ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Dipartimento di Informatica - Scienza e Ingegneria
Corso di Laurea in Ingegneria e Scienze Informatiche

AlpaBot: Costruzione e impiego di un robot autonomo per scopi didattici

Elaborato in

Sistemi Embedded e Internet-Of-Things

Relatore
Prof. Andrea Roli

Presentata da
Alessandro Lombardi

Anno Accademico 2018/2019

Indice

Desidero ringraziare tutti coloro che hanno partecipato direttamente alla realizzazione di questo progetto, i colleghi e amici Paolo Baldini, fidato compagno di gruppo, e Andrea Giulianini, che ha messo a disposizione la propria stampante 3D. Un ringraziamento particolare va al professor Andrea Roli, che ha creduto fin dall'inizio in questa idea. Non potrei fare a meno di ringranziare anche la Biblioteca Centrale del Campus di Cesena per l'impeccabile servizio.

Infine, vorrei dedicare un affettuoso ringraziamento anche alle persone più care e vicine, ovvero la mia famiglia e i miei amici, per la loro presenza costante e il loro fondamentale supporto spirituale.

Introduzione

La seguente tesi tratta della costruzione e dell'impiego di AlpaBot, un robot mobile utilizzabile per scopi didattici. L'evoluzione del seguente progetto ha avuto continui cambiamenti, dovuti non solo all'inesperienza con molti concetti, appresi solo durante le varie fasi dello sviluppo, ma anche per la natura eterogenea e complessa del lavoro. Al principio, l'interesse maggiore era quello di approciarsi alla *Swarm Intelligence*, realizzando un robot economico e di piccole dimensioni. La Swarm Intelligence è una branca dell'intelligenza artificiale che studia e progetta l'intelligenza collettiva di sistemi auto-organizzati e decentralizzati. Si tratta di una disciplina ispirata dal mondo naturale, in particolare dal comportamento degli insetti sociali, ma anche dalle strategie con cui stormi di volatili e banchi di pesci si muovono, arrivando a includere anche lo studio delle colonie batteriche. L'applicazione di questi concetti alla robotica prende il nome di *Swarm Robotics*. La Swarm Robotics offre non solo la possibilità di mettere in atto algoritmi di Swarm Intelligence per scopi pratici, che spaziano dall'ambito militare a quello industriale o agricolo, ma sottoporre le ricerche di questa disciplina in delle condizioni tuttora oggi considerate complesse se non proibitive. Un altro aspetto invitante della Swarm Robotics è la palpabile sensibilità ai costi per la costruzione di un singolo robot, aspetto che fin dall'inizio fu considerato vitale per la realizzazione del progetto, in quanto le spese sarebbero state personali. Nella fase di ricerca delle varie tipologie di movimento è ancora evidente questo orientamento. Successivamente l'obiettivo si è spostato maggiormente verso la realizzazione di una piattaforma prototipica modulare sempre orientata all'ambito didattico e aperta alla Swarm Intelligence, ma anche a scenari diversi. Le motivazioni di questa scelta sono scaturite sia dal fatto che uno swarm di robot sarebbe stato troppo costoso e difficile da produrre in breve tempo, sia da un crescente interesse verso un'implementazione più general purpose, che non fosse troppo vincolata alla Swarm Robotics. In conclusione, la tesi si sofferma principalmente sulle parti svolte in autonomia, ma il progetto è complessivamente frutto di un lavoro di due persone, che ha messo alla prova anche le abilità di condivisione e gestione delle risorse intelletuali e materiali da parte di entrambi.

- Il capitolo 1 elenca le ricerche iniziali sullo stato dell'arte delle tecnologie maggiormente impiegate per muovere i sistemi autonomi, elencando numerosi esempi.

- Il capitolo 2 descrive lo sviluppo del robot diviso in una parte concernente la costruzione meccanica e l'apparato locomotivo e l'altra l'alimentazione.
- Il capitolo 3 descrive alcune implementazioni software per AlpaBot legate alle principali famiglie di architetture di controllo.

1 Movimento

L'obiettivo di questa parte è duplice: da una parte vi è la volontà di mostrare alcune soluzioni studiate e spesso adottate dai ricercatori che negli ultimi decenni hanno svolto ricerche da tutto il mondo, dall'altra quello di analizzare i vantaggi e gli svantaggi dei modelli già esistenti, per trarre delle ispirazioni utili e interessanti. Il movimento è una attività comune in quasi tutte le specie animali, da quelle più semplici a quelle più complesse, e definisce in particolar modo i comportamenti legati alla soppravvivenza, lo stile di vita e la dieta. Anche in questo caso, la natura si presenta come una grande fonte di ispirazione, proponendo numerosi modelli, talvolta molto differenti fra loro a causa degli adattamenti che le specie animali hanno evoluto nei diversi ambienti presenti nel pianeta Terra. Nelle prossime pagine sarà trattato solo il movimento su terra, in quanto lo studio del movimento acquatico e aereo sarebbe molto interessante ma anche troppo vasto e non permetterebbe di soffermarsi in modo più efficace su un tema già ricco di dettagli e sfide ancora aperte. Inoltre, prevedendo la costruzione di un robot per scopi prettamente didattici, sarà considererata opzionale la versatilità del movimento su terreni accidentati o scoscesi. Prima di proseguire nel dettaglio di questo capitolo, può tornare utile presentare un esempio concreto di fusione del mondo della robotica con quello della biologia: la robotica *BEAM (Biology, Electronics, Aesthetics and Mechanics)*. Questa branca della robotica, sviluppata nella seconda metà degli anni ottanta da Mark Tilden, si propone come modello alternativo alle concezioni più tradizionali, diffondendo un proprio insieme di principi, fondati sullo studio delle caratteristiche e dei comportamenti degli organismi biologici [21]. I robot BEAM sono autonomi seppur non siano concepiti per essere dotati di un microcontrollore, in quanto tendono a imitare la naturale risposta agli stimoli ambientali spontaneamente svolta da numerosi esseri viventi, simulando con semplici circuiti analogici le reti neuronali del mondo naturale. I robot BEAM sono in grado di muoversi adottando soluzioni bizzarre ma funzionali, possono seguire fonti di luce e interagire in maniera più o meno attiva con l'ambiente. Per il momento non hanno una implementazione pratica evidente, neppure un legame stretto con la Swarm Robotics, ma si presentano come validi punti di riferimento e ispirazione.

1.1 Tecnologie

1.1.1 Ruote e cingoli

Un approcio immediato che non trova un riscontro immediato nel mondo naturale, in grado però di offire soluzioni anche ottimali per certi problemi, sfrutta un'invenzione molto antica dell'uomo: la ruota. I primi veicoli intelligenti, costruiti già a partire degli anni cinquanta, come la "Walter's Tortoise", fino ad arrivare ai più recenti e complessi rover prodotti dalla NASA per le esplorazioni planetarie, utilizzano le ruote per muoversi nel terreno [5].

Stato dell'arte Esemplari di robot su ruote sono numerosi, in particolar modo all'interno degli ambienti di ricerca di intelligenza collettiva, come dimostra la serie di robot costruiti a partire dagli anni novanta presso la Ecole Polytechnique Fédérale de Lusanne (EPFL), sotto la supervisione del professore di intelligenza artificiale e robotica Francesco Mondada. Fra i modelli più recenti è possibile trovare il Thymio [30] e l'e-puck [28], mentre di origini più datate il Khepera, considerato un punto di riferimento per la comunità scientifica, per via delle migliaia di pubblicazioni che lo citano e per l'ampia diffusione avuta fra i laboratori e le università di tutto il mondo [29]. Thymio, realizzato in collaborazione con la Lusanne Art School (ECAL), è un prodotto pensato appositamente per avvicinare i bambini al mondo della robotica. Khepera è invece un robot professionale, inzialmente utilizzato per studiare le reti neurali, in seguito fu commercializzato e dal 1995 è prodotto dalla K-Team. Dal successo di Khepera venne realizzato Koala, che ha sei ruote, e venne costruito per operare in ambienti reali con terreni difficili. Un robot di dimensioni e capacità estremamente ridotte, ideale per essere prodotto in grandi quantità per studi di Swarm Robotics, è Alice progettato a partire dalla fine degli anni novanta fino a inizio duemila e presentato da un laureando della EPFL [11]. Contributi più recenti, competitivi a livello economico, provengono dal Regno Unito, in particolare dalla University of Lincoln, con la progettazione di Colias [4] un robot su ruote economicamente accessibile, e dalla Manchester University dove nel 2016 venne sviluppato Mona [3] [31], un robot completamente open source. Fra i progetti distribuiti con licenze open source è degno di nota anche Jasmine [34] realizzato presso la University of Stuttgart.

Osservazioni L'elenco dei robot precedentemente presentati sarebbe molto più lunga, ad esempio mancano AMiRO, r-one e tanti altri, ma sono evidenti, talvolta in maniera eccessiva, le somiglianze e le caratteristiche in comune fra loro. La continua ricerca e lo sviluppo di robot di questo tipo è altresì mossa dalla passione e dall'interesse scientifico, ma ultimamente è chiaro anche l'obiettivo di realizzare piattaforme sempre più moderne e accessibili. L'utilizzo di due ruote motrici permette con pochi sforzi, sia economici che progettuali, un movimento omnidirezionale su una superficie piana, inoltre la disposizione delle ruote e la complessità di queste permettono soluzioni creative in grado di migliorare la mobilità di semplici prototipi di tipo tabletop. Anche l'utilizzo dei cingoli, molto comune in ambito militare, può dotare il robot di una maggiore navigabilità nei terreni accidentati. La realizzazione di robot su ruote fa solitamente uso di motori elettrici, reperibili in commercio in un'ampia fascia di scelta.

1.1.2 Vibrazione

Un metodo più sofisticato sfrutta le vibrazioni prodotte dai piccoli motori in corrente continua, che trovano facilmente impiego negli apparecchi telefonici, per segnalare le notifiche, oppure negli spazzolini da denti elettrici, per muovere vorticosamente le spazzole. In commercio sono presenti diversi modelli, che si differenziano per l'ambito applicativo e il funzionamento interno. Alcuni presentano una forma circolare, e sono per via della forma chiamati Coin Vibration Motors o "Pancake Vibrator Motors", mentre altri hanno una struttura cilindrica, e sono comunemente detti "Pager Motors", dal nome dell'apparecchio in cui erano frequentemente utilizzati, o più formalmente Eccentric Rotating Mass Vibration Motors (ERM Motors). Le vibrazioni prodotte da un singolo motore possono essere utilizzate per spostare piccoli robot su una superficie piana, quelle di più motori permettono anche di cambiare la direzione. Per dimostrare l'efficacia di questo metodo basta porre sopra alla testa di uno spazzolino da denti un motore vibrante e una batteria in grado di alimentarlo e osservare come questo oggetto, detto simpaticamente brushbot, sia capace di muoversi freneticamente.

Coin Vibration Motors I Coin Vibration Motors sono integrati in molte applicazioni perché non hanno parti mobili e sono di dimensioni estremamente ridotte, special-

mente per le interfacce aptiche. Un esempio molto interessante che fa uso di questo genere di motori è il Kilobot [36], uno swarm robot sviluppato alla Harvard University che ha avuto un grande successo nella comunità scientifica per via della effettiva adoperabilità in esperimenti su larga scala riguardanti temi di collective artificial intelligence e swarm behaviours. Inoltre i Kilobot si presentano come una soluzione low-cost [16] e dal design open source, mettendo a dispozione a ricercatori e professori di tutto il mondo un ambiente completo per programmare, testare e configurare velocemente vasti gruppi di agenti. Il movimento avviene grazie a due motori vibranti, che si trovano in posizione opposta e che azionati in determinate combinazioni permettono al robot di muoversi in avanti e ruotare in tutte le direzioni. Un altro robot che sfrutta questa tecnica è il Droplet, un robot open source e low-cost, sviluppato presso la Corell Lab della University of Colorado sempre per la ricerca nell'ambito della Swarm Robotics.

ERM Motors Gli Eccentric Rotating Mass Motors sono dei motori in corrente continua che presentano all'estremità del rotore una massa asimmetrica al centro di rotazione. La rotazione del braccio con tale massa posta all'estremità genera una forza centrifuga in grado di agitare fortemente il motore, provocando, nelle piccole dimensioni, quello che gli umani percepiscono come una vibrazione. Oltre alla possibilità di essere utilizzati per produrre una vibrazione distribuita in grado di muovere le gambe del robot, come nei casi precedentemente trattati, possono essere adoperati, privandoli della massa, a contatto diretto con il terreno. Questa soluzione fa compiere piccoli sobbalzi al robot permettendo questo di muoversi, in aggiunta, possono essere fissate alle estremità dei cuscinetti, delle ruote o altri materiali elastici, per diminuire l'attrito con il terreno. Un esempio commerciale rivolto al mondo dell'educazione che fa uso dei Motori ERM poggianti direttamente a terra, è Ringo [15]. Ringo è un robot programmabile basato sul microcontrollore Arduino Uno, in grado di muoversi agilmente su una superficie piana grazie ai due motori montati rispettivamente sui due lati opposti del corpo.

Osservazioni Fra i pregi dell'impiego dei motori a vibrazione vi sono le dimensioni e le forme del sistema locomotivo poco invasive, i bassi valori di tensione e corrente necessari al funzionamento e i prezzi non troppo elevati. Fra i difetti, quello più visibile e difficile da risolvibile è la lentezza del movimento, gli altri riguardano la

comprensione e la modellazione matematica del sistema. Il costo dei motori vibranti nel dettaglio potrebbe superare quello dei motori convenzionali, anche del 50%, ma è possibile acquistare all'ingrosso decine di pezzi a prezzi veramente simili. L'eventuale rimozione della massa degli ERM Motors è possibile manualmente in pochi semplici passi.

1.1.3 Gambe

Il controllo del movimento dei robot dotati di un sistema locomotivo basato su gambe richiede di considerare due problemi molto complessi: il mantenimento di una posizione verticale, e il controllo del movimento, per garantire stabilità e regolabilità [6]. I vantaggi di un bipede sono enormi in termini di mobilità e agilità, ma l'implementazione di un sistema così complesso vanno fuori dall'obiettivo di questo studio. Esistono soluzioni abbastanza semplici per realizzare bipedi, simili a quelli che si potrebbero trovare in certi giocattoli, che sfruttano aree di appoggio molto grandi ma sono in grado di esercitare unicamente un movimento limitato e monodirezionale. Molti mammiferi, che presentano una corporatura grande e pesante, si poggiano su quattro gambe e presentano una maggiore stabilità, ma dal momento in cui ne sollevano anche solo una per muoversi, devono attivamente cambiare la posizione del corpo per fare in modo che la proiezione perpendicolare dal centro di gravità ricada nel triangolo disegnato dalle altre su cui si appoggiano. Per questo motivo animali con sei o più gambe riescono a mantenere molto più facilmente l'equilibrio, ad esempio muovendosi lasciando a contatto con il terreno almeno quattro gambe [7]. Risalgono agli anni settanta, se non prima, i primi studi e i prototipi di robot su sei gambe, notevole ispirazione per la realizzazione di questi modelli locomotivi sono stati gli insetti [8]. Molti insetti hanno sei gambe, una coppia per segmento toracico, che a seconda della posizione prendono il nome di anteriori, mesatoraciche e metatoraciche. Il modo in cui gli insetti sfruttano le gambe per muoversi può variare notevolmente, a seconda della velocità o stabilità che vogliono assumere o dalla morfologia delle zampe, fra i movimenti più famosi troviamo il *metachronal gait* e il *tripod gait*. Il primo, tipico degli insetti più pesanti e lenti, muove individualmente una gamba per volta, provocando un effetto a onda, il secondo, più comune alle specie leggere e veloci, alterna il movimento delle zampe anteriori e metatoraciche di un lato assieme a quella mesotoracica dell'altro con le rimanenti che poggiano a terra.

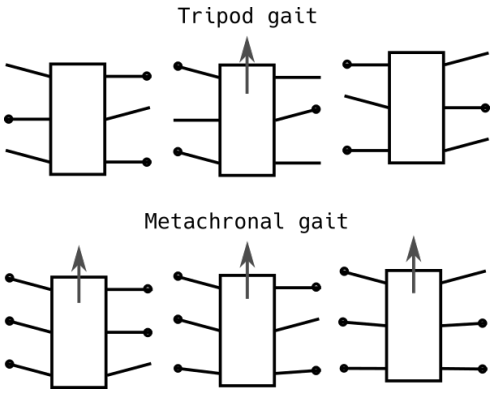


Figura 1: I due movimenti messi a confronto, le zampe che terminano con un pallino stanno poggiando a terra. Il metachronal ha già mosso, una per una, tutte le zampe di un lato. Tendenzialmente il tripod gait è più veloce mentre il metachronal gait è più stabile.

Esempi Un insetto protagonista di molte ricerche, è il ”cockroach”, un Blattoideo che in italiano è comunemente denominato scarafaggio. Da questo insetto nascono diverse imitazioni robotiche, una molto interessante è RHex [37]. RHex si scosta dall'imitazione fedele del mondo biologico, in quanto a differenza di altre gambe robotiche, articolate e complesse, quella di RHex è costituita da un pezzo unico di forma semicircolare capace unicamente di ruotare attorno al punto in cui è fissato al corpo. Ciascuno dei due lati del corpo dispone di tre gambe che grazie alla forma e al movimento rotatorio, permettono al robot di muoversi anche in terreni particolarmente accidentati. Un altro modello degno di nota è Sprawlita, ispirato dalla caratteristica tipica di molti insetti, di mantenere una posizione distesa durante il movimento. Per questo motivo le gambe di Sprawlita, che sono dei piccoli pistoni, sono inclinate rispetto al corpo ad assumere la tipica posizione osservata nel mondo animale. Numerosi sono gli esemplari provenienti anche dalla filosofia BEAM, i Walker, ispirati dallo storico modello costruito da Mark Tilden, sono considerati una vera e propria famiglia, in quanto sono implementabili in vari modi, in base al numero di motori e gambe o alla struttura della rete neuronale che attiva il processo locomotivo. Appartengono sempre alla famiglia BEAM i Turbot, che si muovono ruotando uno o più flagelli, i modelli più complessi possono muoversi in più direzioni oppure invertire la marcia dei motori per uscire da spazi angusti.

Osservazioni Negli esempi precedenti sono state ignorate volutamente le soluzioni complesse, come quelle implementate nei robot umanoidi sviluppati negli ultimi decenni e dotati di capacità locomotive talvolta straordinarie, in quanto andrebbero fuori dallo scopo di questo studio e molto probabilmente non rispetterebbero i limiti del budget. Ignorando quindi la possibilità di costruire un arto complesso in grado di imitare le articolazioni e i muscoli di quelle del mondo biologico, rimane aperta la possibilità di un'implementazione più elementare, che prende comunque ispirazione dal movimento degli insetti. Ad esempio, la società multinazionale Tamiya, che opera nel campo del modellismo statico e dinamico, mette in vendita un kit di costruzione di un six-legged "mechanical insect" telecomandabile e capace di spostarsi e ruotare utilizzando un'ingegnosa combinazione che, sfruttando solo un motore servo per lato, permette il movimento in avanti e la rotazione. Alternativamente, impiegando tre servo motori è possibile alzare e ruotare tre coppie di gambe per sincronizzarle a eseguire il movimento a tripod. L'utilizzo dei motori servo in questo scenario, ma anche per muovere sensori o pinze, è considerato comune, in quanto suddetti motori sono in grado di regolare precisamente l'angolo di rotazione e di mantenerlo in modo stabile. Se messo a confronto con il prezzo dei semplici motori in corrente continua, quello dei motori servo è nettamente maggiore, ma per la facilità di utilizzo e le potenzialità applicative che dispone, il costo può essere considerato comunque accessibile. In conclusione, la costruzione di arti anche con una struttura piuttosto elementare, potrebbe essere realizzabile ed efficace ma non permetterebbe l'ottenimento di enormi vantaggi ai fini di questo studio.

2 Costruzione

Le ricerche iniziali hanno confermato le ruote come principale mezzo locomotivo nell'ambito della Swarm Robotics, e non solo. Inoltre risulta evidente che, la pretesa di sviluppare un robot che sia in grado di soddisfare anche solo parzialmente una richiesta di modularità e adattabilità per l'ambito di applicazione, necessiti di un sistema più robusto, capace di supportare leggere modifiche del peso e della forma. Tali modifiche, su un robot di piccole dimensioni, che magari sfrutta la vibrazione per muoversi, potrebbero gravemente inficiare il funzionamento complessivo. L'uso delle ruote, tende a smorzare la rigidità dei vincoli meccanici e fisici degli altri mezzi locomotivi, offrendo una soluzione valida sia per ambiti applicativi che didattici, versatile e concettualmente più semplice e conosciuta. In questo capitolo viene descritta la costruzione di AlpaBot, a partire dai prototipi, descrivendo nel dettaglio le componenti relative al movimento e l'alimentazione.

2.1 Prototipi

2.1.1 Le componenti di un robot

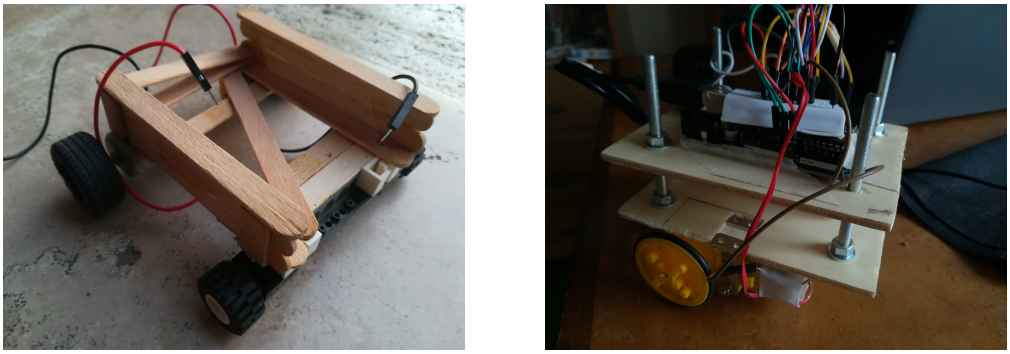
Un robot è un sistema autonomo che esiste nel mondo fisico, può percepire l'ambiente attorno a sé e può influenzarlo per perseguire le proprie intenzioni. La costruzione di un robot presuppone la conoscenza degli aspetti fondamentali che riguardano la sua natura, precedentemente sono stati presentati numerosi esempi di mobile robot, in questa parte si vuole entrare nei dettagli utili alla progettazione. Dalla definizione di robot, è possibile dedurre che è composto da un corpo fisico che esiste e occupa spazio nel mondo reale, da sensori per percepire e attuatori per interagire con l'esterno e opzionalmente un controller per supportare la propria autonomia. Con *embodiment* viene descritta la caratteristica peculiare di un robot: avere un corpo che obbedisce alle medesime regole fisiche alle quali sottostanno anche gli esseri viventi. Affrontare il mondo reale è notoriamente considerata una scelta che comporta molti più ostacoli di quelli che si incontrano in un ambiente simulato, in quanto tute le interazioni fra il sistema autonomo e l'esterno sono inevitabilmente da gestire. Da queste considerazioni si può intuire che la progettazione di un robot è influenzata pesantemente dall'ambiente in cui il robot opererà e dai compiti che questo dovrà svolgere una volta in funzione. Il termine usato per riferirsi all'ambiente di un robot è preso in prestito

dalla biologia ed è *ecological niche*. Per questo motivo, è stato spontaneo realizzare il mobile robot nella forma più modulare possibile, al fine di creare un sistema facilmente modificabile e aperto. E' inevitabile che questa scelta comporti anche un impatto negativo sull'efficienza finale del robot, in particolar modo se comparata con quella di altri costruiti appositamente per lo svolgimento di certi compiti. In conclusione le componenti di un mobile robot sono il sistema locomotivo, in questo caso composto da motori e ruote, una fonte di energia, uno o più microcontrollori ed eventulmente i vari sensori e attuatori installabili per scopi più specifici.

2.1.2 Descrizione dei prototipi

Inizialmente sono stati sviluppati alcuni prototipi con l'obiettivo principale di prendere maggiore confidenza con gli elementi primari dell'apparato locomotivo e conoscere gli aspetti dipendenti da questi. I materiali impiegati per la costruzione di questi primi prototipi sono stati reperiti un po' ovunque e sono principalmente legno e plastica, tenuti assieme senza troppa cura da colle di vario genere e nastro adesivo. Fin dall'origine, la parte elettronica comprendeva un microcontrollore Arduino Uno, due motori in corrente continua, un circuito integrato L293D e quattro batterie stilo ricaricabili AA da 1.2V ciascuna, in grado di fornire complessivamente poco più di 4.8V di tensione e una capacità di 1300mAh. Il microcontrollore ha il compito di comandare i motori e testare il funzionamento del chip L293D, un motor driver che permette di controllare simultaneamente la direzione e la velocità di due motori distinti. Nonostante alcuni successi, rappresentati dalla possibilità di muovere correttamente la maggior parte dei prototipi, emergevano grandi problemi legati in particolar modo alla gestione dell'alimentazione e alla scelta dei motori. L'alimentazione ha destato fin dall'inizio grosse preoccupazioni a causa della vasta scelta di prodotti, differenti fra loro per caratteristiche fisiche, chimiche ed energetiche. Si tratta di una scelta cruciale per il funzionamento complessivo del robot, che deve considerare aspetti che in quel momento erano stati posticipati, evidenziando la necessità di una progettazione molto più consapevole dei limiti e delle capacità che si supponga il robot debba avere. Nel caso specifico, si è sperimentato immediatamente il problema della durata delle batterie, che tendevano a scaricarsi molto velocemente, e della corrente di uscita che non garantiva quel livello di sicurezza richiesto per il corretto funzionamento del microcontrollore e dei due motori. In certe momenti, ad esempio in accensione, la richiesta

di corrente molto elevata dei motori causava lo spegnimento del microcontrollore. Le batterie utilizzate in questo prototipo hanno inoltre indotto a considerare un altro aspetto molto importante, l'importanza del peso e la sua distribuzione nel corpo del robot. Successivamente, per alleviare alcuni di questi problemi, le batterie stilo sono state sostituite da una comune power bank per smartphone, collegata al microcontrollore tramite un cavo USB. I motori utilizzati sono dei motori in corrente continua, reperibili anche con il nome standard "130 size", capaci di operare in una fascia da 3 a 6 Volt di tensione, e solitamente venduti a prezzi molto economici. Oltre all'elevato consumo energetico, che può superare anche un Ampere, in particolare in accensione e in stallo, vi è il problema dell'alta velocità di rotazione. Benché possa sembrare un dato positivo, il fatto che possano arrivare a compiere fino a migliaia di rotazioni per minuto, potrebbe essere la giusta indicazione del fatto che probabilmente non riescano a fornire la "spinta" necessaria a muovere il robot. Solo verso la fine della fase di prototipazione i motori sono stati sostituiti con altri più efficienti, descritti nei capitoli sucessivi.



(a) (b)

Figura 2: Due prototipi del robot. La figura 2a mostra uno dei primi, a quattro ruote e con lo chassis in legno che è anche un alloggiamento per quattro batterie stilo. La figura 2b mostra il prototipo più vicino al risultato finale, i motori ad esempio sono gli stessi. L'alimentazione è fornita da una power bank riposta nel primo piano.

Osservazioni La realizzazione di questi prototipi è stata necessaria per conoscere in modo pragmatico alcuni aspetti davvero critici della progettazione meccanica ed elettronica di un robot e per dimostrare la necessità di adoperare un metodo più analitico per la risoluzione dei problemi elencati precedentemente. Si potrebbero spendere

tesi intere per approfondire molti di questi aspetti, quindi nelle prossime pagine si cercherà di non andare oltre gli obiettivi di questo studio e di considerare col fine ultimo di conoscere, e laddove possibile gestire nel modo più corretto, suddetti argomenti. Per questo motivo, successivamente saranno ripresi nel dettaglio tutti gli aspetti che in questa sezione sono stati volutamente accennati, dividendo lo studio in una parte concernente i motori e la parte meccanica del robot e un'altra l'alimentazione.

2.2 Meccanica e motori

In fisica il numero di *gradi di libertà* di un punto materiale è il numero di variabili indipendenti necessarie per determinare univocamente la sua posizione nello spazio. Un robot su ruote che si muove su una superficie piana possiede tre gradi di libertà che sono: la posizione, una coordinata bidimensionale, e l'orientazione, data dalla rotazione attorno a un asse perpendicolare al piano d'appoggio. Però non è scontato che un robot sia in grado di controllare ogni grado di libertà in modo indipendente, ad esempio una macchina che possiede i medesimi gradi di libertà descritti precedentemente può controllarne solo due, la rotazione e una direzione di movimento. Per questo motivo un automobile che deve parcheggiare in fila deve compiere diverse manovre, assumendo una velocità discontinua. Si dice che un sistema è *olonomico* quando tutti i gradi di libertà sono controllabili, mentre *non olonomico* quando solo alcuni. Le ruote comuni offrono grande stabilità ma non permettono la costruzione di un sistema olonomico, esistono però ruote omnidirezionali che se disposte in certa forma, ad esempio triangolare, permettono al robot di essere olonomico. Le ruote omnidirezionali contengono lungo la loro circonferenza dei dischi che possono ruotare nel piano perpendicolare a quello in cui gira la ruota stesssa, permettendo di scivolare facilmente sul terreno in due direzioni perpendicolari. La possibilità di poter pilotare in modo indipendente le singole ruote prende il nome di *Differential Drive*. Il numero, la dispozione, il controllo delle ruote da origine a differenti sistemi locomotivi, fra i principali si trovano la *Differential Wheel* e la *Ackerman steering*. Quest'ultimo fa uso di quattro ruote, due frontali che hanno lo scopo si direzionare il movimento e due anteriori che sono motrici. I vantaggi di questo metodo sono una maggiore stabilità, una maggiore aderenza al terreno e consumi minori, ma la richiesta di una costruzione meccanica notevole e molto precisa scoraggia la costruzione, soprattutto nelle piccole dimensioni.

2.2.1 Differential Wheel

In robotica, con Differential Wheel si intende un sistema locomotivo basato su due ruote motrici indipendenti, posizionate solitamente ai lati del corpo di un robot lungo il medesimo asse, in grado di ruotare e muovere il robot in base alla velocità e direzione di rotazione che queste assumono. Quando queste coincidono il robot si muove lungo una linea retta, se la direzione è opposta invece, il robot gira su se stesso, in senso orario o antiorario, quando le velocità sono diverse il robot tende a disegnare degli archi più o meno aperti a seconda della velocità angolare. La posizione delle ruote motrici ha degli effetti sulla circonferenza di rotazione del robot, è consigliato cercare di posizionarle all'interno della base del robot, per diminuire il diametro di questo e permettere la rotazione in uno spazio minore. Per migliorare la stabilità i robot si avvalgono di altri punti di appoggio al terreno, come ruote non motrici o pattini. Le ruote, solitamente indicate come caster wheels, sono fra le preferite perché essendo dotate di due gradi di libertà, riescono a evitare di perturbare la direzione di movimento del robot e provocare poco attrito con la superficie di appoggio. Possono essere composte da più ruote tradizionali, oppure essere ruote omnidirezionali o a sfera. Per robot leggeri e di piccole dimensioni non è escluso l'uso di pattini, detti skid, dalla testa arrotondata e di materiale consistente, come metallo o plastica dura. E' preferibile utilizzare pattini piuttosto che caster wheel scadenti o poco efficienti, in quanto difficoltà di rotazione o scivolamento vanificherebbero ogni pregio delle seconde sulle prime. La dimensione dei sostegni deve considerare quella delle ruote motrici e l'altezza del rotori dei motori, che possono trovarsi sopra o sotto la base, in modo tale da far assumere al robot la posizione più bilanciata possibile. La disposizione e il numero dei sostegni può seguire due schemi principali, quello a triciclo che utilizza un unico sostegno o quello a croce che ne utilizza due. Quest'ultimo, nonostante il sostegno maggiore che riesce a fornire, è poco consigliato in ambienti pratici per il fatto che può mettere a rischio il robot di trovarsi nella spiacevole situazione di poggiare solo con i sostegni ma non con le ruote motrici, lasciando il robot incapace di muoversi. Come ultima osservazione è importante tenere in considerazione le relazioni fra lo schema delle ruote scelto, la forma del robot e quindi la distribuzione del peso che deve essere la più uniforme possibile, concentrandola maggiormente sul centro della figura geometrica disegnata dai punti di appoggio. In conclusione a causa del rischio di utilizzare prodotti economici ma scadenti, che potrebbero influire

negativamente sul corretto funzionamento, e per via del peso ridotto del robot si opta per l'uso di appoggi statici a forma di croce. A differenza della Ackerman steering, questo metodo è molto più semplice da implementare ma non fornisce prestazioni eccellenti, soprattutto per il fatto che il numero di rotazioni di una ruota nella stessa unità di tempo può differire dall'altra causando imprecisioni.

Cinematica della Differential Wheel Come accennato precedentemente, posizione e orientamento di un robot su un piano sono rappresentabili da tre numeri, le due coordinate della posizione relativa a un certo sistema di riferimento e l'angolo rispetto all'asse x, o y, di tale sistema. E' possibile prevedere la posizione del robot conoscendo quella attuale, il tempo trascorso, il valore della velocità lineare sui due assi e il valore della velocità angolare, utilizzando la legge oraria del moto e considerando le velocità costanti nel tempo [22].

$$x = x_0 + v_x * \Delta t \tag{1}$$

$$y = y_0 + v_y * \Delta t \tag{2}$$

$$\theta = \theta_0 + \omega * \Delta t \tag{3}$$

Dove Δt è la variazione temporale, x_0 e y_0 è la posizione iniziale, θ_0 è l'angolo inziale, ω la velocità angolare e infine v_x e v_y sono i valori delle componenti della velocità lineare. Sebbene traslazione e rotazione di un sistema basato su Differential Wheel siano fra loro correlati è possibile dividerli, esprimendo ogni combinazione come una rotazione seguita da un movimento rettilineo. Ogni ruota ha un raggio r e una velocità angolare propria che può essere controllata indipendentemente. Il legame fra la velocità angolare (ω_r) e lineare (v) di una ruota è basato sulle seguenti equazioni:

$$v = \omega_r * r \tag{4}$$

$$\omega_r = \frac{v}{r} \tag{5}$$

Se non si considerano scivolamenti o movimenti laterali la velocità lineare del robot, semplificato come punto materiale situato a metà della distanza fra le due ruote (l), può essere facilmente calcolabile sommando le singole velocità assunte dalla ruota sinistra (v_1) e destra (v_2).

$$V = \frac{v_1 + v_2}{2} \tag{6}$$

In particolare se una ruota gira mentre l'altra resta ferma il robot si muove con una velocità pari alla metà di quella assunta dalla ruota in movimento, quindi se si muovono entrambe con la medesima velocità lineare il robot si muove con tale velocità. Se la velocità lineare delle due ruote ha pari intensità ma direzione opposta il robot non si muove, ma gira su se stesso. In generale le due ruote condividono un punto di rotazione detto Instantenuos Center of Curvature (ICC) che varia la propria distanza dal robot a seconda delle velocità assunte dalle singole ruote. Anche in questo caso, per calcolare la velocità di rotazione del robot, si considera una ruota per volta, se solo una gira allora il robot ruoterà attorno al punto di contatto della ruota ferma, quindi in una circonferenza di raggio l. Di seguito sono espresse le formule delle singole ruote in movimento e poi la loro somma.

$$\omega_1 = -\frac{\omega_l * r}{l} = -\frac{v_1}{l} \tag{7}$$

$$\omega_2 = \frac{\omega_r * r}{l} = \frac{v_2}{l} \tag{8}$$

$$\omega = \frac{v_2 - v_1}{l} \tag{9}$$

Quando le velocità delle ruote sono costanti il sistema descrive un movimento circolare di raggio R, che parte dalla posizione del robot che si trova fra a metà della distanza delle due ruote.

$$R = \frac{l}{2} * \frac{v_1 + v_2}{v_2 - v_1} = \frac{V}{\omega} \tag{10}$$

R tende a infinito quando le ruote si muovono alla medesima velocità e verso di rotazione.

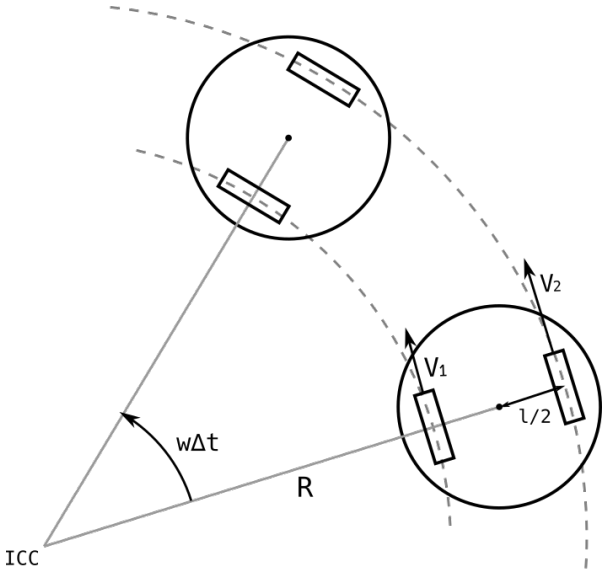


Figura 3: Lo schema mostra un sistema autonomo che sfrutta la Differential Wheel per muoversi.

2.2.2 Caratteristiche tecniche dei motori

La lista seguente permette di comprendere le proprietà elettriche e meccaniche più importanti dei motori in corrente continua.

Velocità angolare in rad/s o rpm, indica il numero di rivoluzioni nell'unità di tempo. Conoscendo il raggio della ruota è possibile ottenere la velocità tangenziale (m/s) con l'equazione 4.

Momento meccanico in N·m, è una forza che produce o tende a produrre una rotazione o un torsione a un corpo rigido attorno a un punto o un asse. Può essere indicata anche coppia motrice, ovvero come l'insieme delle forze esercitate da un motore su una trasmissione.

Potenza in W, indica la quantità di lavoro svolto nell'unità di tempo. Nel caso del movimento rotazionale può essere ottenuta dal prodotto fra la velocità angolare e il momento meccanico. Può essere intesa anche come la potenza data dalla

tensione e dalla corrente applicata al motore, utile per comprendere i limiti del motore, onde evitare che si danneggi.

Tensione in V, solitamente rappresentata da un intervallo entro il quale il motore funziona correttamente.

Corrente in A o mA, richiesta dal motore a seconda dello sforzo compiuto e dal valore di tensione totale.

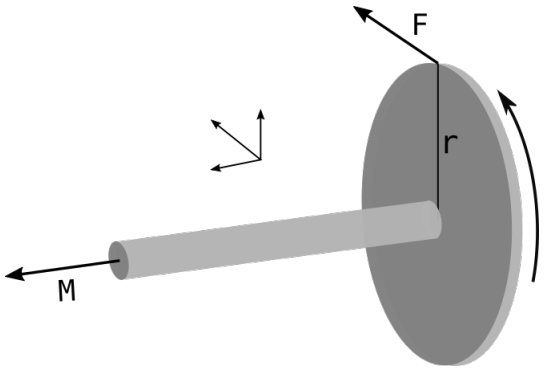


Figura 4: Il momento è uguale al prodotto fra la componente perpendicolare al braccio della forza F e la distanza dall'asse di rotazione.

Parte non integrante dei motori ma che viene utilizzata per modificare alcuni dei valori precedenti è rappresentata dai meccanismi di riduzione a ruote dentate, in inglese dette gears. L'aggiunta di tali componenti permette di sfruttare il principio di conservazione del momento angolare, per trasmettere il moto tramite un numero di ruote di dimensioni differenti, ognuna delle quali inverte la direzione di rotazione. Il rapporto che sussiste fra le dimensioni delle ruote, o meglio fra il numero dei denti delle ruote, permette di calcolare come cambia la velocità e il momento meccanico. Ad esempio in un rapporto 1:2 la prima si dimezza, mentre la seconda si duplica, perché la ruota di dimensioni doppie compie un giro più lentamente dell'altra, ma acquisce maggiore forza. L'utilizzo dei riduttori comporta spesso anche gravi perdite di efficienza a causa dei molteplici problemi che tali meccanismi potrebbero porre, come surriscaldamenti, difficoltà di lubrificazione o presenze di disallineamenti fra le ruote, in particolare quando si fa uso di prodotti economici composti da materiali poco durevoli come la plastica.

2.2.3 Relazioni fra forze e proprietà dei motori

Quando un motore in corrente continua viene alimentato esso può essere visto come un generatore, in quanto produce una forza elettromotrice che si oppone alla corrente che l'ha generata. All'inizio, il passaggio della corrente è limitato solamente dalla resistenza interna degli avvolgimenti all'interno del motore, e quindi assume un valore elevato che viene denominato corrente di spunto. Al crescere della velocità di rotazione, e quindi della tensione generata, il valore della corrente tende a essere limitato sempre di più, fino a quando raggiunge un valore di equilibrio, che in assenza di attriti interni sarebbe nullo, e prende il nome di corrente a vuoto. In generale la corrente può essere calcolata sfruttando la Legge di Ohm:

$$I = \frac{V_a - V_g}{R} \tag{11}$$

I è la corrente, V_a è la tensione applicata, V_g è la tensione generata dal motore e R è la resistenza. Il momento meccanico è proporzionale al valore della corrente e dal flusso totale che circola nel motore secondo la formula:

$$M = k_t * I * \Phi \tag{12}$$

M è il momento meccanico, k_t è una costante che dipende dal motore, I è la corrente applicata e Φ è il flusso creato dal passaggio della corrente nel motore.

Si definisce quindi velocità senza carico, quella che il motore ottiene girando liberamente, e con coppia di stallo, il valore massimo di momento meccanico esercitato tale per cui la velocita di rotazione è nulla. Quando il motore raggiunge il valore di stallo, la corrente che fluisce aumenta e raggiunge il valore massimo aumentando la caduta di potenziale. Se il motore raggiunge la velocità senza carico, la corrente che fluisce nel motore diminuisce, raggiungendo il valore minimo e quindi diminuisce la caduta di potenziale.

La figura 5 mostra il grafico che rappresenta la relazione fra velocità angolare, momento meccanico e potenza di un ideale motore in corrente continua. E' la conferma che all'aumentare della velocità diminuisce la forza per muovere il corpo e viceversa. L'area sotto la curva rappresenta la potenza, che è massima quando il momento

meccanico raggiunge un valore pari alla metà di quello che raggiunge in stallo e la velocità un valore pari alla metà di quello che raggiunge senza carico.

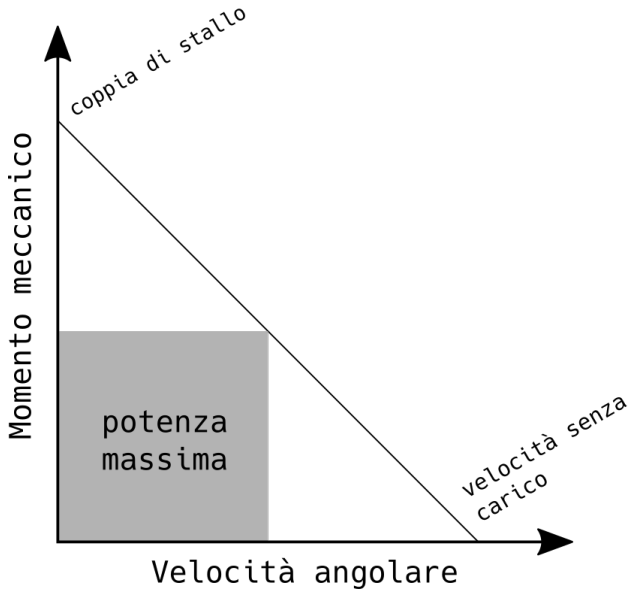


Figura 5: La relazione che sussiste fra il momento meccanico e la velocità in un tipico motore a corrente continua.

La potenza elettrica in ingresso è quindi esprimibile nei seguenti modi:

$$P = V_g * I \tag{13}$$

$$P = M * \frac{2 * \pi * RPM}{60} \tag{14}$$

Raramente tale potenza viene convertita completamente in potenza meccanica, in quanto l'efficienza non è mai 100%, rasentando anche valori molto bassi quando si utilizzano prodotti economici poco professionali.

2.2.4 Scelta del motore

Introdotti i motori in corrente continua e definite le caratteristiche principali, è opportuno scegliere il motore adeguato. La tabella seguente mostra le tre principali famiglie di motori elettrici presenti in commercio, presentando un confronto anche sul piano economico.

Confronto fra motori		
Nome	Breve descrizione	Costi
Brushed DC Motors	Sono la classica implementazione del motore a corrente continua, utilizzano due spazzole per condurre la corrente all'armatura e solitamente dispongono all'interno di un magnete permanente. Sono poco costosi e leggeri, inoltre non richiedono necessariamente la presenza di un controller per funzionare.	I costi possono variare leggermente, dovrebbe essere possibile reperirli a prezzi molto bassi, da uno a due euro.
Geared DC Motors	Sono motori in corrente continua che presentano già all'acquisto un meccanismo a ruota dentata, spesso motore e riduttori sono inseriti all'interno di un box.	Il costo non si allontana molto dai modelli senza gear, sono spesso reperibili in coppia in kit già completi di ruote.
Brusheless DC Motors	Sono un'ottimizzazione dei motori in corrente continua, sono privi di spazzole e per questo possono risultare più duraturi, in compenso il controllo risulta più complesso.	Decisamente più costosi, possono arrivare a costare fino a decine di euro al pezzo.

Con motor sizing si intende lo studio atto a comprendere le proprietà del motore in relazione alle richieste che deve soddisfare. Si vuole un robot che sia in grado di operare nelle seguenti condizioni:

- Sopportare una massa massima di 2 Kg
- Mantenere una velocità costante di almeno 0.1 m/s
- Raggiungere la velocità masssima in 2 secondi
- Operare con una ruota di raggio pari a 23 cm
- Muoversi in terreni inclinati al massimo di 5°
- Escludere l'attrito perché non sono noti i coefficienti di attrito dei materiali coinvolti e altre informazioni che dipendono dall'ambiente in cui si opera.

- Supporre l'efficienza del motore pari al 50% perché i motori e le ruote non sono professionali, anche se restano sconosciute le reali informazioni

Considerando che il momento meccanico è esprimibile come:

$$M = F_m * r \tag{15}$$

F_m rappresenta la forza applicata per girare la ruota attorno al rotore e r la distanza dall'asse di rotazione, quindi il raggio della ruota. Il legame fra la velocità v del robot e l'accelerazione a impiegata per raggiungere tale velocità si esprime con la seguente equazione:

$$v = \frac{a * t}{2} + v_0 \tag{16}$$

t rappresenta il tempo in cui raggiunge la velocità v e v_0 la velocità iniziale. Dai dati iniziali è possibile ricavare l'accelerazione necessaria a portare il robot da fermo ($v_0 = 0$) alla velocità minima costante richiesta:

$$a = \frac{v * 2}{t} \tag{17}$$

L'attrito statico massimo è esprimibile come il prodotto fra la forza normale N, perpendicolare al piano di appoggio, e pari al peso del corpo e il coefficiente di attrito μ, che dipende dai materiali delle superfici a contatto.

$$F_s = N * \mu = m * g * \mu \tag{18}$$

m è la massa del robot e g l'accelerazione gravitazionale.

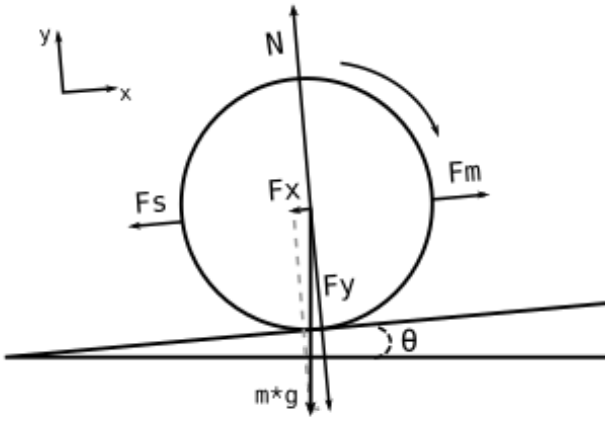


Figura 6: La rappresentazione grafica della forze coinvolte (non in scala).

Dalla seconda legge della dinamica è possibile esprimere la somma delle forze uguale al prodotto della massa con l'accelerazione del corpo. Se la velocità è costante e quindi l'accelerazione è nulla allora anche la somma delle forze deve essere nulla

$$\Sigma F = F_m - F_s - F_x = \frac{M}{r} - m * g * \mu - m * g * sin(\theta) = m * a = 0 \qquad (19)$$

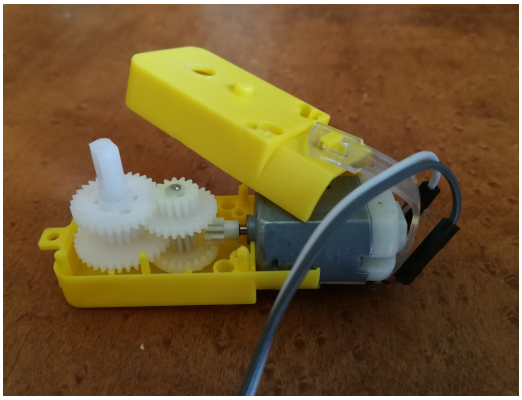
Da tale formula è possibile ricavare il momento meccanico, includendo anche l'efficienza e del motore ma escludendo l'attrito ($F_s = 0$)

$$M = (100/e) * (a + g * sin(\theta)) * m * r \qquad (20)$$

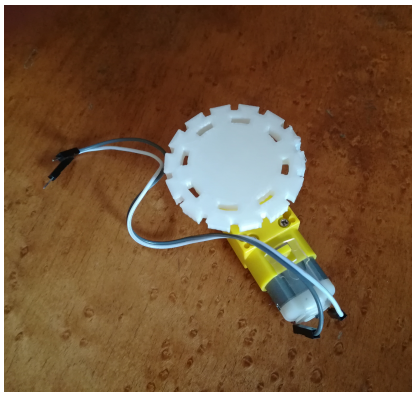
Essendo due i motori utilizzati, il momento meccanico di ciascun motore è pari alla metà del momento meccanico espresso con l'equazione 20. La velocità lineare cresce all'aumentare della dimensione della ruota e della velocità angolare. E' possibile ottenere la velocità angolare da quella lineare sfruttando l'equazione 5 e da questa poi ottenere il numero di giri al minuto e la potenza richiesta da entrambi i motori. Il numero di giri al minuto si ottiene moltiplicando la velocità angolare per $\frac{60}{2\pi}$ mentre la potenza utilizzando l'equazione 14. In conclusione si ottengono:

- il momento meccanico esercitato da ciascun motore pari a 0.044 N·m
- il numero di rotazioni per minuto di ciascuna ruota pari a 41.519
- la potenza richiesta da ciascun motore pari a 0.191 W

Il geared motor acquistato, si trova facilmente in commercio, talvolta viene venduto in kit per la costruzione di veicoli autonomi con Arduino, ma è economicamente reperibile anche al dettaglio. Viene prodotto in una forma standard talvolta denominata TT. Dalle descrizioni tecniche reperite da più venditori, si può apprendere che a 5V possono esercitare fino a 0.8 Kgf·cm ovvero 0.07845 N·m di momento meccanico e compiere 130 rotazioni al minuto, quindi sono in grado di rispettare pienamente i vincoli sopra imposti.



(a)



(b)

Figura 7: La figura 7a mostra l'interno del gearbox, mentre la 7b il motore e la ruota. Da notare la presenza dei fori nella ruota per permettere l'uso futuro di un encoder a infrarossi per misurare i giri del motore.

2.2.5 Motor driver

In commercio esistono numerosi circuiti stampati per il controllo dei motori comunemente denominati motor drivers. I compiti di un motor driver sono di seguito elencati.

- Mantenere in un cicuito compatto tutte le componenti elettroniche necessarie per comandare uno o più motori elettrici. In questo contesto si tratta di regolare la velocità di rotazione e il verso di rotazione.

- Alimentare correttamente i motori e i microcontrollori. I microcontrollori operano solitamente a livelli di tensione bassi, mentre è frequente l'utilizzo di motori che operano a tensioni maggiori. Certi motor driver permettono di separare semplicemente l'alimentazione dei motori da quella del microcontrollore, altri permettono di regolare la tensione in ingresso e alimentare correttamente il microcontrollore.

- Evitare che il rumore provocato dai motori elettrici interferisca direttamente con il microcontrollore.

- Fornire la tensione adeguata ai motori, in particolare durante l'accensione, impiegando resistenze di pull-down.

Per controllare il verso di rotazione e la velocità dei motori è utilizzato il circuito integrato L293D, un dual H-bridge motor driver in grado di comandare velocità e direzione di due motori in corrente continua.

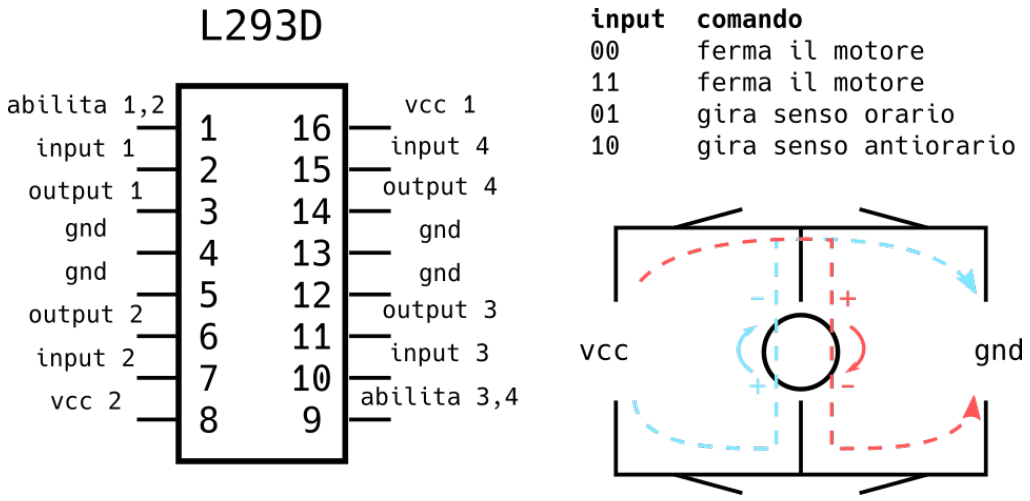


Figura 8: Il pinout del circuito integrato L293D, i comandi da inviare ai pin di input e il funzionamento generale del ponte H. I due lati del circuito integrato sono speculari e ciascuno serve un motore. Il pin enable permette di regolare la velocità e viene collegato con un pin di Arduino che permette di eseguire PWM, i pin di input permettono di regolare la direzione del motore. I pin di output sono collegati ai poli del motore mentre vcc e gnd all'alimentazione.

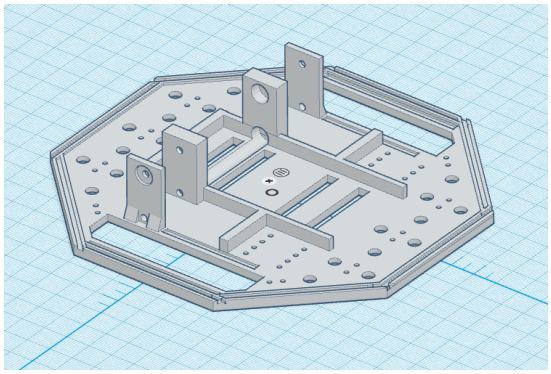
La velocità viene controllata influenzando il modo in cui il campo magnetico dello statore, che è fisso, interagisce con quello prodotto dal rotore che invece è generato dalla corrente fornita in ingresso. Utilizzando Pulse Width Modulation (PWM) è possibile modificare la corrente in ingresso e quindi la tensione generata dal motore determina la velocità di rotazione. Con Pulse Width Modulation si intende l'invio di segnali alti e bassi che determinano il periodo medio in cui il segnale è alto piuttosto che basso, maggiore è il tempo in cui è alto, più velocemente gira il rotore. La direzione di rotazione avviene impiegnado un H-brige (ponte H), un circuito composto da quattro interruttori, diodi e switch, che permettono alla corrente di fluire nel motore nei due modi opposti, permettendo al rotore di ruotare in entrambi i sensi.

2.2.6 Forma e struttura

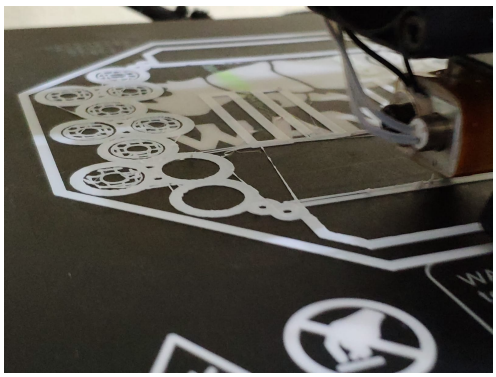
Forma La forma del robot contribuisce a definire le capacità motorie di questo e influenza la scelta, la disposizione, il numero delle parti elettroniche. La forma del robot dovrebbe evitare di limitare i movimenti, ad esempio per i robot mobili di piccole dimensioni è generalmente preferita una forma più circolare che quadrata. La forma è inevitabilmente coinvolta nello sviluppo anche software del robot, ad esempio la presenza di una sporgenza, di una asimmetria o di uno spazio fra i sensori e i bordi del robot devono essere tutti considerati e gestiti. Ad esempio due robot di simili dimensioni, uno di forma circolare e l'altro rettangolare, che devono entrare all'interno di una strettoia devono comportarsi in modo differente perché il secondo deve necessariamente ruotare in modo tale da non toccare le pareti, mentre il primo no. Al fine di rendere più semplice l'implementazione, è stata scelta una forma che obbligasse una certa simmetria nella disposizione dei sensori e degli attuatori, permettendo anche un'equa distribuzione del peso. La forma più vicina a quella circolare ma in grado di mantenere anche un concetto di lato è quella ottagonale. L'ottagono ha un'ampiezza di 15 cm con i lati di lunghezza pari a 6.2 cm circa e il raggio della circonferenza circoscritta pari a 8.1 cm circa, mentre la distanza delle ruote è di circa 11.6 cm.

Struttura La struttura del robot è quella a piani, una tecnica ampiamente utilizzata in altri modelli per via della semplicità con la quale è possibile espandere il robot mantenendo invariate le parti già costruite. L'aggiunta di nuovi componenti

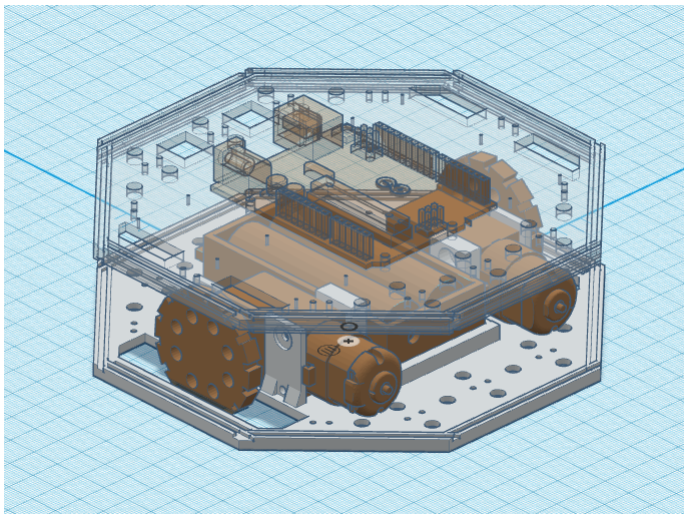
che richiedono più spazio di quello disponibile, avviene aggiungendo degli strati che solitamente ricalcano la forma dgli strati precedenti, permettendo di mantenere la larghezza del robot abbastanza invariata. L'altezza può invece cambiare notevolmente senza compromettere le funzionalità del robot, che è costruito per operare su superfici piane.



(a)



(b)



(c)

Figura 9: La figura 9a mostra il modello tridimensionale del primo piano, quello contenente il vano batterie e gli alloggiamenti per i motori. La figura 9b mostra la stampa del primo piano del robot. Infine la figura 9c mostra il modello 3D dei due piani e delle componenti principali

I piani e alcune componenti strutturali, come ruote e sostegni, sono realizzati dap-

prima come modelli tridimensionali con Tinkercad, un tool di design e prototipazione disponibile online e offerto gratuitamente da Autodesk, e sono successivamente stampati con una stampante 3D in PLA. Il PLA, o l'acido polilattato, è un materiale solitamente derivato dall'amido di mais, ma anche da altre piante, e presenta caratteristiche meccaniche intermedie a quelle del polietilene tereftalato e del polistirene. La configurazione minimale del robot è costituita da due livelli:

Livello 1 Il livello 1 è basilare, contiene lo spazio per l'inserimento delle batterie, centrato per evitare che il peso sbilanci da una parte il robot, e gli slot necessari a inserire i motori e le ruote, che rimangono all'interno della forma del robot.

Livello 2 Il livello 2 è molto generico, contiene lo spazio per il microcontrollore e qualche componente elettronico.

2.3 Alimentazione

Con alimentazione si intendono gli aspetti concernenti la corretta fornitura di energia ai sistemi elettrici di un sistema autonomo. L'energia rappresenta di fatto la risorsa principale sulla quale anche gli esseri viventi basano le proprie attività. Sarebbe difficile conciliare la parola autonomo con un sistema che non lo è energeticamente. La coscienza di un robot sui propri parametri energetici è solitamente denominata *Self-maintenance*, ovvero l'abilità di un robot di prendere cura di se stesso. Tale tema è così importante da essere considerato uno dei temi più affrontati nella ricerca nel campo della *Artificial Life*, in particolare nello sviluppo di meccanismi di raccolta delle risorse in autonomia (*autonomous foraging*). In questa sezione sono analizzate le principali caratteristiche fisiche e chimiche delle batterie ed è descritto il sistema di alimentazione di AlpaBot.

2.3.1 Caratteristiche

Successivamente vengono descritte le principali caratteristiche delle batterie.

Parametri fisico-elettrici La tensione o differenza di potenziale di una batteria, misurata in volt (V), dipende principalmente dai materiali con cui è composta, ma anche dalla temperatura, dal tempo trascorso dall'ultima carica e per le batterie ricaricabili dal numero di ricariche. Con capacità si intende la quantità di carica

elettrica elettrica che può essere immagazzinata espressa in ampere-ora (Ah) o in wattora (Wh) se moltiplicata per la tensione nominale.

Dimensione e forma Le batterie composte da una o più celle vengono assemblate e impacchettate in varie dimensioni e forme standard. Le più comuni sono quelle di dimensione cilindrica, comunemente chiamate stilo, come le note AA e AAA, oppure a bottone o rettangolari. Spesso le batterie cilindriche vengono congiunte fra loro in vari modi a secondo delle necessità a formare pacchi, che possono essere anche piuttosto ingombranti.

Peso Il peso dipende dai materiali con cui sono composte le singole celle e dal numero di quest'ultime.

2.3.2 Tipologie

La lista seguente permette di fare una rapida panoramica delle tipologie più comuni di batterie ricaricabili, precisamente chiamate batterie secondarie o accumulatori di energia.

Nickel-Cadmium (Ni-Cd) Mantengono bene la carica e la tensione anche quando non in uso, ma subiscono l'effetto della memoria che degrada la capacità massima della cella dopo ogni ricarica. Solitamente non hanno molta capacità e tendono a scaricarsi molto velocemente, offrendo buone prestazioni laddove si necessiti una capacità costante per momenti brevi ma energeticamente intensi. Si ricaricano e usano a basse temperature ma richiedono manutenzione per evitare l'aggravarsi dell'effetto della memoria. Si tratta di una tecnologia molto economica ma obsoleta, che fa uso di metalli pesanti e tossici, per questo motivo, in molti ambiti rimpiazzata con quella delle Nickel-Metal Hydride.

Nickel-Metal Hydride (Ni-MH) Simili alle precedenti per gli ambiti di applicazione, con risultati spesso molto più performanti in capacità e densità di energia, che può approssimare quella delle Lithium-Ion. Tendono a deteriorarsi se frequentemente caricate e scaricate, inoltre durante la carica generano maggiore calore delle Nickel-Cadmio. Soffrono maggiormente di self-discharging rispetto a quelle al Cadmio e necessitano talvolta di essere completamente scaricate per questioni di manutenzione, ma in generale soffrono di meno dell'effetto

della memtoria. Un grande pregio sta nella composizione chimica, ottenuta da materiali non tossici. Rappresentano la via di mezzo fra le Nickel-Cadmio e quelle al Litio.

Piombo-Acido Concepite a metà del diciannovesimo secolo rappresentano una delle tecnologie più conosciute e utilizzate. Non sono soggette all'effetto della memoria, sono economiche e non richiedono manutenzione. Hanno un rapporto fra energia e peso molto basso e utilizzano materiali inquinanti.

Lithium-Ion (Li-Ion) A differenza dei precedenti le batterie al Litio offrono un rapporto fra densità energetica e peso buonissimo, rappresentando la scelta migliore per l'alimentazione di dispositivi di piccole dimensioni, come smartphone e laptop. La tecnologia è fragile, può richiedere un circuito di protezione per evitare picchi di tensione durante la ricarica e uno scaricamento non eccessivo. Se usate nel modo non corretto possono causare fiamme o esplosioni. Sono in genere più costose delle Nickel-Metal Hydride e possono soffrire di invecchiamento anche se non utilizzate, soprattutto se riposte in luoghi poco freschi. Le celle cilindriche 18650 composte da ioni di litio, utilizzate ad esempio nelle batterie dei computer portatili, sono fra le migliori nel rapporto fra energia e costo ma richiedono maggiore spazio, soluzioni sottili potrebbero arrivare a duplicare il costo mantenendo la medesima capacità.

Lithium-Ion Polymer (Li-Poly) Si tratta di una forma ibrida fra le batterie Lithium-Polymer e quelle Lithium-Ion, concepite a causa della poca conduttività delle prime. Le caratteristiche sono molto simili alle Lithium-Ion, ma non richiedono il medesimo processo di costruzione e impacchettamento, permettendo forme più semplici e sottili, a discapito di una maggiore sicurezza. Sono più leggere delle Lithium-Ion ma possono essere leggermente più carenti in termini di capacità e numero di cicli di ricarica, inoltre possono essere più costose e meno reperibili.

2.3.3 Scelta del sistema di alimentazione

L'obiettivo è quello di realizzare un sistema di alimentazione che sia in grado di soddisfare la richiesta energetica del circuito, mantenendo anche un leggero margine onde

evitare che picchi di corrente o il deterioramento progressivo delle celle non provochino immediati malfunzionamenti. Altre caratteristiche decisive nella scelta sono il costo, il peso complessivo, le dimensioni e la sicurezza. Infine vengono valutati anche la capacità della batteria, che determina il tempo di autonomia del robot, e il meccanismo di ricarica, anche se con una priorità inferiore rispetto agli aspetti precedenti. Dall'analisi precedente risulta evidente la superiorità delle batterie Li-Ion nella maggior parte delle caratteristiche messe a confronto con le altre tipologie. Nel dettaglio le Li-Ion nella forma cilindrica delle 18650 rappresentano un economico compromesso per quanto riguarda peso e forma, in quanto quelle piatte e rettangolari, utilizzate negli smartphone e nei veicoli radiocomandati, risultano essere più costose al dettaglio.

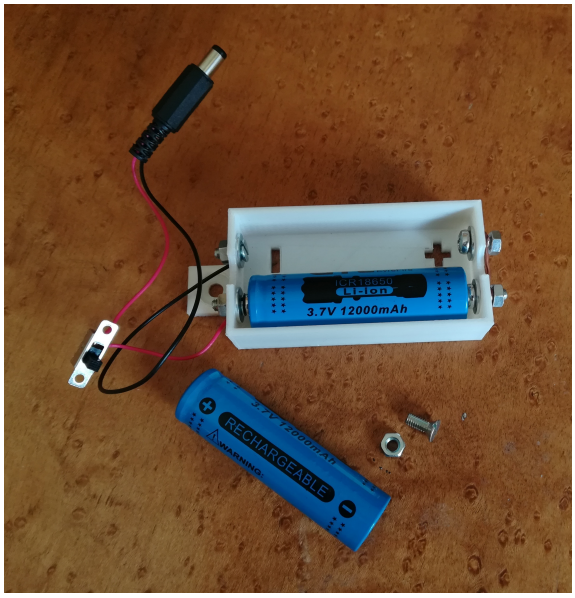


Figura 10: Le batterie e il relativo alloggiamento batterie.

Un aspetto critico delle batterie Li-Ion è la sicurezza, per questo motivo sono state scelte delle batterie che prevedessero un circuito di protezione integrato, onde evitare uno scaricamento eccessivo. Sempre per questioni di sicurezza, la ricarcia avviene in sede separata tramite l'utilizzo di strumenti appositi. Resta comunque aperta la possibilità di realizzare un circuito di protezione all'interno del robot per agevolare una ricarica diretta, anche se questa soluzione non presenta enormi vantaggi, ma apre sicuramente la strada a sistemi più autonomi anche dal punto di vista energetico. Nel

dettaglio le due batterie adoperate hanno una differenza di potenziale di 3,7 V e una capacità di 12000 mAh l'una. In serie arrivano a 7,4 V di tensione, un valore corretto per l'ingresso jack di Arduino, che necessita di operare con un valore di tensione fra 7 e 20 V per assicurare il corretto funzionamento del regolatore di tensione interno. Facendo uso di un multimetro digitale è possibile osservare sperimentalmente che nel circuito quando il motore è in:

In stallo la corrente richiesta dal singolo motore può arrivare a circa 380 mA, con una caduta di potenziale ai capi del motore di circa 3.42V, mentre Arduino complessivamente assorbe 780 mA.

Senza carico la corrente richiesta dal singolo motore non supera i 60 mA, con una caduta di potenziale ai capi del motore di circa 4.07V, mentre Arduino complessivamente assorbe 220 mA.

Essendo la capacità complessiva delle due batterie pari a 12000 mAh è possibile prevedere nel caso peggiore, in cui entrambi i motori siano in stallo un'autonomia di 15.38 ore che sono circa 923 minuti.

2.4 Approfondimenti

Miglioramenti La realizzazione del robot sembra aver soddisfatto gli obiettivi iniziali di una piattaforma modulare e facile da assemblare. Sicuramente le parti da migliorare sono numerose, prima fra tutte è la sostituzione dei pattini con delle caster wheels, per migliorare lo scorrimento durante il movimento. Sempre riguardo alle caster wheels un'altra scelta che potrebbe essere rivalutata è il numero, da portare da due a uno. Due punti di appoggio sono complicati da sistemare in modo tale che non alzino troppo il robot, facendo slittare le ruote, ma striscino entrambi durante il movimento. D'altra parte queste scelte hanno permesso rispettivamente di realizzare un sistema più facile ed economico e una struttura simmetrica. Un'idea contemplata tardivamente è la costruzione di una postazione per caster wheel che preveda anche l'impiego di una coppia formata da un disco encoder e due sensori per trasmettere e ricevere segnali infrarossi. Si tratta del medesimo meccanismo impiegato nei mouse meccanici per comprendere la direzione del movimento da trasmettere al computer. Si tratta infine anche dello stesso meccanismo sfruttato dagli encoder dei motori in

corrente continua. Indipendentemente da questa aggiunta, anche gli encoder dei motori sono necessari per controllare il funzionamento del motore, aggiustare la velocità e fare dead reckoning. Per questo motivo, le ruote stampate presentano otto fori di pari grandezza distribuiti uniformemente lungo tutta la circonferenza. La larghezza dei fori è la medesima dei raggi affinché, ponendo la ruota fra un ricevitore e un trasmettitore a infrarossi, sia possibile misurare il numero di giri che compie, misurando le variazioni del segnale. Infine sarebbe più completo arricchire il sistema di alimentazione con un circuito in grado di misurare approsimativamente la carica residua delle batterie, non solo per motivi pratici, ma anche favorire ad AlpaBot una maggiore coscienza sui propri parametri energetici.

Microcontrollori e microprocessori Una parte degna di nota è la scelta del circuito elettronico per l'elaborazione delle informazioni e i calcoli, tale scelta ricade principalmente fra l'utilizzo di microprocessori o microcontrollori. I microprocessori sono comunemente impiegati per la realizzazione di CPU e GPU montabili sulle schede madre dei moderni computer. I microcontrollori sono un'evoluzione dei microprocessori in quanto integrano su un unico chip le memorie permanenti e volatili, i pin di input/output e i blocchi specializzati. I microcontrollori sono particolarmente adatti per la realizzazione di sistemi embedded e soluzioni specifiche di controllo digitale, perché sono meno costosi e più autosufficienti. D'altra parte però le prestazioni dei microcontrollori sono solitamente inferiori, quindi i microprocessori sono adottati per sviluppare sistemi general purpose. Il mercato offre un'ampia gamma di microcontrollori, la scelta di utilizzare il single-board microcontroller Arduino, e quindi il microcontrollore ATmega168, è stata perlopiù pratica, perché si tratta di una piattaforma nota ed economica. Le schede Arduino forniscono tutte le componenti hardware necessarie per uno sviluppo semplice e veloce. Ad esempio Arduino Uno dispone di dieci convertitori analogici-digitali, tre timer programmabili, una porta seriale e SPI. Da un punto di vista professionale, in ambito robotico, le schede Arduino non sono fra le piattaforme più usate, in particolare quando è necessaria una grande potenza di calcolo oppure una pesante personalizzazione delle componenti hardware. In questi contesti, sono utilizzate componenti elettroniche più generiche e programmabili come FPGA, DSP e PLC, in quanto il microcontrollore RISC ATmega168 dispone di memorie molto limitate: 32 KB di memoria Flash, dove viene caricato anche il programma, 2 KB di memoria SRAM e 1 KB di memoria permanente EEPROM.

Tuttavia in ambito hobbystico Arduino rimane una delle piattaforme più note, tanto che esistono diversi modelli, alcuni ufficiali mentre altri prodotte da terzi, ad esempio le schede Arduino Mini, Nano e LilyPad Arduino USB, inizialmente considerate per la realizzazione di uno swarm robot di piccole dimensioni. Altre piattaforme, Arduino e non, includono direttamente moduli per comunicare tramite WiFi, Zigbee o Bluetooth oppure per controllare motori. Non è da ignorare il fatto che Arduino dispone anche di un ambiente di sviluppo completo disponendo di un IDE basato su un linguaggio proprio detto Processing, orientato verso la semplicità. In alternativa, maggiore potenza di calcolo è ottenibile utilizzando un single-board computer come Raspberry Pi, che rappresenta internazionalmente un'altra nota piattaforma adottata in ambito didattico. Ad esempio la NASA nel 2018 ha lanciato un progetto di condivisione al pubblico di una versione semplificata del rover Curiosity, impiegato per esplorare Marte, utilizzando Raspberry Pi. Una strategia interessante potrebbe essere anche quella di comporre il robot con diverse schede e microcontrollori, ponibili sui diversi piani del robot, per sfruttare i punti di forza di ciascuno e distribuire la computazione.

Costi Infine una nota sui costi, la tabella seguente mostra le spese approssimative delle parti necessarie alla realizzazione di AlpaBot. E' importante considerare che i costi possono variare notevolmente, la stima cerca duque di mediare quello speso realmente con i prezzi medi del mercato al momento della stesura di questa tesi.

Costi	
Componenti	Costo
Due batterie Li-Ion da 3,7 V e capacità di 12000 mAh l'una	10 €
Un microcontrollore Arduino Uno	7 €
Due low cost Geared DC Motors	10 €
Viti, dadi, rondelle e bulloni	4 €
Jumpers di vario tipo, circuito integrato L293D, cavo jack e interruttore	6 €
Stampa 3D, utilizzando circa un terzo di un rotolo di 330 metri di PLA	8 €

Somma	45 €

3 Sistema di controllo

La robotica è una disciplina molto complessa che fonde numerosi campi di studio, compresa la *teoria del controllo*. La teoria del controllo è quella branca dell'ingegneria che studia in maniera matematica il comportamento di un sistema soggetto a variazioni nel tempo. Si tratta di una disciplina alla base di numerose applicazioni automatiche. [23] Il controllo di un sistema autonomo può essere diviso in due livelli di astrazione: quello più basso, e più vicino alla pura teoria del controllo, modella direttamente sensori e attuatori, quello più alto, più vicino all'intelligenza artificiale tradizionale, tende a modellare e definire dei paradigmi di controllo più astratti. I sistemi di controllo sono suddivisi in due famiglie:

Anello aperto (Open loop) basato sull'elaborazione delle informazioni in input per produrne altre di output che il sistema non sfrutta in modo utile.

Anello chiuso (Closed loop) più completo rispetto a quello aperto, perché riporta in input le informazioni prodotte, che si sommano o sottraggono alle altre, influenzando il comportamento complessivo.

Un sistema di controllo ad anello chiuso ampiamente utilizzato a livello industriale è detto *Proportional Integral Derivative (PID)*. Il controllore acquisce con dei sensori il valore reale dell'effetivo lavoro di un attuatore, come ad esempio il numero di rotazioni di un motore, e ne calcola la differenza con quella voluta. Tale differenza, detta errore, viene utilizzata poi da tre funzioni che hanno lo scopo di aggiustare il valore corrente per approssimare il più possibile quello desiderato. Il compito di un sistema proporzionale (P) è quello di rispondere in modo matematicamente proporzionale all'errore, considerando sia la intensità che la direzione di questo. La costruzione di un suddetto sistema richiede una lunga serie di tentativi per calibrare correttamente le costanti proporzionali, in modo tale che siano adatte al problema specifico. Un sistema proporzionale può causare facilmente delle oscillazioni, in quanto all'avvicinarsi dello stato voluto il sistema reagisce in modo indipendente aumentando nuovamente il divario fra il valore voluto e quello misurato. Per diminuire le oscillazioni è possibile considerare la derivata dell'errore, che permette di comprendere quanto velocemente sta cambiando il valore, in modo da limitare l'output finale. L'aggiunta della derivata dell'errore al controllo P da origine a un controllo proportional-derivative (PD). Infine

un controllo PID aggiunge una sorta di memoria dell'errore sommando al PD l'integrale dell'errore, che corrisponde alla somma dell'errore nel tempo, quando questo valore supera una certa soglia il sistema prende le azioni necessarie a compensarlo. Il controllo PID evita che gli errori seppur piccoli ma ripetitivi del sistema si sommino causando gravi imperfezioni. A un livello più astratto invece, si definisce *architettura di controllo* una struttura in grado di gestire più sistemi di controllo, al fine di decidere quali usare in determinate situazioni, con quale priorità assegnare le risorse, determinare l'ordine e i tempi di esecuzione. Le architetture di controllo sono necessarie laddove un sistema autonomo debba compiere più operazioni contemporanemente e concorrentemente, mantenendo un comportamento complessivamente conforme a quello richiesto. Sono numerose le implementazioni che sono state presentate come architetture e utilizzate in ambito didattico o commerciale, la maggior parte di queste possono essere catalogate in una delle quattro ben conosciute famiglie: Controllo deliberativo, reattivo, ibrido e behaviour-based [24].

3.1 Architetture di controllo

Questa sezione riporta gli studi e alcune implementazioni correlate delle architetture più note e reperibili in numerosi testi di robotica, come "Robotics Primer" di M. J. Matarić e "Understanding Intelligence" di R. Pfeifer e C. Scheier, ma anche in moltissimi paper consultabili sul Web. Lo scopo di questa parte è principalmente didattico, ovvero comprendere in maniera analitica le basi dello sviluppo software dei sistemi autonomi. Se finora la parola "sistema autonomo" ha avuto lo scopo limitato di descrivere solamente le caratteristiche fisiche del robot, in questa sezione sarà arricchita di un significato più ampio in grado di coprire anche gli aspetti che concernono le azioni eseguite dal robot in autonomia. Per questo motivo, lo studio e la realizzazione di robot autonomi rientra in una delle tante sottocategorie dell'intelligenza artificiale. In secondo luogo vi è l'interesse di mettere alla prova AlpaBot in scenari più o meno complessi legati al mondo della didattica. E' importante considerare che le implementazioni di seguito presentate sono limitate dalle risorse computazionali e hardware disponibili ma tendono comunque a sviluppare i principali paradigmi che hanno segnato lo sviluppo di diverse soluzioni alla necessità di legare fra loro le tre primitive di un sistema autonomo: *Sense* (percepire), *Plan* (pianificare) e *Act* (agire).

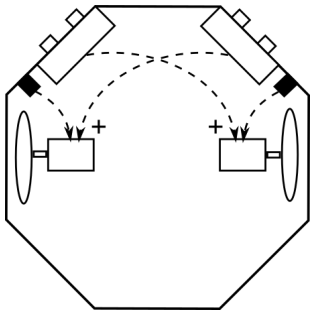
3.1.1 Controllo deliberativo

Il controllo deliberativo è quello più vicino al pensiero più tradizionale dell'intelligenza artificiale, perché mette in primo piano le capacità intellettive del sistema autonomo, rilegando l'interazione con l'ambiente un ruolo marginale. Alla base di questo approcio esiste una sequenza di lavoro molto rigida, che permette di trasformare dei dati in input in un comportamento, passando per una sola procedura detta planning. La fase di planning elabora le possibili azioni e sceglie la migliore, al fine di avvicinarsi in modo ottimale all'obiettivo finale. Essendo il planning composto da un'unica procedura, è molto probabile che, nel caso in cui la richiesta computazionale sia elevata, l'intero processo risulti estramemente lento, limitando le capacità reattive del sistema. In certe situazioni questa architettura risulta necessaria e vincente, ad esempio nella risoluzione di problemi complessi, come una partita di scacchi, ma in molte applicazioni risulta notevolmente lenta rispetto ai cambiamenti che avvengono nell'ambiente circostante. Per questo motivo, tale architettura tende a limitare le interazioni con l'esterno, se non quando necessita dei dati per fare planning, spingendo lo sviluppo di una architettura open loop. I tempi di esecuzione innaturali non sono gli unici aspetti negativi, anche la richiesta di memoria e la necessità di affidarsi su dati precisi e accurati, possono essere delle esigenze difficili da rispettare.

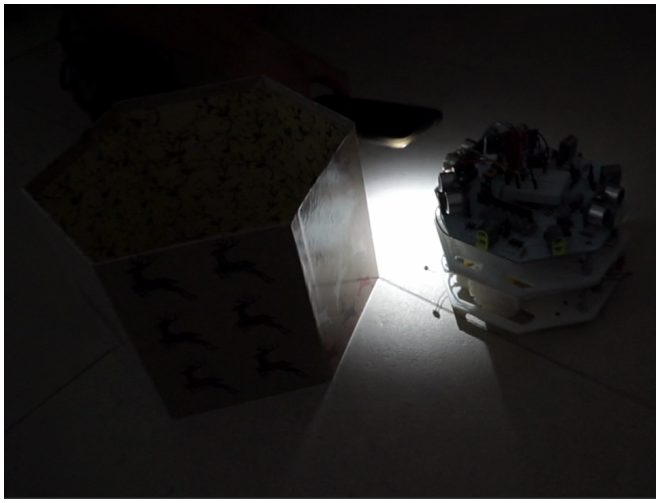
3.1.2 Controllo reattivo

Il controllo reattivo si pone in netto contrasto con quello deliberativo, definendo una struttura che utilizza principalmente i dati ottenuti dai sensori per determinare le azioni da svolgere, arrivando nei casi più estremi a evitare completamente modelli o stati interni. Ad esempio, i *Veicoli di Braitenberg*, quattordici agenti con un livello di complessità crescente, dimostrano che spesso cervelli estremamente semplici possono comunque presentare comportamenti che all'esterno sembrano sofisticati. Braitenberg come neuroscienziato era interessato a comprendere i principi dell'intelligenza perciò, anziché affrontare direttamente suddetti argomenti osservando il complesso mondo biologico, decise di progettare degli agenti che semplificassero notevolemente lo studio e permettessero di costruire da zero un sistema intelligente. Sebbene la maggioranza degli agenti siano puramente reattivi, altri includono forme di apprendimento e memorizzazione, ma in generale la maggior parte di questi possono essere realizzati anche solo con componenti elettronici di base, capaci principalmente di

mappare i segnali ricevuti dai sensori direttamente ai motori. Ad esempio i veicoli 1, 2 e 3 eseguono proprio questa associazione fra valori in input provenienti da qualche sensore ai motori. In particolare il veicolo 1 associa in maniera eccitatoria l'input del sensore, come un fotorivelatore, alla velocità dei motori, permettendo un agente di muoversi lentamente nelle regioni buie e velocemente in quelle illuminate. Il veicolo 2 a differenza dell'1 associa ai due motori due input provenienti da due sensori distinti, immaginando un Differential Wheel Robot su due ruote, se il sensore di un lato eccita il motore dello stesso allora il robot tende ad allontanarsi dalle regioni in cui il sensore legge valori più alti, se invece il sensore di un lato eccita il motore del lato opposto, l'agente si avvicina a tali regioni. Nel veicolo 3 i sensori inibiscono i motori, perciò accade esattamente l'opposto di quello che accade al veicolo 2. Introducendo delle dipendenze non lineari fra motori e sensori è possibile descrivere comportamenti più complessi, oppure introducendo dei nodi intermedi fra i sensori e i motori è possibile che questi si attivino con lo scopo di memorizzare uno stato e quindi introdurre una forma di memoria e apprendimento. I restanti veicoli, arrivano a riconoscere forme, pattern di eventi temporali oppure fare predizioni. La complessità crescente introdotta dai nodi intermedi rimanda a un altro approcio di controllo in ambito robotico, basato sulla costruzione e l'allenamento di una rete neurale artificiale al fine di apprendere il modo migliore con cui coordinare gli attuatori. E' stato scelto di implementare un veicolo di tipo 3c multisensoriale, in quanto dotato di due coppie di sensori, posti rispettivamente ai due lati frontali del corpo, comprendenti un sensore di prossimità, nel dettaglio un sonar, e un fotorivelatore. Il motore di un certo lato è eccitato dal sonar dello stesso lato e dal fotorivelatore dell'altro, permettendo al robot di assumere sia il comportamento "timido" nei confronti degli ostacoli e "aggressivo" nei confronti delle fonti di luce. Ciascun motore regola la velocità facendo la media dei valori eccitatori di entrambi i sensori [32].



(a)



(b)

Figura 11: La figura 11a mostra lo schema di attivazione dei motori del robot secondo i collegamenti eccitatori incrociati. La figura 11b mostra una sequenza del video dell'esperimento.

E' interessante osservare che in questo scenario non è facile stabilire il comportamento del robot in un determinato istante di tempo, neppure provare a estrapolare le azioni che internamente intraprende poiché non possiede delle istruzioni codificate in grado di definirle in maniera netta. Si dice che il comportamento di un robot è *emergente* quando esternamente appare inaspettato, o meglio, si conosce che internamente non è appositamente progettato per eseguire il comportamento che manifesta. Il comportamento emergente si fonda sul principio che il tutto sia più grande della somma delle singole parti, in quanto le varie interazioni che un sistema autonomo ha con il mondo esterno possono dare luogo a situazioni non considerate che prendono il nome di emergenti. Da queste osservazioni nascono una disciplina detta *cibernetica* e, profondamente ispirati ai veicoli di Braitenberg, una vera e propria architettura di controllo detta *Extended Braitenberg Architecture*. Anche i già citati BEAM, essendo tradizionalmente privi di microntrollori, non riescono a svolgere calcoli e memorizzare dati complessi, ma possono avere comportamenti interessanti interagendo con l'ambiente esterno. Un'implementazione semplice consiste nel mappare determinate situazioni, rappresentate da tuple di valori assunti dai vari sensori, a comandi per gli attuatori, definendo quindi un set di regole. Volendo creare un'analogia con il mondo

biologico, tali regole possono essere paragonate ai riflessi caratteristici di molti esseri viventi. Un controllo reattivo permette di costruire un sistema molto veloce ed efficiente, però necessita di una buona progettazione a design time, che associ a tutte le possibili situazioni un comportamento, evitando il più possibile, le intersezioni o l'assenza di tali regole. La struttura a eventi può esere pensata come una serie di task eseguiti in concorrenza, privi di memoria e incapaci di apprendere, che monitorano lo stato reale e corrente del robot.

3.1.3 Controllo ibrido

Il controllo ibrido nasce nel tentativo di fondere il controllo reattivo e deliberativo, definendo una struttura a tre livelli.

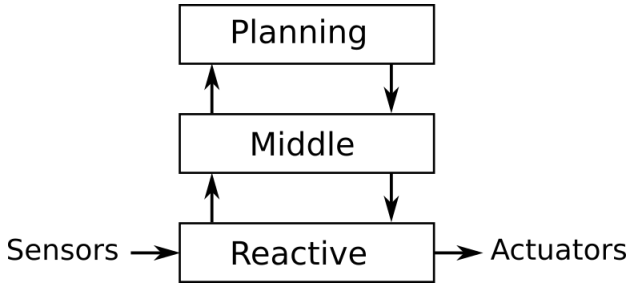


Figura 12: I livelli che compongono l'architettura basata su controllo ibrido.

Il livello più in basso, che dialoga direttamente con i sensori e gli attuatori, rappresenta la parte reattiva dell'architettura, mentre quello più in alto esegue le operazioni di planning tipiche del controllo deliberativo. Lo strato intermedio ha il difficile ruolo di mediatore fra gli altri due livelli, con l'obiettivo di ridurre le differenze fra gli altri due livelli, in termini di tempi di esecuzione e metodi di rappresentazione [25]. Un tipico utilizzo del controllo ibrido permette a un sistema autonomo di eseguire una serie di azioni immediate mentre è in corso il planning per un'azione più complessa. Questo scenario genera però dei conflitti, poiché le informazioni non scorrono solo verso il livello deliberativo, in quanto anche quest'ultimo vuole imporre i propri piani, intervenendo concretamente nelle scelte immediate del sistema. E' per questo motivo che il modo in cui il piano intermedio agisce è decisivo e aperto a combinazioni interessanti, che permettono di cambiare il modo in cui si scelgono le azioni o la frequenza di comunicazione fra la parte reattiva e quella deliberativa. Un tale sistema si

dice che esegue un *dynamic replanning*, in quanto è soggetto a continui e repentini cambi degli obiettivi da compiere nel breve termine. Per evitare questa situazione, il controllo ibrido, a differenza di quello reattivo, può sfruttare la memoria e quindi la possibilità di implementare alcune forme di apprendimento. Essendo molto comune, specie in quei sistemi autonomi progettati per operare in un ambiente specifico, la presenza di eventi simili e frequenti, si è considerato l'uso di risposte preconfezionate, definite a design time e accessibili molto velocemente dal robot. Tali risposte, dette anche *mini plan*, differiscono dalle azioni, perché più complesse, e dai piani del modello deliberativo, perché più veloci. In conclusione il controllo ibrido risulta essere molto adatto in certi contesti, laddove è possibile curare i dettagli significativi del livello intermedio.

AuRA: Autonomous Robot Architecture E' una delle prime architetture ibride sviluppata a metà degli anni ottanta da Ronald Craig Arkin [2]. Essendo un'architettura ibrida, è composta da due parti principali, una rappresentante il controllo deliberativo, mentre l'altra quello reattivo. A sua volta, lo studio originale descrive, la parte di controllo deliberativo così scomposta:

Mission Planner che ha lo scopo di portare il sistema a realizzare gli obiettivi di alto livello, valutando e gestendo eventuali ostacoli. Nasce anche come interfaccia di comunicazione fra uomo e macchina.

Spatial Reasoner utilizzando la memoria a lungo termine costruisce un modello topologico, formato da sequenze di percorsi, da utilizzare per localizzare e sfruttare algoritmi di ricerca dei cammini, come A*.

Plan Sequencer (Pilot) trasforma le sequenze fornite dallo Spatial Reasoner in comandi per il sistema locomotivo. Inizialmente venne concepito in un forma if-then, con regole prestabilite a design time, ma può anche essere progettato come una macchina a stati finiti. Ogni stato rappresenta un comando diverso, le transazioni sono gli eventi che possono accadere in un determinato stato e portare a un altro.

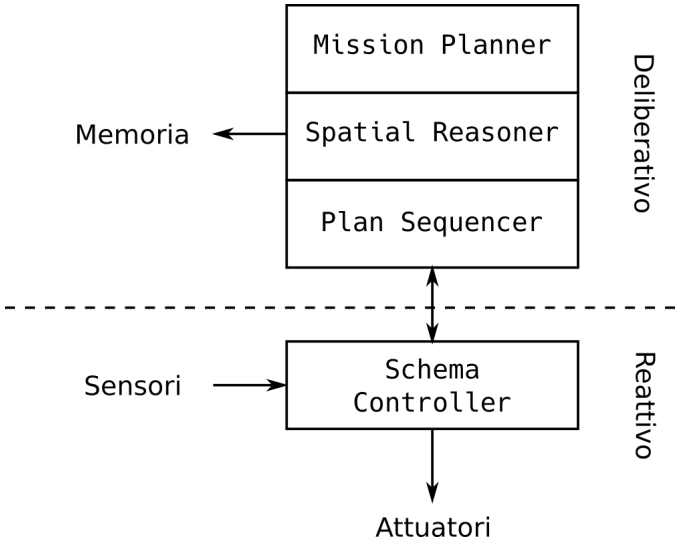


Figura 13: Lo schema dell'architettura ibrida AuRA.

La parte reattiva riceve i dati dai sensori e li processa per contribuire alle decisioni del sistema. La combinazione della parte reattiva e di quella deliberativa avviene usando motor schema, una forma di fusione di comandi che in termini matematici combina più vettori, rappresentanti le direzioni del movimento, in un unico risultato, si rimanda alla sezione 3.3.1 per ulteriori dettagli.

Un esempio di impiego di AuRA Si vuole realizzare un sistema autonomo con architettura AuRA con l'obiettivo di muoversi all'interno di un percorso scegliendo la strada più breve per raggiungere una certa destinazione e gestendo la presenza di eventuali ostacoli lungo il percorso. Per semplicità il percorso è noto a priori ed è rappresentato da un grafo contenente le posizioni assolute e i pesi delle distanze con i nodi vicini. L'implementazione software è costituita da alcune classi in C++ che compongono un programma eseguito sulla scheda Arduino Uno. Il programma esegue un loop continuo in cui vengono letti i valori dei sensori e prodotti i motor schemas da eseguire. La definizione delle classi della parte deliberativa è situata nel file header Deliberative.h, la classe MissionPlanner è inizialmente in attesa di ricevere un nodo di destinazione che viene successivamente comunicato alla classe SpatialReasoner che impiega l'algoritmo di Dijkstra per calcolare il percorso più breve. Il nodo da raggiungere viene comunicato da un operatore umano sfruttando un'interfaccia di comunicazione basata su Bluetooth, ad esempio impartendo i comandi da un'appli-

cazione per smartphone. La funzione interrogata dallo SpatialReasoner che esegue l'algoritmo di Dijkstra ritorna una lista di nodi, che la classe trasforma in una lista di path legs contenente le diverse posizioni da raggiungere e memorizzata fino alla prossima esecuzione dell'algoritmo.

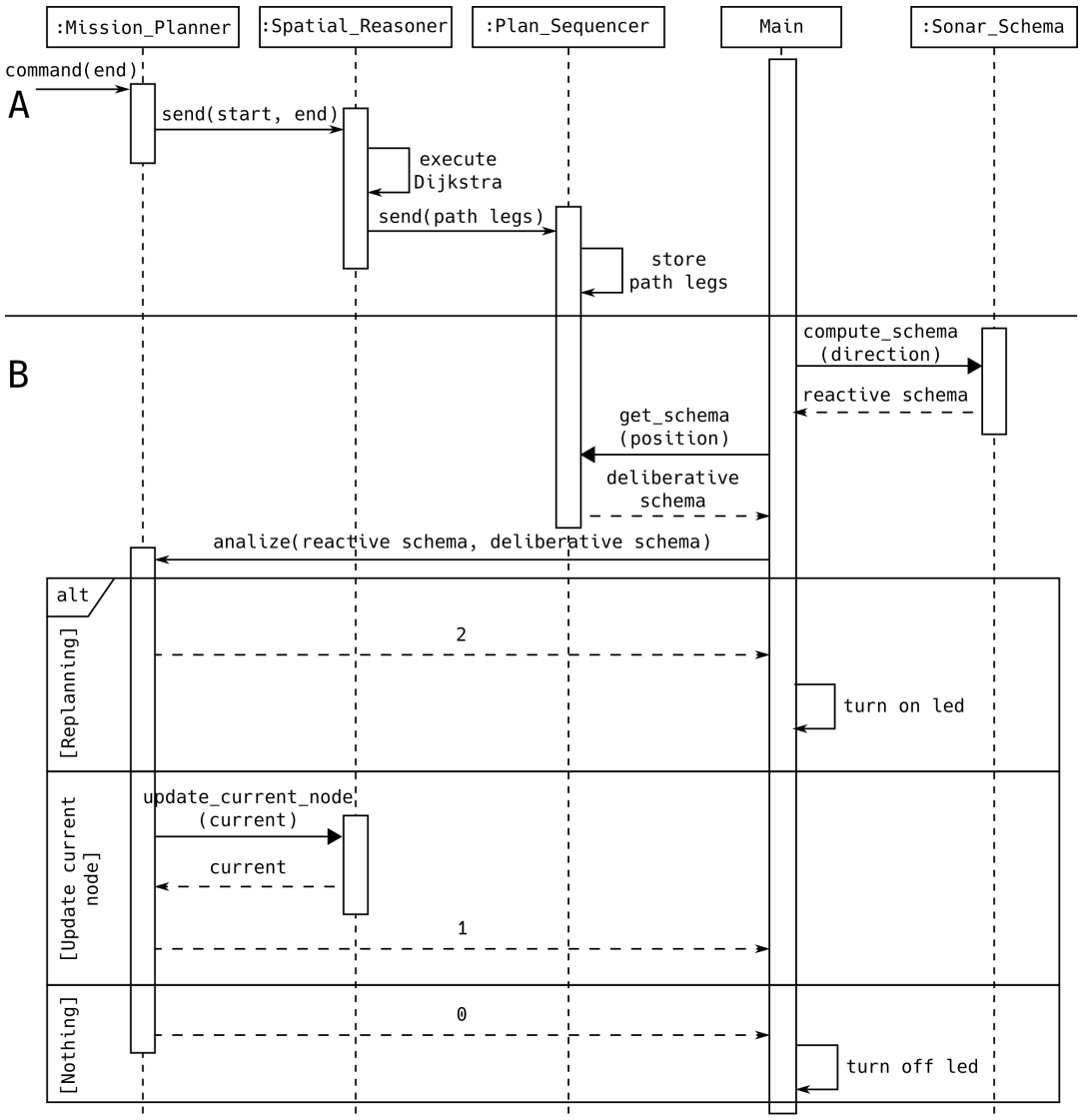
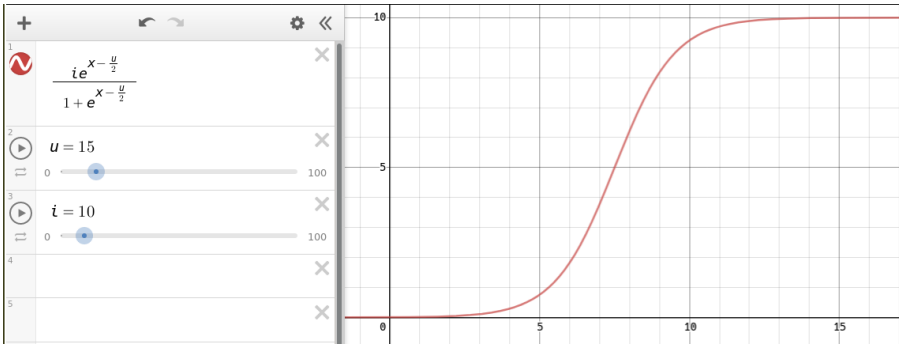
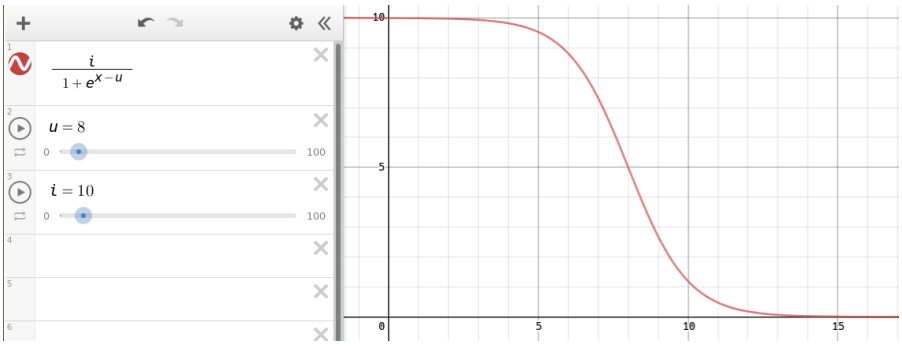


Figura 14: Il sequence diagram del dialogo che avviene fra le classi quando viene scelta una nuova destinazione (A) e quando il flusso principale richiede gli schemi da parte della parte deliberativa e reattiva del sistema che può causare replanning, aggiornamento del percorso oppure nulla (B).

La classe PlanSequencer che accede alla lista dei path legs li transforma man mano in motor schemas utili a generare comandi per il motore. Il flusso principale prima di convertire gli schemi in comandi per i motori permette al MissionPlanner di analizzare se è necessario eseguire un replanning. Gli ostacoli che impongono alla parte reattiva del sistema di generare uno schema complessivo significativamente opposto a quello generato dalla classe del PlanSequencer suggeriscono al robot la presenza di una situazione anomala e l'impossibilità di proseguire senza un replanning. In questo caso il replanning ferma il robot e accende un led rosso per tutta la durata della presenza dell'ostacolo frontale.



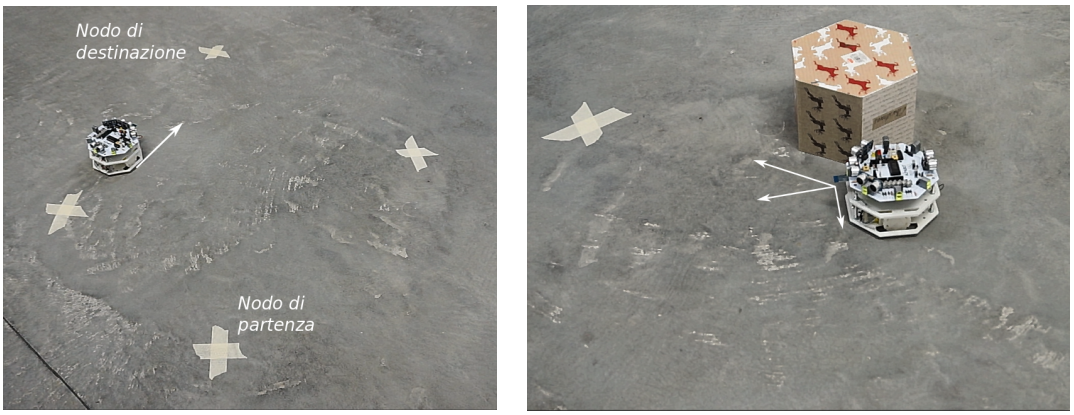
(a)



(b)

Figura 15: La figure mostrano le funzioni e i rispettivi grafi che mappano la distanza nell'intensità dei motor schemas. Nella figura 15a vi è la funzione dell'intensità utilizzata dalla parte deliberativa, all'aumentare della distanza che manca alla posizione finale aumenta l'intensità del movimento. La figura 15b mostra la funzione dell'intensità della parte reattiva, all'aumencompititare della distanza l'intensità tende ad annullarsi.

Gli schemi che invece deviano leggermente il percorso sono fusi assieme a quello deliberativo e gestiti. Durante l'accensione del led è possibile sfruttare l'interfaccia Bluetooth per prendere il controllo del robot e pilotarlo in modo manuale. La classe SchemaController riceve tutti gli schemi, li fonde e poi comanda prima una rotazione su stesso per permettere al robot di essere orientato verso la destinazione voluta e poi una traslazione di intensità variabile a seconda delle distanze fra gli ostacoli e la destinazione. Il robot si muove in modo discontinuo e aggiorna ogni volta la propria posizione e rotazione, ma non possedendo gli strumenti per confermare tali cambiamenti, come encoder o strumenti di visione capaci di localizzare la propria posizione, crea un modello interno che a seguito dell'accumularsi di errori e malfunzionamenti può diventare anche tremendamente errato, si tratta di un tipico esempio di controllo a open loop.



(a) (b)

Figura 16: Le due figure mostrano il robot durante l'esperimento.

ATLANTIS: A Three-Layer Architecture for Navigating Through Intricate Situations Si tratta di un esempio di architettura ibrida introdotta da Erann Gat nel 1992 [13]. In robotica è molto comune l'impiego di macchine a stati finiti per definire il comportamento di un sistema autonomo. In questo scenario il sistema è descrivibile come una sequenza di stati collegati fra loro tramite delle azioni, strettamente legate agli eventi del mondo fisico, ma concettualmente considerate come delle transizioni atomiche e immediate. Questa semplificazione, facilita la modellazione e lo studio, ma rende complessa la progettazione di eventi temporalmente paralleli o l'in-

terruzione di questi e la seguente gestione di stati intermedi. ATLANTIS considera le transizioni come degli eventi temporali anche lunghi, in grado di inizializzare dei processi dette attività. Le attività si attivano fra loro in una forma gerarchica che va da quelle più deliberative e computazionali a quelle reattive. Dalla necessità di realizzare un sistema che permetta il dialogo delle attività, si arriva a una struttura a tre livelli.

Deliberator Rappresenta il classico livello che gestisce le attività più intense e temporalmente più lunghe. Dialoga con il sequencer per aggiornare gli algoritmi e i calcoli necessari al planning o al mantenimento di modelli interni, fornendo dei ”consigli” per il Sequencer.

Sequencer Controlla le sequenze con cui si compongono le attività primitive e complesse del sistema evitando che gli errori compromettano il funzionamento del sistema. L'uso di uno stato interno è essenziale per svolgere correttamente tali compiti, ad esempio per ricordare le azioni svolte nel passato.

Controller Responsabile della gestione delle attività primitive, quelle che hanno una maggiore natura reattiva, tramite l'applicazione diretta della teoria del controllo o di linguaggi pensati appositamente per interagire con e componenti hardware del sistema come ALFA.

3.1.4 Behaviour-Based Controller

the world is its own best model
Rodney Brooks

Tale popolare forma di controllo incorpora la parte migliore dei sistemi reattivi e deliberativi senza scendere ai medesimi compromessi dei sistemi ibridi [26]. In particolare a differenza dei sistemi reattivi, che sono troppo inflessibili perché non inclini ad apprendere e incapaci di rappresentare il mondo esterno che li circonda, i sistemi Behaviour-Based sono costituiti da moduli detti behaviour che sono più complessi delle azioni dei sistemi reattivi, in quanto possono durare più a lungo nel tempo e possono comunicare fra loro, al fine di rappresentare in una forma distribuita uno stato o una rappresentazione. Per rendere il sistema stabile è preferibile impostare i tempi di esecuzione dei behaviour in modo tale da minimizzare le differenze. I behaviour, come le azioni dei sistemi reattivi, sono facilmente modellabili come task

concorrenti, costituiti da una componente di trigger che determina quando è propizio attivarsi e una componente di controllo che mappa i valori dei sensori in comandi per gli attuatori. I behaviours vengono generalmente divisi in due famiglie: i *ballistic behaviours* e i *servo behaviours* [17]. I ballistic behaviours eseguono uno schema di operazioni prestabilito che una volta innescato viene completamente eseguito, come ad esempio una serie di movimenti e rotazioni eseguite per liberare il robot da una posizione senza apparenti vie di uscite. I servo behaviours invece sono sempre attivi, implementano come schema di controllo quello ad anello chiuso, determinando come agire sugli attuatori sempre sulla base dei valori di ingresso. D'altra parte, i ballistic behaviours, possono essere programmati per operare ciecamente una volta innescati, come un controllo open loop.

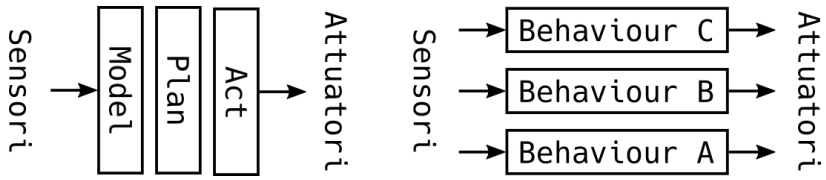


Figura 17: La rappresentazione della differenza fra la sequenza di operazioni svolte dall'architettura deliberativa (a sinistra) e quella svolta dall'architettura behaviour-based (a destra).

3.2 Arbitration

Come descritto in precedenza i sistemi behaviour-based non possiedono un'unità di controllo ben stabilita e centrale, oppure un elenco di regole mutualmente esclusive che determinano in ogni momento le azioni da compiere, ma una serie di moduli o meglio behaviours, che agiscono in modo indipendente, e quindi necessitano di una forma di gestione. Si definisce generalmente arbitration l'insieme delle azioni di mediazione e coordinazione necessarie a soddisfare in modo competitivo una richiesta di risorse che risulta essere maggiore della disponibilità effettiva. In ambito robotico una risorsa può essere un attuatore, ad esempio i motori che permettono il movimento di ruote o pinze, un modulo di comunicazione o un altoparlante. E' possibile costruire una struttura complessa dotata di più arbitri associati a ciascuna risorsa, ma in funzione dello stesso insieme di behaviours [18]. L'implementazione dell'arbitro può variare ma la caratteristica comune è la scelta di un solo comportamento vincente in

modo *competitivo*. Con *Fixed Priority Hierarchy* si definisce una forma di arbitration completamente definita a design time, in cui la scelta dell'ordine di importanza dei behaviour è nota. In caso di conflitto viene scelto automaticamente il behaviour che prevale sugli altri in base alle priorità assegnate. Mentre con *Dynamic Priority Hierarchy* si intende una generalizzazione della Fixed Priority Hierarchy derivante dall'organizzazione di alcune architetture ibride. E' dinamica in quanto evolve la scelta a design time della controparte in una scelta a run time. All'apparenza quest'ultimo metodo risulta essere più maturo e completo, ma in realtà la complessità di un suddetto sistema è notevole, tanto da scoraggiare la maggior parte dei progettisti, in particolar modo a causa delle conseguenze che si possono avere sulla fase di debugging e comprensione dell'interazione fra i diversi behaviour. E' possibile modificare dinamicamente le priorità sfruttando alcuni metodi presenti nel capitolo 3.3 come la Logica Fuzzy o le votazioni, che in questo studio sono rilegati unicamente al controllo degli attuattori.

3.2.1 Subsumption Architecture

Si tratta di una architettura introdotta nel 1986 da Rodney Brooks quando lavorava presso il MIT Artificial Intelligence Laboratory. Essendo Brooks considerato il padre della robotica Behaviour-Based, l'architettura Subsumption è considerata da molti lo standard de facto [9]. Il termine inglese subsumption, dal verbo "to subsume" che in italiano significa incorporare, includere o inglobare, è utilizzato in questo contesto per rappresentare una organizzazione gerarchica in cui ogni livello superiore ha priorità maggiore rispetto a quelli inferiori. L'ispirazione che ha portato all'elaborazione di tale architettura deriva dall'osservazione del mondo naturale, in particolare del processo di evoluzione del cervello umano, che nel suo percorso non ha perso le funzionalità che man mano diventavano meno recenti, ma ne ha acquisite delle nuove basate proprio sulla presenza delle prime [19]. Dal punto di vista progettuale l'architettura Subsumption decompone il sistema di controllo, tradizionalmente centralizzato, in un set di task-achieving behaviours o competenze. La procedura di costruzione di un'architettura basata su Subsumption è bottom-up, perché si parte modellando i livelli più primitivi, quelli legati direttamente ai sensori, muovendosi solo successivamente su quelli più complessi. In particolare si elaborano prima i livelli di competenza, ovvero le descrizioni informali del comportamento esterno desiderato, assegnando a quelli

più basilari i livelli più bassi. E' importante osservare che i livelli superiori dialogano ugualmente con sensori e attuattori senza passare necessariamente da quelli inferiori. Successivamente si progettano i singoli livelli in modo tale che la competenza desiderata avvenga anche senza la presenza degli altri livelli. I livelli sono a loro volta composti da moduli che operano come Augmented Finite State Machines (AFSMs) in grado di interagire fra loro. Le interazioni avvengono solitamente fra AFSM di livelli differenti e possono essere di inibizione o soppressione dei messaggi rispettivamente in input e in output delle AFSM dei livelli inferiori [33].

Myrmix Myrmix è un robot che opera in uno spazio chiuso in cui sono presenti alcuni oggetti che rappresentano simbolicamente delle risorse energetiche. Quando Myrmix trova un oggetto davanti a sé lo "ingerisce" fermandosi alcuni secondi davanti a esso e accendendo un led, poi riprende a muoversi, evitando gli oggetti fino al termine della digestione, rappresentato dallo spegnimento del led. In breve, Myrmix può essere visto come un organismo che si muove cercando cibo ed evitando di collidere con le pareti, poi durante la digestione, anche con gli oggetti stessi. La progettazione del sistema utilizzando l'architettura Subsumption prevede come primo passo la stesura gerarchica dei livelli di competenza, che a partire dal più basso sono: muoversi in avanti in modo sicuro (safe forward), evitare gli ostacoli (avoid obstacle) e nutrirsi (collect). Ogni livello è definito da una serie di moduli indipendenti fra loro. Safe forward è composto da due moduli che controllano i motori in modo da permettere rispettivamente il movimento in avanti e indietro. Il modulo move forward fa uso del sonar frontale per calcolare la distanza dagli ostacoli ed evitare di collidere passando il controllo a move backward, che si comporta in modo analogo. Esternamente è possibile osservare che il robot si muove in avanti o indietro per una quantità di tempo determinata o fin quando non incontra un ostacolo, invertendo di conseguenza la rotta. Per evitare gli ostacoli non visibili dai due sonar impiegati dal livello precedente si aggiunge un nuovo livello che legge le distanze dei sonar posti lateralmente e ruota il robot in modo da evitare le collisioni. I moduli del terzo livello si attivano in modo consecutivo, detect food permette di trovare gli oggetti davanti al robot e attivare il modulo eat che accende il led e ferma il robot davanti all'oggetto inibendo la comunicazione fra i moduli del primo e del secondo livello. Quando eat termina e passa il controllo a digest, che non inibisce nessun modulo, il robot riprende a muoversi evitando gli ostacoli e infine, terminato il periodo di attivazione di digest, il led

si spegne e si riattiva detect food. E' interessante notare alcuni comportamenti emergenti generati dalla combinazione di altri che hanno scopi diversi, come ad esempio il movimento di evasione, che allontana il robot dal cibo una volta terminata la ingestione, e non viene specificato nei moduli del terzo livello ma deriva spontaneamente dai moduli del primo strato.

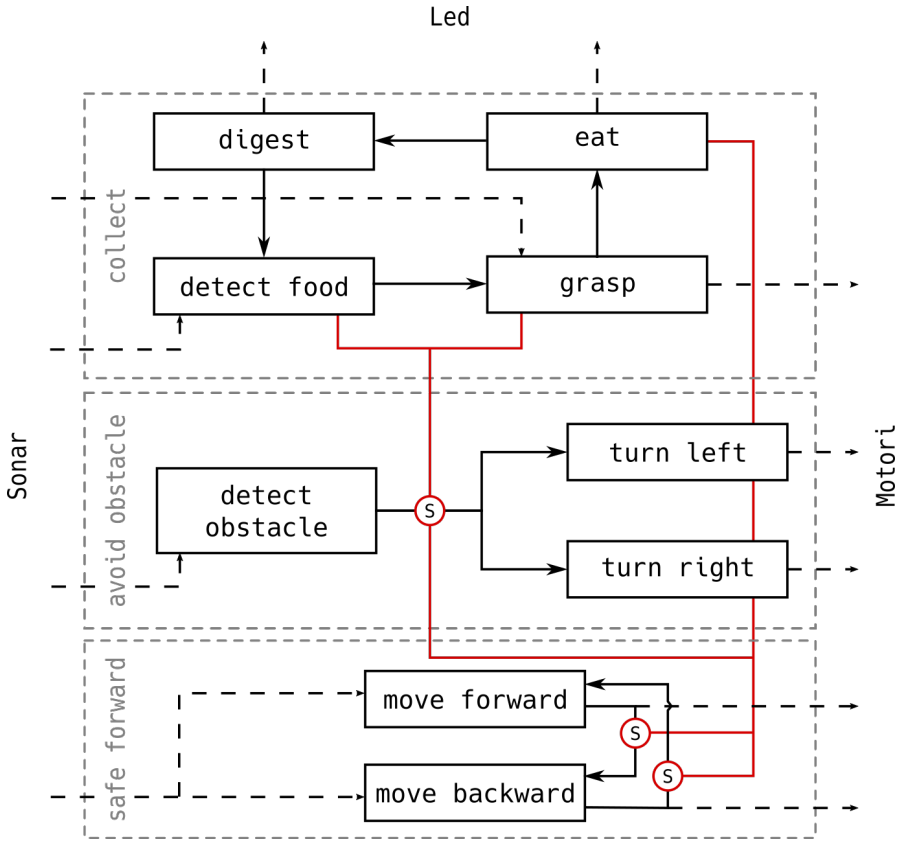


Figura 18: Lo schema del sistema di controllo di Myrmix. Le linee trattegiate indicano sia i valori letti dai sensori che i comandi inviati agli attuattori. Le linee nere continue che terminano con una freccia indicano le linee di attivazione, mentre le rosse che terminano con un pallino contenente una S quelle di soppressione.

Altri comportamenti desiderati invece sono limitati dalle capacità sensoriali del robot, che faticano a riconoscere talvolta il muro dagli ostacoli, in particolare in questa implementazione dotata solo di tre sonar. Un problema che in ogni caso rimane difficile da risolvere è la distinzione fra pareti e gruppi di ostacoli vicini fra loro, si potrebbe specificare che il robot in questo contesto, è interessato a cercare solo quegli

ostacoli che rispettano una certa forma geometrica, computer vision e algoritmi di edge detection potrebbero radicalmente migliorare il sistema sotto questo aspetto.

Implementazione L'implementazione è basata sul paradigma della programmazione a oggetti, ad esempio ogni behaviour è associato a una classe che estende la classe astratta AFSM. AFSM dispone delle proprietà e dei metodi basilari in comune fra tutti i behaviour, come ad esempio la gestione dei vettori dei behaviour attivabili o sopprimibili. Un AFSM può specificare in che stato si trova ma mantiene l'aggiornamento e le decisioni da implementare nelle classi concrete. In particolare la funzione update, richiamata dal flusso di controllo principale è un *Template Method* perché delega al metodo astratto update_body, richiamato all'interno, l'effettiva modifica dello stato, svolgendo solo i controlli necessari. I behaviour sono modellati senza considerare la priorità dato dal livello di appartenenza, è il flusso di controllo che parte eseguendo i behaviour appartenenti ai livelli più alti, gestendo prima le loro richieste come ad esempio quelle di soppressione dei behaviour inferiori e arrivando infine a questi. In questo modo, a seconda dello stato globale del robot, in ogni ciclo vengono attivati e soppressi i giusti behaviour.

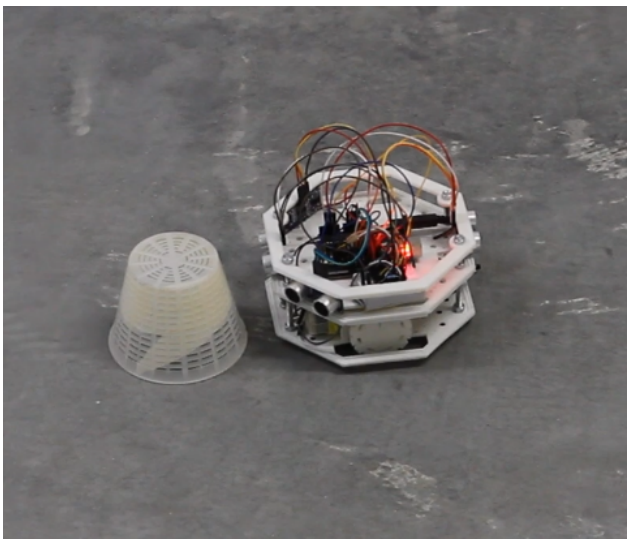


Figura 19: Uno screenshot del video dell'esperimento che mostra il robot in fase di eating di fronte ad un ostacolo.

I sonar sono gestiti dalla classe SonarManager che raccoglie tutti e quattro i sonar

mentre i motori da MotorManager che ogni ciclo è in attesa di riceve dei comandi prestabiliti che sono muoversi in avanti o indietro, girare a destra o sinistra oppure fermarsi.

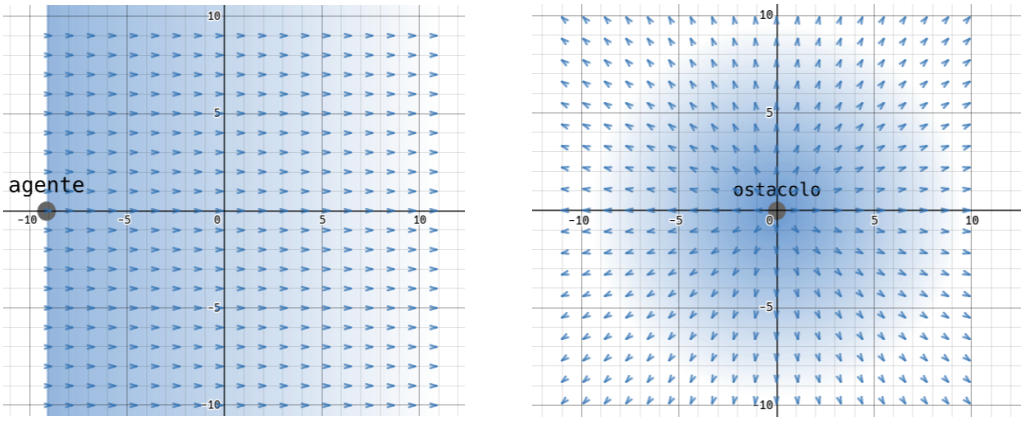
3.3 Fusion

In alternativa all'arbitration, che sceglie in maniera competitiva un vincitore in caso di conflitti fra due o più behaviour, i metodi di fusione combinano fra loro più behaviour in uno singolo, introducendo una filosofia più *cooperativa*. Generalmente si tratta di una tecnica che richiede un'attenzione maggiore durante l'implementazione, un pericolo comune è la presenza all'interno del sistema di comportamenti e regole fra loro contradditorie ed equipotenti, che sommate, possono bloccare il sistema in una fase di stallo, da cui è difficile uscire. Per evitare questo problema si ricorre all'uso di pesi oppure all'implementazione di una logica in grado di rilevare e risolvere tali situazioni. Ad esempio, una tecnica conosciuta come Least Commitment Arbitration sfrutta la frequente situazione in cui i progettisti, sviluppando un sistema autonomo, riescono a definire con maggiore precisione gli eventi che non vorrebbero accadessero piuttosto che tutti quelli che hanno una direzione corretta. Sulla base di questa osservazione, è possibile organizzare un sistema che fonde i coportamenti necessari al corretto funzionamento assieme a uno particolarmente progettato per evitare il verificarsi delle situazioni negative. Di seguito sono presentate delle tecniche più consistenti e ordinate.

3.3.1 Motor Schema

Il concetto di schema deriva dalla psicologia e dalla neurologia, ed è stato applicato alla robotica come modello di riferimento per lo studio dei comportamenti intelligenti. Lo schema è uno strumento di percezione e codifica degli stimoli provenienti dal mondo esterno, in grado di formulare risposte [1]. L'applicazione degli schemi nella robotica è stata estensiva e ha riguardato numerosi aspetti, fra questi il movimento. Un controller famoso che fa uso di motor schema è AuRA, come descritto nel capitolo 3.1.3. Preso un certo spazio, è possibile definire un campo vettoriale che è il risultato della somma dei vettori delle direzioni che i singoli behaviour vorrebbero imporre al movimento del robot. Ad esempio un robot che vuole raggiungere una destinazione necessita di spostarsi verso una certa direzione, per mantenere la rotta e la propria

incolumità deve anche evitare gli ostacoli che incontra sul proprio cammino. Le direzioni estreme che il robot segue per evitare di scontrarsi con gli ostacoli sono quelle perpendicolari alle superfici di quest'ultimi con intensità inversamente proporzionale alla distanza che li separa. La rappresentazione visiva dei campi vettoriali e delle loro somme ricorda quella utilizzata per mostrare le forze attrattive e repulsive dei campi elettrici. Tale metodo, implementato senza altri accorgimenti, non risolve tutti i problemi, ad esempio nei punti in cui tutti i vettori si annullano il robot rimane bloccato. Se il robot può sfruttare la memoria e un buon sistema di localizzazione, allora è possibile aumentare l'intensità dei vettori già percorsi per evitare che la somma sia nulla [20].



(a) (b)

Figura 20: La figure mostrano due motor schema comuni presenti anche nell'esempio del capitolo 3.1.3, l'intensità cresce nelle aree blu. Nella figura 20a vi è uno schema direzionale, il robot si muove verso destra, nella figura 20b vi è invece uno schema repulsivo, dato dall'ostacolo che si trova al centro. La somma di questi schemi permette al robot di muoversi verso destra evitando gli ostacoli.

3.3.2 DAMN: A Distributed Architecture for Mobile Navigation

A Distributed Architecture for Mobile Navigation (DAMN) è un'architettura sviluppata verso la fine degli anni novanta da Julio Rosenblatt Katz presso la Carnegie Mellon University. Rosenblatt, profondamente ispirato dai lavori di Brooks concorda con l'idea di realizzare un sistema distribuito, composto da moduli indipendenti e facilmente intercambiabili fra loro. La critica che muove verso l'architettura Subsumption

di Brooks, lo spinge verso la realizzazione di un sistema di arbitraggio meno gerarchico e più tollerante nei confronti degli errori, che i sistemi autonomi spesso commettono nell'osservazione del mondo esterno [35]. DAMN rappresenta inoltre una valida alternativa ai sistemi centralizzati, come sono frequentemente quelli ibridi che faticosamente bilanciano correttamente le richieste provenienti dai livelli reattivi e deliberativi, offrendo una maggiore flessibilità e robustezza. DAMN è originariamente concepito per operare sfruttando il lavoro concorrete di diversi moduli indipendenti, come i behaviours di Brooks, impegnati a orientare il robot verso un certo comportamento e alla realizzazione di particolari obiettivi. I singoli moduli dotati di un peso, che riflette la priorità rispetto agli altri, assegnano un voto con una certa intensità alle sigole azioni del set disponibile, in base alle loro esigenze. Viene fatta la somma pesata dei voti dei behaviour alle singole azioni e viene eseguita quella con valore maggiore. A differenza dei sistemi reattivi DAMN permette l'uso di modelli interni e la costruzione di moduli con periodi di aggiornamento di lunghezza variabile e obiettivi a breve o lungo termine, rendendo possibile la programmazione di azioni anche di tipo deliberativo. Infine è importante notare che l'intensità dei voti e il valore dei pesi possono cambiare anche a run time, definendo di fatto una dinamicità degli obiettivi del sistema autonomo, similarmente a come un essere vivente cambia i propri comportamenti in base agli stati psicofisici in cui si trova.

Myrmix Per permettere un confronto con l'architettura Subsumption è stato implementato nuovamente Myrmix con un'architettura molto simile a quella DAMN. L'implementazione ricalca abbastanza quella dell'architettura Subsumption, con delle parti di codice, soprattutto quelle di controllo degli attuatori, che sono rimaste pressoché invariate. L'interazione dei vari behaviour in questo caso è indiretta, in quanto i dialoghi avvengono sempre con l'arbitro, permettendo di operare con meno strutture dati e behaviour molto più semplici da progettare. D'altro canto però, non è facile prevedere il comportamento complessivo del robot e decidere a design time l'intensità dei voti. In più, a rendere ancora più complesso il tuning dei valori, è il fatto che i voti sono il prodotto dell'intensità e del peso, che a sua volta dipende dalla modalità in cui si trova a run time il sistema. In questo caso però gli stati che si alternano sono solo tre: HUNGRY, quando il robot cerca cibo, EATING quando è fermo davanti al cibo e DIGESTING quando riprende a muoversi ma non cerca cibo, e sono facilmente prevedibili perché si succedono sempre nel medesimo ordine e raramente sono in

conflitto. In suddetta implementazione il voto espresso dal singolo behaviour è già pesato in quanto ciascun behaviour conosce lo stato corrente e gli eventi che triggerano il passaggio a uno nuovo. Per esempio, quando lo stato del sistema è HUNGRY il behaviour eat, responsabile dell'attesa davanti all'ostacolo e dell'accensione del led, vota sempre con un valore nullo, perché non è interessato a modificare il comportamento corrente. Perciò, sono i behaviour stessi che votando sono responsabili del cambiamento dello stato del sistema.

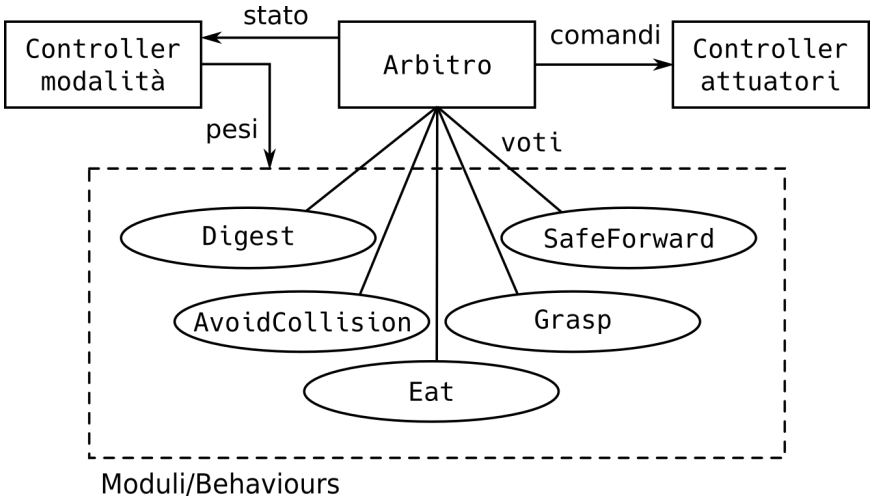


Figura 21: L'organizzazione dell'architettura di controllo per Myrmix profondamente ispirata a quella DAMN.

In conclusione sono stati ottenuti dei risultati pressoché uguali a quelli dell'architettura Subsumption con un'implementazione leggermente più semplice. Non è escluso che in scenari più complessi, ricchi di eventi continuamente in competizione fra loro, l'architettura Subsumption non conceda maggiore controllo delle singole interazioni, evitando una procedura di bilanciamento approssimativa dei voti e dei pesi.

3.3.3 Logica Fuzzy

I primi contributi che portano alla nascita della Logica Fuzzy provengono da alcuni studi fatti nei primi anni sessanta dal professore Lofti A. Zadeh presso la University of California, Berkeley. Nonostante le aspre critiche iniziali, mosse perlopiù dalla comunità scientifica, la Logica Fuzzy si diffuse in tutto il mondo, specialmente in

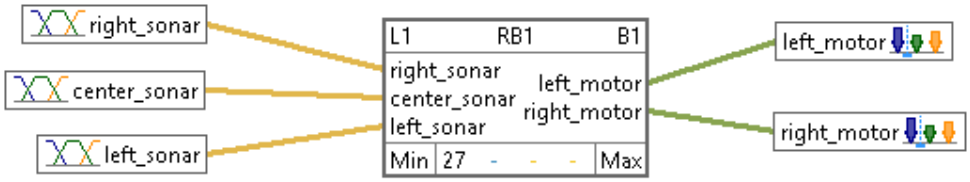
Giappone dove si vide, già a partire dagli anni ottanta, l'impiego pratico per il controllo di numerosi sistemi, dagli elettrodomestici ai trasporti. Nella logica classica, da dove deriva quella booleana, l'appartenenza a un insieme è rappresentabile da due valori distinti: zero, che esprime l'assenza, e uno, che esprime la presenza. Zadeh introduce il termine di Insieme Fuzzy per denotare quegli insiemi i quali elementi posso appartenere parzialmente, con un valore compreso fra zero e uno [12]. E' proprio in questa differenza che si percepisce il motivo per il quale la Logica Fuzzy ha un impatto interessante nel controllo dei sistemi autonomi. La possibilità di utilizzare e generare informazioni che non sono binarie, come invece le macchine sono solite operare, permette di avvicinarsi al mondo reale, in particolare al modo con cui gli umani percepiscono la realtà e ne traggono informazioni utili a prendere decisioni. Infatti, l'uso della Logica Fuzzy in contesti di robotica e automazione, permette l'implementazione dei sistemi tramite l'impiego di regole specificate con un *linguaggio naturale*. L'uso del liguaggio naturale permette di generalizzare più facilmente le regole if-then con cui sono costruiti i controllori, in quanto disaccoppia gli obiettivi astratti del controllo dai valori di soglia effettivi che questo utilizza. Un classico esempio dell'uso della Logica Fuzzy è quello di un sistema di controllo della temperatura, i termini "caldo" o "freddo" non possono essere tradotti in modo assoluto in intervalli rigidi di temperature, è altresì possibile delineare degli intervalli più sfumati dove valori della temperatura sono classificati, con un grado di appartenenza, compreso fra zero e uno, sia all'insieme "caldo" che a quello "freddo". Si definiscono *crisp values* i valori di input e output reali di un sistema, ad esempio la temperatura in gradi centigradi o la distanza in metri, mentre *linguistic variable sets* i gruppi degli insiemi a cui possono appartendere i crisp values, come "caldo" e "freddo" o "lontano" e "vicino". Il controllore che fa uso della Logica Fuzzy può essere composto da tre parti [10]:

Fuzzification Module Transforma i crisp values provenienti dai sensori, in linguistic variables utilizzando una membership function, che rappresenta graficamente il grado di appartenenza fra un crisp value e una linguistic variable.

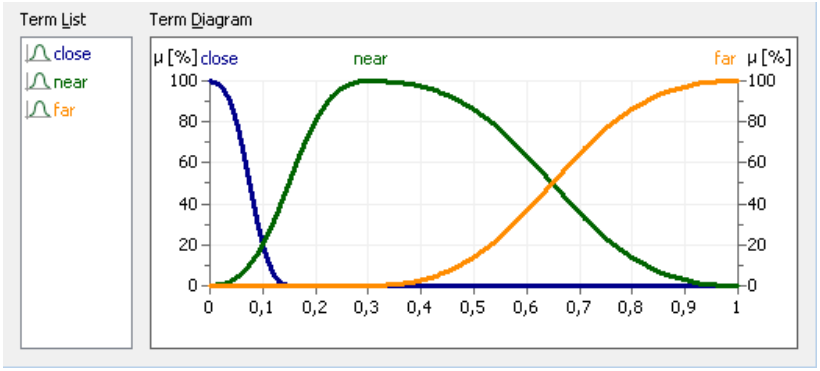
Fuzzy Inference Module Utilizza i valori provenienti dal Fuzzification Module e un set di regole if-then per dedurre il comportamento del sistema generando delle output linguistic variables. E' preferibile che le regole siano definite da esperti del campo di applicazione in cui opera il sistema autonomo.

Defuzzification Module Trasforma le output linguistic variables provenienti dal Fuzzy Inference Module in valori utilizzabili dagli attuatori, detti nuovamente crisp values.

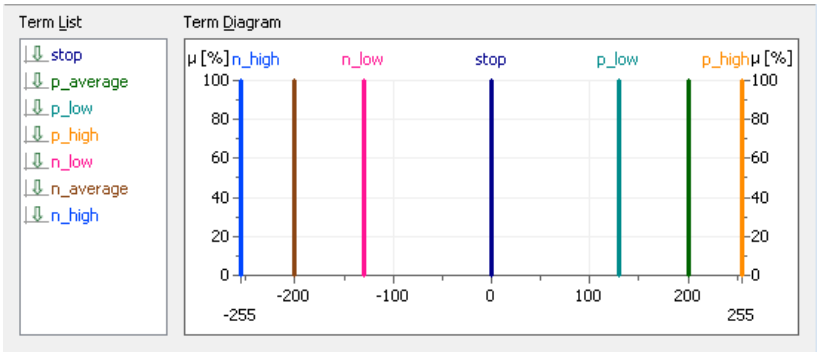
Wall Follower Un robot progettato per evitare gli ostacoli è emergentemente capace di seguire i muri di un percorso. In particolare, si immagini un robot dotato di tre sensori di prossimità, uno frontale e due laterali, tali per cui è possibile misurare la distanza del muro sia a sinistra che a destra del robot. Il robot si muove sempre in avanti, se si avvicina troppo verso una parete devia la rotta verso la direzione opposta, se trova un ostacolo di fronte ed è libero di andare sia a sinistra che a destra, va verso quest'ultima direzione. La scelta di un'implementazione con Logica Fuzzy permetterebbe al robot di realizzare dei movimenti molto più fluidi di quelli che eseguirebbe se operasse con soglie binarie, perché accelererebbe dove è possibile e decelererebbe laddove è necessario. L'implementazione è realizzata utilizzando fuzzyTECH, un software riconosciuto internazionalmente per lo sviluppo di soluzioni che utilizzano Logica Fuzzy e Neural-Fuzzy. Dapprima si scelgono le variabili linguistiche in input e output, la distanza misurata dai sonar può essere classificata nelle variabili linguistiche ”close”, ”near” e ”far”. La velocità dei motori può ricadere in un intervallo positivo o negativo e avere un'intensità nulla, ”low”, ”average” o ”high”. Successivamente si definiscono i moduli di Fuzzification dei crisp values provenienti dai tre sonar impiegando delle membership functions, e i moduli di Defuzzification, per mappare le variabili linguistiche di output all'intensità e la direzione con i quali azionare i singoli motori. Infine sono dichiarate in un blocco le regole if-then che legano le variabili linguisitiche di input con quelle di output. In questo contesto le condizioni sulle variabili di input sono legate con una funzione di minimo, che corrisponde all'operatore AND logico. Ne segue che l'attivazione della regola avviene se tutte le condizioni sono rispettate, è possibile anche utilizzare funzioni di massimo, ovvero l'OR logico, o di composizione, come la media pesata. FuzzyTECH permette di definire il funzionamento del Defuzzification Module utilizzando due metodi diversi, Center-of-Maximum (CoM) o Mean-of-Maximum (MoM). In questo caso è stato scelto CoM, che esegue la media pesata, con i risultati dell'inferenza, dei valori corrispondenti alle variabili linguistiche di output coinvolte, mentre MoM prende il valore della variabile linguistica che ottenuto il valore maggiore. Avviando la simulazione è possibile modificare i valori in ingresso e vedere come variano i comandi ai motori.



(a)



(b)



(c)

Figura 22: La figura 22a mostra come sono legati fra loro i moduli, la figura 22b mostra la membership function dei Fuzzification Modules di uno dei tre sonar mentre la 22c il Defuzzification Module di uno dei due motori.

Conclusioni

In generale, è possibile affermare che il lavoro è terminato raggiungendo sia gli obiettivi iniziali che quelli che si sono aggiunti man mano durante lo sviluppo. AlpaBot è un robot in grado di muoversi, di integrarsi facilmente con altri sensori, disponendo di un certo numero di pin liberi, e in particolare di interfacciarsi col sensore di comunicazione e visione realizzato parallelamente al lavoro qui presentato e utilizzato in alcune implementazioni del capitolo 3. E' possibile replicare alcune delle più importanti architetture di controllo e condurre esperimenti. E' altresì vero che le modifiche e i miglioramenti apportabili sono numerosi, la maggior parte dei quali sono descritti nel dettaglio nelle corrispondenti parti della tesi. Lo sviluppo di alcune architetture di controllo ha avuto un ruolo di benchmarking, oltre che di apprendimento, in fututo sarebbe interessante sviluppare implementazioni per scopi più specifici, magari in scenari con più agenti. A tal proposito, di seguito è presentata una tabella con costi e descrizioni di alcuni robot, principalmente Swarm perché più simili ad AlpaBot, tenendo presente che la professionalità nella realizzazione di alcuni di questi supera talvolta di gran lunga quella di AlpaBot e che l'intenzione è solo illustrativa, non necessariamente comparativa. E' inevitabile osservare che in generale la differenza principale fra questi modelli e AlpaBot, oltre che nell'uso di un numero maggiore di sensori, probabilmente di maggiore qualità, risiede nell'impiego di microcontrollori molto più performanti di quelli usati e considerati in questo studio. Secondo la visione general purpose adottata lungo tutto questo studio, i sensori devono poter essere incorporati o meno a seconda delle esigenze e quindi non delineano intrinsicamente le caratteristiche del robot. Contrariamente, sebbene anche l'unità di calcolo possa subire modifiche on demand, esso rappresenta una parte molto più elementare del sistema, e contraddistingue maggiormente un'implementazione dall'altra. Le prestazioni dei microcontrollori elencati precedentemente rispetto a quelli dell'ATmega168 sono nettamente migliori ne segue quindi una maggiore versatilità del robot. Anche i due modelli che sfruttano l'ATmega168 lo assemblano comunque all'interno di un circuito personalizzato. I motivi della scelta di impiegare Arduino e le alternative sono descritti nella sezione 2.4.

Nome	Costo	Caratteristiche
Colias [4]	31 $	Swarm Robot dal design open source, di dimensioni inferiori, dotato di 6 sensori infrarossi, e relativamente veloce. Monta due ATmega168 in parallelo che si occupano rispettivamente della comunicazione e della locomozione.
e-puck2 [14]	858 $	Robot open source prodotto e venduto da diverse compagnie, dotato di 8 sensori di prossimità, camera a colori, microfoni, speaker e accelerometro 3D. Possiede un'autonomia di 2 ore in movimento e dimensioni intermedie. Monta un 32-bit STM32F407 che opera fino a 168 MHz, possiede 192 KB di RAM e 1024 KB di memoria Flash. Supporta estensioni per comunicazione e visione.
Jasmine [36]	130 $	Swarm Robot dal design open source, di dimensioni estremamente piccole ma aperto a espansioni tramite moduli aggiuntivi, dispone da 4 a 6 sensori di prossimità, di un supporto per batterie Li-Poly di varie capacità ma in generale in grado di fornire un'autonomia di un paio ore. Monta un ATmega168.
r-one [27]	200 $	Swarm Robot totalmente open source, di dimensioni leggermente inferiori ad AlpaBot, con un'autonomia di 4 ore di movimento. Dotato di 8 sensori infrarossi, encoders, giroscopio, accelerometro e fotorivelatori. Monta un Stellaris LM3S8962 basato su processore ARM Cortex-M3 in grado di operare fino a 50 MHz, possiede una memoria Flash da 256 KB e una SRAM da 64 KB.

Sicuramente la sostituzione di Arduino con un sistema più personalizzato e performante eliminerebbe l'attributo "prototipo" da AlpaBot, ma richiederebbe uno sforzo non indifferente per farlo tornare un modello plug and play come è correntemente impiegando Arduino. In conclusione, il desiderio ultimo sarebbe quello di poter considerare AlpaBot come una versione primordiale di un progetto avviabile presso l'Università di Bologna concernente la progettazione di una piattaforma low cost che permetta lo studio e la ricerca nell'ambito di sistemi autonomi e intelligenti.

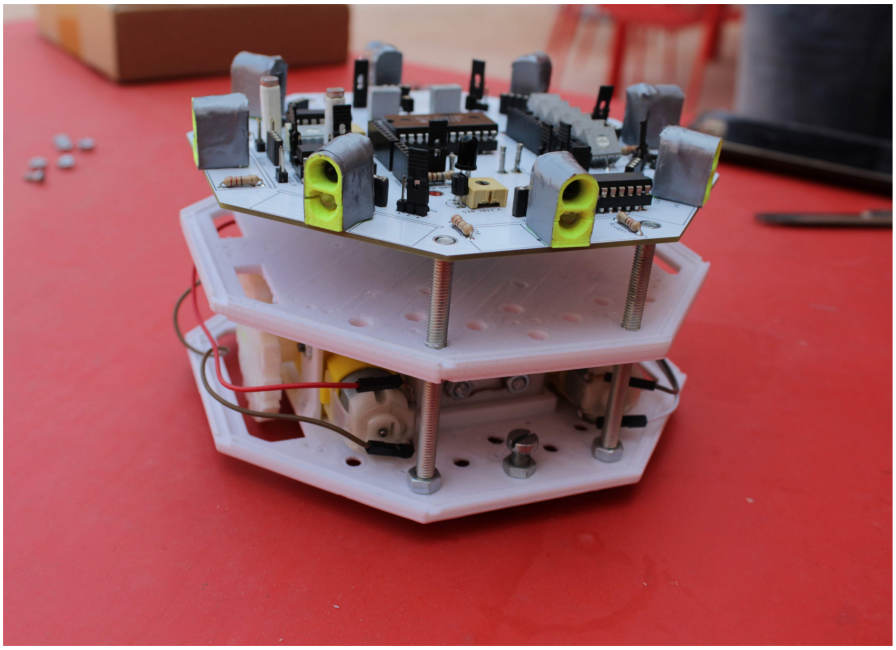


Figura 23: Il robot durante le fasi di aggiunta del sensore di comunicazione e visione, che compone il terzo livello

Riferimenti bibliografici

[1] R. C. Arkin. Motor schema — based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, 1989.

[2] R. C. Arkin and T. Balch. AuRA: principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):175–189, 1997.

[3] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox. Mona: an affordable open-source mobile robot for education and research. *Journal of Intelligent & Robotic Systems*, 05 2018.

[4] F. Arvin, J. Murray, C. Zhang, and S. Yue. Colias: An autonomous micro robot for swarm robotic applications. *International Journal of Advanced Robotic Systems*, 11:1, 07 2014.

[5] G. A. Bekey. *Autonomous Robots: from a Biological Inspiration to Impementation and Control*, chapter 7, pages 186–197. The MIT Press, 2005.

[6] G. A. Bekey. *Autonomous Robots: from a Biological Inspiration to Impementation and Control*, chapter 8, pages 253–254. The MIT Press, 2005.

[7] G. A. Bekey. *Autonomous Robots: from a Biological Inspiration to Impementation and Control*, chapter 9, page 286. The MIT Press, 2005.

[8] G. A. Bekey. *Autonomous Robots: from a Biological Inspiration to Impementation and Control*, chapter 9, page 291. The MIT Press, 2005.

[9] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 3 1986.

[10] Chen C., Wang C., Wang Y. T., and Wang P. T. Fuzzy logic controller design for intelligent robots. *Mathematical Problems in Engineering*, 2017.

[11] G. Caprari. *Autonomous Micro-Robots: Applications and Limitations*. PhD thesis, École polytechnique fédérale de Lausanne, 2003.

[12] T. Lippincott E. Tunstel and M. Jamshidi. Introduction to fuzzy logic with application to mobile robotics. *Proc. 1st National Students Conference of the National Alliance of NASA University Research Centers*, 1996.

[13] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 809–815, 1992.

[14] GCtronic. Shop. `https://www.gctronic.com/shop.php`.

[15] Plum Geek. Home. `https://www.plumgeek.com/ringo.html`, 3 2015.

[16] N. Hurst. These $10 robots will change robotics education. `https://www.wired.com/2012/09/afron-winners/`, 9 2012.

[17] J. L. Jones. *Robot Programming : A Practical Guide to Behavior-Based Robotics*, chapter 3, pages 51,52. McGraw-Hill Education, 2004.

[18] J. L. Jones. *Robot Programming : A Practical Guide to Behavior-Based Robotics*, chapter 4, pages 79–83. McGraw-Hill Education, 2004.

[19] J. L. Jones. *Robot Programming : A Practical Guide to Behavior-Based Robotics*, chapter 4, page 93. McGraw-Hill Education, 2004.

[20] J. L. Jones. *Robot Programming : A Practical Guide to Behavior-Based Robotics*, chapter 4, page 99. McGraw-Hill Education, 2004.

[21] Solarbotics Ltd. Home.

[22] S. K. Malu and J. Majumdar. Kinematics, localization and control of differential drive mobile robot. *Global Journal of Researches In Engineering: H Robotics & Nano-Tech*, 14, 2014.

[23] M. J. Matarić. *The Robotics Primer*, chapter 2, pages 7,8. The MIT Press, 2007.

[24] M. J. Matarić. *The Robotics Primer*, chapter 11, pages 135–142. The MIT Press, 2007.

[25] M. J. Matarić. *The Robotics Primer*, chapter 15, pages 177, 178. The MIT Press, 2007.

[26] M. J. Matarić. *The Robotics Primer*, chapter 16, pages 187–188. The MIT Press, 2007.

[27] J. McLurkin, A. McMullen, N. Robbins, G. Habibi, A. Becker, A. Chou, H. Li, M. John, N Okeke, J. Rykowski, S. Kim, W. Xie, T. Vaughn, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, L. Langford, J. Hunt, A. Boone, and K. Koch. A robot system design for low-cost multi-robot manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 912–918, 2014.

[28] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 5 2009.

[29] F. Mondada, E. Franzi, and A. Guignard. The development of khepera. 12 2010.

[30] F. Mondada, F. Riedo, and P. Dillenbourg. *Thymio: a holistic approach to designing accessible educational robots*. PhD thesis, École polytechnique fédérale de Lausanne, 2015.

[31] University of Manchester. Mona - autonomous mobile swam robot. `https://uomrobotics.com/nuclear/roboticplatforms/miniature/mona.html`, 8 2015.

[32] R. Pfeifer and C. Scheier. *Understanding Intelligence*, chapter 6, pages 181–196. A Bradford Book, 2001.

[33] R. Pfeifer and C. Scheier. *Understanding Intelligence*, chapter 7, pages 202–206. A Bradford Book, 2001.

[34] Swarmrobot Project. Home. `http://swarmrobot.org`.

[35] J. K. Rosenblatt. DAMN: a distributed architecture for mobile navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):339–360, 1997.

[36] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE Intl. Conf on Robotics and Automation (ICRA)*, 2012.

[37] U. Saranli, M. Buehler, and D. E. Koditschek. Rhex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.