Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Progettazione e valutazione sperimentale di una piattaforma di mobilità controllata per UAV-aided sensor networks

Relatore:
Chiar.mo Prof.
Marco Di Felice

Presentata da:
Leonardo Montecchiari

Correlatore:
Dott.
Angelo Trotta

Sessione II - primo appello
Anno Accademico 2018/2019

Ai miei Genitori, di nuovo
perché nonostante tutte le difficoltà
sono riusciti a farmi portare a termine anche questo percorso
senza farmi mai mancare nulla.

A mio fratello,
per tutto l'appoggio sia morale che economico
e perché senza di lui non avrei mai riaperto il capitolo
"elettronica"
ormai fondamentale per la mia carriera accademica.

A lollo,
perché ci siamo dati forza a vicenda
nei momenti difficili.
Compagno di battaglie.

Sommario

Con “mobilità controllata” s’intende la tecnica attraverso la quale si utilizzano veicoli (droni/rover) sprovvisti di equipaggio ed in grado di controllare in maniera autonoma la propria posizione. Molte applicazioni della mobilità controllata fanno riferimento a scenari nel quale il veicolo a guida autonoma funge da collettore dati raccolti da installazione di sensori a terra.

Questo elaborato presenta Sybelius, un piattaforma di mobilità controllata per UAV-aided sensor networks. É stato progettato un sistema di controllo autonomo per “Unmanned Aerial Vehicle” (UAV), in grado di eseguire delle intere “Missioni” di volo con lo scopo di raccogliere dati da sensori a terra. La sperimentazione è stata effettuata in ambiente outdoor usando un drone “manned” ed dotandolo di apposito coordinatore di bordo e di modulo per l’acquisizione dati dai sensori a terra.

Sono stati eseguiti test sperimentali relativi alla direzionalità, distanza e potenza della comunicazione drone-sensore, alla precisione del puntamento verso la coordinata GPS del sensore a terra e all’efficienza energetica in fase di comunicazione, volo stazionario e volo continuo.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco listati di codice

Introduzione

Gli “Unmanned Aerial Vehicle”(UAV), meglio conosciuti come droni, vengono ormai utilizzati ben oltre l’ambito amatoriale. Piccoli droni a rotore o ad ala fissa[AVVR15] senza pilota ad oggi sono utilizzati su larga scala in applicazioni eterogenee quali: videosorveglianza, smart farming (agricoltura 4.0), fotogrammetria, gestione delle calamità, sicurezza civile, tracciamento degli incendi, controllo atmosferico, monitoraggio delle strutture, e controllo della smart grid[Sha+18] Il classico approccio prevede il controllo dell’ UAV effettuato da un utente a terra grazie ad un controller wireless. Negli anni il controllo è stato in parte automatizzato, consentendo azioni complesse ma sempre impartite da una stazione di controllo a terra. In ogni caso, il drone necessita di una connessione continua con il coordinatore in modo da consentire l’esecuzione del task richiesto. L’ elaborato si spinge oltre lo stato attuale e investiga il concetto di “Mobilità Controllata”, ovvero, la possibilità di usare veicoli (droni/rover) in grado di controllare in maniera autonoma la propria posizione. I potenziali vantaggi sono molteplici, quali la copertura aerea di zone rurali, la possibilità di adattare il piano di volo in base alle esigenze di ricarica, di implementare politiche di coordinamento distribuito con altri UAV direttamente, oppure di ottimizzare la comunicazione con rete di sensori a terra[Don+14]. Relativamente a quest’ultimo punto, nella tesi viene presentata la progettazione e la valutazione sperimentale di Sybelius, una piattaforma di mobilità autonoma per UAV in grado di recuperare dati da una rete di sensori a terra. I macro-obiettivi principali di questo progetto sperimentale sono due:

- **Controllo Autonomo**
  ossia, la progettazione e valutazione sperimentale di un'architettura per il controllo autonomo di un drone amatoriale, tipicamente usato in modalità manned.

- **Comunicazione con WSN**
  ossia, l' implementazione di strategie per l'acquisizione dati da un sensore a terra usando tecnologie quali Zibee, Bluetooth Low Energy(BLE), WiFi.

Il coordinatore installato sul Parrot sfrutta una pc-board Raspberry Pi 3 che controlla il drone tramite protocollo MavLINK. Per la parte di comunicazione, è stato testato l'utilizzo di tecnologia BLE per la comunicazione drone-sensore. L'architettura della piattaforma permette l'utilizzo di altri modelli di UAV; anche la comunicazione è completamente estendibile. La piattaforma proposta è stata poi oggetto di validazione sperimentale, mediante test outdoor attraverso i quali sono state caratterizzate le performance del sistema di controllo autonomo, della funzionalità di comunicazione con il sensore, e dell'efficienza energetica.

Questo elaborato è diviso in due parti: Nello stato dell'arte viene presentato il capitolo 1 relativo al mondo dell' IoT e degli UAV. Nel secondo capitolo vengono presentate le tecnologie e i protocolli di comunicazione tra UAV e rete di sensori wireless.

Nella seconda parte, quella di sperimentazione, viene presentata la piattaforma Sybelius. Nel capitolo 3 vengono discussi gli obiettivi di questo progetto, le scelte progettuali che hanno influito nella progettazione ed infine l'architettura finale della piattaforma. Il capitolo 4 descrive tutta l'implementazione del software, soprattutto le scelte hardware e software che hanno portato al prototipo finale nel punto 4.3.

Nel capitolo 5 viene presentata la validazione, in particolare viene descritto lo scenario di partenza ricreato per i vari test, la metodologia, le metriche ed il commento ai risultati ottenuti.

Parte I

Stato dell'arte

Capitolo 1

Utilizzo degli UAV in ambito IoT

L' elaborato si focalizza principalmente nello studio e nell'utilizzo di UAV altresì conosciuti come droni, per agevolare le comunicazioni con una rete di sensori wireless posti al suolo. Inizialmente verrà introdotto il concetto di Internet Of Things (IoT) e di come sia diventato così fondamentale anche nel mondo consumer. Successivamente verranno presentati i sistemi basati su UAV e le loro integrazioni con reti di sensori wireless.

1.1 IoT in generale

L'Internet Of Things letteralmente traducibile in "Internet delle cose", estende la connettività Internet agli oggetti fisici della vita di tutti i giorni. Partendo dalla semplice accensione di una lampadina tramite un comando su un'app, fino ad arrivare al controllo automatico di una serra, i device IoT sono ormai diffusi in molti settori.

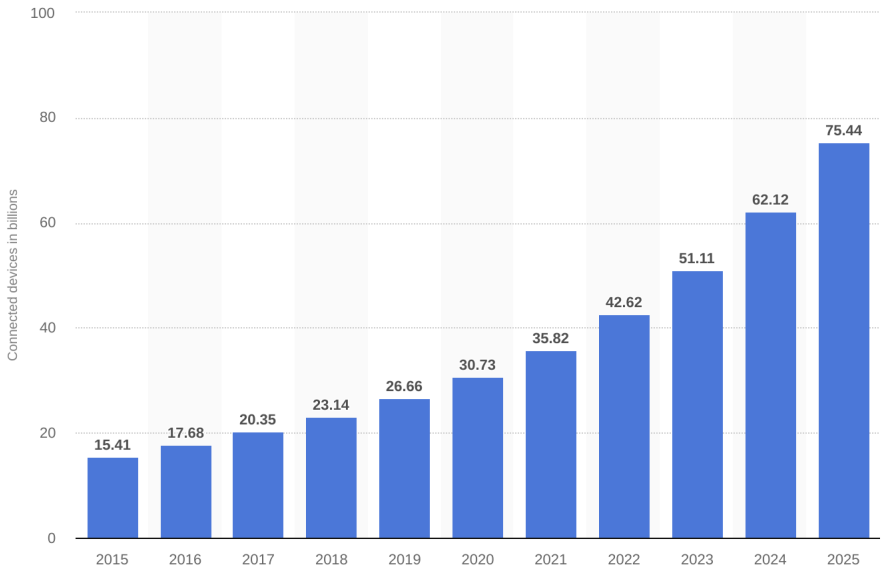

Figura 1.1: Numero di dispositivi IoT installati e connessi a livello mondiale dal 2015 al 2025 (in miliardi)fonte: [Sta19]

Dal grafico 1.1 emerge come l'utilizzo di device IoT sia in costante aumento, dovuto anche principalmente all'abbattimento dei costi dell'hardware e alla realizzazione di dispositivi sempre più eterogenei. Possiamo vedere che nel decennio 2015-2025 il numero di dispositivi installati e connessi si è quasi quintuplicato.

I device connessi sono in grado di automatizzare attività domestiche o industriali e di comunicare dati raccolti tramite il sensing.

- **Consumer:** Smart-tv, Smart speakers, giocattoli.
- **Enterprise:** Sala riunioni smart.
- **Industriali:** Linea di lavoro automatizzata.

Nel seguente grafico possiamo individuare le effettive installazioni di device IoT divise per settori.

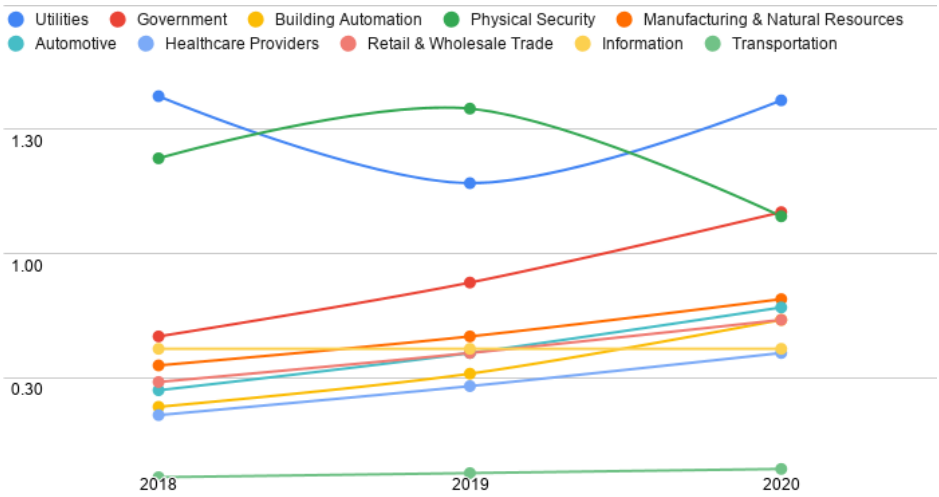

Figura 1.2: Numero di dispositivi IoT divisi per settore nel triennio 2018-2019-2020[Gar19]

Si può notare come le categorie “utilities” e “physical security” siano quelle che offrono gli scenari evidentemente più appetibili per l’utilizzo di dispositivi IoT.

Per definirsi device IoT, è necessaria la presenza di un’unità di computazione in grado di fare sensing, di processare dati raccolti, e connettersi alla rete. Di seguito vengono introdotte le unità di computazione più comuni.

1.1.1 Le unità di computazione

Esistono svariate tipologie di unità di computazione, ma riconducibili principalmente a due grandi famiglie:

- microcontrollori
- microprocessori: pc, single-board pc

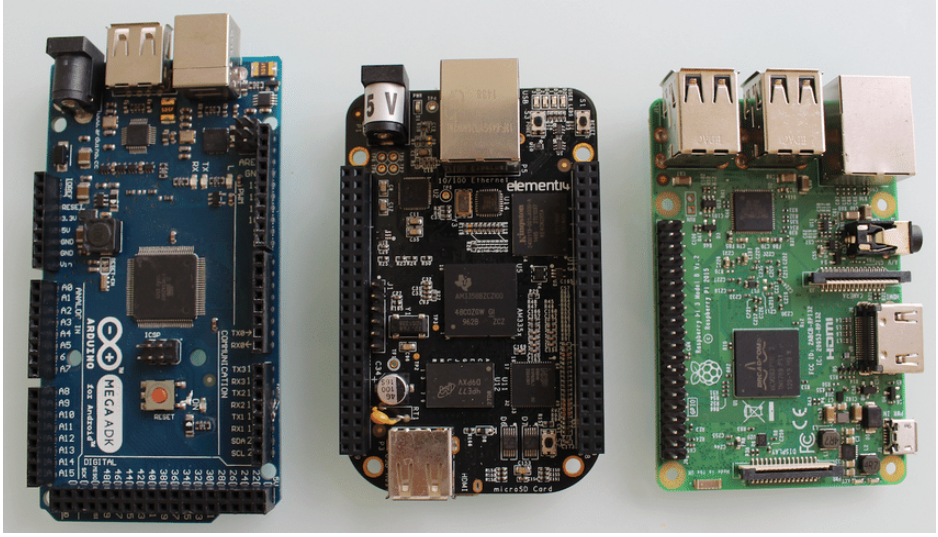

Figura 1.3: : Da sinistra: Arduino Mega ADK, BeagleBone Black, Raspberry Pi 3B

La differenza sostanziale è che il microprocessore presenta nel chip soltanto la logica di elaborazione e quindi necessita delle unità d'appoggio (memorie, dispositivi periferici etc) per poter interagire con l'esterno; al contrario, il microcontrollore nel suo chip presenta già una struttura completa composta da microprocessore, memoria permanente, memoria volatile ed eventuali pin di comunicazione input/output. Per quanto riguarda l'ambito IoT possiamo citare le schede a microcontrollore della serie Arduino e ESP oppure le versatili single-board pc Raspberry Pi, C.H.I.P. . La scelta dipende da requisiti del sistema da sviluppare, in base alla potenza di calcolo richiesta, allo spazio a disposizione, ai consumi energetici, etc.

1.2 Introduzione agli UAV

Gli UAV sono ampiamente utilizzati in modalità di controllo manuale per scopi di video-sorveglianza o per ottenere ricostruzioni fotogrammetriche dell' analizzato. In letteratura, tali dispositivi (rover/droni) sono anche denominati "unmanned vehicles" e ad oggi vengono utilizzati su larga scala per

applicazioni eterogenee quali: videosorveglianza, smart farming (agricoltura 4.0), fotogrammetria, gestione delle calamità, sicurezza civile, tracciamento degli incendi, controllo atmosferico, monitoraggio delle strutture, e controllo della smart grid[Sha+18]. Molte delle applicazioni appena citate sono principalmente legate ad una sottocategoria denominata "unmanned aerial vehicles" (UAV) o più comunemente droni. Quest'ultimi possono appartenere a due tipologie[AVVR15] sulla base della configurazione: ad ala fissa e a rotori, entrambe con vantaggi e svantaggi da tenere in considerazione in base all'applicazione di riferimento.

- Gli **UAV ad ala fissa** presentano una struttura in grado di generare sufficiente portanza da mantenere il velivolo sospeso; il controllo del movimento viene effettuato operando sulle superfici delle ali tramite alettoni e timone. L'aerodinamica del velivolo permette di raggiungere lunghe sessioni di volo, alte velocità, e di avere dei carichi maggiori rispetto alla configurazione a rotori, a costo di una minore manovrabilità, in quanto non è possibile effettuare un'operazione di hovering o volo stazionario. Inoltre, è spesso necessaria una pista per effettuare le operazione di decollo e atterraggio.

Figura 1.4: : UAV ad ala fissa: Delair UX11

- L'altra configurazione, **UAV a rotori**, presenta delle eliche poste su dei motori brushless o stepper in modo da consentire un'alta manovrabilità; sono infatti in grado di effettuare operazioni verticali e di conseguenza anche decollo e atterraggio, volo a bassa quota e volo stazionario. Al tempo stesso, l'uso dei rotori produce abbastanza forza aerodinamica da non richiedere alcun tipo di velocità relativa, ma al tempo stesso ne riduce la durata di volo e la quantità di carico.

Figura 1.5: : UAV a 6 rotori: Falcon Series Six-rotor

In entrambi i casi, è necessario quindi pianificare in maniera opportuna la traiettoria del drone tenendo conto delle limitazioni sulla manovrabilità e dell'autonomia.

A tal proposito, un problema molto studiato in letteratura è il Coverage Path Planning (CPP), ossia come pianificare un percorso del drone in modo da esplorare la totalità dell'ambiente per acquisizione dati interna (videocamera) o esterna (comunicazione con sensori a terra). Gli studi proposti differiscono in base all'obiettivo del CPP, es. minimizzare la lunghezza del percorso del drone o massimizzare il numero di letture dati effettuate[CBF19]. Uno dei vantaggi nell'utilizzo dei droni è quello di poter comunicare con una rete di sensori wireless installata in un'area, con lo scopo di rilevare ulteriori dati significativi non ricavabili solamente dagli strumenti di bordo e di trasferire poi i dati raccolti ad un centro di processamento. In questo caso il tipo di drone ideale è quello a rotori, poiché permette di rimanere sospeso in volo esattamente nel punto perpendicolare rispetto al sensore a terra ottenendo la massima direzionalità del segnale (Wi-Fi, Bluetooth, ZigBee).

In quest'ambito, la ricerca si concentra su due macro-obiettivi:

1. come supportare la comunicazione dati tra sensori statici e droni mobili mediante opportune tecnologie wireless e protocolli di comunicazione[Don+14];

2. come processare i dati, limitando la comunicazione con il cloud, massimizzando il processamento locale degli stessi (edge computing) ed evitando problemi di privacy riguardanti i dati grezzi. [SCZLX16].

A prescindere dalla tipologia di operazione svolta dal drone, la criticità maggiore deriva dalla bassa autonomia da parte del veicolo, dovuta al rapporto peso-potenza (non meno di 2:1) dei rotori. Quindi, installando a bordo una batteria che rispetta la proporzione, normalmente non si riesce a raggiungere la mezz'ora di autonomia, rendendo difficile la pianificazione di operazioni lunghe e al contempo complesse. Per mantenere attivo un determinato servizio e per alleggerire il carico, la ricerca del settore si concentra sul concetto di "swarm" di droni, ovvero, una rete di droni distribuita ed auto-configurante in grado di calibrarsi real-time in base alle esigenze[TFMCB18]. Questa configurazione permette di sfruttare tecniche di edge computing per svolgere un determinato task [CLHGZ19] e di distribuire gli oneri di copertura dello scenario tra vari nodi. Tuttavia, opportuni algoritmi di swarm path-planning e di copertura distribuita devono essere progettati[YZYL17].

1.3 IoT e UAV casi d'uso

Come già annunciato nella sezione precedente, gli UAV vengono impiegati anche in contesti IoT nello specifico nella comunicazione con reti di sensori installate a terra. Nel Farming 4.0 una rete di sensori monitora costantemente lo stato del terreno in vari punti (temperatura, umidità, vento etc), mentre l'UAV ha il compito non solo di monitorare visivamente la situazione, ma anche quella di recuperare i dati generati dai sensori attraverso comunicazioni wireless [NCKS17].

Figura 1.6: : Due droni d'appoggio durante la fertilizzazione dei campi

Nel caso in cui il nodo sensore non possa accedere costantemente alla rete per fare offload dei dati, l'UAV funge da collettore (nodo sink) dati, a patto di trovare un compromesso tra durata delle batterie dell'UAV, potenza di calcolo e connessione. Vi sono studi[Chi+13][Tok+11] per cui il drone nel suo path-planning aumenta l'autonoma attraverso operazioni di ricarica delle batterie presso stazioni di ricarica poste per l'appunto in zone remote, magari alimentate con pannelli solari. Un altro contesto è quello di poter monitorare gli edifici di un intero quartiere sotto l'aspetto strutturale. Gli edifici presi in considerazione potrebbero essere muniti di sensori tali da leggere eventuali anomalie sullo stato del cemento, innalzamento del terreno e conseguenti deformazioni strutturali, picchi di umidità dovuti a delle infiltrazioni d'acqua, etc. Altri studi sono rivolti al garantire una rete di emergenza in caso di calamità naturale, principalmente utilizzando solamente la connettività offerta dagli UAV, o estendendo il raggio di infrastrutture wireless esistenti. Vengono presentati di seguito due casi applicativi in ambito UAV.

1.3.1 Monitoraggio della fauna selvatica

Fin'ora lo strumento principale per il monitoraggio della della fauna è costituito da velivoli ad ala fissa con pilota o elicotteri. Tuttavia, tali metodologie tendono ad essere costose e presentare problematiche sotto aspetti di sicurezza, disturbo della fauna e logistici sebbene rendano accessibili aree

difficili da raggiungere con metodi convenzionali. Tra i tanti possibili utilizzi degli UAV, Watts e altri[WAH12] hanno riscontrato che per monitorare animali terrestri e specie marine, gli uav sono meno invasivi rispetto alle tecniche tradizionali[Wat+10]. Sono stati effettuati degli studi sull'impatto che hanno i classici velivoli rispetto alle specie volatili. In generale tra voli di linea e voli militari, l'inquinamento acustico raggiunge i 103.1dB con un rumore di 60-70dB ad altitudini tra i 18m e i 61m. Il Raven RQ-11A, un drone da "decollo manuale" sviluppato per il dipartimento della difesa US dalla AereoVironment è stato progettato come strumento per missioni di ricognizione e di raccoglitore-dati. Nello specifico, è stato impiegato nel monitoraggio delle Gru Canadesi nella riserva naturale di Monte Vista in Colorado(USA). Il risultato principale è stato quello di dimostrare che una ricognizione del Raven sia effettivamente affidabile e capace di ottenere dati paragonabili all'osservazione classica a terra. Gli osservatori hanno contato 2692 esemplari contro i 2567 ricavati dalle immagini del Raven (differenza del 4.6%) confermando quindi le aspettative.

1.3.2 Deployment di reti di emergenza

Negli anni si sono dovute affrontare situazioni di emergenza a seguito di disastri ambientali; con il progredire della tecnologia mobile, uno degli interventi principali è stato quello di costruire infrastrutture di backup in grado di offrire connettività nelle aree interessate. Tali infrastrutture possono supportare la localizzazione delle vittime e la comunicazione con i first responder. D'altro canto, vi è una reale difficoltà nel ripristinare la rete, poiché normalmente l'infrastruttura che la distribuisce non può essere riparata tempestivamente. Attualmente, sono state proposte due piattaforme per comunicazione aerea in scenari d'emergenza: le High Altitude Platforms (HAP) di Google[KR18], e le stazioni a pallone aereostatico 4G LTE ad alta quota di Oceus[RES13]. Tali soluzioni sono principalmente rivolte per usi commerciali e militare, ma anche come risposta a disastri ambientali. Rispetto ad un intervento su ruote (posizionando le stazioni a terra), l'utilizzo

di reti mesh aree garantisce maggiore copertura e tempestività, soprattutto se sussistono impedimenti fisici quali esondazioni, crolli etc.

In letteratura sono stati effettuati vari esperimenti sulla connettività, copertura, scalabilità e interferenza con le comunicazioni terrestri[MYLZ19] le misurazioni ed i modelli teorici risultano fondamentali per la calibrazione e deployment di reti aeree.

Capitolo 2

Tecnologie e protocolli di comunicazione UAV-WSN

Nell'ambito della robotica, il coordinamento multiplo di unità autonome, come rover o droni, richiede competenze trasversali da tre macro-aree di studio:

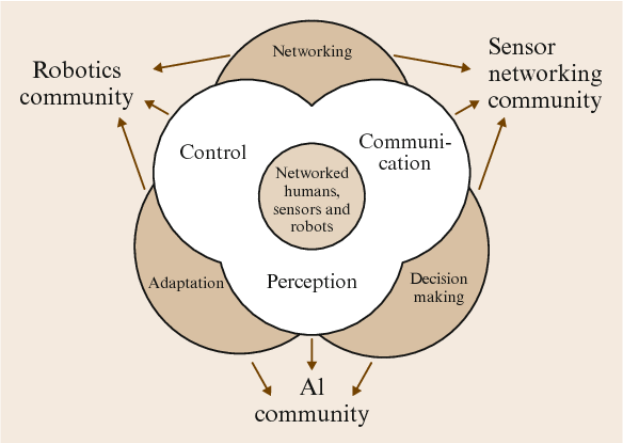

Figura 2.1: : Il paradigma dei robot collegati in rete [KRS08]

- **Comunicazione**
  è relativo allo scambio dati di tipo controllo e di applicazione (come sensoristica) con altre unità, siano essi dispositivi a terra o altri UAV. In

quest'ultimo caso, si formano le cosiddette "Flying Ad Hoc Networks" (Fanet) caratterizzate da:

  - alta densità dei nodi della rete in volo;
  - alta frequenza di cambiamenti della topologia (più alta rispetto a reti mobili quali Manet o Vanet);
  - bassa lifetime della rete a causa dell'autonomia limitata dei dispositivi;
  - possibilità di controllare autonomamente la mobilità dei dispositivi e quindi della rete.

- **Controllo**
  si occupa di decidere la posizione e l'orientamento dell'unità.

- **Percezione**
  acquisisce ed analizza le informazioni dello scenario in cui è l'unità, utilizzando sensori onboard e camera con input-data.

In questo elaborato, verrà principalmente trattato il tema della comunicazione tra UAV e sensori onboard/terra. Soltanto in una minima parte verrà esplorato il tema del controllo.

2.1 Tecnologie di Controllo UAV

Per definizione un UAV è un velivolo senza pilota e viene quindi controllato tramite comandi wireless. Durante la trasmissione vengono scambiati pacchetti di tipo controllo e di stato. A prescindere dalla tecnologia, uno dei protocolli di riferimento è il MAVLink.[1]

[1] https://mavlink.io/en/

2.1.1 Protocollo MAVLink

MAVLink è un protocollo di messaggistica leggero usato per la comunicazione tra droni e elementi onboard. Il protocollo implementa un pattern ibrido tra publish-subscribe e point-to-point: lo streaming di dati viene inviato / pubblicato come argomento; mentre le configurazioni dei sotto-protocolli come il **mission protocol** o il **parameter protocol** sono point-to-point per mezzo di una ritrasmissione. I Messaggi vengono definiti in file XML. Ogni file XML definisce un set di messaggi supportate da un particolare sistema MAVLink, chiamato anche "dialetto". Normalmente si fa riferimento ad un set di messaggi di partenza normalmente usati tra le stazioni di controllo a terra (GCS) e e gli UAV ed è definito nel common.xml; i dialetti vengono solitamente creati partendo dal common. Il tool ufficiale di MAVLink usa le definizioni dei messaggi per generare liberie MAVLink per ogni linguaggio di programmazione supportato. Droni, stazioni di controllo e altri sistemi MAVLink usano le librerie generate per comunicare. Normalmente sono sotto licensa MIT e possono essere usate senza limitazioni in qualsiasi applicazione closed-source senza l'obbligo di pubblicarne il codice sorgente. I vantaggi principali del protocollo si possono riassumere in:

- è molto efficiente. MAVLink 1 usa 8 bytes per pacchetto, incluso il segnale di start e il rilevamento del packet drop. Il MAVLink 2 usa 14 bytes ma è molto più sicuro ed estendibile. Tale caratteristica rende il protocollo particolarmente adatto per applicazioni con larghezza di banda limitata.

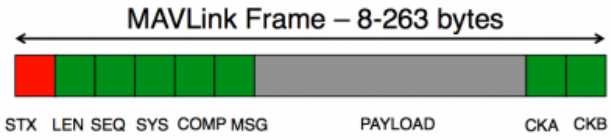

Figura 2.2: : Struttura del pacchetto MAVLink 1

- è molto affidabile. Viene ormai utilizzato dal 2009 per le comunicazioni tra veicoli, stazioni (o altri tipi di nodi) attraverso canali di trasmissione ad alta latenza/rumore. Fornisce metodi per monitorare le prestazioni e le operazioni sul canale, quali packet drop, collisone ed autenticazione.

- supporta molti linguaggi di programmazione e può essere eseguito su diversi microcontrollori/sistemi operativi (lista parziale: ARM7, ATMega, dsPic, STM32 e Windows, Linux, MAcOS, Android e iOS).

- permette fino a 255 sistemi simultanei nella rete (tra veicoli, stazioni e altri nodi)

- abilita sia le comunicazioni verso l'esterno e quelle interne ad esempio GSC-Drone, Drone e Drone-camera.

Mission sub-Protocol

Il Mission sub-Protocol permette alle GCS o agli sviluppatori dell'API di gestire le informazioni delle missioni (o piani di volo), gefence o safe point nel drone/ elemento onboard. Il protocollo include:

- Le operazioni per caricare, scaricare e cancellare missioni, impostare ed esportare il numero corrente dell'elemento e notificare quando la missione cambia elemento.

- i tipi di messaggio per cambiare gli elementi missione.

- i comandi MAVLink che sono comunemente usati per i veicoli/GCS.

Il protocollo si basa sul pattern client/server: le operazioni e gran parte dei comandi vengono avviati dalla GCS/api(client) e accettati dall'autopilota (server). Il Mission Protocol supporta il riavvio delle richieste dei messaggi che non sono arrivati, permettendo alle missioni di essere affidabili anche in connessioni non ottimali.

Parameter sub-Protocol

Il microservizio Parameter viene utilizzato per cambiare la configurazione dei settaggi tra i componenti MAVLink. Ogni parametro è rappresentato da una coppia di chiave/valore. La chiave normalmente è il nome (facilmente intuibile dall'utente) del parametro, massimo 16 caratteri. Il valore può essere un numero presente nella tabella dei tipi del protocollo (normalmente dall'1 fino al 10). La coppia chiave/valore ha le seguenti caratteristiche:

- il nome human-readable è corto ma riesce a codificare i nomi in modo tale da renderne intuibile lo scopo ultimo.
- gli autopilota che implementano il protocollo vengono supportati "out of the box".
- una GCS non deve sapere a priori quali parametri esistono in un sistema remoto.
- aggiungere un parametro richiede delle modifiche nel sistema che definisce i parametri. Invece l GCS che carica i parametri e le librerie MAVLink di comunicazione non necessitano alcuna modifica.

2.1.2 Controllo Human-UAV vs controllo Automatico

Il GCS, ovvero il sistema di controllo a terra, può essere effettuato tramite smartphone o pc in grado di supportare il protocollo ed impartire "ordini". In uso domestico o racing i droni vengono pilotati direttamente da un utente in modalità real-time, decidendo il movimento del velivolo. App o radiocomandi permettono di eseguire questo lavoro in modo egregio anche a diverse decine di metri di distanza. Nell'uso militare e industriale la GCS può essere automatizzata in modo tale da governare il drone in base all'ambiente che lo circonda, portando a termine un pre-determinato task specifico. La DJI ha recentemente introdotto la funzionalità "GS PRO" che permette ai propri modelli top gamma di poter effettuare missioni di volo automatizzate,

registrando i dati di volo sul loro cloud. Nei dati di volo sono comprese le registrazioni video, e immagini ad alta risoluzione con la miglior angolazione possibile, regolabile dall'applicazione ufficiale per iOS, tali da poter ricostruire una mappa 3D della zona di volo. Esiste anche la possibilità di poter rendere completamente autonomo un UAV, aggiungendo non poca complessità hardware e di calcolo. Parte di questo elaborato consiste nel progettare e sviluppare un sistema in grado di rendere un UAV "manned" potenzialmente autonomo.

2.1.3 Swarm UAV

Vista la possibilità di poter controllare un drone in modo automatico, in letteratura viene discusso il concetto di "Swarm UAV". Lo swarm presenta una struttura multi-UAV in grado di andare a compiere un task coordinato in base alle esigenze del momento e allo stato di ogni componente del gruppo. La sfida della ricerca è rappresentata dalla navigazione coordinata di swarm di droni finalizzata alla rapida raccolta di più informazioni rispetto alla semplice fotogrammetria: dal riconoscimento automatico/assistito di criticità al coordinamento posizionale per la copertura di edifici o aree di grandi dimensioni fino alla possibile esplorazione completamente autonoma di spazi ignoti. Considerando che molta parte del lavoro di raccolta e ricostruzione debba avvenire sul campo, è evidente da un lato l'importanza di svolgere parte delle operazioni sui droni stessi attraverso meccanismi di edge computing, dall'altro l'opportunità di sfruttare meccanismi di calcolo distribuito per sfruttare al meglio tutti i dispositivi interconnessi. Tali funzionalità devono tenere conto delle problematiche legate alla mobilità dei droni (connettività intermittente) ed alla limitata autonomia delle batterie. In letteratura sono presenti due approcci di coordinamento dello swarm: coordinamento centralizzato e coordinamento distribuito basato su approcci bio-inspired.

Coordinamento centralizzato

Questa tipologia implementa il controllo dello swarm mediante la presenza di un nodo coordinatore a terra o di un drone leader che gestisce lo stato globale della rete. Per fare ciò, molti studi recenti sfruttano il paradigma del Software Defined Networking (SDN) tramite la piattaforma OpenFlow, al fine di centralizzare il controllo della rete attraverso un software di controllo che controlli dinamicamente lo stato dei droni, quali la mobilità ed il routing[LT18]. Un esempio di implementazione sfrutta l'algoritmo di Particle Swarm Optimization: ad ogni iterazione si identifica il "miglior candidato" in termini di Quality of Service (Qos) ad effettuare un task.

Coordinamento bio-inspired distribuito

Le soluzioni di questo tipo imitano il comportamento di colonie animali nelle quali il coordinamento globale si ottiene come proprietà emergente da comunicazioni locali[YZYL17]. Ad esempio, sono stati proposti studi basati su Ant Colony Optimization (ACO) per l'ottimizzazione della copertura; prima di tutto si procede con il suddividere l'area in celle, per poi individuare i percorsi validi. Successivamente, simulando il comportamento delle formiche, si vanno ad individuare i percorsi con più quantità di feromoni digitali presenti ottimizzando il processo di path-finding. Un ulteriore miglioramento può essere raggiunto grazie agli algoritmi genetici, trattando come cromosomi le sequenze di celle che compongono i percorsi validi.

2.1.4 Problematiche

I problemi nell'utilizzo degli UAV derivano molto spesso dall'autonomia limitata delle batterie, dovute anche dal carico eccessivo degli elementi installati onboard e dall'acquisizione ed elaborazione dati. In letteratura sono presenti molteplici simulazioni di controllo UAV anche in contesti swarm, dando comunque ottimi risultati. Tali studi si scontrano tuttavia con le attuali normative sul pilotaggio automatico di UAV che non permette la messa

in volo se non con una supervisione attiva da parte di un utente a terra, rendendo il concetto di "automatico" limitato al campo visivo del soggetto.

2.2 Comunicazione UAV e Wireless Sensor Networks

Uno dei più grandi problemi nello sviluppo di UAV-aided Wireless Sensor network è il modo con cui i device IoT possano essere attivati; i dati raccolti dagli UAV sono poi trasmessi ad un centro di raccolta/processamento, generalmente sfruttando la rete cellulare. In base all'applicazione, network e UAV è necessario individuare la tecnologia di comunicazione più atta agli scopi. In letteratura[HTA16], sono stati proposti studi sperimentali di comunicazione UAV-sensore, basati su due tipologie di accesso: comunicazioni a corto e lungo raggio.

- Le **comunicazioni a lungo raggio** forniscono una copertura estesa come quella cellulare (es. LTE), broadband (es. Wimax), e SATCOM. Questa tipologia viene utilizzata per la comunicazione con le GCS per il continuo scambio di dati e aggiornamento dello stato.

- Le **comunicazioni a corto raggio**, come Bluetooth e Zigbee, trovano impiego per le comunicazioni nei pressi delle aree d'interesse come: Drone-to-Drone, Drone-to-IoTDevice, IoTDevice-to-IoTDevice

La comunicazione deve soddisfare una serie di requisiti imposti dalla tipologia di applicazione. Partendo dalla limitata durata della sessione di volo del drone per via del rapporto peso-potenza, la comunicazione deve essere veloce e minimale in modo tale da ridurne il tempo finale di trasmissione. Altri requisiti invece possono derivare direttamente dal sensore a terra, poiché potrebbero avere un'autonomia residua bassa per via della mancanza della rete elettrica.

Il continuo progredire delle tecnologie di telecomunicazione cellulare rendono possibile il controllo degli UAV anche ad altitudini elevate e con distanze

considerevoli. Sistemi 4g LTE e l'ormai 5G permettono una comunicazione affidabile mobile per gli UAV, i quali richiedono una comunicazione flessibile, veloce e appunto affidabile. Un nuovo business nato dalla combinazione UAV-cellular connection, è quello di utilizzare gli UAV come ponti radio per aumentare la copertura di aree a terra. Un esempio è costituito dal progetto SkyBender di Google, che utilizza i droni per fornire connettività wireless 5G circa 40 volte più veloce rispetto ai classici sistemi a 4g a terra. Nelle prossime sezioni, verranno illustrate le varie tecnologie di comunicazioni a corto e lungo raggio evidenziandone i vantaggi e i limiti nell'ambito UAV-IoT.

2.2.1 Sistemi cellulari

Il recente sviluppo del controllo remoto dei velivoli attraverso sistemi wireless abilita il pilotaggio degli UAV anche oltre la Line Of Sight. Decisamente una delle tecnologie che lo rende possibile è l'utilizzo della rete cellulare. In queste reti, ogni cella ha una propria sede di trasmissione(comprensiva di antenna), chiamata anche Base Station (BS). Tali celle forniscono copertura radio attorno ad esse. Un qualsiasi ricevitore dello stesso tipo nell'area interessata può comunicare attraverso la rete, anche in movimento. L'architettura cellulare presenta molti vantaggi, uno tra essi è quello di poter estendere il raggio d'azione combinando altre BS vicine. Molteplici BS forniscono una ridondanza naturale tale da garantire in caso di scarsa connessione, un rimpiazzo con una migliore.
Alcune delle tecnologie cellulari più conosciute sono: Global System for Mobile Communication (GSM), LTE, 4G. Nell'articolo[Gue+15] viene descritto uno scenario di emergenza in cui gli UAV, in caso di mancanza della rete cellulare, vanno a ricreare una rete cellulare sostitutiva. In questo esperimento[GDW14] si analizza l'uso degli UAV come ripetitori 3G per aumentare la qualità del segnale in zone remote. I risultati non solo hanno dimostrato l'effettivo aumento di copertura della rete, ma anche in zone urbane ben servite ci è stato un grosso miglioramento.

2.2.2 WiMAX 802.16

WiMAX è una tecnologia wireless creata con lo scopo di trasmettere banda larga in aree molto grandi; rispetto al WiFi ha un costo di deployment più basso e raggiunge distanze nettamente più lunghe. Il data rate per lo standard raggiunge i 75 Mb/s (mediamente 20/30 Mb/s) per utente, mentre per usi mobile raggiunge i 30Mb/s (dai 3 ai 5Mb/s in media). WiMAX è stato sviluppato per gestire dati voce ad alta qualità e video stream rimanendo comunque con una elevata Quality of Service (QoS).
I vantaggi nell'utilizzo di questa tecnologia sono:

1. **Flessibile**: supporta sistemi point-to-point e mesh.
2. **Differenziazione del servizio**: può applicare differenti metodi di gestione basato sul tipo di traffico.
3. **Altamente sicuro**: implementa molti metodi di crittografia, autenticazione e sicurezza.
4. **Alto throughput**.
5. **Grande copertura**.
6. **Mobile**: supporta connessioni fino a 120km/h.
7. **Installazione semplice e costi bassi**.

Rahman[Rah14] ne considera l'utilizzo in sistemi di emergenza per il salvataggio di vite sulle Alpi. Questo perché si adatta perfettamente alle condizioni ostili presenti in quegli ambienti. Lo studio di questo articolo[DW10] va a determinare quanto WiMAX sia performante per il controllo degli UAV. L'analisi prevede varie condizioni del canale di trasmissione in termini di throughput, round trip time (RTT), e packet loss per differenti modulazioni e schema di codifica. Dai risultati si evince come WiMAX abbia un RTT molto basso insieme ad un basso packet loss anche con il canale altamente disturbato.

2.2.3 Comunicazioni Satellitari

Nel momento in cui gli UAV si muovono al di fuori del range di una connessione diretta per via della distanza oppure ostacoli ambientali, le SATCOM (comunicazioni satellitari) vengono utilizzate per le comunicazioni oltre la LoS (BLoS)[Ski14]. Oltretutto le comunicazioni sono sempre presenti durante il controllo degli UAV, per via della necessità di dover sempre sapere la posizione attuale del velivolo e la connessione di emergenza. Le criticità di tale tecnologie sono facilmente individuabili:

1. larghezza di banda limitata
2. un alto costo per la trasmissione dati

2.2.4 WiFi (IEEE 802.11)

L'802.11 permette la trasmissione a 2.4, 3.6, 5 e 60GHz come bande di frequenza. Come già anticipato in precedenza, rimane la tecnologia più diffusa per comunicare tra station-node wireless o node-node. Nello studio [S F15] viene descritto lo sviluppo di un sistema Wi-Fi installato su degli UAV estendendo il raggio standard di 100m fino a 25km. Viene concentrata la potenza del segnale in una direzione utilizzando antenne direzionate. Inoltre, è presentato un sistema di controllo utilizzato per assicurare la stabilità della connessione. Nello studio descritto nell'elaborato [CHKV06] viene fatta notare la necessità di applicare una configurazione delle antenne omnidirezionali tali da essere posizionate orizzontalmente; i risultati dei test evidenziano riscontri positivi sia in fase di connessione sia di throughput.

2.2.5 Bluetooth (IEEE 802.15)

Il Bluetooth è uno standard di comunicazione 2.4 Ghz a basso consumo. Esistono tre versioni di questa tecnologia e variano sostanzialmente nel rate di trasmissione 1-3-24 Mb/s. Nello studio [CSFSA09] viene progettato e sviluppato un prototipo di UAV in cui tutta la sensoristica e il controllo vengono

gestiti da un sistema Bluetooth. Il lavoro ha dimostrato che per la connessione simultanea dai 6 slave (sensori) a master, il sistema non presenta evidenti cedimenti in termini di throughput, perdita o ritardi. Invece per sistemi con limitate capacità di calcolo, viene proposta[Coe+06] un'implementazione di una rete a corto raggio Bluetooth con accesso pianificato tramite algoritmo round robin. Il risultato di questo lavoro dimostra che l'algoritmo applicato ad una struttura con un master e 7 slave (controllo e sensori) è efficiente per situazioni con limitate capacità computazionali e ne permette al contempo una comunicazione affidabile.

2.2.6 Zigbee (IEEE 802.15.4)

Questa tecnologia supporta protocolli di comunicazione ad alto livello, rimanendo comunque nella tipologia low-power e con costi bassi. Gli utilizzi più frequenti ricadono in applicazioni che richiedono una lunga durata della batteria e connessioni sicure. Un altro dei vantaggi della Zigbee è di essere meno soggetto ad interferenze, per via del suo canale low-rate 20/40bk/s a 868/925 GHz con la possibilità di arrivare a 250kb/s nella banda dei 2.4GHz. È particolarmente indicata per lo scambio di dati intermittente da sensore o qualsiasi altro device input. In ambito UAV, viene utilizzata per la comunicazione e per stimare la posizione del velivolo. Nello studio [YJY06] viene discusso come l'errore nel calcolo della posizione sia decisamente inferiore rispetto ad altri sistemi. Inoltre, viene presentato poi un sistema di localizzazione indoor [YFG13]. Lo sviluppo è la combinazione tra Zigbee e il sistema di navigazione, dove anemometro e accelerometro sono usati per incrementarne l'accuratezza; applicando un modello esteso del filtro di Kalman[Eve03], viene migliorata l'accuratezza del sistema di posizionamento.

Parte II

Sperimentazione

Capitolo 3

Progettazione

Nella precedente parte, lo stato dell'arte, sono state elencate le potenzialità nell'utilizzo degli UAV in ambito IoT e nello specifico nella comunicazione con una WSN. Al tempo stesso pochi studi sperimentali sono stati proposti relativi all'integrazione drone-sensore. Lo scopo di questa tesi punta a dimostrare la fattibilità del volo autonomo di un UAV Low Cost e la sincronizzazione per la comunicazione a terra con una rete di sensori.

3.1 Obiettivi

Il progetto di sperimentazione si focalizza sulla creazione di una piattaforma di controllo e comunicazione dell'UAV, chiamato Sybelius. Verranno di seguito descritti i macro-obiettivi di questa tesi.

3.1.1 Controllo autonomo di un UAV

La necessità maggiore riscontrata in letteratura è quella di dover trovare un modo per cui l'UAV non debba dipendere costantemente dalla stazione a terra per poter portare a termine un determinato task. Possiamo definire come una "missione" fonita dalla stazione la serie di macro-task componenti, quali: path-planning, coordinamento con altri UAV, acquisizione dati tramite sensori onboard o WSN. Il controllo autonomo deve tener conto dei vari stati

dei task e soprattutto dello stato dello UAV, ad esempio prevedere sistemi di emergenza in caso di batteria scarica, fail di sistema, perdita di comunicazione con un sensore a bordo. Il sistema descritto in questo elaborato punta a rendere autonomo un UAV low-cost. Definita una missione di volo, l'UAV sarà in grado di raggiungere determinate coordinate GPS, attivare i propri sensori in base agli eventi, scaricare dati ed effettuare una fase di pre-analisi.

Lo scopo finale è quello di valutare quanto il controllo sia effettivamente affidabile rispetto alla missione di partenza, tenendo conto dei vari puntamenti GPS e della capacità di effettuare operazioni di hovering a differenze altezze, coordinandosi con il sistema di comunicazione.

3.1.2 Comunicazione con WSN

Il secondo obiettivo prende in considerazione le varie tecniche di comunicazione con una rete di sensori a terra. Definito il sistema di controllo, esso deve collaborare con il sistema comunicazione in modo tale da agevolare la trasmissione anche per lunghe sessioni di volo stazionario e non. Nella missione vengono indicati i punti GPS per cui vi è la necessità di andare a comunicare con un sensore a terra, definendo anche la tecnologia da impiegare durante l'operazione (BLE, Zigbee, WiFi, etc). Il sistema di controllo farà rimanere l'UAV nella coordinata GPS del sensore, per un tempo tale da permetterne una comunicazione completa. Il software prevede che la comunicazione sia completamente isolata dal resto del sistema, per poi passare eventualmente i dati al coordinatore dell'UAV.

L'obiettivo finale è quello di testare i limiti di comunicazione sotto aspetti di potenza, latenza, e traffico con differenti agenti atmosferici e diverse altitudini rispetto al sensore.

3.2 Premesse

Prima di procedere con la presentazione dell'architettura, vanno fatte delle dovute premesse. Il sistema non è un simulatore ma è un vero software installabile su hardware compatibile per cui è stato implementato. L'architettura iniziale è stata raffinata e migliorata proprio grazie ai continui test sul campo. Dopo una prima fase di analisi progettuale, il sistema è stato sviluppato a partire dalle problematiche hardware e software riscontrate dall'aggiunta di ogni nuova funzionalità.

3.3 Architettura

L'architettura del software è ispirata dal framework Nostromo, un sistema di controllo remoto per unmanned ground vehicle (UGV). Sybelius riprende da Nostromo le caratteristiche di modularità e retrocompatibilità.
Nella figura 3.1 possiamo individuare la struttura del Coordinatore: **Controllo Missione** riceve la Missione o Routine programmando poi a livello temporale i vari task smistandoli in base al tipo verso il **Sistema di Controllo** oppure al **Manager di Comunicazione**. Si noti come ci sia una continua notifica di eventi tra reparto di controllo e quello di comunicazione. Il controllo missione tiene sempre aggiornato lo stato dell' UAV, ricevendo con un'alta frequenza messaggi di tipo Navigazione dal sistema di controllo.

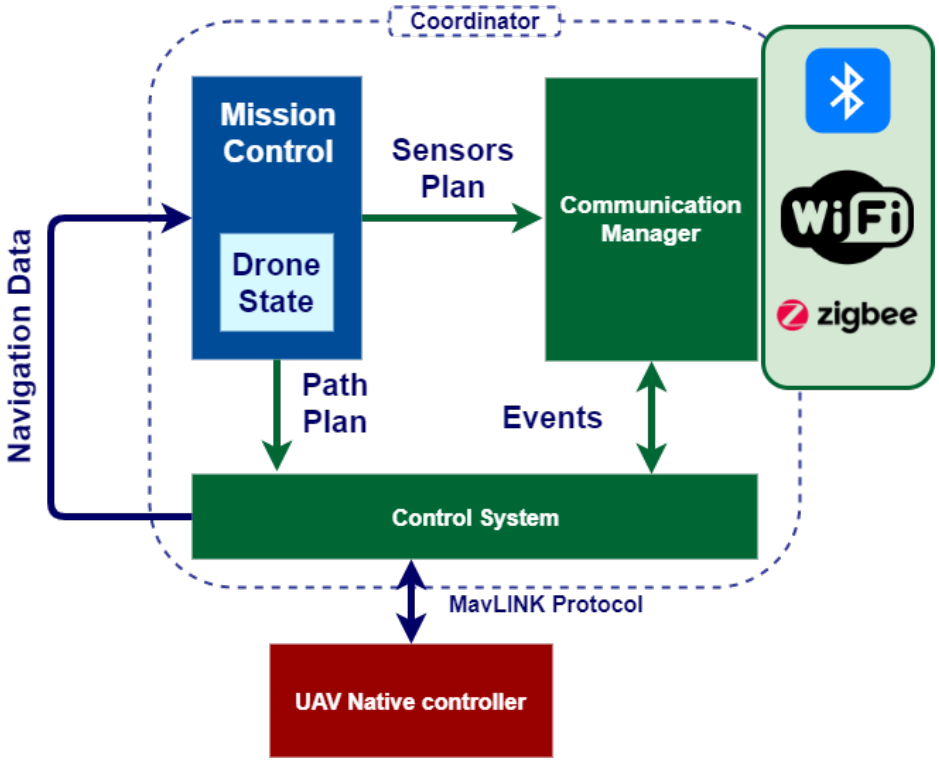

Figura 3.1: : Architettura di Sybelius

3.3.1 Mission Control

Il controllo missione si occupa di caricare la routine presa in consegna dalla GS. Per prima cosa viene valutata la fattibilità della routine, controllando che le finestre temporali siano adeguate per i task indicati; dopodiché, viene lanciato un task fittizio per inizializzare i due componenti controllo e comunicazione. A questo punto il Mission control resta in attesa di un quantitativo minimo di Navigation Data tale da poter fare un controllo iniziale sullo stato della batteria, posizione gps, e motori. Solo successivamente vengono inviati i vari task. Il Mission Control ha la possibilità di avviare task paralleli e di poterli arrestare in qualsiasi momento, come indicato dalla routine. Ad esempio uno dei task più frequenti è quello di orientare l'UAV verso una coordinata GPS, ma una volta raggiunta la direzione desiderata, il task viene ritenuto completato; è possibile anche parallelizzarlo in un-sottoprocesso,

così da avere sempre il puntamento attivo. Il Mission control si prende carico di tenere aggiornato lo stato dell'UAV durante tutta la sessione di volo; ha la capacità di avviare task di emergenza in caso di anomalie in fase di lettura dei dati, batteria scarica, comunicazione intermittente e altro. Ad esempio, può decidere di far atterrare l'UAV in caso di batteria scarica o ancora più semplicemente di non avviare neanche la sessione di volo. Se viene richiamato il comando di atterraggio durante un'altezza troppo elevata, il task viene anticipato da un task di abbassamento del drone e solo successivamente task di atterraggio.

Stato dello UAV

Lo stato viene tenuto aggiornato ricevendo tramite protocollo MavLINK un messaggio di navigazione da parte del UAV Native Controller chiamato Navigation Data; questo messaggio viene persistito dal Mission Control e successivamente viene filtrato in modo tale da renderlo leggero e compatto. A questo punto viene sempre mandato in broadcast ai vari reparti poiché i task di cui hanno preso carico, necessitano molto spesso dello stato generale dell'UAV. Il Navigation data filtrato contiene la posizione GPS (Latitudine, Longitudine, Altitudine), Altezza da terra, Livello Batteria, e dati Magnetometro (Direzione attuale, calcolata, stato calibrazione).

3.3.2 Control System

Qualsiasi task che comporti il movimento o l'attivazione di un sensore onboard viene inviato al sistema di controllo. I task possono essere di due tipologie:

- **Primitive Command**
  Con comando primitivo s'intende un'azione specifica che il drone può compiere già nativamente, e sono:
  - muoversi nelle varie direzioni

- atterraggio/decollo
- calibrazione
- attivazione delle telecamere
- attivazione GPS
- attivazione Magnetometro

I comandi sono di tipo on/off, ovvero vengono terminati in caso di comando primitivo apposto come ad esempio "vai avanti, attendi 5 secondi, vai indietro". Nelle varie routine vengono utilizzati questi task tra i super-task di controllo, cosi da definire meglio, ad esempio, la fase di decollo in caso di forti venti.

- **Super command**
  I super comandi o super-task di controllo servono per eseguire una serie di primitive-task in base a dei parametri di partenza, solitamente in modo ciclico fino alla portata a termine del super-task o fino ad un comando esplicito di stop in caso di parallelizzazione. In Sybelius sono stati definiti ed implementati i seguenti super-task:
  - Spegnimento: si assicura di essere ad altezza pari a 0m, spegne i motori e lo comunica al controllo missione.
  - Blocco task parallelo, utilizzato molto spesso tra i vari super task e dal controllo missione dopo determinate condizioni o eventi. Il Manager di Comunicazione ha accesso a questa funzionalità creando un evento apposito.
  - Attiva la fotocamera e richiede un fotogramma ogni N secondi.
  - Mantiene una determinata altitudine in metri rispetto ad un limite minimo e massimo.
  - Puntamento direzionale verso una coordinata GPS
  - Accorcia le distanze rispetto ad una coordinata Gps: si assicura di avere un task parallelo di puntamento direzionale, altrimenti viene lanciato un errore.

UAV Native Controller

Si tratta del sistema di controllo nativo dell'UAV. Sybelius viene eseguito dall'UAV ma non si sostituisce al controllo originale, semplicemente lo sfrutta utilizzando il protocollo MavLINK. Vengono mappati tutti comandi base del velivolo e vengono acceduti i Navigation Data in modo costante. Sebbene l'architettura supporti qualsiasi tipologia di controller Nativo, è stato sviluppato solamente per il controller del drone di testing.

3.3.3 Communication Manager

Nel momento in cui il Mission Control rileva un task di tipo comunicazione, quest'ultimo viene mandato al relativo Manager. In base alla tecnologia e alle informazioni di handshake per la trasmissione, il modulo avvia un sotto-processo dedicato. Viste le possibili difficoltà in fase di comunicazione con sensori a terra, il processo parallelo agevola la sessione di volo evitando di sovraccaricare il controllo missione con uno streaming continuo di messaggi di errore generici, di connessione o trasmissione. Sono state previste varie tipologie di trasmissione a terra, come WiFi Zigbee e BLE, ma soltanto quest'ultima è stata utilizzata nei test descritti nei Capitoli sucessivi.

Capitolo 4

Implementazione

In questo capitolo verranno illustrate le varie tecnologie software e hardware utilizzate per il prototipo finale. Successivamente verrà presentato parte del codice di Sybelius, concludendo con l'UAV finale sviluppato durante l'anno di sperimentazione.

4.1 Tecnologie: Software e Hardware

Il software è un applicazione NodeJs[1] implementata e testata sia su sistemi Windows che Linux. La scelta deriva dall'esperienza personale maturata in ambito IoT e soprattutto per la gran scelta di librerie create dalla community. Di seguito vengono elencate le varie tecnologie software:

- **TypeScript**[2]: super-set JavaScript, introduce molti costrutti tipo: classi, enum, tipi, interfacce, moduli e altro rendendo più semplice il debugging durante la fase di sviluppo. É stato usato come linguaggio principale dell'applicazione.

- **JavaScript**: per la sua velocità di esecuzione e compatibilità con TypeScript, fondamentale per la parte di comunicazione.

[1] https://nodejs.org/en/docs/
[2] https://www.typescriptlang.org/

- **@hyperloris/tyson**[3]: utilizzato per il filtraggio dei messaggi Navigation Data, converte un json in un oggetto TypeScript partendo da un modello. Viene anche utilizzato per la lettura e la conversione delle Routine.

- **cron**: come suggerisce il nome, implementa esattamente la funzione cron dei sistemi Linux, fondamentale per parallelizzare funzioni in NodeJs

- **geolib**: questa libreria mette a disposizione una serie di funzioni per il calcolo delle distanze tra coordinate geografiche, offset dal nord e offset rispetto la propria posizione e una di arrivo.

- **winston**: si occupa di gestire lo spam della console e di andare a persistere i dati provenienti dai vari componenti software in log formato json.

- **ar-drone**[4]: utilizzato per lo sviluppo del sistema di controllo per una tipologia di drone, AR DRONE 2.0, implementa l'sdk della parrot sotto forma di libreria node con la gestione del protocollo MavLINK.

- **Noble**[5]: libreria NodeJs in grado di gestire il Bluetooth Low Energy di varie piattaforme sia come device centrale che come periferica.

Come componenti hardware sono stati utilizzati parte del progetto Nostromo, del Laboratorio di Informatica del Disi, e personali. Nello specifico possiamo elencare:

- **PowerBank 3000mA**: utilizzato per alimentare la board del coordinator

- **PowerBank 13000mA**: per l'alimentazione sensore a terra

[3]https://github.com/hyperloris/tyson
[4]https://www.npmjs.com/package/ar-drone
[5]https://github.com/noble/noble#readme

- **ESP-32**: microcontrollore utilizzato come nodo-sensore a terra, utilizza la libreria noble.
- **Raspberry PI 3 model B**: è la board utilizzata come coordinatore nella maggior parte dello sviluppo del prototipo.
- **Raspberry PI Zero W**: per via dei bassi consumi, è diventata la board del prototipo, ma soltanto con routine poco impegnative per via della bassa capacità di calcolo.
- **Modulo WiFi USB**: connessione extra per la comunicazione con la GS.
- **Ar Drone 2.0**: UAV domestico in grado di volare indoor (con protezioni) e outdoor via WiFi.
- **Convertitore 12v to 5v USB**: usato per alimentare la board direttamente con le batteria del drone.

4.2 Sybelius

Riprendendo l'architettura illustrata nel precedente capitolo, possiamo dividedere per reparti il codice andando ad individuare le 3 macro-classi corrispondenti:

- RoutineManager è la classe principale del Mission Control
- Drone è la classe del Sistema di Controllo
- Il Communication Manager al momento supporta solo la tecnologia BLE, la classe di riferimento è bleOperation.js

4.2.1 Mission Control: RoutineManager

Questo singleton gestisce tutto il piano di volo, gli eventi, setup e stato. La fase di setup inizia dal main file, index.ts, in cui viene subito avviato il

sistema di controllo per poi istanziare il RoutineManager. A questo punto inizia il setup interno della classe che consiste nel caricare ed avviare la routine letta da file.

```
1     public loadFromFile(): Routine {
2     let tyson = new TysonBuilder()
3       .registerTypeAdapter(Ignore, new IgnoreAdapter())
4       .build();
5     this.routine = tyson.fromJson(data, Routine);
6     return this.routine;
7   }
```

Listing 4.1: Caricamento delle Routine

Una **Routine** ha le informazioni generali di una missione come "titolo", "owner" e "description" ma la parte più importante è l'array di Actions. La classe dispone della seguente struttura:

```
1  export class Routine {
2    @JsonProperty()
3    title: string = undefined;
4    @JsonProperty()
5    owner: string = undefined;
6    @JsonProperty()
7    description: string = undefined;
8    @JsonProperty({ name: 'actions', type: [Action] })
9    actions: Action[] = undefined;
10
11   constructor() {}
12 }
```

Listing 4.2: Oggetto Routine

Una **Action** è un task descritto nel .json della routine. La struttura di questo oggetto rimane molto generica, ed accoglie al suo interno il nome dell'action, una descrizione opzionale, il delay dal precedente task, la velocità (campo molto usato dai comandi primitivi di controllo, e quindi obbligatorio ed esplicitato) e dei parametri aggiuntivi opzionali nella forma "nome-valore".

```
export class Action {
  @JsonProperty()
  name: string = undefined;

  @JsonProperty()
  description: string = undefined;

  @JsonProperty()
  delay: number = undefined;

  @JsonProperty()
  speed: number = undefined;

  @JsonProperty({ name: 'params', type: [Param] })
  params: Param[] = undefined;

  constructor() {}
}
```

Listing 4.3: Oggetto Action

Le possibili Action sono state definite con un Enum, e il json della routine deve necessariamente rispettare questa sintassi.

Nome Action	descrizione	speed	parametri
takeoff	decollo	-	
land	atterraggio	-	
stop	ferma tutte le azioni di movimento correnti mette in stato di hovering	-	
up	movimento verticale verso l'alto	Si	"max_ltitude": metri esegue il comando fino all'altezza indicata
down	movimento verticale verso il basso	Si	"time": millisec viene richiamato "land" dopo il tempo indicato
clockwise	movimento sul posto in senso orario	Si	
counterClockwise	movimento sul posto in senso antiorario	Si	
front	movimento orizzontale in avanti	Si	
back	movimento orizzontale indietro	Si	
left	movimento laterale a sinistra	Si	
right	movimento laterale a destra	Si	
turnToWaypoint	puntamento direzionale verso una coordinata Gps di Default è parallelizzata.	-	"latitude": num, "longitude": num, "altitude": num va indicata una coordinata Gps
goToWaypoint	riduce la distanza tra la posizione attuale e la coordinata Gps si assicura che ci sia "turnToWaypoint" attivo di Default è parallelizzata	-	"latitude": num, "longitude": num, "altitude": num va indicata una coordinata Gps opzionale: "takePicture": freq frequenza del takePicture

Tabella 4.1: : Lista delle Action, parte 1

altitude	mantiene l'altezza fissata tra due limiti di Default è parallelizzata	Si	"max_ltitude": metri, "min_ltitude": metri limite massimo e limite minimo devono essere presenti
takePicture	viene scattata una foto con la camera inferiore e salvata nella cartella /images nella root di sybelius		
calibrate	esegue la calibrazione con un movimento 360° veloce		
turnOnBle	esegue la comunicazione in BLE con un sensore a terra		"packets": num, "frequency": millisec numero pacchetti e frequenza di invio richiesti
shutdown	termina il processo principale controllando lo stato batteria avviene soltanto se il drone è in stato di "land"		
kill	termina un task parallelo		"action": nome_ction viene lanciato un errore sia che l'action non esista, o che non sia in running

Tabella 4.2: : Lista delle Action, parte 2

GpsManager

Classe singleton che gestisce la posizione corrente del drone. Viene continuamente aggiornato con i dati GPS provenienti dai messaggi Navigation Data. Ha un ruolo fondamentale poiché implementa due funzionalità core di sybelius:

- **offsetDistanceFromWaypoint(origin, destination)**: questa funzione calcola la distanza in metri tra la posizione corrente(se non viene fornito un altro punto di origine) e un punto di destinazione
- **offsetOrientationFromWaypoint(origin, destination)**: questa funzione calcola l'angolo di differenza di orientamento rispetto al nord tra due coordinate gps.

Compass

Gestisce l'orientamento del drone e converte il dato proveniente dal magnetometro. La conversione avviene nel costruttore e nel set dell'orientamento (Bearing). Consiste nel mantenere il dato in una scala +180° e -180; per via di anomalie del sensore non è raro che si riceva un dato 0-360° anche dopo la calibrazione.

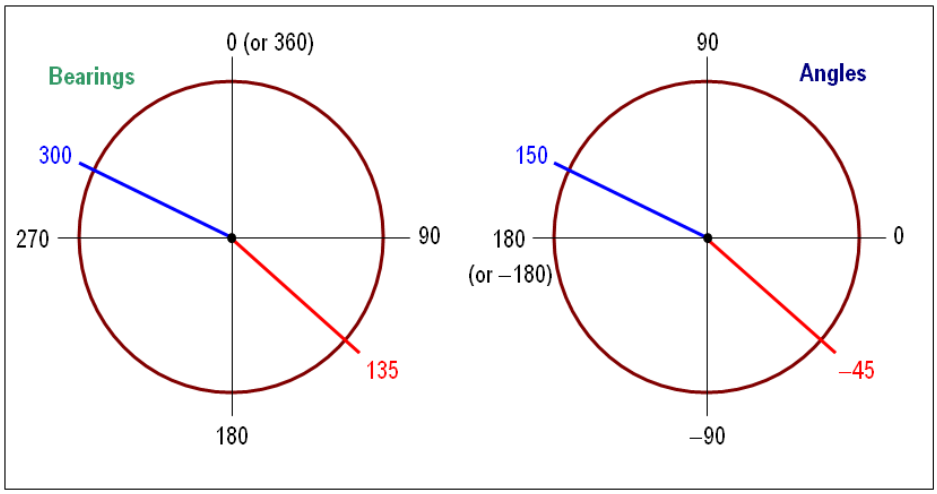

Figura 4.1: : Orientamento o Bearing nelle due possibili scale

La funzione principale è quella di assistere il sistema di controllo nel capire la direzione da prendere per puntare una coordinata GPS.

```
public get nearestDirection(): DirectionList {
    let normalizedBearing = this.bearing;
    let direction = DirectionList.CLOCKWISE;
    if (normalizedBearing > this._bearingToReach) {
      if (normalizedBearing - this._bearingToReach > 180) {
        direction = DirectionList.CLOCKWISE;
      } else {
        direction = DirectionList.COUNTERCLOCKWISE;
      }
    } else {
      if (this._bearingToReach - normalizedBearing > 180) {
        direction = DirectionList.COUNTERCLOCKWISE;
      } else {
        direction = DirectionList.CLOCKWISE;
      }
    }
```

```
17      return direction;
18    }
```

Listing 4.4: funzione dell'oggetto Compass che indica la direzione più breve rispetto al puntamento attuale

Il task di puntamento usa questa funzione per evitare di fare movimenti maggiori di 180°.

4.2.2 Control System

Il sistema di controllo come già anticipato, mappa i comandi base del controller nativo dell'UAV in questione. Sono state definite poi le super-task come il puntamento automatico e il "goToWaypoint". Una delle operazioni è quella di setup, in cui vengono avviati i vari sensori e viene notificata la possibilità di ricevere i messaggi di navigazione e vien esposto l'Engine, la libreria che manda effettivamente tutti i comandi al controller nativo

```
1  private navdata_option_mask(c) {
2      return 1 << c;
3    }
4    public enableMagnetometer() {
5      var default_navdata_options =
6        this.navdata_option_mask(this.constants.options.DEMO) |
7        this.navdata_option_mask(
8          this.constants.options.VISION_DETECT);
9      this.droneClient.config(
10       'general:navdata_options',
11       default_navdata_options |
12       this.navdata_option_mask(this.constants.options.MAGNETO)
13     );
14     this.droneClient.config(
15       'general:navdata_options',//Enable the magnetometer data.
```

```
16        this.navdata_option_mask(this.constants.options.MAGNETO)
17      );
18    }
19    public enableGPS() {
20      this.droneClient.config('general:navdata_options',
21      777060865);
22    }
23
24    public enableNavdata() {
25      this.droneClient.config('general:navdata_demo', 'FALSE');
26    }
27    public enableBottomCamera() {
28      this.droneClient.config('video:video_channel', 3);
29    }
30
31    public init(): boolean {
32      this.enableGPS();
33      this.enableMagnetometer();
34      this.enableNavdata();
35      this.enableBottomCamera();
36      return true;
37    }
38
```

Listing 4.5: fase di setup dell'engine del drone

Si noti come i vari sensori vengano avviati tramite formato MavLINK; queste attivazioni di sensori devono necessariamente avvenire prima della fase di decollo, in caso contrario non sarà possibile effettuare questa operazione per via di una protezione interna del controller.

Caso Specifico: Puntamento automatico

Per comprendere meglio come il sistema di controllo possa effettuare i suoi task, verrà spiegata in questa sezione una parte del super-task del puntamento automatico.

1. Il sistema di controllo riceve il task “turnToWaypoint” memorizzando subito la coordinata GPS di destinazione;
2. viene preso l’oggetto che gestisce il cronJob del puntamento e viene reinizializzato;
3. ad ogni tick del cronJob, viene calcolato il bearing attuale e quello da raggiungere utilizzando il GpsManager e viene nstanziato un Compass;
4. vengono mandati messaggi di debug a console e scritti su delle apposite log pre-formattate con la librearia winston, in modo da permetterà un’analisi post sessione volo.
5. viene utilizzato l’oggetto Compass per determinare la direzione più vicina da intraprendere, e viene quindi eseguito un piccolo scatto utilizzando un comando primitivo tra clockwise e counterClockwise. Viene lasciato un valore di scarto di sicurezza pari a 5° gradi per cui non vengono effettuati scatti.
6. l’iterazione finisce e riparte dal punto 3

```
1                 if (compass.bearingGap < 5) {
2                   logger.verbose('little bearing gap,
3                   skipping turn action');
4                 } else {
5                   if (!doingAnSubAction) {
6                     doingAnSubAction = true;
7
8                     switch (compass.nearestDirection) {
```

```
9                       case DirectionList.CLOCKWISE:
10                        drone.clockwise(0.3);
11                        setTimeout(() => {
12                          drone.clockwise(0);
13                          doingAnSubAction = false;
14                        }, 500);
15                        break;
16
17                      case DirectionList.COUNTERCLOCKWISE:
18                        drone.counterClockwise(0.3);
19                        setTimeout(() => {
20                          drone.counterClockwise(0);
21                          doingAnSubAction = false;
22                        }, 500);
23
24                        break;
25                      default:
26                        break;
27                    }
28                  } else {
29                    logger.debug(
30                      'doing another operation,
31                      skipping this one (turning)',
32                      { label: 'TURNTOWAYPOINT' }
33                    );
34                  }
35                }
```

Listing 4.6: Parte del codice di puntamento automatico

4.2.3 Communication Manager

Il manager di comunicazione come già anticipato, implementa il BLE come tecnologia di trasmissione con il sensore a terra di test. Una volta ricevuto il task di avvio, viene fatto partire un sotto-processo grazie alla funzione fork[6] di Node. A questo punto tutta la comunicazione avviene per messaggi non tipizzati tra processi ricevendo e loggando solamente i dati significativi dal sotto-processo. Quindi la comunicazione si sposta in un modulo JavaScript ottenendo i seguenti vantaggi:

- Il sotto-processo non va ad interferire il processo principale, in caso di errore viene fermato e rilanciato.
- utilizza la libreria Noble precedentemente citata, perfettamente testata e supportata dalla community ma soltanto in formato JavaScript.
- Il modulo scritto in Javascript non viene compilato come il resto del progetto. La fase di debug del BLE è stata agevolata da questo aspetto, non dovendo ogni volta ricompilare il progetto per intero.

Nel prossimo listato di codice possiamo vedere come il file bleOperation faccia da intermediario tra il Main process e le operazioni BLE.

```
process.on('message', (message) => {
  logger.verbose('Received command ', {label: 'MESSAGE'});
  var string = String(message);
  var req = JSON.parse(string);
  switch (req['command']) {

  .........

  // GETDATA command: force a sensor reading
    case 'getdata':
```

[6] https://tinyurl.com/Node-child-process

```
11      (async () => {
12        try {
13          if (STATISTICS_ON) {
14            var key = 'ble';
15            if (!(key in STATISTICS)) {
16              STATISTICS[key] = 0;
17            }
18            STATISTICS[key] = performance.now();
19          }
20          await ble.forceGetData();
21        } catch (err) {
22          var answerString = { 'type': 'result',
23          'operation': 'getdata', 'value': null };
24          var answerJSON = JSON.stringify(answerString);
25          process.send(answerJSON);
26        }
27      })();
28      break
```

Listing 4.7: Parte del codice di puntamento automatico

Viene inizialmente ricevuto un messaggio dal main process con il comando "getdata", poi viene eseguito un forceGetData, implementato per scrivere la caratteristica corrispondente tramite BLE. Dopodiché, avviene il percorso inverso in questo modo:

1. Il sottomodulo del BLE esegue una lettura con una frequenza preimpostata, appositamente rimossa per avere quasi in tempo effettivo la risposta dal sensore

2. il BLE notifica tramite evento il nuovo dato a bleOperation;

3. ricevuto il valore richiesto, viene preparata una stringa più strutturata tipo JSPM e rimbalzata tramite messaggio tra processi al main thread.

4. ricevuto il messaggio, viene appositamente persistito nella log e visualizzato a console.

Nel sotto-processo BLE e bleOperation vanno a persistere, in un file dedicato, tutti gli eventi, i cambi di stato e i messaggi che vengono scambiati. Il log si è reso particolarmente utile nelle fasi di analisi poiché il log del processo principale viene utilizzato praticamente da tutto il software.

```
ble.bleEmitter.on('getdata-notify', function (value) {
  logger.verbose('Send getdata reply, value ' + value,
  { label: 'DATA' });
  logger.debug(value, { label: 'getdata-notify' });
  var answerString = { 'type': 'result',
  'operation': 'getdata', 'value': value };
  var answerJSON = JSON.stringify(answerString);
  process.send(answerJSON);
})
```

Listing 4.8: ricezione dato dal sensore e mandato al main process

4.2.4 Navigation Data

I dati di navigazione vengono gestiti tramite evento *navdata* direttamente dall'engine di Drone. Il client riceve periodicamente con finestre al di sotto dei 5ms lo stato del drone come Angolazioni, altitudine, camera, velocità e altro.

Il Navigation Data o navdata come denominato dalla Parrott, viene inviato dal drone usando la porta UDP 5554. Le informazioni sono inviate in formato binario e vanno a creare dei blocchi di dati chiamati Options. Ogni opzione ha un header (2 byte) che identifica il tipo di informazione contenuto in essa, un blocco da 16 bit di interi, e altri tipi di informazioni in blocchi di 32 bit.

Header	**Drone**	**Sequence**	**Vision**	**Option 1**			...	**Checksum block**		
0x55667788	**state**	**number**	**flag**	id	size	data	...	cks id	size	cks data
32-bit	32-bit	32-bit	32-bit	16-bit	16-bit	...	...	16-bit	16-bit	32-bit
int.	int.	int.	int.	int.	int.	...	...	int.	int.	int.

Figura 4.2: : Struttura del pacchetto Navigation Data (navdata)

4.3 Il Prototipo di UAV: PhoeniX

Questa sezione è dedicata interamente alle fasi di prototipazione del drone utilizzato per i vari test sul campo.

Figura 4.3: : AR Drone 2.0 Elite edition con protezione da esterno e da interno in basso

Il modello del drone è prima di tutto L'AR-Drone 2.0 Elite Edition, un quadricottero di fascia bassa concepito per l'uso amatoriale. Le caratteristiche principali sono:

- Drone a 4 rotori applicati agli estremi di un telaio a croce; nel punto di unione è installata la batteria e tutto il resto dell'hardware. Coppie opposte si muovono nella stessa direzione orario e antiorario.
- L'UAV si muove grazie a motori brushless a 3 fasi controllati da un microcontrollore. Automaticamente viene identificato il tipo di motore collegato e viene tarato di conseguenza. Viene effettuato un controllo dei motori se sono in azione o fermi. Se una delle eliche colpisce un ostacolo, il controller rileva il blocco del motore e successivamente fermerà tutti i motori immediatamente.
- Le batterie standard sono da 1000mAh, 11.1V LiPo; durante il volo il voltaggio delle batteria si abbassa da 12.5V fino a 9V. Il drone converte questo range in percentuale da 100% a 0%; quando è in low-voltage, viene inviato un messaggio di emergenza all'utente e automaticamente atterra. A livelli critici, viene effettuato lo spegnimento del sistema in modo tale da evitare comportamenti inaspettati.
- AR-Drone è equipaggiato con 6 unità di misurazione del coefficiente di inerzia DOF, MIEMS-based. Permette di calcolare il rollio, la torsione e l'angolo di volo. Questa misurazione serve per automatizzare i micromovimenti di stabilizzazione.
- Dispone di un sensore ad ultrasuoni che fornisce l'altitudine per la stabilizzazione verticale.
- La camera inferiore rivolta verso il terreno fornisce la velocità da impiegare per l'hovering e i vari spostamenti orizzontali.

Sono stati impiegati tre droni del medesimo modello. A rotazione sono stati utilizzati nei vari test e modificati di conseguenza in base al test di riferimento.

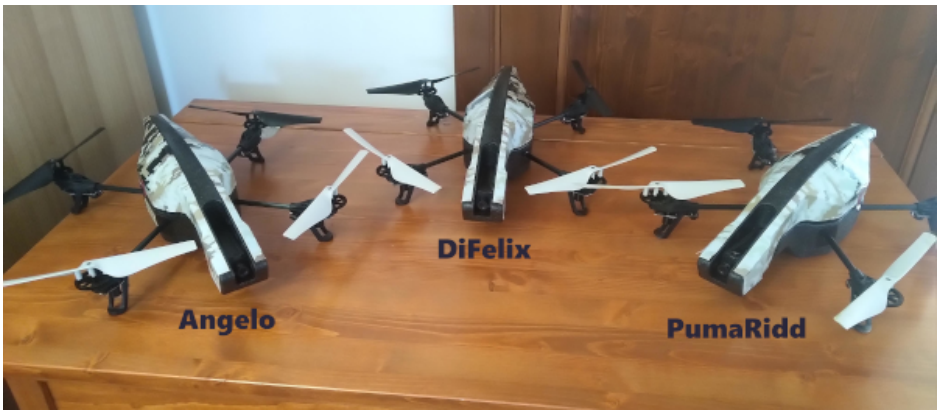

Figura 4.4: : La flotta di Droni composta da Angelo, PumaRidd e DiFelix

L'idea iniziale è stata quella di installare un power-bank direttamente sul guscio di protezione così come il Raspberry pi 3. Dall'immagine 4.5 possiamo vedere la prima versione del prototipo.

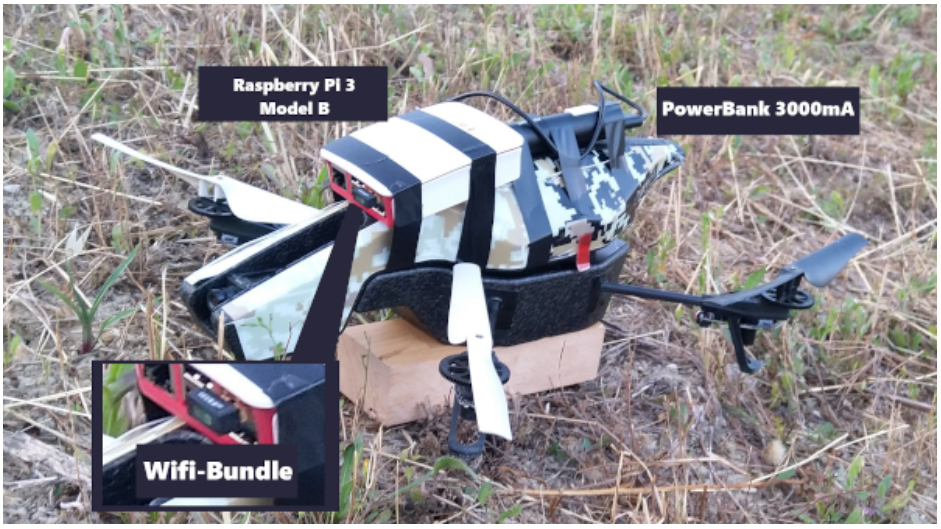

Figura 4.5: : Primo prototipo con raspberry Pi, PowerBank e Wifi-bundle

Il drone non ha un'elevata capacità di stabilizzazione, infatti il peso aggiuntivo non ne permetteva un controllo adeguato ad effettuare un volo corretto. A quel punto è stata stilata una lista di pesi per capire come alleggerire il drone, lasciando solamente lo stretto necessario a bordo.

component	weight
skull	30g
GPS module	34g
battery 1000mA	104g
battery 2000mA	126g
drone only	294g
raspberry zero W	24g
raspberry 3 B	126g
power bank 3000mA	73g

Tabella 4.3: : pesi dei vari componenti

Nella tabella 4.3 sono stati riportati i vari pesi dei componenti aggiuntivi installabili sul drone, guscio di protezione e powerbank sono stati esclusi ed è stato installato un convertitore dc-dc da 12V a 5V (gentilmente offerto da Lorenzo Montecchiari) per sfruttare solamente la batteria a bordo senza utilizzarne altre aggiuntive per il raspberry. Il secondo prototipo risulta essere "nudo" se messo a confronto con il primo, ma molto più stabile e preciso negli spostamenti.

Figura 4.6: : Secondo prototipo: niente protezione e niente powerbank, con convertitore 12v-to-5v

A questo punto è stato riscontrato un errore di alimentazione al raspberry, infatti se messo troppo sotto stress tra calore (sopra 30° agosto 2019) o batteria non carica al 100%. Ciò determinava il blocco della board per emergenza low-power, e quindi la terminazione del programma.

Figura 4.7: : Ultimo volo di DiFelix, ritrovato dopo aver perso il segnale a 20m di altezza

Tali interruzioni hanno portato al danneggiamento dei tre UAV. Durante i test sono state sostituite delle parti concesse dal drone più danneggiato. Sono stati sostituiti gli ingranaggi e i perni che sorreggono le eliche, moduli Gps e sensore di prossimità, fino alla realizzazione di un nuovo drone, con parti totalmente nuove ma con l'utilizzo di un Raspberry pi Zero W, molto meno performante ma nettamente più efficiente in termini di consumi.

Figura 4.8: : Prototipo finale, PhoeniX

Il prototipo finale in 4.8 è stato impiegato nei test finali di Sybelius. Sono state installate delle nuove eliche versione elite da competizione, è stato sostituito il telaio e sono stati ripresi i motori ancora in buono stato dai precedenti droni.

Capitolo 5

Validazione

Nella sezione 3.1 sono stati descritti gli principali obiettivi di questo elaborato. In questo capitolo verranno descritti i vari test effettuati sul campo con relativa analisi dei dati ottenuti. Verranno descritti scenario, metriche e metodologie per ogni set di risultati.

5.1 Scenario

La scelta di effettuare test sul campo in ambito UAV ha comportato molte difficoltà, soprattutto in termini di logistica. I test sono stati effettuati in ambiente outdoor (terreno privato a Colbuccaro, una frazione del comune di Corridonia (MC)). É stata allestita una postazione di controllo direttamente sul campo per agevolare i vari sviluppi del software e per avere accesso ai vari strumenti di supporto come: corrente elettrica per la ricarica dei vari device, attrezzi di riparazione drone, copertura per il raffreddamento delle board, postazione pc.

Figura 5.1: : Postazione di controllo, Colbuccaro di Corridonia(MC)

Ogni test è stato eseguito mediante la stessa configurazione di partenza, impartendo di volta in volta differenti routine in base alle necessità. La configurazione consiste nel creare un Hotspot che funga da bridge per la postazione di test e il raspberry a bordo del'UAV. La rete rende possibile l'upload e l'esecuzione di una routine da remoto e contemporaneamente permette di controllare il sistema di logging in tempo reale. La board è connessa con il suo WiFi integrato alla rete costruita dal controller del drone. Il Bluetooth del Raspberry potrà essere usato dal Manager di comunicazione se necessario. Grazie a questa configurazione, si ha pieno controllo dell'UAV anche durante una sessione di volo autonomo, poiché in caso di emergenza è possibile interrompere qualsiasi azione per arrestare il drone.

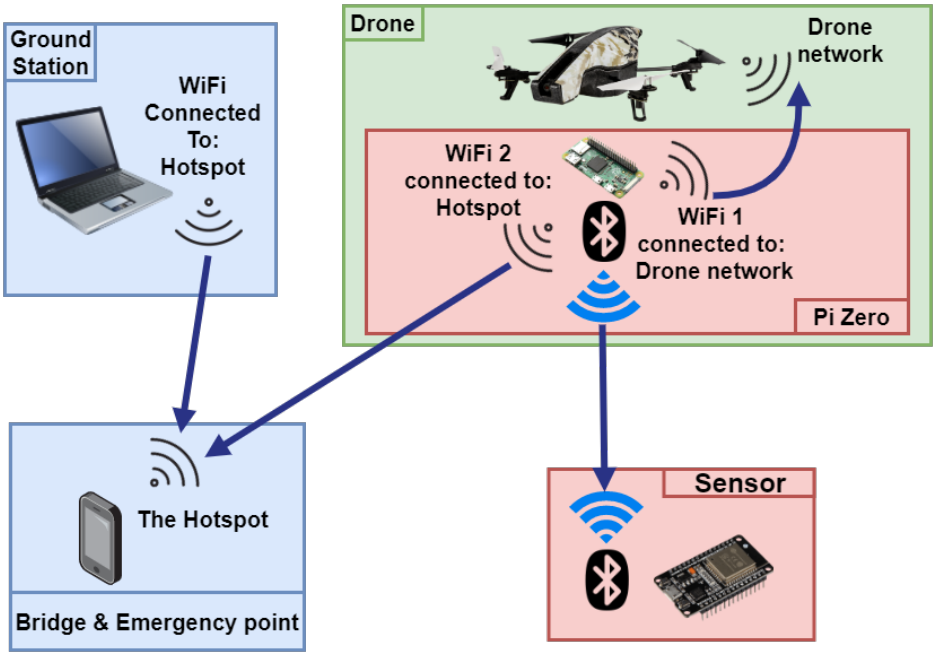

Figura 5.2: : Configurazione dello scenario

5.2 Metriche e Metodologie

Una volta implementati i task per le routine, sono state determinate delle prove per valutare i limiti di comunicazione, posizionamento e efficienza energetica.

5.2.1 Comunicazione a Terra

É stata scelta la tecnologia BLE per via della disponibilità del relativo modulo nella board Raspberry Pi (sia Zero W che 3 Model B) e per la facilità di utilizzo. L'intero progetto presenta un elevato rischio di point of failure hardware, dovuto principalmente al consumo repentino della batteria, quindi è stato deciso di ridurre al minimo il numero di componentistica aggiuntiva. La preparazione a questo test richiedeva un'alta precisione nel controllo dell'UAV, poiché il task principale era quello di eseguire un'azione di puntamento GPS direttamente sopra il sensore a svariate altezze ed effettuare di volta in volta connessioni BLE. Stabilita la connessione venivano salvati va-

lori RSS, RTT di ogni pacchetto, e numero di pacchetti inviati. La sequenza eseguita dal drone può essere riassunta come segue:

1. Viene posizionato il drone sulla piattaforma di decollo, e viene avviato Sybelius da remoto. [Punto A]
2. Decollo e calibrazione
3. Raggiungimento quota di trasmissione
4. Orientamento verso la coordinata GPS del sensore [Punto B]
5. Orientamento e puntamento verso la coordinata gps del sensore [Punto B]
6. al raggiungimento (entro 1 metro), avvio sequenza di comunicazione [comunicazione]
7. Connessione BLE ed invio di pacchetti (numero indicato nella routine) [comunicazione]
8. OPZIONALE: in caso di batteria sopra il 50%, ritorno al punto di partenza. [Punto A]
9. Slow-down e atterraggio
10. Completamento salvataggio log su file, spegnimento motori
11. Check stato batteria, shutdown

La routine è stata denominata anche A-to-B-communication o “complete”; questo secondo termine deriva dal fatto che per testare l’utilità di un UAV per la comunicazione con un sensore a terra, è stato impiegato ogni tipo di task implementato nel software. Ritroviamo i movimenti base, orientamento/ puntamento GPS e comunicazione. La Routine è stata svolta dall’UAV a differenti altezze da terra con il metodo delle prove ripetute.

5.2.2 Posizionamento durante l'hovering

Per disporre di una connessione ottimale, l'UAV deve posizionarsi esattamente sopra al sensore, o quantomeno molto vicino in termini di linea d'aria. Per calcolare questo errore, sono stati effettuati i medesimi voli dei test di comunicazione, ma questa volta scattando delle foto con la camera inferiore al raggiungimento della coordinata GPS. Tenendo conto di un errore al di sotto del metro (parliamo di voli superiori ai 40m), sono state scattate delle foto a svariate altezze ad un bersaglio a terra ogni volta che il software decideva di aver raggiunto la coordinata gps. Quindi come punto B da raggiungere, è stata inserita la coordinata GPS del bersaglio.

(a) Bersaglio originale (b) Errore di Hovering

Figura 5.3: Bersaglio in alta qualità e foto scattata dal drone.

Dalla figura 5.3b è stata estrapolata la distanza in pixel tra i due centri (immagine e bersaglio). Ad esempio in questo caso la foto è stata scattata circa a 5 metri di altezza; tenendo conto della grandezza reale del bersaglio, l'errore di hovering è di circa 196pixel ovvero 60cm.

5.2.3 Efficienza energetica

Questo tipo di misurazione è stato effettuato con delle routine apposite, diverse da quelle usate negli altri test. La misurazione veniva fatta sia controllando lo stato della batteria proveniente dal controller del drone(software), sia misurandone con il tester il voltaggio. Le routine sono state divise sempre per altezza e sono tre:

- **Hovering**: Il drone è stato lasciato in stato di hovering per tutta la durata della batteria fino all'atterraggio di emergenza.
- **Hovering + Comunicazione**: Il drone sempre in stato di hovering, ma scambiando pacchetti con il sensore a terra tramite BLE.
- **Full Movement**: Il drone effettuava puntamento e orientamento tra due Coordinate GPS fino all'atterraggio di emergenza.

Ogni sessione di volo è stata cronometrata utilizzando i timestamp dei log, ma validando l'efficacia soltanto se il voltaggio era veramente a livello critico. É emerso che infatti il software giudicava come critico anche un valore superiore al 50% di batteria.

5.3 Risultati

Grazie ai test portati a termine durante le varie fasi di sviluppo, sono stati raccolti svariati dati sperimentali. Questi ci permettono di analizzare i limiti delle scelte implementative del progetto. La sperimentazione è stata indirizzata molto verso la parte di comunicazione, poiché è l'obiettivo principale di questa tesi. Il controllo invece ritrova uno spazio ristretto, poiché ci si basa soltanto sulla precisione durante l'hovering su un punto fisso a terra.

5.3.1 Analisi della Comunicazione a Terra

Durante i vari test di comunicazione, sono stati calcolati i tempi di risposta dei pacchetti inviati dal Raspberry pi verso il sensore a terra. La difficoltà della routine ha permesso soltanto di effettuare 2 prove valide per ogni altitudine (2m, 5m, 8m , 11m) ottenendo sempre il 100% dei pacchetti inviati ma con un repentino packet loss praticamente totale appena la connessione diventava instabile.

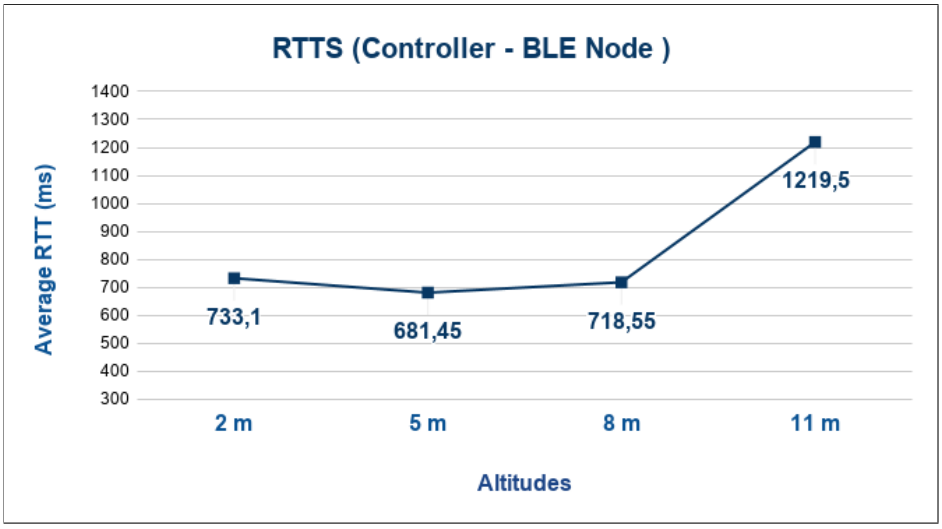

Figura 5.4: : RTTS medi a diverse altezze

Dal grafico 5.4 si può notare come i tempi medi siano praticamente gli stessi fino a 8 metri e si aggirino intorno ai 700ms. All'altezza di 11 metri, l' RTT medio si raddoppia rispetto alle configurazioni precedenti. Per approfondire il discorso, vengono introdotti anche i valori della potenza di segnale sempre alle medesime altezze.

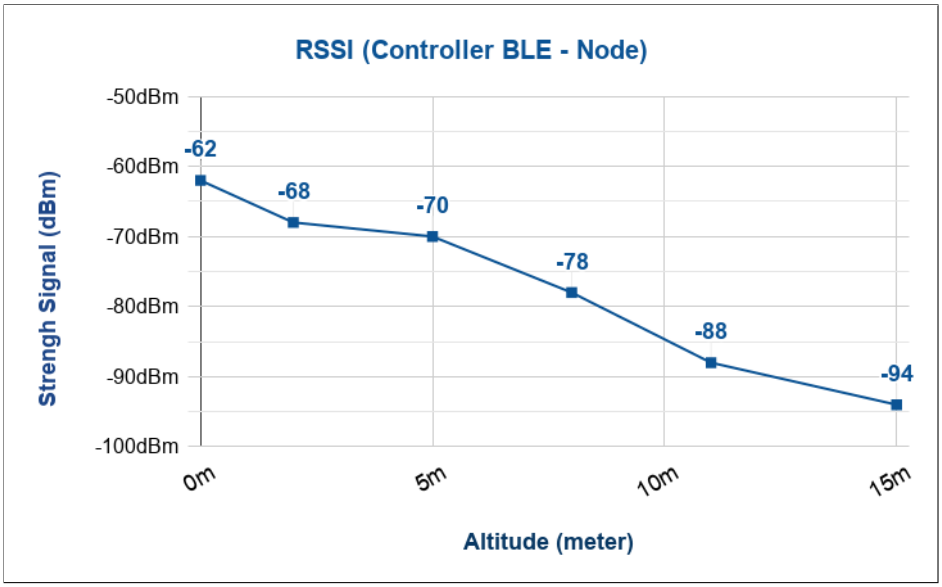

Figura 5.5: : RSSI medi a diverse altezze

Dal grafico 5.5 la potenza di segnale a distanza 0m già non è assolutamente ottimale (-62dBm), infatti il BLE viene dato per assurdo che a quella distanza debba essere pari a -30dBm o comunque a -50dBm parlando di effettivo segnale di potenza reale. Entro gli 8 metri vediamo che la potenza arrivi a -78dBm, molto in linea con gli RTT del medesimo range di altezze. Intorno agli 11 metri la potenza di segnale decresce fino al limite disconnessione (solitamente -90dBm). Ulteriori test sono stati effettuati per individuare il limite di connessione tra i device, vediamo come l'RSSI a 15 metri sia praticamente inutilizzabile (-94dBm), facendo intuire che il range di connessione in campo aperto sia intorno agli 11 metri, come già epurato.

5.3.2 Analisi dell'errore di Hovering

L'errore di Hovering deriva molto spesso dai venti che spingono il drone oltre la coordinata di destinazione. Inoltre, anche altri fattori come il magnetometro non calibrato correttamente o semplicemente la mancanza di copertura GPS possono influenzare la stabilizzazione del drone. In linea generale, il software tende a correggere la deriva ed riposizionare il drone nella coordinata GPS indicata nella routine. L'offset dal punto ideale da quello raggiunto è calcolato tramite un'immagine presa direttamente dalla camera inferiore come descritto in 5.2.2; il processo si avvia solamente se viene calcolata una distanza da raggiungere inferiore al metro. Tale distanza minima è stata introdotta come tolleranza del sistema poiché il drone non è in grado di stopparsi completamente, ma conserva parte del moto dall'ultimo movimento effettuato slittando leggermente per qualche decina di cm. Tenendo quindi conto dei problemi hardware del drone precedentemente menzionati, i risultati possono essere considerati molto soddisfacenti.

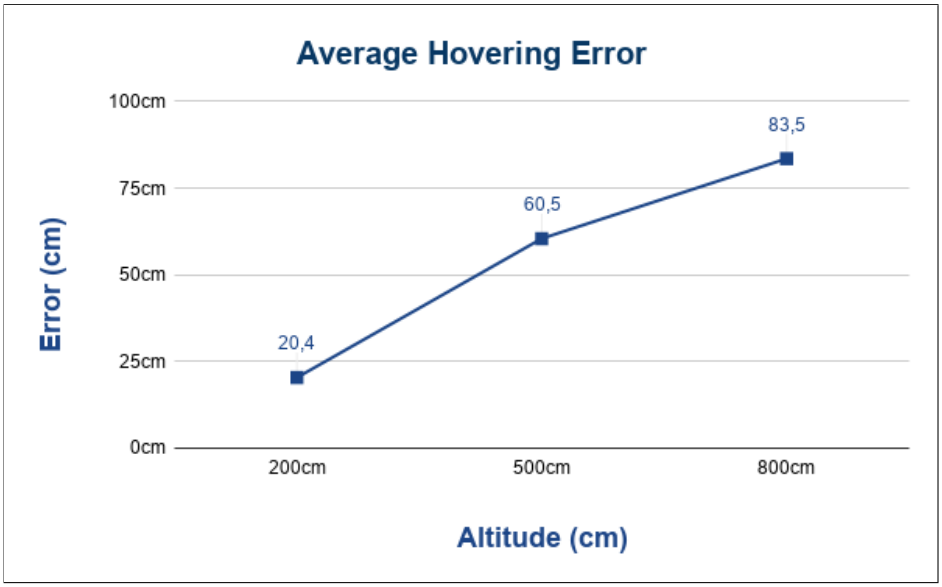

Figura 5.6: : Errore Medio di Hovering

Dal grafico 5.6 si può notare che l'hovering su una coordinata GPS perde di precisione all'aumentare dell'altezza, proprio perché più esposto alle folate di vento improvvise. I test sono stati effettuati sia facendo un pre-volo per raggiungere la coordinata (anche maggiori ai 40metri) sia direttamente sul posto.

5.3.3 Efficienza energetica sul percorso

Le batterie del drone hanno una bassa autonomia e quindi le sessioni di volo classico con smartphone non superano i novi minuti. Il test in questo caso rende evidente il consumo energetico in fase di comunicazione durante l'hovering, ed al tempo stesso investiga l'efficienza energetica di ciascuna fase.

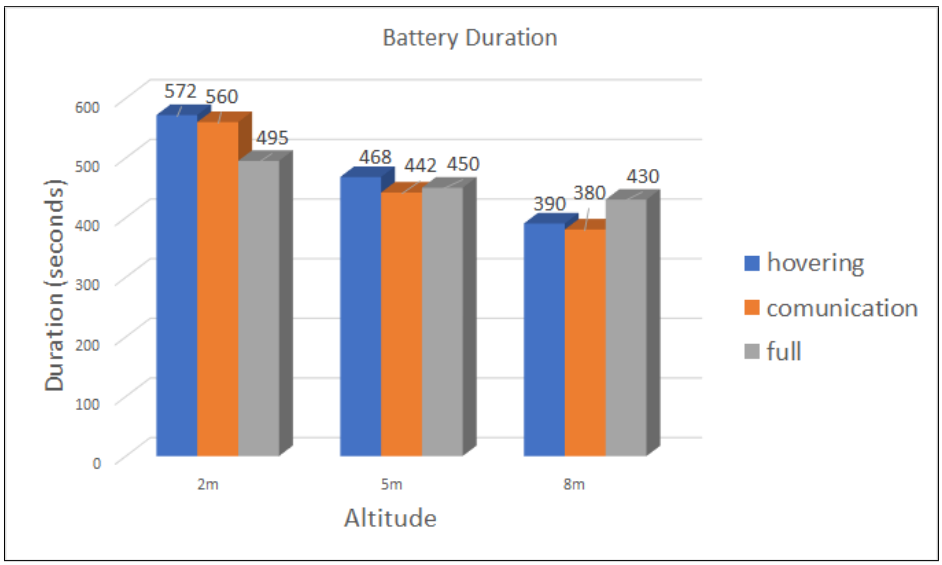

Figura 5.7: : Durata delle batteria in differenti tipologie di volo

Nel grafico vengono riportate 3 tipologie di volo ripetute a diverse altezze. I nomi delle routine sono rispettivamente:

- Hovering : "hovering"
- Hovering + Comunicazione : "communication"
- Complete : "full"

In tutte le prove possiamo notare che le routine "hovering" e "communication" hanno una durata simile inversamente proporzionale all'altezza. Il dato interessante invece riguarda la routine "full"; l'aumentare dell'altezza non incide moltissimo come negli altri casi. Durante il movimento, il controller nativo del drone stabilisce una velocità da mantenere. Questa velocità cambia in base alla spinta attuale del drone. In presenza di venti a favore, il sistema si accorge di aver superato questa velocità e quindi va a ridurre la potenza dei motori. Il sistema quindi sfrutta le folate di vento, risparmiando energia. Questo è dovuto anche alla disattivazione della stabilizzazione di volo al di fuori dell'hovering, caratteristica presente soltanto nei droni domestici.

Conclusioni

In questo elaborato è stato progettato e sviluppato Sybelius, un software di mobilità autonoma per UAV-aided sensor networks. Gli obiettivi principali del progetto sono stati rispettivamente il controllo autonomo di un UAV generico e la comunicazione drone-sensore. La piattaforma ideata doveva rispettare dei requisiti ben precisi, dettati dall'hardware a disposizione. Tutti gli UAV utilizzati nei test sono del medesimo modello, un Parrot AR Drone 2 a rotori di tipo amatoriale a basso costo.

L'architettura è stata sviluppata direttamente sul campo, sfruttando la metodologia del rapid prototyping, utilizzando quindi cicli di sviluppo estremante rapidi molto vicini al mondo agile. Essa è composta da un coordinatore principale e due moduli rispettivamente di controllo e di comunicazione. Quello di controllo gestisce tutte le azioni di movimento del drone e comunica continuamente con il controller nativo del Parrot; quello di comunicazione gestisce la tecnologia da utilizzare per la trasmissione, che in questo caso consiste nel BLE. Il sistema permette di caricare delle "missioni" di volo contenenti le azioni da far compiere al drone. Oltre alle azioni del controller nativo del drone, sono state create e testate delle azioni complesse come il puntamento e il volo autonomo verso una coordinata GPS, comunicazione BLE a terra sincronizzata alla posizione GPS del sensore, sistemi di emergenza in caso di movimenti anomali. Durante una qualsiasi sessione di volo è stato previsto un sistema di log real-time e allo stesso tempo persistito su file per un'analisi post test. Lo scenario base prevede l'utilizzo di PhoeniX, drone equipaggiato con un convertitore 12v-5v per alimentare tramite bat-

teria nativa, un raspberry pi 3 Model B che funge da coordinatore(sybelius stesso). Come sensore è stato utilizzato un ESP-32 alimentato a batteria, ed è stata allestita una stazione di controllo sempre connessa a PhoeniX in grado di caricare le Routine, controllare il sistema di log-realtime e in caso di emergenza effettuare uno stop di emergenza. Tutta la fase sperimentale è stata eseguita in ambiente outdoor. Per validare il lavoro, sono stati eseguiti dei test sotto gli aspetti di controllo, comunicazione ed efficienza energetica. I risultati evidenziano che il controllo durante la comunicazione è molto soddisfacente, ottenendo un errore di posizionamento inferiore al metro. Per quanto riguarda la comunicazione, la trasmissione BLE entro gli 8 metri ha un delay medio pari a 700ms; come parametro di riferimento troviamo una potenza di segnale pari a -78 dBm fino agli 8 metri per poi diminuire raggiungendo i -94dBm a 15 metri. Dal punto di vista della gestione della batteria, compiere azioni di hovering per la comunicazione non comporta nessuna perdita evidente in termini di durata. É interessante invece come il volo prolungato faccia risparmiare le batterie, poiché il drone sfrutta la potenza del vento invece di contrastarla andando a diminuire la potenza dei rotori. Il sistema può essere migliorato inserendo un sistema di Coverage Path Planning, così da poter sfruttare il drone nell'esplorazione di zone ignote; dal punto di vista della comunicazione sarebbe interessante andare a testare altre tecnologie di comunicazione, sia hovering ma sia in regime di volo continuo. Un ulteriore possibile sviluppo è quello di utilizzare tecnlogie wake-up radio per risvegliare un sensore a terra in standby così da risolvere il problema della sincronizzazione drone-sensore.

Bibliografia

Articoli

[Eve03] Geir Evensen. "The Ensemble Kalman Filter: theoretical formulation and practical implementation". In: *Ocean Dynamics* 53.4 (nov. 2003), pp. 343–367. ISSN: 1616-7228. DOI: 10.1007/s10236-003-0036-9. URL: https://doi.org/10.1007/s10236-003-0036-9.

[Coe+06] Ezequiel Coelho et al. "Distributed Sensing and Actuation over Bluetooth for Unmanned Air Vehicles". In: (mag. 2006).

[Wat+10] Adam Watts et al. "Small Unmanned Aircraft Systems for Low-Altitude Aerial Surveys". In: *The Journal of Wildlife Management* 74 (dic. 2010), pp. 1614–1619. DOI: 10.1111/j.1937-2817.2010.tb01292.x.

[WAH12] Adam Watts, Vincent Ambrosia e Everett Hinkley. "Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use". In: *RS* 4 (dic. 2012), pp. 1671–1692. DOI: 10.3390/rs4061671.

[Don+14] Mianxiong Dong et al. "UAV-assisted data gathering in wireless sensor networks". In: *The Journal of Supercomputing* 70.3 (dic. 2014), pp. 1142–1155. ISSN: 1573-0484. DOI: 10.1007/s11227-014-1161-6. URL: https://doi.org/10.1007/s11227-014-1161-6.

[AVVR15] Jessica Alvarenga, Nikolaos I. Vitzilaios, Kimon P. Valavanis e Matthew J. Rutherford. “Survey of Unmanned Helicopter Model-Based Navigation and Control Techniques”. In: *J. Intell. Robotics Syst.* 80.1 (ott. 2015), pp. 87–138. ISSN: 0921-0296. DOI: 10.1007/s10846-014-0143-5. URL: http://dx.doi.org/10.1007/s10846-014-0143-5.

[HTA16] N. Hossein Motlagh, T. Taleb e O. Arouk. “Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives”. In: *IEEE Internet of Things Journal* 3.6 (dic. 2016), pp. 899–922. DOI: 10.1109/JIOT.2016.2612119.

[SCZLX16] W. Shi, J. Cao, Q. Zhang, Y. Li e L. Xu. “Edge Computing: Vision and Challenges”. In: *IEEE Internet of Things Journal* 3.5 (ott. 2016), pp. 637–646. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2579198.

[YZYL17] X. Yi, A. Zhu, S. X. Yang e C. Luo. “A Bio-Inspired Approach to Task Assignment of Swarm Robots in 3-D Dynamic Environments”. In: *IEEE Transactions on Cybernetics* 47.4 (apr. 2017), pp. 974–983. ISSN: 2168-2267. DOI: 10.1109/TCYB.2016.2535153.

[LT18] Majd Latah e Levent Toker. “Artificial Intelligence Enabled Software Defined Networking: A Comprehensive Overview”. In: *CoRR* abs/1803.06818 (2018). arXiv: 1803.06818. URL: http://arxiv.org/abs/1803.06818.

[Sha+18] Hazim Shakhatreh et al. “Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges”. In: *CoRR* abs/1805.00881 (2018). arXiv: 1805.00881. URL: http://arxiv.org/abs/1805.00881.

[TFMCB18] A. Trotta, M. D. Felice, F. Montori, K. R. Chowdhury e L. Bononi. "Joint Coverage, Connectivity, and Charging Strategies for Distributed UAV Networks". In: *IEEE Transactions on Robotics* 34.4 (ago. 2018), pp. 883–900. ISSN: 1552-3098. DOI: 10.1109/TRO.2018.2839087.

[CBF19] Tauã M. Cabreira, Lisane B. Brisolara e Paulo R. Ferreira Jr. "Survey on Coverage Path Planning with Unmanned Aerial Vehicles". In: *Drones* 3.1 (2019). ISSN: 2504-446X. DOI: 10.3390/drones3010004. URL: http://www.mdpi.com/2504-446X/3/1/4.

[CLHGZ19] W. Chen, B. Liu, H. Huang, S. Guo e Z. Zheng. "When UAV Swarm Meets Edge-Cloud Computing: The QoS Perspective". In: *IEEE Network* 33.2 (mar. 2019), pp. 36–43. ISSN: 0890-8044. DOI: 10.1109/MNET.2019.1800222.

[MYLZ19] M. E. Mkiramweni, C. Yang, J. Li e W. Zhang. "A Survey of Game Theory in Unmanned Aerial Vehicles Communications". In: *IEEE Communications Surveys Tutorials* (2019), pp. 1–1. DOI: 10.1109/COMST.2019.2919613.

Libri

[NCKS17] K. Namuduri, Serge Chaumette, Jae Kim e Sterbenz. *UAV Networks and Communications*. Nov. 2017. DOI: 10.1017/9781316335765.

Online

[RES13] Va RESTON. *Oceus Networks Flies Deployable LTE Solution Above the Clouds to Support Public Safety*. 2013. URL: https://www.oceusnetworks.com.

[S F15] Y. Wan S. Fu. *Spotlight: UAVs for disaster area communication.* 2015. URL: https://www.hdiac.org/sites/default/files/spotlights/UAVs.pdf.

[Gar19] Gartner. *Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020.* [ultima vista 22.09.2019]. 2019. URL: https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io.

[Sta19] Statista. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions).* [ultima vista 22.09.2019]. 2019. URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.