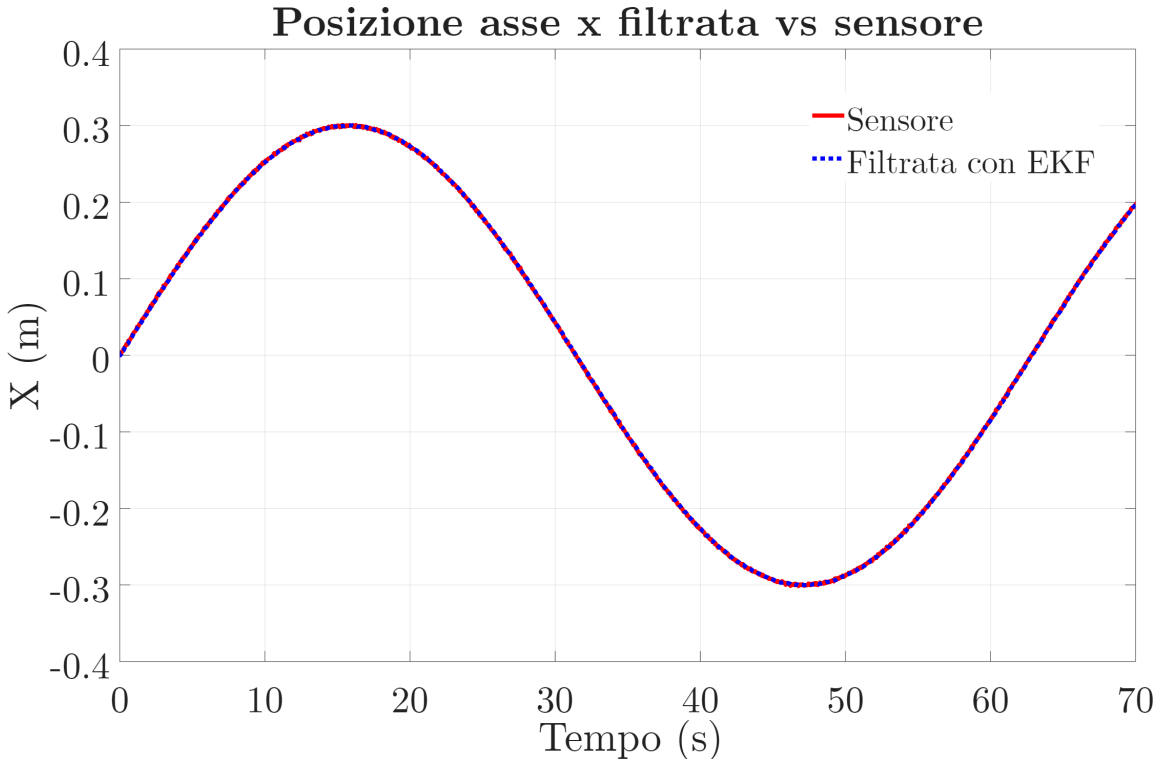


Figura 4.10: Performance del filtro di Kalman sulla stima della posizione x

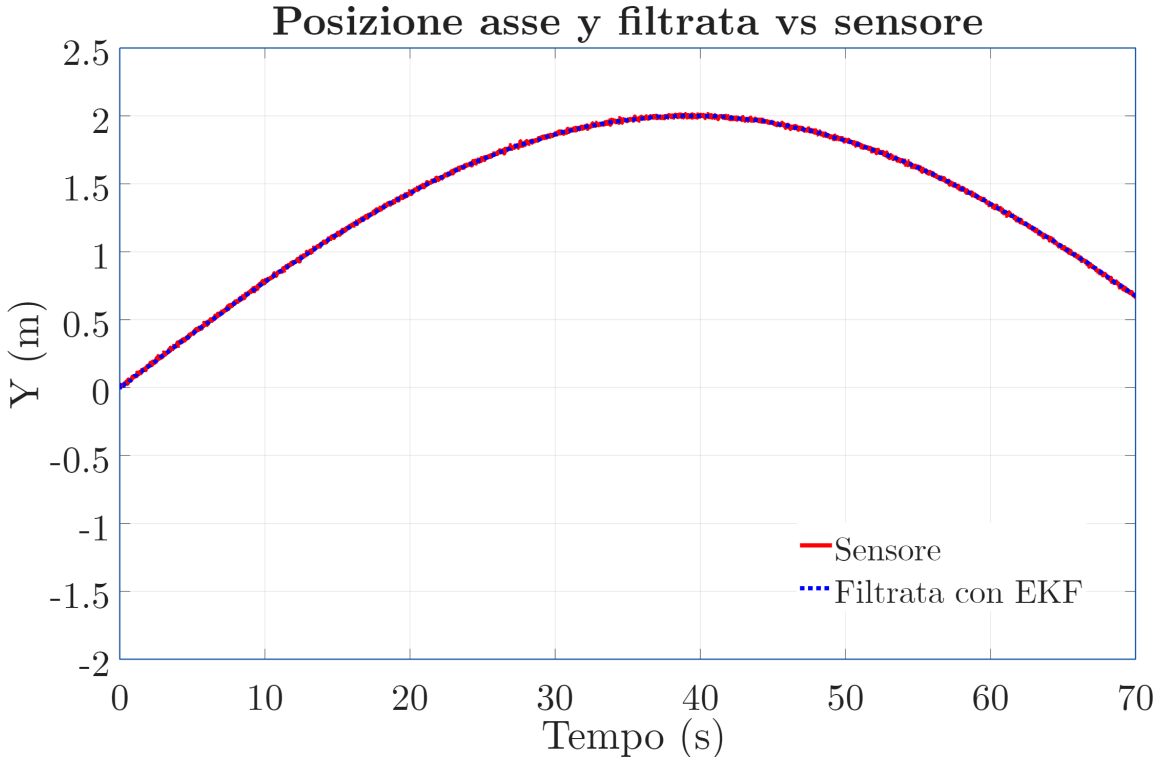


Figura 4.11: Performance del filtro di Kalman sulla stima della posizione y

Per quanto riguarda entrambe le stime della posizione, riportate nelle Figure 4.10 e 4.11, si può vedere come siano allineate con il valore di misura della camera e prevedano uno scenario di avvicinamento, allontanamento e oscillazione nel raggio di azione di un paio di metri fino a qualche decina di centimetri.

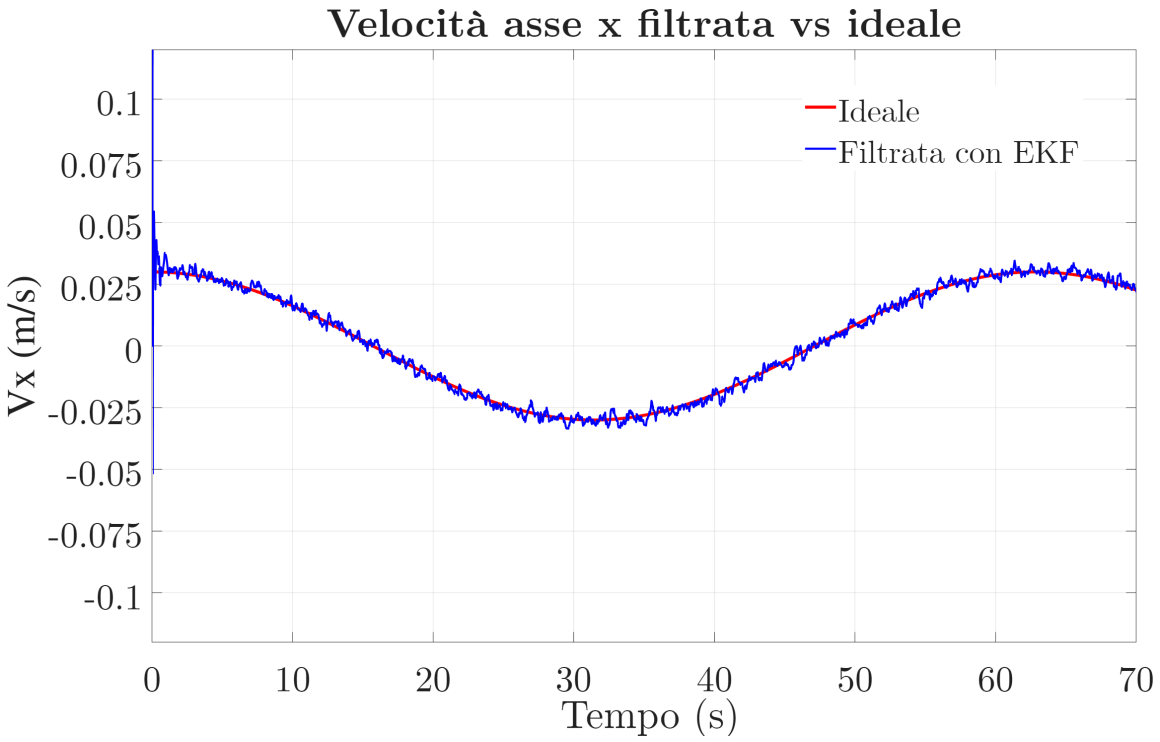


Figura 4.12: Performance del filtro di Kalman sulla stima della velocità v_x

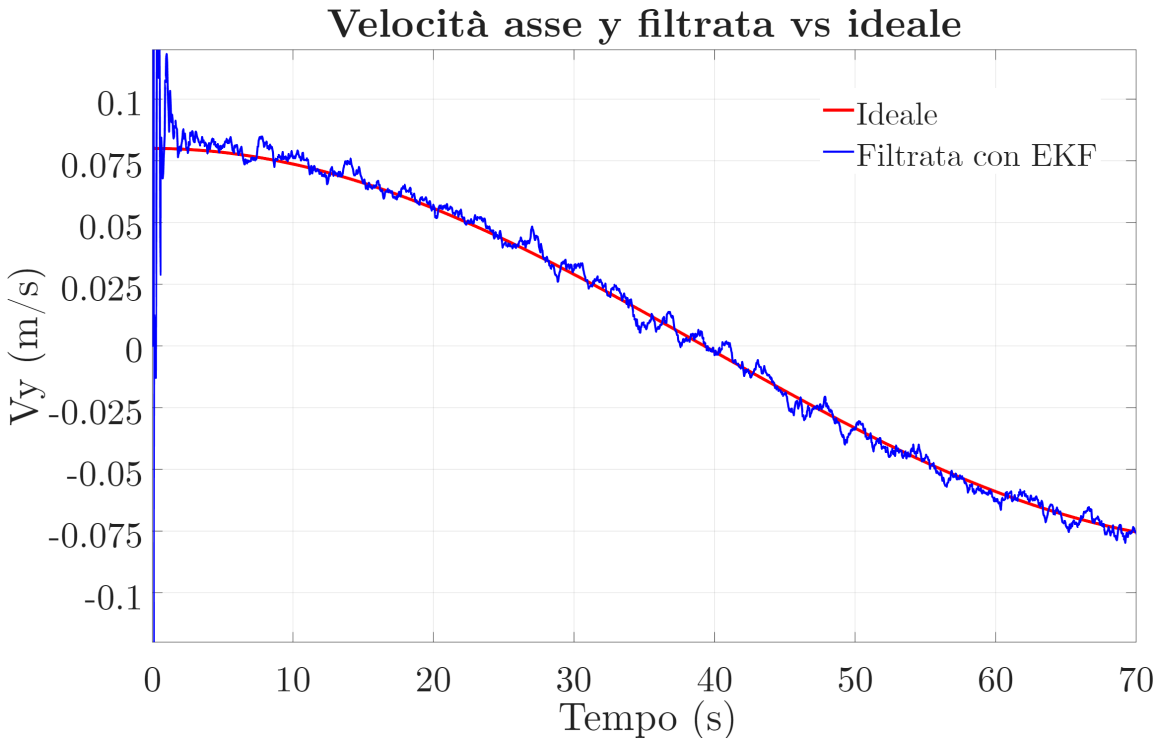


Figura 4.13: Performance del filtro di Kalman sulla stima della velocità v_y

Il caso delle velocità, riportato nelle Figure 4.12 e 4.13, è particolare in quanto non c'è nessun valore misurato direttamente che possa essere usato da confronto, motivo per il quale in questi grafici viene confrontato con la ground truth. Come si può vedere la stima risulta fedele all'andamento vero, sebbene conservi delle componenti di rumore ad alta frequenza ereditate dalla misura dall'accelerometro.

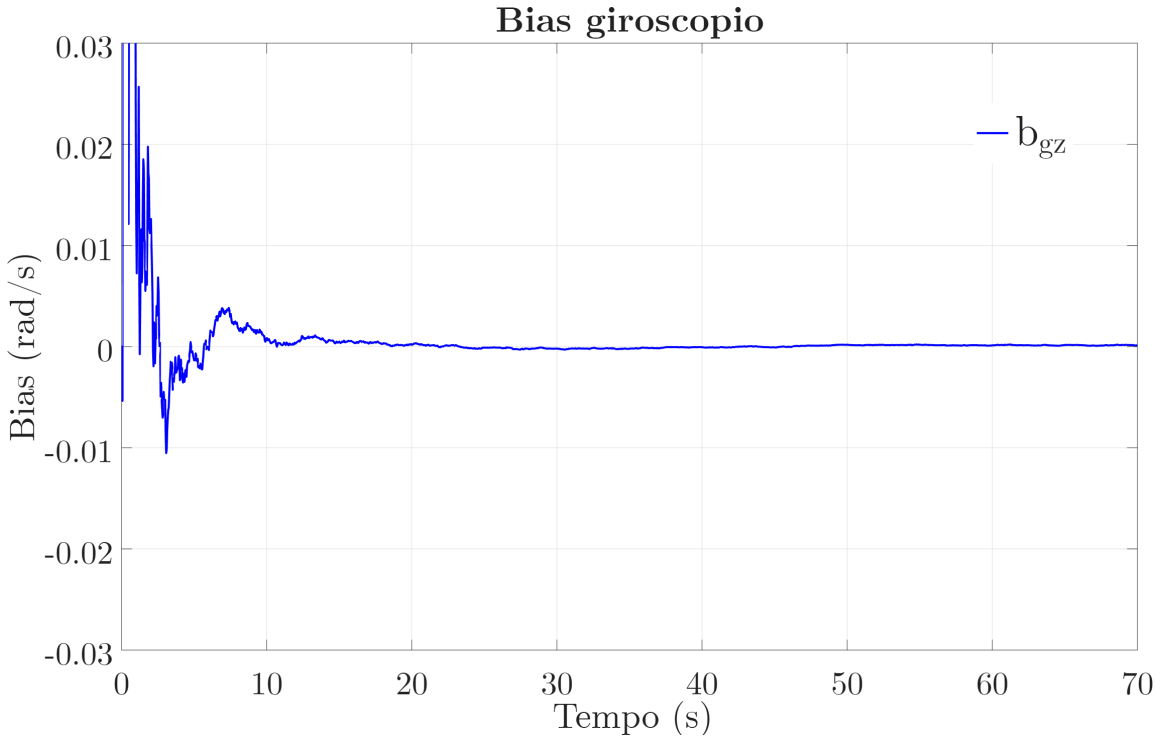


Figura 4.14: Stima dei bias dell'accelerometro del filtro di Kalman

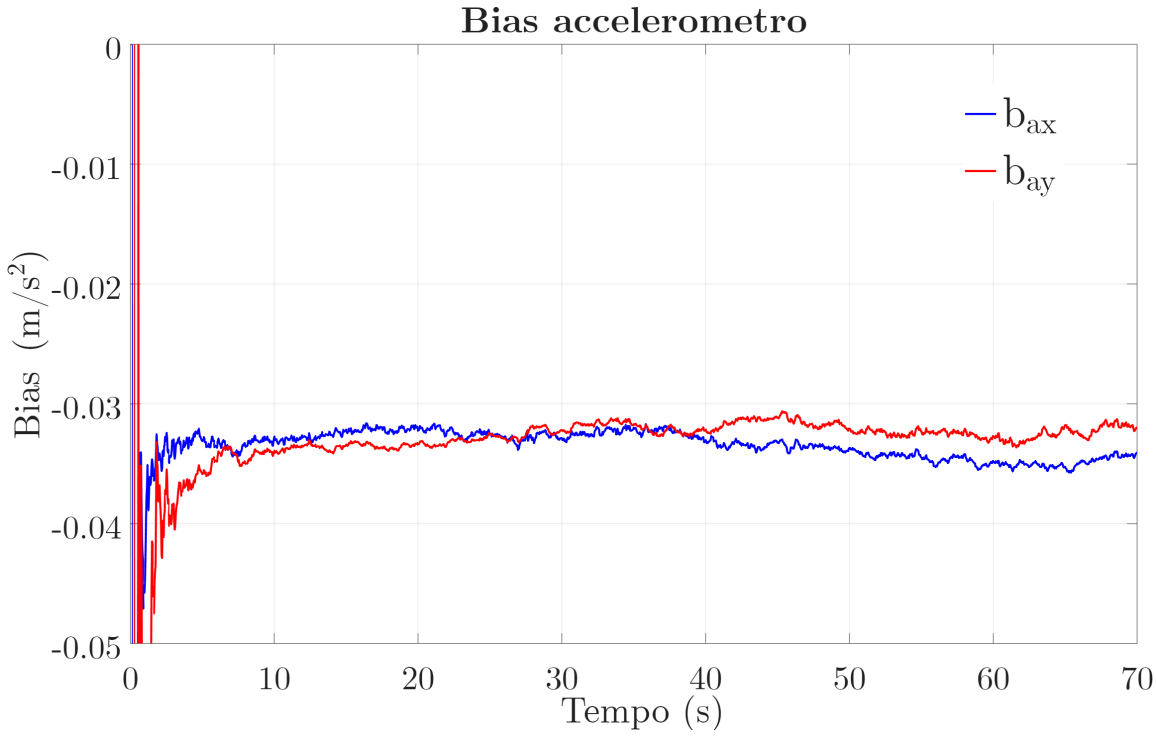


Figura 4.15: Stima del bias del giroscopio del filtro di Kalman

Si riportano in Figura 4.14 e 4.15 le ultime grandezze dello *state space* espanso di questo filtro di Kalman. Le grandezze si riferiscono alla stima del bias sui sensori

dell'unità inerziale e svolgono la funzione di una calibrazione in tempo reale. Come si può vedere, dopo un breve periodo, i bias si assestano sul valore rilevato sperimentalmente e riportato nelle Tabelle 4.2 e 4.1. In particolare, per quanto riguarda quelle dell'accelerometro intorno al valore di -0.3 e per il giroscopio con valori molto prossimi allo 0.

Infine, si calcolano i valori di RMSE delle grandezze stimate rispetto alla *ground truth* e, come si può vedere in Figura 4.16 e 4.17, tutti i valori rimangono uguali o minori rispetto al valore dei sensori grezzi, con un buon risultato anche sui valori di velocità e un ottimo risultato sull'angolo ψ.

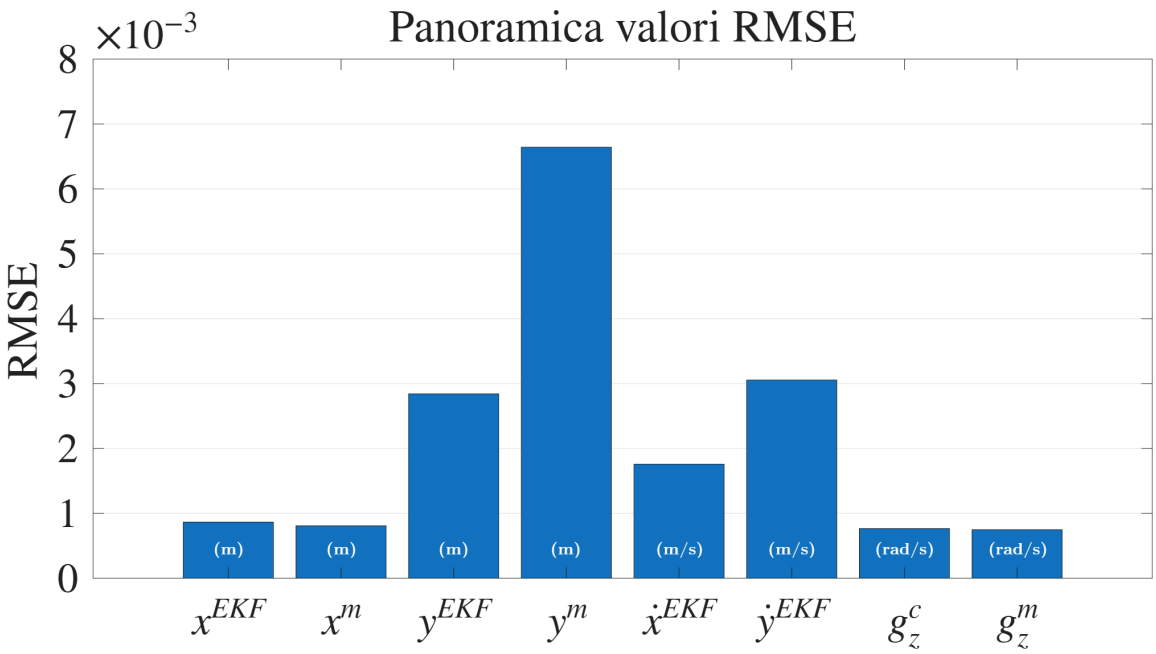


Figura 4.16: Valori RMSE del filtro di Kalman rispetto ai dati di riferimento ideali

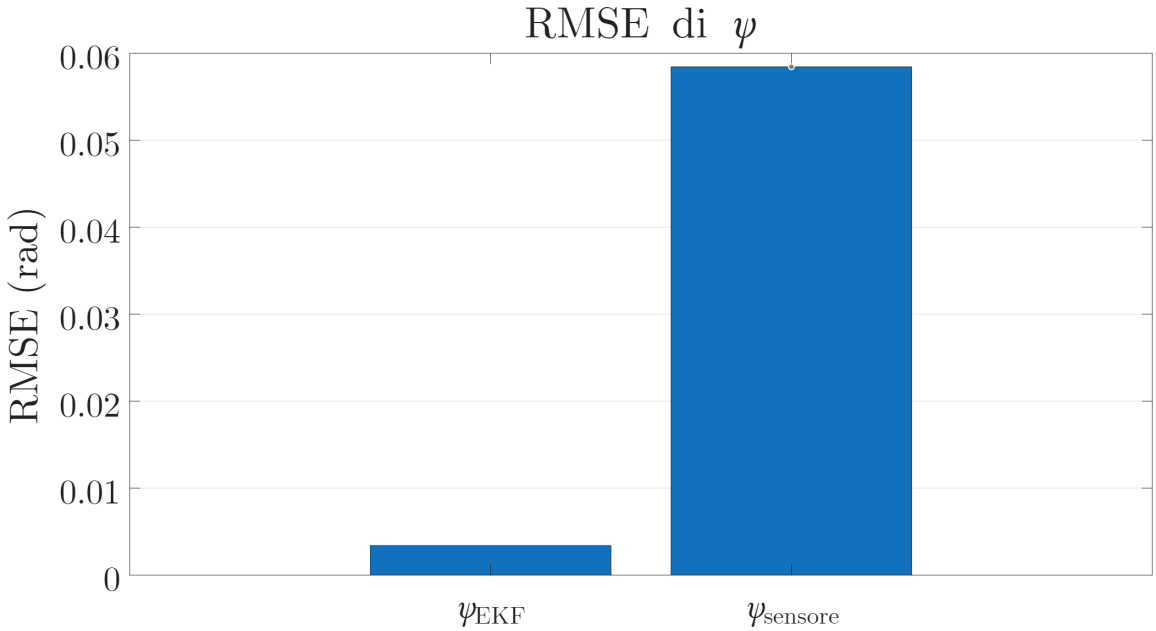


Figura 4.17: Valori RMSE del filtro di Kalman rispetto ai dati di riferimento ideali dell'angolo di yaw

É possibile ottenere una stima dell'angolo ψ utilizzando un semplice algoritmo di integrazione del segnale del giroscopio, tuttavia come mostrato nelle Figure 4.18 e 4.19 va incontro a un degrado irreversibile. Risultati peggiori si ottengono integrando dall'accelerometro, essendo per natura più rumoroso e andando incontro ad una doppia integrazione per ottenere la misura di posizione. Considerando la breve durata di queste simulazioni rispetto a quella delle operazioni di OOS, questi risultati evidenziano la necessità della sensor fusion adottata, di cui si approfondisce il discorso nel prossimo capitolo.

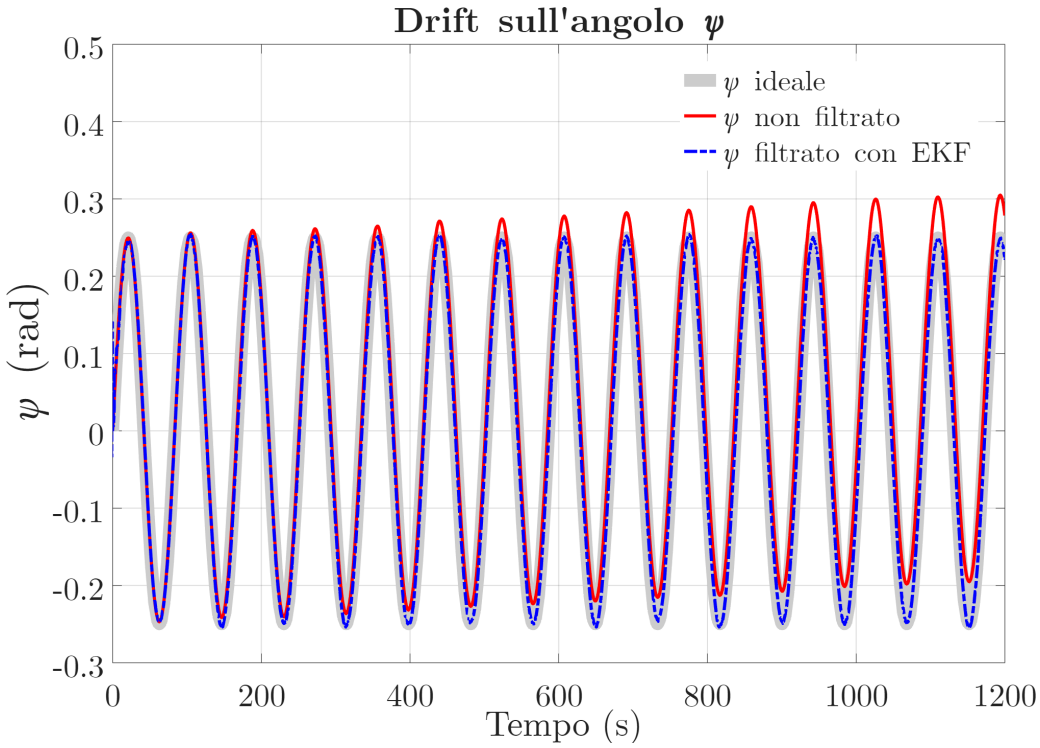


Figura 4.18: Drift di ψ integrando dal giroscopio, rispetto al valore ideale e filtrato

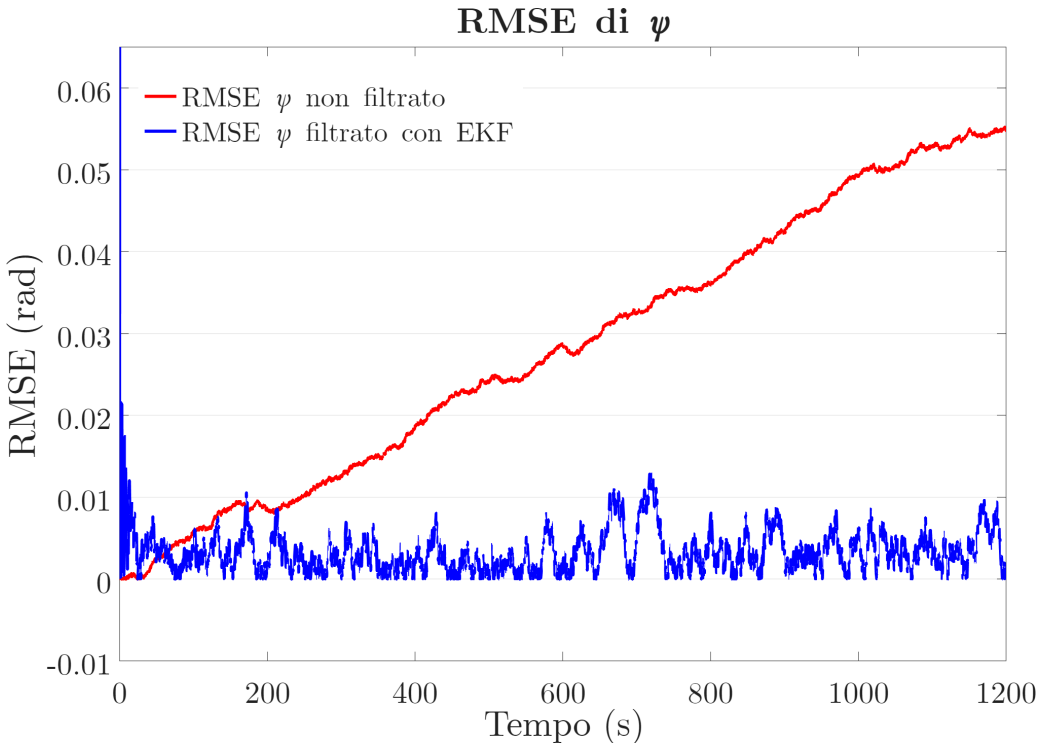


Figura 4.19: Valori RMSE del filtro di Kalman rispetto al valore di ψ integrando nel tempo senza sensor fusion

Capitolo 5

Extended Kalman Filter per la sensor fusion visual-inertial

La stima dello stato costituisce un elemento chiave del sistema di navigazione, poiché fornisce al controllore una descrizione dell'evoluzione cinematica del satellite a partire da misure sensoriali provenienti da diverse fonti che sono inevitabilmente affette da rumore e bias. Nel progetto AirSat l'esigenza di una sensor fusion è particolarmente rilevante, in quanto la piattaforma integra un sistema di misura inerziale con MPU9250 e un sistema di visione artificiale con camera RealSense D435, i cui dati devono essere combinati per ottenere una stima robusta di posizione, velocità e orientazione nel piano. Inoltre, non potendo contare su un sensore di misura della velocità diretto, emerge l'esigenza di doverla stimare in modo robusto e affidabile dalle misurazioni disponibili dei sensori presenti.

L'impiego di un singolo sensore non è sufficiente a garantire accuratezza e continuità della stima in tutte le condizioni operative. I sensori inerziali forniscono aggiornamenti rapidi e continui, ma soffrono di deriva quando le misure vengono integrate nel tempo; al contrario, le osservazioni visive sono più affidabili sul lungo periodo e consentono correzioni assolute della posa, ma sono disponibili a frequenza inferiore e possono degradarsi in presenza di ostacoli, motion blur o perdita temporanea del target. Per questo motivo, l'uso della sola IMU di cui era inizialmente dotato l'Airsat può essere adatto alla propagazione di breve periodo, ma non alla stima assoluta a lungo termine. La sensor fusion nasce proprio per sfruttare la natura complementare di queste due sorgenti. Per questo scopo, l'Extended Kalman Filter (EKF) rappresenta una delle tecniche più consolidate per sistemi non lineari. L'idea di base consiste nel combinare un modello cinematico del moto alimentato dalle misurazioni inerziali, con le osservazioni della camera, aggiornando ricorsivamente la stima dello stato e la

relativa incertezza [22, 23]. Il filtro opera quindi come uno stimatore dinamico che propaga lo *statespace* sulla base della cinematica e lo corregge ogni volta che una nuova misura è disponibile. La letteratura sulla navigazione visual-inertial mostra che i due sistemi possiedono caratteristiche complementari poiché garantiscono un ottima precisione della stima dello *statespace* e limitano gli effetti del drift[24, 25], come visibile anche dalle simulazioni sul digital twin riportati in Figura 4.18.

Nel caso dell'AirSat, il sistema di percezione utilizza un marker ArUco per stimare la posa nel frame della camera, subito traslata nel frame inerziale, e pubblicare la misura di posizione e orientazione, successivamente implementata dal nodo del filtro Kalman per correggere la predizione cinematica costruita a partire dalle informazioni dell'IMU.

Questo schema sfrutta sensori interni, come l'IMU, e sensori esterni, come le camere, ispirandosi a molte applicazioni già presenti nel panorama attuale [26, 27]. L'obiettivo della sensor fusion non è quindi la semplice media delle misure, ma la produzione di una stima più accurata, continua e robusta di uno qualunque dei due sensori preso singolarmente.

5.1 Richiami teorici sull'EKF

I filtri di Kalman sono uno strumento per stimare l'evoluzione nel tempo dello stato di un sistema, sulla base di equazioni che descrivono la dinamica dello stesso ed equazioni che descrivono la relazione tra lo stato del sistema e le quantità misurate. Il filtro di Kalman semplice offre una stima per sistemi lineari, tuttavia, i problemi di navigazione reale sono raramente lineari: il moto di un veicolo include trasformazioni tra sistemi di riferimento e rotazioni che dipendono da parametri geometrici in tempo reale. Perciò, nel caso dell'Airsat è necessario una formulazione per sistemi non lineari, che siano inoltre, discreti e soggetti ad un rumore bianco di tipo gaussiano.

Per questi casi si introduce l'Extended Kalman Filter (EKF), che estende il formalismo del Kalman filter ai sistemi non lineari mediante una linearizzazione locale attorno alla stima corrente [23]. Considerando un sistema discreto descritto da

$$\mathbf{SS}_t = f(\mathbf{SS}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t, \qquad \mathbf{w}_t \sim \mathcal{N}(0, Q), \tag{5.1}$$

e un modello delle misurazioni del tipo

$$\mathbf{z}_t = h(\mathbf{SS}_t) + \mathbf{v}_t, \qquad \mathbf{v}_t \sim \mathcal{N}(0, R), \tag{5.2}$$

dove $\mathbf{SS}_t$ è il vettore di statespace, $\mathbf{u}_t$ l'ingresso noto, ovvero le acquisizioni dell'IMU, $\mathbf{z}_t$ il vettore di misura, ovvero quelle della camera, $\mathbf{w}_t$ il rumore di processo, $\mathbf{v}_t$ il rumore di misura, $\mathbf{Q}$ la matrice di covarianza del rumore processo ed $\mathbf{R}$ la matrice di covarianza del rumore di misura, infine, le funzioni f, h descrivono il modello matematico dello stato e della misura dei sensori. Il filtro procede mediante due passi successivi: prediction e update [22, 24].

Si indicano i calcoli della fase di predizione con l'accento circonflesso. Lo stato viene propagato nel tempo attraverso il modello cinematico:

$$\hat{\mathbf{SS}}_t = f(\mathbf{SS}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t \tag{5.3}$$

mentre la covarianza dell'errore di stima globale del filtro $\mathbf{P}_t$, viene aggiornata come

$$\hat{\mathbf{P}}_t = \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^{\top} + \mathbf{V_{t-1}}\mathbf{Q}\mathbf{V_{t-1}^{T}} \tag{5.4}$$

dove $\mathbf{F}_t$ e $\mathbf{V}_t$ sono gli jacobiani del modello di processo rispetto allo statespace e rispetto al rumore di processo

$$\mathbf{F}_t = \left.\frac{\partial f(\mathbf{SS}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{SS}}\right|_{\mathbf{SS}_{t-1}, \mathbf{u}_t, \mathbf{w}_t = 0} \tag{5.5}$$

$$\mathbf{V}_t = \left.\frac{\partial f(\mathbf{SS}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{w}}\right|_{\mathbf{SS}_{t-1}, \mathbf{u}_t, \mathbf{w}_t = 0} \tag{5.6}$$

e $\mathbf{Q}$ è la covarianza del rumore di processo [22, 24].

Nel passo di update o aggiornamento, il filtro confronta la misura della camera con quella attesa secondo il modello cinematico:

$$\mathbf{i}_t = \mathbf{z}_t - h(\mathbf{SS}_t) \tag{5.7}$$

dove $\mathbf{i}_t$ è il residuo di innovazione e la funzione h seleziona le grandezze misurate, tra quelle dello stato. La jacobiana del modello di misura è

$$\mathbf{H}_t = \left.\frac{\partial h(\mathbf{SS}_t)}{\partial \mathbf{SS}}\right|_{\mathbf{SS}_t, v_t = 0} \tag{5.8}$$

mentre la covarianza dell'innovazione $\mathbf{S}_t$ è

$$\mathbf{S}_t = \mathbf{H}_t\hat{\mathbf{P}}_t\mathbf{H}_t^{\top} + \mathbf{R}, \tag{5.9}$$

con $\mathbf{R}$ covarianza del rumore di misura [22].

Il guadagno di Kalman assume quindi la forma

$$\mathbf{K}_t = \hat{\mathbf{P}}_t \mathbf{H}_t^\top \mathbf{S}_t^{-1}, \tag{5.10}$$

e consente di correggere la predizione:

$$\mathbf{SS}_t = \hat{\mathbf{SS}}_t + \mathbf{K}_t \mathbf{i}_t \tag{5.11}$$

aggiornando contestualmente la covarianza:

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\hat{\mathbf{P}}_t(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)^T + \mathbf{K_t} \mathbf{R} \mathbf{K_t}^T \tag{5.12}$$

A livello operativo, il filtro assegna maggiore peso al modello cinematico quando le misure sono incerte, e maggiore peso alle osservazioni quando il modello risulta meno affidabile. La qualità della stima dipende quindi in misura decisiva dalla corretta scelta delle matrici $\mathbf{Q}$ e $\mathbf{R}$, che descrivono il livello di fiducia rispettivamente nel modello di processo e nel modello di misura [22, 26]. Il contributo della matrice di Q viene aggiornato dal filtro in modo ricorsivo tramite la matrice di propagazione del rumore di processo V^T, per definire sempre un livello di fiducia coerente alla dinamica in tempo reale dell'algoritmo.

5.2 Struttura del filtro EKF nell'AirSat

5.2.1 Definizione dello spazio degli stati

Il filtro implementato nell'AirSat si attiene all'impostazione planare del satellite, coerente con la natura del banco prova a tre gradi di libertà (x, y, ψ). In questo contesto non è necessario adottare un modello visual-inertial completo a sei gradi di libertà, ma è sufficiente definire uno stato ridotto che descriva il moto nel piano, e presenti tutte le grandezze necessarie per il controllo come mostrato in Figura 5.1. Per un ulteriore accortezza, viene utilizzato uno stato aumentato che introduce i bias sulle acquisizioni dell'unità IMU, in modo da attuare tramite il filtro anche una calibrazione continua del sensore in tempo reale [28].

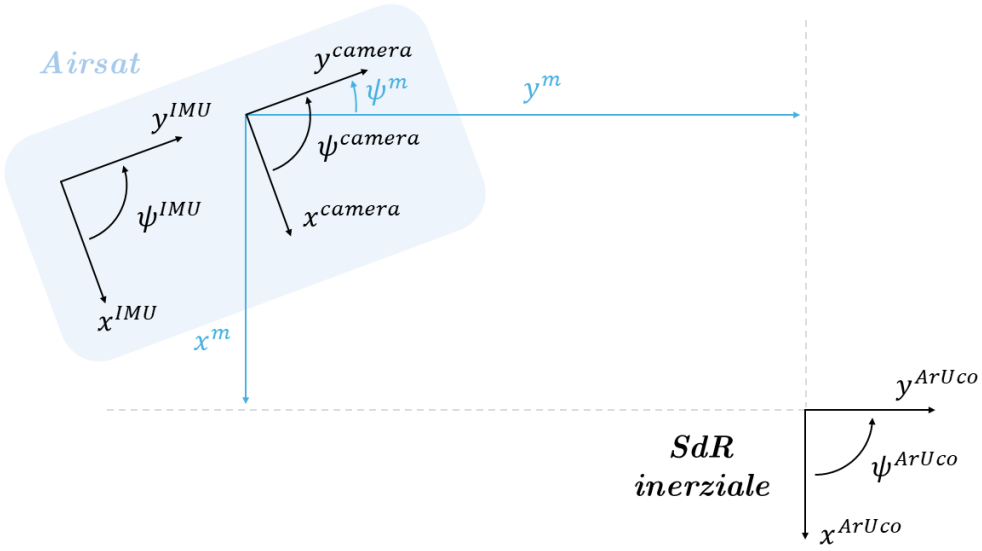


Figura 5.1: Schema dei sistemi di riferimento del sistema Airsat

Il vettore di stato è composto dalle seguenti grandezze fisiche:

$$\mathbf{SS}_t = \begin{bmatrix} x_t & y_t & \dot{x}_t & \dot{y}_t & \psi_t & b_{ax,t} & b_{ay,t} & b_{gz,t} \end{bmatrix}^\top, \tag{5.13}$$

Simbolo	**Significato**	**Unità**
x	Posizione sull'asse x nel frame inerziale	m
y	Posizione sull'asse y nel frame inerziale	m
$\dot{x}$	Velocità lungo x nel frame inerziale	m/s
$\dot{y}$	Velocità lungo y nel frame inerziale	m/s
ψ	Angolo yaw rispetto al frame inerziale	rad
b_{ax}	Bias accelerometro asse IMU x	m/s^2
b_{ay}	Bias accelerometro asse IMU y	m/s^2
b_{gz}	Bias giroscopio asse IMU z	rad/s

Tabella 5.1: Variabili di stato nell'EKF dell'AirSat.

5.2.2 Fase di prediction: modello di processo e covarianza del filtro

La fase di predizione avviene a partire dalle misure IMU, utilizzate come input. Tuttavia, è necessaria la trasformazione delle accelerazioni dal sistema di riferimento solidale all'Airsat al sistema di riferimento inerziale dell'ArUco, questo è particolarmente importante per il moto traslatorio. Infatti, posizioni e velocità sono espresse nel sistema di riferimento dell'ArUco e non nel sistema di assi dell'IMU, che nel frattempo compie un moto rotatorio. Dunque le misure dell'accelerometro devono essere ruotate nel frame inerziale mediante l'assetto istantaneo del corpo, prima di essere integrate nel tempo; in caso contrario, nella fase di controllo e attuazione la stessa forza specifica produrrebbe contributi diversi in funzione dell'orientazione dell'AirSat rispetto al marker, compromettendo la navigazione; dunque le accelerazioni misurate a_x^m e a_y^m, acquisite nel sistema di riferimento solidale al satellite, vengono corrette sottraendo i bias stimati e successivamente ruotate nel riferimento inerziale mediante la matrice di rotazione dipendente da ψ_t.

Indicando con Δt il passo di integrazione, il modello discreto si esprime:

$$\hat{\mathbf{SS}}_\mathbf{t} = f(SS_{t-1}, u_t) + w_t \qquad w_t \sim \mathcal{N}(0, Q) \tag{5.14}$$

dove w_t è il rumore di processo propagato degli input dell'IMU che ha covarianza Q, assunta costante come in precedenza, e la funzione $f(SS_{t-1}, u_t)$ è composta dallo StateSpace al passo $t-1$ e dagli input dell'IMU corretti dai bias e trasformati nel sistema di riferimento inerziale della navigazione. In particolare, come anticipato in precedenza, si utilizza uno spazio degli stati "aumentato", comprendente anche i bias degli input dell'IMU per ridurre la deriva e l'integrazione dell'errore nel tempo [28]. La funzione assume questa forma:

$$\mathbf{f}(\mathbf{SS}_{t-1}, u_t) = \begin{bmatrix} x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}\ddot{x}_t^I \Delta t^2 \\ y_{t-1} + \dot{y}_{t-1}\Delta t + \frac{1}{2}\ddot{y}_t^I \Delta t^2 \\ \dot{x}_{t-1} + \ddot{x}_t^I \Delta t \\ \dot{y}_{t-1} + \ddot{y}_t^I \Delta t \\ \psi_{t-1} + \dot{\psi}_t^I \Delta t \\ b_{ax,t-1} \\ b_{ay,t-1} \\ b_{gz,t-1} \end{bmatrix}, \tag{5.15}$$

con la velocità angolare e l'accelerazione inerziale al netto delle relative correzioni dai

bias e trasformazioni così definite (si noti l'uso dell'angolo ψ_{t-1} per la trasformazione delle accelerazioni, di cui per comodità verrà omesso il pedice procedendo nei calcoli):

$$\dot{\psi}_t^I = g_{z,t}^c = g_{z,t}^m - b_{gz,t-1}, \tag{5.16}$$

$$\begin{bmatrix} \ddot{x}_t^I \\ \ddot{y}_t^I \end{bmatrix} = \begin{bmatrix} \cos\psi_{t-1} & -\sin\psi_{t-1} \\ \sin\psi_{t-1} & \cos\psi_{t-1} \end{bmatrix} \begin{bmatrix} a_{x,t}^c \\ a_{x,t}^c \end{bmatrix} = \begin{bmatrix} \cos\psi\, a_x^c - \sin\psi\, a_y^c \\ \sin\psi\, a_x^c + \cos\psi\, a_y^c \end{bmatrix}, \tag{5.17}$$

con

$$\begin{bmatrix} a_{x,t}^c \\ a_{y,t}^c \end{bmatrix} = \left(\begin{bmatrix} a_{x,t}^m \\ a_{y,t}^m \end{bmatrix} - \begin{bmatrix} b_{ax,t-1} \\ b_{ay,t-1} \end{bmatrix} \right) \tag{5.18}$$

Simbolo	**Significato**	**Unità**	**Origine**
$\ddot{x}^I$	Accelerazione trasformata nel frame inerziale	m/s^2	(5.17)
$\ddot{y}^I$	Accelerazione trasformata nel frame inerziale	m/s^2	(5.17)
$\dot{\psi}^I$	Velocità angolare nel frame inerziale	rad/s	(5.16)
a_x^c	Accelerazione corretta dal bias b_{ax}	m/s^2	(5.18)
a_y^c	Accelerazione corretta dal bias b_{ay}	m/s^2	(5.18)
g_z^c	Velocità angolare corretta dal bias b_{gz}	rad/s	(5.16)
a_x^m	Accelerazione misurata lungo x	m/s^2	IMU
a_y^m	Accelerazione misurata lungo y	m/s^2	IMU
g_z^m	Velocità angolare misurata attorno a z	rad/s	IMU

Tabella 5.2: Variabili di input del modello di processo nell'EKF dell'AirSat.

Successivamente, si definiscono le matrici Jacobiane $\mathbf{F_t}$ e $\mathbf{V_t}$.

$$F_t = \frac{\partial f}{\partial SS} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2}\frac{\partial \ddot{x}^I}{\partial \psi} & 0 & \frac{\Delta t^2}{2}\frac{\partial \ddot{x}^I}{\partial b_{ax}} & \frac{\Delta t^2}{2}\frac{\partial \ddot{x}^I}{\partial b_{ay}} & 0 \\ 0 & 1 & 0 & \Delta t & \frac{\Delta t^2}{2}\frac{\partial \ddot{y}^I}{\partial \psi} & 0 & \frac{\Delta t^2}{2}\frac{\partial \ddot{y}^I}{\partial b_{ax}} & \frac{\Delta t^2}{2}\frac{\partial \ddot{y}^I}{\partial b_{ay}} & 0 \\ 0 & 0 & 1 & 0 & \Delta t\frac{\partial \ddot{x}^I}{\partial \psi} & 0 & \Delta t\frac{\partial \ddot{x}^I}{\partial b_{ax}} & \Delta t\frac{\partial \ddot{x}^I}{\partial b_{ay}} & 0 \\ 0 & 0 & 0 & 1 & \Delta t\frac{\partial \ddot{y}^I}{\partial \psi} & 0 & \Delta t\frac{\partial \ddot{y}^I}{\partial b_{ax}} & \Delta t\frac{\partial \ddot{y}^I}{\partial b_{ay}} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

Ci cui si esplicitano le derivate puntuali per la linearizzazione della matrice

$$\frac{\partial \ddot{x}^I}{\partial \psi} = -\sin\psi\, a_x^c - \cos\psi\, a_y^c, \tag{5.20}$$

$$\frac{\partial \ddot{y}^I}{\partial \psi} = \cos\psi\, a_x^c - \sin\psi\, a_y^c. \tag{5.21}$$

$$\frac{\partial \ddot{x}^I}{\partial b_{ax}} = -\cos\psi, \tag{5.22}$$

$$\frac{\partial \ddot{x}^I}{\partial b_{ay}} = \sin\psi, \tag{5.23}$$

$$\frac{\partial \ddot{y}^I}{\partial b_{ax}} = -\sin\psi, \tag{5.24}$$

$$\frac{\partial \ddot{y}^I}{\partial b_{ay}} = -\cos\psi. \tag{5.25}$$

$$\frac{\partial \psi}{\partial b_{gz}} = -\Delta t, \tag{5.26}$$

$$\tag{5.27}$$

Sostituendo nella matrice di transizione si evince la dipendenza dal tempo e dalla

grandezza dello statespace ψ

$$\mathbf{F_t} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & -\frac{\Delta t^2}{2}\left(\sin\psi\, a_x^c + \cos\psi\, a_y^c\right) & 0 & -\frac{\Delta t^2}{2}\cos\psi & \frac{\Delta t^2}{2}\sin\psi & 0 \\ 0 & 1 & 0 & \Delta t & \frac{\Delta t^2}{2}\left(\cos\psi\, a_x^c - \sin\psi\, a_y^c\right) & 0 & -\frac{\Delta t^2}{2}\sin\psi & -\frac{\Delta t^2}{2}\cos\psi & 0 \\ 0 & 0 & 1 & 0 & -\Delta t\left(\sin\psi\, a_x^c + \cos\psi\, a_y^c\right) & 0 & -\Delta t\cos\psi & \Delta t\sin\psi & 0 \\ 0 & 0 & 0 & 1 & \Delta t\left(\cos\psi\, a_x^c - \sin\psi\, a_y^c\right) & 0 & -\Delta t\sin\psi & -\Delta t\cos\psi & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.28}$$

e infine

$$V_t = \frac{\partial f}{\partial w} = \begin{bmatrix} \cos\psi\frac{\Delta t^2}{2} & -\sin\psi\frac{\Delta t^2}{2} & 0 & -\cos\psi\frac{\Delta t^2}{2} & \sin\psi\frac{\Delta t^2}{2} & 0 \\ \sin\psi\frac{\Delta t^2}{2} & \cos\psi\frac{\Delta t^2}{2} & 0 & -\sin\psi\frac{\Delta t^2}{2} & -\cos\psi\frac{\Delta t^2}{2} & 0 \\ \cos\psi\Delta t & -\sin\psi\Delta t & 0 & -\cos\psi\Delta t & \sin\psi\Delta t & 0 \\ \sin\psi\Delta t & \cos\psi\Delta t & 0 & -\sin\psi\Delta t & -\cos\psi\Delta t & 0 \\ 0 & 0 & \Delta t & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.29}$$

In questo modello l'accelerometro e il giroscopio vengono trattati come ingressi di controllo u_t nella fase di predizione, mentre il sistema di visione camera più ArUco agisce come vincolo di correzione. Una formulazione di questo tipo è vantaggiosa perché riduce il numero di stati e il costo computazionale, ma soprattutto mantiene la capacità di propagare il moto tra due aggiornamenti visivi successivi, il che risulta utile nei casi già citati di ostruzioni o motion blur del marker di riferimento [24, 25]. Infine, in questa fase di predizione, si calcola la matrice di covarianza dell'errore di stima dello stato del filtro.

$$\hat{\mathbf{P}}_t = \mathbf{F}_t\mathbf{P}_{t-1}\mathbf{F}_t^\top + \mathbf{V}_t\mathbf{Q}\mathbf{V}_t^\top \tag{5.30}$$

dove la matrice Q è la covarianza del rumore di processo, la quale verrà approfondita nei prossimi paragrafi.

5.2.3 Fase di update: modello di misura

Come appena accennato, nell'AirSat è presente una sorgente di misura per la fase di correzione o aggiornamento del filtro, ovvero il sistema di visione Camera più ArUco.

Esso fornisce una stima della posa planare e si presenta in questa forma:

$$\mathbf{z}_t = h(SS_t) + v_t = \mathbf{H}\mathbf{S}\mathbf{S}_t + v_t = \begin{bmatrix} x^m \\ y^m \\ \psi^m \end{bmatrix} + v_t, \qquad v_t \sim \mathcal{N}(0, R) \tag{5.31}$$

con $\mathbf{v}_t$ rumore di misura con covarianza $\mathbf{R}$ associato all'osservazione visiva.

Simbolo	Significato	Unità	Origine
x^m	Posizione lungo x del frame inerziale	m	Camera
y^m	Posizione lungo y del frame inerziale	m	Camera
ψ^m	Angolo di yaw nel frame inerziale	rad	Camera

Tabella 5.3: Variabili del modello di misura nell'EKF dell'AirSat.

La corrispondente matrice di osservazione risulta lineare, poiché la posa fornita dal nodo di percezione visiva è già espressa nel sistema inerziale di navigazione dell'ArUco e non richiede ulteriori trasformazioni:

$$\mathbf{H} = \frac{\partial h}{\partial SS} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.32}$$

5.2.4 Matrici di covarianza dei rumori di processo e di misura

Le matrici di covarianza del rumore di processo e del rumore di misura sono costituite dai valori di covarianza delle rispettive grandezze rappresentate [29]. Questi valori si rifanno dunque alla modellazione dei sensori effettuata per il digital twin, descritta nel Capitolo 4. In particolare, per semplicità computazionale viene mantenuto solo l'ordine di grandezza delle covarianze.

In forma generale, la covarianza di processo prende una forma simmetrica poichè si assume che il rumore di una misurazione non influenzi il rumore di un altra:

$$\mathbf{Q} = \mathrm{diag}(q_{ax}, q_{ay}, q_{gz}, q_{b_{ax}}, q_{b_{ay}}, q_{b_{gz}}), \tag{5.33}$$

allo stesso modo la covarianza di misura risulta diagonale per lo stesso motivo:

$$\mathbf{R} = \mathrm{diag}(r_x, r_y, r_\psi), \tag{5.34}$$

Parametro	**Valore**	**Significato**
$q_{ax} = \sigma_{ax}^2$	$5 \cdot 10^{-4}$	Rumore di processo su a_x
$q_{ay} = \sigma_{ay}^2$	$5 \cdot 10^{-4}$	Rumore di processo su a_y
$q_{g_z} = \sigma_{g_z}^2$	$5 \cdot 10^{-7}$	Rumore di processo su g_z
$q_{b_{ax}} = \sigma_{b_{ax}}^2$	$1.0 \cdot 10^{-8}$	Rumore di processo su b_{ax}
$q_{b_{ay}} = \sigma_{b_{ay}}^2$	$1.0 \cdot 10^{-8}$	Rumore di processo su b_{ay}
$q_{b_{gz}} = \sigma_{b_{gz}}^2$	$1.0 \cdot 10^{-10}$	Rumore di processo su b_{gz}
$r_x = \sigma_x^2$	$5 \cdot 10^{-7}$	Rumore di misura su x
$r_y = \sigma_y^2$	$5 \cdot 10^{-5}$	Rumore di misura su y
$r_\psi = \sigma_\psi^2$	$5 \cdot 10^{-3}$	Rumore di misura su ψ
p_0	1.0	Covarianza iniziale diagonale

Tabella 5.4: Parametri di taratura dell'EKF utilizzati nell'implementazione AirSat.

infine la covarianza iniziale del filtro EKF assume la forma

$$\mathbf{P}_0 = \mathbf{I}^{8\times 8} p_0. \tag{5.35}$$

La taratura di queste matrici ha un impatto diretto sul comportamento del filtro. Se $\mathbf{Q}$ è troppo piccola, il filtro tende a fidarsi eccessivamente del modello cinematico e reagisce lentamente agli aggiornamenti reali della camera; se è troppo grande, la stima diventa più rumorosa e meno stabile. In modo analogo, una $\mathbf{R}$ troppo piccola attribuisce alle misure un'affidabilità eccessiva, rendendo il filtro sensibile a outlier e disturbi, mentre una $\mathbf{R}$ troppo grande riduce l'efficacia della correzione [26, 21]. Allo stesso modo la matrice $\mathbf{P_0}$ può influenzare il comportamento del filtro, poichè regola la fiducia nello stato iniziale: se è troppo grande il filtro parte correggendo aggressivamente con le osservazioni della camera, mentre se è troppo piccola il filtro resta più rigido e si attiene al modello matematico, reagendo più lentamente alle correzioni delle misure [30].

La letteratura evidenzia che una cattiva modellazione delle matrici di covarianza può portare a divergenza del filtro [26, 21]. Per questo motivo è importante effettuare una corretta analisi dei sensori a disposizione, come svolto nel Capitolo 4, nel quale si sono individuati i valori di covarianza caratteristici delle diverse misurazioni.

Matrice	Dim	Ruolo
$\mathbf{F}_t$	8×8	Matrice jacobiana del modello di processo
$\mathbf{V}_t$	8×8	Matrice jacobiana di propagazione del rumore di processo
$\mathbf{H}$	3×8	Matrice di osservazione
$\mathbf{Q}$	8×8	Matrice di covarianza del rumore di processo
$\mathbf{R}$	3×3	Matrice di covarianza del rumore di misura
$\mathbf{P}_t$	8×8	Matrice di covarianza del filtro di Kalman

Tabella 5.5: Elenco delle matrici utilizzate nell'EKF di AirSat.

5.2.5 Fase di update: calcolo del guadagno di Kalman e covarianza del filtro

Successivamente il filtro prevede il calcolo del residuo o innovazione i_t

$$\mathbf{i}_t = \mathbf{z}_t - h(\mathbf{SS}_t), \tag{5.36}$$

mentre la covarianza dell'innovazione S_t è

$$\mathbf{S}_t = \mathbf{H}\hat{\mathbf{P}}_t\mathbf{H}^\top + \mathbf{R} \tag{5.37}$$

con $\mathbf{R}$ covarianza del rumore di misura come appena discusso.

Il guadagno di Kalman assume quindi la forma

$$\mathbf{K}_t = \hat{\mathbf{P}}_t\mathbf{H}^\top\mathbf{S}_t^{-1} \tag{5.38}$$

e consente di correggere la predizione:

$$\mathbf{SS}_t = \hat{\mathbf{SS}}_t + \mathbf{K}_t\mathbf{i}_t \tag{5.39}$$

aggiornando contestualmente la covarianza, in particolare viene aggiornata con la forma di Joseph, scelta numericamente più robusta rispetto alla forma semplificata classica. Se $\hat{\mathbf{P}}_\mathbf{t}$ è la covarianza predetta, $\mathbf{K}_t$ il guadagno di Kalman e $\mathbf{H}$ la matrice di osservazione, l'aggiornamento è dato da

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\hat{\mathbf{P}}_t(\mathbf{I} - \mathbf{K}_t\mathbf{H})^\top + \mathbf{K_t}\mathbf{R}\mathbf{K_t}^T. \tag{5.40}$$

Questa formulazione preserva meglio la simmetria della matrice di covarianza e la proprietà di semidefinita positiva, soprattutto in presenza di arrotondamenti numerici.

5.3 L'Extended Kalman Filter nell'Airsat

Il filtro EKF di AirSat è una forma di stima dello stato visual-inertial semplificata e mirata al problema planare ed è implementato a livello software come un nodo nel workspace ROS 2, come verrà approfondito nel Capitolo 6. Un aspetto rilevante è la stretta integrazione tra stimatore e controllore. Il nodo LQI utilizza infatti direttamente la stima pubblicata dal filtro per calcolare il comando della reaction wheel. Ciò significa che ogni miglioramento nella qualità della stima si riflette direttamente nella qualità del controllo, riducendo oscillazioni, ritardi e sensibilità al rumore.

L'adozione dell'Extended Kalman Filter in AirSat presenta vantaggi pratici tra cui la capacità di fondere informazioni eterogenee a frequenze diverse, mantenendo una struttura teorica solida ma sufficientemente leggera per un sistema sperimentale, oltre alla possibilità di includere nello stato gli errori sistematici dei sensori inerziali, mantenendo la qualità della stima nel tempo [24, 25].

Accanto a questi vantaggi, l'EKF può presentare qualche limite se la linearizzazione locale diventa inaccurata in presenza di forti non linearità. Oltre a sottostimare la reale incertezza della stima quando il modello è troppo semplificato o quando le ipotesi statistiche sul rumore non siano rispettate [22, 21]. Perciò è necessario un'accurata modellazione di tutti componenti matematici del filtro per ottenere dei buoni risultati.

Capitolo 6

Architettura ROS2

In questo capitolo si illustra lo sviluppo della parte software che sta alla base del sistema AirSat. Lo scopo di questo software è l'azionamento e l'interazione della strumentazione di bordo per svolgere funzioni come l'acquisizione delle misure, la stima dell'odometria, il controllo e l'attuazione del free-flyer. La necessità è quella di adottare un'architettura modulare, che sia in grado di gestire contemporaneamente tutti i sottosistemi. In particolare, tale architettura deve soddisfare due requisiti: consentire una comunicazione efficace tra diversi programmi e permettere a ciascun modulo di operare in modo indipendente, senza la necessità di una esecuzione sincronizzata. Questa struttura risulta adatta al banco prova sperimentale, poiché consente di sviluppare, testare e modificare ciascun modulo facilmente, in modo indipendente.

ROS2 (Robot Operating System 2) è un sistema adatto a questo contesto poiché nasce con lo scopo di supportare sistemi modulari. La comunicazione tra i diversi programmi, detti *nodi*, avviene tramite *topic* e *messaggi* standardizzati, permettendo di organizzare il software in blocchi separati ognuno associato a un diverso sottosistema del free-flyer Airsat. ROS2 è un middleware per la robotica basato su Linux e, all'interno di un workspace, consente di organizzare un progetto in package, cioè insiemi di codice dedicati allo svolgimento di funzioni specifiche. I *nodi* sono codici elementari Python che eseguono un programma, ad esempio leggere le acquisizioni di un sensore, stimare lo stato odometrico o generare un comando di controllo, mentre i *topic* sono i canali logici attraverso cui i nodi scambiano dati tra loro in modo non necessariamente sequenziale. Un nodo che invia dati su un topic è chiamato *publisher*, mentre quello che li riceve è un *subscriber*; in questo modo l'architettura resta flessibile, perché ogni modulo può operare indipendentemente dagli altri. I dati scambiati sono *messaggi* strutturati secondo tipi definiti, ad esempio messaggi per IMU, odometria o comandi di attuazione, così da garantire che ogni informazione abbia una forma univoca e interpretabile da tutti i nodi coinvolti. Ogni nodo può fare riferimento a

uno o più file di configurazione contenente dei parametri di funzionamento dai quali attingere, ciò consente di tenerli sotto controllo in un unico punto e facilitarne la regolazione. È possibile generare anche tipi di messaggi personalizzati.

In questo capitolo viene descritta nel dettaglio l'architettura basata su ROS2 di AirSat. Verranno analizzate le logiche di funzionamento di ciascun nodo, le interfacce con Arduino gestite tramite micro-ROS, e la trattazione matematica che governa gli algoritmi di stima, controllo e attuazione.

6.1 Panoramica Generale

L'architettura software di *AirSat* è predisposta per consentire il controllo in anello chiuso della posizione e dell'assetto del free-flyer. Considerando lo sviluppo della testbench ancora in fase iniziale, non risulta possibile la sperimentazione del moto traslatorio, per cui la trattazione che segue svilupperà il sistema nel complesso fino alla stima dell'odometria, lasciando l'ultima parte relativa all'attuazione riservata unicamente al moto rotatorio, ovvero al controllo della Reaction Wheel. La configurazione è composta da un PC Linux, un microcontrollore Arduino Nano ESP32 e le diverse periferiche, tra cui la camera, il sensore inerziale e il motore brushless insieme al suo digital controller. Come visibile in Figura 6.1 il workspace ROS2 viene eseguito sul PC, che si occupa anche delle acquisizioni della camera poichè richiede la gestione di un flusso di dati elevato. Dall'altra parte, il microcontrollore ESP32 gestisce l'interfaccia di basso livello con il sensore inerziale e il motore.

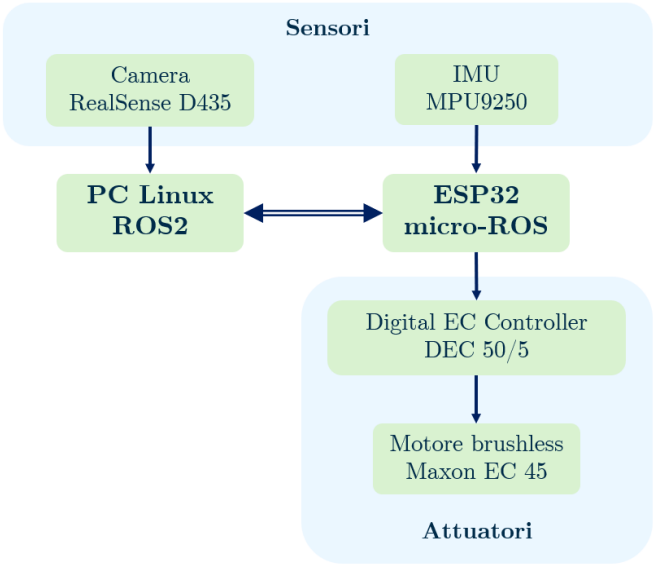


Figura 6.1: Schema delle connessioni delle periferiche di Airsat.

Nel workspace sono presenti i nodi logici che implementano gli algoritmi di GNC e nodi che non hanno un vero e proprio ruolo nell'economia del controllo ma sono di supporto al sistema, ad esempio quello di avvio generale o quello per la creazione di messaggi di tipo custom. Il flusso logico dei nodi segue lo schema riportato in Figura 6.2. Il nodo di Percezione acquisisce le immagini tramite la camera, e stima la posa del veicolo rispetto al marker ArUco, pubblicando sul topic *airsat/aruco_marker_pose* un messaggio con la misura diretta della posizione e dell'orientamento. Contemporaneamente, il programma sul microcontrollore ESP32 campiona l'accelerazione lineare e la velocità angolare dall'unità IMU MPU9250, pubblicando i dati tramite un messaggio sul topic *airsat/imu_lite*. L'interfaccia micro-ROS permette al microcontrollore di operare come un nodo del workspace ROS2 su PC. Successivamente interviene il nodo di Sensor Fusion che si iscrive ai suddetti topic sfruttando queste letture per implementare un Kalman Filter e pubblicare la stima dell'odometria su *airsat/odometry/filtered*. Il nodo Guidance pubblica sul topic *airsat/reference/setpoint* il riferimento per il successivo controllo, che può essere di tipo fisso o può implementare un'algoritmo specifico di navigazione in base alle informazioni dell'odometria. Infine, il nodo di Controllo, calcola l'errore tra i dati di setpoint e l'odometria per produrre un comando di coppia, che viene tradotto in tensione per il motore e pubblicato tramite un'altro topic micro-ROS *airsat/rw/voltage_cmd* verso il microcontrollore ESP32, che impone la tensione calcolata al motore.

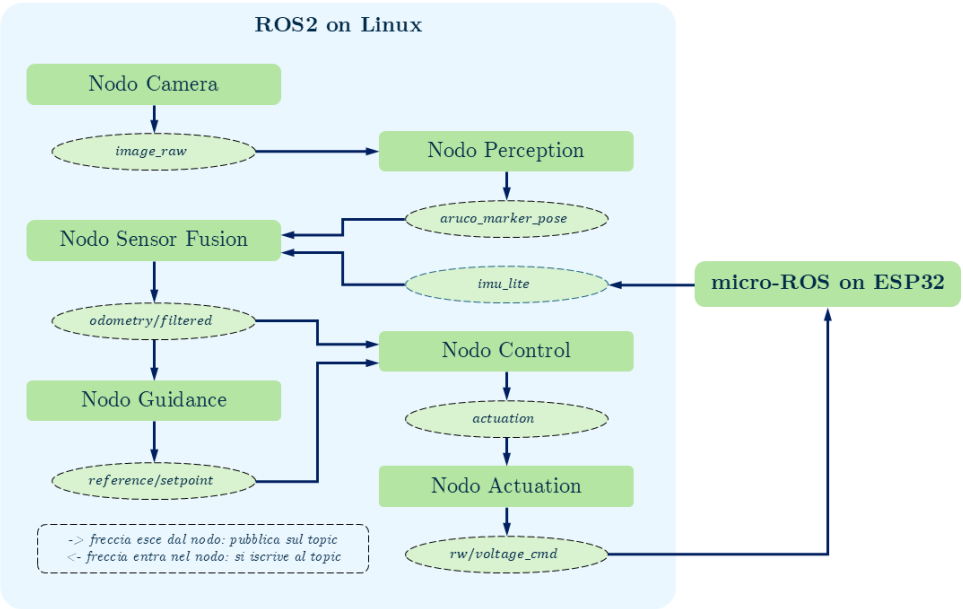


Figura 6.2: Rappresentazione del grafo di comunicazione ROS2, in cui i nodi sono indicati mediante riquadri e i topic mediante elementi ovali. L'orientamento delle frecce evidenzia le relazioni di pubblicazione e sottoscrizione.

6.2 Nodo microROS e interfaccia con il microcontrollore

L'interfaccia a basso livello viene gestita da un firmware in linguaggio C++ caricato sul microcontrollore ESP32, integrato con l'ecosistema ROS2 tramite il protocollo di comunicazione microROS. Il firmware è strutturato per operare in tempo reale secondo una logica a chiamata. Il loop principale coordina l'acquisizione dei sensori e la ricezione dei comandi, evitando l'uso di ritardi bloccanti che potrebbero compromettere la stabilità del sistema di controllo. L'esecuzione delle operazioni segue uno schema suddiviso in tre compiti principali:

1. la lettura periodica e la pubblicazione dei dati del sensore inerziale, regolate da un timer;
2. la ricezione e l'applicazione dei comandi di tensione sul driver del motore;
3. il supporto per la gestione di situazioni straordinarie.

6.2.1 Configurazione dei dispositivi hardware

Il firmware controlla due dispositivi principali a bordo della piattaforma: il driver del motore della ruota di reazione e il sensore inerziale. In Figura 6.3 si riporta lo schema logico dei pin del microcontrollore Arduino Nano ESP32 dal datasheet ufficiale della scheda [31].

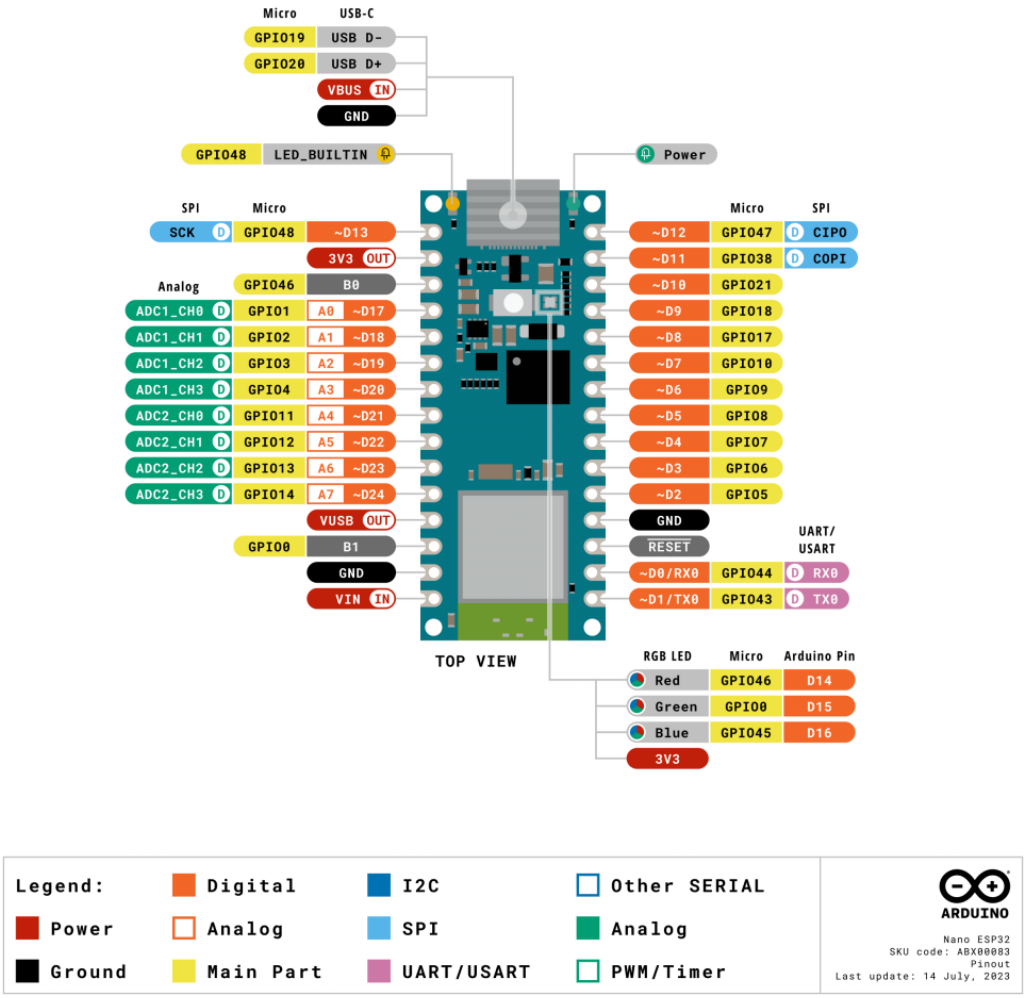


Figura 6.3: Schema logico dei pin del microcontrollore Arduino Nano ESP32.

Driver del motore DEC Module 505

Il controllo del motore della reaction wheel avviene tramite il driver DEC Module 505, che si interfaccia con il microcontrollore attraverso cinque pin totali. La Tabella 6.1 elenca i pin utilizzati e la loro funzione.

Pin	Nome variabile	Funzione
D7	PIN_ENABLE	Abilitazione del driver (HIGH = abilitato, LOW = disattivato)
D8	PIN_DIRECTION	Verso di rotazione (LOW = orario, HIGH = antiorario)
A2	PIN_SET_SPEED	Segnale di comando in tensione analogica (via PWM a 10 bit)
A3	PIN_DIGIN1	Configurazione del range di velocità (Bit 0)
A6	PIN_DIGIN2	Configurazione del range di velocità (Bit 1)

Tabella 6.1: Connessioni tra microcontrollore e DEC Module 50/5.

Nel datasheet del driver viene indicato come un segnale LOW un valore inferiore a 0.8V, mentre un segnale superiore a 2.4V come HIGH [15]. Attraverso i pin D7 e D8 viene impostata l'attivazione del motore e il verso di rotazione che vengono gestiti in base al segnale di set in arrivo dal sistema di controllo. Il driver DEC Module 50/5 permette due modalità di regolazione della velocità, una in anello chiuso e una in anello aperto; nella modalità ad anello chiuso, permette di selezionare diversi intervalli di velocità per affinare la precisione di regolazione. La configurazione dei pin PIN_DIGIN1 e PIN_DIGIN2 stabilisce in che modalità opera il driver. In Tabella 6.2 sono riportate le modalità.

PIN_DIGIN1	PIN_DIGIN2	rpm min	rpm max
LOW	LOW	Modalità ad anello aperto	
HIGH	LOW	62	625
LOW	HIGH	62	2500
HIGH	HIGH	62	10000

Tabella 6.2: Configurazioni del range di velocità del motore.

La configurazione utilizzata nel firmware è quella in anello aperto. In questa modalità, il driver non esegue una regolazione retroazionata, ma modula direttamente la velocità del motore tramite un segnale PWM interno con duty cycle D che va da 0% al 95%, proporzionale alla tensione analogica imposta V_{set} applicata dal microcontrollore al PIN_SET_SPEED.

$$D = 0.95 \cdot \frac{V_{\text{set}}}{5.0\ \text{V}} \tag{6.1}$$

Il driver traduce il duty cycle del PWM in velocità, la velocità massima raggiungibile dipende dai parametri del motore e viene calcolata con la seguente equazione [15]:

$$n_{max} = 0.95 \cdot \left[(V_{CC} - 0.3V) \cdot \frac{n_0}{U_N} \right] - \left[\frac{\Delta n}{\Delta M} \cdot M_B \right] \tag{6.2}$$

con i seguenti parametri in Tab. 6.3. [14].

Nome variabile	**Simbolo**	**Valore**	**Unità di misura**
Operating torque	M_B	55	mNm
Nominal motor Voltage	U_N	12	V
Nominal no-load speed at U_N	n_0	4370	rpm
Speed/Torque gradient of the motor	$\Delta n/\Delta M$	17.6	rpm/mNm
Supply Voltage	V_{CC}	12	V

Tabella 6.3: Parametri del Motore Maxon EC45

Poichè il microcontrollore può generare una tensione massima di 3.3V, mentre il driver accetta segnali di V_{set} fino a 5V, si crea una condizione di utilizzo che non arriva a saturare la massima velocità del motore, ma rimane comunque più che sufficiente per il range operativo del progetto. La risoluzione del segnale di V_{set} nel microcontrollore è impostata a 10 bit per garantire un passo minimo di comando di 3.23 mV, per sfruttare la sensibilità offerta dal driver che presenta una risoluzione di 4.88 mV.

Sensore inerziale MPU9250

Il sensore inerziale MPU9250 integra un accelerometro e un giroscopio e si occupa dell'acquisizione dell'accelerazione e della velocità angolare del free-flyer. Il dispositivo è collegato al microcontrollore tramite 2 pin e comunica sfruttando il protocollo seriale I2C.

Pin	**Nome variabile**	**Funzione**
A4	PIN_SDA	trasmette i dati
A5	PIN_SCL	sincronizza il clock

Tabella 6.4: Connessioni tra microcontrollore e MPU9250.

Per seguire adeguatamente la dinamica di moto del satellite, il sensore è configurato con i seguenti parametri:

- fondo scala accelerometro: $\pm 2G$, per massimizzare la sensibilità alle piccole variazioni di accelerazione sul banco di prova;
- fondo scala giroscopio: $\pm 250°/s$, per avere la massima risoluzione angolare a basse velocità.

Inoltre, le acquisizioni vengono filtrate al fine di attenuare le vibrazioni ad alta frequenza con un filtro digitale passa-basso (DLPF) attivato su entrambi i canali di misura, con frequenza di taglio pari a 45 Hz compatibili con il campionamento a 100 Hz.

6.2.2 Inizializzazione e protocollo di calibrazione del firmware

All'avvio del sistema, il microcontrollore esegue le seguenti operazioni:

1. Inizializzazione e calibrazione del sensore inerziale: una volta stabilita la comunicazione tra microcontrollore e MPU9250 inizia la fase di calibrazione del sensore inerziale che avviene sottraendo i bias stazionari dalle acquisizioni, questa fase richiede che la piattaforma rimanga in uno stato di quiete per un intervallo di tempo sufficiente in cui avviene il calcolo dei bias. Grazie alla calibrazione i valori di offset vengono salvati dal sensore per sottrarli automaticamente ad ogni successiva lettura.

2. Inizializzazione del driver: una volta stabilita la connessione viene disattivato il motore e impostata la velocità a zero, successivamente viene selezionata la modalità di funzionamento in anello aperto. Infine, rimane in attesa del primo segnale di comando.

3. Inizializzazione protocollo di comunicazione micro-ROS: l'integrazione del firmware nell'anello di controllo di ROS2 avviene tramite una sessione seriale di micro-ROS stabilita sulla porta USB del microcontrollore. All'avvio, il firmware, col ruolo di Client micro-ROS, tenta la connessione con il micro-ROS Agent eseguito sul PC. Se la sincronizzazione ha successo, vengono inizializzate le entità ROS2 associate al nodo. Il nodo di micro-ROS, denominato airsat_esp32_node, configura due canali di comunicazione per lo scambio dei dati e dei comandi.

6.2.3 Struttura del ciclo principale del firmware

Di seguito viene presentata la struttura del ciclo principale del firmware che comprende:

1. la lettura dei dati del sensore e la loro trasmissione tramite microROS;

2. la ricezione dei comandi di controllo tramite microROS e la loro attuazione;

3. la gestione di situazioni di emergenza.

Acquisizione e trasmissione IMU

L'acquisizione delle misurazioni avviene a 100 Hz, mentre la loro trasmissione avviene a una frequenza di 30Hz, regolate da un timer di microROS associato alla funzione di callback dell'IMU. La differenza tra la frequenza di campionamento con quella di pubblicazione viene adottata in questa fase iniziale del progetto per evitare l'eccessivo affollamento della comunicazione seriale in ingresso e uscita dal microcontrollore. Allo stesso tempo, mantiene la predisposizione ad un futuro aggiornamento per l'aumento della frequenza di trasmissione dei campioni di misura.

All'interno della funzione di callback, se la lettura dell'MPU9250 ha esito positivo, le accelerazioni lineari e le velocità angolari grezze vengono convertite in unità fisiche del Sistema Internazionale:

$$a_{x,\mathrm{SI}} = a_{x,\mathrm{sensore}} \cdot 9.806 \quad [\mathrm{m/s^2}] \tag{6.3}$$

$$a_{y,\mathrm{SI}} = a_{y,\mathrm{sensore}} \cdot 9.806 \quad [\mathrm{m/s^2}] \tag{6.4}$$

$$g_{z,\mathrm{SI}} = g_{z,\mathrm{sensore}} \cdot \frac{\pi}{180} \quad [\mathrm{rad/s}] \tag{6.5}$$

Il firmware assegna al messaggio un timestamp sincronizzato con l'orologio di sistema di ROS2. I dati convertiti vengono inseriti nel messaggio custom *AirsatImuLite* composto dai campi *ax*, *ay*, *gz*, una flag logica *valid* e pubblicati sul topic ROS2 /airsat/imu_lite.

Ricezione dei comandi di tensione

Il nodo microROS airsat_esp32_node è iscritto al topic */airsat/rw/voltage_cmd* per ricevere i comandi di tensione calcolati dal controllore LQI in ROS2. La ricezione è gestita in modo asincrono tramite la funzione callback del comando di tensione. Ogni volta che viene ricevuto un nuovo comando:

1. Il valore viene estratto e in caso di valori indefiniti o infiniti, la tensione viene forzata a 0.0 V.
2. Il comando viene limitato nell'intervallo operativo $[-3.3\ \mathrm{V}, 3.3\ \mathrm{V}]$.
3. Se il valore assoluto della tensione richiesta è inferiore a una soglia minima di rumore (0.05 V), il motore viene disattivato.
4. Se la tensione è valida, il firmware imposta la direzione tramite il pin dedicato e calcola il duty cycle PWM da applicare al pin analogico.

Algoritmi di sicurezza

Il firmware implementa diverse procedure di salvaguardia hardware direttamente nel microcontrollore, garantendo l'integrità del motore e della ruota di reazione anche in caso di blocco del sistema operativo del PC di controllo:

1. Supervisione del comando di tensione: durante il loop viene eseguita periodicamente una funzione che calcola l'intervallo di tempo trascorso dall'ultimo comando valido ricevuto. Nel caso risultasse superiore al limite impostato di 700 ms, il firmware rileva un'interruzione anomala del flusso di controllo e attiva la procedura di arresto di emergenza del motore.

2. Rampa di decelerazione controllata: per evitare brusche variazioni di coppia che potrebbero danneggiare le componenti elettriche del sistema, il firmware non azzera istantaneamente la tensione in caso di emergenza. Viene invece attivata una rampa di decelerazione programmata. Durante l'esecuzione della rampa viene ignorato temporaneamente qualsiasi comando di tensione in ingresso fino al completo arresto del sistema. La rampa viene attivata anche durante il funzionamento ordinario se il controllore richiede una variazione eccessiva di tensione addolcendo la frenata della ruota di reazione.

3. Gestione dei guasti e riconnessione: Il firmware monitora continuamente la qualità del collegamento microROS e dichiara uno stato *degraded* quando rileva assenza di trasmissioni, errori di pubblicazione o mancati aggiornamenti di comando del motore. In questo stato esegue un arresto controllato del sistema e, avvia la procedura di riconnessione della sessione seriale.

6.3 Nodo Bringup

Il nodo di Bringup è un nodo ausiliaro implementato tramite lo script *airsat_full_system.launch.py*. La sua funzione è automatizzare l'avvio di tutti i nodi del workspace in modo centralizzato e caricare i file di configurazione di ciascuno di essi contenenti i relativi parametri funzionali. In particolare carica cinque file di configurazione:

1. **`control_params.yaml`**: contiene i guadagni K_{yaw}, K_{yawrate}, $K_{i,\text{yaw}}$ del controllo LQI dell'assetto; i limiti di coppia massima dell'attuatore; la tensione massima ammessa e l'inerzia della Reaction Wheel.

2. **`estimation_params.yaml`**: definisce le matrici di covarianza del rumore di processo Q e di misura R per il Kalman Filter e la sua covarianza iniziale P_0.

3. **`guidance_params.yaml`**: a seconda della modalità di funzionamento selezionata, specifica i parametri per il calcolo live della traiettoria di riferimento come limiti di velocità e accelerazione angolare oppure fornisce delle coordinate di setpoint fisse.

4. **`io_params.yaml`**: definisce i topic di comunicazione per l'attuatore fisico.

5. **`perception_params.yaml`**: contiene i parametri per la camera RealSense, tra cui l'ID del marker ArUco da tracciare e la frequenza di campionamento.

6.4 Nodi Camera e Perception

Il sistema di visione artificiale è implementato attraverso due nodi distinti, il primo è quello dedicato alla camera che pubblica la stream a colori sul topic `image_raw`; il secondo è il nodo di percezione, deputato alla stima e alla pubblicazione della posizione e dell'assetto del veicolo sul topic `aruco_marker_pose` oltre ad un flag booleano di acquisizione del marker sul topic `target_visibile`.

Il sistema di visione si compone di due elementi, la Camera RealSense D435 montata a bordo veicolo e il Marker ArUco posizionato di fronte al free-flyer. Di seguito sono illustrati i parametri di configurazione dei nodi.

Parametro	**Valore Default**	**Descrizione**
enable_color	true	Abilita il flusso RGB.
enable_depth	false	Disabilita il flusso depth.
enable_infra	false	Disabilita il flusso infrared.
enable_infra1	false	Disabilita infrared 1.
enable_infra2	false	Disabilita infrared 2.
pointcloud.enable	false	Disabilita la generazione di point cloud.
align_depth.enable	false	Disabilita l'allineamento depth/RGB.
enable_sync	false	Disabilita la sincronizzazione dei flussi.
rgb_camera.profile	1280x720x30	Profilo del flusso RGB.
rgb_camera.color_format	RGB8	Formato colore del flusso RGB.

Tabella 6.5: Parametri del nodo Camera.

Come è visibile dalla tabella, è possibile implementare in futuro sistemi di visione più avanzati o combinati per ampliare gli scenari di utilizzo del free-flyer.

Parametro	**Valore Default**	**Descrizione**
image_topic	`/camera/realsense2_camera_node/color/image_raw`	Topic dell'immagine in ingresso.
marker_pose_topic	`/airsat/aruco_marker_pose`	Topic della posa del marker rilevato.
target_visible_topic	`/airsat/target_visible`	Topic booleano della visibilità del target.
aruco_dictionary	DICT_5X5_100	Dizionario ArUco utilizzato.
marker_length	0.115	Lunghezza del marker in m.
target_id	-1	ID target da rilevare, -1 qualsiasi target.
process_every_n_frames	1	Processa ogni frame acquisito.

Tabella 6.6: Parametri del Nodo Perception.

Acquisizione e stima della *pose*

Il nodo `aruco_pose_node.py` costituisce il componente di percezione visiva dedicato all'acquisizione della posa relativa tra il sistema di visione installato su AirSat e il marker ArUco, impiegato come riferimento del target nel banco sperimentale. Come anticipato, il nodo riceve le immagini a colori della camera dal topic `image_raw` e le elabora con una logica *best effort*, ovvero preferisce un flusso di informazioni aggiornato al costo di perdite di pacchetti di informazioni, piuttosto che introdurre ritardi nel tentativo di recuperarli.

Nella configurazione del nodo viene selezionato il dizionario `DICT_5X5_100`, impostata la dimensione fisica del lato del marker pari a 0,115 m e caricati i dati di calibrazione della camera, ottenuti offline mediante *cv2.calibrateCamera*, in questo modo si garantisce il legame tra l'informazione grafica acquisita e la scala metrica per la ricostruzione tridimensionale. Inoltre, tramite il parametro `target_id` si può scegliere uno specifico marker ID da individuare, ma per semplicità si imposta il riconoscimento generico di tutti gli ID del dizionario ArUco selezionato. Questa funzione può stare alla base di future evoluzioni del progetto, facendo in modo di attribuire una specifica manovra di servicing in base al numero del marker individuato dalla camera.

L'elaborazione della pose del marker avviene tramite la libreria OpenCV. In un primo momento, la funzione `cv2.aruco.detectMarkers()` ricerca i pattern quadrati del dizionario ArUco selezionato e restituisce l'identificativo e le coordinate in pixel dei quattro vertici del marker. Successivamente, la funzione `cv2.aruco.estimatePoseSingleMarkers()`, una volta riconosciuto un marker valido, sfrutta la sua dimensione nota e i parametri intrinseci della camera, ovvero la matrice di calibrazione `cameraMatrix` e il vettore dei coefficienti di distorsione `distCoeffs`, per risolvere il problema Perspective-n-Point (PnP). Tramite la suddetta funzione viene determinato il vettore di rotazione `rvec` e il vettore di traslazione `tvec` del marker. Il vettore `rvec` descrive l'orientamento del marker rispetto alla camera e viene convertito nella matrice di rotazione R mediante `cv2.Rodrigues()` dal quale viene estratto l'angolo di *yaw* ψ. Il vettore `tvec`, invece, esprime la posizione del centro del marker nel sistema di riferimento della camera. Nello specifico, indicando `tvec` $= [t_x,\ t_y,\ t_z]^T$, la componente t_z rappresenta la profondità, la componente t_x rappresenta lo scostamento laterale, infine, t_y descrive lo scostamento verticale, seguendo la convenzione classica di OpenCV. Tuttavia, per unificare le convenzioni di riferimento tra la camera e il sensore IMU, viene assegnato il valore di profondità all'asse y e il valore di scostamento laterale all'asse x, entrambi cambiati di segno per ottenere la misura già nel sistema di riferimento inerziale come visibile in Figura 6.4. Per quanto riguarda la rotazione non sono necessarie modifiche alle misurazioni.

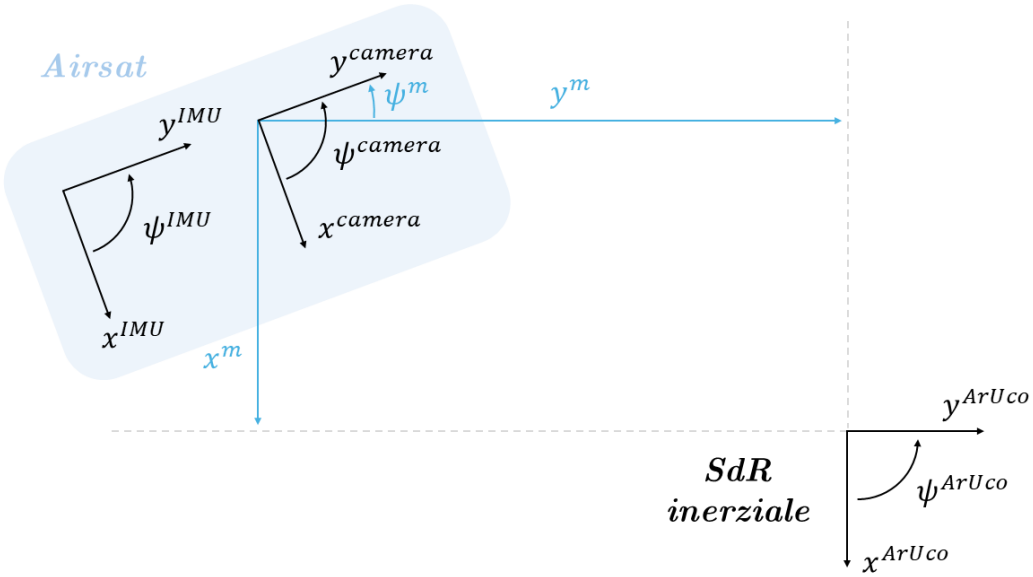


Figura 6.4: Rappresentazione schematica dei sistemi di riferimento

Il nodo, infine, pubblica la posa ricavata sul topic `aruco_marker_pose` e diffonde

simultaneamente sul topic `target_visible` un flag di presenza/assenza del marker. Quest'ultimo segnale ha una funzione rilevante anche sotto il profilo della sicurezza: il controllore LQI considera valido il bersaglio soltanto se ha ricevuto recentemente una condizione di visibilità positiva e forza il comando della reaction wheel a zero quando il marker non è visibile. La misura ArUco non viene tuttavia utilizzata da sola, bensì integrata in un algoritmo di sensor fusion come l'Extended Kalman Filter di cui si parla nel prossimo paragrafo.

6.5 Nodo Sensor Fusion con filtro di Kalman

La stima dello stato dinamico di AirSat è affidata al Kalman Filter tramite lo script `ekf_state_estimator_node.py`, che ha il compito di unire i dati dinamici del sistema inerziale con le misure del sistema di visione artificiale.

Definizione dello spazio degli stati

Lo stato stimato del sistema è rappresentato dal vettore a 9 dimensioni:

$$SS = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \psi & \dot{\psi} & b_{a_x} & b_{a_y} & b_{g_z} \end{bmatrix}^T \tag{6.6}$$

dove:

- x, y: posizione 2D nel SdR inerziale,
- $\dot{x}, \dot{y}$: velocità lineari nel SdR inerziale,
- ψ: angolo di *yaw* nel SdR inerziale,
- $\dot{\psi}$: velocità angolare *yawrate* nel SdR inerziale,
- b_{a_x}, b_{a_y}: bias dell'accelerometro definiti,
- b_{gz}: bias del giroscopio.

Si precisa, che la velocità angolare, non costituisce un vero e proprio stato per il filtro di Kalman (come visto nel Capitolo 5), bensì, dopo essere stato corretto dal bias e usato come input nella fase di prediction dell'algoritmo, viene aggiunto a posteriori nello stato stimato dell'odometria del free-flyer.

Fase di propagazione: modello di processo

All'arrivo di ogni campione IMU, ad intervalli di tempo Δt, le letture grezze dell'accelerometro a_x^m, a_y^m e del giroscopio g_z^m vengono compensate rimuovendo la stima corrente dei bias:

$$a_x^c = a_x^m - b_{a_x} \tag{6.7}$$

$$a_y^c = a_y^m - b_{a_y} \tag{6.8}$$

$$\dot{\psi}^c = g_z^m - b_{gz} \tag{6.9}$$

Le accelerazioni nel frame Airsat vengono ruotate nel frame Inerziale tramite l'angolo di yaw corrente ψ (Figura 6.4):

$$a_x^I = \cos(\psi)a_x^c - \sin(\psi)a_y^c \tag{6.10}$$

$$a_y^I = \sin(\psi)a_x^c + \cos(\psi)a_y^c \tag{6.11}$$

L'integrazione temporale dello stato predittivo $\hat{SS}_{k+1} = f(SS_k, u_k)$ avviene tramite le equazioni cinematiche:

$$\hat{SS}_k = \begin{bmatrix} x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}a_x^I \Delta t^2 \\ y_{k-1} + \dot{y}_{k-1}\Delta t + \frac{1}{2}a_y^I \Delta t^2 \\ \dot{x}_{k-1} + a_x^I \Delta t \\ \dot{y}_{k-1} + a_y^I \Delta t \\ \psi_{k-1} + g_z^c \Delta t \\ b_{a_x,k-1} \\ b_{a_y,k-1} \\ b_{gz,k-1} \end{bmatrix} \tag{6.12}$$

La matrice Jacobiana del modello di processo $F \in \mathbb{R}^{8\times 8}$ viene calcolata analiticamente come:

$$F_k = \frac{\partial f}{\partial SS} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & F_{1,5} & 0 & F_{1,7} & F_{1,8} & 0 \\ 0 & 1 & 0 & \Delta t & F_{2,5} & 0 & F_{2,7} & F_{2,8} & 0 \\ 0 & 0 & 1 & 0 & F_{3,5} & 0 & F_{3,7} & F_{3,8} & 0 \\ 0 & 0 & 0 & 1 & F_{4,5} & 0 & F_{4,7} & F_{4,8} & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.13}$$

I cui elementi non costanti derivano dalla rotazione dell'accelerazione:

$$F_{1,5} = \tfrac{1}{2}\Delta t^2(-\sin\psi\, a_x^c - \cos\psi\, a_y^c) \tag{6.14}$$
$$F_{2,5} = \tfrac{1}{2}\Delta t^2(\cos\psi\, a_x^c - \sin\psi\, a_y^c) \tag{6.15}$$
$$F_{3,5} = \Delta t(-\sin\psi\, a_x^c - \cos\psi\, a_y^c) \tag{6.16}$$
$$F_{4,5} = \Delta t(\cos\psi\, a_x^c - \sin\psi\, a_y^c) \tag{6.17}$$
$$F_{1,7} = -\tfrac{1}{2}\Delta t^2\cos\psi \tag{6.18}$$
$$F_{1,8} = \tfrac{1}{2}\Delta t^2\sin\psi \tag{6.19}$$
$$F_{2,7} = -\tfrac{1}{2}\Delta t^2\sin\psi \tag{6.20}$$
$$F_{2,8} = -\tfrac{1}{2}\Delta t^2\cos\psi \tag{6.21}$$
$$F_{3,7} = -\Delta t\cos\psi \tag{6.22}$$
$$F_{3,8} = \Delta t\sin\psi \tag{6.23}$$
$$F_{4,7} = -\Delta t\sin\psi \tag{6.24}$$
$$F_{4,8} = -\Delta t\cos\psi \tag{6.25}$$

La predizione della covarianza dell'errore viene aggiornata con:

$$\hat{P}_k = F_{k-1}P_{k-1}F_{k-1}^T + V_{k-1}QV_{k-1}^T \tag{6.26}$$

dove è la matrice di covarianza del rumore di processo $Q \in \mathbb{R}^{6\times6}$ è:

$$Q = \begin{bmatrix} q_{ax} & 0 & 0 & 0 & 0 & 0 \\ 0 & q_{ay} & 0 & 0 & 0 & 0 \\ 0 & 0 & q_{gz} & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{b_{a_x}} & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{b_{a_y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{b_{gz}} \end{bmatrix} \tag{6.27}$$

e $V \in \mathbb{R}^{8\times6}$ è la matrice di propagazione del rumore di processo

$$V_k = \frac{\partial f}{\partial w} = \begin{bmatrix} \cos\psi\frac{\Delta t^2}{2} & -\sin\psi\frac{\Delta t^2}{2} & 0 & -\cos\psi\frac{\Delta t^2}{2} & \sin\psi\frac{\Delta t^2}{2} & 0 \\ \sin\psi\frac{\Delta t^2}{2} & \cos\psi\frac{\Delta t^2}{2} & 0 & -\sin\psi\frac{\Delta t^2}{2} & -\cos\psi\frac{\Delta t^2}{2} & 0 \\ \cos\psi\Delta t & -\sin\psi\Delta t & 0 & -\cos\psi\Delta t & \sin\psi\Delta t & 0 \\ \sin\psi\Delta t & \cos\psi\Delta t & 0 & -\sin\psi\Delta t & -\cos\psi\Delta t & 0 \\ 0 & 0 & \Delta t & 0 & 0 & -\Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.28}$$

con w vettore dei process noise.

Fase di update: modello di misura

Quando è disponibile una nuova misura $z_k = h(SS)$ della posa ArUco:

$$z_k = \begin{bmatrix} x^m \\ y^m \\ \psi^m \end{bmatrix} \tag{6.29}$$

i cui valori sono già trasformati nel frame inerziale in precedenza, il filtro esegue la correzione dello stato. La matrice di misura $H \in \mathbb{R}^{3\times8}$ mappa linearmente lo stato nello spazio delle misure:

$$H = \frac{\partial h}{\partial SS} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.30}$$

L'innovazione definita come la differenza tra la misura reale e la predizione è calcolata:

$$i_k = z_k - H \cdot \hat{SS}_k = \begin{bmatrix} x^m - \hat{x}_k \\ y^m - \hat{y}_k \\ \text{wrap}(\psi^m - \hat{\psi}_k) \end{bmatrix} \tag{6.31}$$

dove la funzione wrap assicura che l'errore angolare rimanga limitato nell'intervallo $[-\pi, \pi]$. Il guadagno di Kalman K_k e lo stato corretto SS_k sono calcolati tramite la formulazione classica:

$$S_k = H\hat{P}_k H^T + R \tag{6.32}$$

$$K_k = \hat{P}_k H^T S_k^{-1} \tag{6.33}$$

$$SS_k = \hat{SS}_k + K_k \cdot i_k \tag{6.34}$$

$$P_k = (I^{8\times8} - K_k H)\hat{P}_k (I^{8\times8} - K_k H)^T + K_k R K_k^T \tag{6.35}$$

dove $R \in \mathbb{R}^{3\times3}$ e contiene la covarianza del rumore associato alle misure della camera:

$$R = \begin{bmatrix} r_x & 0 & 0 \\ 0 & r_y & 0 \\ 0 & 0 & r_\psi \end{bmatrix} \tag{6.36}$$

Nella Tabella di seguito si riportano i valori per le matrici di covarianza di rumore di processo Q e di misura R, ovvero gli stessi individuati nel digital twin.

Matrice di Rumore di Processo Q			
Parametro	**Simbolo**	**Valore**	**Descrizione**
`q_sigma_ax`	q_{ax}	$5.0 \cdot 10^{-4}$	Varianza del rumore dell'accelerometro su x
`q_sigma_ay`	q_{ay}	$5.0 \cdot 10^{-4}$	Varianza del rumore dell'accelerometro su y
`q_sigma_gz`	q_{gz}	$5.0 \cdot 10^{-7}$	Varianza del rumore del giroscopio
`q_sigma_bax`	$q_{b_{a_x}}$	$1.0 \cdot 10^{-8}$	Varianza del bias dell'accelerometro su x
`q_sigma_bay`	$q_{b_{a_y}}$	$1.0 \cdot 10^{-8}$	Varianza del bias dell'accelerometro su y
`q_sigma_bgz`	$q_{b_{gz}}$	$1.0 \cdot 10^{-10}$	Varianza del bias del giroscopio
Matrice di Rumore di Misura R			
Parametro	**Simbolo**	**Valore**	**Descrizione**
`r_aruco_x`	r_x	$5.0 \cdot 10^{-7}$	Varianza di misura ArUco su x^m
`r_aruco_y`	r_y	$5.0 \cdot 10^{-5}$	Varianza di misura ArUco y^m
`r_aruco_yaw`	r_ψ	$5.0 \cdot 10^{-3}$	Varianza di misura ArUco ψ^m

Tabella 6.7: Parametri di Process Noise Q e Measurement Noise R

Logica di stima in dead reckoning

Se la telecamera perde la visibilità del marker ArUco, a causa di ostruzioni o riflessi, il nodo smette temporaneamente di eseguire la fase di update. In questo scenario, l'EKF opera in modalità **dead reckoning**, ovvero puro calcolo integrativo delle equazioni di prediction basato sulle sole misure IMU. La covarianza P crescerà gradualmente riflettendo l'aumento dell'incertezza sulla posizione e sullo yaw, per poi contrarsi rapidamente non appena un nuovo campione ArUco riattiverà la fase di correzione.

6.6 Nodo Guidance

Il nodo Guidance è contenuto nel codice `carrot_guidance_node.py`, ha il compito di generare il riferimento per il sottosistema di controllo, pubblicando in modo periodico un setpoint coerente con lo stato stimato del veicolo e con i parametri di navigazione impostata nel file `guidance_params.yaml`. Il nodo riceve lo stato filtrato dal topic `/airsat/odometry/filtered` e, a partire da esso, costruisce

il riferimento di assetto che viene poi inoltrato al controllore attraverso il topic `/airsat/reference/setpoint`.

Logica interna

La logica del nodo è deliberatamente semplice e robusta: a ogni ciclo del timer, con periodo fissato da `publish_period`, viene pubblicato un riferimento che può essere fisso oppure derivato dallo *statespace* ricevuto. Nel caso attuale, considerando la dinamica permessa dal banco di prova, la configurazione prevede `use_fixed_reference = true`, quindi il nodo emette sempre il riferimento parametrico definito da `fixed_x`, `fixed_y`, `fixed_yaw`, `fixed_vx`, `fixed_vy` e `fixed_yawrate`. Questo approccio risulta adatto allo stato attuale del progetto, ma conserva la possibilità di implementare calcoli di traiettorie live e evitamento ostacoli nel proseguo dello sviluppo del free-flyer. In Tabella 6.8 sono riportati i principali parametri del nodo.

Parametro	**Default**	**Descrizione**
state_topic	`/airsat/odometry/filtered`	Topic dello stato stimato in ingresso.
reference_topic	`/airsat/reference/setpoint`	Topic del riferimento di assetto pubblicato.
publish_period	30 ms	Periodo di pubblicazione del setpoint
max_yaw_rate	0.8 rad/s	Massima velocità angolare di yaw consentita.
max_yaw_accel	1.5 rad/s^2	Massima accelerazione angolare di yaw consentita.
yaw_tolerance	0.01 rad	Tolleranza sull'errore di yaw.
rate_tolerance	0.01 rad/s	Tolleranza sull'errore di velocità angolare.
use_fixed_reference	true	Se true, usa un riferimento fisso predefinito.
fixed_x	0.0 m	Coordinata x del riferimento fisso.
fixed_y	0.0 m	Coordinata y del riferimento fisso.
fixed_yaw	0.0 rad	Yaw del riferimento fisso in radianti.
fixed_vx	0.0 m/s	Velocità x del riferimento fisso.
fixed_vy	0.0 m/s	Velocità y del riferimento fisso.
fixed_yawrate	0.0 rad/s	Velocità angolare di yaw del riferimento fisso.

Tabella 6.8: Parametri del file `guidance_params.yaml`.

6.7 Nodo di controllo LQI

Il nodo di controllo implementato tramite il `lqi_control_node.py` utilizza un controllore di tipo Linear Quadratic Integral (LQI) per regolare l'angolo e la velocità angolare della piattaforma. Il controllo LQI, è una strategia di controllo ottimo utilizzato nella teoria dei controlli moderni. Il suo obiettivo è ottenere un controllore a retroazione che minimizzi una funzione di costo quadratica, costruita per bilanciare la precisione nella regolazione dello stato e l'ampiezza dell'azione di controllo, così da

ottenere il miglior compromesso possibile tra prestazioni e sforzo di controllo. Questo tipo di strategia risulta indicato per il contesto di simulazione del free-flyer in quanto le risorse a bordo risultano limitate e la loro gestione assume un ruolo importante.

Come anticipato in precedenza, date le attuali caratteristiche della postazione di test, in questa parte del workspace ci si concentrerà puramente sull'assetto del free-flyer tralasciando la parte di moto traslatorio, pur mantenendo la predisposizione all'aggiornamento del modulo.

Il nodo di controllo si iscrive ai topic `odometry/filtered` e `reference/setpoint` e pubblica il suo comando di controllo sul topic `actuation/reaction_wheel_cmd`. Di seguito sono riportati i parametri principali di configurazione contenuti all'interno di `control_params.yaml`.

Parametro	**Default**	**Descrizione**
state_topic	/airsat/odometry/filtered	Topic dello stato stimato in ingresso.
reference_topic	/airsat/reference/setpoint	Topic del riferimento di assetto.
command_topic	/airsat/actuation/reaction_wheel_cmd	Topic del comando verso l'attuatore.
target_visible_topic	/airsat/target_visible	Topic booleano della visibilità del target.
eyaw_topic	/airsat/control/eyaw	Topic dell'errore di yaw per i plot.
eyawrate_topic	/airsat/control/eyawrate	Topic dell'errore di velocità angolare per i plot.
tau_body_topic	/airsat/control/tau_body_desired	Topic della coppia body desiderata per i plot.
tau_rw_topic	/airsat/control/tau_rw_command	Topic della coppia commandata alla reaction wheel per i plot.
control_rate	33.0 Hz	Frequenza del loop di controllo in Hz.
K_yaw	0.0025	Guadagno proporzionale sull'errore di yaw.
K_yawrate	0.375	Guadagno sull'errore di velocità angolare.
Ki_yaw	−7.07e−9	Guadagno integrale sull'errore di yaw.
voltage_max	3.3 V	Massimo valore di tensione pubblicabile.
voltage_min	−3.3 V	Minimo valore di tensione pubblicabile.
rw_inertia	0.685e−4 kgm^2	Inerzia equivalente della reaction wheel.
rw_tau_max	0.5e−3 Nm	Saturazione massima della coppia equivalente.
rw_tau_min	−0.5e−3 Nm	Saturazione minima della coppia equivalente.
rw_speed_max_rpm	2960.0 rpm	Velocità massima della reaction wheel in rpm.
rw_zero_deadband	0.05 V	Banda morta sotto la quale la tensione viene forzata a zero.

Tabella 6.9: Parametri del file `control_params.yaml`.

Algoritmo di controllo e legge di regolazione

Il controllore riceve l'odometria stimata dal nodo del filtro di Kalman ed il setpoint di riferimento ψ_{ref} e $\dot{\psi}_{\text{ref}}$ generato dal nodo di guidamce e definisce gli errori:

$$e_\psi = \text{wrap}(\psi_{\text{ref}} - \psi_{\text{est}}) \tag{6.37}$$

$$e_{\dot{\psi}} = \dot{\psi}_{\text{ref}} - \dot{\psi}_{\text{est}}, \tag{6.38}$$

dove la funzione wrap evita salti artificiali dell'angolo nell'intervallo $[-\pi, \pi]$. Il termine integrale dell'errore di yaw viene integrato numericamente nel tempo discreto:

$$x_i(t) = x_i(t-1) + e_\psi \Delta t. \tag{6.39}$$

La coppia di controllo desiderata da applicare sul corpo del satellite è calcolata come:

$$\tau_{\text{body, desired}} = -(K_{\text{yaw}} e_\psi + K_{\text{yawrate}} e_{\dot{\psi}} + K_{i,\text{yaw}} x_i(t)) \tag{6.40}$$

Tale valore viene limitato entro i valori fisici della ruota di reazione (rw_tau_min e rw_tau_max):

$$\tau_{\text{body, sat}} = \text{clamp}(\tau_{\text{body, desired}}, \tau_{\text{rw, min}}, \tau_{\text{rw, max}}) \tag{6.41}$$

Integrazione e dinamica della ruota di reazione

La coppia di controllo effettivo applicato sul corpo del satellite è generato per reazione accelerando la ruota di reazione montata sull'asse di yaw.

$$\tau_{\text{rw, cmd}} = -\tau_{\text{body, sat}}, \tag{6.42}$$

Trascurando gli attriti dinamici nel modello semplificato, la coppia scambiata è legata all'accelerazione angolare della reaction wheel:

$$\alpha_{\text{rw}} = \frac{\tau_{\text{rw, cmd}}}{J_{\text{rw}}}, \tag{6.43}$$

Dove J_{rw} è l'inerzia della reaction wheel. La velocità angolare teorica della ruota ω_{rw} viene stimata integrando l'accelerazione mediante la regola dei trapezi:

$$\omega_{\text{rw}}(k) = \omega_{\text{rw}}(k-1) + \frac{1}{2}\Delta t(\alpha_{\text{rw}}(k-1) + \alpha_{\text{rw}}(k)) \tag{6.44}$$

La velocità espressa in giri al minuto (rpm) è quindi:

$$n_{\text{rw}} = \omega_{\text{rw}} \cdot \frac{60}{2\pi} \tag{6.45}$$

Se n_{rw} supera in valore assoluto la velocità massima del motore, essa viene saturata a `rw_speed_max_rpm`.

Conversione velocità-tensione e saturazione

Il driver del motore accetta in ingresso un comando in tensione, perciò il nodo calcola la tensione corrispondente alla velocità desiderata della ruota tramite una mappatura lineare che tiene conto dell'intervallo totale di tensione 4.9 V [15]:

$$V_{\text{cmd}} = \text{sgn}(n_{\text{rw}}) \cdot \left(\frac{|n_{\text{rw}}| - n_{\text{min}}}{n_{\text{max}} - n_{\text{min}}} \cdot 4.9V \right) \tag{6.46}$$

La tensione calcolata viene, infine, limitata nell'intervallo di sicurezza $[V_{\text{min}}, V_{\text{max}}]$ e imposta a zero nel caso risulti all'interno della deadband minima di azionamento, in modo da migliorare la stabilità all'equilibrio. I valori `voltage_max`, `voltage_min` e `rw_zero_deadband` sono riportati in Tabella 6.9.

Il comando viene pubblicato sul topic `/airsat/actuation/reaction_wheel_cmd` con un messaggio custom ReactionWheelCommand contenente la tensione V_{set} e la flag di abilitazione enable = True. Nel caso in cui vengano a mancare l'odometria o il setpoint, e nel caso in cui non sia più visibile alcun target, la flag viene immediatamente reimpostata su False. Lo schema logico di questo nodo è riprodotto in Figura 6.5.

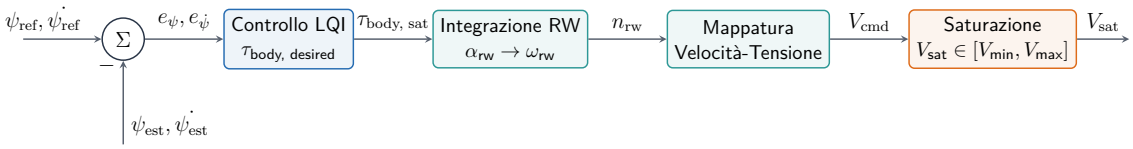


Figura 6.5: Schema a blocchi del flusso di controllo LQI e attuazione Reaction Wheel.

6.8 Nodo microROS per l'attuazione

L'ultimo nodo prima richiudere il ciclo sul microcontrollore è `actuator_command_bridge_node.py`. Il nodo sottoscrive il topic */airsat/actuation/reaction_wheel_cmd* ed esegue le seguenti operazioni:

1. Estrae la tensione desiderata V_{sat} ed applica un ulteriore filtro di sicurezza, imponendo una saturazione hardware al valore di tensione nell'intervallo fisso $[-3.3\text{ V}, 3.3\text{ V}]$ associato alle specifiche massime del DAC del microcontrollore.

2. Se la flag enable del messaggio è False, la tensione di uscita viene immediatamente forzata a 0.0 V.

3. Pubblica il valore scalare risultante sul topic standard */airsat/rw/voltage_cmd* ad una frequenza costante di 30 Hz.

A valle del nodo, il microcontrollore ESP32 riceve il comando sul topic */airsat/rw/voltage_cmd* ed applica fisicamente la tensione al driver DEC Module 505, come spiegato nella Sezione 6.2.

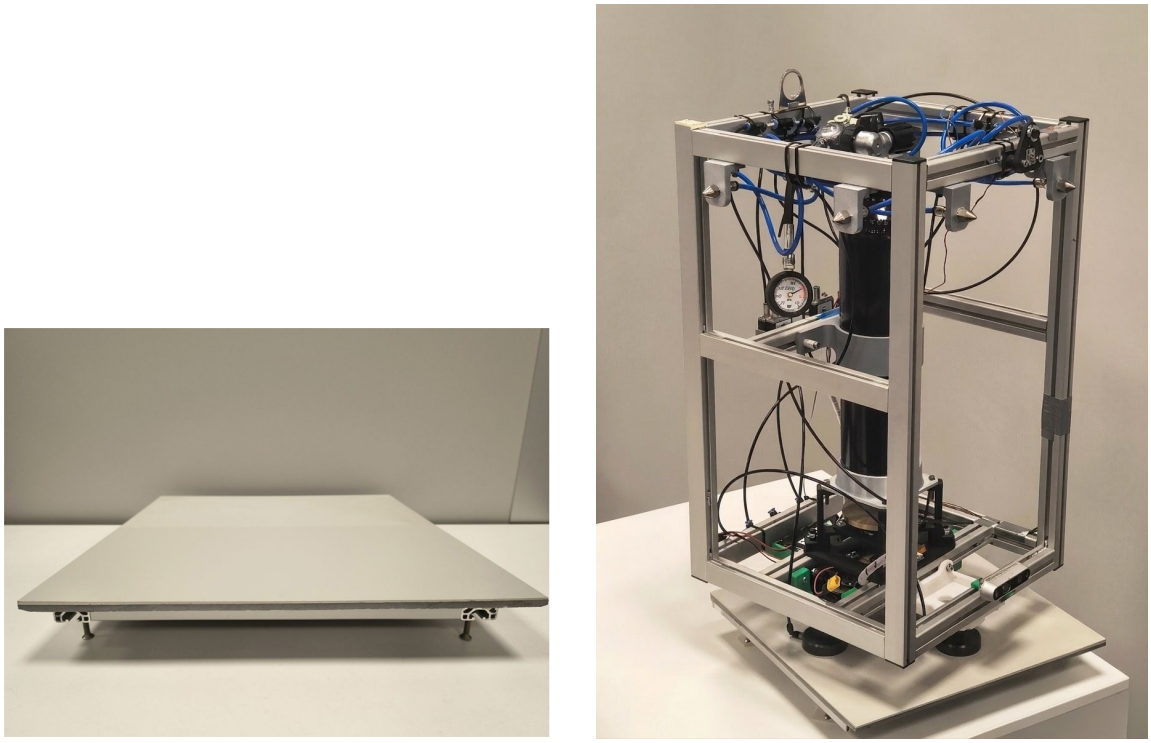
Capitolo 7

Test e Risultati

In questo capitolo si presentano i risultati sperimentali con l'obiettivo di validare l'implementazione del nuovo sistema di visione artificiale e dell'algoritmo di Sensor Fusion. Come per il precedente lavoro, si evidenzia che la superficie in resina inizialmente prevista per le prove del free-flyer presenta alcune pendenze indesiderate che non consentono il test con l'uso dei thruster durante la fase di sperimentazione. Per questo motivo, le prove descritte in questo capitolo sono condotte su una superficie liscia di dimensioni ridotte, per garantire il corretto funzionamento degli air bearings, ma si limiteranno all'investigazione delle sole grandezze del moto rotatorio, in quanto è l'unico effettivamente praticabile.

7.1 Setup del testbench

Il setup sperimentale utilizzato per le prove di laboratorio è composto dal prototipo di cui si è discusso nel Capitolo 3 e dalla superficie liscia regolabile mostrata in Figura 7.1.



(a) Superficie liscia utilizzata per i test

(b) Test bench sperimentale

Figura 7.1: Setup sperimentale per le prove di laboratorio

Infine viene sfruttato un cavalletto posto di fronte al free-flyer per sostenere il marker ArUco e renderlo rilevabile dalla camera (Figura 7.2).

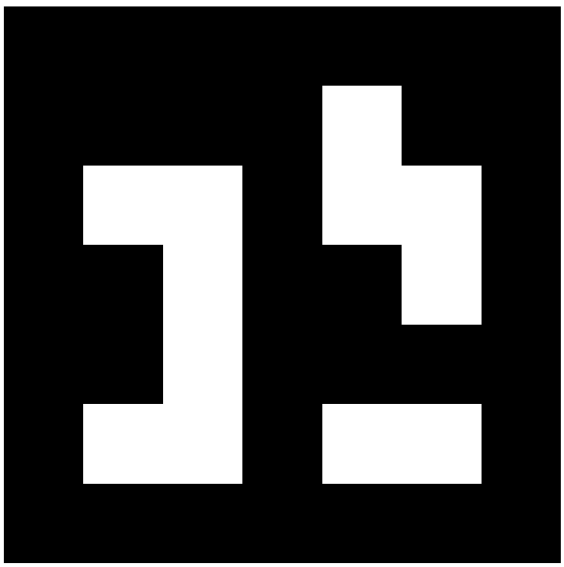


Figura 7.2: Marker ArUco utilizzato nel test

7.2 Validazione filtro di Kalman

Nel capitolo dedicato al digital twin è stato dimostrato che i sensori virtuali, modellati all'interno del digital twin, riproducono con elevata fedeltà il comportamento dei sensori fisici installati a bordo del robot. La corrispondenza tra il prototipo virtuale e quello reale costituisce il presupposto fondamentale per poter utilizzare il digital twin non solo come strumento di progettazione, ma anche come piattaforma di verifica delle prestazioni degli algoritmi di stima come il filtro di Kalman implementati sul sistema fisico. Nello stesso capitolo, l'implementazione del filtro di Kalman ha fornito risultati robusti, come evidenziato dai valori di RMSE ottenuti nel confronto tra le traiettorie stimate e quelle di riferimento ideali; questo dimostra che l'algoritmo di sensor fusion, è in grado di ricostruire con accuratezza lo stato del sistema anche in presenza di rumore realistico introdotto nei sensori virtuali.

Questi risultati rappresentano gli elementi alla base della strategia di validazione proposta in questa sezione, poiché confermano che il filtro all'interno del digital twin può essere considerato rappresentativo del comportamento ideale atteso dall'algoritmo.

Sulla base di queste due evidenze, rappresentate dalla fedeltà dei sensori virtuali e dall'accuratezza del filtro di Kalman simulato, è possibile impostare una procedura di validazione incrociata in due fasi per verificare la robustezza del filtro di Kalman implementato sulla testbench reale:

1. nella prima fase verrà condotta una prova sperimentale di rotazione sul banco fisico, durante la quale il filtro di Kalman reale stimerà l'assetto del robot a partire dai dati grezzi acquisiti in tempo reale,

2. nella seconda fase, i medesimi dati raw, registrati durante la prova reale, verranno riutilizzati come dati grezzi in ingresso del filtro nel digital twin. In questo modo sarà possibile eseguire una simulazione del filtro di Kalman virtuale operante sugli stessi identici dati grezzi elaborati dal filtro reale.

Questa impostazione consente un confronto diretto tra le due stime di assetto, poiché entrambi i filtri partono dagli stessi dati grezzi. Calcolando l'RMSE tra l'uscita del filtro reale e quella del filtro virtuale, di cui è già verificata la robustezza, è possibile confermare l'equivalenza dei due sistemi di stima. Infatti, un valore di RMSE contenuto tra le due stime rappresenta una prova diretta della correttezza dell'algoritmo reale. Tale approccio metodologico fornisce un elemento di verifica basato sul digital twin sviluppato in precedenza.

7.3 Risultati sperimentali

Il confronto tra l'EKF eseguito offline nel digital twin e l'implementazione online in ROS2 evidenzia una sostanziale coerenza delle stime di ψ e $\dot{\psi}$ (Figure 7.3 e 7.4), pur in assenza di una sovrapposizione perfetta tra le rispettive traiettorie. Il risultato rientra nello scenario atteso, poiché l'uguaglianza del modello teorico e dei parametri nominali non garantisce l'identità della sua evoluzione numerica discreta. L'EKF è un algoritmo ricorsivo e non lineare: lo stato stimato, la matrice di covarianza e il guadagno di Kalman vengono aggiornati a partire dai valori ottenuti al passo precedente, di conseguenza, anche differenze limitate nelle condizioni delle grandezze citate, dovute a piccoli errori numerici, possono determinare scostamenti locali tra le due implementazioni. A riprova di ciò, si nota che lo scostamento maggiore si ottiene nella misura dell'angolo ψ che nella fase di prediction del filtro viene propagata utilizzando l'integrazione della velocità angolare $\dot{\psi}$, introducendo ulteriore approssimazione numerica.

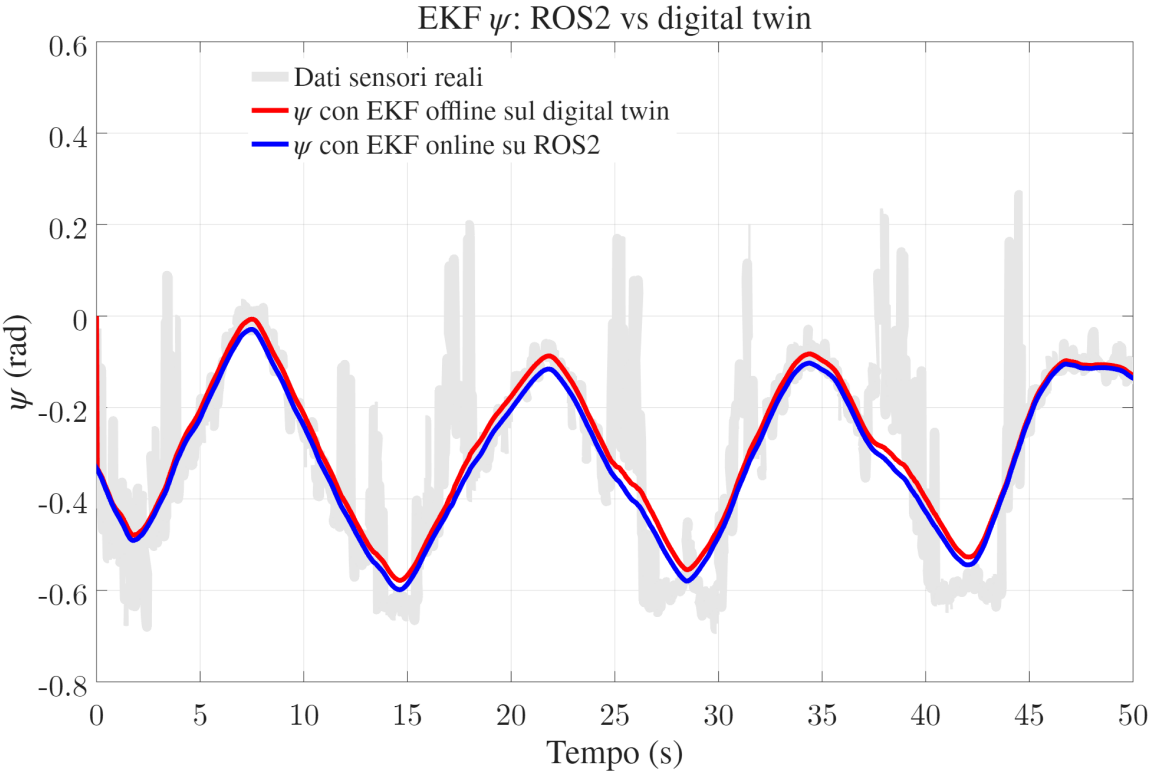


Figura 7.3: Confronto dei risultati dell'EKF per la stima dell'angolo ψ

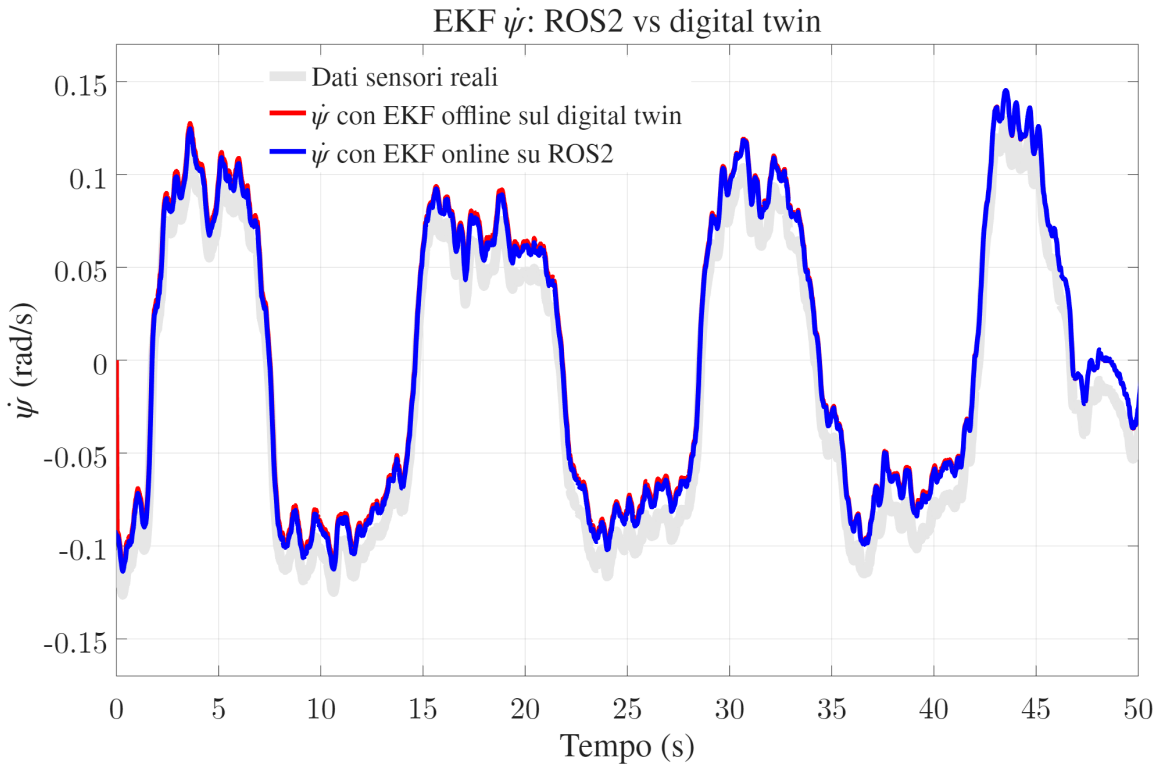


Figura 7.4: Confronto dei risultati dell'EKF per la stima della velocità angolare $\dot{\psi}$

I valori iniziali impostati per la matrice di covarianza P e il vettore di stato SS sono riprodotti a partire dalle misurazioni reali per poter ricreare le stesse condizioni di partenza del filtro e ridurne il più possibile la differenza a meno di approssimazioni numeriche. Questo passaggio risulta importante per evitare un overshoot che influenzi il confronto nelle fasi iniziali e ottenere un valore di RMSE falsato, derivato da questa dinamica.

Nel complesso, i risultati confermano il corretto comportamento dell'EKF online. Entrambe le implementazioni ricostruiscono le medesime variazioni dinamiche dell'assetto e della velocità angolare, ne individuano correttamente gli estremi e restituiscono stime regolari, prive di instabilità, di deriva progressiva o ritardi di alcun tipo. La ridotta differenza residua tra la soluzione ROS2 e quella offline è compatibile con errori di natura numerica e di inizializzazione, e non compromette la validità della trasposizione dell'algoritmo dal digital twin al sistema sperimentale. Come visibile in Figura 7.5, la RMSE tra il filtro online e quello offline si discosta mediamente di 0.0286 rad per l'angolo ψ, equivalenti a 1.64°, e circa 0.0174 rad/s sulla velocità angolare $\dot{\psi}$, equivalenti a 1°/s. Nel complesso, considerando la peggiore delle ipotesi dove i valori di RMSE tra il *digital twin* e la *ground truth* e tra il *digital twin* e ROS2 siano cumulativi, risulterebbe in totale un RMSE medio di circa 0.031 rad sull'angolo e 0.018 rad/s sulla velocità angolare, ovvero 1.78° e 1°/s.

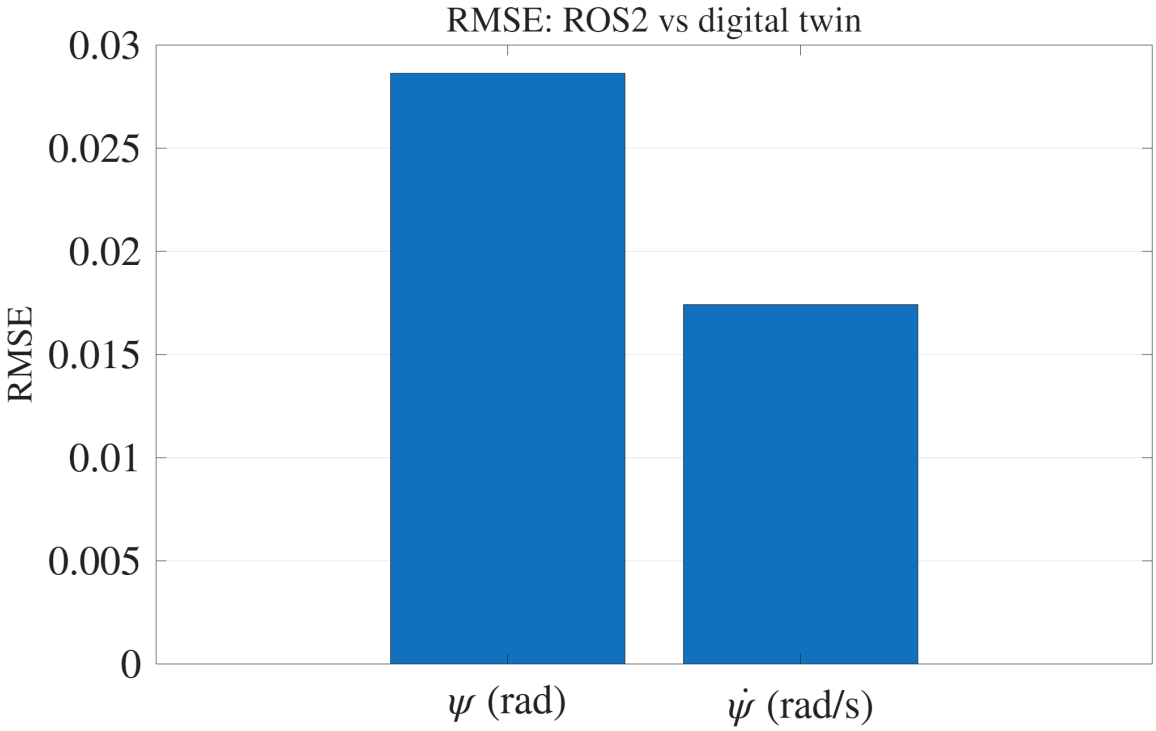


Figura 7.5: Confronto RMSE delle grandezze stimate tra l'implementazione su ROS2 e offline

Capitolo 8

Conclusioni

In questo capitolo finale si espongono le conclusioni tratte dal presente lavoro di tesi, riassumendo i passaggi chiave e i principali risultati ottenuti.

L'obiettivo di implementare un sistema di visione artificiale e un algoritmo di sensor fusion è stato conseguito. L'attività svolta comprende la definizione del modello digitale dei sensori, di un filtro di Kalman e della loro implementazione sperimentale sul sistema reale, ponendo particolare attenzione alla corrispondenza tra architettura simulata e architettura fisica.

Un primo risultato riguarda la modellazione dei sensori, attraverso la replicazione delle caratteristiche di rumore ottenute sperimentalmente dai sensori reali. Successivamente i segnali provenienti dalla catena di misura inerziale e visiva del digital twin si sono verificati coerenti con le grandezze fisiche attese, confermando la correttezza delle ipotesi adottate nella rappresentazione del sistema. Un secondo risultato è rappresentato dalla validazione del filtro di Kalman impiegato nel digital twin. I risultati ottenuti hanno evidenziato che l'algoritmo elabora in modo efficace le informazioni sensoriali disponibili, fornendo una stima dello stato corretta e stabile. Infine, è stata verificata la corrispondenza tra il Digital Twin e l'implementazione ROS2 eseguita sul free-flyer reale. Il confronto tra i due sistemi ha mostrato una coerenza dal punto di vista dinamico, dimostrando che il modello digitale costituisce una rappresentazione affidabile della piattaforma sperimentale. Questo risultato conferma l'utilità del Digital Twin come strumento di validazione preliminare delle strategie di stima e controllo prima della loro applicazione sull'hardware fisico.

Nel complesso, il lavoro svolto ha dimostrato la solidità dell'architettura di visione artificiale e sensor fusion proposta e la sua capacità di supportare lo sviluppo di sistemi autonomi per applicazioni spaziali. La piattaforma AirSat, insieme al relativo ambiente digitale, rappresenta quindi una base sperimentale robusta per futuri sviluppi

orientati al perfezionamento dei modelli, al miglioramento delle prestazioni di controllo e all'estensione verso scenari operativi più complessi.

8.1 Lavori futuri

A partire dai risultati ottenuti in questo lavoro, si possono individuare diverse linee di sviluppo future di particolare interesse. In primo luogo, il framework realizzato costituisce una base solida per la progettazione di nuovi algoritmi di GNC, ad esempio estendendo l'attuale architettura verso strategie più avanzate di *obstacle avoidance* e *live path planning*, in modo da consentire la generazione online di traiettorie robuste e adattative in funzione dell'ambiente operativo.

Un'evoluzione naturale del prototipo, inoltre, consiste nell'abilitazione della dinamica traslatoria mediante l'attivazione controllata dei *thrusters*, così da estendere le capacità del sistema oltre il solo moto rotazionale e rendere possibile la sperimentazione di manovre planari più complete.

Infine, un ulteriore sviluppo di rilievo riguarda la migrazione dell'*workspace* ROS2 su un sistema *Raspberry Pi*, soluzione che permetterebbe di integrare in modo fisso l'infrastruttura software all'interno del prototipo, riducendo la dipendenza da calcolatori esterni e migliorando la portabilità dell'intero sistema.

Bibliografia

[1] Intel Corporation. Intel RealSense D400 Series Product Family Datasheet, 2024. Datasheet: https://realsenseai.com/wp-content/uploads/2025/09/Intel-RealSense-D400-Series-Datasheet-October-2025.pdf, consultata il 27 giugno 2026.

[2] Markus Wilde, Casey Clark, and Marcello Romano. Historical survey of kinematic and dynamic spacecraft simulators for laboratory experimentation of on-orbit proximity maneuvers. *Progress in Aerospace Sciences*, 110:100552, 2019.

[3] Richard Zappulla II, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. *Journal of Spacecraft and Rockets*, 54(4):825–839, 2017.

[4] Josep Virgili-Llop, Jerry V. Drew II, and Marcello Romano. Design and parameter identification by laboratory experiments of a prototype modular robotic arm for orbiting spacecraft applications. In *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, Darmstadt, Germany, March 2016.

[5] Marco Sabatini, Marco Farnocchia, and Giovanni B. Palmerini. Design and tests of a frictionless 2d platform for studying space navigation and control subsystems. *Proceedings of the 2012 IEEE Aerospace Conference*, pages 1–12, 2012.

[6] Pedro Roque, Sujet Phodapol, Elias Krantz, Jaeyoung Lim, Joris Verhagen, Frank Jiang, David Dorner, Roland Siegwart, Ivan Stenius, Gunnar Tibert, Huina Mao, Jana Tumova, Christer Fuglesang, and Dimos V. Dimarogonas. Towards open-source and modular space systems with atmos. *IEEE Transactions on Field Robotics*, 2025.

[7] Simone Chesi, Octavio Perez, and Marcello Romano. A dynamic, hardware-in-the-loop, three-axis simulator of spacecraft attitude maneuvering with nanosatellite dimensions. *Journal of Small Satellites*, 2015.

[8] Guglielmo Cervettini, Stefano Pastorelli, Hyeongjun Park, Dae Young Lee, and Marcello Romano. Development and experimentation of a cubesat magnetic attitude control system testbed. *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[9] item Industrietechnik GmbH. Profilato 6 30x24 leggero, naturale, 2026. Pagina prodotto: https://www.item24.com/it-it/profilato-6-30x24-leggero-naturale-60888?length=300&category=tecnica-dei-profilati%2Fprofilati-di-alluminio&gallery=1, consultata il 26 giugno 2026.

[10] MAGER S.r.l. MAGER xDST HPx Series, 2026. Datasheet: https://mager-ab.it/wp-content/uploads/2026/05/MAGER_xDST_HPx-series.pdf, consultata il 27 giugno 2026.

[11] Mx 8212/2 NC 2/2 Solenoid Valve Datasheet. Product datasheet.

[12] Shenzhen CP-Link Electronic Co., Ltd. Smaco S400 Portable Scuba User's Manual. Pagina manuale: https://manuals.plus/m/e69e8604371b8ffe66bc67ed6fc2632ce3c35591d6173d3be29d0c2b44641cb9, consultata il 27 giugno 2026.

[13] RS Components Ltd. RS PRO NiCd Battery Pack Datasheet, 2024. Datasheet: https://assets.rs-online.com/image/upload/v1726762438/Datasheets/01265bfd7e994b97dcbf56f7fa169881.pdf, consultata il 27 giugno 2026.

[14] maxon motor ag. EC 45 flat, brushless, 30 watt Datasheet, 2021. Datasheet: https://www.maxongroup.net.au/medias/sys_master/root/8882562924574/EN-21-295.pdf, consultata il 27 giugno 2026.

[15] maxon motor ag. DEC Module 50/5 Operating Instructions, 2026. Operating instructions: https://www.maxongroup.com/medias/sys_master/root/8834313125918/380200-Operating-Instruction-En.pdf, consultata il 27 giugno 2026.

[16] NASA Glenn Research Center. Rocket Thrust Summary, 2021. https://www.grc.nasa.gov/www/k-12/airplane/rktthsum.html, consultata il 27 giugno 2026.

[17] TDK Electronics. MPU-9250 Product Specification, Accelerometer Specifications, 2016. Datasheet, documento PS-MPU-9250A-01, revisione 1.1, pagina 9: https://www.alldatasheet.com/html-pdf/1132035/TDK/MPU-9250/623/9/MPU-9250.html, consultata il 27 giugno 2026.

[18] TDK Electronics. MPU-9250 Product Specification, Gyroscope Specifications, 2016. Datasheet, documento PS-MPU-9250A-01, revisione 1.1, pagina 8: https://www.alldatasheet.com/html-pdf/1132035/TDK/MPU-9250/553/8/MPU-9250.html, consultata il 27 giugno 2026.

[19] MathWorks. Band-Limited White Noise, 2026. Documentazione Simulink: https://it.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html, consultata il 27 giugno 2026.

[20] Julius S. Bendat and Allan G. Piersol. *Random Data: Analysis and Measurement Procedures*. Wiley-Interscience, New York, 1971.

[21] Leopoldo Jetto, Sauro Longhi, and Giuseppe Venturini. Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 15(2):219–229, 1999.

[22] Fernando Sansò, Chiara Pileggi, and Ludovico Biagi. A tutorial on kalman filtering: The linear, extended and unscented filters, 2024. Tutorial monografico.

[23] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. Chapman and Hall/CRC, Boca Raton, FL, 2 edition, 2011.

[24] Arif Tanju Erdem and Ali Ozer Ercan. Fusing inertial sensor data in an extended kalman filter for 3d camera tracking. *IEEE Transactions on Image Processing*, 24(2):538–553, 2015.

[25] Gabriele Bleser and Didier Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computers & Graphics*, 33(1):59–72, 2009.

[26] J. Z. Sasiadek and P. Hartana. Sensor data fusion using kalman filter. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 19–25, 2000.

[27] Mary B. Alatise and Gerhard P. Hancke. Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter. *Sensors*, 17(10):2164, 2017.

[28] Eric Foxlin. Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS '96)*, pages 185–194, Santa Clara, CA, USA, 1996. IEEE Computer Society.

[29] Marcello Romano, David A. Friedman, and Tracy J. Shay. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. *Journal of Spacecraft and Rockets*, 44(1):164–173, 2007.

[30] Stanford University. Lecture 9: The extended kalman filter, 2008. EE363 course lecture notes.

[31] Arduino S.r.l. Arduino Nano ESP32 Datasheet, 2026. Datasheet: https://docs.arduino.cc/resources/datasheets/ABX00083-datasheet.pdf, consultata il 2 luglio 2026.