UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale ingegneria dell'informazione per videogame e realtà virtuale

OTTIMIZZAZIONE DELLE STRATEGIE DI GIOCO IN TEMPO REALE PER MINI SUMOBOT TRAMITE FIRMWARE E SENSORISTICA DEDICATI

REAL-TIME GAME STRATEGY OPTIMIZATION FOR MINI SUMOBOT USING DEDICATED FIRMWARE AND SENSORS

Relatore:
Dott.ssa **Laura Falaschetti**

Tesi di Laurea di:
Antonio Basili

Correlatore:
Dott. **Michele Alessandrini**

A.A. 2025/2026

SOMMARIO

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

INTRODUZIONE

Il mondo dei mini sumo robot si rifà alle competizioni dello sport tradizionale giapponese del sumo. Come previsto da tale sport, lo scopo di queste competizioni è riuscire a spingere l'avversario oltre il perimetro del ring l'avversario; lo scontro viene consideratosi concluso nel momento in cui uno dei due partecipanti tocca il pavimento esterno del ring.

Nel caso dei mini sumobot questi si muoveranno in maniera totalmente autonoma in base ad un software realizzato precedentemente e caricato sulla loro scheda di controllo embedded.

L'aspetto affascinante di questo mondo è che, oltre all'aspetto ricreativo, si è visto quanto questo genere di competizione possa stimolare l'apprendimento per materie quali elettronica ed informatica. La realizzazione di uno di un sumobot con lo scopo di riuscire a vincere, infatti, necessita di conoscenze in diversi campi come programmazione ed elettronica, ma a giocare un ruolo fondamentale sono anche la logica e l'elasticità di pensiero [1].

Importante ruolo ricopre anche l'idea della competizione, questa infatti stimola gli studenti a sviluppare strategie sempre più innovative ed imprevedibili; consentendo l'acquisizione di abilità come la creatività ingegneristica, il problem solving avanzato e la capacità di operare al di fuori degli schemi metodologici convenzionali [2].

Il presente elaborato ha lo scopo di fornire in maniera dettagliata la trattazione del processo di studio, ingegnerizzazione e realizzazione di un mini sumobot autonomo chiamato Lancelot.

Il progetto nasce da un modello hardware di base sviluppato da AM Microsystems, il quale è stato progressivamente migliorato attraverso lavori sull'architettura hardware e sull'infrastruttura software.

I miglioramenti effettuati hanno riguardato inizialmente la parte di ottimizzazione del software di base con l'implementazione di nuove strategie in grado di utilizzare al meglio il singolo sensore ToF montato sull'architettura di base. Successivamente si è andati a modificare la struttura hardware implementando due nuovi sensori ToF per la rilevazione dell'avversario.

Grazie a questa modifica è stata sviluppata una nuova strategia in grado di eseguire un tracciamento dell'avversario e una funzione di mappatura dell'avversario basata su una griglia radiale composta di celle.

Per mettere alla prova il lavoro realizzato sono stati eseguiti diversi test, riportando i dati tramite una tabella e confrontando le varie strategie grazie a degli istogrammi.

Questa tesi si svilupperà nel seguente modo:

- Nel Capitolo 1 verrà presentato il mondo dei sumobot e delle sue varie categorie, con una particolare attenzione per la categoria dei mini sumobot. Verranno poi esposti il regolamento, il funzionamento fisico e le architetture più utilizzate.

- Nel Capitolo 2 si avrà una piccola sezione dedicata allo stato dell'arte in letteratura con un focus su alcuni progetti realizzati da altri studenti e su quali nuove tecniche software si stanno studiando per migliorare le prestazioni.
- Il Capitolo 3 riguarderà la descrizione del mini sumobot realizzato, descrivendo in maniera dettagliata la struttura hardware e software utilizzata, inclusi sistemi di alimentazione, sensori e software utilizzato.
- Nel Capitolo 4, si parlerà di come sono state realizzate le diverse strategie e del loro funzionamento.
- Nel Capitolo 5 verranno descritte le metodologie utilizzate per effettuare i test e verranno mostrati risultati delle varie strategie in comparazione con relativi istogrammi e tabelle.
- Nel Capitolo 6 si esporrà come è stata realizzata la tecnica per la mappatura dell'avversario tramite griglia radiale e verranno mostrati alcuni esempi di tale tecnica.

Con questa tesi si vuole raggiungere l'obiettivo di realizzare un mini sumobot capace di massimizzare il proprio tasso di successo in scenari diversificati grazie all'implementazione di strategie ottimizzate, progettate per sfruttare al massimo i sensori presenti.
Necessario sarà dunque lo studio e la comprensione del funzionamento dei sensori ToF e l'ideazione di strategie in grado di gestire più casistiche possibili.

CAPITOLO 1
GENERALITÀ SUMO BOT

1.1 Descrizione delle categorie

Le competizioni dei sumobot prevedono che questi siano divisi in diverse categorie in maniera tale da consentire scontri equilibrati e differenti sfide per gli sviluppatori [3].

Categoria	Altezza (cm)	Larghezza (cm)	Lunghezza (cm)	Peso (g)
Mega-sumo	Illimitata	20	20	3000
Mini-sumo	Illimitata	10	10	500
Micro-sumo	5	5	5	100
Nano-sumo	2,5	2,5	2,5	25
Feta-sumo	1	1	1	1
Lego-sumo	20	15	15	1000

Tabella 1: Categorie competizioni sumobot.

1.2 Regole di gioco

L'incontro di robot da combattimento per la categoria mini-sumobot è stabilito da un regolamento. Questo prevede che l'incontro avvenga in un ring circolare, chiamato Dohyo [do.hio], realizzato o in alluminio con uno strato di gomma nera, oppure in legno indeformabile verniciato di nero. A prescindere dal materiale esso deve avere un'altezza di 2,5 cm ed un diametro di 77 cm, con un bordo largo 2,5 cm di colore bianco.

Le linee di partenza ("Sikiri-Sen") sono designate come due linee marroni (scuro opaco) con una larghezza di 1cm e una lunghezza di 10cm. Ogni linea è posizionata a 5 cm dal centro del ring (Figura 1). I contendenti devono posizionare il loro robot sopra o dietro le linee di partenza. [4]

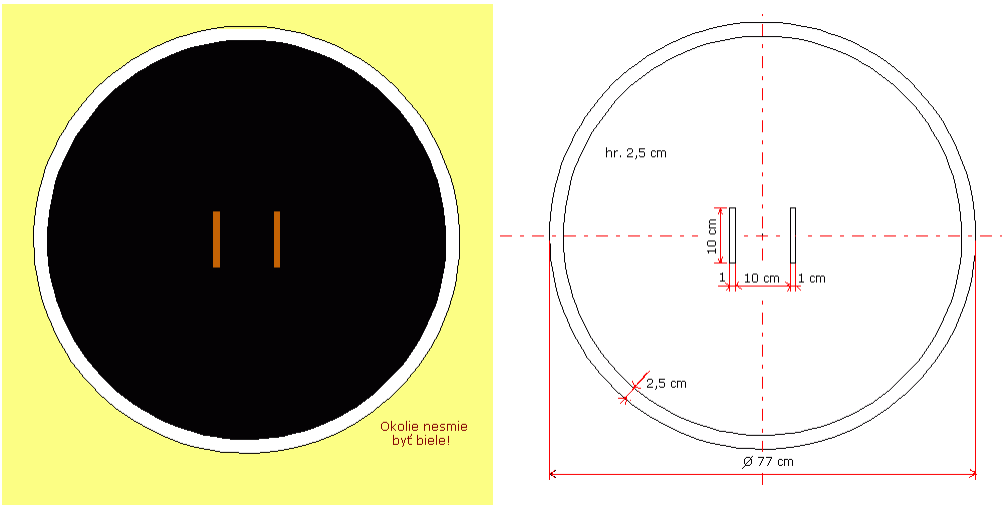


Figura 1: Dohyo per le competizioni mini sumobot.

Il mondo competitivo dei mini sumobot prevede due grandi eventi mondiali: l'"International Robot Sumo Tournament" di Tokyo, e i "RoboGames" degli USA. Nel panorama nazionale si hanno invece la "RomeCup", Organizzato dalla Fondazione Mondo Digitale a Roma, e le "Olimpiadi di Robotica" che organizzano competizioni in diverse fiere locali della Makerspace.

1.3 Cenni di fisica

Per riuscire a spingere l'avversario fuori dal Dohyo, si ha bisogno di una forte spinta in grado di sovrastare quella dell'avversario. Tale spinta però viene influenzata da diverse forze; come quella torcente, normale e di frizione. La più impattante tra queste è la forza di attrito statico (frizione); applicata nel punto di contatto tra gli pneumatici e il terreno con direzione opposta al sumobot e con modulo dato dalla formula:

$$F_s = \mu_s \cdot F_N$$

Dove μ_s sta per il coefficiente di attrito statico, dato dal materiale della ruote del robot e dal materiale del Dohyo, e con F_N la forza normale data dal peso del robot a causa della gravità.

1.4 Architettura

La realizzazione della struttura rappresenta un elemento fondamentale nel processo di creazione del mini sumobot. Nelle ultime generazioni gli elementi a cui si tende fare maggiore attenzione sono:

- Distribuzione delle masse: mentre i prototipi didattici si basano principalmente sulla stampa 3D, i robot da competizioni di alto livello utilizzano telai realizzati in Ergal (Alluminio 7075), ottone o fibra di carbonio. L'ottone viene spesso impiegato per le

parti inferiori del telaio per abbassare il baricentro il più vicino possibile al suolo, riducendo il momento di ribaltamento durante la spinta.

- Sistemi di trazione: solitamente vengono preferite le ruote con cerchio in metallo e pneumatici in silicone colato o poliuretano. Questi materiali richiedono spesso la pulizia del Dohyo prima del match per evitare che la polvere ne comprometta il coefficiente d'attrito.
- La lama (Scoop): essa è la parte frontale del robot che esso usa per provare a sollevare l'avversario. Solitamente prevede l'uso di acciaio temperato o lame di taglierina industriali affilate a livello micrometrico. Se il robot riesce a sollevare anche solo di un millimetro le ruote nemiche, ne azzera la forza normale, annullando istantaneamente la capacità di spinta dell'avversario.

1.5 Sensoristica

Affinché il robot si possa muovere in maniera autonoma, i sensori sono una risorsa essenziale per aver riscontri sull'ambiente circostante.
Sebbene l'utilizzo di un solo sensore possa essere sufficiente in molti casi, il multisensing può risultare elemento principale per il raggiungimento della vittoria. I loro compiti sono semplicemente il rilevamento del bordo bianco e la localizzazione dell'avversario.

Per il primo scopo vengono usati dai due ai quattro sensori riflessione infrarossa posti agli angoli estremi del robot e tarati per rilevare la variazione di riflettanza tra il fondo nero opaco del ring e la linea perimetrale bianca lucida. Un esempio dei più comuni sono i ML1 - QRE1113GR.

Per quanto riguarda la localizzazione dell'avversario i più utilizzati sono i sensori TOF (TIme-of-Flight). Questi calcolano la distanza misurando il tempo di volo dei fotoni riflessi, offrendo tempi di campionamento fino a pochi millisecondi e un'immunità quasi totale al colore e al materiale del robot avversario. Un esempio dei più usati sono i VL53L1X.

CAPITOLO 2
STATO DELL'ARTE

La scrittura di articoli scientifici riguardanti il mondo dei sumobot ha visto negli ultimi anni una grande crescita, dovuta soprattutto ad uno spostamento di focus centrato maggiormente sullo sviluppo di architetture logiche e algoritmiche avanzate. Le condizioni di vittoria non sono più determinate dalla sola forza di spinta, ma vanno a dipendere principalmente dalle strategie di attacco e difesa realizzate dal programmatore [5].

2.1 Studi in letteratura nell'ambito dei mini sumobot

Un gruppo di studenti della State University of Rio de Janeiro (UERJ) hanno deciso di realizzare un mini sumobot capace di gareggiare e con la peculiarità di poter essere con guida autonoma o controllato tramite un dispositivo remoto. Tale mini sumobot, denominato Ultra T, è stato realizzato tramite stampante 3D per quanto riguarda la struttura esterna, mentre per i sensori hanno deciso di utilizzare due sensori QRE1113 per il controllo del Doyho e quattro sensori JSF40 posizionati frontalmente per il rilevamento dell'avversario. Dal punto di vista della programmazione tuttavia si sono limitati alla realizzazione di una strategia molto semplice basata su 3 stati: "PREPARE", "ON" e "STOP". Lo stadio principale (ON) prevede che il sumobot giri intorno a se stesso lentamente fino al rilevamento dell'avversario, verso il quale il sumobot andrà a muoversi ad alta velocità [6].

Un altro caso di studio è stato svolto dallo studente Derian Guevara dell'Università Politecnica Salesiana dell'Ecuador. Anche qui la struttura è stata realizzata grazie alla stampa 3D e sono stati utilizzati 3 sensori JSF40 (uno centrale e uno per lato) per il rilevamento dell'avversario mentre per evitare di uscire dal ring sono stati utilizzati sensori ML1. La board utilizzata è stata invece la XMotion.

La parte software è stata realizzata anche qui con un approccio molto semplice: il sumobot controlla per prima cosa se sta per uscire dal Dohyo e nel caso si dà una spinta in retromarcia. Successivamente vengono controllati i vari sensori JSF40 e se nessuno di questi rileva l'avversario, il sumobot inizia a girare su sé stesso lentamente. Nonostante la strategia sia molto semplice il sumobot sembra essere efficace, anche se molto limitato dai sensori JSF40 che possiedono un range massimo di rilevamento di 20 cm che portano spesso il sumobot ad agire troppo tardi [7].

2.2 Approcci recenti

Negli ultimi anni, la ricerca e lo studio tramite sumobot, ha portato studenti e ricercatori ad esplorare il mondo delle reti neurali per cercare di implementare all'interno dei sumobot; andando così a realizzare un complesso sistema di analisi dell'ambiente capace di prendere decisioni a run time specifiche per ogni competizione. La ricerca in questo ambito ha portato la realizzazione di diverse metodologie per l'implementazione di reti neurali veloci, andando a toccare anche l'ambito di sistemi fuzzy.

2.2.1 Fuzzy

Il sistema fuzzy, a differenza della logica booleana classica basata su vero e falso, introduce il concetto di verità parziale, ovvero l'assegnazione di valori compresi tra 0 e 1 che consentono una maggiore precisione e fluidità. L'utilizzo di questa tecnica negli scontri dei sumobot consente di interpretare i dati dei sensori in tempo reale e prendere decisioni adattive in condizioni incerte e dinamiche.

La realizzazione di questa tecnica inoltre risulta molto semplice e poco dispendiosa. Inizialmente vengono lette le variabili dai sensori e vengono mappate tramite funzioni decise precedentemente in modo da ridare il risultato dell'azione da intraprendere in base alla posizione del nemico. Un esempio può essere lo studio svolto da M.S. Mohd Shukr et al. [8] in cui vengono lette le informazioni fornite da tre sensori (Sharp GP2Y0A21YK0F) posti nella parte frontale del mini sumobot e rivolti verso l'interno. Il sistema fuzzy andava poi ad associare ad ogni situazione dei vari sensori, quale velocità associare al motore destro e sinistro. Si è visto poi, tramite simulazione fisica contro un semplice sistema if-else, come il sistema fuzzy abbia consentito un movimento fluido e continuo in un contesto dinamico.

2.2.2 Reti neurali e neuro-fuzzy

L'elemento fondamentale di una rete neurale è il neurone artificiale; il suo meccanismo prevede la somma algebrica degli ingressi pesati e la successiva elaborazione tramite una funzione di attivazione responsabile del controllo fluido e continuo del sistema (come la funzione identità o la funzione di Elliott). I neuroni vengono poi connessi tra loro tramite l'architettura a cascata o l'architettura "Multi Layer Perception" (MLP).

Elemento fondamentale nell'implementazione delle reti neurali è la fase di addestramento che richiede un enorme quantitativo di dati di input con associato già il corrispettivo output [9].

Nonostante le reti neurali siano in grado di ottenere risultati più puliti e definiti con un codice più breve rispetto alla fuzzy; la necessità di una mole così dispendiosa di informazioni e un processo di progettazione molto complicato, ha reso l'implementazione di tale tecnica nei sumobot alquanto complicata. Ad aggiungere difficoltà è la mancanza di un database pregenerato e la grande variazione di possibili sumobot avversari. Per questo motivo la ricerca ha iniziato a pensare di sfruttare le reti neurali per andare a supporto della tecnica fuzzy. Grazie alla combinazione delle due, infatti, si riesce ad ottenere delle funzioni e delle regole tramite reti neurali, che vengono poi gestite dalla logica fuzzy. Si ottiene così una la grande flessibilità delle reti neurali, con la leggerezza computazionale dei sistemi fuzzy [10] [11].

Sul mondo dei sumobot vi è ancora un dibattito molto ampio che tuttavia rimane spesso fermo nella teoria o al limite nelle simulazioni digitali, perdendo così un importante e fondamentale aspetto dello studio tramite sumobot, ovvero l’attuazione delle strategie in situazioni vere, in cui entrano in atto anche i possibili problemi fisici.

CAPITOLO 3
DESCRIZIONE MINI SUMOBOT LANCELOT

3.1 Descrizione hardware

Il Lancelot è realizzato con una struttura che potremmo dividere in tre livelli. Partendo dal livello più in alto abbiamo un PCB ai quali sono stati saldati e collegati due sensori TOF VL53L4CX [12]. Collegata a questa vi è il secondo livello, composto da una STM32 Nucleo L476RG [13] che rappresenta il cervello del sumobot, all'interno della quale risiede il codice di controllo per la gestione delle diverse strategie. Infine, nel primo livello (quello più in basso) è posizionata la scheda principale [14] la quale è incaricata di gestire i vari collegamenti con i diversi componenti hardware specifici. Questi includono:

- Caricabatterie (da USB-C)
- Convertitore USB-seriale (da USB-C)
- Modulo ESP32-S3 (Wi-fi, Bluetooth)
- 4 LED RGB
- Accelerometro - giroscopio
- Sensore temperatura - umidità
- Driver motori
- Sensore di prossimità di tipo VL53L4CX ToF (time-of-flight)
- Sensore ottico line-follower.

3.1.1 Alimentazione

L'alimentazione può avvenire tramite due modi: da connettore USB-C, oppure tramite batteria. Nel primo caso la batteria viene ricaricata e allo stesso tempo viene fornita alimentazione V_{DC} per la scheda. Mentre nel secondo abbiamo che la sola batteria fornisce alimentazione. La tensione V_{DC} viene sfruttata per generare diverse tensioni che alimentano le diverse parti della scheda; per i motori abbiamo 6V, per la scheda Nucleo si necessita di 7,5V e per il line-follower si usa la 7,5V e la 5V. Per il resto dei componenti si usa la 3,3V.
Vi è inoltre la possibilità di misurare la tensione della batteria tramite il pin 28 del connettore CN7 della scheda Nucleo e un partitore con rapporto 0,69. Il segnale è connesso al pin PA0 del microcontrollore e può essere letto tramite ADC. Dato che l'ADC satura a 3,3V, con tale rapporto la lettura massima è 4,78 V.

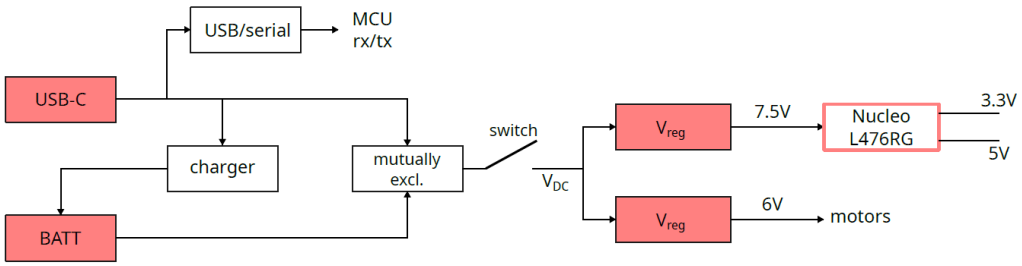


Figura 2: Rappresentazione del sistema di alimentazione del Lancelot.

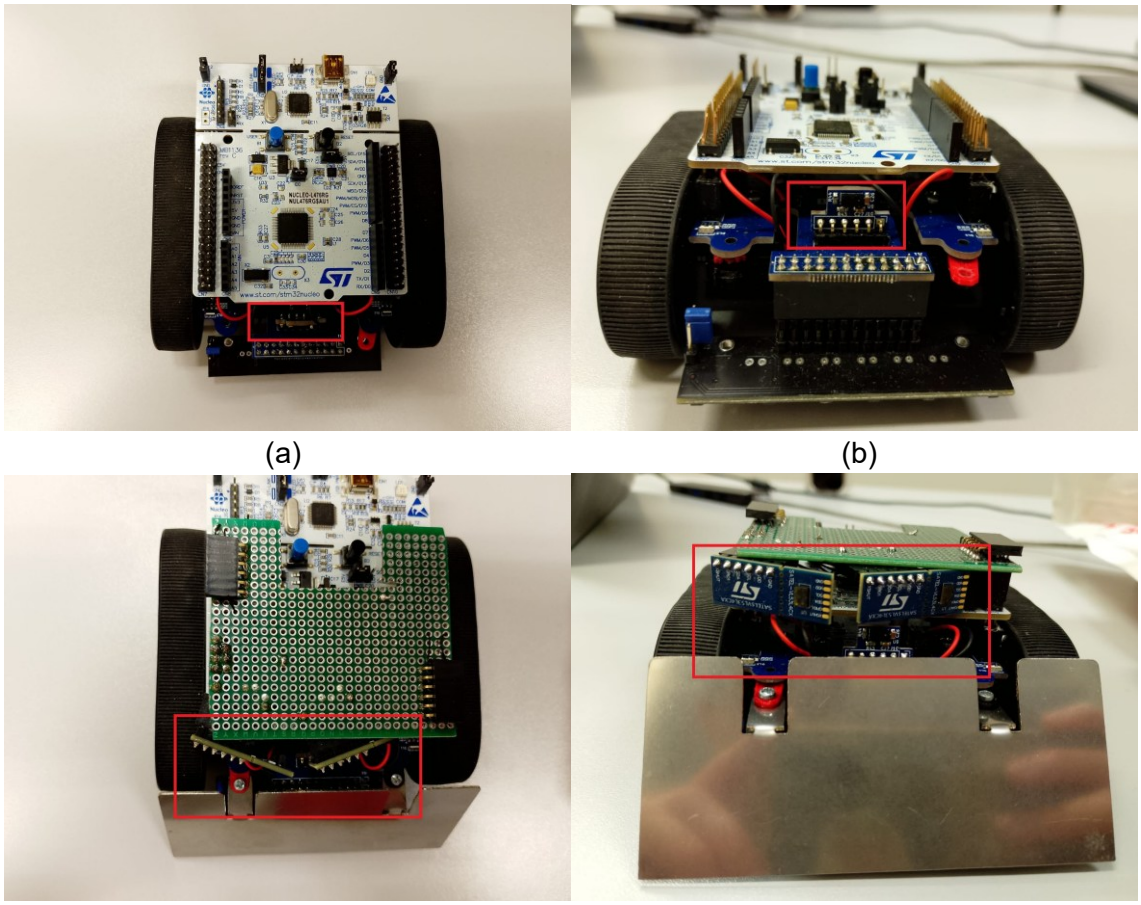
3.1.2 Gestione sensori ToF

Viene definito sensore ToF (Time of Flight) quel sensore che, generando un fascio luminoso, va a misurare la distanza tra sé stesso e un altro oggetto calcolando il tempo che questo fascio impiega a colpire l'oggetto e tornare indietro al sensore. Per effettuare questa misura il sensore ha bisogno di tre fasi fondamentali:

- Emissione: viene emesso il fascio di luce.
- Riflessione: l'oggetto colpito riflette l'impulso luminoso.
- Rilevazione: il rilevatore del sensore misura il tempo totale impiegato dal fascio di luce per tornare.
- Calcolo della distanza: si va ad utilizzare il tempo ottenuto (T) e la velocità della luce (c) per ottenere la distanza (D) dall'oggetto grazie alla formula:

$$D = \frac{c \cdot T}{2}$$

La versione base dei sumobot utilizzati, come anche la prima versione del Lancelot, prevede solamente un sensore ToF centrale montato sulla scheda principale (*Figura 3 (a)* e *(b)*). Poiché si è constatato che un singolo sensore avrebbe limitato molto l'efficacia dell'individuazione dell'avversario, si è pensato ad una nuova versione con in aggiunta due nuovi sensori, uguali al precedente, posizionati sopra al sensore centrale ad una distanza di circa 1,5 cm da esso e con un'inclinazione di 20° verso l'esterno (*Figura 3 (c)* e *(d)*). Questi ultimi sono stati collegati al PCB e connessi poi alla scheda Nucleo.
Un problema rilevato è stato la gestione delle inizializzazioni dei nuovi sensori ToF aggiunti, in quanto tutti e tre condividono nativamente lo stesso indirizzo I2C di fabbrica. Per riuscire a risolvere questo problema si è optato per l'accensione e l'inizializzazione di questi uno alla volta. Così facendo si riesce ad aggirare il problema dato che, una volta inizializzato, possiamo spostare il sensore su un altro indirizzo senza riscontrare problemi. Nel caso del Lancelot sono state usate le porte dedicate inizialmente ai led posteriori a destra.



(a) (b)

(c) (d)

Figura 3: Configurazione dei sensori ToF nel sumobot base e nel modello Lancelot: (a–b) singolo sensore ToF nel sumobot di base; (c–d) configurazione con tre sensori ToF nel modello Lancelot.

3.2 Descrizione software

Il software utilizzato per andare a lavorare con il Lancelot è stato il “STM32CubeIDE” nella versione 2.1.0 e il “STM32CubeMX” nella versione 6.16.1, in quanto software ufficiale della scheda Nucleo. Questa scheda è stata la base per la gestione software dei vari componenti del sumobot, come i motori o la gestione dei sensori.

A permettere la comunicazione tra la scheda Nucleo e l’apparecchiatura hardware si trova la scheda principale posta in basso. All’interno dell’architettura software si trovano diverse classi con lo scopo di andare a gestire i vari elementi hardware. Per i motori, ad esempio, vengono gestiti tramite i codici “PWMChannel.h” e “Motor.h” dai quali possiamo controllare diversi parametri come la velocità, la direzione e lo stato (start o stop) grazie alle diverse funzioni implementate. I vari sensori ToF vengono gestiti dai quattro script nella cartella “TOF” e da “RangingSensor.h” nella cartella “sources_hal”; tramite questi possiamo andare a gestire i dati raccolti dai sensori come la distanza ottenuta o la potenza del segnale di ritorno; molto utile per limitare i falsi negativi. Per quanto riguarda invece i sensori line-follow, essi vengono gestiti

nello script “LineFollower.h”. Infine, l'accensione delle luci (utilizzata principalmente per feedback visivi) viene gestita da “TurnLight.h”.

La struttura principale viene gestita principalmente dal file “main.c”, il quale richiama la funzione di “setup()” e di “loop()” presenti nel file “application.cpp”; oltre a contenere diverse configurazioni per l’inizializzazione delle periferiche.

CAPITOLO 4
IMPLEMENTAZIONE DELLE STRATEGIE

Nella versione iniziale del Lancelot, come negli altri sumobot di base, era presente una sola strategia descritta nel file “application.cpp”. Questa funzionava sfruttando le funzioni “setup()” e “loop()”. La prima si occupava di inizializzare i vari componenti hardware mentre la seconda andava a controllare la distanza ottenuta dal sensore ToF per decidere come comportarsi.
Successivamente si è pensato ad una migliore scalabilità, andando a creare una classe genitore chiamata “Strategy” da cui avrebbero ereditato tutte le classi per le diverse strategie. Così facendo si è potuto raggruppare ciò che tutte le strategie avevano in comune in un’unica classe e lasciando così alle classi figlie solamente il compito di descrivere il comportamento della loro strategia.
La classe genitore “Strategy.h” prevede principalmente le funzioni: “EnableAll()”, “getDistance()” e “StartStrategy()”. La prima viene chiamata in “application.cpp” nella funzione “setup()” e serve all’inizializzazione di tutti i sensori, i motori, dei vari led e del monitor seriale. La seconda invece viene richiamata dalle diverse strategie per ottenere la distanza del bersaglio colpito dal sensore ToF passato come argomento, consentendo così di andare a scegliere di quale sensore si vuole calcolare la distanza dal possibile avversario. Infine si ha “StartStrategy()”, la funzione più importante in quanto richiamata come override dalle classi figlie per descrivere l’intera strategia che dovrà usare il Lancelot in quella situazione.
Questo approccio ha permesso di avere un codice più pulito sia nell’ “application.cpp” che nelle diverse strategie sviluppate, consentendo inoltre una facile aggiunta di eventuali nuove strategie.

4.1 Slow search

La prima strategia presente è stata rinominata “Slow search”, e prevede l’avvio dei motori a velocità media mantenendo una traiettoria rettilinea. Tale stato di ricerca può venire fermato solo nei casi in cui i sensori di line-follower percepiscono il cambiamento di colore del ring, indicando così il bordo di questo. In questo caso il sumobot inverte il senso di marcia ed esegue una rotazione in senso orario per cambiare direzione. Un ulteriore caso in cui viene fermato lo stato di ricerca è quando la funzione “getDistance()” restituisce un valore pari o inferiore al valore di threshold scelto precedentemente, in questo caso 250 mm. In tal caso il sumobot riconosce l’oggetto davanti a sé come il nemico, procede ad aumentare la sua velocità al massimo e a caricarlo in linea retta.
Come già accennato, il funzionamento di questa strategia sfrutta la logica a stati (Search, Attack, Back e Off) che vengono associati nel codice in base alle risposte dei sensori.

Si è provato ad aggiungere un nuovo stato chiamato “StateFalse” ma si è subito riscontrata la poca efficienza di tale stato, nonché la maggiore possibilità di portare il sumobot fuori dall’arena involontariamente. Si era previsto il passaggio in tale stato in maniera randomica quando lo stato era “StateSearch”, dopodiché avrebbe effettuato una fermata per poi girarsi e ripartire con lo stato di Search.

Alla fine, quindi, l’unico cambiamento apportato a questa strategia, è stata la gestione della rotazione del sumobot quando percepisce di essere sul bordo del ring. Poiché nel sensore line-follower sono presenti sei sensori, si è pensato di usufruire di questo fatto. Il Lancelot ora controlla quali di questi sensori rilevano il bordo e, in base a sé sono quelli più a sinistra o quelli più a destra, dopo essere indietreggiato esegue una rotazione nella direzione migliore per allontanarsi il prima possibile ed in maniera più sicura. Se si attivano tutti quanti vuol dire che si trova completamente sopra al bordo, in questo caso allora viene fatto scattare indietro con maggiore potenza.

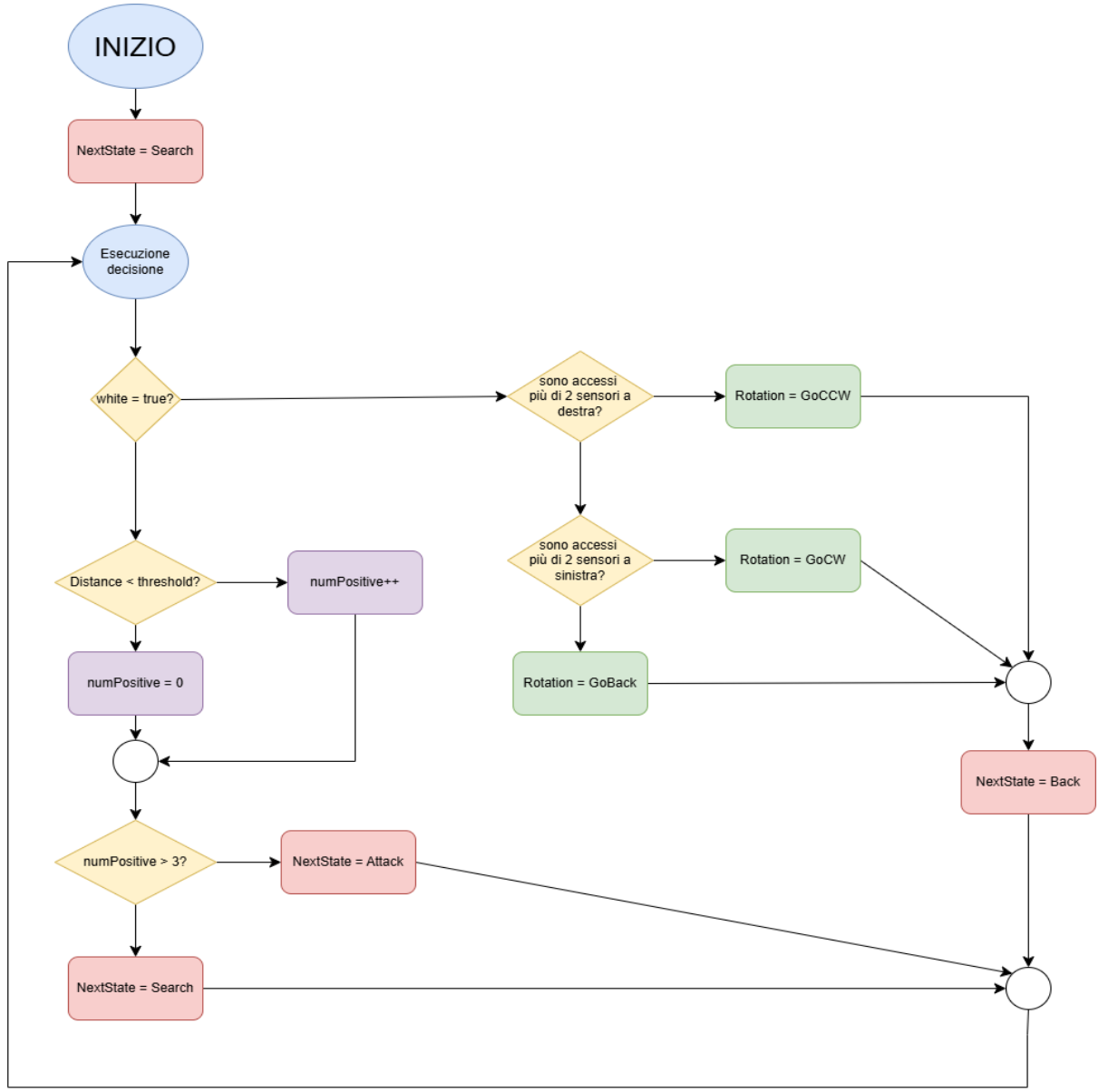


Figura 4: Flowchart della strategia Slow search.

4.2 Tornado

La seconda strategia presente sul Lancelot è la “Tornado”. Essa si basa sul far girare il sumobot su sé stesso a velocità medio-alta e di fermarlo solamente quando si percepisce di stare per uscire dal ring. In questo caso si comporterà come nella prima versione della strategia “Slow search” in modo da allontanarsi dal bordo. Se mentre si sta girando su se stessi il Lancelot rileva l’avversario, allora caricherà in linea retta con velocità al massimo. Tuttavia, data la velocità elevata con cui il sumobot gira su se stesso, si è rilevato come non riuscisse sempre ad allinearsi con il bersaglio per caricarlo, si è dunque deciso di fargli eseguire una piccola curvatura nel senso opposto in maniera da allinearlo.

La logica che segue è sempre quella della macchina a stati avente Search, Attack, Back e Out e viene cambiato solamente come il Lancelot entra negli stati di Search e Attack.
Questa strategia è stata ideata con lo scopo di avere una maggiore difesa e un potente contrattacco. Qualora il nemico dovesse rilevare il Lancelot e caricarlo per primo, parte dell'energia cinetica dell'impatto verrebbe dissipata a causa del movimento rotatorio del Lancelot, riducendo di molto gli effetti della spinta dell'avversario. Inoltre, oltre a fungere da ottima difesa, il mantenimento di un'alta velocità permette al Lancelot di caricare in risposta l'avversario senza dover vincere l'attrito prodotto da un'accelerazione improvvisa.

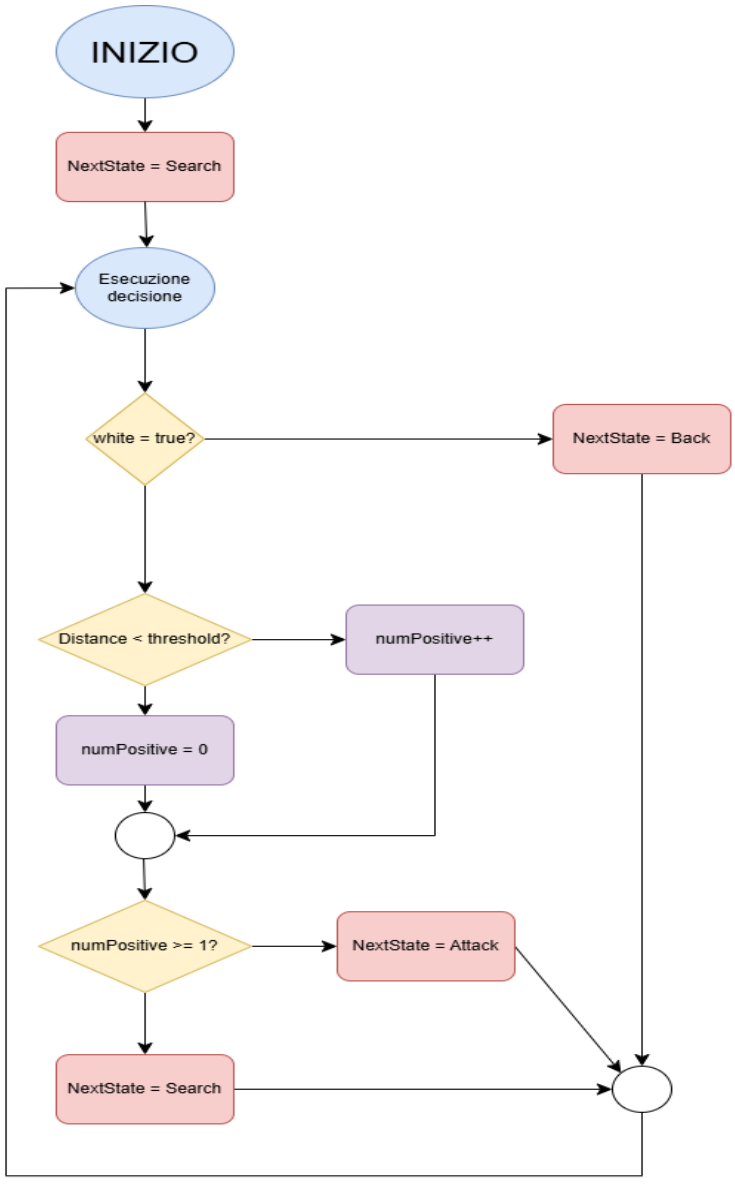


Figura 5: Flowchart della strategia Tornado.

4.3 Tracking

L’ultima strategia presente è la “Tracking”, per la quale è stata fondamentale l’aggiunta dei due sensori TOF in modo da ottenere una visione a ventaglio dello spazio d’avanti al Lancelot. Per riuscire a tenere traccia dell'avversario si è lavorato sul calcolo della distanza da questo da parte dei tre sensori TOF. Nel caso in cui la distanza letta dai sensori di destra o sinistra sia minore della soglia prefissata, il Lancelot effettuerà una rapida rotazione rispettivamente o in senso orario o antiorario; in modo da riuscire ad avere l’avversario esattamente di fronte. Un ulteriore controllo che avviene è se il sensore di destra rileva una distanza minore di quello di sinistra e viceversa; controllo che serve per far ruotare il Lancelot nel senso in cui il nemico è più vicino, in quanto l’altro potrebbe essere un falso positivo.
Nel momento in cui il nemico si trova allineato al suo sensore centrale, la velocità verrà impostata al massimo per passare all'attacco. Nel caso invece in cui il nemico non sia ancora individuato o sia stato perso, allora il Lancelot eseguirà una lenta rotazione su sé stesso fino all'individuazione del nemico.
Anche qui viene sfruttata la logica a stati, in cui viene modificata principalmente la maniera in cui il Lancelot si comporta nella fase di Search.
Questa strategia è stata pensata per far sì che l’avversario non possa fuggire una volta rilevato, situazione che può capitare spesso date le velocità elevate dei sumobot durante gli incontri.

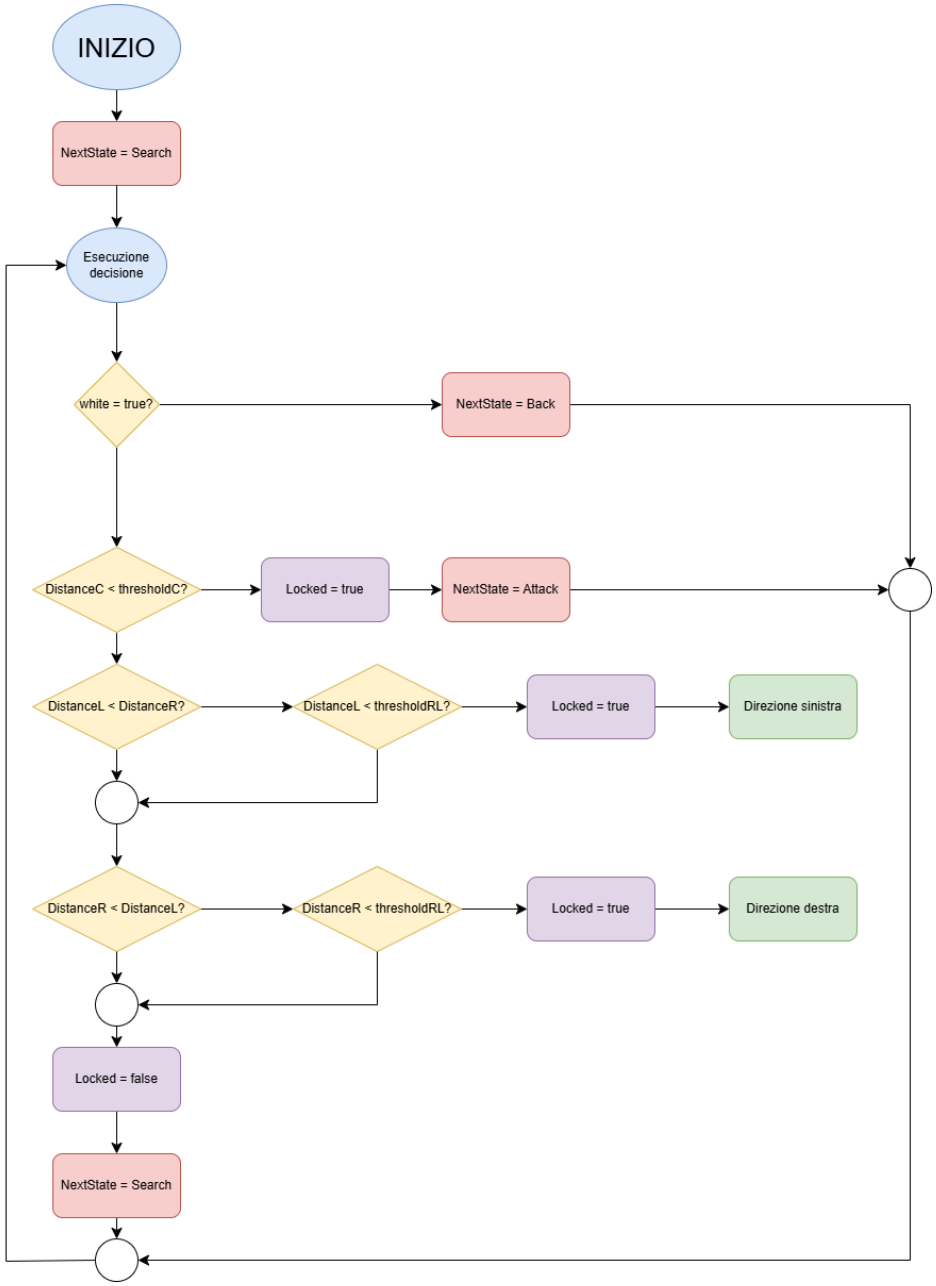


Figura 6: Flowchart della strategia Tracking. Avvengono dei controlli aggiuntivi per la presenza degli altri sensori ToF. Nello stato di “Search” se il bool “Locked” è vero smette di girare su sé stesso ed effettua la curvatura in base alla variabile “Direzione".

CAPITOLO 5
COMPARAZIONE DELLE STRATEGIE

Questo capitolo prevede l'esposizione dei risultati ottenuti dalle diverse strategie in due situazioni differenti. Nella prima i due sumobot saranno il Lancelot (riconoscibile grazie al PCB verde posto sopra) mentre l'altro sarà un modello base spento e senza cingoli per evitare incastri della lamina e il troppo attrito della gomma. Questi inoltre verranno fatti partire in posizioni casuali in modo da simulare una situazione realistica. Nel secondo caso invece sia il Lancelot che l'avversario saranno in movimento, quest'ultimo avrà come strategia la Slow search priva di modifiche. Questa volta i due sumobot verranno fatti partire dal centro dell'arena come da regolamento.
L'intento di questi test è di valutare quale strategia sia la migliore e denotare i possibili punti di forza e critici di queste su cui dover ancora lavorare. I test effettuati vanno da considerarsi solo come punto di riferimento iniziale; nel caso si volessero emulare bisognerebbe considerare la differenza dei modelli, che potrebbe portare a risultati diversi.

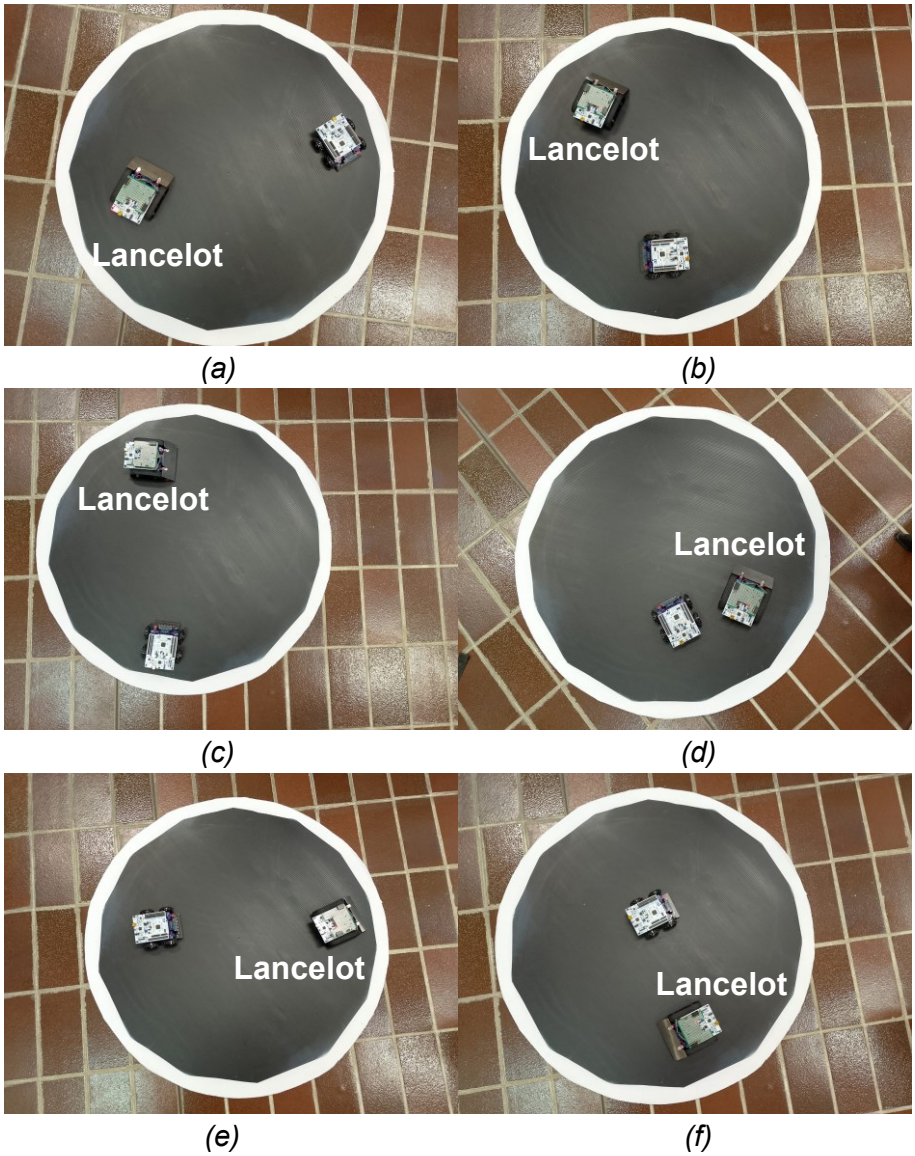
5.1 Setup di misura

In entrambi i casi sono stati effettuati test da massimo 30 s e sono stati riportati su delle tabelle cinque particolari valori:

- Il numero di volte in cui il Lancelot ha percepito di rilevare il nemico, ovvero quante volte è entrato nello stato "Attack".
- Il numero di volte che ha rilevato il nemico e questo era effettivamente presente.
- Il tempo impiegato per rilevare il nemico per la prima volta (vengono considerati anche i falsi positivi).
- L'eventuale tempo impiegato dal Lancelot per vincere (dove non è riportato nulla significa che non ha vinto o pareggiato).
- L'eventuale tempo impiegato dal Lancelot per perdere (dove non è riportato nulla significa che non ha perso o pareggiato).

Per calcolare il numero di volte che il Lancelot individuava il nemico (con i falsi positivi) e il tempo impiegato del Lancelot per effettuare il primo rilevamento, sono stati utilizzati delle variabili interne del sumobot salvate poi nei registri di backup RTC (Real-Time Clock) presenti nella board e realizzati appositamente in modo da poter mantenere memorizzati i dati anche dopo un reset del software.

Nelle tabelle, così come nei grafici, in caso di pareggio è stato associato valore 0 sia al tempo di vittoria che di sconfitta. Inoltre, nei grafici riguardanti il tempo di rilevamento dell'avversario (*Figura 9, Figura 12*) i casi aventi dei falsi positivi sono stati considerati con il valore 0.
Nel primo caso sono stati effettuati dieci test, ognuno dei quali prevedeva una posizione di partenza differente (*Figura 7*). Nel secondo caso invece sono state scelte solamente quattro posizioni di partenza differenti (*Figura 8*) ma per ogni posizione sono stati effettuati quattro test, ottenendo un totale di dodici test.



(a) *(b)*

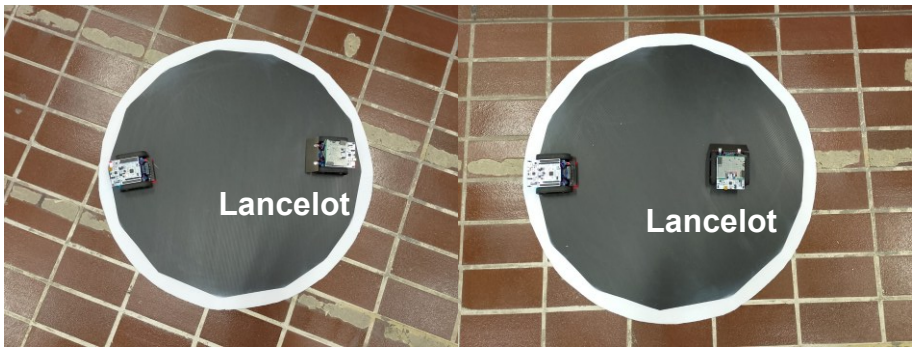
(c) *(d)*

(e) *(f)*

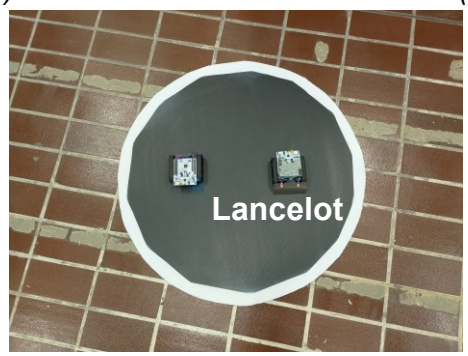
(g) (h)

(i) (j)

Figura 7: Posizioni di partenza del Lancelot con avversario fermo. a) Test 1; b) Test 2; c) Test 3; d) Test 4; e) Test 5; f) Test 6; g) Test 7, h) Test 8; i) Test 9; j) Test 10.



(a) (b)



(c)

Figura 8: Posizioni di partenza del Lancelot con avversario in movimento: a) Test 1-4; b) Test 5-8; c) Test 9-12.

5.2 Caso A, avversario fermo

Nei primi test che sono stati svolti, l'obiettivo è stato quello di verificare quanto le diverse strategie si adattassero in scenari differenti; andando a dare meno importanza a scenari realistici con un avversario in movimento. Si è preferito dunque constatare la velocità di rilevamento dell'avversario e la velocità di vittoria in un ambiente "di laboratorio" per avere una prima visione del comportamento delle diverse strategie.
Di seguito vengono riportati i dati relativi ai vari valori elencati precedentemente per ogni strategia sotto forma di tabella.

Slow search:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	0	0	0	0	0
2	12,872	1	1	13,61	
3	23,598	1	1	24,68	
4	22,59	1	1	23,5	
5	6,335	1	1	6,87	
6	0	0	0	0	0
7	23,995	1	1	24,98	
8	5,335	1	1	5,5	
9	0	0	0	0	0
10	14,488	1	1	15,57	

Tabella 2: Risultati Slow search con avversario fermo.

Il comportamento del Lancelot durante l'uso di questa strategia è stato congruo a quanto ci si aspettasse in maniera teorica: il sumobot vagava per l'arena fino a che non rilevava l'avversario. Una volta rilevato lo caricava tentando di buttarlo fuori.
I rilevamenti, come si vede dalla *Tabella 2*, sono stati sempre accurati, non andando mai ad individuare dei falsi positivi. Tuttavia, diverse volte è arrivato molto vicino all'avversario senza però rilevarlo; problema sorto dall' avere come unico sensore attivo quello centrale e non anche i due laterali.
Si è potuto dedurre quindi che questa strategia sicuramente rappresenta la scelta più solida e sicura; tuttavia, si deve sperare di riuscire a rilevare l'avversario prima che questo rilevi il Lancelot, situazione nella quale potrebbe non avere la meglio.

Tornado:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	4,114	2	0		6,89
2	5,257	1	0		7,04
3	5,167	1	0		6,43
4	2,203	1	1	3,48	
5	12,102	1	0		12,76
6	2,296	1	0		3,48
7	1,24	2	2	4,62	
8	0	0	0	0	0
9	1,324	1	0		2,7
10	2,026	1	1	2,8	

Tabella 3: Risultati Tornado con avversario fermo.

Come si può leggere dai dati della *Tabella 3,* la strategia Tornado sembra essere la più fallimentare in quanto ha perso nella maggior parte dei casi. Nonostante vi sia anche qui solamente il sensore centrale attivo (per evitare problemi mentre gira ad alta velocità), sembrerebbe comunque rilevare il nemico anche quando esso non è presente, finendo così per uscire dall'arena senza accorgersi del bordo bianco. Gli unici casi in cui questa strategia ha avuto esito positivo sono i test: 4, 7, 10 (*Figura 7 (d), (g) e (j)*); ovvero i casi in cui l'avversario si trovava molto vicino al Lancelot. Questo ci può far intuire che tale strategia non risulta efficace come ipotizzato probabilmente a causa di problemi del sensore ToF che, percorrendo distanze troppo grandi durante un moto rotatorio, fornisce dati non affidabili.

Tracking:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	10,681	6	2	17,47	
2	2,234	7	1	16,15	
3	5,456	4	1	12,6	
4	1,273	1	1	2,2	
5	2,256	7	1	8,76	
6	2,853	2	2	3,75	
7	0,655	4	2	2,92	
8	0	0	0	0	0
9	2,611	4	2	7,98	
10	2,052	1	1	2,88	

Tabella 4: Risultati Tracking con avversario fermo.

Durante la strategia Tracking il Lancelot ha avuto un comportamento particolare: nonostante rilevasse il nemico anche dove non presente, non caricava costantemente come nella Tornado, ma effettuava dei piccoli scatti per poi tornare a ruotare lentamente su sé stesso fino a trovare veramente il nemico. L'utilizzo di questa strategia con un avversario fermo, inoltre, non mostra il massimo potenziale del Tracking in quanto il suo punto forte sta nel trovare ed allinearsi costantemente con il nemico, in modo da evitargli la fuga.

Una volta ottenuti questi dati si è potuto usare degli istogrammi per vedere l'andamento delle varie strategie, in particolar modo riguardante la velocità di rilevamento dell'avversario (considerando solo i veri positivi), la velocità di vittoria e la precisione di ogni strategia.
Si può dunque notare come la strategia Slow search sia la più affidabile e precisa in quanto le rilevazioni del nemico sono sempre state corrispondenti ad un vero positivo. Tuttavia si può anche notare come i tempi di vittoria di tale strategia siano in media sempre superiori ai 10 secondi, rendendola affidabile ma poco efficace in una competizione che in media dura solo pochi secondi.
Sebbene la Trackin invece non sia una strategia altrettanto affidabile (avendo solo il 36,1% di affidabilità), si può vedere dalla *Figura 10* come riesca in ogni caso a vincere i vari scontri, notando inoltre come riesca a farlo in tempi nettamente inferiori rispetto alle altre strategie, portandola dunque ad essere una strategia molto performante.

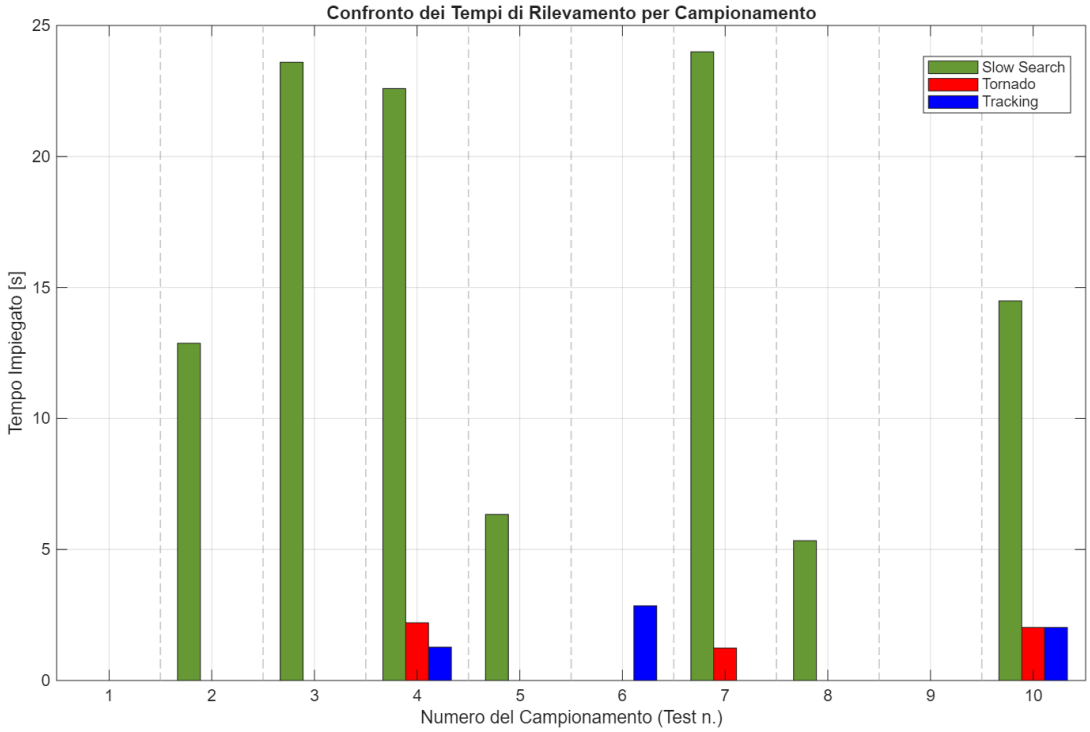


Figura 9: Confronto dei tempi di rilevamento (solo veri positivi) con avversario fermo.

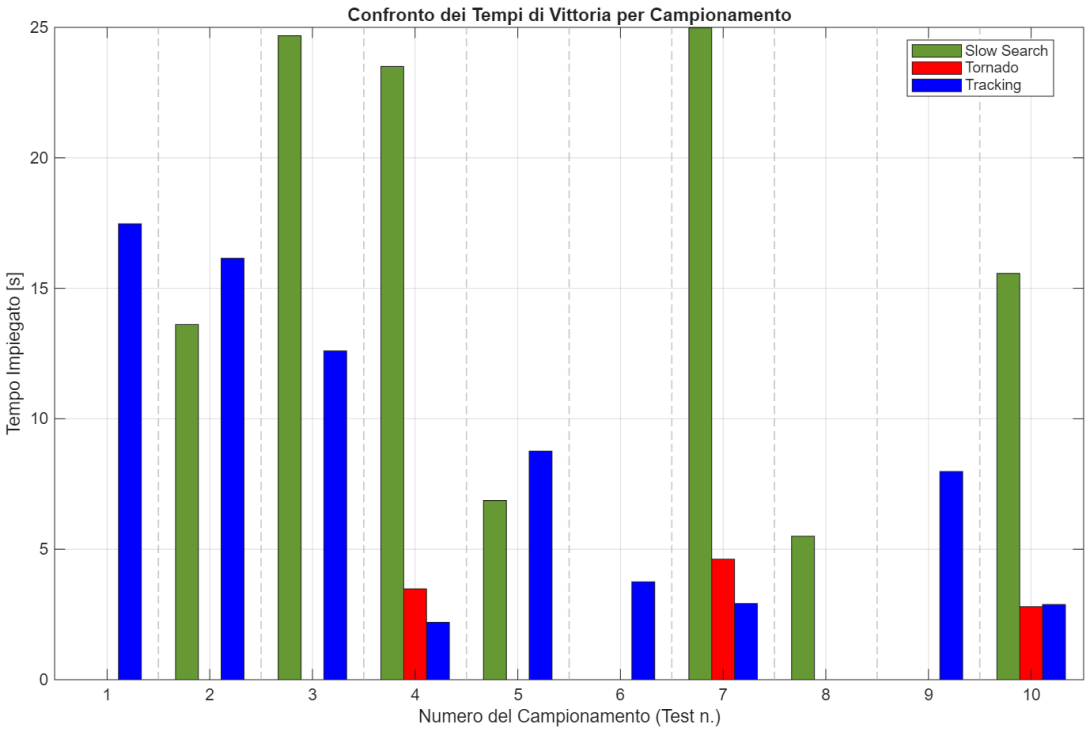


Figura 10: Confronto dei tempi di vittoria con avversario fermo.

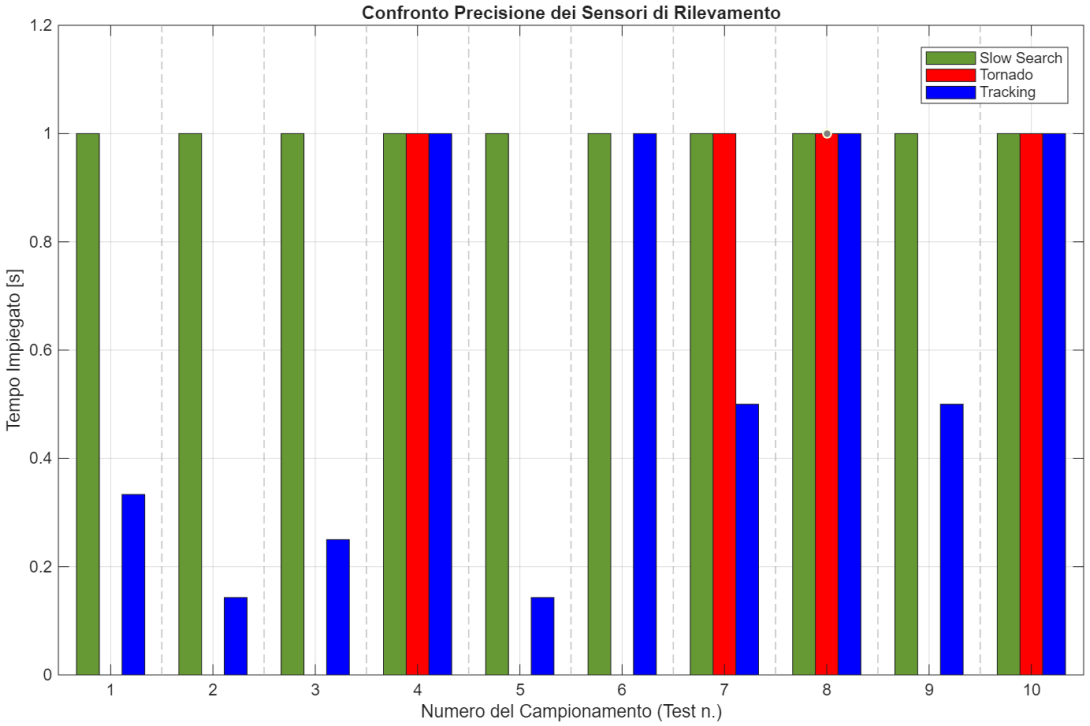


Figura 11: Confronto della precisione dei sensori di rilevamento con avversario fermo.

5.3 Caso B, avversario in movimento

In questo caso invece i test sono stati effettuati cercando di simulare un vero e proprio scontro tra due mini sumobot. Come da regolamento sono stati fatti partire entrambi allo stesso momento e posizionati dietro le linee di partenza come previsto. Poiché il regolamento permette di partire in posizioni diverse, sono state usate tre posizioni di partenza differenti e per ognuna di queste si sono eseguiti quattro test. I test da 1 a 4 sono stati eseguiti secondo la *Figura 8 (a)*, i test da 5 a 8 secondo la *Figura 8 (b)* e i test da 9 a 12 secondo la *Figura 8 (c)*.

La strategia caricata sulla board dell'avversario, come anticipato precedentemente, prevede una semplice macchina a stati simile alla Slow search ma senza modifiche per lo stato "Back". Alcune accortezze adottate per i test sono state la rimozione della lamina davanti all'avversario per evitare eventuali incastri con i cingoli e la diminuzione della sua velocità nella fase di attacco. Tali misure sono state introdotte per focalizzare l'analisi principalmente su come il Lancelot gestisca situazioni in cui l'avversario è libero di muoversi. Ciò ha permesso così di capire il comportamento e l'adattabilità del Lancelot in ambienti dinamici, così da delineare punti di forza o di debolezza in eventuali scontri effettivi. Di seguito sono riportate le tabelle con i diversi valori ottenuti e le immagini delle tre diverse partenze.

Slow search:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	1,927	2	2	3,6	
2	1,752	1	1	3,23	
3	1,224	2	1	7,45	
4	1,576	1	1	4,37	
5	10,721	2	2	13,6	
6	0	0	0		13,32
7	0	0	0		15,6
8	0	0	0		10,98
9	24,124	1	1	0	0
10	4,129	1	1	6,14	
11	4,317	1	1	0	0
12	0	0	0	0	0

Tabella 5: Risultati Slow search con avversario in movimento.

In questo caso, poiché entrambi i sumobot sfruttavano una strategia molto simile, il loro comportamento è stato molte volte speculare; andando a diversificarsi solo nel momento in cui arrivavano al bordo dell'arena, entrando nello stato "Back". Il Lancelot, infatti, effettuava una rotazione in base a quali sensori venivano attivati mentre l'avversario girava sempre in senso orario.

Si può notare dalla *Tabella 5* che l'affidabilità di questa strategia è rimasta molto alta in tutti i vari test, confermando come questa strategia sia altamente affidabile.

Nella prima posizione di partenza si è visto come fosse particolarmente efficace, rilevando subito l'avversario di fronte e buttandolo fuori poco dopo. Nella seconda posizione invece è stato principalmente il Lancelot a finire fuori, soprattutto per provare a caricare il nemico fallendo, eccezion fatta per il test 8, in cui è stato proprio l'avversario a farlo uscire dal Dohyo. Infine, con l'ultima posizione, si è visto quanto le due strategie si muovessero in maniera specchiata, rendendo difficile l'incontro tra i due anche dopo essersi già scontrati una volta.

Tornado:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	1,766	3	2		8,62
2	1,601	3	2		5,65
3	1,59	2	1		4,35
4	1,855	1	0		2,64
5	1,857	2	1		2,99
6	1,735	3	2		5,65
7	1,856	3	2		10,45
8	1,765	1	0		3,3
9	4,061	3	2		9,2
10	3,626	3	2	7,74	
11	1,503	1	0		2,87
12	1,329	1	0		3,43

Tabella 6: Risultati Tornado con avversario in movimento.

La strategia Tornado ha dimostrato, anche in questo contesto, di essere una strategia molto poco efficace, mostrando però un piccolo miglioramento. Durante questi test il Lancelot è riuscito diverse volte ad individuare correttamente l'avversario e a caricarlo. Tuttavia, il problema sorto è la rapida perdita di traccia dell'avversario; tale situazione ha portato il sumobot a tornare a girare su sé stesso fino ad uscire in maniera autonoma dal Dohyo, similmente al caso A.

Tracking:

Num test	Velocità primo rilevamento (s)	Num rilevamenti totali	Num rilevamenti effettivi	Eventuale tempo di vittoria (s)	Eventuale tempo di sconfitta (s)
1	0,476	5	2	7,78	
2	1,442	13	5	18,9	
3	0,998	1	0		3,95
4	0,997	3	1	20,75	
5	1,005	7	5	12,62	
6	1,093	3	1	3,48	
7	0,657	7	3	13,38	
8	1,005	8	2	9,93	
9	4,682	4	1		8,83
10	2,062	4	2	18,05	
11	4,083	1	1	11,32	
12	4,541	1	1	6,58	

Tabella 7: Risultati Tracking con avversario in movimento.

La strategia Tracking ha avuto comportamento simile a quello del caso A, individuando l'avversario anche quando questo non era presente, ma portando comunque il Lancelot ad uscire solamente due volte dal Dohyo per tale motivo. La maggior parte delle volte, infatti, è riuscito a tornare allo stato di ricerca fino a trovare correttamente l'avversario e a spingerlo oltre il perimetro del ring. Come visibile dalla *Tabella 7*, i tempi di vittoria questa volta sono aumentati e soprattutto sono diventati maggiori dei tempi di vittoria della Slow search. Tuttavia, grazie ad un tasso di vittoria pari al 83%, risultato nettamente migliore in confronto al 50% della Slow search, rimane una strategia da considerare decisamente efficiente.

Osservando i grafici possiamo avere una migliore panoramica sulle prestazioni delle tre strategie con un avversario in movimento. Come già anticipato la Slow search si presenta come la più precisa e stavolta anche la più veloce, soprattutto per quanto riguarda i casi con la partenza iniziale direttamente di fronte all'avversario (*Figura 8 (a)*). Da notare inoltre come, rispetto al caso A, l'andamento nella *Figura 12* e nella *Figura 13* della Slow search non sia più così conforme. Questa diversificazione, dovuta principalmente a un maggior tempo impiegato dal sumobot per spingere fuori l'avversario dopo averlo individuato; questa dinamica è causata sia dalla resistenza opposta dall'avversario, sia da repentine e temporanee perdite di tracciamento da parte del Lancelot durante la spinta stessa. Altre motivazioni sono la

presenza nell'avversario di cingoli di gomma che generano maggiore attrito ed eventuali incastri tra la lamina del Lancelot e questi.

Sebbene la Tracking abbia perso il primato di strategia più veloce, grazie al suo tasso di vittoria molto alto in tutte le situazioni, si è dimostrata particolarmente efficace e con alta adattabilità.

Mettendo a confronto la *Figura 11* e la *Figura 14* possiamo notare come in quest'ultimo ci sia una minore precisione da parte di tutte le strategie, risultato abbastanza prevedibile in quanto con un avversario in movimento aumenta il tempo di ricerca, aumentando così anche la probabilità di generare falsi positivi.

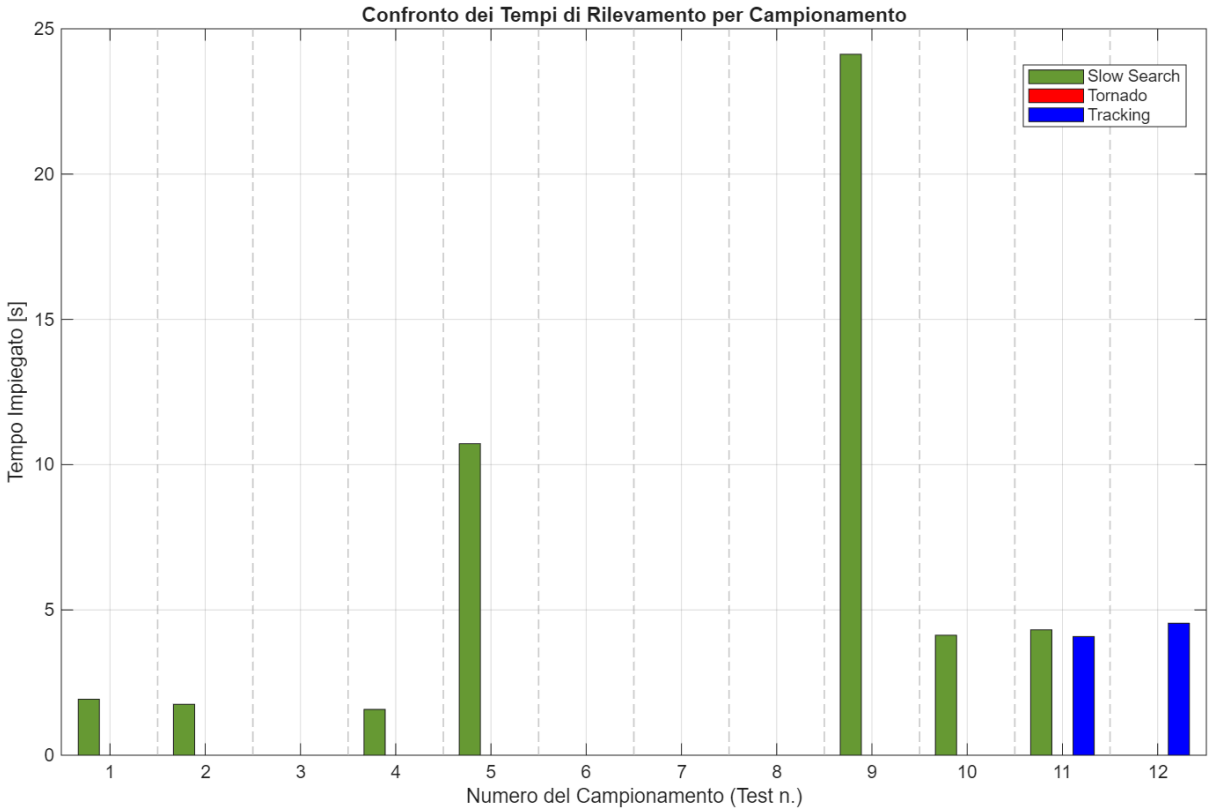


Figura 12: Confronto dei tempi di rilevamento (solo veri positivi) con avversario in movimento.

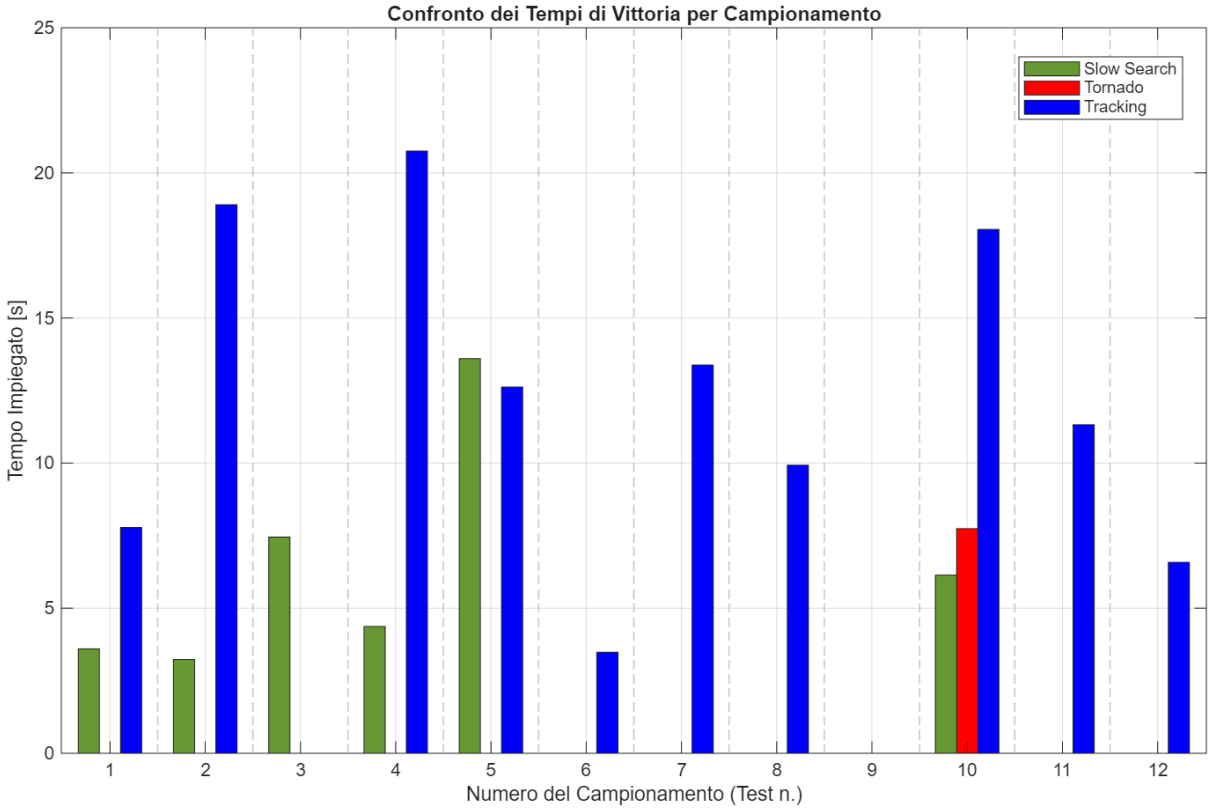


Figura 13: Confronto dei tempi di vittoria con avversario in movimento.

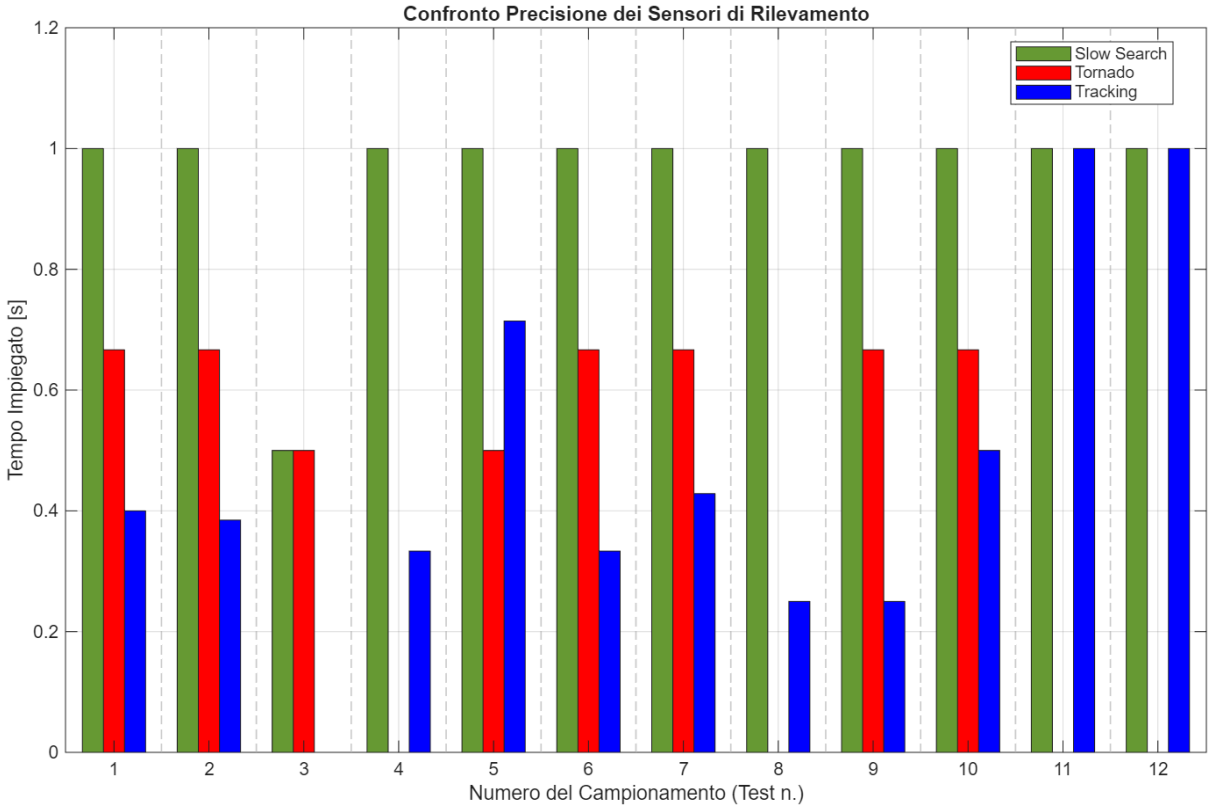


Figura 14: Confronto della precisione dei sensori di rilevamento con avversario in movimento.

CAPITOLO 6
MAPPATURA DELL'AVVERSARIO

Poiché negli scontri si è visto quanto importante fosse la percezione dell'ambiente intorno e soprattutto quella dell'avversario, si è pensato di realizzare una funzione in grado di tenerne traccia durante lo scontro. Questa funzione è stata realizzata con lo scopo di simulare l'effetto di una triangolazione, tecnica nella quale si sfruttano due o più sensori (generalmente tre) i cui assi di rilevamento convergono sul bersaglio. In questo modo, incrociando le distanze misurate da ciascun sensore, è possibile associare all'avversario delle coordinate spaziali relative all'interno del Dohyo. Poiché la posizione dei sensori del Lancelot impediva l'applicazione diretta della triangolazione classica, in quanto posizionati a ventaglio e non a sovrapposizione, si è realizzata una tecnica basata su una mappa di celle.
Tale approccio prevede una discretizzazione dello spazio davanti al Lancelot tramite una griglia radiale virtuale suddivisa in celle. Ognuna di queste possiede due coordinate particolari; la prima (che rappresenterebbe la x) è definita in base a quali sensori rilevano il sumobot avversario, consentendo così di creare sei possibili opzioni: Sinistra, Centro sinistra, Centro, Centro destra, Destra e Out. La seconda coordinata (la controparte della y) viene invece definita dalla distanza del target dal sensore, considerando un intervallo di 5 cm che ogni cella possiede e in cui potrebbe trovarsi l'avversario.

Tramite questa tecnica, una volta individuato l'avversario, il Lancelot provvede a posizionarlo in una delle celle della griglia, seguendo come criterio le due variabili caratteristiche di ogni cella:

- Inizialmente il sistema verifica quali sensori hanno rilevato il target, in modo da poter capire in quale posizione delle ascisse porlo. Tale procedimento è implementato mediante la funzione "TrackEnemy()" definita in "Tracking.cpp", questa controlla tutti i casi possibili delle risposte dei sensori, associando al sumobot avversario la sua posizione rispetto al Lancelot.
- Successivamente il sistema analizza la distanza dell'avversario rilevata dai diversi sensori in modo da poterlo piazzare nell'intervallo corrispondente. Nei casi in cui il target si trovi nelle posizioni di Sinistra, Centro o Destra, la posizione viene mantenuta uguale a quella letta dal relativo singolo sensore. Se invece si dovesse trovare nelle posizioni di Centro sinistra o Centro destra; allora si dovrà calcolare la mediana del triangolo formato dagli assi dei due sensori interessati e l'avversario, che parte dal Lancelot e cade sul lato opposto. Per trovare tale misura si sfrutta il teorema di Apollonio che consente di trovare la mediana avendo tutti e tre i lati. Tuttavia, si può trovare il terzo lato (il segmento che congiunge le letture dei sensori) usando il teorema

di Carnot (o del coseno) che, grazie alla lunghezza dei due lati e l'angolo tra loro, restituisce proprio il terzo lato del triangolo. Mettendo insieme i due teoremi si ottiene la seguente formula:

$$\frac{\sqrt{a^2+b^2+2ab*cos(\alpha*(3{,}14/180))}}{2}$$

Dove a è la distanza letta dal sensore centrale, b la distanza letta dal sensore destro o sinistro (in base al caso) e α è l'angolo dato di 20° tra i due sensori che viene poi convertito in radianti.

Altra funzione molto importante per questa tecnica è la funzione "getPositionInGrid" descritta nel modulo "Strategy.cpp". Tale funzione riceve come parametri di input le varie distanze lette dai sensori insieme alla posizione in cui è stato ubicato l'avversario e procede a posizionare quest'ultimo nella cella corretta tramite una serie di cicli while e for, restituendo il puntatore di tale cella. Nel caso la posizione dell'avversario sia Out il sistema restituisce un puntatore nullo (nullptr), altrimenti restituisce il puntatore della cella in cui è stato posizionato l'avversario impostando ad 1 il flag intero enemySpotted.
Per la digitalizzazione della griglia radiale si è realizzata una matrice 15×5 composta da variabili struct denominate "Cella". Tale struct prevede:

- Un intero per la posizione sulle ascisse (casting esplicito della struttura enum "EnemyPosition" ad int).
- Due variabili di tipo int32_t per delimitare la distanza massima e minima del range della cella.
- Un intero con funzione di flag per definire se in quella cella è situato l'avversario (0 non presente, 1 presente).

L'utilizzo di un intero invece di un normale booleano fornisce una maggiore chiarezza nella stampa di debug della griglia sul monitor seriale.
La griglia viene creata ed inizializzata nella fase di Start nel modulo "application.cpp" tramite la funzione "CreateGrid()" e, prima di ogni aggiornamento, viene resettata.
Sulla base di questa architettura si potrebbe anche realizzare, tramite un sistema di incremento temporale dei nuovi valori inseriti, un modo per poter tenere traccia del percorso compiuto nel tempo dell'avversario, previo annullamento della routine di azzeramento sistematico della griglia a ogni aggiornamento.

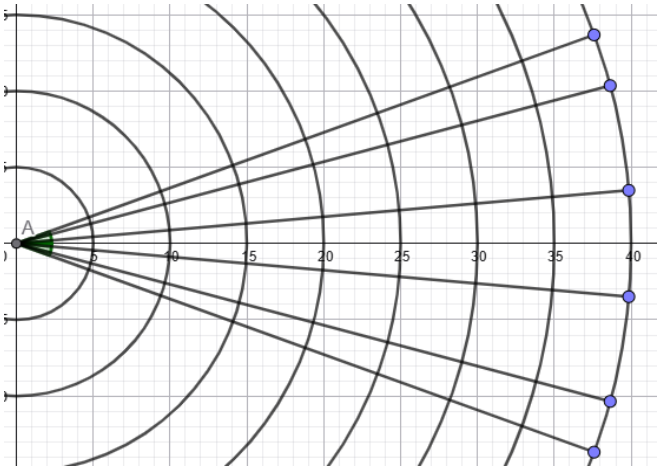


Figura 15: Versione semplificata della griglia radiale per la mappatura dell'avversario.

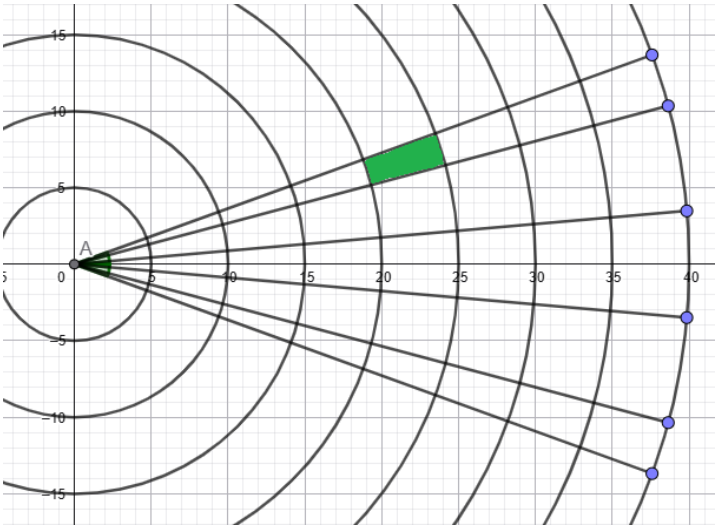
Come si può vedere nella *Figura 15*, il Lancelot è stato approssimato a un punto (punto A) e per tale motivo le distanze tra i sensori sono state approssimate a 0. Anche gli angoli sono stati approssimati, infatti i segmenti interni sono stati realizzati considerando angoli di 15° e 5° con l'asse centrale mentre i due esterni sono rimasti fedeli ai 20°.

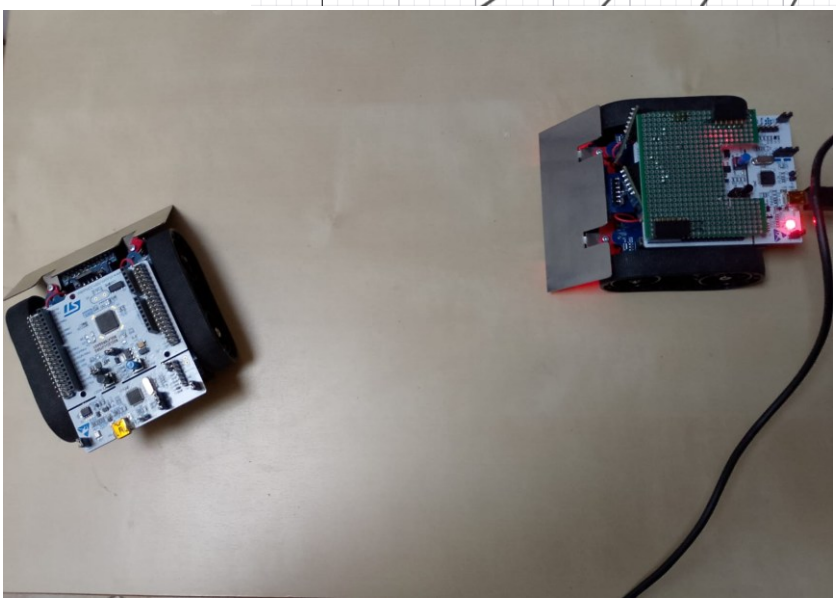
Successivamente vengono riportati degli esempi del funzionamento di tale tecnica in cui vengono mostrati: la matrice di debug, la griglia radiale e la foto reale per rappresentare alcuni esempi. Dove la cella è colorata in verde sta ad indicare la cella in cui è stato posizionato il sumobot avversario rilevato.

Esempio 1:

Il robot avversario si trovava sul lato sinistro ad una distanza di 22,3 cm.

```
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[1] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
--------------------
```

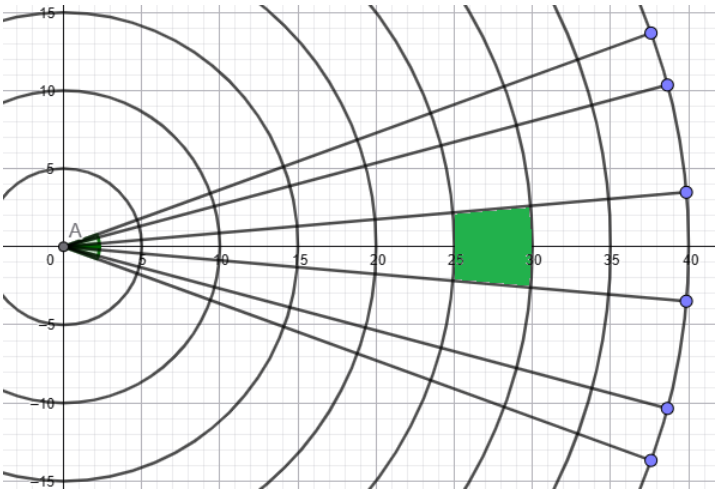


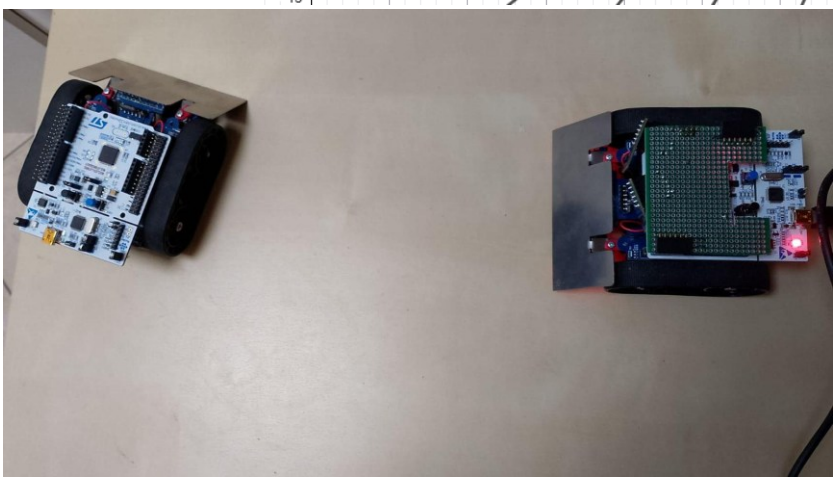


Esempio 2:

Il robot avversario si trovava allineato al centro ad una distanza di 26,7 cm.

```
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [1] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
--------------------
```

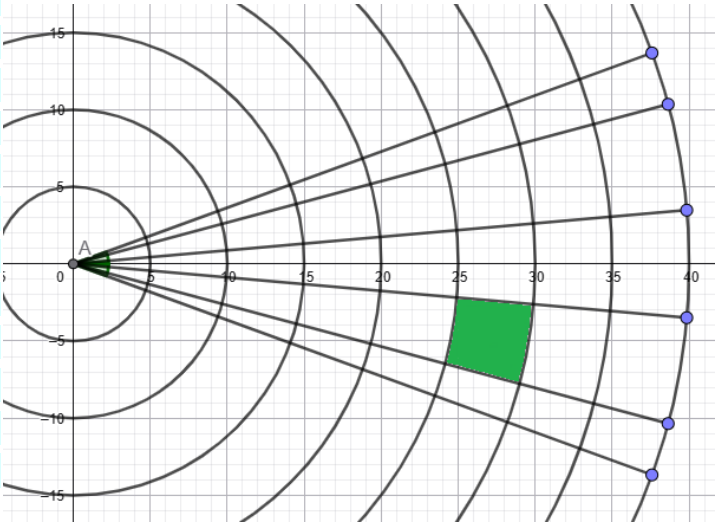


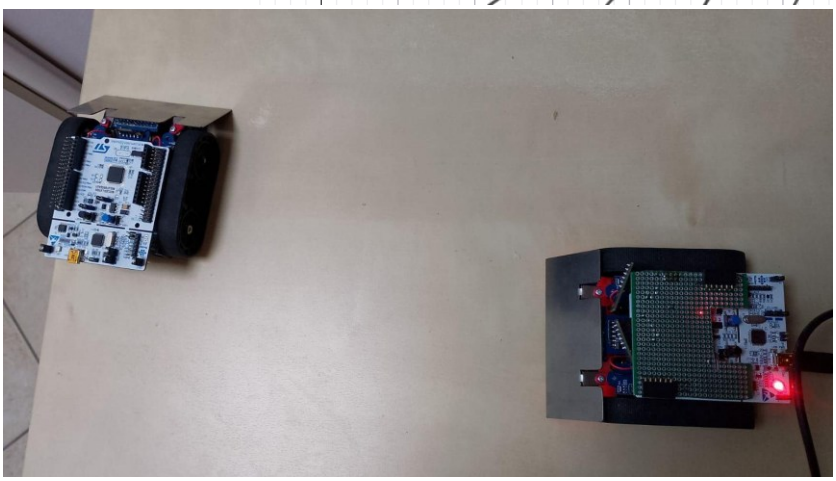


Esempio 3:

Il robot avversario si trovava sul lato centro-destra ad una distanza dal centro di 27,5 cm e di 28,2 cm dal sensore sinistro.

```
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [1] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
[0] [0] [0] [0] [0]
--------------------
```





CONCLUSIONI

Andando ad osservare i risultati ottenuti si può quindi dedurre come lo sviluppo software e hardware apportato al Lancelot abbiano incrementato di molto le sue potenzialità strategiche e le probabilità di vittoria.

L'analisi effettuata nei diversi scenari di test ha evidenziato i punti di forza e le criticità di ciascuna strategia implementata. La strategia *Slow search* si è confermata la scelta più accurata in termini di affidabilità dei sensori, sebbene sia penalizzata da tempi di vittoria maggiori. Al contrario, la strategia di *Tracking*, grazie all'introduzione della visione a ventaglio garantita dai sensori TOF aggiuntivi, ha dimostrato un'eccellente reattività e il più alto tasso di vittoria complessivo, compensando la minore precisione iniziale dovuta a falsi positivi. La strategia *Tornado*, invece, pur offrendo in maniera teorica una valida difesa cinetica, ha mostrato forti limiti soprattutto nella rilevazione dell'avversario, ricca di falsi positivi, andando a posizionarsi come strategia meno efficiente.

Si è dunque potuto vedere quanto l'implementazione di sensori aggiuntivi abbia portato il Lancelot ad avere maggiore consapevolezza dell'ambiente circostante e a strategie maggiormente efficaci.

Inoltre, la realizzazione di strategie basate sull'architettura a classi ereditarie ha conferito al codice un'elevata scalabilità. Questo approccio software ha permesso non solo di isolare i comportamenti specifici di ogni strategia, ma anche di garantire una notevole flessibilità per l'integrazione futura di nuove possibili strategie.

Un altro importante contributo è stato l'implementazione della funzione di mappatura dell'avversario tramite griglia radiale. Si è riusciti a sfruttare la disposizione a ventaglio dei sensori ToF in maniera tale da realizzare un sistema di localizzazione discreta capace di determinare la posizione dell'avversario in tempo reale; rendendo così il Lancelot ancora più consapevole dell'ambiente intorno e dell'avversario.

In fine, a differenza dei modelli teorici o delle simulazioni digitali spesso presenti in letteratura, il confronto delle strategie in un ambiente reale ha permesso di affrontare e risolvere problematiche concrete come l'attrito dei cingoli, i falsi positivi generati dal movimento rotatorio e gli incastri delle lamine frontali.

Sviluppi futuri

Il mondo dei sumobot è molto vasto e dinamico, così come le possibili migliorie che si potrebbero apportare al Lancelot. Per prima cosa si potrebbe pensare ad una fusione tra la strategia Slow search e Tracking in modo da ottenere la precisione della prima e il tasso di vittoria della seconda. Successivamente si potrebbe andare ad implementare un algoritmo che sfrutti la logica fuzzy oppure, volendo implementare le reti neurali, si potrebbe utilizzare la fuzzy per generare simulazioni virtuali da usare come database per allenare la rete neurale. Infine, un ulteriore possibile sviluppo è l'utilizzo della tecnica di mappatura con associati algoritmi predittivi in maniera da poter prevedere le mosse dell'avversario ed adattarsi di conseguenza con la strategia migliore.

Bibliografia

1. B. G. Antunes, L. F. F. P. Bezerra, A. L. C. Canella and M. F. Pinto, "Development of an Autonomous Robot for Competitive and Educational Purposes in the Sumo Category," 2024 Brazilian Symposium on Robotics (SBR) and 2024 Workshop on Robotics in Education (WRE), Goiania, Brazil, 2024, pp. 209-214, doi: 10.1109/SBR/WRE63066.2024.10837592. keywords: {Technological innovation;Codes;Conferences;Education;Autonomous robots;Mini Sumo;Robotics;Education;Competition;STEM},
2. Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 5, 161-165 (2018)
3. DELEA, Cosmin, Sabin-Mihai GÎNȚA, and Norbert-Andras LOVASZ. "A STUDY CONCERNING THE USE OF ELECTRONICALLY COMMUTATED MOTORS FOR SPECIAL APPLICATIONS ROBOTS."
4. https://www.makerslab.it/olimpiadi-robotiche/
5. J. B. . Vera Vera, C. A. Moreira Zambrano, Y. S. Loor Vera, y J. E. Navarrete Solano, «Eficiencias en algoritmos y estrategias de ataque en robots mini sumo. Perspectiva desde la IA», REVISTA UNTELS, vol. 4, n.º 2, sep. 2024.
6. E. de Souza Rodrigues, A. G. L. dos Santos, L. A. Braga and T. C. Revoredo, "Ultra-T: UERJBotz's Mini Autonomous Sumo Robot," 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE), Vitoria, Brazil, 2025, pp. 261-266, doi: 10.1109/SBR/WRE66973.2025.11249662. keywords: {Knowledge engineering;Fabrication;Educational robots;Laboratories;Entrepreneurship;Robot sensing systems;Search problems;Teamwork;Engineering education;Programming profession;Sumo robots;Robot competitions;Engineering education},
7. Derian Joseph Guevara Riofrío, UNIVERSIDAD POLITÉCNICA SALESIANA SEDE GUAYAQUIL CARRERA DE MECATRÓNICA DESARROLLO DE ROBOT MICRO SUMO AUTÓNOMO PARA COMPETENCIAS DE ROBÓTICA PROFESIONAL http://dspace.ups.edu.ec/handle/123456789/30427
8. M. S. Mohd Shukri, N. Abdul Rahim, and B. Ilias, “Development of an Intelligent Sumo Robot based on Embedded Fuzzy Logic”, IJARIS, vol. 1, no. 1, pp. 93–105, Aug. 2025.

9. J. Binfet and B. M. Wilamowski, "Microprocessor implementation of fuzzy systems and neural networks," IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), Washington, DC, USA, 2001, pp. 234-239 vol.1, doi: 10.1109/IJCNN.2001.939023. keywords: {Microprocessors;Fuzzy systems;Neural networks;EPROM;Fuzzy control;Microcontrollers;Clocks;Frequency;Program processors;Rough surfaces},
10. A. Melingui, R. Merzouki and J. B. Mbede, "Neuro-fuzzy controller for autonomous navigation of mobile robots," 2014 IEEE Conference on Control Applications (CCA), Juan Les Antibes, France, 2014, pp. 1052-1057, doi: 10.1109/CCA.2014.6981474. keywords: {Navigation;Mobile robots;Robot sensing systems;Fuzzy logic;Uncertainty;Pragmatics},
11. Hamit Erdem, Application of Neuro-Fuzzy Controller for Sumo Robot control, Expert Systems with Applications, Volume 38, Issue 8, 2011, Pages 9752-9760, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.02.024.
12. ST TOF VL53L4CX, https://www.st.com/en/imaging-and-photonics-solutions/vl53l4cx.html
13. STM32 Nucleo L476RG, https://www.st.com/en/evaluation-tools/nucleo-l476rg.html
14. robot base, https://www.am-microsystems.com/