POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Sviluppo di un Drone Autonomo per Robogame

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Andrea Bonarini

Tesi di Laurea di:

Alberto Vettolani Matr. 780838

Marco Visconti Matr. 782993

Anno Accademico 2012-2013

Alla mia famiglia
perchè ci siete sempre per me

Il primo ringraziamento va ai miei genitori e a mia sorella che mi hanno sempre sostenuto ed incoraggiato durante tutto il periodo di studio universitario. Durante la tesi non si sono mai tirati indietro alle mie richieste di aiuto. Grazie!
Insieme a loro ringrazio i nonni, gli zii, le cugine e gli amici di famiglia per essersi sempre interessati a me e al mio lavoro.

Un ringraziamento agli amici di sempre che mi sono stati vicini durante la tesi e con cui mi diverto ogni volta. In particolare ringrazio Fuma e Colo per il grandissimo aiuto durante le prove finali del gioco e per il tempo che ci hanno dedicato.
Ringrazio inoltre tutti gli amici dell'università con cui mi trovo benissimo, per i momenti felici e le difficoltà superate insieme durante gli studi.
Vorrei poi ringraziare tutte le persone che durante questo periodo mi hanno chiesto: "Vetto, ma allora vola?". Ho apprezzato molto il vostro appoggio!
Un ringraziamento speciale al mio compagno di tesi e amico, Visco, con cui ho condiviso l'ultimo anno di lavoro, nelle mille difficoltà e nei momenti di gioia.

Ringrazio inoltre l'Oratorio di Merate e l'OSGB Merate che ci hanno permesso di utilizzare liberamente la palestra per le prove del gioco.

Un ringraziamento va poi al professor Bonarini per i consigli e l'aiuto che hanno reso possibile questo lavoro e per avermi avvicinato al mondo dei droni.
In ultimo un ringraziamento a tutti i ragazzi dell'AirLab per l'aiuto e per i bei momenti passati insieme.

Alberto

A mio papà,
sarai per sempre il mio punto di riferimento

Un grazie ai miei genitori, per avermi dato la possibilità di intraprendere questi studi.
Grazie mamma per quest'ultimo anno, sei una persona buona e ti sei dimostrata più forte di tutti quanti.
Un grazie a mia sorella, per la pazienza e il supporto dimostrato durante tutti questi anni.

Ringrazio il mio amico e compagno di avventure Vetto, con cui ho affrontato le gioie e i dolori di questo lavoro.
Un grazie di cuore va a tutte le persone che ho conosciuto qui dentro: siete dei veri amici e mi avete dato più di quanto possiate mai immaginare.
Un grazie particolare a tutte quelle persone che durante quest'ultimo anno mi sono state vicino e che mi hanno aiutato nel momento più difficile di sempre. Non lo dimenticherò mai.

Infine, un doveroso ringraziamento va al professor Andrea Bonarini e a tutti i ragazzi dell'AirLab, per le loro indicazioni e i consigli, che sono stati di ottimo aiuto durante la realizzazione di questo lavoro, e a tutte le persone che hanno in qualche modo contribuito a tutto questo.

Marco

Indice

Sommario

L'innovazione delle tecnologie informatiche e dell'intelligenza artificiale hanno permesso negli ultimi anni enormi progressi nell'ambito dell'interazione tra uomo e robot. In particolare l'interazione è arrivata a un livello di precisione tale da proporre giochi robotici, i cosiddetti Robogame.
Dopo un'analisi dei giochi esistenti, abbiamo ideato un laser-game che prevede l'utilizzo di un drone volante autonomo e uno telecomandato da un uomo. Il drone autonomo è stato interamente costruito nell'ambito di questo lavoro, ai fini di creare una macchina robusta anche per applicazioni future. L'intera tecnologia è stata studiata per effettuare le operazioni nel modo migliore per i fini proposti con la particolarità di essere completamente a bordo del drone. Nessun sensore esterno è fondamentale per il funzionamento autonomo.

I risultati finali mostrano come l'utilizzo di sensori adeguati a bordo di un drone, unito allo studio delle dinamiche di volo, abbiano permesso di realizzare un drone adatto a creare un gioco particolarmente dinamico e divertente.

Parole chiave: Quadricottero, Drone, Autonomo, Robogame.

Abstract

During the last years information technologies and artificial intelligence innovation has allowed huge progress in the interaction between humans and robots. Especially, the interaction is now so precise that it is possible to offer the so called Robogame. We examined the existing games and then we designed a laser-game which consists in using a flying autonomous drone and a remote-controlled one. We built the whole autonomous drone in order to create a solid machine which could also be used for future applications. The whole technology has been worked out in order to fulfil procedures in the best way; it has a peculiarity: it is situated on the drone. No external sensor is essential for the autonomous functioning.

Final outcomes show how using suitable sensors on the drone together with the study of flight dynamics allowed us to build a drone suitable to create a really dynamic and funny game.

Keywords: Quadcopter, Drone, Autonomous, Robogame.

Capitolo 1

Introduzione

> *"La novità ha per noi un fascino al quale difficilmente possiamo resistere."*
>
> Charles de Saint-Évremond

Scopo di questo lavoro è lo sviluppo di un drone autonomo in grado di realizzare un gioco robotico (Robogame [17]).
Il gioco consiste in una rivisitazione dell'ormai classico laser game, in cui un giocatore umano controlla un drone telecomandato il quale deve fuggire dal drone autonomo costruito.

L'area dei robogame è quella parte della Robotica che studia l'interazione tra persone e robot autonomi in attività ludiche. In questo ambito, alle problematiche tipiche della robotica autonoma (in particolare mobile) si aggiungono quelle dell'interazione persona-robot (Human Robot Interaction - HRI), in cui il robot, per svolgere il suo compito, deve coinvolgere la persona in un'attività interessante e divertente. Questo compito richiede particolare attenzione alle modalità di interazione.

Nei robogame, il giocatore può interagire direttamente oppure attraverso un avatar, come nel caso che abbiamo trattato.
È molto importante che il robot abbia una risposta tempestiva e intelligente per rendere il gioco interessante.

I multirotori, cioè velivoli dotati di almeno tre motori, stanno assumendo sempre più importanza soprattutto nell'ambito delle riprese in volo, ma anche per l'esplorazione e per il semplice divertimento nel pilotarli.
Tra i multirotori esistenti abbiamo scelto di sviluppare ex-novo un drone dotato di quattro motori (quadricottero) che garantisca un buon rapporto

qualità/prezzo in termini di stabilità e capacità di carico, ed offra buone capacità di calcolo per elaborare dati dai sensori e dalla camera e guadagnare una piena autonomia.

Il robogame che abbiamo sviluppato consiste in un drone autonomo che insegue e spara colpi laser a uno telecomandato, il quale deve evitare di essere colpito. Ideato per ambienti chiusi, questo gioco è particolarmente adatto per spazi abbastanza ampi, al fine di sfruttare al meglio la velocità di volo dei droni.
L'interfaccia utente è un telefono su cui è installata un'applicazione per il controllo del drone telecomandato. Per gestire il drone autonomo viene utilizzato un software modulare, completamente residente a bordo.

Il drone autonomo, interamente costruito nell'ambito della nostra tesi, è collegato a un PC a terra, il cui unico scopo è quello di ricevere le informazioni necessarie per il monitoraggio del volo. È inoltre possibile inviare alcuni comandi da PC al drone per la gestione del gioco.

In questo lavoro, abbiamo ideato un gioco in cui le dinamiche di volo, le metodologie hardware adottate e le strategie comportamentali del drone autonomo hanno lo scopo di coinvolgere e di far divertire il più possibile il giocatore.
In particolare è stato necessario creare una struttura HW/SW in grado di analizzare in tempo reale numerosi dati provenienti da svariati sensori tra cui una telecamera IR (WiiCam), un sonar, una webcam, un magnetometro e alcuni sensori di prossimità.

I risultati ottenuti mostrano come sia possibile permettere ad un drone autonomo di seguire le regole di un gioco altamente dinamico qual'è un laser game, attraverso l'integrazione di una grande quantità di dati ed elaborati dal computer di bordo.

1.1 Struttura tesi

Di seguito è riportata una breve descrizione del contenuto di ciascun capitolo della tesi:

- **Capitolo 2** *Stato dell'arte:* viene illustrato il funzionamento dei quadricotteri e i loro utilizzi. Inoltre vengono presentati gli ultimi sviluppi nell'ambito dei robogame.

- **Capitolo 3** *Struttura del gioco:* vengono presentate le regole del gioco e gli agenti coinvolti.
- **Capitolo 4** *Architettura hardware:* vengono spiegati i componenti hardware utilizzati per la realizzazione del quadricottero autonomo e le scelte concettuali compiute per lo sviluppo del gioco.
- **Capitolo 5** *Architettura software:* viene trattata la struttura software del progetto, in particolare la comunicazione HW/SW.
- **Capitolo 6** *Logica di gioco:* vengono esposti gli algoritmi per il volo autonomo e per le varie fasi di gioco.
- **Capitolo 7** *Prove sperimentali:* viene qui analizzato il funzionamento delle varie fasi di volo e del gioco, con test specifici.
- **Capitolo 8** *Conclusioni:* riporta i risultati finali ottenuti e illustra i possibili sviluppi futuri.

Capitolo 2

Stato dell'arte

"La scienza di oggi è la tecnologia di domani."

Edward Teller

Lo sviluppo tecnologico ha portato negli ultimi anni ad avere una notevole crescita nell'ambito della robotica e dell'intelligenza artificiale.
È sempre più comune l'utilizzo di robot per migliorare lo stile di vita applicando queste scienze in svariati campi come quello civile, industriale o medico. Specialmente grazie alla riduzione dei costi e a tecnologie più sicure, la robotica sta muovendo i suoi primi passi anche all'interno delle case, perfino per quanto riguarda il semplice divertimento.

Una delle nuove frontiere della robotica è, infatti, l'interazione uomo-robot con scopo di intrattenimento. Telecomandare robot è stato uno dei primi metodi per interagire con essi, pilotando piccole auto, aerei o elicotteri.
Un altro tipo di interazione ludica con strumenti tecnologici si è sviluppata attraverso i videogiochi. In questo caso si gioca contro una macchina attraverso l'utilizzo di uno schermo e di opportune interfacce.

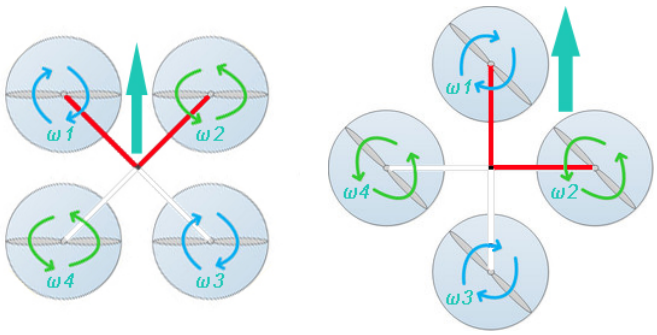
La fusione di questi due tipi di gioco, ha dato vita a quelli che vengono definiti con il termine Robogame. I Robogame non sono altro che giochi in cui vi è una partecipazione fisica del giocatore non più attraverso lo schermo, ma con un oggetto reale, mediante l'impiego di uno o più robot.
La partecipazione al gioco può avvenire in due modi, utilizzando un robot telecomandato (avatar) oppure partecipando direttamente. In entrambi i casi, sia giocatore umano sia robot autonomo dovranno rispettare le regole del gioco. È quindi necessario che il gioco sia stimolante e divertente e che il robot autonomo si comporti nella maniera più adeguata possibile per creare un interesse, che può essere attivato da un certo livello di sfida.

Possono essere utilizzati diversi tipi di robot, distinguibili essenzialmente in robot di terra, d'acqua e volanti. I robot di terra possono muoversi su ruote, cingoli oppure gambe. I robot d'acqua sono barche di vario tipo. I robot volanti autonomi rientrano nella categoria dei droni. In particolare stiamo assistendo a un enorme diffusione dei multirotori, nello specifico di quadricotteri, che con la loro facilità di utilizzo, attraggono molto sia hobbiysti sia professionisti in svariati settori, dalle riprese video, al monitoraggio, al trattamento agricolo.

L'intento di questa tesi è stato quello di sviluppare un Robogame, capace di divertire il giocatore non solo grazie alle dinamiche del gioco, ma anche attraverso il pilotaggio di un robot volante. Il tutto inserendo elementi d'innovazione come lo sviluppo ex novo di un quadricottero con tecnologia a bordo in grado senza alcun strumento esterno di volare autonomamente e reagire correttamente alle fasi di un gioco.

2.1 Quadricottero

Il quadricottero è un multirotore dotato di quattro motori, posti alle estremità delle braccia che vanno a formare una croce. I motori adiacenti ruotano nella direzione opposta. Si tratta di un sistema robotico senza pilota a decollo e atterraggio verticale. Utilizzano un sistema di controllo che si basa sulla lettura dei dati provenienti dalla IMU (Inertial Measurement Unit) grazie al quale è possibile un volo stabilizzato. Nella Figura 2.1 sono rappresentate le configurazioni possibili per un quadricottero, cioè a 'X' o a '+'.



(a) QuadX (b) Quad+

Figura 2.1: Configurazioni quadricottero

Il sistema di controllo si basa su quattro canali: throttle, pitch, roll e yaw

(Figura 2.2).

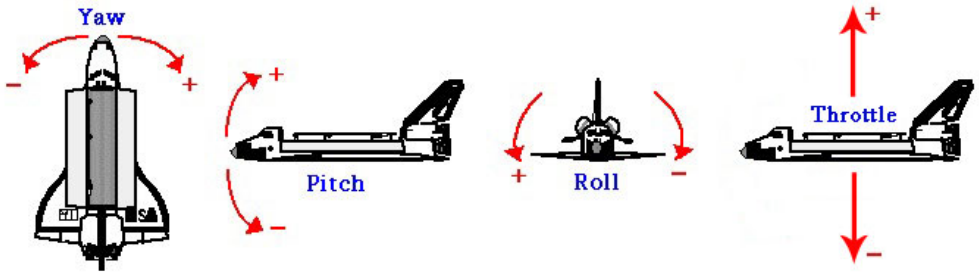


Figura 2.2: roll, pitch, yaw e throttle

Consideriamo $\omega_1, \omega_2, \omega_3$ e ω_4 come le velocità angolari di ciascun motore nel caso della configurazione a 'X' (Figura 2.1.a). Il throttle aumenta ($\Delta_A \in \mathbb{R}^+$) o diminuisce ($\Delta_A \in \mathbb{R}^-$) le velocità di tutti e quattro i motori, causando la salita o la discesa del quadricottero.

$$\begin{cases} \omega_1 = \omega_1 + \Delta_A \\ \omega_2 = \omega_2 + \Delta_A \\ \omega_3 = \omega_3 + \Delta_A \\ \omega_4 = \omega_4 + \Delta_A \end{cases}$$

Il pitch permette al quadricottero di andare in avanti ($\Delta_A, \Delta_B \in \mathbb{R}^+$) o indietro ($\Delta_A, \Delta_B \in \mathbb{R}^-$), aumentando o diminuendo la velocità dei motori posteriori rispetto a quelli anteriori.

$$\begin{cases} \omega_1 = \omega_1 - \Delta_B \\ \omega_2 = \omega_2 - \Delta_B \\ \omega_3 = \omega_3 + \Delta_A \\ \omega_4 = \omega_4 + \Delta_A \end{cases}$$

Allo stesso modo funziona il roll, in cui però ci si muove a destra ($\Delta_A, \Delta_B \in \mathbb{R}^+$) e sinistra ($\Delta_A, \Delta_B \in \mathbb{R}^-$), cambiando la velocità dei motori laterali.

$$\begin{cases} \omega_1 = \omega_1 + \Delta_A \\ \omega_2 = \omega_2 - \Delta_B \\ \omega_3 = \omega_3 - \Delta_B \\ \omega_4 = \omega_4 + \Delta_A \end{cases}$$

Lo yaw invece determina la direzione di volo modificando la velocità di rotazione dei motori opposti. A destra con $\Delta_A, \Delta_B \in \mathbb{R}^+$, a sinistra con $\Delta_A, \Delta_B \in \mathbb{R}^-$.

$$\begin{cases} \omega_1 = \omega_1 + \Delta_A \\ \omega_2 = \omega_2 - \Delta_B \\ \omega_3 = \omega_3 + \Delta_A \\ \omega_4 = \omega_4 - \Delta_B \end{cases}$$

2.1.1 Utilizzi

Il principale vantaggio dei quadricotteri è la loro semplicità di utilizzo. L'operatore infatti non deve preoccuparsi di stabilizzare il velivolo né tantomeno di controllare la potenza dei singoli rotori. Proprio per questi motivi i quadricotteri si sono diffusi in molti ambiti.
L'utilizzo maggiore riguarda le riprese a video. Aggiungendo una telecamera è possibile infatti raggiungere punti di vista ed effetti altrimenti impossibili. Vengono impiegati anche nel controllo dell'ordine pubblico, per esempio durante le manifestazioni, in cui la situazione può diventare critica molto velocemente e sorvolando la zona di interesse possono essere raccolte importanti informazioni per le forze dell'ordine (Figura 2.3).

Figura 2.3: Poliziotto che comanda un quadricottero

Un altro ambito interessa le riprese durante i matrimoni. Molte sono le coppie che decidono di ricordare il giorno del loro matrimonio attraverso le riprese suggestive che un quadicottero è in grado di dare (Figura 2.4).

Figura 2.4: Quadricottero che riprende un matrimonio

Anche in caso di incendi, o calamità naturali, i quadricotteri forniscono una risposta ottimale a queste emergenze svolgendo un ruolo chiave per valutare l'impiego delle risorse e le necessità. Applicando una telecamera termica sarà inoltre possibile aumentare la possibilità di individuare eventuali persone intrappolate negli edifici.
Le riprese vengono sfruttate anche dalle recenti olimpiadi di Sochi, dove si sono visti molteplici quadricotteri riprendere gli atleti nelle varie attività sportive (Figura 2.5).

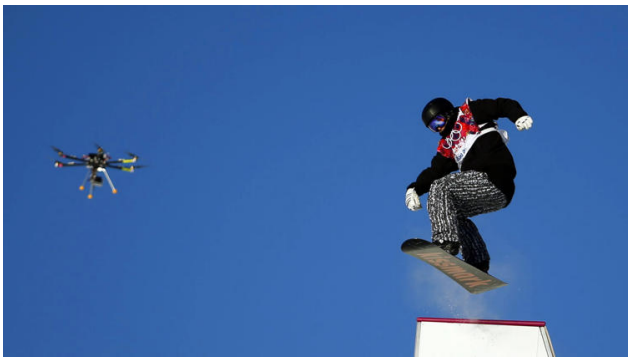


Figura 2.5: Quadricottero che riprende un atleta

2.1.2 Ricerca

Il volo autonomo è l'area di ricerca in cui i quadricotteri vengono maggiormente usati. L'obiettivo è quello di farlo volare senza l'intervento umano, gestendo tutte le informazioni di volo e agendo di conseguenza autonomamente. Esistono vari progetti di questo genere.

L'Università della Pennsylvania, ha sviluppato delle tecniche di volo che permettono a un quadricottero di seguire la traiettoria migliore per raggiungere una posizione desiderata.
La posizione del quadricottero è calcolato attravero il sistema di tracciamento Vicon[16]. Questo sistema permette attraverso delle telecamere a infrarossi poste all'interno di un ambiente, di catturare i raggi IR emessi da alcuni marcatori collocati a bordo del quadricottero (Figura 2.6).
Questi quadricotteri sono in grado di compiere manovre molto aggressi-

(a) Telecamera (b) Marcatori

Figura 2.6: Telecamera e marcatori Vicon

ve, come passare attraverso spiragli, grazie a un solido modello dinamico che tiene conto dei possibili errori dei sensori di bordo e del posizionamento calcolato[18].
Un altro progetto proveniente dall'Università del Minnesota mostra come sia possibile pilotare un quadricottero attraverso il pensiero. Attraverso un cappello dotato di elettrodi, sono stati analizzati gli impulsi elettrici del cervello, consentendo al pilota di comandare il quadricottero (Figura 2.7).

In particolare pensando di muovere una delle due mani il quadricottero andrà da quella parte, pensando di avvicinare (allontare) entrambe le mani, questo aumenterà (diminuirà) l'altezza. Il movimento in avanti è invece costante[19].

Figura 2.7: Pilotaggio attraverso il pensiero

2.1.3 Giochi con quadricottero

Un altro genere di applicazione deriva dalla semplificazione nel pilotaggio dei quadricotteri. La compagnia francese Parrot ha infatti sviluppato dei quadricotteri (ARDrone) estremamente facili da guidare con costi molto accessibili[1].
Gli ARDrone nascono quindi con lo scopo di divertire l'utenza. In particolare sono state sviluppate applicazioni che sfruttano la telecamera di bordo per creare una realtà aumentata.
Specialmente con AR.Rescue[3] è possibile attraverso il telefono o tablet sulla quale è installata l'applicazione, eseguire alcune missioni come sparare o raccogliere oggetti virtuali (Figura 2.8).

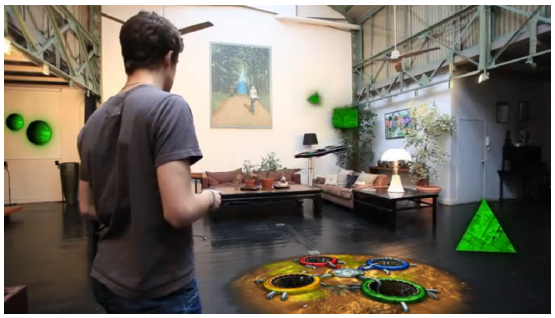


Figura 2.8: Giocatore immerso nella realtà aumentata di AR.Rescue

Un'altra applicazione è AR.FlyingAce[2] dove viene simulata una battaglia con un altro ARDrone. Attraverso due dispositivi è possibile in-

fatti vedere il proprio quadricottero sparare, con lo scopo di distruggere virtualmente l'altro.

2.1.4 Regolamento italiano

L'ENAC, l'ente nazionale per l'aviazione civile, ha recentemente pubblicato una circolare che regolarizza il volo di droni a pilotaggio remoto[9]. È la prima autorità europea ad emettere una norma in questo ambito.
Vengono definiti due principali classi di droni alle quali vengono applicate regole diverse:

- Aereomodello: *"dispositivo aereo a pilotaggio remoto, senza persone a bordo, impiegato esclusivamente per scopi ricreativi e sportivi, non dotato di equipaggiamenti che ne permettano un volo autonomo, e che vola sotto il controllo visivo diretto e costante dell'aeromodellista, senza l'ausilio di aiuti visivi".*

- Aeromobile a Pilotaggio Remoto (APR) *"mezzo aereo a pilotaggio remoto senza persone a bordo, non utilizzato per fini ricreativi e sportivi".*

L'attività degli APR con peso inferiore ai 25 kg, deve essere autorizzata dall'ENAC nei casi di criticità delle operazioni, altrimenti è possibile l'autocertificazione. APR e base di terra devono aver una targhetta che identifichi il pilota. È inoltre necessario un manuale di volo o un documento equivalente.
Gli APR con peso uguale a superiore ai 25 Kg vengono registrati mediante l'iscrizione nel Registro degli Aeromobili a Pilotaggio Remoto.
Il loro pilotaggio è permesso solo tramite un permesso di volo rilasciato dall'ENAC. Il pilota, di età minima di 18 anni, deve conoscere le regole di volo, dimostrate tramite licenza di volo e deve avere seguito un programma di addestramento per l'APR specifico.

Gli aereomodelli invece non necessitano di alcuna autorizzazione. L'aereomodellista ha l'obbligo di avere un continuo contatto visivo diretto con l'aereomodello, senza l'utilizzo di apparecchiature.
Il volo è permesso di giorno, in aree non abitate e non devono essere superati i 70 m di altezza e i 200 m di raggio da dove si trova il pilota. Nel caso che l'aeromodello superi i 25 kg, è richiesta un'età minima di 18 anni.
In ogni caso, l'operatore dovrà assumere la responsabilità di segnalare gli eventuali incidenti causati.

2.2 Robogame

Con il termine Robogame intendiamo quel tipo di interazione tra uomo e robot ai fini di riprodurre un gioco. Nel gioco partecipano due o più agenti, con il vincolo che ci sia almeno un robot autonomo e una persona, la quale può partecipare direttamente, oppure utilizzando un altro robot telecomandato (avatar).
I Robogame sono stati ideati come l'evoluzione dei videogiochi, in cui la partecipazione del giocatore diventa fisica. È molto importante pertanto creare un'interazione in grado di divertire, mantenendo il costo il più possibile accessibile.
Il robot autonomo deve essere in grado di fornire un'esperienza piacevole e coinvolgente per il giocatore, attraverso il movimento, segnali audio o visivi. Deve inoltre essere in grado di inserirsi in un ambiente dinamico, rispondendo in modo efficace ai cambiamenti.
Il robot si trova infatti in un ambiente a lui sconosciuto, in grado di cambiare durante il gioco. Caratteristica principale del robot autonomo non è solo il saper reagire alle mosse del giocatore, ma anche dimostrare di essere intelligente, elaborando tattiche e strategie.

Il fattore divertimento è legato anche alla difficoltà del gioco stesso. Un gioco troppo facile in cui il robot autonomo non è sufficientemente competitivo non dà stimoli. Viceversa se troppo difficile, può spazientire il giocatore. È indispensabile trovare il giusto equilibrio per creare un livello di sfida ottimale.
Come in ogni gioco, esistono delle regole da rispettare che devono essere definite a priori. Queste regole devono essere rispettate sia dal giocatore umano, sia dal robot autonomo.

I ricercatori dell'AIRLAB del Politecnico di Milano, hanno sviluppato insiame agli studenti alcuni giochi con robot autonomi, classificabili come Robogame.

Shooter Game

Questo gioco nasce come l'evoluzione dei classici videogame sparatutto. Il robot autonomo ha l'obiettivo di portare un messaggio alla base principale, mentre il giocatore umano ha l'obiettivo di contrastarlo utilizzando delle armi.
Queste armi danneggiano il robot autonomo fino a immobilizzarlo. Se ferito,il robot può raggiungere delle basi amiche sparse sul percorso, in cui ha

la possibilità di ripararsi restando immune dai colpi dei giocatori. Inoltre, gli è consentito utilizzare un raggio che disabilita le armi dei giocatori per un certo lasso di tempo. Tuttavia l'uso di questa difesa è limitata in quanto richiede energia e viene perciò impiegato solo in caso di estrema necessità.
Il robot autonomo è una versione modificata con degli appositi sensori del robot commerciale della Meccano®, chiamato Spykee (Figura 2.9).

Sono stati aggiunti sonar, per il rivelamento degli ostacoli, led per segna-

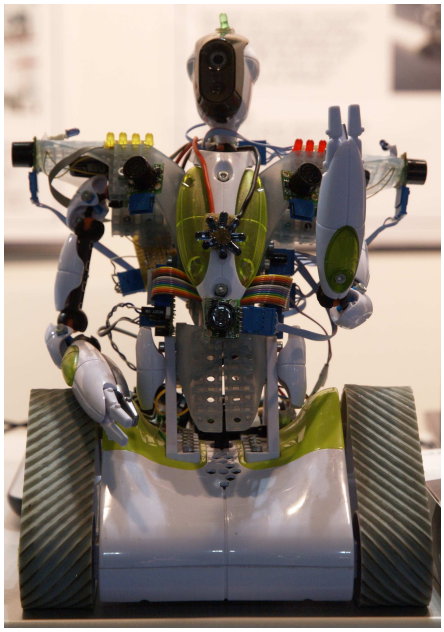


Figura 2.9: Versione di Spykee modificata

lare l'energia restante al giocatore, e dei led infrarossi per renderlo visibile al WiiMote.
Il WiiMote viene utilizzato dal giocatore, con il quale, attraverso delle gesture, può sparare, cambiare arma e ricaricare.
L'intero sistema funziona con una comunicazione Wi-Fi tra Spykee e un pc esterno che decide movimenti e velocità. Le informazione del WiiMote sono invece inviate al pc tramite bluetooth.

Spike Game

In questo gioco si utilizzano due robot della Lego®, uno autonomo e uno telecomandato. Quello autonomo rappresenta la preda. È dotato di alcuni sonar utili per schivare gli ostacoli ed è in grado di seguire alcune tipologie di strategie per fuggire. Il robot telecomandato (Spike) rappresenta il predatore

e viene comandato tramite gesti rilevati da un WiiMote.
Lo scopo del giocatore è quello di colpire tramite la coda di Spike il robot autonomo.

Jedi Trainer Game

Questo gioco si ispira alla saga cinematografica di Guerre Stellari, in particolare alla scena in cui Luke Skywalker impara a usare la spada laser tipica dei cavalieri Jedi: utilizzando la spada schiva e devia i colpi laser sparategli addosso da un drone.
Allo stesso modo, viene utilizzato un ARDrone Parrot il quale tenta di colpire il giocatore umano. I colpi laser vengono simulati attraverso un suono emesso dal drone, il quale colpisce solo nel caso in cui il petto del giocatore si trovi al centro della telecamera frontale posta su di esso e che non vi sia la spada a difesa. Il giocatore dispone di un tubo rosso che rappresenta la spada laser, attraverso la quale può difendersi, ponendola tra il proprio petto e l'ARDrone.
Il punteggio viene annunciato in tempo reale da un computer esterno, sul quale è implementata tutta la logica del drone autonomo, che comunica con esso tramite Wi-Fi, inviandogli le immagini della camera interna e ricevendo i comandi.
Il drone cerca di allinearsi dunque con il petto del giocatore umano, individuato grazie a una tunica blu che indossa, ma compirà anche movimenti casuali e aggressivi che portano il giocatore a muoversi maggiormente durante il gioco (Figura 2.10).

Figura 2.10: Visione da una camera esterna, dall'ARDrone e l'elaborazione dell'immagine effettuata in Jedi Trainer

Capitolo 3

Struttura del gioco

"Il gioco è bello quando dura poco."

Proverbio

Il gioco che abbiamo realizzato presenta tutte le caratteristiche necessarie per essere considerato un Robogame.
Si tratta di un gioco coinvolgente e vivace, in grado di divertire il giocatore sia per le dinamiche del gioco, sia per il pilotaggio di un robot volante. Il gioco è costituito da due agenti robotici: un drone autonomo e uno telecomandato.
Per motivi di sicurezza il drone autonomo può essere comandato anche attraverso un radiocomando, che deve essere affidato a un operatore specializzato, in grado di pilotarlo in caso di necessità.
Un gioco tra elementi volanti è un elemento innovativo nel campo dei Robogame, che lo rende unico nel suo genere.
Oltre a questo ci sono elementi tecnici e ingegneristici innovativi che restano invisibili agli occhi del giocatore che resta focalizzato sul gioco nel quale è completamente immerso.

3.1 Specifiche del gioco

Il Robogame da noi sviluppato è la rivisitazione di un gioco conosciuto, il laser-game, in cui i protagonisti sono robot, nello specifico droni.
In particolare il drone telecomandato (ARDrone) deve fuggire dal drone autonomo, il quale, dopo averlo individuato, lo insegue cercando di colpirlo con il proprio laser. Il gioco deve necessariamente svolgersi in un ambiente chiuso e delimitato, che permetta al drone autonomo di vedere il drone-avatar e non permetta a questi di portarsi fuori portata. Il giocatore umano deve

mostrare la sua abilità nello sfuggire al drone autonomo.
Per rendere il gioco più dinamico, abbiamo deciso di fare atterrare il drone autonomo una volta durante il tempo di gioco, a simboleggiare una ricarica d'energia. In questa fase se il drone telecomandato è capace di atterrare davanti ad esso ad una determinata distanza segnalata da un laser, impedisce il rifornimento d'energia vincendo direttamente il gioco. Se invece fallisce il gioco riprende con il decollo del drone autonomo.
In questo caso il drone-avatar dovrà scappare nuovamente dal drone autonomo fino al termine del gioco. Se il giocatore riesce a non far colpire il proprio drone durante tutto il tempo di gioco, ottiene la vittoria. Se invece viene colpito, il giocatore perde.

L'ARDrone viene comandato tramite l'applicazione proprietaria AR.FreeFlight, installata su uno smartphone o un tablet.

3.1.1 Regole e linee guida

Oltre alle regole di gioco utili a renderlo più vivace, sono state inserite anche alcune linee guida, soprattutto legate a motivi di sicurezza in presenza di un drone autonomo:

- Prima di cominciare il gioco è necessaria una fase di training per il giocatore che deve prendere dimestichezza con il drone telecomandato.
- I droni decollano simultaneamente da due punti distanti almeno 8 metri.
- Il drone telecomandato non può passare sotto o sorvolare l'altro drone.
- Il drone telecomandato deve mantenere una distanza di sicurezza sia dagli ostacoli, sia dal drone autonomo.
- Il drone telecomandato non può superare i 3 metri di altezza.
- Il drone telecomandato non può superare l'inclinazione di 15°.
- In caso di pericolo d'impatto, il giocatore deve far atterrare il drone telecomandato. Il gioco riparte riportando i due droni al punto di partenza.
- Il drone autonomo necessita di un pilota, il quale deve intervenire attraverso il telecomando solo in caso di pericolo, riportandolo in una posizione sicura. Durante questa fase il drone non può sparare.

- Durante la partita, sia il giocatore sia il pilota devono stare sul campo di gioco, senza interferire con lo svolgimento del gioco stesso.
- Il tempo di gioco è di 5 minuti, dovuti alla limitata autonomia energetica dei droni.
- Il drone autonomo spara attivando il laser solo quando il drone telecomandato è allineato con esso e distante tra i 220 cm e i 280 cm. Se viene colpito, il giocatore perde e il gioco termina.
- Il drone autonomo deve atterrare una sola volta tra i 2 e i 3 minuti di gioco per 30 secondi. In questa fase, se il drone telecomandato atterra di fronte a lui orientato nello stesso modo e tra i 120 e i 170 cm di distanza (segnalata da un laser), il giocatore vince e il gioco termina. Altrimenti, viene emesso un segnale audio per indicare la ripartenza del drone autonomo. Quando viene emesso questo suono il drone telecomandato deve allontarsi dal drone autonomo e il gioco riprende.
- Terminato il tempo di gioco, se il drone autonomo non ha colpito quello telecomandato, il giocatore vince.

3.2 Agenti

Come già introdotto precedentemente, il gioco è composto da due agenti robotici: un drone autonomo e un drone telecomandato.

3.2.1 Drone autonomo

Il drone autonomo è stato costruito interamente nell'ambito della nostra tesi. Tutte le scelte HW/SW sono state effettuate per seguire le specifiche del gioco e sono descritte nel dettaglio nei prossimi capitoli.
Il drone autonomo deve essere in grado di compiere le seguenti macro-azioni:

- *Decollo*: il drone decolla autonomamente portandosi ad una quota determinata.
- *Ricerca*: il drone inizia la ricerca dell'avversario all'interno di un ambiente di gioco non conosciuto.
- *Gestione ostacoli*: il drone evita gli ostacoli portandosi a una distanza di sicurezza.
- *Inseguimento*: il drone si avvicina all'avversario allineandosi con esso.

- *Fuoco*: il drone attiva il laser e riconosce se ha colpito o meno l'avversario.

- *Atterraggio*: il drone atterra autonomamente e spegne i motori.

3.2.2 Drone telecomandato

Il drone telecomandato scelto per il nostro gioco è l'ARDrone 2.0 dell'azienda francese Parrot® (Figura 3.1)[20].

Figura 3.1: AR Drone 2.0 Parrot

È stato scelto questo drone per la sua estrema facilità di utilizzo. Basta provarlo pochi minuti per apprendere le tecniche di volo basilari per il gioco. Le sue caratteristiche sono:

- Dimensioni: 51.5 x 51.5 cm (con scocca).

- Peso: 436 g (con scocca).

- Velocità massima: 18 Km/h.

- Quota massima: 50 m.

Le caratteristiche hardware del drone sono:

Processore	ARM Cortex A8 da 1GHz 32bit
Video	800 MHz DSP TMS320DMC64x
RAM	DDR2 da 1GB a 200 MHz
Giroscopio	a 3 assi per una precisione fino a 2000°/sec
Accelerometro	a 3 assi per una precisione fino a +/-50 mg
Magnetometro	a 3 assi per una precisione fino a 6°
Sensore di pressione	a 3 assi per una precisione fino a +/- 10 Pa
Motori	4 brushless a rotore interno.
Potenza motori	14.5 W e 28500 rpm in volo statico
Telecamera verticale	QVGA a 60 fps
Telecamera orizzontale	HD 720p a 30 fps
Wi-Fi	b,g,n
Batterie	LiPo ricaribabili da 1000 mA/h o 1500 mA/h

Il drone può essere comandato tramite tablet o smartphone. In particolare è disponibile l'applicazione AR.FreeFlight, che interfaccia l'utente ai comandi del drone (Figura 3.2).

Figura 3.2: Interfaccia di AR.FreeFlight

Tramite essa è possibile decollare, muoversi, atterrare e monitorare le principali informazioni di volo come il livello di batteria, l'altitudine e la velocità raggiunta.
Inoltre, è possibile stabilire i valori massimi e minimi di velocità e di quota raggiungibili (Figura 3.3).

Modifiche

L'aspetto del drone telecomandato è stato modificato appositamente per il nostro gioco. In particolare abbiamo modificato la scocca per interni in modo tale da renderlo maggiormente riconoscibile da parte dell'altro drone.

Figura 3.3: Settaggio delle impostazioni di volo

Il drone autonomo utilizza due metodi per individuare e seguire il drone telecomandato. Il primo metodo consiste nel riconoscimento della forma e del colore: il drone autonomo è in grado di identificare i rettangoli rossi con fattore di forma orizzontale.
Per questo motivo abbiamo rivestito la scocca con un cartoncino di colore rosso alto circa 8 cm. Questa modifica è stata fatta in modo tale da non creare asimmetrie sul drone, che altrimenti avrebbero influito sulla dinamica di volo.
Il secondo metodo consiste invece nel riconoscimento di sorgenti ad infrarossi. Abbiamo dunque posizionato due gruppi composti da tre led infrarossi ciascuno, su ognuno dei quattro lati della scocca. I due gruppi sullo stesso lato sono distanti 20 cm.
I raggruppamenti servono solamente a potenziare l'effetto dei led, che in questo modo possono essere individuati a una distanza maggiore e da diverse inclinazioni.
Il risultato finale è riportato in Figura 3.4. I led infrarossi vengono alimen-

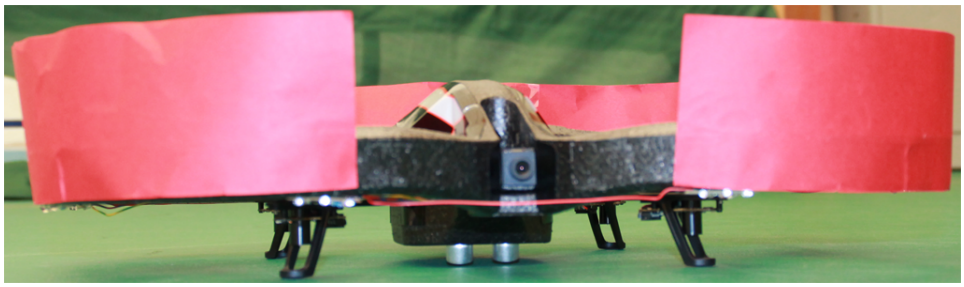


Figura 3.4: La scocca dell'ARDrone modificata

tati dalla stessa batteria che alimenta i motori che pertanto sacrificherà ad essi una parte dell'autonomia. Essi consumano un totale di circa 250 mA.

Il peso totale del drone risulta di conseguenza superiore al valore iniziale. Questo può causare problemi di stabilità al drone, ma abbiamo verificato che questo avviene solamente quando la batteria è ormai scarica (circa 10%).

3.3 Ambiente

L'ambiente di gioco deve presentare alcune caratteristiche necessarie sia per un corretto funzionamento del gioco, sia per un maggior coinvolgimento del giocatore.
Data la dinamica del gioco è indispensabile una stanza chiusa di dimensioni non inferiori ai 10 x 10 m. In questo modo i droni hanno sufficiente libertà di movimento e non vengono influenzati dai fattori climatici che condizionano la qualità del volo. L'altezza deve essere almeno di 3 m e non devono essere presenti ostacoli o dislivelli significativi all'interno del campo.
Un'altra caratteristica fondamentale è l'assenza di rosso nella stanza che, anche se viene filtrata dal drone autonomo attraverso il riconoscimento della forma, può comunque causare falsi.
Allo stesso modo non si devono avere finestre con luce diretta, poiché la luce del sole proveniente da esse potrebbe causare false letture ai sensori a infrarossi. Occorre inoltre ricordare che, dato che il rilevamento di ostacoli è fatto con tecniche visive, i vetri non vengono rilevati come ostacoli.
In Figura 3.5 è mostrato l'ambiente utilizzato per le nostre prove di gioco.

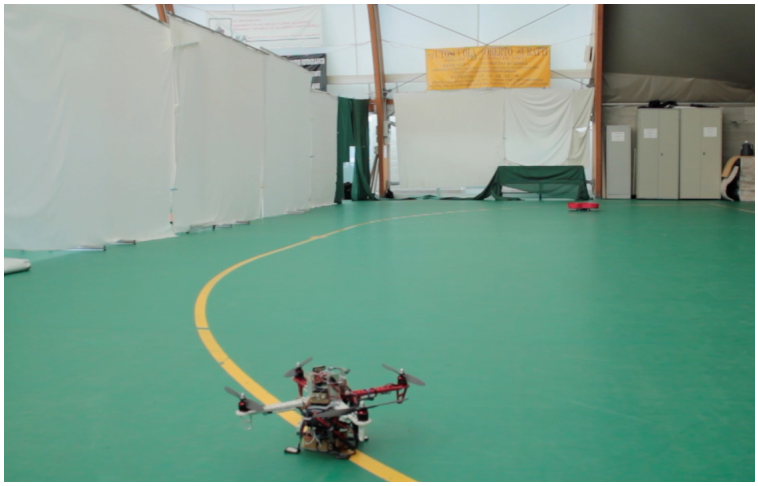


Figura 3.5: L'ambiente di gioco

Capitolo 4

Architettura hardware

"Mai far fare ad un essere umano il lavoro che dovrebbe fare una macchina"

Matrix

La progettazione e la realizzazione del drone utilizzato per il nostro gioco sono state una parte molto significativa di questa tesi.
La varietà di componenti esistenti e di soluzioni possibili ha dato luogo ad una ricerca che ha avuto come obiettivo quello di costruire un drone affidabile, robusto, espandibile, semplice da utilizzare e con un prezzo contenuto.
Tra i componenti scelti, alcuni sono strettamente collegati al gioco creato mentre la maggior parte sono generici e si possono adattare molto bene a tantissimi altri scopi.
La scelta più importante effettuata è stata quella di mettere tutta la potenza di calcolo a bordo del drone. Esso infatti non necessita di alcun calcolatore esterno per poter volare in autonomia. Non richiede inoltre alcun sensore che non sia già montato a bordo.
In questo modo, il drone risulta essere un oggetto a sé, completamente indipendente, facile da spostare e utilizzare ovunque si voglia.

4.1 Struttura base

La struttura base del drone è stata realizzata utilizzando un kit commerciale ARTF (Almost Ready To Fly) prodotto dall'azienda DJI.
Il kit in questione è il DJI F450 Flame Wheel Multi Rotor Quadcopter ARTF Kit che comprende:

- il frame del drone composto da quattro braccia colorate e due piastre di collegamento (Figura 4.1).

Figura 4.1: Frame

- quattro motori brushless da 920 Kv (Figura 4.2a).
- quattro ESC OPTO da 30 Ampère (Figura 4.2b).
- quattro eliche 10x4.5 (Figura 4.2c).

(a) ESC

(b) Motori

(c) Eliche

Figura 4.2: Parti principali della struttura

Un ESC (Electronic Speed Controller) è un circuito elettronico che permette di variare la velocità di un motore elettrico, la sua direzione e di agire come freno. Un ESC per motori brushless, pilota il motore trifase inviando una sequenza di segnali per la sua rotazione.
Essi possono fornire al massimo una data corrente al motore che nel nostro

caso è di 30 A, per un totale di 120 A.
Gli ESC utilizzati sono di tipo OPTO ovvero non contengono al loro interno un BEC integrato.

Un BEC (Battery Eliminator Circuit) è un circuito che permette di stabilizzare la tensione d'ingresso proveniente dalla batteria ad un valore utile ad alimentare correttamente un altro circuito.
Sono utilizzati nell'ambito dei droni per alimentare la scheda di volo utilizzando la batteria che alimenta anche i motori.
Non avendo a disposizione un BEC integrato, è stato necessario utilizzarne uno esterno da 3 A per poter alimentare la scheda di volo.

La struttura ha una diagonale lunga 45 cm, a cui va aggiunta la lunghezza delle eliche per arrivare ad un totale di circa 65 cm.
Questa dimensione fa sì che il drone possa muoversi bene in ambienti chiusi ed essere sufficientemente maneggevole.
Con questo tipo di configurazione è possibile far volare il drone con un peso complessivo di 1700 g.

4.2 Scheda di volo

La scheda di volo è il componente fondamentale per quanto riguarda tutta la gestione dei motori e del volo. Tramite il software al suo interno ha infatti il compito di gestire adeguatamente i comandi ricevuti dal radiocomando e di fornire la giusta potenza ad ogni motore.
Grazie ai sensori di bordo, tra cui accelerometri, giroscopi e magnetometro, è in grado di stabilizzare il volo del drone e di controllare la corretta esecuzione dei comandi.
L'accelerometro misura le accelerazioni sui 3 assi x,y e z e, grazie a questi valori, il controllore è in grado di calcolare gli angoli di inclinazione relativi agli assi di pitch e roll.
Il giroscopio misura invece la velocità angolare sui 3 assi.
Il magnetometro misura l'intensità del campo magnetico sempre sui 3 assi e, a partire da questi valori, il controllore calcola l'angolo relativo all'asse di yaw.

In commercio c'è una grandissima varietà di schede di volo, le quali si differenziano per il loro prezzo, il software installabile e per la quantità e qualità

dei sensori presenti.
I software open source più utilizzati sono il Multiwii e l'Arducopter.
Questi software hanno molte caratteristiche in comune ed entrambi hanno una buona documentazione presente sul web e un buon seguito di sviluppatori che li mantengono sempre aggiornati.
E' molto importante che il software di volo sia open source poiché è necessario modificarlo, come mostrato in seguito, per poter aggiungere o cambiare alcune funzionalità utili al nostro lavoro.

Tra le schede presenti in commercio abbiamo scelto la Crius All In One Pro Flight Controller V2 (Figura 4.3) e utilizzato il software Multiwii. Questa scheda è una delle più economiche e versatili presenti poiché è possibile installarvi sia il Multiwii che una versione modificata dell'Arducopter chiamata MegaPirateNG. Quest'ultima caratteristica è stata determinante nella scelta insieme al fatto che la qualità dei sensori presenti al suo interno è paragonabile a quello delle schede direttamente concorrenti.

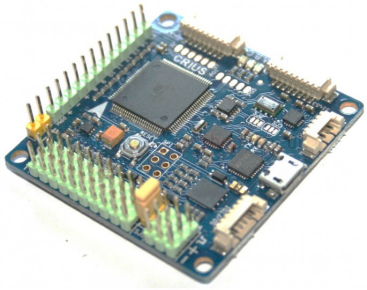


Figura 4.3: Crius All In One Pro Flight Controller V2

La Crius presenta le seguenti caratteristiche:

- Controllo fino ad un massimo di 8 motori
- 8 canali di ingresso per le riceventi più canale PPM SUM
- 2 uscite per il sistema di gimbal su due assi
- 4 porte seriali per i moduli di debug, Bluetooth, OSD, GPS, telemetria
- 8 porte analogiche per sensori di velocità, corrente, voltaggio, led
- porta I2C a 5 V

- memoria dedicata per la registrazione dei dati
- microcontrollore ATMega 2560-16AU
- MPU6050 giroscopio-accelerometro a sei assi
- HMC5883L magnetometro digitale a tre assi
- MS5611-01BA03 altimetro di precisione
- porte FT232RQ USB-UART e Micro USB

4.2.1 Installazione

L'installazione della scheda di volo sulla struttura del drone deve tenere conto delle pesanti vibrazioni a cui viene sottoposta durante il volo. Queste vibrazioni devono essere smorzate in modo tale da non essere propagate alla scheda che, in caso contrario, non potrebbe fare correttamente il suo lavoro. Le vibrazioni sono attenuate sia fisicamente che lato software. Fisicamente occorre montare la scheda senza fissarla direttamente al frame, utilizzando dei distanziali in plastica fissati tramite viti dello stesso materiale, oppure incollando la scheda ad un pezzo di gomma, a sua volta incollato al frame. Un altro modo può essere quello di utilizzare un dispositivo anti-vibrazioni formato da due piastre plastiche collegate tra loro da speciali gommini smorzavibrazioni (Figura 4.4).

Figura 4.4: Gommino anti-vibrazioni

Poiché le vibrazioni con il tipo di frame scelto sono molte, abbiamo optato per utilizzare sia la gomma che il dispositivo anti-vibrazioni insieme. Lato software abbiamo introdotto un filtro ad una certa frequenza sui dati letti dai sensori presenti sulla scheda. L'insieme di queste soluzioni ha reso il drone completamente esente da vibrazioni.

Un'altra problematica di cui tenere conto durante l'installazione della scheda è l'interferenza sul magnetometro provocata dai cavi di alimentazione. Le forti correnti che passano attraverso gli ESC per alimentare i motori, che nel nostro caso arrivano fino a 30 A per ogni motore, potrebbero compromettere le funzionalità fondamentali del magnetometro, influenzando così la stabilità del volo.
Per evitare che tutto ciò accada, occorre arrotolare su stessi tutti i cavi di alimentazione proveniente dalla batteria e dagli ESC ed allontanarli il più possibile dalla scheda.

4.2.2 Software

Nell'ambito di questa tesi abbiamo optato per il sistema open source Multiwii che, per la sua conformazione, è il migliore in termini di semplicità d'uso e di personalizzazione. Inoltre il software in questione gestisce al meglio tutte le funzionalità di cui abbiamo avuto bisogno.

Il software è stato modificato per poter avere sul radiocomando un interruttore fisico che permetta di passare velocemente dalla modalità automatica a quella manuale.
Per ragioni di sicurezza abbiamo infatti deciso che questo era il metodo più sicuro per evitare incidenti ed intervenire tempestivamente in caso di comportamento anomalo del drone.
Il software è stato strutturato in modo tale da poter ricevere i comandi o dal radiocomando o dalla porta seriale la quale viene utilizzata per il volo autonomo. La nostra modifica consiste nel leggere sempre un canale (AUX1) del radiocomando anche in modalità automatica. Un'analoga modifica è stata effettuata anche su un secondo canale (AUX2) per poter dare il comando di atterraggio durante il volo in automatico.
Questa configurazione permette di essere completamente indipendenti da un computer di terra potendo prendere tutte le decisioni tramite il radiocomando.

4.2.3 Messa a punto

Dopo aver installato il software si passa alla messa a punto di tutti i parametri propri del volo, utilizzando l'interfaccia software fornita dal Multiwii (Figura 4.5). Prima di fare ciò però si devono calibrare il magnetometro e gli accelerometri, per poi verificare che il cablaggio non interferisca con il

magnetometro.

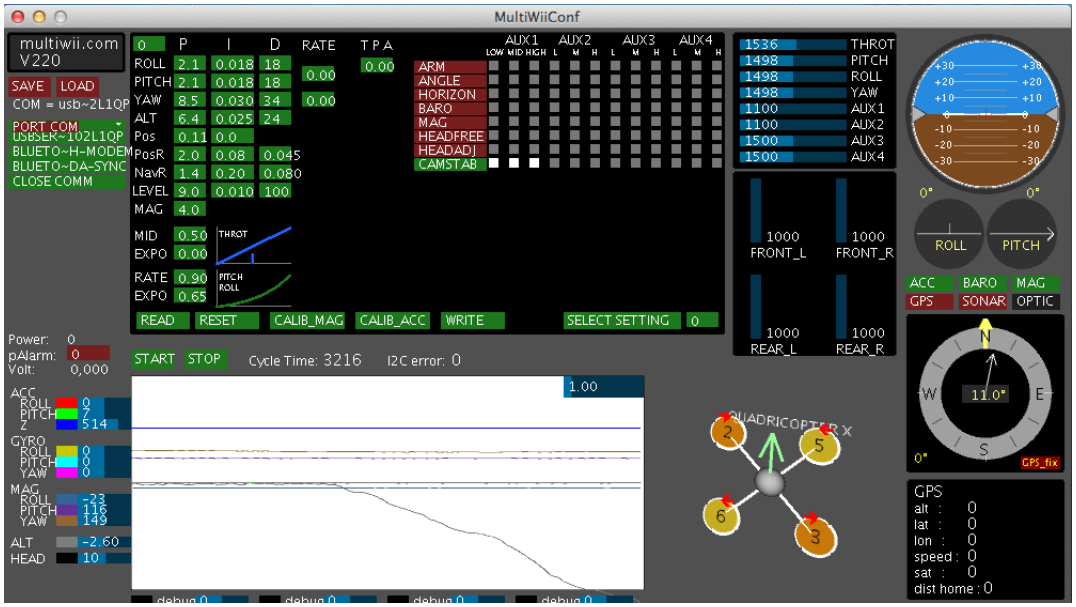


Figura 4.5: Interfaccia Multiwii

Per calibrare il magnetometro, si fa ruotare manualmente il drone lungo i tre assi di roll, pitch e yaw per 30 secondi. La calibrazione degli accelerometri invece si effettua semplicemente mettendo il drone in piano ed attivando una particolare sequenza con il radiocomando (Appendice C). Questa procedura per gli accelerometri è da effettuare tutte le volte prima di un volo.

Per verificare che il cablaggio non influenzi il magnetometro, occorre collegare la scheda di volo ad un PC ed aprire il software in dotazione. Bisogna quindi accendere i motori del drone al massimo, senza le eliche collegate, e verificare che il valore dell'angolo di prua visualizzato non cambi. Nello stesso tempo, si può verificare che le vibrazioni introdotte dalla rotazione dei motori non influenzino il comportamento degli accelerometri. Anche in questo caso basta verificare che i valori degli accelerometri rimangano costanti.
La stessa procedura si può effettuare anche con le eliche collegate, prestando molta attenzione alla forza che si dà ai motori.
Se queste operazioni hanno dato un esito negativo è inutile continuare perché il drone non volerà mai correttamente.

La messa a punto vera e propria della scheda, riguarda i parametri dei controllori PID (ampiamente spiegati in seguito), presenti nel software che hanno lo scopo di far reagire correttamente il quadricottero a seguito di un comando ricevuto.
Questi parametri dipendono strettamente dalla forma e dalla struttura del drone. Per trovare i valori corretti per ogni asse su cui opera il drone, siamo partiti dai valori standard proposti dal software Multiwii, andandoli a modificare uno ad uno fino a raggiungere la stabilità ed il comportamento desiderati.
Modificando i valori dei PID il modo di agire del drone si trasforma in questo modo:

- **aumentare la P:** aumenta la stabiltà e la solidità. Si nota una grossa resistenza ad ogni tentativo di muovere il drone. Quando tale valore è troppo alto, il drone inizia ad oscillare e perdere il controllo.
- **diminuire la P:** diminuisce la resistenza ai movimenti. Se è troppo basso, il drone diventa molto instabile.
- **aumentare la I:** aumenta la capacità di mantenere una posizione e di ridurre gli sbandamenti ma, allo stesso tempo, aumenta il tempo di ritorno alla posizione di partenza. Diminuisce l'importanza della P.
- **diminuire la I:** migliora la reazione ai cambiamenti ma aumenta gli sbandamenti.
- **aumentare la D:** aumenta la velocità con cui gli errori vengono recuperati. Un valore troppo alto introduce oscillazioni. Aumenta l'importanza della P.
- **diminuire la D:** riduce le oscillazioni quando si torna nella posizione iniziale dopo un errore. Questo recupero diventa però più lento.

I valori che abbiamo ottenuto dopo molte prove pratiche, sono mostrati in Figura 4.6.

4.3 Odroid

L'Odroid è una piattaforma di sviluppo commercializzata dall'azienda sudcoreana Hardkernel che si basa su architettura ARM, ed è stata selezionata come elaboratore principale di bordo.

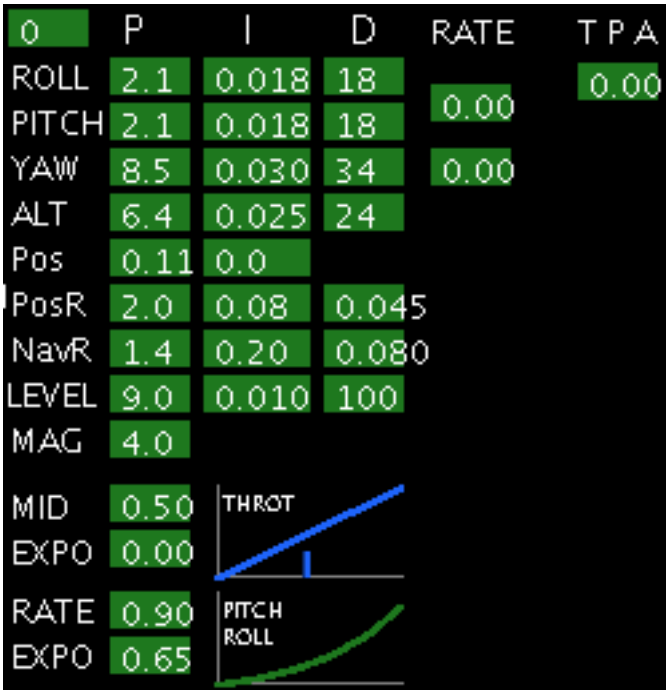
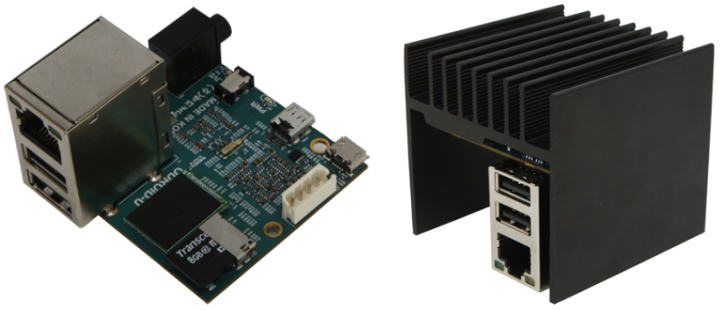


Figura 4.6: Valori ottenuti per i PID



(a) Odroid-U2 senza dissipatore (b) Odroid-U2 con dissipatore

Figura 4.7: Odroid-U2

Esso rappresenta il cuore del nostro lavoro. Tutti i componenti utilizzati, infatti, sono collegati all'Odroid che, grazie al software sviluppato, gestisce il flusso di informazioni e prende le decisioni utili a far volare autonomamente il drone.

4.3.1 Caratteristiche

Nel nostro progetto abbiamo scelto di utilizzare l'Odroid-U2 (Figura 4.7) le cui caratteristiche principali sono:

- Processore Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.7 Ghz con 1 MB di cache L2
- 2 GB di memoria ram LP-DDR2
- Processore grafico Mali-400 Quad Core 440 MHz con acceleratore 3D
- Supporto video fino a 1080p con cavo HDMI (H.264+AAC)
- Uscita video con connettore micro HDMI
- Una porta lan 10/100 Mbps con connettore RJ-45
- Due porte USB ad alta velocità
- Una porta micro USB per ADB
- Uno slot per scheda microSD per poter installare il sistema operativo
- Alimentazione a 5 V con un consumo massimo di 2 A
- Case di alluminio per protezione e raffreddamento
- Dimensioni contenute (solo scheda 48x52 mm)
- Possibilità di installare una distribuzione Linux con kernel 3.0.x oppure Android 4.x
- Prezzo contenuto (89 $)

Il performante processore quad core e l'avanzato processore grafico rendono l'Odroid un prodotto dalle prestazioni incredibili e quindi assolutamente adatto in un contesto in cui la velocità di reazione ai cambiamenti dell'ambiente circostante, è fondamentale.
La documentazione di questo prodotto è disponibile sul sito web ufficiale[7] dove è altresì presente un forum[6] in cui vengono analizzate e discusse numerose possibilità software ed espansioni hardware per la scheda. Questo permette di avere sempre numerosi aggiornamenti e un'assistenza in tempo reale.

4.3.2 Piattaforme concorrenti

Sul mercato sono presenti piattaforme analoghe a quella scelta ma che si differenziano per alcune caratteristiche che le rendono meno adatte o meno performanti.
Le principali concorrenti al momento della scelta della piattaforma erano: Odroid-X2, Raspberry Pi e pcDuino.

Odroid-X2

Il primo concorrente è prodotto dalla stessa Hardkernel e si differenzia principalmente per la presenza di sei porte USB contro le due dell'U2. Presenta poi una scheda grande quasi quattro volte l'altra e un prezzo più alto del 50%. Nonostante le porte USB in più costituissero un buon motivo per scegliere questa versione, i 50 $ di differenza sono esagerati e non corrispondono ad un aumento effettivo delle prestazioni, mentre le dimensioni avrebbero creato problemi a bordo.

Raspberry Pi

Tra i prodotti concorrenti dell'Odroid, il Raspberry Pi è il più famoso e diffuso.
Il suo punto di forza sta nel prezzo bassissimo (appena 35 $) e nella comunità che lo segue. La documentazione presente online è infatti molto sviluppata, ed è accompagnata da innumerevoli esempi di utilizzo e progetti svolti.
Il principale svantaggio rispetto all'Odroid, sta nella configurazione hardware. Il Raspberry infatti, monta un processore ARM1176JZF-S single core (famiglia ARM11) a 700 MHz e un processore grafico Broadcom VideoCore IV. La memoria ram disponibile è di 512 MB, condivisa tra cpu e gpu.

Con queste caratteristiche non è possibile avvicinarsi alle performance dell'Odroid-U2 e non è possibile gestire in tempo utile tutti i sensori. In particolare, il Raspberry Pi non è in grado di processare le immagini proveniente dalla webcam con la stessa velocità dell'Odroid.
In Figura 4.8 è riportato un grafico, diffuso dall'Hardkernel, che mette a confronto le performance del Raspberry Pi e dell'Odroid-U3. L'Odroid-U3 è il naturale successore dell'Odroid-U2 che, al momento della scelta, non era ancora in produzione. Le componenti hardware dell'U3 sono identiche a quelle dell'U2 e pertanto anche le prestazioni totali.

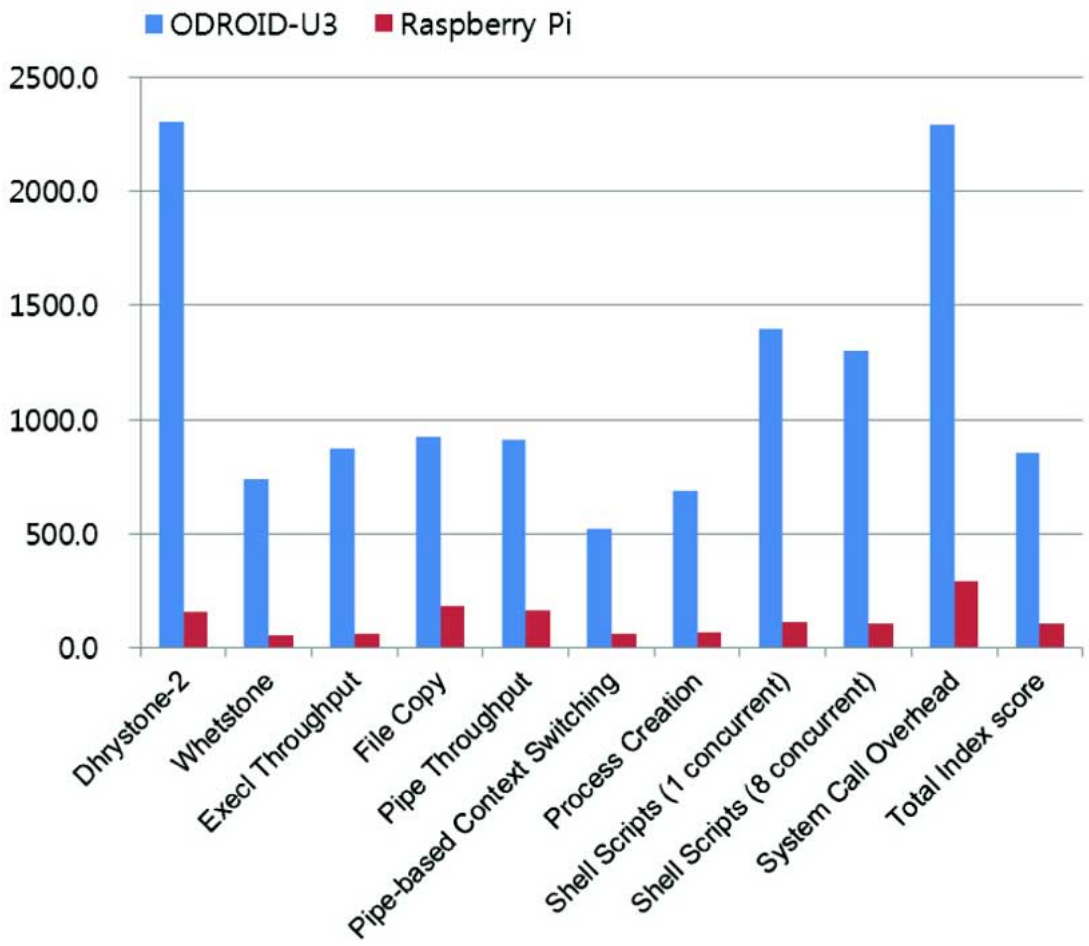


Figura 4.8: Prestazioni a confronto fra Odroid-U3 e Raspberry Pi

pcDuino

Il pcDuino è un mini computer che può montare un sistema operativo basato su Linux oppure Android 4.x. La sua principale caratteristica è quella di avere un interfaccia hardware compatibile con Arduino integrata.
Esso ha a bordo un processore ARM Cortex A8 da 1 Ghz, 1 GB di memoria ram ed un processore grafico Mali 400. Oltre ad un prezzo pari a quello dell'Odroid-U2 con una cpu molto meno potente, il motivo principale per cui abbiamo scartato questa scheda è stato il fatto che, essendo stata appena rilasciata al momento della scelta, la documentazione e la comunità erano praticamente inesistenti.

In seguito sono uscite altre schede (Intel Galileo, Arduino YUM, YUDOO, ecc.) che rimangono comunque a prestazioni inferiori a Odroid.

4.3.3 Componenti aggiuntivi

Per poter lavorare correttamente, Odroid ha bisogno di alcuni componenti aggiuntivi che ne aumentino le funzionalità.

Scheda microSD

Il sistema operativo di Odroid, deve essere installato su una scheda microSD, poi riposta nell'apposito slot. Abbiamo scelto di utilizzare una scheda microSD da 16 GB classe 10 per avere uno spazio disponibile considerevole ed una buona velocità di lettura e scrittura.

Modulo Wi-Fi

Il modulo Wi-Fi esterno è un componente molto importante del sistema. Odroid, infatti, non è dotato di un modulo Wi-Fi integrato ed è quindi fondamentale aggiungergli questa caratteristica con un modulo esterno.
Il modulo scelto (Figura 4.9) è basato sul chip Realtek RTL8191SU ed è completamente compatibile con Odroid. Grazie ad esso è possibile sfruttare la connessione Wi-Fi che ci permette di monitorare il gioco da remoto.

Figura 4.9: Modulo Wi-Fi

Hub USB

Poiché l'Odroid dispone solo di due porte USB, è stato necessario aggiungere un hub USB per incrementare fino a cinque il numero totale di porte utilizzabili.

In tutto abbiamo quattro dispositivi connessi all'Odroid tramite porta USB.

Figura 4.10: Hub USB Digicom

L'hub scelto (Figura 4.10) è prodotto dalla Digicom e permette di aggiungere quattro porte USB. Viene collegato all'Odroid tramite un cavo mini-USB. La cosa più importante è stata alimentare l'hub a 5 V poiché, mantenendo l'hub autoalimentato, esso non era in grado di fornire la corrente necessaria a far funzionare tutti i dispositivi collegati. Abbiamo dunque alimentato l'hub tramite il BEC che alimenta anche l'Odroid stesso.
Per eliminare del peso inutile, abbiamo smontato l'hub togliendo il contenitore plastico.

4.3.4 Montaggio

Abbiamo montato l'Odroid al di sotto della struttura del drone completo di dissipatore. Per fissarlo abbiamo utilizzate delle L metalliche.
Al di sotto del dissipatore abbiamo posto l'hub USB, collegato poi all'Odroid. Per proteggere queste parti delicate in caso di caduta, abbiamo aggiunto una gomma al di sotto dell'hub. Questa gomma ed il dissipatore, forniscono un'adeguata protezione alla scheda dell'Odroid che è la parte più delicata ed importante dell'intero sistema.

4.3.5 Problematiche

Nel corso della tesi abbiamo avuto molteplici problemi riguardanti l'Odroid e le periferiche a lui connesse. In particolare i problemi più gravi riguardavano il riavvio improvviso dell'Odroid e la mancata risposta ai comandi. Entrambi i problemi, poi risolti, erano dovuti alle forti vibrazioni a cui è sottoposto il drone durante il volo.
Queste vibrazioni provocavano la mancata alimentazione della scheda per

un brevissimo lasso di tempo comunque sufficiente a far riavviare l'Odroid e conseguentemente a far cadere il drone. La mancata risposta ai comandi era sempre causata dalle vibrazioni ma agenti sulle prese USB collegate all'hub o direttamente all'Odroid. Quest'ultima problematica non causava la caduta del drone ma l'impossibilità di continuare il volo autonomo.
Abbiamo risolto i problemi fissando tutti i collegamenti USB e le alimentazioni tra i vari dispositivi utilizzati con la colla a caldo. In questo modo nessun componente è più influenzato dalle vibrazioni ed è comunque semplice togliere la colla per modifiche future.

4.3.6 Sistema operativo

Come sistema operativo è possibile utilizzare una distribuzione Linux oppure Android. Abbiamo scelto di installare una distribuzione Linux per avere una maggiore disponibilità di software utilizzabile e per sfruttare al massimo la velocità e le prestazioni dell'Odroid. In particolare abbiamo utilizzato una versione di Ubuntu sviluppata dall'azienda no-profit Linaro che si occupa di software specifico per piattaforme ARM. La versione del kernel è la 3.0.62.

4.4 Webcam

La webcam è una videocamera che invia immagini in tempo reale ad un computer, collegato generalmente tramite USB.

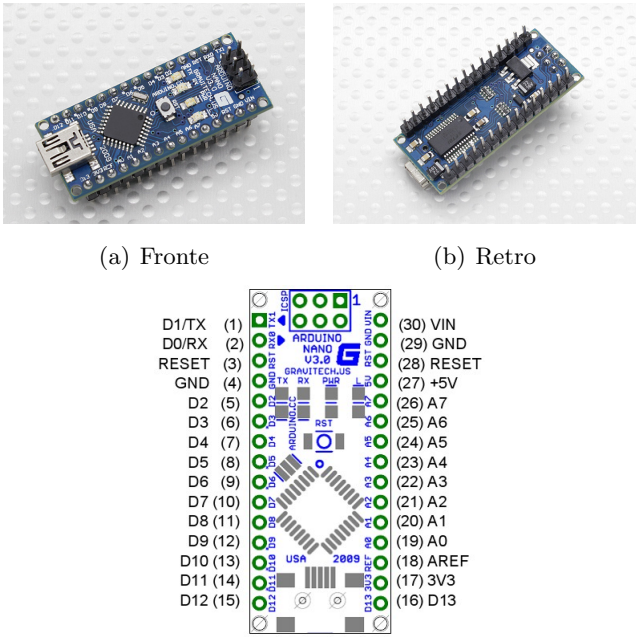
Figura 4.11: Webcam Logitech c270

Nel nostro progetto è stata utilizzata per cercare il drone avversario e per stabilire se, una volta sparato il laser, l'avversario sia stato colpito.
La webcam scelta è la Logitech c270 (Figura 4.11) per la possibilità di ricevere fotogrammi ad una risoluzione massima di 1280×960 e con una velocità di 30 frame al secondo. Inoltre ha un ottimo rapporto qualità/prezzo. Per diminuirne il peso totale, è stata rimossa la parte posteriore che era adibita al fissaggio della webcam stessa.

Poiché le vibrazioni hanno un notevole impatto anche sulla webcam, essa è stata montata sulla struttura utilizzando una gomma anti-vibrazioni.

4.5 Arduino

Arduino è una piattaforma elettronica open source per la prototipazione, basata su hardware e software flessibili e semplici da utilizzare.
Nell'ambito del nostro lavoro è stato utilizzato per gestire tutti i sensori presenti a bordo del drone. Per la sua semplicità d'uso infatti è particolarmente indicato per ricevere input provenienti dall'esterno e fornire risposte controllando luci ed altri attuatori. Tra la grande varietà di Arduino presenti sul mercato, abbiamo scelto di utilizzare l'Arduino Nano con microcontrollore Atmel ATmega168 (Figura 4.12).



(a) Fronte (b) Retro

(c) Pinout

Figura 4.12: Arduino Nano

La scelta è ricaduta su questo tipo di Arduino per le sue dimensioni molto ridotte (appena 2x4 cm) e per la presenza di otto ingressi analogici contro i sei normalmente presenti sugli altri modelli. Gli ingressi analogici sono molto importanti poiché permettono di ricevere input da tutti i sensori presenti sul drone.

4.5.1 Sensori di prossimità

Il drone è dotato di quattro sensori di prossimità ad infrarossi posti agli estremi delle direzioni dei due assi di movimento di pitch e roll.

Figura 4.13: Sharp GP2Y0A02

Grazie alla presenza di questi sensori abbiamo potuto implementare un sistema utile ad evitare gli ostacoli. Il numero di sensori però, non è in grado di rilevare qualunque ostacolo potenzialmente pericoloso per il drone. Sarebbe infatti opportuno avere due o tre sensori in ogni lato per poter coprire una superficie maggiore. Tale numero non è però gestibile con l'Arduino Nano.

Il sensore scelto nell'ambito del nostro lavoro è il sensore di prossimità Sharp GP2Y0A02 (Figura 4.13). Questo sensore permette di rilevare oggetti da una distanza minima di 20 cm fino ad una distanza massima di 150 cm con un angolo mostrato in Figura 4.14.

Il sensore opera ad una tensione di 5 V e genera un'uscita in tensione proporzionale alla distanza rilevata. Questo valore viene facilmente letto da Arduino e trasformato in centimetri.

Per stabilizzare la tensione in ingresso ai sensori abbiamo aggiunto, per ognuno di loro, un condensatore da 22 μF tra Vcc e GND.

Come mostra il grafico riportato in Figura 4.15, è impossibile ottenere delle letture coerenti e stabili senza l'ausilio del condensatore, soprattutto quando non viene rilevato nessun oggetto.
I dati mostrati sono stati appunto rilevati senza alcun oggetto nel campo di visibilità del sensore.

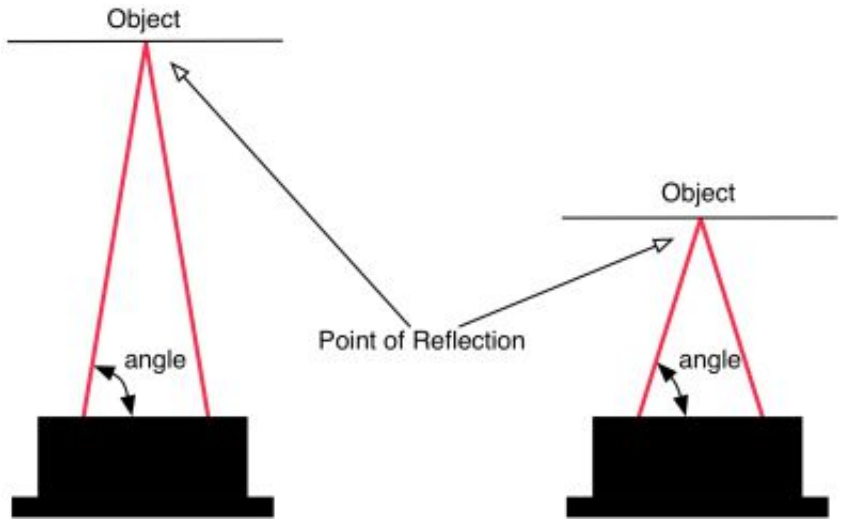


Figura 4.14: Angolo di rilevazione degli oggetti

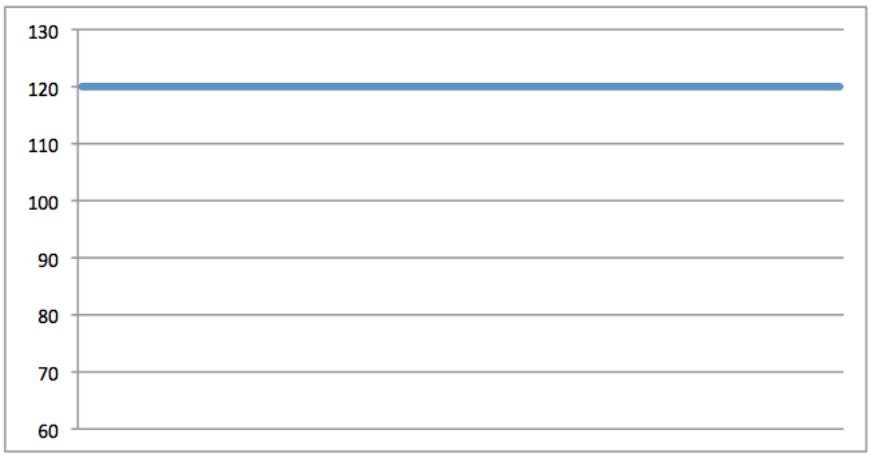
4.5.2 Sonar

Il sonar è un componente molto importante del nostro progetto. Montato al di sotto del drone permette di conoscere in ogni momento la sua esatta altezza dal suolo, consentendoci di lavorare sul comando di throttle per mantenere una determinata quota di volo.

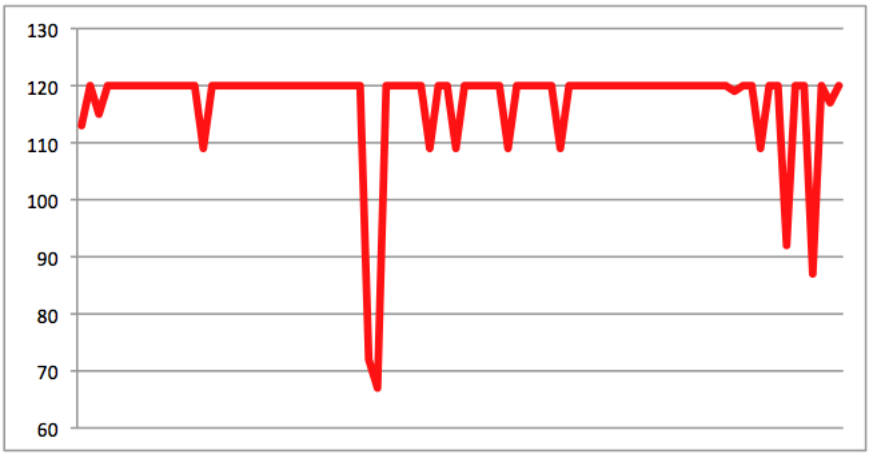
Il sonar scelto nell'ambito della nostra tesi è il MaxBotix MB1340 (Figura 4.16). L'utilizzo di questo sensore è raccomandato sui multirotori per le sue caratteristiche di estrema precisione e tolleranza ai rumori.

Il montaggio di un sonar su un quadricottero deve tener conto di alcune problematiche quali turbolenze d'aria, disturbo acustico delle eliche e rumore sul segnale elettrico. La soluzione ai primi due problemi è quella di montare il sonar il più lontano e riparato possibile dalle eliche. Il miglior posto è sotto al drone vicino il centro del frame.

Per quanto riguarda il problema elettrico invece, la soluzione è quella di creare un filtro della tensione di ingresso con un condensatore da 100 μF e una resistenza da 10 Ω come mostrato in Figura 4.17.
Come i sensori di prossimità, anche questo sensore lavora ad una tensione di 5 V e fornisce un'uscita analogica in tensione proporzionale alla distanza dell'oggetto rilevato. Il sonar ha un range di rilevazione che va dai 20 cm ai 765 cm con una frequenza di lettura di 10 Hz.
Oltre al filtro sulla tensione di ingresso è stato implementato un filtro software sui valori letti.



(a) Con il condensatore



(b) Senza il condensatore

Figura 4.15: Confronto lettura dati dal sensore di prossimità

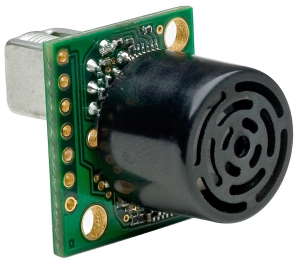


Figura 4.16: Sonar MaxBotix MB1340

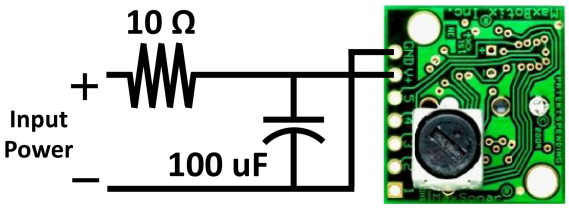


Figura 4.17: Filtro per tensione di ingresso

4.5.3 WiiCam

La WiiCam (Figura 4.18) è un sensore molto performante, che permette di tracciare fino a quattro sorgenti infrarosse contemporaneamente ad una frequenza fino a 100 Hz e con una risoluzione di 1024x768. Tale sensore proviene dal WiiMote, il controller della nota console Nintendo Wii.

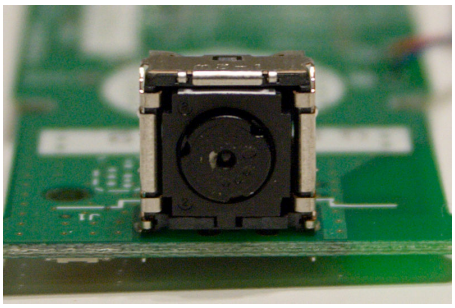


Figura 4.18: WiiCam

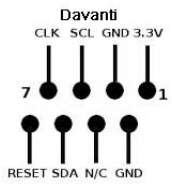
Per il nostro lavoro abbiamo utilizzato questo tipo di sensore per poter tracciare e seguire correttamente un altro drone con a bordo dei led infrarossi. La velocità del sensore infatti, permette di rilevare anche il minimo spostamento, potendo così reagire tempestivamente con il drone autonomo. Davanti alla WiiCam abbiamo montato un pezzo di plastica scura proveniente dallo stesso WiiMote. Questo non permette ai raggi infrarossi più deboli di filtrare all'interno del sensore e di venire riconosciuti come fonti infrarosse. In questo modo ad esempio, i raggi solari non interferiscono con il funzionamento della WiiCam.

Il sensore è stato montato su due servo in cascata (Figura 4.19) che costituiscono un sistema di gimbal. I servo, collegati alla scheda di volo, permettono di mantenere sempre il giusto allineamento del sensore che non risente dunque dell'inclinazione del drone.

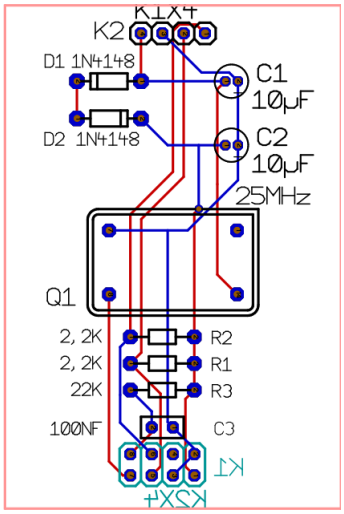
Figura 4.19: La WiiCam montata sui servo

Circuito per Arduino

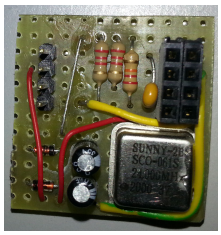
Il sensore dispone di un'interfaccia I2C che può essere letta facilmente da un microcontrollore, nel nostro caso, Arduino.
Contrariamente al sonar e ai sensori di prossimità, la WiiCam è un dispositivo che opera alla tensione di 3.3 V. E' quindi necessaria una conversione dai 5 V di Arduino ai 3.3 V del sensore. Per fare questo occorrono due diodi in serie che trasformano i 5 V in circa 3.6 V da dove il sensore prende la sua alimentazione.



(a) WiiCam pin



(b) Schema circuitale



(c) Scheda prodotta

Figura 4.20: WiiCam scheda e circuito

Due resistenze di pull-up sulla linea I2C, permettono il corretto funzionamento della linea stessa.
Un altro componente fondamentale del circuito è l'oscillatore al quarzo necessario al sensore. Questo oscillatore può essere da 24 o 25 MHz.
In Figura 4.20 sono riportati i pin del sensore (a), lo schema circuitale completo (b) e la scheda prodotta (c).

4.5.4 Laser

Per poter realizzare un laser game è necessario avere a bordo un puntatore laser.

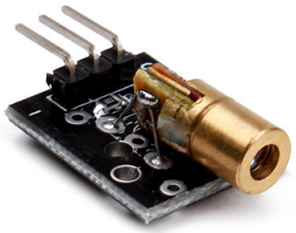


Figura 4.21: Il modulo laser utilizzato

Il modulo scelto per questo compito è il Modulo Laser 650 nm (Figura 4.21) che opera a 5 V e pertanto è completamente compatibile con Arduino. Per farlo accendere è necessario solamente portare un pin a livello di tensione alto.
Il puntatore laser è montato sul frame del drone nella parte anteriore, sotto alla WiiCam e alla webcam. In questo modo è possibile colpire l'avversario quando è in posizione favorevole.

Per poter segnalare al giocatore umano la zona di atterraggio durante la fase di ricarica, abbiamo montato un secondo modulo laser anch'esso nella parte anteriore del drone. Esso è puntato verso il terreno e indica il punto centrale della zona di atterraggio.
In Figura 4.22 sono visualizzati i due moduli laser montati sul drone.

4.5.5 Led

Per poter avere un riscontro immediato e chiaro di cosa percepisce il drone in ogni momento, abbiamo deciso di installare alcuni led di segnalazione.

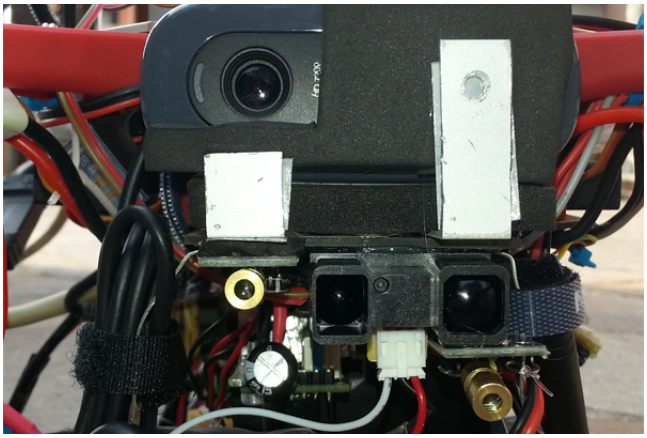


Figura 4.22: I laser anteriori montati sul drone

I led sono montati sul retro del drone in modo tale da essere sempre ben visibili alla persona che controlla il corretto funzionamento del drone in autonomia. In questo modo infatti la persona in questione può sempre sapere se il drone è in una situazione di pericolo ed intervenire tempestivamente, senza aver bisogno di guardare un pc.

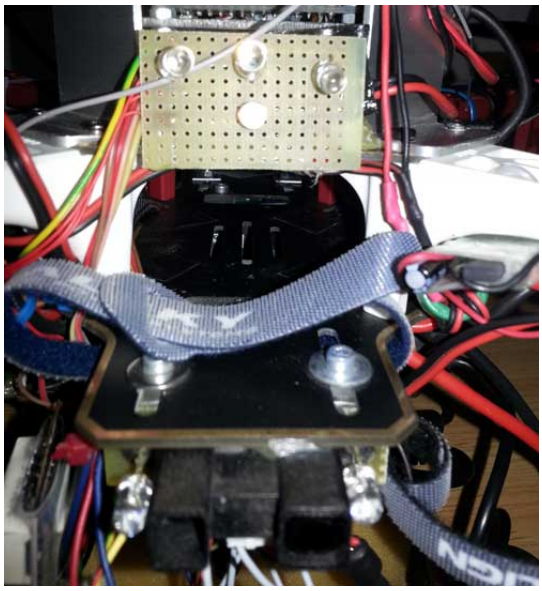


Figura 4.23: I led posteriori presenti

I led presenti (Figura 4.23) sono divisi in due gruppi.
Il primo gruppo è composto da quattro led rossi di cui tre allineati ed uno centrato sotto di essi.
I tre led allineati forniscono un'informazione su cosa il drone stia vedendo. I due led sui lati vengono accesi quando la webcam riconosce il drone avversario.
Il led centrale della stessa fila, invece, viene acceso quando la WiiCam percepisce delle sorgenti infrarosse nel suo raggio di azione.
Il led sotto di essi è il più importante di tutti in termini di sicurezza poiché viene acceso quando Arduino non comunica più con il software a bordo di Odroid. In questa situazione occorre uscire immediatamente dalla modalità automatica ed atterrare manualmente poiché il sistema di controllo del drone non ha più a disposizione nessun sensore che gli permetta di capire cos'ha intorno a sé.

Il secondo gruppo di led è formato da due led posti uno alla destra ed uno alla sinistra del sensore di prossimità sul retro del drone. Questi due led servono a capire se il drone sta volando alla quota desiderata. Il led alla sinistra del sensore, di colore verde, viene acceso quando il drone vola ad una quota inferiore a quella indicata. Il led alla destra invece, è di colore blu e viene acceso quando il drone vola ad una quota superiore. Vengono accesi entrambi quando il drone vola alla giusta distanza dal suolo.

4.5.6 Schema

I collegamenti tra tutte le parti descritte ed Arduino sono riportati in Figura 4.24.

4.6 Batterie

Le batterie utilizzate nell'ambito del nostro lavoro sono di tipo litio-polimero, più comunemente indicate come LiPo o Li-Poly. Questo tipo di batterie presentano numerosi vantaggi rispetto alle loro predecessori, le batterie litio-ione.
In primo luogo, le LiPo sono più leggere e sagomate poiché non necessitano di alcun contenitore metallico. Inoltre, sempre per lo stesso motivo e per la mancanza di spazi tra le celle, la densità energetica delle LiPo è maggiore più del 20% rispetto alle litio-ione. In questo modo è possibile avere batterie piccole e leggere con una grandissima capacità di scarica.
Vengono largamente utilizzate nell'ambito dell'aeromodellismo proprio per

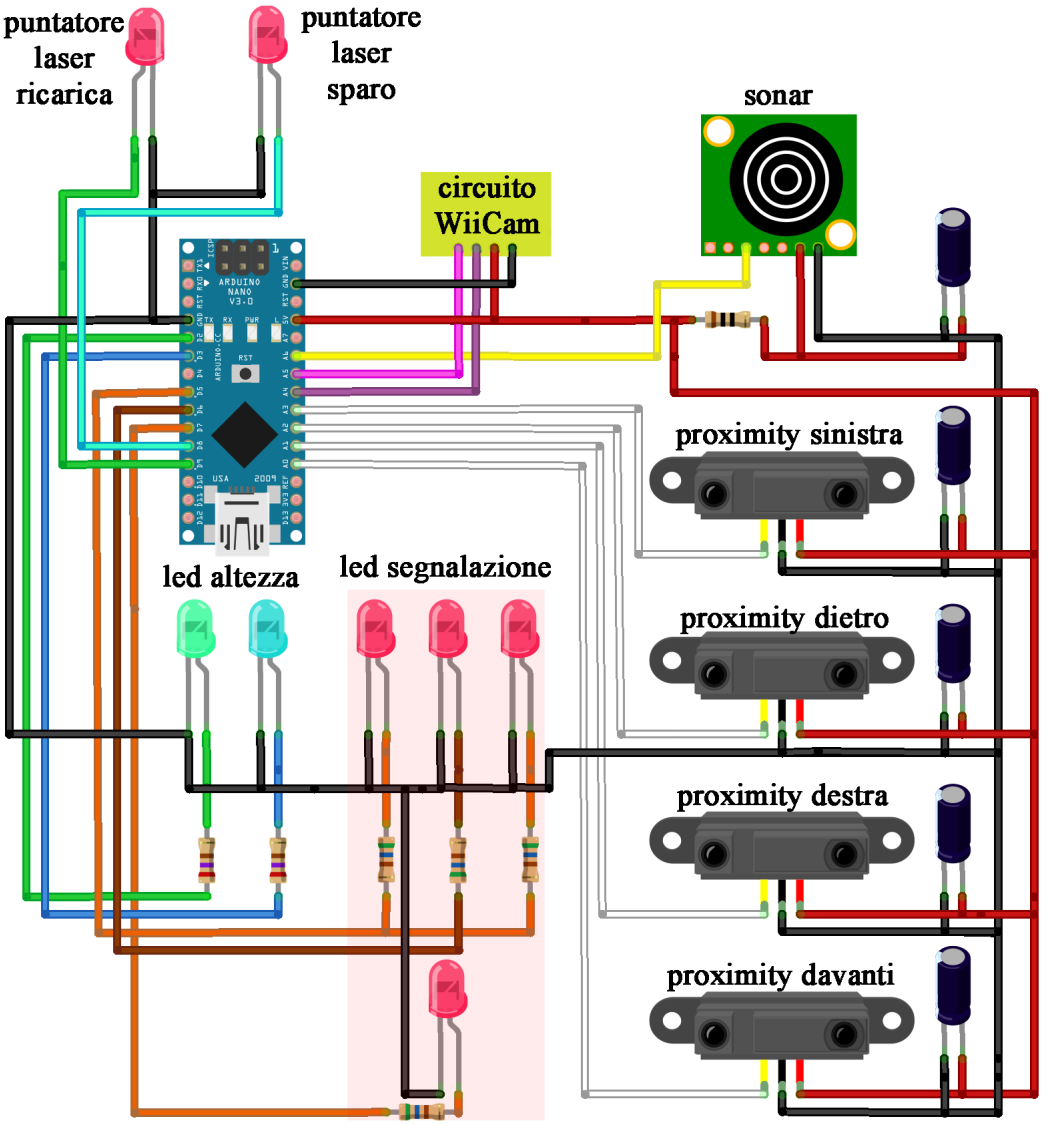


Figura 4.24: Schema collegamenti arduino-sensori

le caratteristiche descritte che giustificano i prezzi leggermente più elevati rispetto alle litio-ione.

Le LiPo sono composte da più celle in serie o in parallelo per aumentarne la durata oppure il voltaggio. Ogni cella ha un valore di tensione nominale di 3.7 V e può andare da un minimo di 2.7 V da scarica ad un massimo di 4.23 V a piena carica.
Scendere sotto il limite minimo può compromettere seriamente la funziona-

lità della batteria. Il valore di soglia sotto cui non si dovrebbe scendere è di 3 V per cella. Per evitare che ciò accada vengono utilizzati dei segnalatori acustici che emettono un allarme appena una cella scende sotto un valore soglia prestabilito.

(a) Batteria per il volo (b) Batteria per Odroid ed elettronica

Figura 4.25: Le batterie utilizzate

A bordo del drone sono presenti due batterie LiPo diverse.
La prima batteria (Figura 4.25a) serve per alimentare i motori e la scheda di volo ed è dunque utilizzata solo per volare.
La batteria scelta tra la vastissima gamma presente in commercio, è la Turnigy nano-tech 3300 mah 40-80C da 11.1 V (3 celle). Questa batteria ci permette di produrre i 120 A che possono richiedere al massimo i motori del drone, ha un peso di soli 264 g e garantisce una durata di circa 7 minuti in volo.

La seconda batteria (Figura 4.25b), invece, serve per alimentare l'Odroid e tutta l'elettronica presente. La scelta è ricaduta sulla Turnigy nano-tech 1000 mah 20 40C da 7.4 V (2 celle) soprattutto per il suo peso di 51 g. Poiché l'Odroid lavora a 5 V, abbiamo utilizzato un BEC per trasformare i 7.4 V nominali della batteria in 5 V stabili. Oltre all'Odroid, questa batteria alimenta l'hub usb, Arduino e tutti i sensori presenti. Con tutti i componenti connessi e funzionanti la batteria ha una durata di circa 45 minuti.

I connettori utilizzati sul drone e sulle batterie sono di tipo TRX (Figura 4.26). Questi connettori sono risultati i migliori tra quelli testati poiché

(a) TRX Maschio (b) TRX Femmina

Figura 4.26: Connettori TRX

garantiscono un'ottima tenuta contrariamente, ad esempio, ai connettori mini tamiya.

4.7 Radiocomando

Il radiocomando è un dispositivo che permette di comandare a distanza il drone, utilizzando le onde radio come mezzo di trasmissione.
Nell'ambito del nostro lavoro, viene utilizzato per fornire il comando di decollo e di atterraggio al drone autonomo e per ragioni di sicurezza. In qualunque momento, infatti, è possibile passare alla modalità manuale con uno specifico interruttore presente sul radiocomando stesso, potendo così pilotare manualmente il drone.

Figura 4.27: Radiocomando Spektrum DX5e

Il trasmettitore scelto è lo Spektrum DX5e Modo 2 (Figura 4.27) poiché possiede il numero minimo di canali (6) utili al nostro lavoro ed ha un

ottimo rapporto qualità/prezzo. Questo specifico radiocomando lavora alla frequenza di 2.4 Ghz ed ha un raggio d'azione maggiore di 1 km.
Modo 2 significa che il radiocomando in questione ha il comando del gas (throttle) sulla sinistra.

4.8 Interconnessioni e risultato finale

Lo schema in Figura 4.29 riporta le connessioni tra tutti i componenti spiegati nel capitolo corrente.
In Figura 4.28 viene mostrato il drone autonomo costruito.

Figura 4.28: Drone autonomo completo

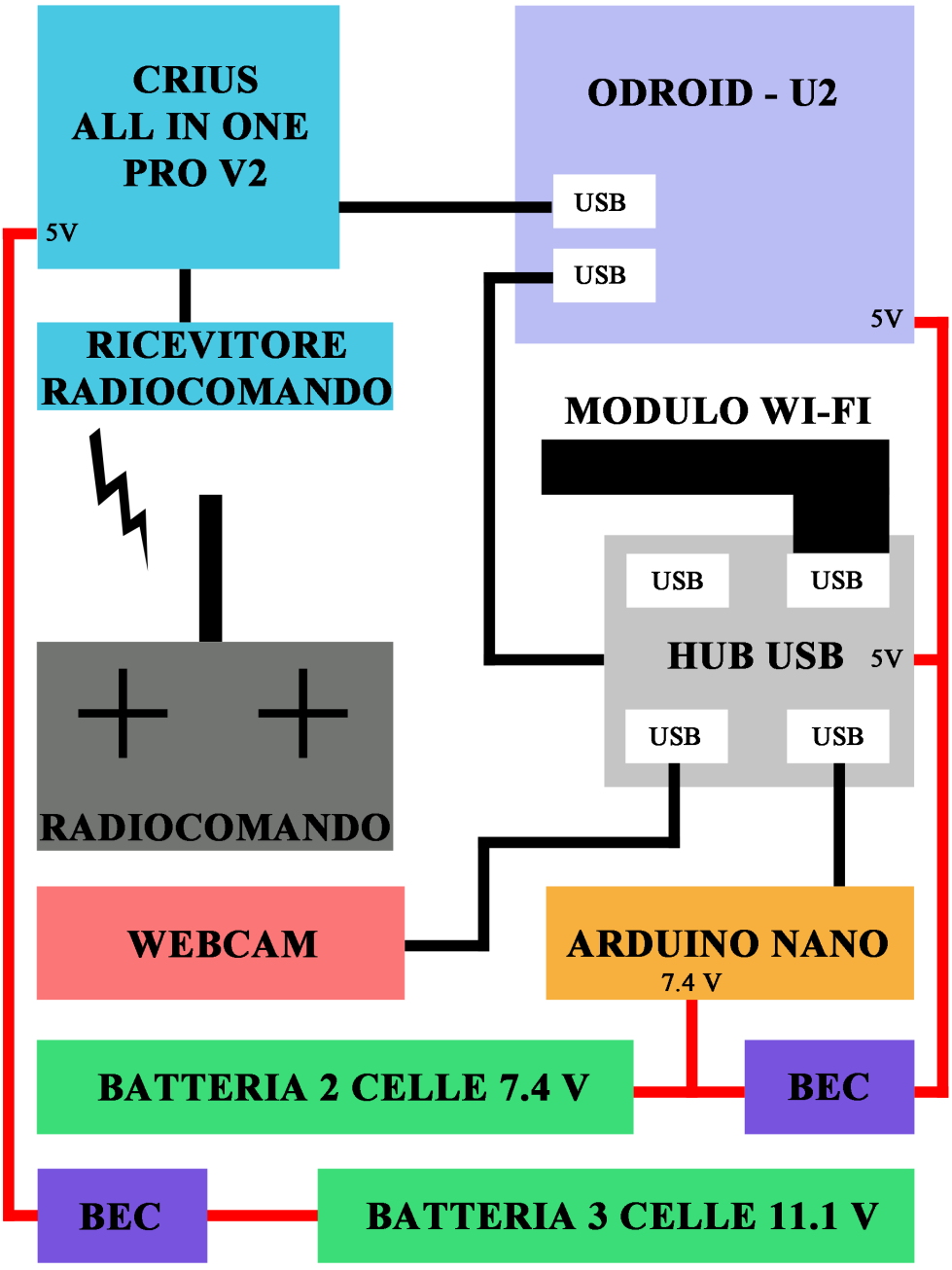


Figura 4.29: Interconnessioni di tutto il sistema

Capitolo 5

Architettura software

"I computer sono incredibilmente veloci, accurati e stupidi. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti. Insieme sono una potenza che supera l'immaginazione"

Albert Einstein

La progettazione di un software per applicazioni robotiche deve tenere conto dell'implicazione di molte parti diverse tra loro che interagiscono per concorrere allo scopo finale. È dunque utile utilizzare un approccio modulare, che consiste nella realizzazione di programmi suddivisi in moduli che svolgono una determinata funzione.
Ogni modulo è il più possibile indipendente dagli altri facilitando così la scrittura del codice in team poiché è ridotta l'interazione tra i programmatori. Per ogni modulo risultano semplificate anche le operazioni di manutenzione e di test.
I moduli scritti possono essere poi riutilizzati in altri progetti diminuendo il tempo di sviluppo.
Essi comunicano tra loro creando un'architettura complessa ma allo stesso tempo flessibile per la gestione del comportamento del drone.

Nell'ambito del nostro lavoro abbiamo realizzato un'applicazione funzionante sul computer di bordo del drone e quattro applicazioni esterne, utilizzabili da un PC remoto. L'applicazione a bordo gestisce tutta la logica del volo e del gioco ed è indispensabile per il funzionamento del drone autonomo.
Le applicazioni esterne servono invece per monitorare lo stato in cui si trova il drone e nessuna di esse è essenziale per il suo funzionamento.
Il collegamento tra PC e drone avviene tramite una rete Wi-Fi dedicata.
La comunicazione utilizza la piattaforma Robot Operating System (ROS)[15], la quale fornisce librerie e strumenti che aiutano lo sviluppo di applicazioni

robotiche. Il sistema si basa su pacchetti costituiti da uno o più processi (nodi) i quali si scambiano messaggi (strutture dati) pubblicati su topic.

5.1 Applicazione a bordo del drone

L'applicazione che abbiamo sviluppato è stata scritta in C++ utilizzando l'editor Code::Blocks [4]. È un'applicazione modulare basata su pthread, messi a disposizione dal linguaggio.
I moduli presenti sono:

- **BoardManagement:** si occupa della gestione della comunicazione con la scheda di volo
- **SensorManagement:** si occupa della gestione della comunicazione con Arduino e i relativi sensori
- **ServerConnection:** si occupa dell'invio dei dati per monitorare il volo e il gioco
- **ServerReader:** si occupa della ricezioni di comandi da terra
- **VideoManagement:** si occupa dell'acquisizione delle immagini dalla webcam e della loro elaborazione
- **FlightManagement:** si occupa della gestione del volo ed utilizza i dati provenienti da tutti gli altri moduli

Ogni modulo utilizza delle classi di supporto in cui vengono memorizzati e successivamente letti i valori provenienti dai vari sensori.

5.1.1 BoardManagement

La comunicazione tra la scheda di volo e l'elaboratore di bordo avviene in maniera seriale su bus USB.
La scheda di volo infatti è dotata di una porta micro USB che è collegata all'Odroid.
Il software Multiwii installato come controllore sulla scheda di volo, mette a disposizione un protocollo chiamato MSP (Multiwii Serial Protocol) che permette di comunicare con la scheda. Il modulo in questione richiede ciclicamente alla scheda i dati necessari e li processa.
In figura 5.1 è riportato il diagramma delle classi di questo modulo.

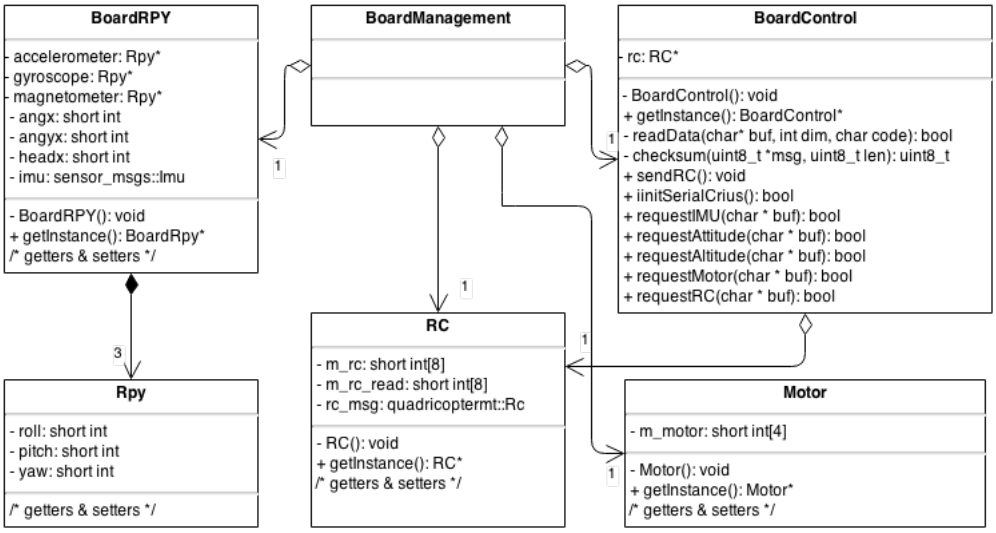


Figura 5.1: Diagramma delle classi del modulo BoardManagement

Multiwii Serial Protocol

Il Multiwii Serial Protocol permette di inviare richieste o comandi alla scheda e di ricevere risposte.
I messaggi diretti alla scheda sono strutturati come:

- Tre byte contenti i caratteri $ M <
- Un byte che rappresenta il numero N di byte aggiuntivi da inviare
- Un byte con il codice del comando
- N byte di dati da inviare
- Un byte di checksum

I messaggi provenienti dalla scheda sono invece strutturati come:

- Tre byte contenti i caratteri $ M >
- Un byte con il codice del comando
- Un byte che rappresenta il numero N di byte inviati dalla scheda
- N byte di dati inviati
- Un byte di checksum

Il checksum viene calcolato come l'or esclusivo bit a bit delle varie componenti del messaggio esclusi i primi tre byte.

Messaggi scambiati

Nell'ambito del nostro lavoro abbiamo utilizzato tre messaggi di richiesta ed un messaggio di comando. In questo modo possiamo controllare tutti gli aspetti che interessanti del volo ed inviare i comandi alla scheda.
Le richieste utilizzate sono:

- **MSP_RAW_IMU:** vengono richiesti alla scheda tutti i dati provenienti dalla IMU. In particolare otteniamo come risposta nove byte che rappresentano i valori di accelerometro, giroscopio e magnetometro per ognuno dei tre assi sui quali lavorano.

- **MSP_ATTITUDE:** vengono richiesti gli angoli di inclinazione del drone. Otteniamo come risposta due byte per ogni angolo, che rappresentano quanto è piegato sugli assi di roll e pitch e dove è orientato rispetto al nord. Questi valori vengono calcolati direttamente dalla scheda a partire dai dati rilevati dalla IMU.

- **MSP_RC:** viene richiesto lo stato del radiocomando. Otteniamo come risposta due byte per ognuno degli otto canali del radiocomando. Grazie a questa richiesta possiamo controllare se l'interruttore fisico è impostato su manuale o automatico ed agire di conseguenza.

I valori ricevuti dalla varie richieste vengono salvati in classi specifiche e sono utilizzati da altri moduli per raggiungere il loro scopo.

L'unico comando che inviamo alla scheda è il comando di **MSP_SET_RAW_RC**. Esso serve per settare i valori degli otto canali del radiocomando in modo tale da simularlo. In questo modo possiamo fornire i comandi via seriale alla scheda di volo che li processa come se arrivassero dal radiocomando vero e proprio.
Per ogni canale dobbiamo inviare due byte che rappresentano il suo valore. Dopo aver ricevuto il messaggio, il Multiwii risponde inviando una conferma.

Poiché il comando di settaggio del radiocomando viene mandato in maniera asincrona rispetto alle richieste, abbiamo utilizzato un lock per evitare interferenze nei messaggi.
In particolare una risorsa che intende inviare un comando o una richiesta alla scheda, acquisisce il lock prima dell'invio, attende il suo messaggio di risposta e poi rilascia il lock.

5.1.2 SensorManagement

La comunicazione tra Arduino e l'elaboratore avviene, come nel caso della scheda di volo, in maniera seriale su bus USB.
I due componenti sono infatti collegati tramite un cavo mini USB.
Il protocollo di comunicazione è però differente rispetto al precedente. In particolare, Arduino non aspetta nessun messaggio di richiesta, ma invia immediatamente i dati appena sono disponibili e rimane sempre in attesa di comandi da parte dell'Odroid.
In figura 5.2 è riportato il diagramma delle classi di questo modulo.

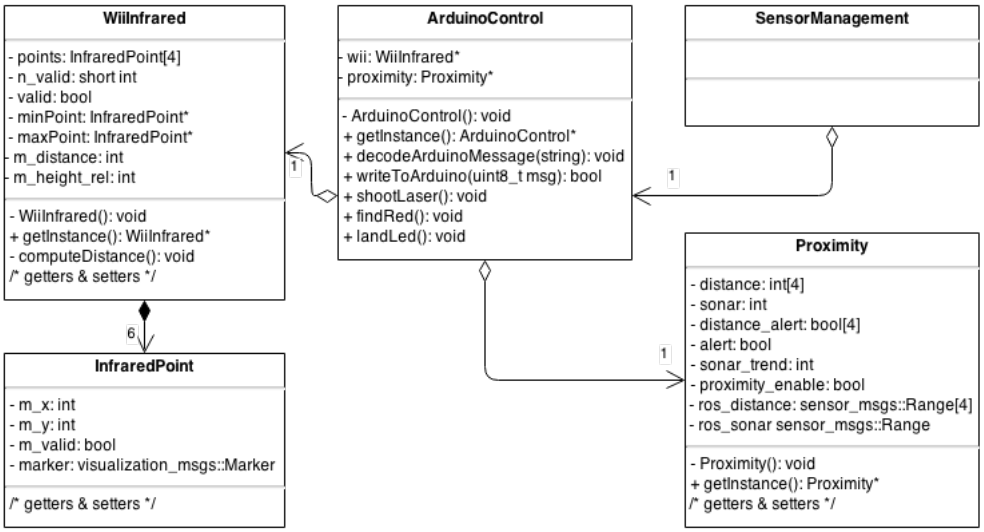


Figura 5.2: Diagramma delle classi del modulo SensorManagement

Messaggi inviati

I messaggi che vengono inviati da Arduino sono messaggi di testo e possono essere di due tipi: distanza ed infrarossi.
Il primo tipo di messaggio contiene tutte le distanze lette dai quattro sensori di prossimità e dal sonar. Il formato del messaggio è il seguente:

@p distanza1 distanza2 distanza3 distanza4 distanza_sonar

Questo messaggio viene inviato con una frequenza di circa 25 Hz.

Il secondo tipo di messaggio contiene le informazioni fornite dalla WiiCam. In particolare fornisce le coordinate x,y delle quattro sorgenti infrarosse che

può rilevare la WiiCam. Il formato del messaggio è il seguente:

@i x1 y1 x2 y2 x3 y3 x4 y4

Questo messaggio viene inviato con una frequenza di circa 50 Hz.

Il modulo considerato resta sempre in ascolto di nuovi messaggi sulla seriale e, una volta capito che tipo di messaggio è stato ricevuto, manda le informazioni alle classi dedicate che memorizzano correttamente i dati ricevuti.
Qualunque altro tipo di messaggio viene automaticamente scartato.

Messaggi ricevuti

Arduino può ricevere alcuni comandi dall'elaboratore di bordo in modo tale da compiere determinate azioni. In questo caso la comunicazione si basa semplicemente sulla ricezione di un singolo carattere.
I messaggi possibili sono:

- **SPARA:** quando viene ricevuto questo comando, viene azionato il laser per un secondo.
- **TROVATO_ROSSO:** si accendono per un secondo i led di segnalazione del rosso. Questo significa che la webcam ha rilevato il drone avversario.
- **LASER_RICARICA:** viene acceso per 30 secondi il laser anteriore di ricarica che mostra al giocatore il centro dell'area in cui far atterrare il drone telecomandato.
- **SONO_VIVO:** questo messaggio viene inviato dal nostro programma ogni secondo. Se per 3 secondi non viene ricevuto, supponiamo che la comunicazione tra Arduino e il nostro sofware si sia interrotta per qualche motivo. In questo caso si accende un led di segnalazione.
- **CAMBIO_ALTEZZA:** quando viene ricevuto un byte il cui valore è compreso tra 1 e 50, significa che è cambiata la quota di volo che il drone deve mantenere. La nuova quota in centimetri viene trovata moltiplicando il valore ricevuto per 10. Questa indicazione ci permette di accendere adeguatamente i led di segnalazione dell'altezza per poter capire molto intuitivamente se il drone sta volando alla giusta quota.

Il led per la segnalazione degli infrarossi viene invece gestito direttamente da Arduino, senza passare per il software principale. Questo led infatti viene acceso automaticamente quando la WiiCam rileva almeno una sorgente infrarossa valida.

5.1.3 ServerConnection

La comunicazione con le applicazioni esterne avviene tramite ROS. Il nodo ROS presente nell'applicazione è il nodo **server**.
Lo scopo di questo modulo è quello di pubblicare periodicamente tutti i dati di telemetria, visualizzati tramite un PC remoto, per controllare il corretto svolgimento del volo e del gioco.
I dati che vengono pubblicati sono precaricati in strutture dati specifiche prima di essere inviati.
I dati pubblicati vengono ricevuti da alcune applicazioni presenti sul PC remoto che restano in ascolto dei topic interessati.
I topic utilizzati sono range, *imu*, *infrared*, *rc*, *camera/image*, *gioco* e *tempo*.
Per non sovraccaricare eccessivamente la rete, i dati sono pubblicati con frequenza pari a 6 Hz. In figura 5.3 è riportato il diagramma delle classi di questo modulo.

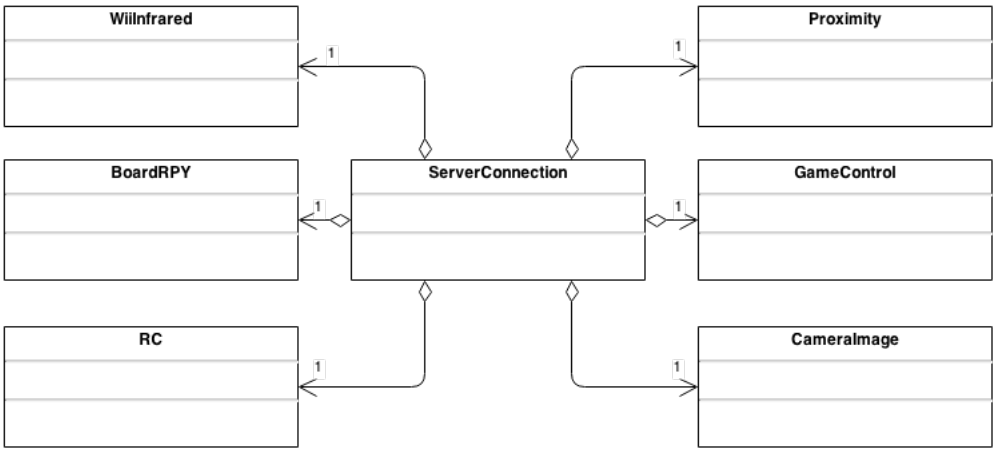


Figura 5.3: Diagramma delle classi del modulo ServerConnection

Topic range

Il topic *range* si occupa della pubblicazione dei dati riguardanti i sensori di prossimità e il sonar.
Per ogni sensore viene utilizzato il messaggio *sensor_msgs/Range.msg* [13]

fornito da ROS e definito come segue:

uint8	radiation_type
float32	field_of_view
float32	min_range
float32	max_range
float32	range

Topic imu

Il topic *imu* si occupa della pubblicazione dei dati provenienti dalla IMU.
Il messaggio utilizzato è di tipo *sensor_msgs/Imu.msg* [11] fornito da ROS e definito come segue:

geometry_msgs/Quaternion	orientation
geometry_msgs/Vector3	angular_velocity
geometry_msgs/Vector3	linear_acceleration

Poiché i dati forniti dalla nostra IMU sono in un formato differente, abbiamo provveduto a convertirli per poterli pubblicare correttamente.

Topic infrared

Il topic *infrared* si occupa della pubblicazione di marcatori che segnalano se e dove, sono state viste sorgenti infrarosse. Si occupa inoltre di pubblicare un marcatore che segnala l'individuazione del rosso da parte della webcam.
Sono presenti quattro marcatori per gli infrarossi circolari, di colore verde.
Il marcatore del rosso è invece quadrato e di colore rosso.
Il loro posizionamento è stato studiato in modo tale da rendere immediata l'idea di dove si trovino rispetto alla prua del drone automatico.
Il messaggio utilizzato è di tipo *visualization_msgs/Marker.msg* [12] fornito da ROS e definito come segue:

int32	type
geometry_msgs/Pose	pose
geometry_msgs/Vector3	scale
std_msgs/ColorRGBA	color
duration	lifetime

Topic rc

Il topic *rc* si occupa della pubblicazione dei valori di roll, pitch, yaw e throttle calcolati. Questi sono i valori che vengono forniti alla scheda di volo per permettere il corretto svolgimento del gioco.
A differenza degli altri messaggi, questi non sono ascoltati da Rqt ma da un'applicazione che abbiamo creato ad-hoc. Grazie a questa applicazione possiamo monitorare agilmente i comandi ed agire tempestivamente in caso di errori.
Il messaggio utilizzato è di tipo *Rc.msg* che abbiamo creato appositamente, poiché non ne esisteva uno adeguato fornito da ROS. Il messaggio è definito come segue:

uint16	roll
uint16	pitch
uint16	yaw
uint16	throttle

Topic camera/image

Il topic *camera/image* si occupa della pubblicazione delle immagini provenienti dalla webcam.
Esso è basato sull'utilizzo di *image_transport* [10] la cui parte fondamentale è la matrice contenente l'immagine da pubblicare.

Topic gioco

Il topic *gioco* si occupa della pubblicazione di messaggi riguardanti la partita in corso tra il drone autonomo e quello telecomandato.
I messaggi pubblicati vengono ascoltati da un'applicazione creata ad-hoc, che ha lo scopo di informare il giocatore umano circa lo stato della partita.
Il messaggio utilizzato è di tipo *std_msgs/String.msg* [14] fornito da ROS e definito semplicemente come una stringa di testo.
Le stringhe inviate da questo topic sono:

- **inizio:** segnala l'inizio della partita ed è inviato appena il drone autonomo accende i motori. Dal momento della sua ricezione, il drone telecomandato può decollare.
- **sparo:** segnala che il drone autonomo sta sparando il laser per colpire il suo avversario.
- **preso:** segnala che il drone autonomo ha colpito l'avversario.

- **atterra:** segnala che il drone autonomo sta per iniziare l'atterraggio poiché ha terminato l'energia. Dal momento della sua ricezione, il drone telecomandato ha 30 secondi a disposizione per impedire la ricarica di energia e vincere la partita.

- **decolla:** segnala che il drone autonomo sta ripartendo dopo la fase di ricarica.

- **vinto:** segnala la fine della partita. Il giocatore umano ha vinto. Al momento della sua ricezione, il drone telecomandato deve atterrare.

- **perso:** segnala la fine della partita. Il giocatore umano ha perso. Al momento della sua ricezione, il drone telecomandato deve atterrare.

Topic tempo

Il topic *tempo* si occupa della pubblicazione del tempo rimanente per giocare la partita.
I suoi messaggi sono ascoltati dalla stessa applicazione che ascolta quelli del topic *gioco*.
Il messaggio utilizzato è di tipo *std_msgs/String.msg* ed il formato della stringa inviata è minuti:secondi.

5.1.4 ServerReader

Questo modulo completa la parte di comunicazione con le applicazioni esterne di cui fa parte anche ServerConnection.
Esso è sempre basato su ROS ed ascolta i messaggi pubblicati dal topic *remotecommand*. Questi messaggi sono prodotti da un'applicazione creata appositamente che gira sul computer remoto.
Il messaggio utilizzato è di tipo *std_msgs/String.msg*
Le stringhe che è possibile inviare sono:

- **land:** atterraggio. Alla ricezione di questo comando, il drone automatico atterra immediatamente.

- **laser:** viene mandato ad Arduino il comando di accendere il puntatore laser.

- **enable:** viene abilitata la lettura dei sensori di prossimità.

- **disable:** viene disabilitata la lettura dei sensori di prossimità.

- **h:** viene modificata la quota di volo del drone. In particolare il comando h deve essere seguito da un numero che rappresenta la nuova altezza in cm da mantenere.

Il modulo ascolta costantemente i messaggi e, in base al comando contenuto, delega il metodo corretto all'esecuzione di tale comando. I comandi sconosciuti vengono direttamente scartati.
Questo modulo è stato utilizzato molto in fase di debug, ma può tornare utile anche durante il gioco.
In figura 5.4 è riportato il diagramma delle classi di questo modulo.

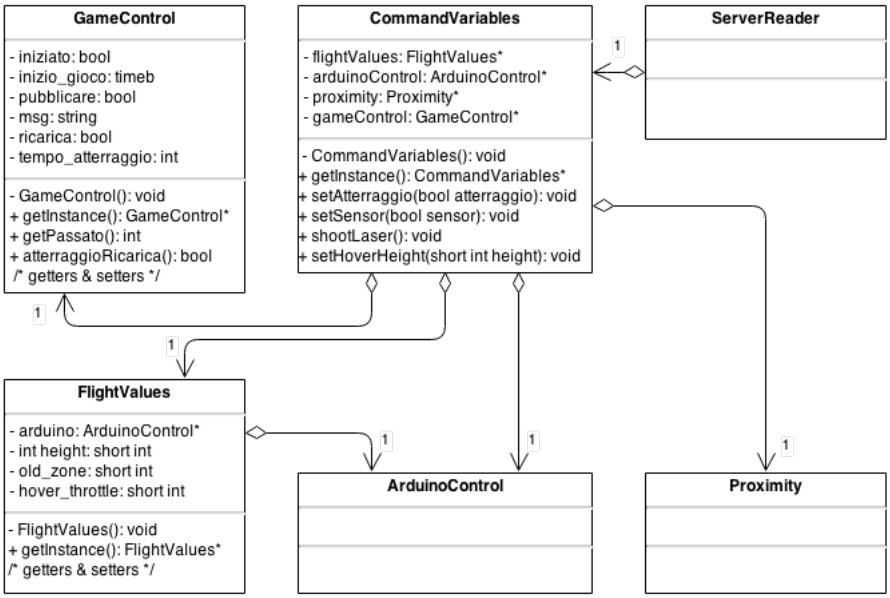


Figura 5.4: Diagramma delle classi del modulo ServerReader

5.1.5 VideoManagement

Il modulo di VideoManagement si interfaccia con la webcam al fine di acquisire ed elaborare le immagini ricevute. È dunque il responsabile di tutta la parte di visione del drone.
Le immagini vengono acquisite ad una risoluzione di 640x480 pixel. Questo ci permette di elaborarle ad una velocità di 13 fps e, allo stesso tempo, di rilevare il drone avversario fino ad una distanza di 12 m.
L'analisi delle immagini non viene effettuata quando il drone rileva delle sorgenti infrarosse valide poiché in quel caso le decisioni di volo vengono prese a partire dagli infrarossi stessi.

Il funzionamento e le motivazioni dell'elaborazione verranno trattate dettagliatamente nel prossimo capitolo.
In figura 5.5 è riportato il diagramma delle classi di questo modulo.

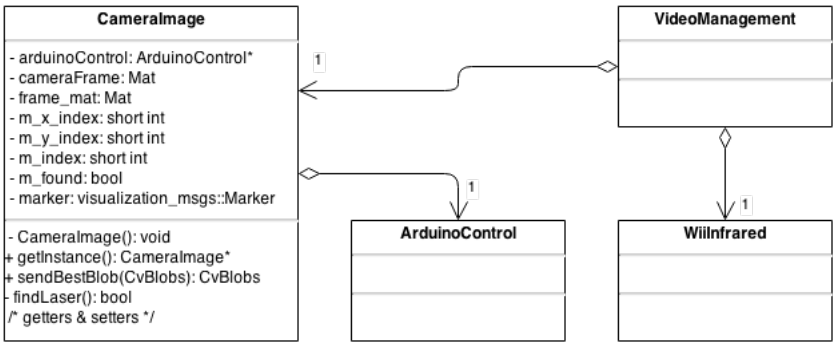


Figura 5.5: Diagramma delle classi del modulo VideoManagement

5.1.6 FlightManagement

Il FlightManagement è il modulo principale del software che abbiamo sviluppato. Esso ha infatti il compito di calcolare i valori di roll, pitch, yaw e throttle che vengono forniti alla Crius in modo tale da permettere il corretto svolgimento del gioco e del volo.
Per fare ciò utilizza tutti i dati predisposti dagli altri moduli. Contiene inoltre la logica del gioco.
Il suo funzionamento è spiegato dettagliatamente nel prossimo capitolo.
In figura 5.6 è riportato il diagramma delle classi di questo modulo.

5.2 Applicazioni esterne

Le applicazione esterne che abbiamo sviluppato sono state scritte in C++ ed utilizzano la piattaforma ROS per la comunicazione con l'applicazione principale posta a bordo del drone.
Esse non sono fondamentali per il funzionamento del sistema, ma molto utili per capire lo stato del drone autonomo.

5.2.1 Monitoraggio volo

Questa applicazione comunica costantemente con il drone autonomo, ricevendo le principali informazioni di volo utili a monitorare il suo stato.

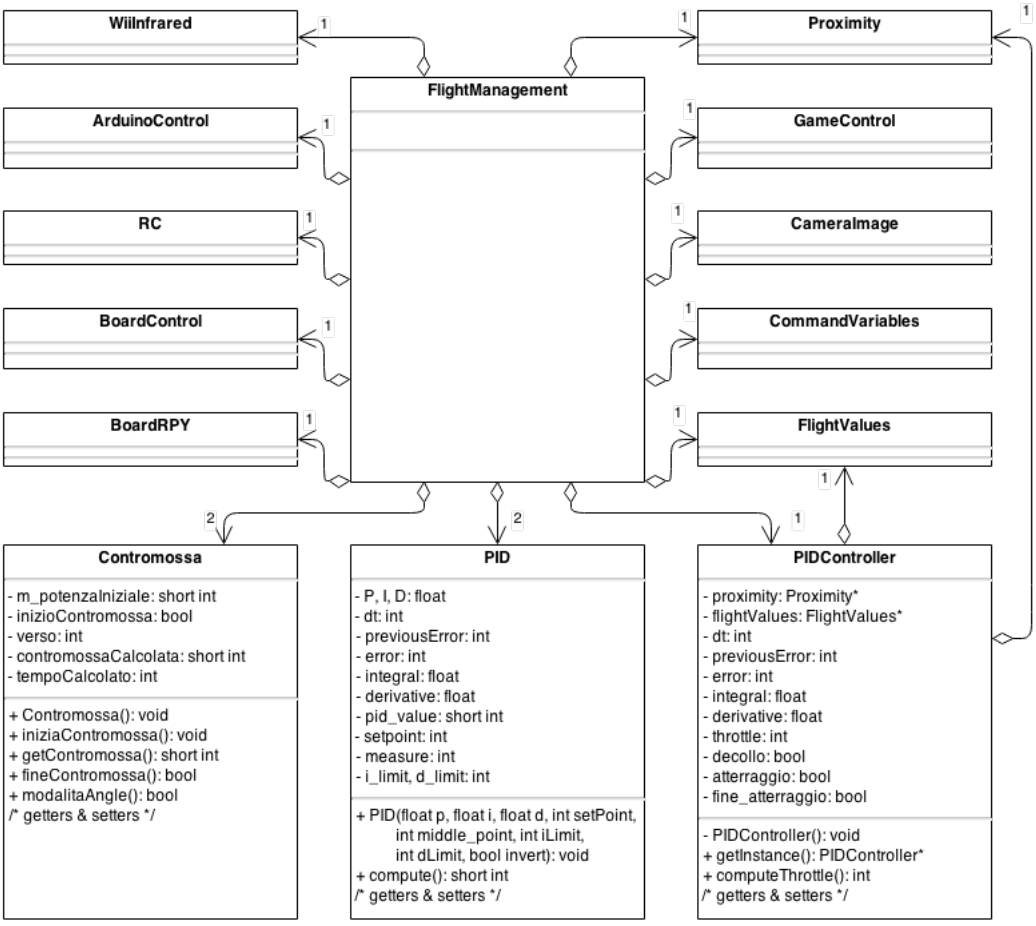


Figura 5.6: Diagramma delle classi del modulo FlightManagement

Per realizzarla ci siamo basati sul pacchetto ROS Rqt, che è in grado di racchiudere in un'unica finestra più applicazioni, tra cui quelle di altri pacchetti ROS.

In particolare abbiamo usufruito dei pacchetti RViz e Image View.
RViz è uno strumento di visualizzazione 3D in grado di ricevere diversi tipi di messaggi. Tramite esso visualizziamo in tempo reale il modello del drone autonomo, il suo orientamento e i dati provenienti dai sensori.
Con Image View invece vengono ricevute le immagini trasmesse dal drone autonomo stesso.
L'applicazione ascolta i topic *range*, *imu*, *infrared* e *camera/image*.
In Figura 5.7 è mostrata la visuale fornita dall'applicazione di monitoraggio del volo.

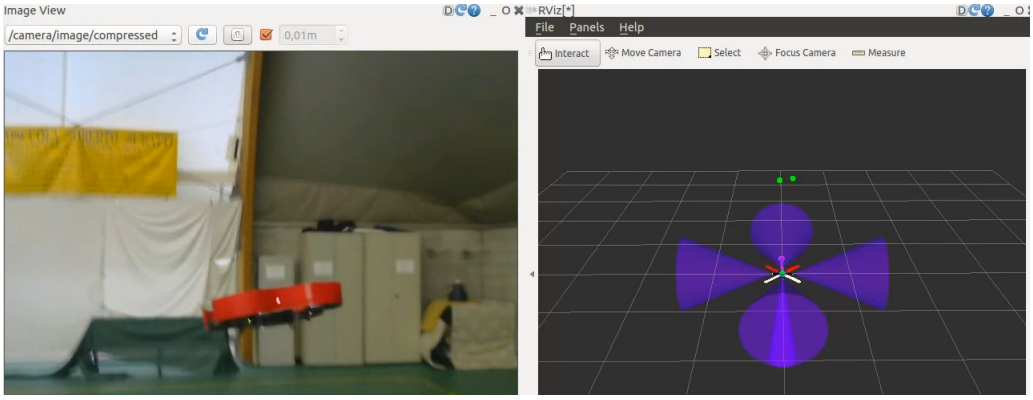


Figura 5.7: Interfaccia di monitoraggio sviluppata

5.2.2 Monitoraggio radiocomando

L'applicazione considerata, utilizzata più in fase di debug, permette di visualizzare agilmente lo stato del radiocomando in modalità automatica. In particolare visualizza a schermo i valori che l'applicazione principale ha calcolato per i canali di roll, pitch, yaw e throttle che permettono il volo autonomo del drone. Il nodo ROS presente è il nodo **listener**, che ascolta il topic *rc*.

5.2.3 Invio comandi

Questa applicazione permette di trasmettere al drone autonomo alcuni comandi che verranno poi eseguiti dal software di bordo.
Il nodo ROS dell'applicazione è il nodo **commander** che pubblica i dati sul topic *remotecommand*.
I comandi possibili sono stati trattati nel dettaglio precedentemente.

5.2.4 Applicazione di interfaccia del gioco

L'applicazione di interfaccia del gioco si presenta con una finestra che mostra un timer e con segnalazioni sonore relative ai vari eventi (Figura 5.8). Il giocatore, essendo concentrato nel pilotaggio del drone, non può seguire costantemente uno schermo che mostra lo stato del gioco e quindi abbiamo predisposto alcuni suoni che gli facciano capire che cosa stia succedendo.
In particolare, vengono riprodotti dei suoni nelle seguenti situazioni:

- Inizio del gioco.
- Fine del gioco.

- Il drone autonomo spara il laser.
- Il drone autonomo colpisce quello telecomandato.
- Il drone autonomo sta per atterrare perché ha terminato l'energia di gioco.
- Il drone autonomo sta per decollare.

Il nodo ROS presente nell'applicazione è il nodo **game** che è in ascolto dei topic *gioco* e *tempo*.

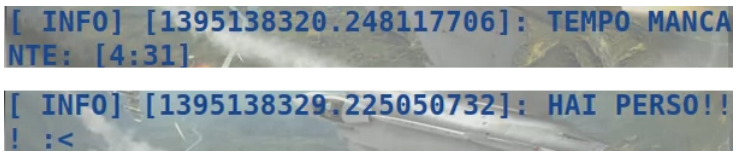


Figura 5.8: Applicazione di interfaccia del gioco

5.3 Diagramma ROS

Lo schema in Figura 5.9 riporta le relazioni tra le varie applicazioni spiegate nel capitolo corrente.

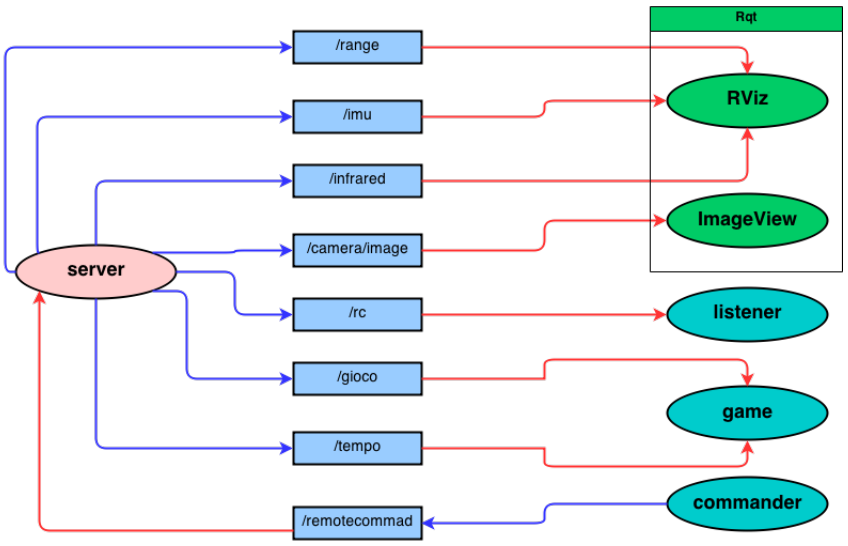


Figura 5.9: Diagramma delle relazioni tra nodi e topic ROS

Capitolo 6

Logica di gioco

> *"Parte della disumanità del computer sta nel fatto che, una volta programmato e messo in funzione, si comporta in maniera perfettamente onesta."*
>
> Isaac Asimov

La complessa gestione del gioco viene suddivisa in più stati ognuno dei quali agisce per raggiungere un determinato obiettivo.
Una divisione di questo tipo ha permesso di sviluppare le singole fasi di gioco separatamente, permettendo di testare stato per stato l'intera logica del gioco.
In questo modo non risulta nemmeno complesso inserire un nuovo stato di gioco oltre quelli già presenti dando estendibilità al software.
Il modulo di FlightManagement utilizza i dati provenienti da tutti gli altri moduli per determinare quale stato eseguire.
Lo stato evocato compie diverse analisi dei dati per poi far svolgere al drone autonomo delle azioni per avvicinarlo al suo scopo.

In questo capitolo vengono illustrati i vari stati che compongono la logica di gioco, necessari a far svolgere al drone autonomo diverse azioni che gli permettono di volare autonomamente e di rispettare le regole del gioco. Gli stati che formano l'interna applicazione sono:

- **Controllore di quota:** stato necessario per il decollo, l'atterraggio, il mantenimento e i cambi di quota.
- **Gestione ostacoli:** stato necessario per evitare prontamente gli eventuali ostacoli.
- **Ricerca dell'avversario:** stato necessario per esplorare l'ambiente di gioco alla ricerca del drone telecomandato.

- **Inseguimento basato sul colore:** stato necessario per l'individuazione e l'inseguimento dell'altro drone su lunghe distanze.

- **Inseguimento basato su infrarossi:** stato necessario per l'individuazione e l'inseguimento dell'altro drone su brevi distanze.

- **Ricarica:** stato in cui viene simulata la ricarica di energia del drone che da la possibilità al giocatore di vincere il gioco.

- **Contromossa:** stato necessario alla stabilizzazione del drone.

6.1 Controllore di quota

Un aspetto molto importante è relativo alla gestione della quota di volo, dato che abbiamo deciso che il drone autonomo deve mantenere sempre una determinata altezza, il più possibile costante.
Il gestore dell'altezza risulta quindi essere sempre attivo in tutte le fasi di volo e lavora ad una frequenza di 10 Hz. Il suo compito è quello di portare il drone a un valore calcolato il più velocemente possibile e in modo tale che sia stabile.
Infatti, mentre gli altri stati si occupano di stabilire a che altezza il drone deve volare, questo componente software si occupa di attuare la decisione presa, gestendo il canale di throttle del quadricottero.
Questa parte utilizza il sonar per stabilire la propria altezza, la quale è stata filtrata per eliminare possibili errori di lettura. Un valore letto non può avere una differenza maggiore di 40 cm da quello letto precedentemente altrimenti viene scartato mantenendo l'ultimo valore valido. Poiché la lettura del sonar avviene ad una frequenza di 10 Hz, una differenza di quota di 40 cm tra due letture consecutive corrisponde ad una velocità verticale di 4 m/s. Questa velocità non è realistica all'interno del gioco e pertanto rappresenta un errore.
Tuttavia, se tale errore persiste per 10 valori consecutivi, l'ultimo valore letto diventa valido. Questo perché una variazione di lettura per più di un secondo consecutivo rappresenta un dislivello nel terreno. Questa possibilità non dovrebbe mai verificarsi nell'ambiente di gioco designato ma è stata comunque gestita per sviluppi futuri.
Per effettuare il controllo di quota è stato implementato un controllore di tipo PID.

6.1.1 Controllore PID

Il controllore PID è un sistema di controllo con retroazione negativa usato ampiamente nel settore industriale. Come tutti i sistemi a retroazione negativa, questo controllore è in grado di rendere stabile un sistema che altrimenti non lo sarebbe. Il motivo principale della sua diffusione è la sua robustezza, che gli permette di operare in diverse condizioni in maniera efficace. È inoltre piuttosto semplice dal punto di vista funzionale.

La sigla PID si riferisce alle azioni Proporzionale-Integrale-Derivativa che intervengono in modo tale da portare l'errore tra il valore attuale di una grandezza osservata (variabile) e il valore desiderato (setpoint) vicino allo 0.

L'azione proporzionale è basata sulla differenza tra setpoint e variabile che prende il nome di errore (e). Il guadagno K_p è, come dice il nome stesso, proporzionale all'errore.

$$u_P = K_P * e$$

L'azione proporzionale serve a velocizzare la risposta del sistema, tuttavia se troppo alta, può causare oscillazioni sempre più ampie che causano instabilità.

L'azione integrale dipende da tutti gli errori commessi nel corso del tempo. Il guadagno K_I tiene perciò conto di tutti gli errori, anche infinitesimali, calcolati fino a quel momento.

$$u_I = K_I * \int_0^t e(\tau)d\tau$$

L'azione integrale non aumenta quando l'errore è nullo, tuttavia può essere diversa da 0 a causa degli errori precedenti. Se il guadagno è eccessivo può causare problemi d'instabilità (fenomeno di windup).

L'azione derivativa si basa sulla velocità con cui cambia l'errore. Il guadagno K_D ha la funzione di prevedere quindi l'andamento dell'errore, limitandone l'effetto.

$$u_D = K_D * \frac{de(t)}{dt}$$

L'azione derivativa è per la maggior parte delle volte non necessaria, poiché serve solo a migliorare la prestazione del sistema di controllo. Il suo utilizzo fornisce una grande velocità di risposta. Anche in questo caso, se il guadagno è elevato, può causare instabilità nel sistema. Inoltre è necessaria cautela se il setpoint dovesse cambiare in maniera rapida e con valori elevati, in quanto potrebbe far crescere l'azione derivativa in maniera eccessiva.

Come mostrato in Figura 6.1 i valori calcolati vengono successivamente sommati tra loro contribuendo a far convergere l'errore a 0.

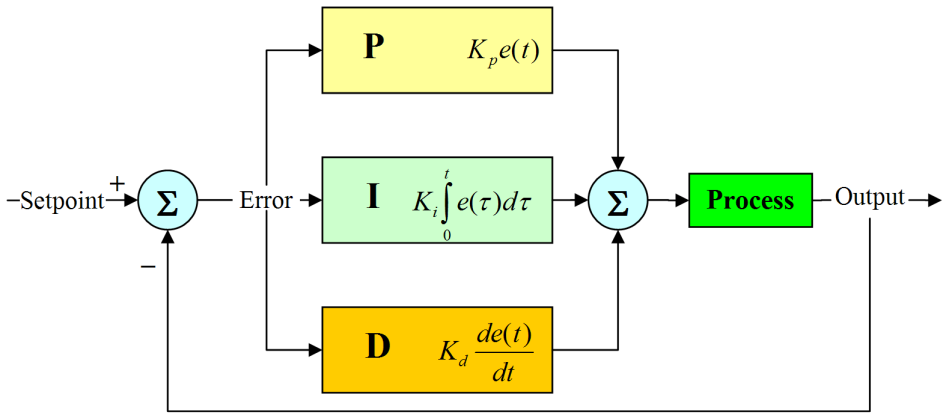


Figura 6.1: Controllore PID

I guadagni K_P, K_I e K_D vengono scelti per ottenere un sistema stabile al setpoint.
Esistono metodi di calcolo che permettono di individuare questi valori, ma necessitano una modellizzazione del sistema.
Tuttavia, per quanto riguarda il nostro lavoro, una modellizzazione del sistema sarebbe risultata troppo complessa per i fini imposti.

Abbiamo perciò trovato sperimentalmente questi valori, che hanno fornito un risultato più che soddisfacente.

L'algoritmo che implementa il PID è stato modificato in modo tale da limitare l'azione integrale e quella derivativa (Algoritmo 1).
L'azione integrale è stata limitata per impedire un eccessivo aumento dell'accumulo di errori, che potrebbe portare a problemi d'instabilità.
L'azione derivativa invece è stata limitata a causa dei possibili errori di lettura dei sensori, che possono far aumentare enormemente il suo valore.
I valori calcolati delle azioni proporzionale, integrale e derivativa necessarie a eliminare l'errore dal setpoint, vengono poi sommati ad un valore base. Esso rappresenta il valore centrale da fornire al processo per ottenere lo scopo desiderato.

Algoritmo 1 Controllore PID

Require: $setpoint$
  $errore = setpoint - misura$;
  $derivata = (errore - errorePrecedente)/dt$;
  $integrale+ = (errore) * dt$;
  if $(integrale > limite\_integrale)$ **then**
    $integrale = limite\_integrale$;
  end if
  if $(integrale < -limite\_integrale)$ **then**
    $integrale = -limite\_integrale$;
  end if
  $p\_calc = P * errore$;
  $i\_calc = I * integrale$;
  $d\_calc = D * derivata$;
  if $(d\_calc > d\_limite)$ **then**
    $d\_calc = d\_limite$;
  end if
  if $(d\_calc < -d\_limite)$ **then**
    $d\_calc = -d\_limite$;
  end if
  $errorePrecedente = errore$;
  $pidvalue = p\_calc + i\_calc + d\_calc + valoreBase$;

6.1.2 Decollo

La fase di decollo è la prima fase del gioco, necessaria al drone autonomo per mettersi in moto e raggiungere la quota base di 150 cm.
La prima cosa ad essere eseguita durante questa fase, è la calibrazione degli accelerometri e l'armamento della scheda di volo.
Il PID per il controllo quota in caso di decollo è stato implementato in modo tale da eliminare il limite massimo del valore integrale necessario durante il mantenimento della quota. Infatti, soprattutto a batteria scarica, il drone

autonomo non riesce a raggiungere la quota base avendo un margine ridotto di potenza.
Per eliminare questo problema viene calcolato l'andamento del drone a partire dai valori letti dal sonar. In fase di decollo l'andamento calcolato deve indicare un aumento di quota. Quando questo non avviene, significa che il margine di potenza necessario a salire non è sufficiente e pertanto viene incrementato. L'incremento avviene assegnando al valore base la somma tra l'ultima uscita del PID calcolato e il vecchio valore base. Inoltre, l'azione integrale viene resettata.

6.1.3 Atterraggio

L'atterraggio, così come il decollo, è una fase che utilizza il PID per il controllo di quota in modo tale da far giungere a terra senza danni il drone autonomo.
Anche in questo caso, il PID viene gestito diversamente, cambiando gradualmente il setpoint. Si passa dall'attuale altezza di volo fino a un altezza minima dove è possibile spegnere i motori senza causare danni al velivolo. Ogni 400 ms si controlla che il nuovo setpoint sia stato raggiunto correttamente e, in caso positivo, la quota di volo viene abbassata di 30 cm.
Il tempo e il valore di abbassamento della quota sono stati studiati appositamente in modo tale da far lavorare in condizioni ottimali il controllore sviluppato. In questo tempo infatti, il drone riesce a portarsi alla nuova quota di volo senza oscillazioni che comprometterebbero il funzionamento dell'atterraggio.
Se il setpoint variasse più velocemente, si rischierebbe una caduta troppo violenta, con possibilità di pericolosi rimbalzi a terra.
Terminato l'atterraggio, per motivi di sicurezza, viene disarmata la scheda di volo.

6.2 Gestione ostacoli

La gestione degli ostacoli è una fase molto importante per quanto riguarda la sicurezza del drone e dell'ambiente nel quale si svolge il gioco.
Essa viene attuata a partire dai dati forniti dai quattro sensori di prossimità presenti sul drone autonomo. I sensori sono posti alle estremità degli assi di movimento di pitch e roll ed è appunto su questi due canali che lavora la gestione ostacoli.
L'idea di base è quella di mantenere sempre una distanza di almeno 150 cm da qualunque oggetto rilevato. Questo valore è stato scelto poiché è il

massimo rilevabile dai sensori utilizzati e, allo stesso tempo, fornisce una buona distanza di sicurezza.
In ogni momento del volo viene controllato lo stato dei sensori e se almeno uno rileva un ostacolo al di sotto dei 150 cm, viene attivata immediatamente questa parte di gestione ostacoli. La fase in questione ha infatti priorità maggiore rispetto a tutte quelle di gioco tranne la ricarica, in cui il drone è fermo a terra.

6.2.1 Casi di rilevazione

Avendo a disposizione quattro sensori, ci sono 15 diverse possibilità di lettura.
Il caso in cui tutti i sensori rilevano un oggetto non è stato implementato poiché non è realistico all'interno del nostro gioco.
I casi trattati sono schematizzati in Figura 6.2.

Le azioni intraprese dal drone sono suddivise in base al numero di rilevazioni contemporanee:

- **1 ostacolo** (casi 1-2-3-4): il drone autonomo si sposta lungo la stessa direzione dell'ostacolo rilevato, ma nel verso opposto.
- **2 ostacoli su assi diversi** (casi 5-6-7-8): il drone si comporta come nel caso di un singolo ostacolo. Essendo infatti i canali di pitch e roll indipendenti, il drone si muove contemporaneamente nelle due direzioni seguendo il verso opposto agli ostacoli rilevati.
- **2 ostacoli sullo stesso asse** (casi 9-10): non potendo muoversi nella stessa direzione degli ostacoli poiché è presente un altro oggetto, il drone si muove nell'altra direzione, seguendo un verso scelto casualmente.
- **3 ostacoli** (casi 11-12-13-14): il drone si muove nell'unico verso libero da ostacoli. Procedendo da quella parte infatti, esso si libera dell'ostacolo nel verso opposto e successivamente si libererà dei due rimasti rientrando nella gestione dei casi a 2 ostacoli sullo stesso asse.

La gestione degli ostacoli è stata studiata appositamente per l'ambiente in cui si svolge il gioco.

6.2.2 Implementazione

Nei casi in cui il drone deve allontanarsi da un ostacolo percorrendo la sua stessa direzione, il controllore che implementa le azioni descritte, è di tipo

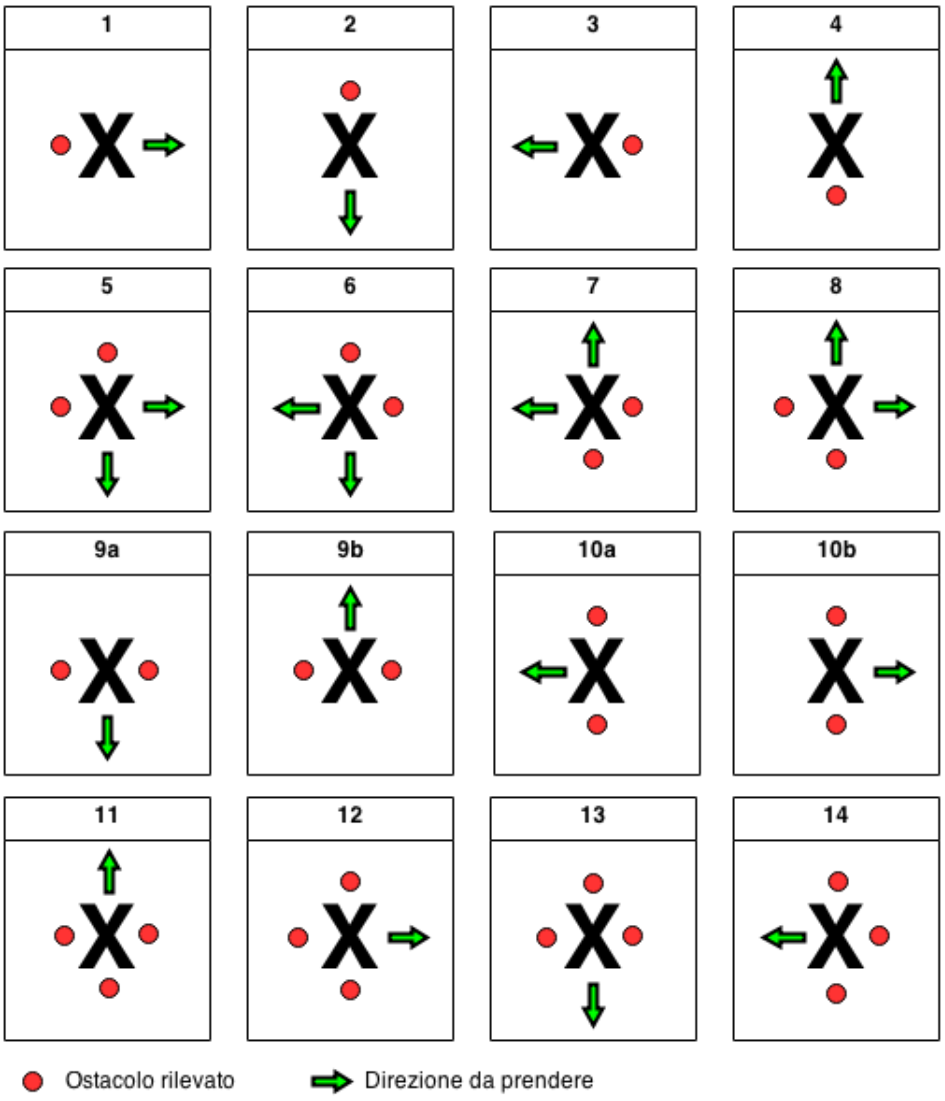


Figura 6.2: Diagramma della gestione degli ostacoli

proporzionale-integrale. Viene infatti fornita al drone una forza proporzionale alla distanza dall'ostacolo rilevato, ma di verso opposto, in modo tale da allontanarsi da esso.
Se la forza fornita non è sufficiente a muoverlo nel verso desiderato, verrà incrementata nelle successive letture dall'azione integrale.
Nel caso in cui si procede lungo l'altra direzione viene invece fornita semplicemente una forza costante.
L'Algoritmo 2 descrive come è stata implementata questa fase.

Algoritmo 2 Gestione ostacoli

```
Require: ostacolo, distanzaOstacolo
  morePitch = 0;
  moreRoll = 0;
  if (ostacolo(destra)&&ostacolo(sinistra)) then
     if (ostacolo(davanti)||ostacolo(dietro)) then
        morePitch = forza(distanzaOstacolo);
     else
        morePitch = forzaCostante;
     end if
  else if (ostacolo(davanti)&&ostacolo(dietro)) then
     if (ostacolo(destra)||ostacolo(sinistra)) then
        moreRoll = forza(distanzaOstacolo);
     else
        moreRoll = forzaCostante;
     end if
  else
     if (ostacolo(davanti)||ostacolo(dietro)) then
        morePitch = forza(distanzaOstacolo);
     else
        moreRoll = forza(distanzaOstacolo);
     end if
  end if
  if (rollOld < moreRoll) then
     moreRoll+ = (moreRoll − rollOld) ∗ K_I;
  end if
  if (pitchOld < morePitch) then
     morePitch+ = (morePitch − pitchOld) ∗ K_I;
  end if
  roll = rollBase ± moreRoll;
  pitch = pitchBase ± morePitch;
  rollOld = moreRoll;
  pitchOld = morePitch;
```

6.3 Ricerca dell'avversario

La ricerca è la prima fase ad essere eseguita dopo il decollo del drone autonomo. Il suo scopo è quello di muoversi all'interno di un ambiente non conosciuto al fine di individuare il drone telecomandato.
Questa esplorazione utilizza solo i canali di yaw e di pitch.
Lo yaw viene utilizzato poiché permette di vedere il più velocemente possibile tutto l'ambiente circostante. Il pitch serve per muoversi nella direzione in cui è posizionata la webcam, in modo tale da diminuire l'eventuale distanza tra i due droni.
In questa fase, il movimento sull'asse di roll non fornisce un'informazione aggiuntiva per la ricerca dell'altro drone e pertanto non viene utilizzato.
La strategia di ricerca è molto semplice, ma ottimale per quanto riguarda l'ambiente in cui si trova: il drone autonomo compie una rotazione attorno al proprio asse e, una volta effettuata, procede in avanti per un determinato

tempo per poi ricominciare.
Per quanto riguarda la rotazione, viene dato un valore costante di yaw fino a quando il drone non compie un angolo giro più un angolo casuale tra 0° e 100°. Con l'angolo giro viene visualizzato in un tempo molto limitato tutto l'ambiente circostante. Il successivo angolo serve, in combinazione con il pitch, a far avanzare il drone in una direzione casuale.
Per verificare la corretta rotazione si fa uso del magnetometro presente sulla scheda di volo.
La rotazione attorno al proprio asse avviene a scatti di 30°, poiché un suo utilizzo troppo prolungato causa instabilità del mezzo. La direzione della rotazione è casuale.
Yaw e pitch non vengono dunque utilizzati contemporaneamente.
Se in qualsiasi momento viene rilevato l'altro drone, questa fase si conclude, passando ad un altro stato specifico. L'Algoritmo 3 descrive questa fase di gioco.

Algoritmo 3 Ricerca

```
angoloCompiutoTotale = 0;
angoloCompiutoScatti = 0;
angoloCasuale = random()%100;
if (angoloCompiutoTotale < 360 + angoloCasuale) then
    angoloCompiuto = calcolaAngolo();
    controlloAngoloScatti+ = angoloCompiuto;
    angoloCompiutoTotale+ = angoloCompiuto;
    if (controlloAngoloScatti < 30) then
        yaw = yawBase ± forzaYaw;
    else
        controlloAngoloScatti = 0;
    end if
else
    pitch = pitchBase + forzaPitchAvanzamento;
end if
```

6.4 Inseguimento basato sul colore

Questa fase del gioco avviene quando il drone autonomo individua quello telecomandato tramite il riconoscimento del colore della sua scocca.
L'obiettivo di questo stato è quello di avvicinarsi al drone telecomandato fino al momento in cui non vengono individuati i led infrarossi posti sotto di esso.
Nello specifico, per rappresentare i colori abbiamo considerato due modelli: RGB e HSV.
Il modello RGB, Red Blue Green, utilizza l'intensità luminosa dei colori primari, per l'appunto rosso, blu e verde.

Combinando questi colori è possibile infatti individuare tutti gli altri. L'assenza o la presenza di tutti e 3 i valori forma rispettivamente il nero e il bianco (Figura 6.3). L'intensità di ogni colore primario varia da 0 a 255.

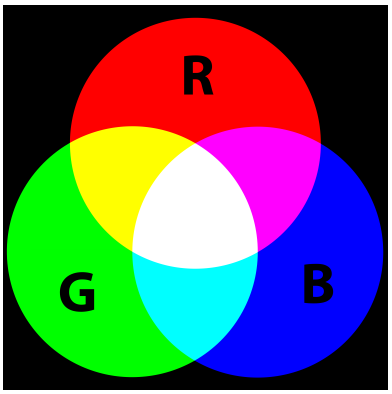


Figura 6.3: Funzionamento tramite RGB

Il modello HSV, dall'inglese Hue Saturation Value (tonalità, saturazione e valore), definisce una rappresentazione in coordinate cilindriche del colore (Figura 6.4).

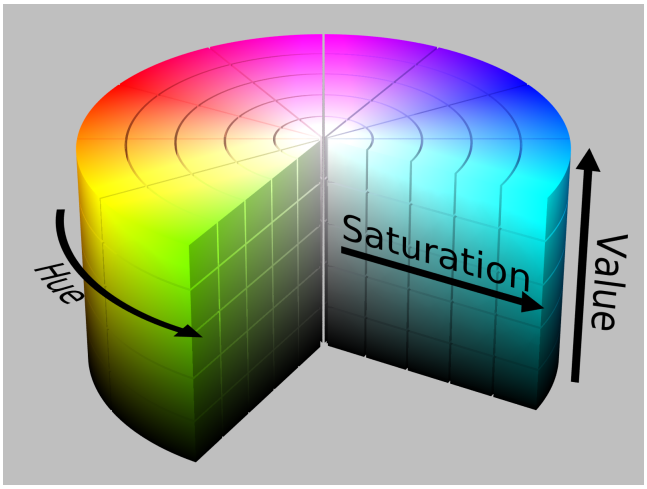


Figura 6.4: Modello cilindrico HSV

Con il valore di Hue è possibile muoversi sull'angolo della sezione circolare del cilindro. I colori primari si trovano rispettivamente a 0° con il rosso, 120° con il verde e 240° con il blu.

La Saturation rappresenta la distanza dall'asse del cilindro che vale 0 nel suo centro e 1 in corrispondenza della superficie. Questo valore indica l'intensità e la purezza del colore che con 0 risulta essere neutrale, mentre con 1 appare puro.
Infine, Value indica la luminosità del colore, e corrisponde all'altezza del cilindro.

Nell'ambito della nostra tesi abbiamo utilizzato il modello HSV che, con la sua proprietà di separare l'intensità dal colore, fornisce un sistema robusto al cambio di luce.
Il riconoscimento del colore avviene utilizzando la webcam, in particolare l'immagine catturata viene analizzata tramite la libreria OpenCV [8].
OpenCV è un efficace framework per la gestione dei dati provenienti da un dispositivo video.
Questa libreria è stata utilizzata per filtrare, tramite valori HSV, il colore rosso dall'immagine video catturata (frame). In questo modo è stato possibile isolare il colore del drone telecomandato dal resto dell'immagine.
È stato scelto questo colore perché poco diffuso rispetto ad altri. A causa di come OpenCV ottiene i valori HSV, è stato necessario unire due differenti filtri di colore. In questo modo abbiamo ottenuto un buon intervallo di tonalità, saturazione e valore con cui siamo riusciti a individuare la scocca del drone telecomandato anche sotto diverse condizioni di luce. Per semplificare il tracciamento del colore rosso è stata utilizzata la libreria cvBlob[5] che fornisce facili strumenti per l'analisi della forma di colore riconosciuta.
Per eliminare il più possibile falsi sono stati infatti utilizzati due filtri: uno sulla forma e uno sull'area.
Il filtro sulla forma serve a determinare solo gli oggetti rettangolari, corrispondenti alla scocca del drone telecomandato.
Il filtro dell'area serve invece a determinare la sua grandezza, limitando ulteriormente i falsi.
Grazie a questa analisi delle immagini riusciamo ad ottenere tutti i rettangoli rossi che si trovano dai 4 ai 12 metri di distanza. Abbiamo filtrato questi rettangoli al di sotto dei 4 metri perché sotto questa distanza interviene l'inseguimento basato su infrarossi.
Poiché il drone telecomandato è uno solo, se sono presenti più rettangoli con quelle caratteristiche (blob), viene scelto il primo trovato. Questo perché è impossibile fare ulteriori discriminazioni che sprecherebbero solo tempo e risorse.
Grazie alla libreria cvBlob è possibile estrarre molto semplicemente il centro del blob trovato.

6.4.1 Risposta del drone

Il drone autonomo, una volta ottenuto il centro del miglior blob, ha lo scopo di avvicinarsi ad esso portandolo il più possibile al centro dell'immagine.
Per fare ciò abbiamo diviso il frame in una griglia 5x5. In questo modo è possibile determinare quale direzione indicativa deve prendere il drone. Non è rilevante infatti dove sia precisamente l'avversario, poiché questa fase è utilizzata solo su grandi distanze.
Vengono utilizzati due canali durante questo stato: yaw e pitch.
Il canale yaw ha lo scopo di portare al centro dell'asse x il blob selezionato. A seconda della sua posizione all'interno della griglia, il drone autonomo gira attorno al proprio asse, più o meno velocemente, fino a quando l'avversario non è visto al centro.
Il drone gira velocemente anche nel caso in cui il blob viene perso dopo essere stato individuato sulla prima o sull'ultima colonna della griglia.
Il canale pitch ha lo scopo di avvicinarsi una volta che il drone-avatar risulta al centro dell'immagine. In questa fase, il drone va avanti per un tempo prefissato. Questo avanzamento a scatti è dovuto al fatto che risulta problematico utilizzare i canali di pitch e yaw contemporaneamente per portare al centro il blob trovato, poiché la webcam non è stabilizzata. Infatti, muovendosi in avanti, l'inclinazione della webcam non permette di vedere il blob nella sua corretta posizione. Per quanto riguarda il centro nell'asse y invece, questa fase si occupa di incrementare o diminuire il setpoint del gestore della quota solamente quando il blob si trova nella prima o ultima riga della griglia.
L'Algoritmo 4 descrive quanto spiegato.

Algoritmo 4 Inseguimento basato sul colore

Require: *blob*
  if (*blob*) **then**
    $x = ottieniGriglia\_x(blob)$;
    $y = ottieniGriglia\_y(blob)$;
    if $((y == limiteSuperioreGriglia)||(y == limiteInferioreGriglia))$ **then**
      $PIDh \rightarrow setH(altezzaAttuale \pm cambiamentoAltezza)$;
    end if
    $yaw = yawBase + calcolaForzaYaw(x)$;
    if $(x == centroGriglia)$ **then**
      $pitch = pitchBase + forzaPitchAvanzamento$;
    end if
    $ultima\_x = x$;
  else if $((!blob)\&\&(tempoTrascorsoUltimoBlob() < limiteTempoScomparsaBlob))$ **then**
    if $((ultima\_x == limiteDestroGriglia)||(ultima\_x == limiteSinistroGriglia))$ **then**
      $yaw = yawBase \pm forzaYaw$;
    end if
  end if

6.5 Inseguimento basato su infrarossi

Questa fase avviene quando il drone autonomo individua delle sorgenti infrarosse, che si assume siano i led posti sul drone telecomandato. Scopo della fase è quello di allinearsi perfettamente con il drone-avatar a una determinata distanza.
Viene utilizzata la WiiCam che con la sua velocità riesce a seguire tempestivamente i cambi di direzione del drone-avatar.
Per fare questo sono stati implementati 2 controllori PID: uno per quanto riguarda la distanza che utilizza il canale di pitch e l'altro per portarsi in centro utilizzando il canale di roll.
La distanza reale tra drone autonomo e telecomandato viene calcolata grazie alla lunghezza focale della WiiCam. Dati la distanza reale (D) tra le due fonti infrarosse poste sulla scocca, nota e fissa (20 cm), e la loro distanza (d) calcolata dalla WiiCam a vari intervalli (Z), si può risalire alla lunghezza focale (f).

$$f = d * \frac{Z}{D}$$

Varie sperimentazioni hanno portato a una lunghezza focale pari a 1350 la cui unità di misura è la stessa del sistema di riferimento della WiiCam utilizzata.
Con questa grandezza è possibile ricavare la distanza reale (Z') tra i due droni avendo come unico dato la distanza ottenuta dalla WiiCam tra le due fonti infrarossi (d').

$$Z' = D * \frac{f}{d'}$$

La distanza calcolata viene successivamente utilizzata dal controllore PID che opera sul pitch, con lo scopo di mantenere il drone autonomo a 250 cm da quello telecomandato.

L'allineamento orizzontale tra i due droni avviene in modo tale da portare

il punto medio delle due fonti infrarosse in corrispondenza del fascio laser alla distanza di 250 cm.

Anche questo processo avviene utilizzando un controllore PID che però agisce sul canale di roll.

L'allineamento verticale viene elaborato utilizzando l'altezza relativa (H) tra drone autonomo e telecomandato.

L'altezza relativa (H) viene calcolata utilizzando la lunghezza focale (f), la distanza Z', la coordinata verticale obiettivo ($\hat{h}$) e quella attuale (h).

$$H = z' * \frac{(\hat{h} - h)}{f}$$

La coordinata obiettivo $\hat{h}$ rappresenta la coordinata relativa all'asse verticale in cui il drone colpisce il centro della scocca avversaria a 250 cm di distanza nel sistema di riferimento della WiiCam. h è invece il valore medio delle coordinate relative all'asse verticale delle sorgenti infrarosse individuate dalla WiiCam.

Al controllore di quota verrà quindi settata una nuova altezza pari a quella attuale rilevata dal sonar sommata all'altezza relativa (H) calcolata.

Per far funzionare entrambi i PID sono necessarie esattamente due fonti infrarosse, altrimenti la lettura non viene considerata valida e si rientra in uno degli altri stati del gioco. Con una sola fonte infrarossa non è possibile stabilire la distanza alla quale si trova il drone e nemmeno la sua altezza relativa.

Il caso in cui si rilevano tre o quattro fonti viene interpretato come un errore poiché non è possibile vederle contemporaneamente sul drone avversario.

L'Algoritmo 5 mostra il funzionamento sopra descritto.

6.5.1 Fuoco

Questa fase viene attivata solamente quando il drone autonomo si trova alla distanza corretta da quello telecomandato. Portatosi ai giusti valori grazie all'inseguimento con infrarossi, il drone deve attivare il laser di bordo in modo tale da colpire la scocca dell'avversario.

Per attivare il laser, il drone avversario deve essere ad una distanza compresa tra 220 e 280 cm da quello autonomo ed allineato con esso. In questo modo il laser ha una buona probabilità di colpire la scocca del drone-avatar.

Algoritmo 5 Inseguimento basato su infrarossi

Require: wii
  if (wii) **then**
    $x = ottienixmedia(wii)$;
    $y = ottieniymedia(wii)$;
    $d = trovaDistanza(wii)$;
    $h = trovaAltezzaRelativa(y)$;
    $PIDroll \rightarrow setxAttuale(x)$;
    $PIDpitch \rightarrow setdAttuale(d)$;
    $PIDh \rightarrow setH(altezzaAttuale + h)$;
    $pitch = PIDpitch \rightarrow calcola() + pitchBase$;
    $roll = PIDroll \rightarrow calcola() + rollBase$;
    $gestioneFuoco(x, y, d)$;
  end if

Per stabilire se il drone autonomo abbia colpito o meno l'avversario, viene analizzata l'immagine proveniente dalla webcam durante il periodo di attivazione del laser, ossia 1 secondo. L'immagine viene ritagliata in corrispondenza di dove si trova il drone-avatar in modo tale da analizzare solo ed esclusivamente la parte della scocca.
Il laser che riflette sul cartoncino rosso crea un cerchio, pertanto viene analizzata l'immagine alla ricerca di cerchi. Per distinguerli meglio all'interno dell'immagine, questa viene portata in scala di grigi e si applica un filtro di sfocatura.
Se il riscontro del laser è positivo il drone autonomo atterra e segnala la vittoria all'applicazione di gioco esterna.
L'Algoritmo 6 descrive il comportamento spiegato.

Algoritmo 6 Gestione Fuoco

Require: x, y, d
  if $(allineatoADistanza(x, y, d))$ **then**
    $fuocoLaser()$;
    $frame = ritagliaImmagine()$;
    $filtroBlur(frame)$;
    $scalagrigi(frame)$;
    $colpito = ricercaCerchio()$;
    if $colpito$ **then**
      $inviaSconfittaGiocatore()$;
      $fine = true$;
    end if
  end if

6.6 Ricarica

Questa fase di gioco utilizza alcuni stati precedentemente descritti in maniera diversa per creare una nuova dinamica all'interno del gioco.

Il drone autonomo, dopo un tempo casuale tra i 2 e i 3 minuti di gioco, atterra e disarma la scheda di volo.
A questo punto viene attivato il laser di segnalazione del drone che indica la zona di atterraggio in cui il drone telecomandato deve atterrare il più presto possibile.
Per controllare la corretta posizione in cui il drone telecomandato atterra, viene utilizzata ancora una volta la WiiCam.
In questo modo è stato possibile calcolare, in maniera analoga a quanto descritto precedentemente, la distanza del drone e il suo allineamento con esso. È stato fornito un intervallo di tolleranza in cui è possibile individuarlo. In particolare la distanza deve essere compresa tra 120 e 170 cm e i droni devono essere allineati. Se il drone telecomandato rispetta questi vincoli, il drone autonomo invia all'applicazione di gioco un segnale che determina la vittoria del giocatore.
Dopo 30 secondi dall'avvenuto atterraggio, il drone autonomo, se non ha riconosciuto il drone nemico nella zona delimitata, decolla nuovamente ricominciando la sua ricerca.
L'Algoritmo 7 spiega tale procedimento.

Algoritmo 7 Ricarica

```
Require: tempoRicarica
  if (tempoDiGioco == tempoRicarica) then
    atterra();
    if (wii) then
      x = ottienixmedia(wii);
      y = ottieniymedia(wii);
      d = trovaDistanza(wii);
      if (atterraggioCorretto(x, y, d)) then
        inviaVittoriaGiocatore();
        fine = true;
      end if
    end if
  end if
  if ((tempoDiGioco == tempoRicarica + tempoADisposizioneAtterraggio) then
    decolla();
  end if
```

6.7 Contromossa

La contromossa non è un vero e proprio stato di gioco. Il suo unico scopo è infatti quello di stabilizzare il più possibile il drone autonomo, permettendogli di frenare.
Senza eseguire una contromossa, ossia una mossa di verso opposto a quello

precedente, il drone continuerebbe a muoversi per molto più tempo del voluto in tale verso.
Il suo movimento infatti non può terminare fino a quando l'attrito dell'aria non contrasta la forza applicata fino a quel momento.
Questo stato è dunque indispensabile per una corretta esecuzione del gioco.
La contromossa viene evocata alla fine di ogni azione sugli assi di pitch e roll.
Il canale di yaw non necessità di contromosse poiché non muove il drone ma lo ruota e basta.
La contromossa deve tenere conto della forza e del tempo per cui è stata eseguita. In questo modo è possibile contrastare il lavoro eseguito e fermare dunque il drone.
In particolare abbiamo trovato una corrispondenza numerica tra la massima forza data e la forza della frenata. Allo stesso modo esiste una corrispondenza tra la durata della forza applicata e quella necessaria per la frenata.

6.8 Funzionamento globale

Tutti questi stati vengono attivati secondo una certa priorità, in modo tale da attivare solo quello più utile ai fini del gioco e della sicurezza.
Tralasciando il controllore di quota, che, come detto precedentemente, è sempre attivo in tutte le fasi di gioco, e la contromossa, necessaria per stabilizzare il drone autonomo, è stato dato il seguente ordine di priorità:

1. Ricarica
2. Gestione ostacoli
3. Inseguimento basato su infrarossi
4. Inseguimento basato sul colore
5. Ricerca dell'avversario

Il funzionamento è facilmente schematizzabile con una macchina a stati, riportata in Figura 6.5.

L'Algoritmo 8 riporta il funzionamento globale della logica di gioco.

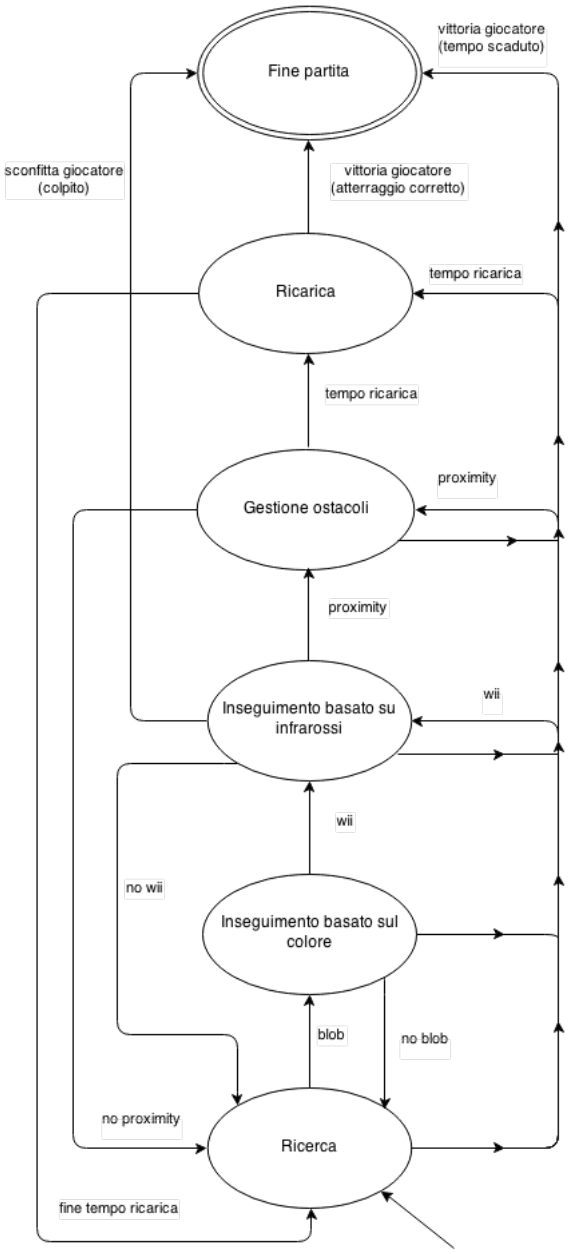


Figura 6.5: Macchina a stati

Algoritmo 8 Funzionamento globale

```
loop
  if (comandoAutomatico()) then
    ottieniValoriSensori();
    if (primoVoltaComandoAutomatico()) then
      decolla();
      tempoRicarica = rand()%60 + 120;
      inizioTempoDiGioco();
    end if
    if (tempoDiGioco == tempoRicarica) then
      ricarica();
    else if (proximity) then
      gestioneOstacoli();
    else if (wii) then
      inseguimentoBasatoSuInfrarossi();
    else if (blob) then
      inseguimentoBasatoSulColore();
    else
      ricerca();
    end if
    throttle = PIDh → calcola() + throttleBase;
    if (tempoDiGioco == tempoTotaleDiGioco) then
      fine = true;
    end if
    if (fine == true) then
      atterra();
    end if
  end if
end loop
```

Capitolo 7

Prove sperimentali

> *"Il principio cardine della scienza, quasi la sua definizione, è che la verifica di tutta la conoscenza è l'esperimento. L'esperimento è il solo giudice della «verità» scientifica."*
>
> Richard Feynman

Data la gestione del gioco in stati indipendenti tra loro, è stato possibile testare separatamente ogni singola fase di gioco. In questo modo abbiamo cercato un risultato ottimale per ogni fase isolata, per poi testare l'insieme nella prova finale di gioco.
Per questo motivo, il seguente capitolo mantiene la stessa struttura del precedente.

7.1 Controllore di quota

Il controllore di quota, nella fattispecie il controllore PID che regola l'altezza, è stato progettato per dare una risposta rapida e precisa ai cambiamenti di quota.
I test seguenti puntano a verificare tali aspetti.
Per prima cosa abbiamo testato il mantenimento di una quota fissata. In Figura 7.1 è possibile notare che il drone mantiene il setpoint in maniera stabile. Le oscillazioni sono brevi e limitate a pochi cm, un valore del tutto soddisfacente ai fini del gioco.

Sono stati effettuati test anche per quanto riguarda i cambiamenti di altezza che il drone affronta per le altre varie fasi di gioco.
In particolare per testare questi cambiamenti di quota è stata utilizzata l'applicazione esterna per l'invio di comandi di cambiamento d'altezza.
La Figura 7.2 rappresenta la risposta del controllore PID a un gradino di 20

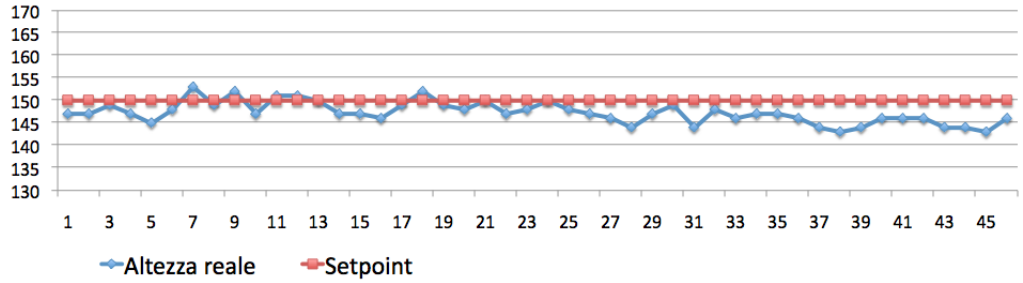


Figura 7.1: Mantenimento di una quota di volo costante

cm in discesa e uno di 30 cm in salita.

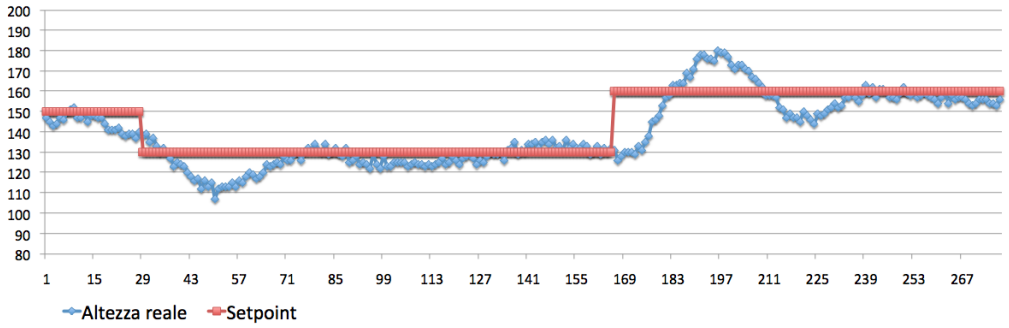


Figura 7.2: Risposta del controllore ad un gradino in discesa e uno in salita. Sull'asse X di questo grafico e dei successivi viene rappresentato il tempo in decimi di secondo.

Come si può vedere, la risposta del drone è rapida e senza oscillazioni eccessive. Cambiamenti di altezza di entità maggiore portano ad oscillazioni maggiori e di conseguenza a un maggior tempo necessario per stabilizzarsi. Tuttavia i cambiamenti all'interno del gioco non possono essere troppo repentini e non differiscono mai più di 50 cm rispetto al valore precedente. In questo modo si riesce a mantenere le oscillazioni a un valore accettabile.
Infine, sono stati testati il decollo e l'atterraggio. Come spiegato nel capitolo precedente essi fanno parte del controllore di quota ma con alcune modifiche.
Le Figure 7.3 e 7.4 mostrano rispettivamente il funzionamento del decollo e dell'atterraggio.

Durante il decollo, il drone raggiunge velocemente la quota prefissata senza eccessive oscillazioni.

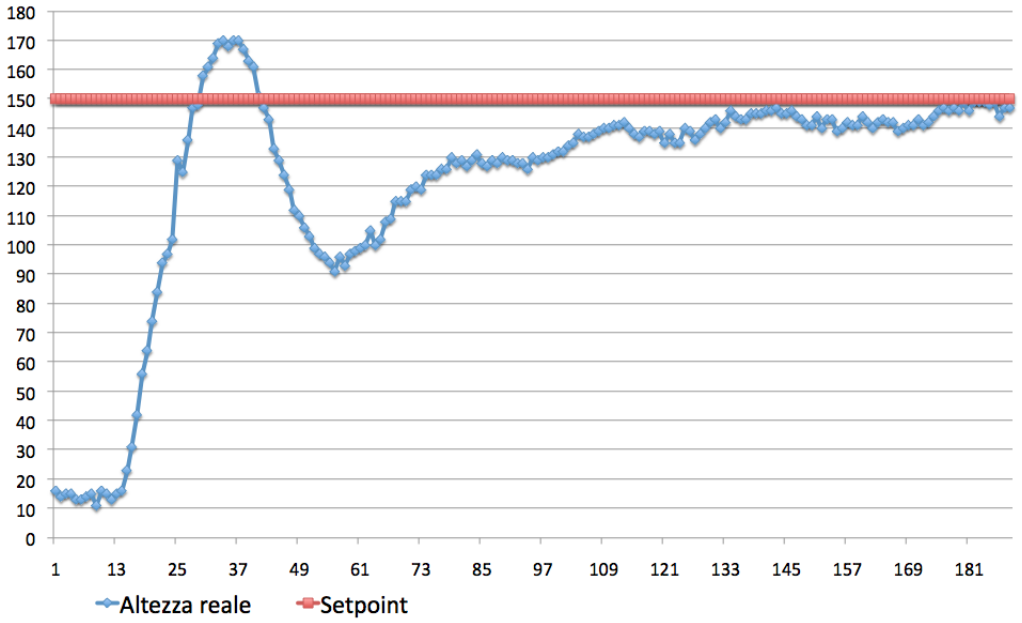


Figura 7.3: Risposta del drone ad un decollo

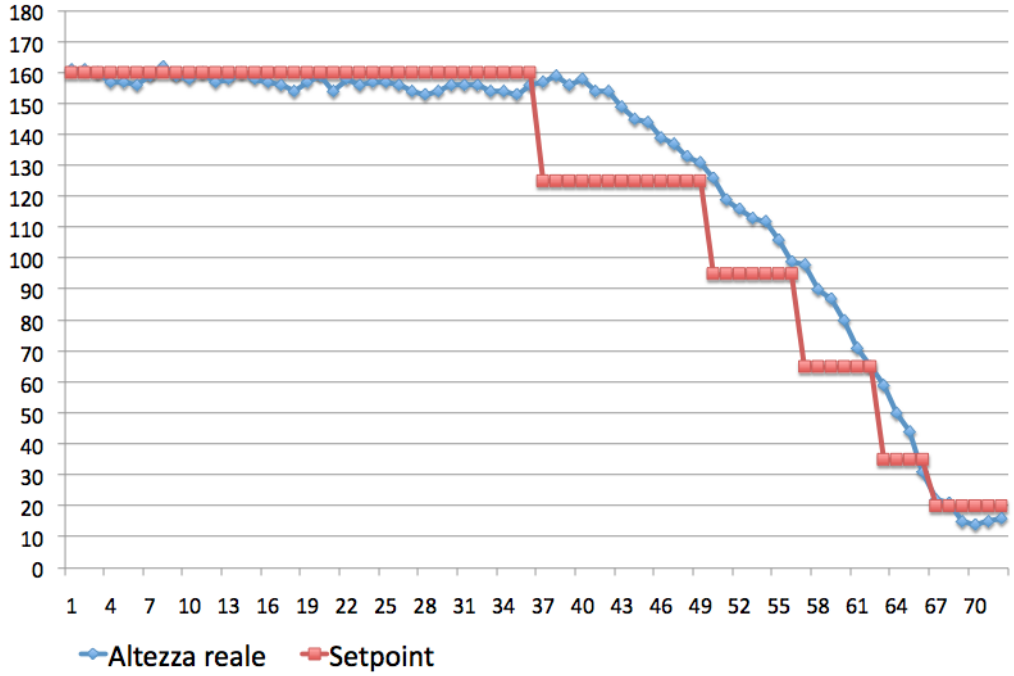


Figura 7.4: Risposta del drone ad un atterraggio

Allo stesso modo la discesa è dolce e non permette pericolosi rimbalzi al suolo.
In Figura 7.5 è riportato il grafico di un decollo con batteria scarica. Come si può vedere, con la stessa potenza con cui prima raggiungeva e superava ampiamente il setpoint, adesso il drone non riesce più a salire.
Non appena il controllore si accorge di questa situazione, viene aumentato il margine di potenza utilizzabile così da permettere il completamento del decollo.

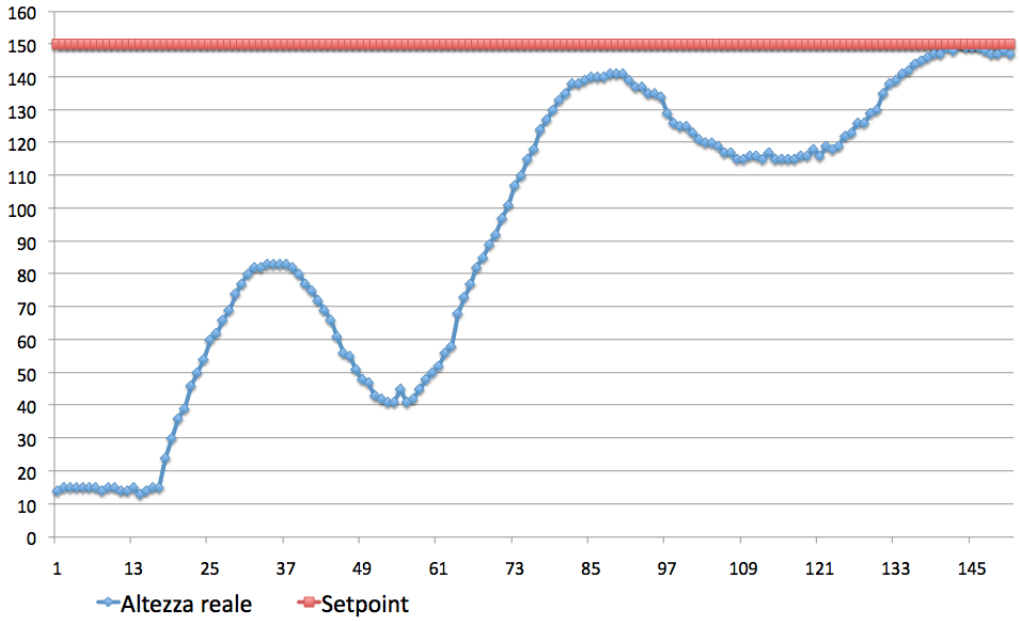


Figura 7.5: Risposta del drone ad un decollo con batteria scarica

7.2 Gestione ostacoli

Come già detto più volte solo 4 sensori di prossimità non sono sufficienti a evitare qualsiasi tipo di ostacolo. Come si evince dalle prove effettuate, il drone autonomo non è in grado di rilevare ostacoli che non siano precisamente disposti sul suo asse di pitch e roll.
Un numero di sensori superiore non era tuttavia gestibile dal punto di vista hardware.
La gestione degli ostacoli è comunque accettabile, considerato il limite di

sensori e l'ambiente utilizzato: il drone che individua un ostacolo deve allontanarsi da esso.
In Figura 7.6 è riportato un grafico che rappresenta la sequenza di distanze lette da un sensore dal momento della prima rilevazione dell'ostacolo al suo allontanamento.

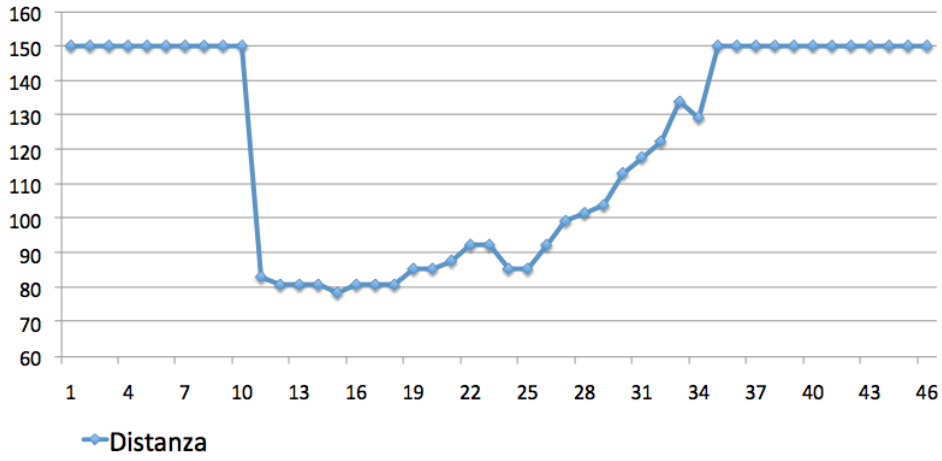


Figura 7.6: Sequenza di letture di un sensore al rilevamento di un ostacolo

È possibile osservare che il drone autonomo riesce ad allontanarsi dall'ostacolo abbastanza velocimente, evitando così l'impatto con esso.
Sull'asse X è sempre riportato il tempo in decimi di secondo.

7.3 Ricerca dell'avversario

La ricerca dell'avversario è stata ottimizzata per funzionare all'interno di un ambiente con delle precise caratteristiche già elencate nei capitoli precedenti.
Non risultano prove in cui il drone-avatar non viene mai individuato all'interno della partita purché questo rispetti le regole comportamentali di sicurezza in particolare quella di non stare sopra o sotto il drone autonomo.

7.4 Inseguimento basato sul colore

I test effettuati per quanto riguarda questa fase mirano a verificare il corretto funzionamento sia dell'individuazione del drone, sia della risposta attuata.
La calibrazione dei valori HSV ha portato a dei risultati che permettono di

individuare correttamente il colore rosso (Figura 7.7) anche in condizioni di luce differenti. Alcuni falsi vengono rilevati soprattutto rispetto al il colore marrone.

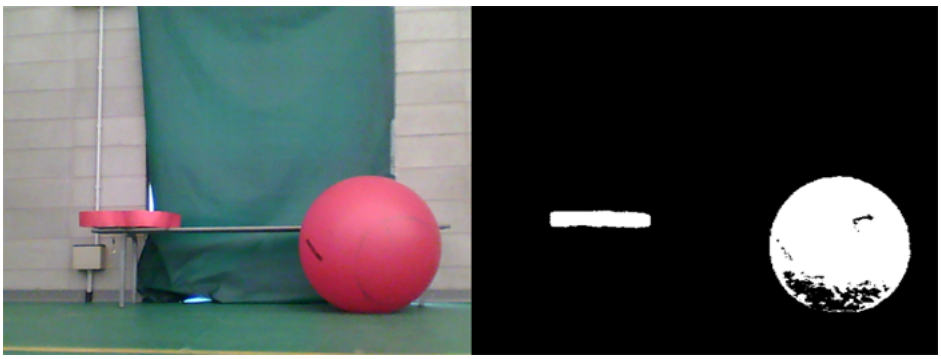


Figura 7.7: Individuazione del rosso: immagine reale e immagine filtrata

Il filtro applicato sulle forme ha permesso di eliminare ulteriori falsi e i test effettuati dimostrano che vengono effettivamente selezionati solo i rettangoli orizzontali rossi.
In Figura 7.8 viene rappresentato come il drone autonomo individua correttamente la scocca del drone telecomandato all'interno di un ambiente con diverse forme color rosso.

Figura 7.8: Individuazione del drone avversario

Esiste comunque la possibilità che il drone veda dei falsi positivi, anche

se le probabilità sono ridotte.
Varie prove hanno dimostrato che il drone riesce a individuare la scocca dell'avversario a partire dai 12 m di distanza (Figura 7.9).

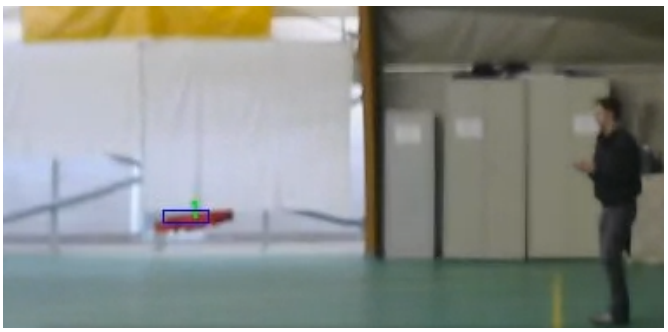


Figura 7.9: Individuazione del drone avversario dalla lunga distanza

La risposta del drone risulta tempestiva ed efficiente riuscendo ad avvicinarsi in brevissimo tempo al drone telecomandato.
Manovre rapide da parte del drone telecomandato, specialmente se ai bordi della visione della webcam del drone autonomo, riescono comunque a far perdere momentaneamente la sua traccia. In questo caso, come spiegato nel capitolo 6, la strategia prevede che il drone ruoti dalla parte in cui lo ha individuato l'ultima volta e, a meno che il giocatore non sia particolarmente abile nel fuggire, può riagganciarlo.

7.5 Inseguimento basato su infrarossi

I test dell'inseguimento basato su infrarossi hanno verificato la prontezza del drone autonomo a seguire quello telecomandato quando questi è vicino.
Si è verificato che la distanza massima in cui è possibile individuare i led infrarossi posti sulla scocca è di circa 5 m.
I controllori PID risultano efficaci per seguire costantemente un giocatore con abilità media. Alcuni problemi nascono quando non vengono rilevate le due fonti infrarossi per via dell'angolazione tra i due droni o semplicemente per la troppa lontananza. In questo caso, il drone non riesce ad allinearsi correttamente con quello telecomandato.

7.5.1 Fuoco

Per quanto riguarda questa fase è stato testato il riconoscimento del marcatore laser sulla scocca del drone telecomandato.

I test dimostrano che il puntatore laser viene identificato correttamente sopra la scocca del drone nella grande maggioranza dei casi. In Figura 7.10 si possono vedere l'immagine reale e l'immagine ritagliata e lavorata per il riconoscimento del laser.

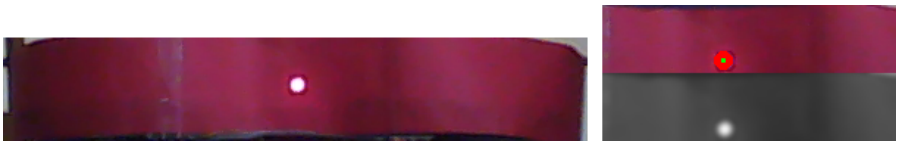


Figura 7.10: Puntatore laser individuato sulla scocca del drone avversario

Esiste però la possibilità di avere falsi positivi, specialmente nel caso in cui il drone autonomo spari colpendo un vicino muro.

In Figura 7.11 si può vedere una fase di gioco in cui il drone autonomo spara mancando l'obiettivo.

Figura 7.11: Fase di gioco

7.6 Ricarica

I test di questo stato servono a verificare che il drone atterri e decolli nei momenti opportuni e che il riconoscimento del giusto atterraggio da parte del drone telecomandato avvenga nella maniera corretta.
In nessun test è risultato un malfunzionamento di questa fase. La Figura 7.12 mostra un tentativo fallito da parte del giocatore di impedire la ricarica. Si noti il puntatore laser che segnala il centro della zona corretta di atterraggio.

Figura 7.12: Tentativo di atterraggio fallito dal giocatore

7.7 Contromossa

La contromossa è stata testata per verificare se effettivamente riesce a controbilanciare una forza applicata.
I vari test eseguiti hanno mostrato che la contromossa serve effettivamente a frenare il drone lungo la direzione dell'ultima mossa eseguita.
Resta comunque insufficiente a fermarlo completamente, in quanto vi sono dei movimenti incontrollabili che non possono essere individuati e quindi controbilanciati dal drone stesso.

7.8 Funzionamento globale

Per quanto riguarda il funzionamento globale sono stati effettuati sia test che verificano il giusto flusso logico per il funzionamento del drone autonomo che test per misurare il divertimento da parte del giocatore.
I test eseguiti sul flusso logico non fanno emergere problemi per quanto riguarda i cambiamenti di stati. Ogni stato si comporta correttamente e viene chiamato in causa al momento opportuno.
L'unico problema presente è che il drone autonomo esegue dei movimenti impossibili da rilevare con i sensori montati a bordo e quindi impossibili da compensare. Sono movimenti minimi, ma che con l'avanzare del tempo fanno acquisire una certa velocità molto difficile da frenare autonomamente. In quasi tutti i test è risultato almeno un intervento da parte dell'operatore specializzato proprio per questo motivo.

Per quanto concerne il divertimento del giocatore si è notata una difficoltà nella vittoria soprattutto per giocatori che provano il gioco le prime volte. Quando il giocatore prende abbastanza dimestichezza con i comandi

del drone telecomandato e quando capisce i movimenti che effettua il drone autonomo per inseguirlo o per cercarlo il gioco diventa più competitivo.
Di particolare interesse è risultata la parte di ricarica, unica fase in cui il giocatore assume un ruolo attivo (e non solo reattivo) all'interno del gioco, che richiede una certa abilità.
Le fasi di gioco sono ben capite dal giocatore soprattutto grazie ai suoni che vengono emessi, non solo quelli dall'applicazione del gioco, ma anche per il rumore stesso del drone autonomo. Viene subito percepita la necessità di scappare da esso.
In generale si può affermare che il gioco diverta, sia vivace e di una complessità tale da non stancare o spazientire un giocatore medio.

Capitolo 8

Conclusioni

"È difficile che la scienza e la tecnologia, nelle loro linee generali, superino la fantascienza. Ma in molti, piccoli e inattesi particolari vi saranno sempre delle sorprese assolute a cui, gli scrittori di fantascienza, non hanno mai pensato."

Isaac Asimov

8.1 Conclusioni

Lo scopo della tesi è stato quello di sviluppare un drone autonomo capace di svolgere un robogame. Il robogame creato consiste in un giocatore umano che, pilotando un drone-avatar, sfida il drone autonomo in un laser game.
La particolarità del drone autonomo consiste nell'avere tutta la potenza di calcolo a bordo. Nessun sensore esterno è necessario per la corretta esecuzione dei comandi di volo.

Lo sviluppo del drone è stato studiato in modo tale da renderlo facilmente estendibile ed utilizzabile per moltissimi altri scopi. Esso è risultato particolarmente robusto e adatto al gioco che ci eravamo prefissati.
Il fatto di avere a bordo tutta la tecnologia necessaria a farlo volare autonomamente lo rende un oggetto molto interessante e utile in tutte situazioni in cui la velocità di reazione è fondamentale. Infatti, poiché la scheda di volo è direttamente collegata all'elaboratore di bordo, non c'è alcun tempo di latenza tra la disponibilità del comando e l'attuazione dello stesso.
Per tutte le operazioni principali di volo è necessaria solamente la presenza di un radiocomando, mentre per il monitoraggio e le applicazioni di gioco è utile una rete Wi-Fi dedicata, comunque non indispensabile per il volo autonomo.

Come emerso dai test effettuati il gioco nel suo complesso risulta divertente, soprattutto perché il giocatore vi è completamente immerso. Per pilotare il drone-avatar infatti, egli deve prestare costantemente attenzione al gioco lasciandosi trasportare dalle sue dinamiche. Maggiore è l'esperienza del giocatore nel pilotaggio e migliore sarà il risultato finale.
Dai test sono risultati alcuni problemi. In particolare non è ancora possibile terminare una partita completa senza l'intervento dell'operatore che controlla il drone autonomo. Questo è dovuto soprattutto ai movimenti involontari del drone che al momento non sono controllabili automaticamente.
L'ambiente di gioco deve rispettare precisamente le specifiche date poiché la presenza di rosso o infrarossi indesiderati può compromettere il funzionamento del gioco. Tale ambiente non è semplice da creare o trovare.
Il costo del gioco è ancora troppo elevato rispetto al prezzo di una console per videogame, termine di paragone per quanto riguarda i robogame. Questo è dovuto soprattutto alla presenza di due agenti robotici e alla tecnologia montata a bordo del drone autonomo.

8.2 Sviluppi futuri

Considerando il lavoro svolto in questa tesi, ci sono alcuni aspetti che possono essere rivisti e migliorati.
Per quanto riguarda il drone, per prima cosa, si potrebbe aggiungere un sensore per l'analisi del flusso ottico. Tale sensore funziona come una camera con una risoluzione molto bassa ed una velocità elevata. In questo modo, montando il sensore vicino al sonar rivolto verso il terreno, è possibile stabilizzare il drone eliminando tutti gli slittamenti e i movimenti involontari che al momento non sono sotto al nostro controllo.
Come detto più volte all'interno della tesi, il numero di sensori di prossimità presenti non sono sufficienti a garantire una sicurezza totale. Si potrebbe quindi pensare di aumentare il loro numero e migliorarne la gestione.
Un ulteriore sviluppo potrebbe consistere nell'alleggerire complessivamente il drone, togliendo ad esempio il dissipatore dell'Odroid. Un drone più leggero risulta essere più manovrabile.
Si potrebbe inoltre pensare di dotare il sistema di una mappa. Al momento infatti, il drone si muove in un ambiente completamente sconosciuto che rimane tale per tutta la partita. Con un sistema di posizionamento, si potrebbe costruire una mappa dell'ambiente a partire dalle immagine analizzate dalla webcam e dai sensori di prossimità. Questo porterebbe ad un notevole miglioramento della strategia di ricerca e della gestione degli osta-

coli all'interno del gioco.

Nel gioco vero e proprio, si potrebbero pensare ulteriori strategie per la ricerca del drone avversario utilizzando sempre tutti e quattro i canali di roll, pitch, yaw e throttle disponibili, che danno una notevole libertà di movimento.
Si potrebbero prevedere alcuni livelli di difficoltà da impostare prima dell'inizio della partita in base all'esperienza del giocatore umano oppure nel corso della stessa, in base alla risposta del giocatore. Le difficoltà si potrebbero basare sull'aggressività della ricerca e sulla precisione dell'inseguimento, che diventa sempre migliore con l'aumentare del livello.
L'ultimo aspetto da considerare potrebbe essere quello di trasformare il gioco singolo in un gioco a squadre con la presenza di più droni-avatar e più droni autonomi che collaborano intelligentemente al fine di creare una vera e propria battaglia laser.

Bibliografia

[1] ARDrone 2.0 Parrot Website. http://ardrone2.parrot.com/.

[2] AR.FlyingRace. http://blog.parrot.com/2011/04/27/ar-flyingace-free-app-available-on-the-app-store/.

[3] AR.Rescue. http://blog.parrot.com/2012/01/04/rescue/.

[4] Code::Blocks Website. http://www.codeblocks.org/.

[5] cvBlob Website. http://code.google.com/p/cvblob/.

[6] Forum Odroid. http://forum.odroid.com/.

[7] Odroid-U2 Website. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G135341370451.

[8] OpenCV Website. http://opencv.org/.

[9] Regolamento ENAC. http://www.enac.gov.it/repository/ContentManagement/information/N122671512/Regolamento_APR_ed.1.pdf.

[10] ROS imagetransport. http://wiki.ros.org/image_transport.

[11] ROS Imu msg. docs.ros.org/api/sensor_msgs/html/msg/Imu.html.

[12] ROS Marker msg. http://docs.ros.org/api/visualization_msgs/html/msg/Marker.html.

[13] ROS Range msg. http://docs.ros.org/api/sensor_msgs/html/msg/Range.html.

[14] ROS String msg. http://docs.ros.org/api/std_msgs/html/msg/String.html.

[15] ROS Website. http://wiki.ros.org/.

[16] Vicon Website. `http://www.vicon.com/`.

[17] D. Calandriello, D. Martinoia, A. Bonarini. Physically interactive robogames: Definition and design guidelines. *Robotics and Autonomous Systems*, 61:739–748, 2013.

[18] D. Mellinger, N. Michael, V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *International Journal of Robotics Research*, 31:664–674, 2012.

[19] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin and B. He. Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface. *Journal of Neural Engineering*, 10, 2013.

[20] P.-J. Bristeau, F. Callou, D. Vissire, N. Petit. The navigation and control technology inside the ar.drone micro uav. *18th IFAC World Congress*, pages 1477–1484, 2011.

Appendice A

Configurazione

A.1 Configurazione Odroid

Sul computer di bordo è stato installato il sistema operativo Ubuntu Linaro versione 12.11.
L'utente principale è *linaro* con password *linaro*.
Di seguito vengono riportati tutti i pacchetti software e le configurazioni necessarie per il funzionamento della nostra applicazione. Viene comunque fornita l'immagine del sistema completamente funzionante.

A.1.1 ROS

La versione ROS utilizzata è quella ros-groovy-ros-base, la quale contiene tutti i pacchetti base necessari per la comunicazione. Il comando da eseguire nel terminale è il seguente:

```
$ sudo apt-get install ros-groovy-ros-base
```

Prima di utilizzare ROS è necessario installare anche rosdep:

```
$ sudo apt-get install python-rosdep
$ sudo rosdep init
$ rosdep update
```

Infine, per avere automaticamente le variabili di ambiente ogni volta che si apre il terminale, va eseguito questo comando:

```
$ echo source /opt/ros/groovy/setup.bash >>~/.bashrc
$ source ~/.bashrc
```

Dopo aver installato ROS si procede creando la cartella di workspace:

```
$ mkdir -p ~/workspace/src
$ cd ~/workspace/src
$ catkin_init_workspace
```

All'interno del workspace è necessario copiare il contenuto della cartella OdroidSW.
Successivamente per compilare il codice si utilizza il seguente comando:

```
$ cd ~/workspace/
$ catkin_make
```

Per quanto riguarda la comunicazione di rete vanno svolti questi comandi:

```
$ echo export ROS_IP=proprio_indirizzo >>~/.bashrc
$ echo export ROS_MASTER_URI=http://proprio_indirizzo:11311
>> /.bashrc
```

A.1.2 OpenCV

Per installare le OpenCV dobbiamo prima di tutto configurare il sistema installando alcune dipendenze:

```
$ sudo apt-get update
$ sudo apt-get -y install build-essential cmake pkg-config
$ sudo apt-get -y install libjpeg62-dev
$ sudo apt-get -y install libtiff4-dev libjasper-dev
$ sudo apt-get -y install libgtk2.0-dev
$ sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
$ sudo apt-get -y install libdc1394-22-dev
$ sudo apt-get -y install libxine-dev libgstreamer0.10-dev libgstreamer-
plugins-base0.10-dev
$ sudo apt-get -y install python-dev python-numpy
$ sudo apt-get -y install libtbb-dev
$ sudo apt-get -y install libqt4-dev
```

Si passa poi al download vero e proprio della versione di OpenCV voluta (nel nostro caso la versione 2.4.4).

```
$ mkdir opencv
$ cd opencv
$ wget http://sourceforge.net/projects/opencvlibrary/files/
opencv-unix/2.4.4/OpenCV-2.4.4a.tar.bz2/download
$ tar -xvf OpenCV-2.4.4a.tar.bz2
```

E infine alla compilazione e installazione:

```
$ cd OpenCV-2.4.4a
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMA-
KE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D
BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
INSTALL_C_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D
WITH_QT=ON -D WITH_OPENGL=ON ..
$ make
$ sudo make install
```

A.1.3 Cvblob

Per prima cosa è necessario scaricare dal sito `http://code.google.com/p/cvblob/downloads/list` la versione 0.10.4 di cvblob e scompattarlo nella cartella CVBLOB.
Per compilare eseguire questi comandi:

```
$ cd CVBLOB
$ cmake . -DOpenCV_DIR=indirizzo_OpenCv
$ make
```

Successivamente cvblob viene installato con il seguente comando:

```
$ sudo make install
```

Una volta installato è necessario modificare il file *cvlabel.cpp*. Nello specifico vanno modificate le linee 34 e 40 portandole da:

const char movesE ...
const char movesI ...
a:
const signed char movesE ...

const signed char movesI ...

A.2 Configurazione pc remoto

Il pc remoto non è indispensabile per il volo autonomo del drone, tuttavia può essere utile per monitorare le informazioni di base del quadricottero e del gioco stesso.
Il sistema operativo del pc utilizzato è Ubuntu 12.10.
Anche sul pc remoto deve essere installato ROS, la cui versione è però ros-groovy-desktop-full. In questo modo sono presenti tutti i pacchetti utili per gli scopi del pc remoto (e molti altri non indispensabili). Per installare questa versione eseguire il comando da terminale:

```
$ sudo apt-get install ros-groovy-desktop-full
```

Eseguire poi la stessa procedura usata per l'installazione di ROS su Odroid, con l'unica differenza di copiare nel workspace il contenuto della cartella RemoteSW.
Andare poi nella cartella */opt/ros/groovy/stacks/robot_model_tutorials/urdf_tutorial* e copiare al suo interno il contenuto della cartella ModelQuadri. Per quanto riguarda la comunicazione via rete vanno svolti questi comandi:

```
$ echo export ROS_IP=proprio_indirizzo >>~/.bashrc
$ echo export ROS_MASTER_URI=http://indirizzo_Odroid:11311
>>~/.bashrc
```

Come ultima cosa è necessario copiare il contenuto della cartella Sounds e quello della cartella RVizConfiguration.

A.3 Configurazione scheda di volo

Sulla scheda di volo occorre caricare il software di volo scelto ovvero il Multiwii. La versione che abbiamo utilizzato e modificato nel nostro lavoro è la 2.2 (scaricabile dall'indirizzo `http://code.google.com/p/multiwii/downloads/list`).
Poiché la scheda utilizzata ha a bordo un microcontrollore ATMega2560, il software si carica attraverso l'interfaccia di Arduino.
Collegare quindi la scheda di volo ad un pc (oppure farlo direttamente dall'Odroid) ed aprire l'interfaccia di Arduino. Tramite il menu Strumenti →

Tipo di Arduino, bisogna selezionare Arduino Mega 2560 or Mega ADK.
A questo punto si può caricare lo sketch fornito nella cartella MultiwiiSketch.

A.4 Configurazione Arduino

Tramite l'interfaccia di Arduino aperta dall'Odroid, scegliere Arduino Nano w/ATmega 168 dal menu Strumenti → Tipo di Arduino.
A questo punto si può caricare lo sketch fornito nella cartella ArduinoSketch.

Appendice B

Manuale Utente

B.1 Esecuzione applicazione di bordo

I seguenti comandi vengono svolti sull'Odroid. Il metodo per collegarsi al computer di bordo è lasciato al lettore. Nel nostro caso sono stati utilizzati sia un desktop remoto che il protocollo ssh. Per accedere tramite il desktop remoto, bisogna connettersi tramite protocollo vnc all'indirizzo IP dell'Odroid. La password per accedere è *vvtesi*.
Prima di far partire il gioco è necessario eseguire roscore, cioè l'applicazione centrale del sistema ROS. Da un terminale:

```
$ roscore
```

Per avviare l'applicazione di bordo è sufficiente eseguire da un altro terminale il seguente comando:

```
$ rosrun quadricoptermt server
```

Per interrompere il programma premere 0 e il tasto Invio dallo stesso terminale in cui è in esecuzione l'applicazione.
Attenzione: se il programma viene terminato durante il volo si spengono i motori causando la caduta del drone.

B.2 Esecuzione applicazioni del pc remoto

Queste applicazioni vengono eseguite dal pc remoto. Per essere avviate necessitano roscore e l'applicazione di bordo spiegata precedentemente in esecuzione sull'Odroid.

B.2.1 Esecuzione applicazione monitoraggio volo

Questa applicazione ha la funzione di mostrare le principali informazioni di volo ottenute dai sensori e dalla webcam.
Prima di eseguire l'applicazione è necessario caricare il modello del drone:

```
$ roscd urdf_tutorial
$ roslaunch urdf_tutorial display2.launch model:=quadricopter.urdf
```

Per questa applicazione è stato utilizzato il pacchetto ROS rqt, eseguibile dal comando:

```
$ rqt
```

Dalla finestra aperta è necessario aggiungere due plugin: Image View e RViz.
Nella finestra di Image View è necessario selezionare il topic /camera/image/compressed.
Se la procedura è stata svolta correttamente, all'interno della finestra comparirà lo streaming video della webcam di bordo.
Per quanto riguarda RViz bisogna caricare la configurazione estratta nella posizione scelta durante la configurazione del pc remoto.

B.3 Esecuzione applicazione monitoraggio radiocomando

L'applicazione serve a monitorare i comandi del radiocomando che vengono calcolati e quindi fatti eseguire dal software di bordo del drone autonomo.
Per essere eseguita digitare:

```
$ rosrun quadricoptermt listener
```

B.4 Esecuzione applicazione gioco

Questa applicazione fornisce le informazioni di gioco e i suoni necessari a far capire in che fase ci si trova.
Per eseguire l'applicazione è necessario:

```
$ cd cartella_dove_sono_stati_estratti_i_suoni
$ rosrun quadgame game
```

B.5 Esecuzione applicazione comandi

Lo scopo di questa applicazione è stato essenzialmente quello di testare alcuni fasi di gioco. Per avviare l'applicazione aprire il terminale e eseguire:

```
$ rosrun quadcmd commander
```

Successivamente è possibile inserire nello stesso terminale i vari comandi disponibili. Ricordiamo che tali messaggi possono essere:

- **land:** il drone atterra.
- **laser:** il drone accende il puntatore laser.
- **enable:** il drone attiva la gestione ostacoli.
- **disable:** il drone disabilita la gestione ostacoli.
- **h:** seguito da un valore serve a cambiare l'altezza del drone.

Appendice C

Manuale operatore

C.1 Radiocomando

Il radiocomando utilizzato mette a disposizione 6 canali indipendenti di cui 4 utilizzati per il volo e 2 per il gioco.
Come mostrato in Figura C.1 la leva sulla sinistra è collegata orizzontalmente al canale di roll e verticalmente a quello di throttle. La leva sulla destra è collegata orizzontalmente al canale di yaw e verticalmente a quello di pitch.

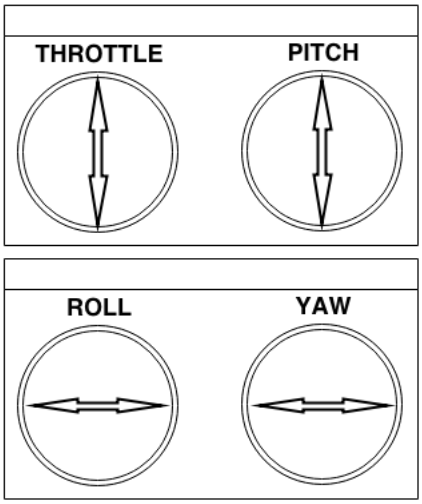


Figura C.1: Comandi leve radiocomando

I due interruttori posti sopra al radiocomando sono invece collegati ai canali di AUX1 e AUX2 come mostrato in Figura C.2.
L'interruttore collegato ad AUX2 è un interruttore a scatto che ritorna automaticamente alla sua posizione naturale dopo averlo rilasciato.

Figura C.2: Interruttori radiocomando

C.2 Combinazioni comandi per il radiocomando

Tramite particolari combinazioni del radiocomando è possibile inviare comandi specifici alla scheda di volo.
Essi servono in particolari situazioni come ad esempio per armare il drone o per calibrare gli accelerometri.

C.2.1 Multiwii

Le seguenti combinazioni sono proprie del software di volo utilizzato ovvero il Multiwii.

Armare i motori

Come mostrato in Figura C.3, per armare i motori occorre posizionare il comando di throttle al minimo (verso il basso) ed il comando di yaw al massimo (verso destra).
Quando i led di segnalazione sulla scheda di volo si accendono in maniera fissa, i motori sono armati. Per ragioni di sicurezza, se la scheda è inclinata con un angolo maggiore di 25 gradi, non si possono armare i motori.

Disarmare i motori

Come mostrato in Figura C.4, per disarmare i motori occorre posizionare il comando di throttle al minimo (verso il basso) ed il comando di yaw al minimo (verso sinista).

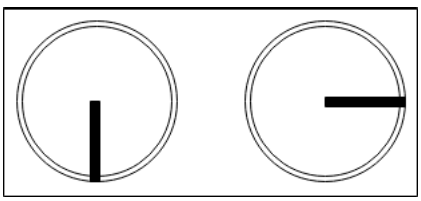


Figura C.3: Combinazione per armare i motori

Quando i led di segnalazione sulla scheda di volo si spengono i motori sono disarmati.

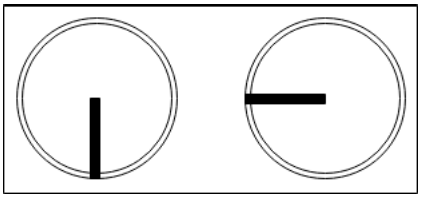


Figura C.4: Combinazione per disarmare i motori

Calibrare gli accelerometri

La calibrazione degli accelerometri è un'operazione che bisogna svolgere ogni volta che si parte per un nuovo volo. Durante l'operazione il drone deve essere disarmato e posto perfettamente in piano.
Come mostrato in Figura C.5, per calibrare gli accelerometri occorre posizionare il comando di throttle al massimo (verso l'alto), il comando di pitch al minimo (verso il basso) ed il comando di yaw al minimo (verso sinistra). Una volta calibrati gli accelerometri, i led di segnalazione sulla scheda di volo lampeggiano per qualche secondo per poi spegnersi.

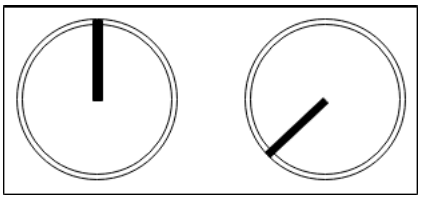


Figura C.5: Combinazione per calibrare gli accelerometri

Calibrare il magnetometro

La calibrazione del magnetometro si compie una sola volta dopo aver montato il drone. L'operazione consiste nel ruotare manualmente il drone lungo i tre assi di roll, pitch e yaw per 30 secondi. Come mostrato in Figura C.6, per iniziare la calibrazione occorre posizionare il comando di throttle al massimo (verso l'alto), il comando di pitch al minimo (verso il basso) ed il comando di yaw al massimo (verso destra).
I led di segnalazione sulla scheda di volo lampeggiano per 30 secondi per poi spegnersi a calibrazione terminata.

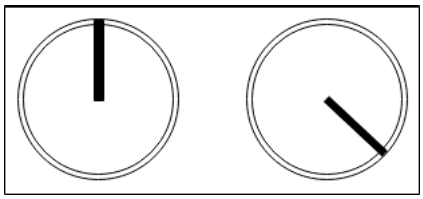


Figura C.6: Combinazione per calibrare il magnetometro

Scegliere modalità

La scelta della modalità di volo automatica o manuale non è propria del Multiwii ma è stata aggiunta nell'ambito del nostro lavoro per poter intervenire tempestivamente in caso di pericolo.
In particolare quando l'interruttore dell'AUX1 è posto verso il pilota, la modalità scelta è quella manuale. In questo caso la scheda di volo legge regolarmente i comandi provenienti dal radiocomando. Se l'interruttore è invece posto nel verso opposto al pilota (nella posizione che si può vedere in Figura C.2), la modalità scelta è quella automatica. In questo caso la scheda di volo legge i comandi dalla seriale e scarta automaticamente quelli proveniente dal radiocomando.

C.2.2 Gioco

Le combinazioni seguenti sono utilizzate all'interno del gioco. Contrariamente a quelle mostrate in precedenza, queste sono rilevate dal nostro software e non dalla scheda di volo.

Inizio partita

Per poter cominciare una nuova partita occorre posizionare il comando di throttle al minimo (verso il basso) e portare l'interruttore della modalità su

automatico.
In questo modo, se è attivo il software il bordo, verranno calibrati gli accelerometri, armati i motori e iniziato il decollo.
Attenzione: assicurarsi che sia attiva la modalità manuale quando si avvia l'applicazione di bordo per evitare che il drone parta inaspettatamente.

Atterraggio

Il comando di atterraggio si può fornire quando il drone è in modalità automatica ed è attivo il software di bordo.
Per cominciare l'atterraggio occorre semplicemente tirare verso di sé l'interruttore di AUX2 e rilasciarlo subito dopo.