UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE AGRARIE E ALIMENTARI

REALIZZAZIONE DI UN PROTOTIPO DI *CABLE ROBOT* A SCALA DI LABORATORIO PER APPLICAZIONI DI MONITORAGGIO E GESTIONE COLTURALE A ELEVATA PRECISIONE

Relatore:

Prof. OBERTI Roberto

Correlatore:

Prof. FERRARI Enrico

Candidato:

TORRENTE Marco Davide Michel

Matr.902355

A.A. 2018/2019

INDICE

RIASSUNTO

Il contesto generale di innovazione tecnologica di questo periodo storico, fortemente basato sull'impiego di enormi quantità di dati acquisiti da diverse sorgenti, sta portando in maniera sempre più importante a una generale adozione delle metodologie e tecnologie proprie dell'informatica e dell'automazione in tutti i settori.

L'agricoltura nella sua evoluzione ha attraversato diversi stadi di sviluppo tecnologico, a cominciare dalla sostituzione del lavoro manuale grazie alla meccanizzazione. I mezzi meccanici impiegati in agricoltura, nel tempo, sono diventati sempre più complessi e dotati di soluzioni tecnologiche innovative.
L'avvento dell'agricoltura di precisione, basata sulla modulazione degli *input* produttivi secondo le potenzialità produttive sito-specifiche, ha portato a un ulteriore sviluppo del settore che ha acquisito, in questo modo, strumenti tecnici ad elevato contenuto tecnologico in grado, dunque, di garantire un risparmio o un'ottimizzazione di risorse e tempo, portando al contempo a una riduzione del potenziale impatto ambientale delle operazioni di campo.

In futuro ci si attende che l'agricoltura compia un ulteriore salto tecnologico per far fronte alle sfide dei prossimi decenni, dato che, le stime di aumento della popolazione mondiale accompagnate ad una riduzione della popolazione rurale, renderanno indispensabile un livello di efficienza di impiego dei fattori produttivi possibile solo grazie ad approcci di altissima precisione.
Per questi motivi, applicazioni di robotica potrebbero trovare spazio in agricoltura, facendo evolvere l'attuale risoluzione spazio-temporale degli approcci di agricoltura di precisione verso valori che si potrebbero definire propri di un'agricoltura di ultra-precisione.
In questo contesto, diversi componenti sensoristici sarebbero in grado non solo di acquisire elevati volumi di dati ad elevata risoluzione spaziale e temporale da processare per prendere decisioni a posteriori ma collegarsi in tempo reale a reti multi-nodo così da scambiare e confrontare dati in modo da prendere decisioni istantanee basate su più livelli informativi e compiendo direttamente operazioni colturali con l'esecuzione di interventi mirati dotati di grande precisione spaziale.
In questo contesto e considerando la scala di lavoro dell'attività agricola, tra le possibili applicazioni robotiche diventa interessante considerare l'impiego dei *cable driven parallel*

robot o più semplicemente, *cable robot,* ovvero macchine capaci di movimentare carichi, anche di entità notevole, su ampie distanze per mezzo di cavi flessibili manovrando lo spostamento tridimensionale di un terminale (*end effector*) nello spazio. Sull'*end effector* è ovviamente possibile montare piattaforme multi-sensore per compiere azioni di monitoraggio ad elevata precisione, oppure attuatori dedicati allo svolgimento di operazioni colturali mirate, quali ad esempio trattamenti selettivi sulla coltura, anche in caso di inaccessibilità del campo a causa della taglia delle piante o per le condizioni del fondo.

In questo lavoro di tesi, dopo un'analisi della letteratura scientifica di settore focalizzata sulle principali applicazioni dei *cable robot* e delle problematiche progettuali di tali sistemi, si è proceduto alla progettazione e alla realizzazione di un prototipo di *cable robot* a quattro motori, costruito a scala di laboratorio ma, comunque, sufficiente a svolgere alcune sperimentazioni applicative preliminari su lotti di piante coltivate in vaso.

L'architettura generale del sistema è stata basata su un controllo di tipo distribuito, gestito localmente da microcontrollori *low-cost* (Arduino Uno), programmati in ambiente Arduino per pilotare un singolo motore in corrente continua, la cui posizione viene monitorata in tempo reale da *encoder* angolari che forniscono un segnale di *feedback*. Ogni unità (microcontrollore + motore), è inserita dentro una rete di comunicazione dati che la interconnette ad altri nodi mediante un protocollo di tipo *CAN BUS,* lo standard universalmente adottato nelle macchine e impianti agricoli per la modularità e versatilità di funzionamento.
Oltre ai quattro microcontrollori, fanno parte della rete *CAN* anche l'*end effector*, anch'esso dotato di un microcontrollore dedicato a interfacciarsi con eventuali sensori o attuatori montati, oltre che a un *device* di gestione manuale del *robot* e di interfaccia con l'unita *master,* costituita da un PC preposto al calcolo in tempo reale della cinematica del robot oltre che all'acquisizione e registrazione di dati.
Il prototipo realizzato è stato poi oggetto di una serie di prove sperimentali finalizzate a validarne l'effettiva funzionalità. Tali prove hanno riguardato: l'affidabilità di posizionamento dell'*end effector* in punti *target* all'interno dello spazio di lavoro; la scansione dello spazio di lavoro mediante sensore a ultrasuoni per ricostruire tridimensionalmente la mappa di occupazione dell'area di lavoro, individuando la posizione di bersagli (piante di cetriolo in vaso) o di ostacoli presenti al suo interno applicando e confrontando due diversi metodi di analisi geometrica; una missione di lunga durata con interventi di fertirrigazione di precisione su piante di cetriolo in serra.

1 INTRODUZIONE

Questo lavoro di tesi si inquadra nel contesto generale del settore agricolo a livello comunitario, con particolare riferimento sfide future che coinvolgeranno molto probabilmente questo settore e al ruolo che può rivestire l'innovazione tecnologica nella ricerca di soluzioni sostenibili. Segue poi un approfondimento sul concetto di agricoltura di precisione, sulle principali tecnologie attualmente impiegate e su quelle che potrebbero essere adottate in futuro.
I dati presi in considerazione in questa parte introduttiva fanno per lo più riferimento ad uno studio commissionato dal Parlamento europeo (Schrijver,2016).

1.1 Contesto del settore agricolo europeo

Per fornire una contestualizzazione e un inquadramento generale del settore agricolo a livello comunitario, si possono descrivere in modo sintetico le sue caratteristiche principali.

La superficie agricola utilizzata (*SAU*) europea è di circa 174 (Mha) che equivale a circa il 40% della superficie totale del continente. Del terreno agricolo europeo, il 60% è destinato ai seminativi, il 34% è occupato da pascoli permanenti e pascoli e il restante 6% da colture permanenti come: frutta, vigneti, olive, agrumi, bacche e noci.
Nel 2013 si contavano 10,8 milioni aziende agricole nell'*UE*, dotate di un'estensione media di 16,1 ha.

Negli ultimi anni si è verificato un calo del numero assoluto di aziende agricole, in particolar modo tale fenomeno ha riguardato quelle aziende caratterizzate da dimensioni più contenute e ciò ha portato di conseguenza ad un aumento delle dimensioni medie aziendali. Questo probabilmente è dovuto al consolidamento graduale delle aziende più grandi e di quelle che sono riuscite ad ampliare nel tempo le proprie dimensioni perché condotte in maniera maggiormente efficiente e innovativa.

Un'altra caratteristica che ha influenzato le dinamiche settoriali è che più del 30% degli agricoltori ha un'età superiore ai 65 anni e solo il 6% ha meno di 35 anni. Inoltre, la maggior parte degli agricoltori europei non ha compiuto studi specifici in campo agricolo,

infatti, il 70% di questi possiede solo esperienza pratica acquisita sul campo, il 20% ha ricevuto una formazione di base e solamente l'8% ha portato a termine un percorso formativo completamente dedicato al settore. Tali caratteristiche hanno contribuito nel tempo in maniera negativa alla promozione dell'innovazione tecnologica e al rinnovo del settore in generale.

I quattro pilastri del settore agricolo comunitario sono rappresentati dall'allevamento di bovine da latte, di suini e pollame mentre per ciò che riguarda le coltivazioni rivestono un ruolo di prima importanza i cereali, le oleaginose e le colture proteiche. Tra le colture vanno citate anche la vite, la frutticoltura e l'arboricoltura per il ruolo molto importante che rappresentano a livello di valore della produzione, specie in Italia e Francia.

Complessivamente, l'area di terreno disponibile per l'agricoltura sta gradualmente diminuendo principalmente a causa del consumo di suolo dovuto all'urbanizzazione. Tale riduzione comporterà necessariamente in futuro la necessità di un aumento dell'efficienza dei fattori della produzione per mantenere costante o aumentare la produzione totale, specie se si considerano le previsioni future di crescita demografica globale, con proiezioni che vedono la popolazione globale toccare 9 miliardi entro il 2050.

Si prevede che l'agricoltura globale negli anni a venire dovrà dunque affrontare una serie di importanti sfide riconducibili principalmente alle seguenti cause: rapida crescita della popolazione mondiale, cambiamenti climatici, crescente domanda di energia, carenza di risorse, aumento del fenomeno dell'urbanizzazione, cambiamenti delle abitudini alimentari specie nei paesi che hanno registrato uno sviluppo economico più marcato, invecchiamento della popolazione nelle zone rurali dei paesi sviluppati e per l'aumento della concorrenza sui mercati mondiali.

Queste sfide renderanno, in futuro, ancora più centrale il ruolo dell'agricoltura e risulterà cruciale la sua corretta ed efficiente gestione. Un aiuto all'agricoltura del futuro dovrà quindi necessariamente arrivare dall'innovazione tecnologica che, attraverso l'applicazione delle metodologie dell'agricoltura di precisione, potrebbe supportare il settore agricolo nel raggiungimento dei suoi obiettivi di aumento della produzione migliorando i livelli di efficienza e attenuandone gli impatti.

1.2 L'agricoltura di precisione

1.2.1 Che cos'è

L'agricoltura di precisione o *precision farming*, è un moderno insieme di tecniche di gestione dell'azienda agricola basato sull'utilizzo di tecnologie elettroniche ed informatiche atte a monitorare e ottimizzare il processo produttivo, sia nella produzione di specie vegetali che di quelle animali *(precision feeding)*.
Tali metodologie consentono di aumentare la qualità e la quantità dei prodotti modulando in modo sito-specifico e tempo-specifico gli interventi colturali (dove, quando e quanto occorre) che, utilizzando con maggiore efficienza gli *input* (acqua, energia, fertilizzanti e fitofarmaci), sono caratterizzati da un minore impatto ambientale e da una maggiore sostenibilità economica.
Lo scopo è quindi di ridurre i costi di produzione e diminuire l'impatto ambientale senza penalizzare la produzione totale, o eventualmente incrementandola, aumentando il reddito dell'agricoltore.

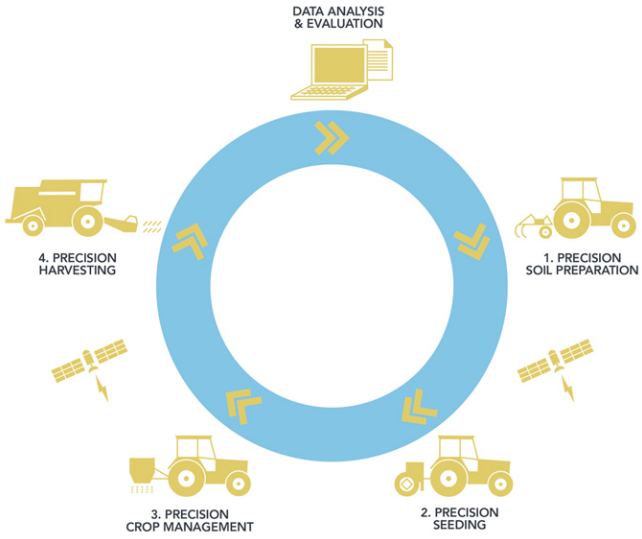

Figura 1.1 Processo produttivo agricolo e precision farming

Il requisito fondamentale per lo svolgimento di tale attività è di possedere informazioni sufficientemente accurate sui fenomeni che coinvolgono le colture e che tali informazioni siano corrette, tempestive e spazialmente correlate.
Il processo che coinvolge l'agricoltura di precisione può essere riassunto in maniera sintetica con il diagramma ciclico riportato in figura 1.1.

La circolarità del processo evidenzia non a caso la continuità e l'importanza che riveste questa metodologia di gestione all'interno del processo di impresa e dunque la sua integrazione in tutte le fasi del processo produttivo aziendale dentro e fuori dal campo. Il processo inizia con la raccolta dei dati opportunamente rielaborati per poter fornire informazioni utili alla pianificazione delle attività di campo, dalla preparazione del suolo alla raccolta e si può basare inoltre anche sulle informazioni puntuali rilevate in corso d'opera e derivanti dal monitoraggio della coltura.

1.2.2 Tecnologie chiave

L'attività di *precision farming* necessita di tecnologie informatiche ed elettroniche di base per poter essere applicata in maniera corretta. Tra queste possono essere individuate alcune fondamentali:

• sistemi di posizionamento ad alta precisione *(GPS),* tecnologia chiave per raggiungere la precisione necessaria durante le operazioni sul campo. Fornisce il posizionamento statico e dinamico della macchina all'interno del sistema di riferimento utilizzato. Questi sistemi individuano efficacemente e registrano la posizione della macchina all'interno del campo utilizzando le coordinate geografiche (latitudine e longitudine) fornendo in questo modo anche la possibilità di associare ad un dato rilevato l'informazione della corrispettiva posizione. L'accuratezza del posizionamento così ottenuto può dipendere dal livello di complessità del sistema adottato e dalle condizioni al contorno;

• sistemi di guida automatica, in grado di svolgere specifici compiti di guida in maniera autonoma o con funzione di supporto all'operatore. Questa tecnologia riduce l'errore umano durante l'esecuzione delle traiettorie ottimali predefinite. Tra di questi possiamo distinguere:

- sistemi di guida assistita, utili a garantire una guida più accurata che evita le eccessive sovrapposizioni delle passate eseguite durante le lavorazioni ma sempre eseguita dall'operatore che deve comunque azionare lo sterzo manualmente, coadiuvato dalle indicazioni del sistema di assistenza;
- sistemi automatici di guida, sistema che prende il pieno controllo del volante permettendo al conducente di tenere maggiormente d'occhio l'operatrice senza penalizzare la precisione di guida;

• georeferenziazione, ovvero, tecnica utilizzata per produrre mappe tra cui per esempio quella del tipo di terreno, del livello di nutrienti o quella di resa che contengono informazioni utili alla gestione delle operazioni di campo e utilizzare questi dati per ottimizzare lo svolgimento delle operazioni e la regolazione delle macchine operatrici;

• sensori prossimali e telerilevamento permettono di raccogliere dati utili a valutare le condizioni del suolo e dello stato delle colture, umidità, grado di compattazione, stato nutrizionale della coltura, presenza di eventuali malattie.
L'accuratezza dei dati dipende dalla scala di misura che deve essere utilizzata e dal tipo di sensore utilizzato. Tali caratteristiche dovranno essere concordi con lo scopo della misurazione.

Si può distinguere, per sommi capi, a seconda delle distanza che esiste tra il sensore e l'informazione da raccogliere, due metodologie principali:

- si parla di *proximal sensing* quando vengono impiegati sensori utilizzati dall'operatore a breve distanza o montati direttamente a bordo della macchina. Questi sono in grado di raccogliere direttamente le informazioni e spesso vengono utilizzati anche per agire come *feedback* utile a compiere regolazioni della macchina operatrice sulla base delle informazioni raccolte;
- si intende telerilevamento quando il sensore si trova a una grande distanza dall'informazione da monitorare, spesso sono sensori che operano su satelliti o droni e sono in grado di fornire informazioni con una scala di dettaglio minore.

• la tecnologia a tasso variabile *(VRT)* conferisce la possibilità di modificare con precisione i parametri operativi di una macchina in maniera automatica a seconda della sua posizione in campo o sulla base delle informazioni derivanti da sensori a bordo o memorizzati in mappe di prescrizione. Dunque questa tecnologia permette di applicare dosaggi differenti di *input* e spesso è utilizzata con sementi o fertilizzanti.

• connettività e compatibilità tra le diverse componenti hardware e software del sistema.

1.2.3 Possibili effetti sulle politiche

Per ciò che attiene alle sfide per il futuro, la *precision farming* può fornire in alcuni casi possibili soluzioni utili ai *policies makers* per perseguire gli obiettivi delle loro politiche e fronteggiare le sfide legate alla gestione del comparto agricolo. E' possibile individuare alcuni ambiti principali in cui questa metodologia può aiutare le politiche comunitarie e non solo, a titolo di esempio si può citare:

- l'aumento della competitività del settore agricolo europeo. Le aziende agricole applicando le metodologie dell'agricoltura di precisione producono "di più con meno", aumentando così la competitività delle singole aziende e di conseguenza dell'intera filiera agroalimentare. Le aziende con dimensioni medio-grandi beneficerebbero di più di questa tecnologia consolidando il processo di aumento delle dimensioni aziendali medie in atto;
- la diminuzione della manodopera impiegata nel settore agricolo, specie dove quest'ultima ha un basso livello di specializzazione e istruzione;
- un aumento del fabbisogno di manodopera altamente specializzata in *Information and Communications Technology (ICT)* e un parallelo aumento del *business* della

fornitura di servizi all'industria legati a sensoristica, *IoT* (*Internet of Things*) e dell'industria della meccanizzazione;

- l'incremento della *food safety* e della *food security* in forza dei sistemi di monitoraggio e supporto alle decisioni, resi possibili grazie alla disponibilità e tempestività delle informazioni ottenute con i sistemi di *precision farming*;
- un maggior grado di trasparenza e tracciabilità della filiera agro-alimentare;
- un miglioramento della sostenibilità delle produzioni agro-zootecniche grazie alla riduzione degli *input* e all'aumento di conseguenza dell'efficienza del sistema, inclusi i contributi alla mitigazione degli impatti climatici del settore.

Per quanto riguarda la mitigazione degli impatti ambientali, tra i diversi effetti viene individuata principalmente:

- la riduzione degli sprechi di acqua mediante l'irrigazione di precisione;
- la riduzione delle dosi di fertilizzanti applicate in virtù della migliore efficienza di utilizzo, grazie alla possibilità di usare le tecnologie a tasso variabile ed evitando sovradosaggi;
- la possibilità di effettuare trattamenti antiparassitari mirati e di gestire precocemente eventuali focolai di infezioni;
- la possibilità di verificare le condizioni e pianificare gli interventi in campo in finestre di tempo che migliorino la tempestività, riducendo in questo modo fenomeni come il compattamento e i consumi energetici associati alle operazioni.

1.3 Applicazioni robotiche come agricoltura a elevatissima precisione

L'agricoltura di precisione, prevede l'utilizzo della tecnologia per consentire alle macchine impiegate in agricoltura di operare in maniera differenziata all'interno dello spazio di lavoro compiendo in maniera automatizzata regolazioni sito specifiche che consentono di personalizzare l'intervento secondo una scala di dettaglio che identificabile con la capacità di lavoro della singola unità regolabile. Ad esempio se si prende in considerazione un'irroratrice dotata di diversi ugelli regolabili automaticamente, la scala dell'intervento di precisione sarà definita dalla larghezza di lavoro di quest'ultimo, dalla velocità con cui questo può variare la sua portata e dalla velocità di avanzamento e la sua capacità di compiere lavori di precisione sarà dunque maggiore di uno spandiconcime con *variable rate technology* ma comunque nell'ordine di qualche metro quadrato.

Secondo De Clercq, (De Clercq, 2018) dal concetto di agricoltura di precisione è necessario passare al concetto di agricoltura a elevatissima precisione e Agricoltura 4.0 che prevede l'incorporazione di tecnologie e applicazioni intersettoriali.
In ottica di un'agricoltura 4.0, la vera svolta riguarda le sfide nel campo della sensoristica, viene richiesto di semplificare ciò che adesso sembra essere complicato, fornendo strumenti "*smart*" (*multi-causal decision-making systems*), ovvero capaci non solo di raccogliere dati in maniera corretta e automatizzata, ma anche che questi sensori abbiano la possibilità di prendere decisioni sugli interventi da compiere, garantendo un accorciamento dei tempi di raccolta ed elaborazione dei dati in *post processing*.
Lo scopo di questa modalità di analisi è quello di combinare i dati organizzati su diversi livelli informativi in modo da poter derivare nuova conoscenza e quindi possedere un maggior numero di elementi utili per prendere decisioni corrette in maniera automatica e istantanea attingendo anche da diverse fonti. L'obiettivo di questi sistemi complessi deve essere inoltre la facilità di utilizzo, la comprensibilità, la trasparenza dei concetti e la possibilità di supportare meglio l'esperienza dell'agricoltore (Weltzien, 2016).
L'obiettivo dei ricercatori è quello di sviluppare modelli dei processi produttivi agricoli capaci di andare incontro alle specifiche condizioni che caratterizzano i climi e le colture oltre che alle condizioni mutevoli del meteo e, parallelamente, razionalizzare gli *input* in modo da aumentare non solo la quantità ma anche la qualità delle produzioni (Weltzien, 2016).
Per ottenere modelli caratterizzati da un tale livello di complessità, è necessario avere a disposizione dati caratterizzati da elevatissima risoluzione spaziale e temporale, cosa che

con i moderni metodi di monitoraggio remoto non può avvenire, per poter analizzare simultaneamente numerosi aspetti che coinvolgono la coltura durante tutto il suo sviluppo.

In questo sviluppo avranno ruolo chiave alcune tecnologie e metodologie come:

- Le tecnologie *IoT*, che permettono di raccogliere e condividere anche a distanza dati e compiere correlazioni tra questi. Piattaforme come la *IBM Watson* applicano le metodologie del *machine learning* ai dati provenienti da sensori o droni trasformando il sistema di gestione in un sistema gestito da intelligenza artificiale. E' stato stimato che nel 2020 più di 75 milioni di apparecchi *IoT* saranno in uso nel settore agricolo e il volume dei dati raccolti e scambiati mediante connessione aumenterà notevolmente;
- L'automazione della forza lavoro, fenomeno già in atto e che dovrà essere potenziato a livello globale data la previsione che per il 2050 i due terzi della popolazione mondiale vivrà in aree urbane riducendo di conseguenza la forza lavoro disponibile nelle aree rurali.
  Le nuove tecnologie dovranno assicurare livelli di automazione più elevati consentendo la possibilità di esecuzione in remoto delle operazioni oltre che la capacità di svolgerlo in maniera automatizzata;
- Le abilità dell'agricoltore dovranno comprendere un *mix* di conoscenze tecnologiche e biologiche oltre a quelle puramente agronomiche;
- I dati avranno un ruolo centrale e dovranno indirizzare l'azione dell'azienda mediante l'analisi e la correlazione delle informazioni riguardanti: le condizioni meteo, le varietà utilizzate, il tipo di suolo, le analisi sulle probabilità di infezione, i dati storici, i trend di mercato e i prezzi.
  L'agricoltore sarà dunque in possesso di maggiori informazioni per prendere decisioni;
- L'utilizzo di *chatbot*, o assistenti virtuali, dotati di intelligenza artificiale e utilizzati già in altri settori, potrebbero assistere l'agricoltore rispondendo in riguardo ad alcune domande su specifici problemi;
- L'utilizzo dei droni per la raccolta dei dati riguardanti: il suolo, la *canopy*, lo stato nutrizionale e idrico delle piante.
  Alcune recenti *startup* utilizzano questi mezzi anche per l'esecuzione di operazioni come la semina e il trattamento con prodotti fitosanitari oltre che per valutare lo stato di salute della coltura e verificare la presenza di eventuali patologie.
  I droni, per esempio, garantiscono la possibilità di acquisire dati caratterizzati da risoluzioni spaziali e temporali elevate, consentendo di creare vere e proprie

animazioni che mostrano lo sviluppo della coltura e possono rivelare le inefficienze produttive consentendo possibili azioni correttive.

Per raggiungere l'obiettivo di elevare la precisione degli interventi non sono necessari però solo i dati e gli strumenti capaci di assicurare elevate risoluzioni spaziali, come per esempio avviene già con l'utilizzo dei droni, ma unire queste capacità con una pari accuratezza spaziale di intervento, portando la scala di intervento non più nell'ordine del metro quadro nella migliore delle ipotesi ma addirittura a livello della singola pianta o parte di essa.

Per sviluppare un sistema di controllo e gestione di questo tipo è necessario utilizzare applicazioni di tipo robotico. Un esempio di applicazione della robotica per fare agricoltura di altissima precisione capace di gestire in maniera automatica e connessa tutte le operazioni di gestione della coltura è rappresentato dallo strumento prodotto dalla *startup* californiana *FarmBot* che realizza e vende sistemi basati su una macchina a controllo numerico (*CNC*) in grado di svolgere in maniera automatizzata, o manualmente anche in remoto, tutte le operazioni di gestione di una o più colture definendone i singoli spazi occupati.

Limite di questo *robot* sta nel limitato spazio di lavoro gestibile che, nella sua versione più grande arriva a 2,9 m di larghezza, 5,9 m di lunghezza e un'altezza di 0,5 m, pari d un volume di poco superiore a 8,5 m^3 con un costo di circa 3˙500 € raggiungendo un prezzo per metro cubo di spazio di lavoro utile di circa 412 €/m^3.

Per l'applicazione in concreto dell'agricoltura di altissima precisione con costi ragionevoli è dunque necessario sperimentare sistemi robotici che siano in grado di garantire spazi di lavoro maggiormente elevati e costi contenuti.

A questo scopo potrebbe rivelarsi utile il campo della robotica dei manipolatori paralleli e più precisamente di una loro particolare sottofamiglia definita *cable driven parallel robot*.

1.4 I Cable Driven Parallel Robot (Cable robot)

Una tecnologia utilizzata nell'ambito dell'automazione che potrebbe essere utile in campo agricolo è quella che riguarda i manipolatori paralleli, ovvero macchine costituite in genere da due piattaforme di cui una fissa, detta base, ed una mobile denominata *end effector*. La base e l'*end effector* sono interconnessi da attuatori lineari, il cui numero di solito coincide con quello dei gradi di libertà conferiti all'*end effector* (Ottaviano, 2001).
In figura 1.2 è riportata, a titolo di esempio, l'immagine di un manipolatore parallelo dove si nota, in alto, la base fissa e, vincolato al di sotto di questa per mezzo di attuatori lineari, l'*end effector*.

Figura 1.2 Manipolatore parallelo

Un manipolatore di questo genere è dotato di grande accuratezza di posizionamento ma è in grado di sfruttare un'area di lavoro limitata.
Tra questa categoria di macchine rientrano anche i *Cable Driven Parallel Robot*, più semplicemente definiti anche "*cable robot*" o *robot* a cavi, che a differenza dei manipolatori paralleli classici, utilizzano cavi al posto degli attuatori lineari. Il funzionamento dei *cable robot*, dunque, si basa sul fatto che i cavi sono vincolati a una struttura *(frame)* che funge da supporto e infine questi ultimi vengono singolarmente vincolati ad un argano (*drum o winch*) azionato da un motore elettrico.
Il controllo del movimento dell'*end effector* all'interno dello spazio di lavoro *(work space)* è ottenuto variando opportunamente la lunghezza dei singoli cavi per mezzo dei motori elettrici che, compiendo rotazioni in senso orario o antiorario, permettono di variare la lunghezza dei cavi arrotolandoli o srotolandoli dai rispettivi *drum*. Spesso per questo motivo i cavi vengono definiti come "attivi" (figura 1.3) perché utili a determinare lo spostamento dell'*end effector* e si differenziano, per esempio, da quelli "passivi" (Figura 1.4) dove l'*effettore* è in grado di traslare lungo uno o più cavi tesi, sempre per mezzo di motori elettrici ma senza ottenere la variazione della lunghezza di questi.

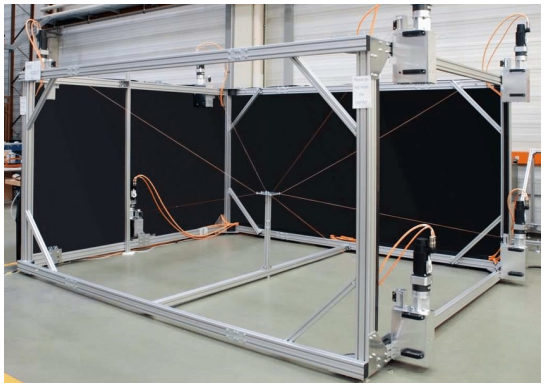

Figura 1.3 Cable robot con cavi attivi

Figura 1.4 Cable robot con cavi passivi

I *cable robot* sono dunque macchine costruttivamente abbastanza semplici e relativamente economiche capaci di operare, per quanto concerne lo sviluppo dimensionale dello spazio utile di lavoro e a seconda della tipologia di configurazione, a livello di piano (2 assi) o nello spazio (3 assi), conferendo diversi gradi di libertà di movimento all'*end effector* a seconda del numero e del tipo di arrangiamento dei cavi (Trevisani et al., 2006).

In generale un *cable robot*, a seconda del numero di gradi di libertà (*n*) e di quello dei cavi (*c)* che utilizza può essere classificato come un meccanismo (Ouyang, 2014):

- *Sotto-vincolato* se $c < n + 1$ (figura 1.5)
- *Completamente vincolato* con $c = n + 1$ (figura 1.6)
- *Vincolato con ridondanza* se $c > n + 1$ (figura 1.7)

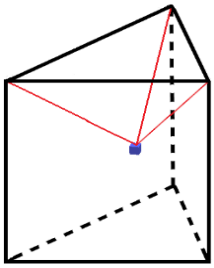

Figura 1.5 Cable robot sotto-vincolato

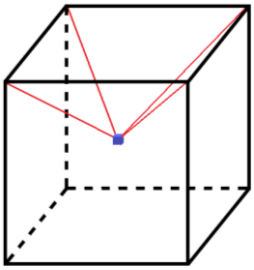

Figura 1.6 Cable robot completamente vincolato

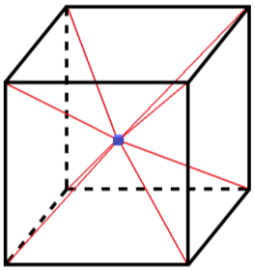

Figura 1.7 Cable robot vincolato con ridondanza

Queste tre grandi famiglie appena individuate, a loro volta, possono essere ulteriormente suddivise in sotto-famiglie a seconda dell'arrangiamento effettivo dei cavi utilizzato che può conferire diversi livelli di manovrabilità dell'*end effector* o diverse capacità di portata massima.

I diversi tipi di *setup* presentano peculiarità proprie che vanno approfondite in sede di progettazione. Sarà necessario quindi, per garantirne il corretto e razionale funzionamento, fare delle appropriate scelte progettuali preliminari coerenti con:

- i compiti che dovranno essere svolti dall'*end effector*;
- la conformazione e la dimensione del volume di lavoro;
- la velocità massima di traslazione dell'*end effector*;
- la portata massima necessaria;
- l'eventuale presenza di ostacoli all'interno del *workspace*.

L'analisi che segue, vuole evidenziare solo alcune delle caratteristiche salienti che riguardano le principali tipologie di arrangiamento dei cavi più utilizzate nella realizzazione di *robot* a cavi.

Per quanto riguarda i sistemi sotto-vincolati e completamente vincolati, si può affermare che essi abbiano il difetto di poter utilizzare unicamente la forza di gravità come fonte di tensione nei cavi e questa non sempre ne garantisce livelli adeguati, comportando una restrizione dello spazio utilizzabile rispetto a quello potenzialmente raggiungibile all'interno del *workspace*, mentre nei sistemi vincolati con ridondanza la tensione ottimale può essere opportunamente conferita in ogni posizione dall'azione dei cavi opposti o antagonisti.

Se da un lato i sistemi sotto e completamente vincolati presentano questa limitazione, dall'altro, forniscono la possibilità di ridurre i rischi di collisione tra i cavi e i possibili ostacoli che questi possono incontrare all'interno del volume di lavoro dato che l'*end effector*, essendo unicamente sospeso di fatto al di sotto di questi, trova completamente libero lo spazio di lavoro che si trova al di sotto (Wei, 2016).

Va ricordato che una tensione costante è indispensabile per poter determinare posizioni stabili e corrette oltre che per garantire una navigazione fluida dell'*end effector* all'interno del *workspace*.
Sostituendo le giunzioni rigide tipiche dei manipolatori paralleli (figura 1.2) con i cavi, molti dei limiti dei robot paralleli convenzionali vengono meno, ma al contempo sorgono altre complicazioni e sfide di tipo gestionale.
Il fatto di utilizzare i cavi fornisce la possibilità di sviluppare diverse applicazioni, altrimenti impossibili da realizzare con i classici manipolatori paralleli, come per esempio (Khosravi, 2014):

- realizzare robot caratterizzati da spazi di lavoro utile compatibili con la scala di lavoro dell'agricoltura;
- garantire la possibilità di raggiungere velocità di manipolazione maggiori;
- rendere possibile la movimentazione di carichi più elevati;
- fornire la possibilità di utilizzare queste macchine in ambienti pericolosi o difficilmente accessibili;
- conferire maggiore riconfigurabilità alla macchina e rendere possibile il trasporto e l'assemblaggio veloce e facile direttamente sul sito di lavoro.

Nel contempo l'utilizzo dei cavi in luogo degli attuatori lineari fa sorgere alcune complicazioni tipiche di tali macchine, per esempio, i cavi dato che possono solo esercitare forza per mezzo della tensione possono essere usati per tirare un oggetto e non per spingerlo, perciò per evitare errori di posizionamento o guasti, la corretta progettazione dell'algoritmo di controllo e gestione del sistema riveste un'importanza fondamentale. L'algoritmo deve garantire che tutti i cavi rimangano in tensione durante lo svolgimento

delle operazioni evitando le zone del *workspace* dove anche solo uno dei cavi non abbia un certo grado di tensione e fare inoltre in modo che la tensione non superi livelli definiti critici per la tenuta dei cavi o per evitare il manifestarsi di livelli di coppia resistente non sopportabili dal motore garantendo in questo modo il mantenimento della posizione voluta. Lo stesso discorso della coppia resistente vale anche per le velocità attese dai motori che diminuiscono all'aumentare della coppia resistente sviluppata a livello dei *drum*.

Da un'analisi della bibliografia in merito, emerge che molti autori hanno un'opinione concorde nell'evidenziare quali siano le criticità principali inerenti al tema indicando spesso (Trevisani, 2006):

- la determinazione del valore della tensione dei cavi;
- la difficoltà di predire con precisione tale tensione;
- la necessità di sviluppo di schemi di controllo che garantiscano tensioni positive;
- quali siano le migliori strategie di pianificazione delle traiettorie capaci di assicurare costantemente tensioni positive su tutti i cavi durante il movimento dell'*end effector*.

Una soluzione spesso adottata in letteratura è quella della ridondanza, che però rappresenta una configurazione non sempre applicabile per l'interferenza creata dai cavi aggiuntivi.

Un'altra criticità deriva dal fatto che, per semplicità, spesso i cavi vengono considerati come elementi privi di massa e perfettamente anelastici, approccio che viene utilizzato di frequente in molti studi e nella realizzazione di prototipi che, nonostante rappresentino diverse interpretazioni del tema specie in riguardo agli algoritmi di controllo e gestione delle configurazioni strutturali del sistema, in comune hanno proprio il fatto che compiano la suddetta semplificazione.
Nelle potenziali applicazioni ad elevata precisione e specialmente in quelle su larga scala i cavi, però, vanno necessariamente considerati come elementi elastici e soggetti all'influenza della propria massa, il che ne modifica il comportamento e complica non poco la realizzazione e il perfezionamento di modelli di gestione e controllo. Ad oggi la ricerca in questo campo è tutt'ora abbastanza limitata (Khosravi, 2014).

Per comprendere l'importanza che riveste la modellazione matematica corretta di questi elementi, in letteratura si possono trovare alcuni esempi dove viene evidenziato che un errata gestione può portare a (Duan, 2010):

- errori di posizionamento e orientamento dell'*end effector*;

- errori nella stima delle forze in gioco dovuti a errori nel calcolo della lunghezza dei cavi.
  In alcune posizioni all'interno del volume di lavoro un errore anche solo nell'ordine dello 0,1% sulla lunghezza totale può portare a ingenti errori di considerazione delle forze anche nell'ordine del 50% tra il calcolato o previsto e il valore effettivo.

Per rendere più chiaro il concetto appena espresso, nella figura 1.8 viene rappresentata un'ipotetica situazione in cui tre cavi (blu giallo e nero) di uguale lunghezza sono vincolati per un estremità a un punto di emanazione comune rappresentato in verde e per l'altra estremità a un ipotetico *end effector* dotato di massa che a sua volta è retto da un secondo cavo rappresentato in grigio.

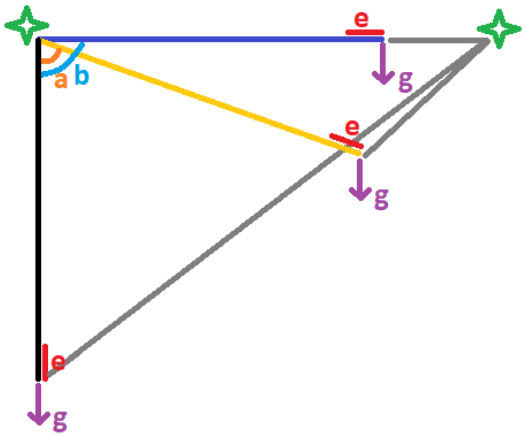

Figura 1.8 Effetto dell'errore di calcolo della lunghezza del cavo sul posizionamento e sulla tensione

Indipendentemente dalla massa di cui possa essere dotato l'*end effector*, il contributo di un errore di calcolo della lunghezza ottimale del cavo (nella figura per semplicità viene rappresentato in rosso un errore del 10% sulla lunghezza totale del cavo) varia a seconda dell'inclinazione che il cavo assume rispetto al vettore peso rappresentato in viola. Tale errore sarà un errore prevalentemente di posizionamento se l'angolo interno tra cavo e punto di emanazione sarà di 0° come nel caso del cavo nero, mentre nel caso in cui l'angolo misurasse ipoteticamente 90° come nel caso del cavo blu, tale errore, tenuto conto anche del cavo opposto, sarebbe praticamente da imputare unicamente al valore della tensione, che potrebbe raggiungere in questo caso livelli non sostenibili dalle componenti meccaniche utilizzate.
Nelle altre posizioni, il contributo all'errore sarà misto tra posizionamento e valutazione della tensione tra ideale ed effettiva.

Finora sebbene le applicazioni pratiche siano state abbastanza limitate, nel tempo sono stati sviluppati numerosi prototipi da parte di diversi centri di ricerca. Queste attività si sono intensificate negli ultimi anni a testimonianza dell'interesse crescente per il tema. Viene riportata una breve trattazione che illustra i principali centri di ricerca che nel tempo sono stati attivi e sulle loro realizzazioni salienti nel campo dei *cable robot*.

I primi prototipi operativi di robot a cavi sono stati sviluppati negli Stati Uniti e in Giappone, verso la fine degli anni ottanta del secolo scorso, periodo in cui fu sviluppato il *RoboCrane®* ad opera del *National Institute of Standards and Technology (NIST) (USA)*, presentato come il primo prototipo indicato per l'utilizzo su media-larga scala di

applicazione e utilizzato successivamente per esempio per la movimentazione di carichi (Pott, 2013).

Nel tempo sono state realizzate altre versioni sperimentali da parte di questo ente, come per esempio il *NIST Spider* con un telaio di 6 metri di lato che utilizzava *encoder*, servomotori e sensori per la rilevazione della tensione dei cavi per ottenere il posizionamento corretto o il *NIST Mini Tetra*, prototipo di dimensioni più contenute che utilizzava motori passo-passo al posto dei servomotori.

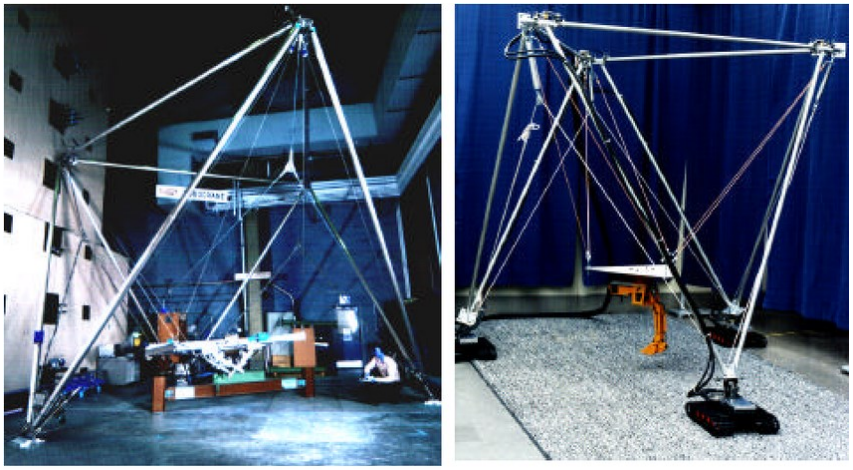

Figura 1.9 Realizzazioni del National Institute of Standards and Technology

Nei primi anni novanta viene realizzato in Giappone il *FALCON (FAst Load CONveyance)* di *Kawamura*, definito anche come *ultra high speed cable robot*. Questo fu progettato per applicazioni *pick-and-place* combinate a elevate velocità di spostamento dell'*end effector* (Mroz, 2004).

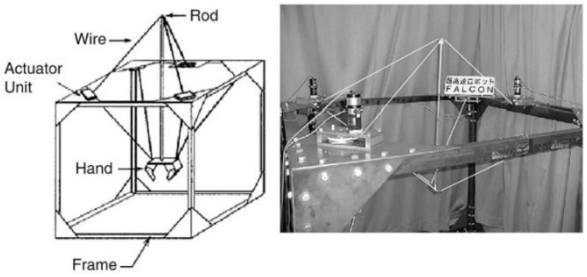

Figura 1.10 FALCON

Gli esempi riportati in precedenza rappresentano le prime prove di operabilità concreta e caratterizzata da buone performance in riguardo a velocità di esecuzione e precisione di posizionamento (Izard, 2013).
Successivamente verso la fine degli anni novanta *Tadokoro* (Tadokoro, 1999) sviluppa una macchina caratterizzata da facilità e velocità di assemblaggio, dotata di un algoritmo di controllo che ne permetteva l'ancoraggio a elementi strutturali costituiti o del paesaggio, riducendo al minimo indispensabile la parte di *frame* e garantendo buona operabilità anche in ambiente *outdoor*. Lo scopo di quest'ultimo progetto è quello di essere facilmente utilizzabile durante operazioni di soccorso o a seguito di eventi catastrofici come per esempio dopo i terremoti.

Nei primi anni duemila la ricerca nel campo dei *cable robot* si amplia e si approfondisce.

Viene realizzato in questo periodo in Germania il prototipo *SEGESTA* (figura 1.11) da parte dell'università di *Duisburg-Essen* da utilizzare come sistema di ricerca e studio sulla cinematica, sui sistemi di controllo e sui parametri utili alla progettazione e all'ottimizzazione del funzionamento.

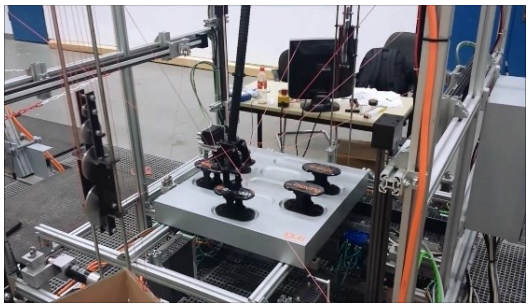

Figura 1.11 SEGESTA

Parallelamente sempre in Germania fu anche realizzato da parte dell'*University of Rostock* il robot *RoboCable V* a scala di prototipo per compiere operazioni di manipolazione (Pott, 2013).

Anche il *Fraunhofer IPK* di Berlino (*Institute for Production Systems and Design Technology*) si interessa in questo periodo al tema dei *cable robot* sviluppando un modello di robot chiamato *StringMan*, riportato in figura 1.12, utilizzato nella riabilitazione alla deambulazione delle persone con infortuni. Il fine di questo strumento è quello di garantire maggiore sicurezza per la persona e un accurato controllo delle forze generate durante il procedimento riabilitativo (Dragoljub, 2004).

Figura 1.12 StringMan

In Francia l'*INRIA* (*Institut national de recherche en informatique et en automatique*) viene sviluppata, invece, un'intera famiglia di robot chiamata *Marionet*, essa include: un prototipo di dimensioni ridotte caratterizzato da elevate velocità di esecuzione destinato a operazioni di *pick-and-place* e per la riabilitazione, denominato *Marionet Rehab*; una gru portatile da utilizzare in operazioni di soccorso (*M. Crane*); un sistema per l'assistenza casalinga ad anziani e portatori di handicap (*M. Assist*) e infine un robot da utilizzare per scopi ludici integrato in sistemi di simulazione per la realtà virtuale (*M. VR*) (Pott, 2013).

*Figura 1.13 Realizzazioni dell'*INRIA

Anche in Svizzera all'*ETH* (Istituto Federale di Tecnologia) di Zurigo è stato realizzato un simulatore di movimento da integrare in sistemi di realtà virtuale destinato sempre a scopi di tipo ludico.

In Cina, invece, recentemente è stato realizzato il più grande robot a cavi esistente utilizzato per la gestione del posizionamento del riflettore utilizzato dal radiotelescopio più grande al mondo dotato di un'apertura di ben 500 metri denominato *FAST* (Five hundred Spherical Telescope figura 1.14) (Pott,2013).

Figura 1.14 Telescopio FAST

In Iran anche il *KNTU* (Pott, 2013) si è interessato alla tematica valutando l'utilizzo di questa tecnologia in campo medico specie per applicazioni inerenti alla chirurgia.

Dal 2006 all'istituto di tecnologia della produzione e automazione *IPA* di *Fraunhofer* sono stati sviluppati una famiglia di robot chiamati *IPAnema* con lo scopo di essere utilizzati principalmente in operazioni di ispezione, assemblaggio e manipolazione su media-larga scala (Pott, 2013).

Figura 1.15 Realizzazione dell'IPA

L'applicazione più celebre e che per prima ha trovato un'applicazione commerciale portando alla conoscenza del grande pubblico di questa tecnologia è rappresentata dalla *SkyCam* (figura 1.16), *cable robot* utilizzato per la realizzazione di riprese video di eventi sportivi, concerti e altre manifestazioni (Wei, 2016).

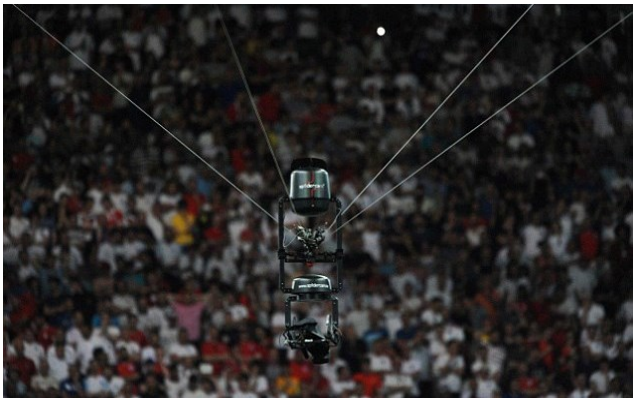

Figura 1.16 SkyCam

Questi esempi riportati in precedenza sono solo alcuni dei prototipi realizzati, quelli che possono essere definiti più significativi e che, insieme a tutti gli altri prototipi e ai modelli di calcolo anche solo simulativi reperibili in letteratura, riescono a testimoniare il crescente livello di interesse e studio per il tema.

Per quanto riguarda il campo agricolo l'utilizzo di questa tecnologia è tutt'ora marginale e si ha notizia di prime applicazioni nel 2017, anno in cui sono stati sviluppati parallelamente: il sistema *Tarzan* dall'*University of Georgia*, *cable robot* di tipo passivo visto in precedenza nella figura 1.4, capace di traslare lungo un cavo teso, pensato per essere destinato al monitoraggio delle colture e due impianti più complessi ad opera rispettivamente dell'*University of Nebraska*, che ha realizzato un *end effector* da applicare a un *cable robot* utilizzato per svolgere operazioni di *crop sensing* mediante l'utilizzo combinato di spettrometri, sensori *IR*, *NIR*, *VIS* e *LIDAR*.

In maniera simile l'*ETH* ha realizzato un *cable robot* capace di coprire un area utile di lavoro di 1 ha dislocando un carico di 12 Kg (Kirchgessner, 2017).
L'*end effector* è stato equipaggiato anche in questo caso con un insieme di sensori in grado di misurare diversi fenomeni fisici tra cui: temperatura, umidità, altezza, architettura della *canopy* e di sensori ottici per la raccolta di immagini multispettrali. Lo scopo di queste installazioni è ottenere dati molto utili ai *breeders* e ai modellisti di sistemi predittivi.
Questo è reso possibile dal fatto che tale tecnologia consente di garantire misurazioni anche in condizioni meteorologiche e di transitabilità avverse garantendo dunque continuità alla campagna di *crop sensing*, dall'emergenza fino al raccolto e conferendo livelli di risoluzione spaziale e temporale molto elevati.
In campo agricolo tale tecnologia, dunque, ha cominciato a trovare spazio in applicazioni sperimentali volte al monitoraggio colturale a scala di campo e alla raccolta di numerosi dati utili alla calibrazione di modelli predittivi ma potrebbe trovare in futuro diverse applicazioni utili ad altre necessità tipiche del settore agricolo come ad esempio l'ambito dell'agricoltura di precisione e la possibilità di valutare l'utilizzo dei *robot* a cavi anche per lo svolgimento di alcune delle operazioni di campo definibili come "leggere".

Figura 1.17 Cable robot in agricoltura: a sinistra compare l'impianto dell'University of Nebraska, a destra quella dell'ETH di Zurigo

2 SCOPO DELLA TESI

Lo scopo della tesi è quello di realizzare un prototipo di *cable robot* sfruttando il più possibile gli strumenti tecnologici offerti da piattaforme *open source* come Arduino e utilizzando per buona parte conoscenze fisico-matematiche e informatiche apprese durante l'intero corso di laurea e in particolare nel corso di *farm automation*.
La realizzazione del prototipo ha lo scopo di indagare sulle principali criticità legate alla gestione della meccanica, dell'elettronica e di tutte quelle conoscenze necessarie alla gestione ottimale e automatizzata del moto dell'*end effector* all'interno del volume di lavoro con lo scopo di rendere possibile lo sviluppo futuro di piattaforme dotate di sensori e attuatori da utilizzare in operazioni di monitoraggio e di intervento sulla coltura, oltre che di prendere confidenza con gli strumenti tecnici e matematici da riutilizzare anche nella realizzazione di altri progetti.

3 MATERIALI E METODI

Per la realizzazione del prototipo sono stati necessari diversi materiali, caratterizzabili principalmente come *hardware* e *software*. Tali strumenti sono stati selezionati vagliando le possibilità fornite dal mercato e dimensionati per mezzo di simulazioni e calcoli preliminari in modo da poter essere in grado di andare incontro alle esigenze di funzionamento dettate dalla scala del progetto.
Per questi motivi ha rivestito un ruolo importante la verifica della compatibilità degli elementi, sia *hardware* che *software*, mediante la realizzazione di diversi *test* e prototipi a scala ridotta.

3.1 Geometria e struttura del prototipo

Per definire un'architettura generale del prototipo è necessario individuare in via preliminare il volume e la geometria del potenziale volume di lavoro.
Si è pensato di utilizzare il prototipo in serra e per questo motivo le dimensioni di massima sono state scelte per essere compatibili con le vasche già presenti e con l'ingombro di altri elementi e impianti già esistenti come per esempio l'impianto di illuminazione.
Determinate le dimensioni di massima, è stato possibile individuare le altre caratteristiche progettuali vincolanti come la velocità di manipolazione desiderata e il livello di carico utile trasportabile dall'*end effector*.
Definiti questi elementi chiave, sono state effettuate alcune simulazioni necessarie a determinare l'entità dei fenomeni fisici in gioco come per esempio la coppia motrice necessaria e la velocità di rotazione di cui debbano essere dotati i motori e di conseguenza le caratteristiche del cavo da utilizzare per rispettare i vincoli di progetto.

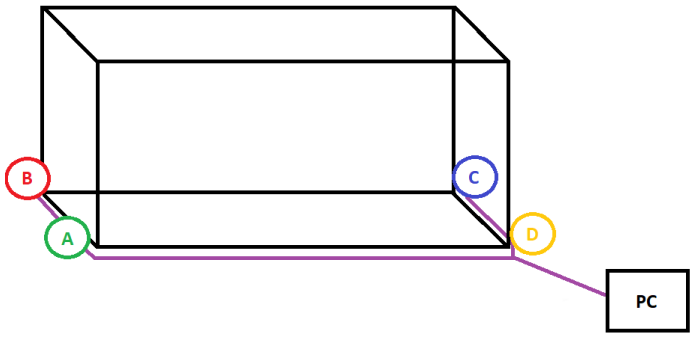

Figura 3.1 Schema semplificato del progetto

In sede di progettazione sono state determinate anche quali fossero le variabili da tenere sotto controllo per gestire il funzionamento e quali fossero i sensori esistenti in grado di monitorarli.
In generale sono necessari 4 motori, uno per ogni cavo da gestire e di altrettanti *encoders* utili a monitorare la posizione angolare dei motori oltre che di un protocollo efficace di comunicazione in grado di trasmettere con adeguata velocità e a sufficiente distanza le informazioni dagli attuatori all'unità di calcolo e viceversa, presupposti necessari al corretto funzionamento.
L'architettura del sistema prevede dunque, come illustrato in figura 3.1, l'utilizzo di quattro unità motore denominate rispettivamente *A*, *B*, *C* e *D*. Ogni unità motore è composta da un motoriduttore, un *encoder* e da tutti i componenti necessari a gestire il loro funzionamento. Le singole unità motrici sono interconnesse fisicamente tra loro e all'unità di calcolo per mezzo di un sistema di comunicazione in grado di garantire l'interscambio dei dati necessari alla gestione del funzionamento.

3.2 Principio di funzionamento generale

Il funzionamento di un *cable robot* prevede che l'*end effector* sia in grado di traslare liberamente all'interno del *workspace*. Per rendere ciò possibile è necessario variare opportunamente la lunghezza dei cavi a cui l'*end effector* è vincolato, adeguando tale dato il più accuratamente e tempestivamente possibile, in modo da consentirgli di seguire traiettorie articolate e in maniera fluida.

I calcoli inerenti al dato della lunghezza ottimale dei cavi sono funzione della posizione occupata dall'*end effector* e al variare di essa in funzione del tempo. Le caratteristiche fondamentali per garantire il funzionamento sono dunque:

- il passo di calcolo con cui quest'ultimo dato viene ricalcolato che è funzione della velocità di calcolo e allo scopo è stato scelto di fare compiere tale operazione al *PC* anche in ottica di facilitare l'analisi dei dati raccolti durante il funzionamento del prototipo e di eventuali funzioni implementabili in futuro;
- la distanza e la velocità con cui questo dato riesce ad essere inviato alle unità motrici che è funzione del protocollo di comunicazione e la robustezza di quest'ultimo. Per questo motivo dopo diverse prove e valutazioni si è deciso di utilizzare il *CAN BUS*.

Duque se i dati vengono elaborati e inviati con sufficiente velocità le unità motrici sono in grado di conseguenza di gestire in parallelo e dunque con maggiore prontezza il proprio

cavo consentendo all'*end effector* di seguire le traiettorie imposte.
A questo scopo le unità motrici sono programmate e gestite da microcontrollori in modo da seguire in maniera autonoma le istruzioni a loro impartite attuando un controllo di tipo distribuito con prestazioni più elevate rispetto al caso in cui i motori fossero gestiti da un'unica unità centrale.
Questa architettura offre anche un elevato grado di riconfigurabilità perché consente di aggiungere o eliminare singole unità e di cambiare la loro posizione nello spazio senza effettuare grandi modifiche a livello di *software*.

Infine è stata aggiunta un'unità definita come *master*, dotata di dispositivi di *input* e *output* utile a mettere in connessione il *CAN BUS* con il *PC* e a permettere la gestione delle diverse modalità di funzionamento e calibrazione oltre che consentire l'utilizzo in modalità manuale.
Questa scelta è stata fatta per evitare di utilizzare direttamente un'interfaccia *PC-CAN* che sarebbe stata meno economica e che avrebbe richiesto in ogni caso un unità *master* per la gestione degli *input*.
Per quanto riguarda la parte *software* è stato utilizzato *MATLAB* per la gestione dell'algoritmo di controllo su *PC* mentre tutte le unità, motrici e *master*, sono state programmate mediante il linguaggio utilizzato dall'ambiente di sviluppo di Arduino.
Lo schema del progetto di massima in figura 3.2 illustra complessivamente gli elementi principali e individua le principali linee di comunicazione e alimentazione necessarie.

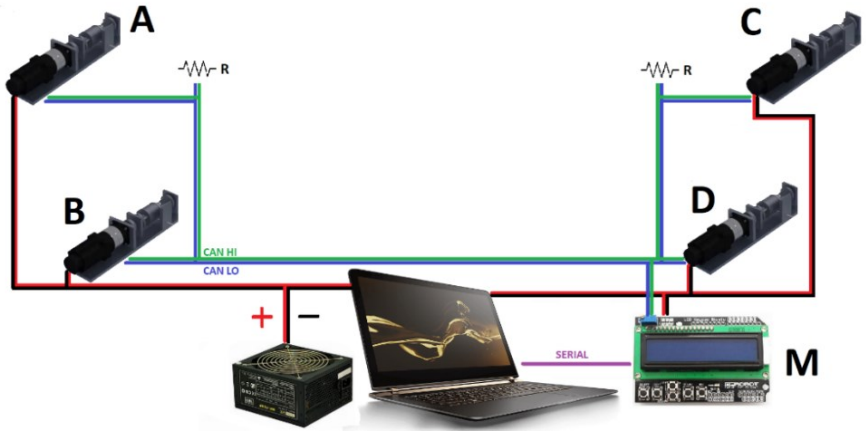

Figura 3.2 Schema del progetto di massima

3.3 Dinamica dell'*end effector*

L'*end effector* è il terminale in grado di muoversi all'interno dello spazio di lavoro (figura 3.3) è quindi necessario, ai fini della realizzazione del prototipo, approfondire il concetto di cinematica o geometria del movimento.
Per trattare tale argomento occorre effettuare la semplificazione di ritenere il corpo in movimento come puntiforme, cioè un oggetto la cui estensione non ha alcun interesse tanto da poter essere rappresentato da un punto materiale.

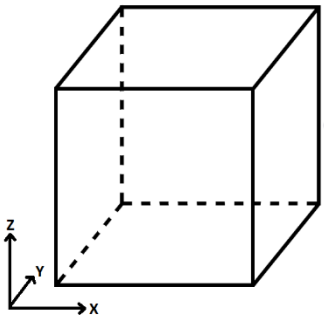

Figura 3.3 Workspace

La cinematica del punto si può definire come geometria dello spazio vettoriale quadridimensionale formato dalle tre coordinate spaziali (X, Y, Z) e dalla coordinata temporale (t).

3.3.1 Il punto

L'*end effector* sarà opportunamente vincolato tramite cavi alla struttura di sostegno che contiene il volume di lavoro e la lunghezza di questi determinerà il suo posizionamento.
Il dato della lunghezza del singolo cavo$_{(i)}$ viene considerato come la distanza che esiste tra il suddetto punto e il punto di emanazione del cavo$_{(i)}$ definito come *exit point* (*Ep*$_{(i)}$).

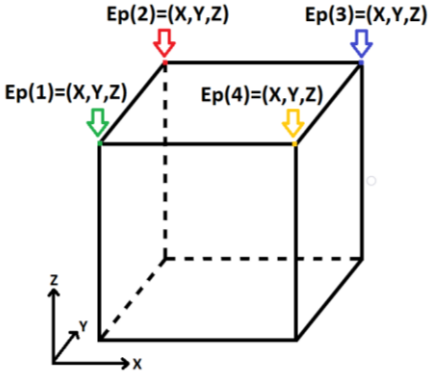

Figura 3.4 Exit point

Per punto di emanazione si intende il punto in cui il cavo si inserisce all'interno del *workspace*. Questo dato è molto importante perché le sue coordinate spaziali all'interno del sistema di riferimento cartesiano impostato in precedenza saranno necessarie per i calcoli che verranno effettuati successivamente. Nella figura 3.4 viene rappresentato quanto appena esposto.

3.3.2 Distanza

Conoscendo le coordinate di tutti gli *exit point* si può calcolare la distanza che esiste tra l'*Ep*$_{(i)}$ e l'*end effector* (o il punto di ancoraggio del cavo all'*end effector* se i cavi non convergono tutti nello stesso punto).
Queste distanze rappresentano, in prima approssimazione, le lunghezze ottimali dei cavi,

necessarie per ottenere il posizionamento voluto. Questa lunghezza può essere calcolata considerando il singolo punto di ancoraggio del $cavo_{(i)}$ come il centro di una sfera e tale distanza il suo raggio. In geometria cartesiana la sfera infatti è definita come l'insieme dei punti equidistanti dal centro e la sua equazione prende appunto in considerazione le coordinate del suo centro (indicate con $pedice_0$).
Con riferimento alla figura 3.5, e secondo l'equazione 3.1 si ricava che la lunghezza dei cavi è idealmente:

$$r^2 = \left(X - X_{(0)}\right)^2 + \left(Y - Y_{(0)}\right)^2 + \left(Z - Z_{(0)}\right)^2 \quad [m] \quad [Eq.3.1]$$

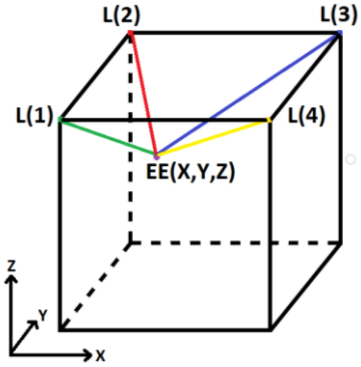

Figura 3.5 Distanza

$$L_{(1)} = \sqrt[2]{\mathrm{X}^2 + \mathrm{Y}^2 + \left(\mathrm{Z\ Ep}_{(1)} - \mathrm{Z}\right)^2} \quad [m] \quad [Eq.3.2\ a]$$

$$L_{(2)} = \sqrt[2]{\mathrm{X}^2 + \left(\mathrm{Y\ Ep}_{(2)} - \mathrm{Y}\right)^2 + \left(\mathrm{Z\ Ep}_{(2)} - \mathrm{Z}\right)^2} \quad [m] \quad [Eq.3.2\ b]$$

$$L_{(3)} = \sqrt[2]{\left(\mathrm{X\ Ep}_{(3)} - \mathrm{X}\right)^2 + \left(\mathrm{Y\ Ep}_{(3)} - \mathrm{Y}\right)^2 + \left(\mathrm{Z\ Ep}_{(3)} - \mathrm{Z}\right)^2} \quad [m] \quad [Eq.3.2\ c]$$

$$L_{(4)} = \sqrt[2]{\left(\mathrm{X\ Ep}_{(4)} - \mathrm{X}\right)^2 + \mathrm{Y}^2 + \left(\mathrm{Zep}_{(4)} - \mathrm{Z}\right)^2} \quad [m] \quad [Eq.3.2\ d]$$

Questo dato rappresenta un'approssimazione iniziale, il cavo in realtà è soggetto, infatti:

- alle tensioni provocate dal peso dell'*end effector*, che possono determinarne l'allungamento dovuto alla sua elasticità, che dipende dalle caratteristiche tecnologiche del cavo utilizzato (Riehl, 2010);
- al suo stesso peso che, in funzione anche del carico di tensione applicato, possono determinarne una traiettoria che non è perfettamente rettilinea come viene considerato nel calcolo della distanza (Bin, 2008).
  La funzione che meglio approssima tale forma effettiva assunta dal cavo prende il nome di catenaria, particolare funzione che somiglia a una parabola di cui si parlerà in maniera maggiormente approfondita nel paragrafo 3.5.

Per i motivi sopra citati potrebbe essere necessaria la correzione di tale lunghezza approssimata, mediante l'utilizzo del metodo ipotizzato nel paragrafo 3.6.

3.3.3 *Moto del punto nello spazio*

Dopo aver calcolato le lunghezze ideali dei cavi in situazione statica (*end effector* fermo) rimane il problema di correlare tali valori alla variabile tempo (t).
Supponendo che l'*end effector* non sia fermo sempre nello stesso punto, ma che la sua posizione cambi nel tempo, andrà calcolata la variabile lunghezza del cavo ogni volta che la posizione del terminale cambia e sarà importante anche determinare la velocità con cui questa misura cambia.
Per fare ciò occorre, innanzitutto, definire il campo di movimento dell'*end effector*, ovvero trovare il dominio dei delta (Δ) ammissibili. Per Δ ammissibili si intendono quelli che legano le coordinate iniziali (situazione di partenza t_1) e finali (situazione di arrivo t_2) che l'*end effector* può raggiungere.
Lo scopo di questa fase è di determinare, sotto forma di vettore, la direzione del moto utilizzando le coordinate polari (angolo α e β) e l'intensità del vettore per determinare la velocità di moto del punto. Supponendo per semplicità intervalli di Δt costanti l'entità dello spostamento sarà funzione solo della velocità.

$$\Delta S = V \times \Delta t \quad [m] \quad [Eq. 3.3]$$

Tale concetto viene illustrato in figura 3.6, ponendo in evidenza i due angoli di navigazione, α (zenitale) raffigurato in rosso, β (azimutale) in verde e in viola l'intensità del vettore spostamento.

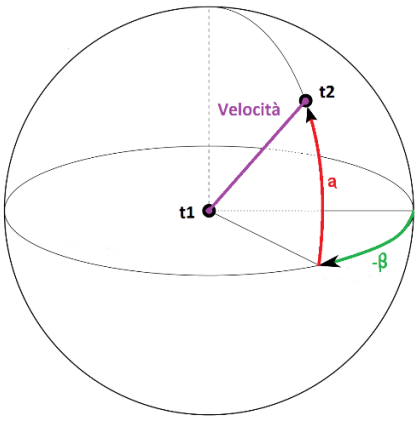

Figura 3.6 Direttrici del moto

Considerando il caso del moto uniformemente accelerato, l'entità del vettore spostamento potrà cambiare a seconda del caso in cui il punto sia sottoposto ad accelerazioni positive o negative nel modo in cui viene riportato sotto.

$$\Delta S = (V_{iniziale} + (acc \times \Delta t)) \times \Delta t \quad [m] \quad [Eq. 3.4]$$

Dopo aver introdotto lo spostamento del punto nello spazio e la velocità, resta da definire i Δ da attribuire alle coordinate spaziali tra l'istante t_1 e t_2 che corrispondono allo spostamento effettuato. Per il fatto che i dati noti sono rappresentati dalle coordinate possedute dall'*end effector* nell'istante iniziale (in figura 3.6 indicato con t_1) e la direzione, il verso e l'intensità del vettore spostamento, si possono calcolare le coordinate della posizione finale possedute dal punto in t_2 nel seguente modo:

$$\Delta X = \Delta S \times \cos(\alpha) \times \cos(\beta) \quad [m] \quad [Eq.\ 3.5\ a]$$

$$\Delta Y = \Delta S \times \cos(\alpha) \times \sin(\beta) \quad [m] \quad [Eq.\ 3.5\ b]$$

$$\Delta Z = \Delta S \times \sin(\beta) \quad [m] \quad [Eq.\ 3.5\ c]$$

Il movimento del punto nello spazio è vincolato all'opportuna e tempestiva modifica della lunghezza dei cavi, è necessario perciò monitorare con la maggiore frequenza possibile la lunghezza effettiva del cavo per mezzo di un *encoder* e quella ottimale (o *target*) calcolato dall'algoritmo di controllo in modo da ottenere il dato dell'errore di posizionamento.

3.3.4 Feedback di posizionamento

Il sistema di posizionamento ha bisogno di monitorare lo spostamento angolare (Δω) che deve compiere il motore per allungare o accorciare il cavo nel modo corretto.
Questa variabile sarà tenuta sotto controllo per mezzo di un *encoder*. E' necessario tradurre la logica lineare considerata fino ad ora con la logica basata sugli *step* (o impulsi di *encoder* equivalenti), ovvero porzioni di circonferenza (*rad*) rilevabili dall'*encoder*. Tali porzioni di circonferenza dipendono dalla dimensione del *drum* su cui viene arrotolato il cavo e dalla risoluzione fornita dall'*encoder* utilizzato. Maggiore sarà quest'ultima, più elevata sarà l'accuratezza potenzialmente raggiungibile dal posizionamento e allo stesso modo tale accuratezza aumenterà al diminuire delle dimensioni del *drum* a parità di prestazioni dell'*encoder*.
Per questo motivo in fase di progettazione bisogna valutare accuratamente le necessità di accuratezza del posizionamento e scegliere *encoder* in grado di fornire una risoluzione compatibile con le dimensioni adottate per il *drum* e con le esigenze di progetto.

$$Risoluzione\ encoder = \frac{numero\ step}{rotazione} \quad [Step/rotazione] \quad [Eq.3.6]$$

$$\text{Step°} = \frac{360°}{Risoluzione\ encoder} \quad [°/step] \quad [Eq.3.7]$$

$$\Delta\text{rad}_{Step} = \frac{2\pi \times \text{Step°}}{360°} \quad [rad/step] \quad [Eq.3.8]$$

$$r_{\ drum} = \frac{\emptyset_{drum}}{2} \quad [m] \quad [Eq.3.9]$$

$$\Delta L_{\ Step} = 2 \times r_{drum} \times \Delta\text{rad}\ Step \quad [m/step] \quad [Eq.3.10]$$

3.3.5 Target

Dopo avere introdotto il *feedback* di posizionamento non rimane che introdurre il concetto di *target*. Esso corrisponde al valore espresso in *step* della lunghezza del cavo che assicura il posizionamento corretto.
In questo modo il sistema sarà in grado di calcolare il posizionamento voluto o ottimale e confrontarlo con il posizionamento attuale.
Il posizionamento attuale viene fornito dalla lettura del conteggio raggiunto del contatore pilotato dall'*encoder*, accertandosi che in fase di calibrazione, il punto di emanazione del cavo in questione risulti pari a un conteggio dell'*encoder* di zero.
Sarà così possibile calcolare l'errore di posizionamento come la differenza tra il conteggio atteso e quello reale acquisito con la lettura del valore fornito dal contatore. Se l'errore sarà minore di 0 il motore dovrà girare in un senso e se sarà maggiore nell'altro, cercando di portare il sistema all'equilibrio ovvero verso una situazione di errore nullo.

$$Target_{(i)} = \frac{L_{(i)ottimale}}{\Delta L_{step}} \quad [step] \quad [Eq.3.11]$$

$$\varepsilon_{(i)} = Target_{(i)} - Conteggio\ encoder_{(i)} \quad [step] \quad [Eq.3.12]$$

3.4 Dinamica dei cavi

I cavi utilizzati per ottenere il posizionamento dell'*end effector* hanno caratteristiche tecniche proprie che dipendono in buona parte dal tipo di materiale di cui essi sono costituiti e dal tipo di fattura utilizzata nel caso dei cavi in treccia.

Se vengono utilizzati cavi non considerabili come inestensibili, l'accuratezza del posizionamento dipenderà non solo dalla risoluzione dell'*encoder* e dalle dimensioni del *drum* ma anche dalla capacità di predire con sufficiente certezza le dinamiche elastiche che coinvolgono i cavi durante il lavoro. Risulta per questo importante valutare il comportamento di tali elementi sottoposti ai carichi di tensione a cui saranno soggetti durante il lavoro con lo scopo di valutare l'eventuale utilizzo di un meccanismo di correzione per contrastare il fenomeno.

La tensione è un fenomeno fisico rappresentabile come una forza che tende a fare comportare il cavo come una molla, provocando in esso deformazioni non permanenti se sollecitato entro una certa soglia. Tali deformazioni longitudinali sono funzione diretta della tensione a cui è sottoposto il cavo per l'azione del peso dell'*end effector*.
Sarà utile allo scopo misurare il peso *del terminale* e determinare, oltre alla costante elastica del cavo utilizzato, l'angolazione assunta dai cavi rispetto alla verticale definito angolo interno.

$$P_{EE} = m_{EE} \times g \quad [N] \quad [Eq.3.13]$$

3.4.1 Angolo interno

Per calcolare la tensione è necessario determinare come il peso si distribuisca in tutti i cavi. Essendo il peso dipendente dall'accelerazione gravitazionale, la variabile chiave per risolvere il problema risiede nell'angolo interno.
Tale angolo è quello che si crea tra il cavo$_{(i)}$ e la verticale passante per l'*Ep*$_{(i)}$. Per calcolare questo angolo è necessario confrontare la lunghezza dei due cateti che definiscono il triangolo rettangolo come mostrato in figura 3.7.
Il primo cateto è determinato dalla distanza esistente tra l'*end effector* e la verticale passante per l'*Ep*$_{(i)}$ (Dist), mentre il secondo cateto è definito dalla distanza che esiste tra la quota Z dell'*Ep*$_{(i)}$ (ZEp$_{(i)}$) e la quota Z dell'*end effector* o punto di ancoraggio del cavo (ΔZ).

Dunque si può procedere come segue:

$$\Delta Z_{(i)} = Z_{Ep(i)} - Z_{EE(i)} \quad [m] \quad [Eq.3.14]$$

$$Dist_{(1)} = \sqrt[2]{X^2 + Y^2} \quad [m] \quad [Eq.3.15\ a]$$

$$Dist_{(2)} = \sqrt[2]{X^2 + \left(\mathrm{Y\ Ep_{(2)}} - \mathrm{Y}\right)^2} \quad [m] \quad [Eq.3.15\ b]$$

$$Dist_{(3)} = \sqrt[2]{\left(X\ \mathrm{Ep_{(3)}} - \mathrm{X}\right)^2 + \left(\mathrm{Y\ Ep_{(3)}} - \mathrm{Y}\right)^2} \quad [m] \quad [Eq.3.15\ c]$$

$$Dist_{(4)} = \sqrt[2]{(\mathrm{X\ Ep_{(2)}} - \mathrm{X})^2 + \mathrm{Y}^2} \quad [m] \quad [Eq.3.15\ d]$$

$$\alpha_{(i)} = \tan^{-1}\left(\frac{Dist_{(i)}}{\Delta Z_{(i)}}\right) \quad [°] \quad [Eq.3.16]$$

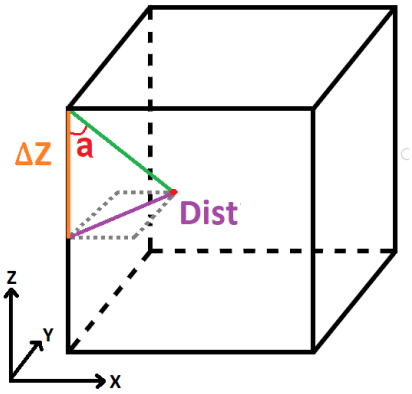

Figura 3.7 Angolo interno del cavo

3.4.2 Tensione

Il secondo dato da calcolare è la tensione generata dal peso dell'*end effector* e come essa si ripatisce nei diversi cavi. Questo risulta utile per determinare le eventuali correzioni necessarie e per poter evitare anche posizioni del *workspace* caratterizzate da livelli di tensione non compatibili con il funzionamento siano esse elevate o troppo deboli.

Tenendo presente che il peso dell'*end effector* (P) sarà ripartito tra i diversi cavi, si può affermare che il vettore della risultante (R) sarà dato dalla somma delle componenti verticali delle tensioni dei cavi e dovrà dunque avere intensità uguale a P e verso opposto.

In via preliminare, si procede determinando la tensione ipotetica (T) generata da P, considerando tale valore come se fosse gravante

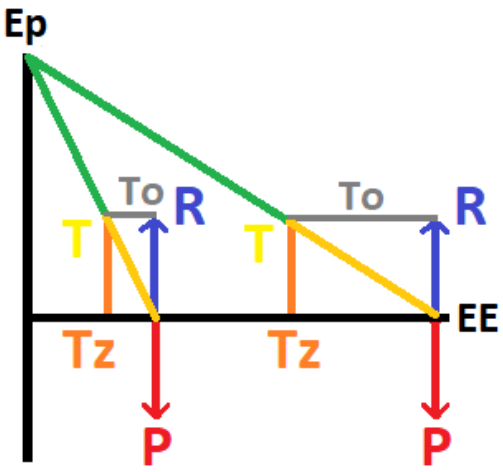

Figura 3.8 Ripartizione delle tensioni nei cavi

completamente su un unico cavo e ripetendo l'operazione per ogni cavo. Questo primo calcolo serve a definire la proporzione tra la componente T e Tz, ovvero la componente verticale corrispondente.
Le componenti verticali così determinate vengono sommate e tale somma viene riportata al peso P calcolando i singoli rapporti tra le componenti verticali $Tz_{(i)}$ e la somma di tutte le componenti verticali, moltiplicando il valore così ottenuto per P.
In questo modo si ottengono le componenti verticali effettive determinate dalla tensione del cavo e di conseguenza sarà possibile ricavare anche la componente orizzontale e la tensione effettiva corrispondente. Nella figura 3.8 nella pagina precedente viene mostrato graficamente il principio esposto e di seguito le formule per ottenere il risultato.

Considerato che :

$$R = P \quad [N] \qquad [Eq.3.17]$$

Se α è minore o uguale a 45°

$$T_{(i)} = \frac{P}{\cos\left(\alpha_{(i)}\right)} \quad [N] \qquad [Eq.3.18\ a]$$

Se α è maggiore di 45°

$$T_{(i)} = \frac{P}{\sin\left(\alpha_{(i)}\right)} \quad [N] \qquad [Eq.3.18\ b]$$

$$Tz_{(i)} = T_{(i)} \times \cos\left(\alpha_{(i)}\right) \quad [N] \qquad [Eq.3.19]$$

$$Tz_{totale} = \sum_{i=1}^{4} Tz_{(i)} = P \quad [N] \qquad [Eq.3.20]$$

$$Tz_{corretta(i)} = \frac{P}{Tz_{totale}} \times Tz_{(i)} \quad [N] \qquad [Eq.3.21]$$

In questo modo la somma delle componenti verticali di tutti i cavi sarà uguale a P, realizzando l'uguaglianza $P = R$.
Non rimane dunque che calcolare il valore corretto delle tensioni a cui sono soggetti i cavi e la componente orizzontale della tensione.

$$T_{(i)} = \frac{Tz_{corretta(i)}}{\cos(\alpha_{(i)})} \quad [N] \quad [Eq. 3.22]$$

$$To_{(i)} = T_{(i)} \times \sin(\alpha_{(i)}) \quad [N] \quad [Eq. 3.23]$$

Il momento resistente a livello del motore può essere calcolato nel seguente modo:

$$M_{(i)} = T_{(i)} \times r\ drum_{(i)} \quad [Nm] \quad [Eq. 3.24]$$

Utilizzando questo metodo si possono calcolare le tensioni dei cavi all'interno del volume di lavoro, escluso il caso in cui la quota Z dell'*end effector* coincide con ZEp ovvero quando l'angolo interno $\alpha = 90°$ perché nel calcolo appena introdotto il denominatore sarebbe uguale a 0.

Dato che la regione di workspace che forma angoli interni di $90°$ non viene presa in considerazione nello svolgimento delle operazioni ciò non rappresenta un ostacolo al corretto funzionamento.

3.4.3 Caratteristiche del cavo

I cavi sottoposti alla tensione generata da un carico applicato ad essi possono subire, a seconda delle proprie caratteristiche costruttive e tecnologiche, un allungamento che può essere più o meno marcato. Per questo motivo, dato che l'accuratezza di posizionamento dipende dalla lunghezza dei cavi, è necessario verificare le caratteristiche tecniche del cavo utilizzato per verificare se l'intensità del fenomeno dell'allungamento è tale da determinare possibili errori rendendo necessario il calcolo della correzione.
Il metodo di calcolo che può essere utilizzato è la legge di *Young*, che rende possibile il calcolo dell'allungamento a partire dalla sezione del cavo e dal modulo di *Young, valore* tipico del materiale che compone il cavo. Il modulo di *Young* (E) è ricavato dal rapporto tra il carico specifico (σ) e la deformazione specifica (ε).
Il carico specifico è dato dal rapporto tra la forza a cui è sottoposto il cavo (F) e l'area della sua sezione (S), mentre la deformazione specifica è data dal rapporto tra l'allungamento lineare del materiale (Δl) e la sua lunghezza a riposo (l) in assenza di carico.

$$\sigma = \frac{F}{S} \quad [\ N/m^2\ ] \quad [Eq.3.25]$$

$$\varepsilon = \frac{\Delta l}{l} \quad [\ ] \quad [Eq.3.26]$$

$$E = \frac{\sigma}{\varepsilon} \quad [\ N/m^2\ ] \quad [Eq.3.27]$$

Partendo dalla legge di *Young* si può ricavare la formula per ricavare l'entità del fenomeno di allungamento del cavo:

$$\Delta l = \frac{F \times l}{S \times E} \quad [m] \quad [Eq.3.28]$$

Nel caso del prototipo in questione, il cavo utilizzato è una treccia *Dyneema,* ovvero una fibra sintetica brevettata da un'azienda olandese (*DSM*) che si rivela particolarmente adatta alla produzione di cavi da trazione che viene utilizzata in diverse applicazioni sportive e militari per le sue caratteristiche tecnologiche.
I cavi prodotti con questo materiale hanno una resistenza paragonabile a quella dell'acciaio e presentano il vantaggio di resistere bene sia a sforzi derivanti da torsione che da piegamento ed in particolare quest'ultimo risulta praticamente esente da elasticità il che lo rende dimensionalmente molto stabile.
Dunque in questo caso tenendo conto dei livelli di tensione sviluppati all'interno del volume di lavoro e grazie alle caratteristiche del cavo scelto non si rende necessario applicare una correzione dovuta all'elasticità del cavo ma resta comunque utile considerarla per consentire al prototipo di funzionare correttamente anche nel caso in cui si utilizzi un cavo con caratteristiche diverse.

3.5 Catenaria

3.5.1 Un po' di storia

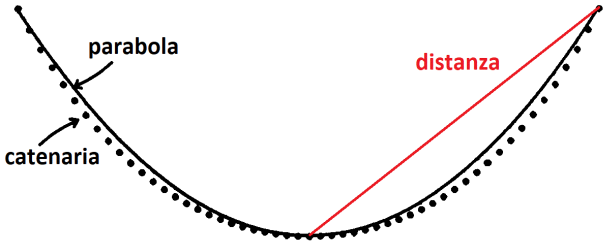

Figura 3.9 Catenaria, parabola e distanza

La catenaria, detta anche curva funicolare, è la curva secondo cui si dispone una fune omogenea, perfettamente flessibile e non estendibile, appesa a due punti estremi, che sia lasciata pendere soggetta al proprio peso. Per semplicità spesso la disposizione spaziale assunta da una fune nello spazio viene rappresentata o immaginata come rettilinea e questo modello di rappresentazione si rivela spesso errato.
A volte tale disposizione spaziale viene rappresentata mediante l'andamento di una parabola ma nella realtà il comportamento della fune non è esattamente questo specie nei casi in cui tale fune è soggetta a limitati livelli di tensione (Pidatella, 2012).
Come si può notare in figura 3.9 il concetto di distanza tra due punti non offre la possibilità di determinare con precisione la lunghezza di un ipotetico cavo che li colleghi entrambe così come neanche la parabola riesce ad avere un sufficiente grado di accuratezza.
Questo problema aveva interessato in passato molteplici personaggi illustri (Bocus, 2012) come Galileo Galilei che nei "*Discorsi e dimostrazioni matematiche intorno a due nuove scienze*"(1638), affermava che una fune in tali condizioni assumesse una forma molto simile in certi casi a una parabola.
Tale intuizione si rivelò successivamente in parte imprecisa e nel 1673 *Ignace Gaston Pardies* dimostrò come una fune omogenea dotata di massa non possa atteggiarsi secondo un profilo parabolico dando inizio, qualche anno dopo, a un dibattito sul tema.
Nel 1690 si accese la discussione tra gli scienziati dell'epoca quando *Jakob Bernoulli* propose il problema alla loro attenzione attraverso una sua memoria pubblicata dagli "*Acta Eruditorum*" di Lipsia. *Gottfried Wilhelm von Leibniz* si interessò anch'esso all'argomento e promise di pubblicare le soluzioni ottenute con il suo metodo (calcolo differenziale) entro la fine dell'anno se nessuno fosse giunto a un risultato entro la tale data. In realtà, Johann *Bernoulli*, fratello di *Jakob* trovò una sua soluzione e la mandò all'editore chiedendo gentilmente di aggiungere alla sua anche la soluzione raggiunta da *Leibniz*. Parallelamente anche *Christian Huygens* si era messo al lavoro.
L'epilogo fu che nell'estate del 1691 uscì un volume degli "*Acta Eruditorum*" presentato

dall'editore che, richiamando il problema espresso da *Bernoulli* in precedenza, riproponeva le soluzioni pervenute.
Il risultato fu che *Huygens* arrivò a una sua interpretazione utilizzando il metodo geometrico, *Leibniz* diede la formula analitica corretta della catenaria mentre, *Johann Bernoulli* offrì alcune costruzioni corrette della catenaria e ne riportò diverse proprietà.

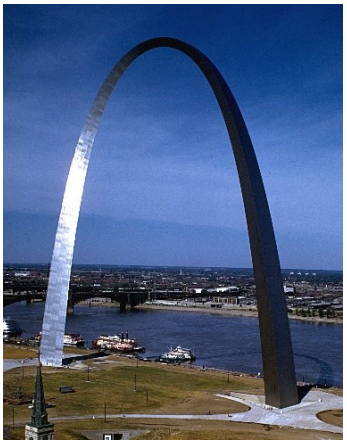
Figura 3.10 Gateway Arch

Questo studio è ritenuto molto importante perché fu il primo degli argomenti trattati e risolti con rigore dalla meccanica moderna con l'utilizzo dei primi elementi dell'analisi infinitesimale.
Inoltre, grazie alle sue proprietà fisiche e geometriche conferite dal fatto che una catenaria ha la peculiarità di avere in ogni suo punto una distribuzione uniforme del peso totale, spesso questa sua proprietà è stata utilizzata per realizzare manufatti come per esempio il *Gateway Arch, (St. Louis, Missouri (USA)), riportato in* figura 3.10, o infrastrutture portanti come il *Tyne Bridge di New Castle* (*UK*) in figura 3.11.


Figura 3.11 Tyne Bridge

Questi elementi, infatti, sono realizzati con una conformazione definita come arco catenario ovvero una catenaria riflessa.

Per quanto riguarda la realizzazione del prototipo, il problema della gestione della variabile lunghezza dei cavi è stato affrontato cercando di applicare il modello matematico delle catenarie come avviene in alcune applicazioni su larga scala come per esempio per la *SpiderCam* e verificare se, a livello di configurazione e scala del prototipo, sia necessario tale meccanismo di correzione.

3.5.2 *L'equazione della catenaria*

L'equazione della catenaria fa parte delle funzioni iperboliche ovvero particolari funzioni dotate di analogie con le funzioni trigonometriche. Le funzioni iperboliche possono essere definite come:

- data un'iperbole equilatera unitaria (con i parametri a e b uguali a 1) centrata nell'origine degli assi;

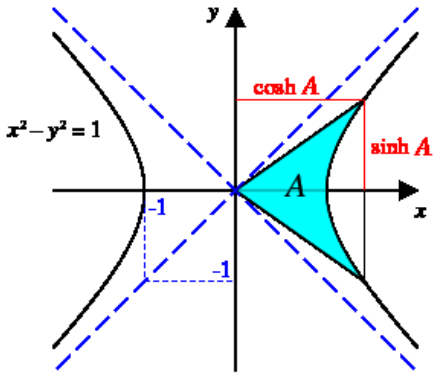

Figura 3.12 Funzioni iperboliche

$$\frac{X^2}{a^2} - \frac{Y^2}{b^2} = 1 \qquad [Eq.3.29]$$

- Dato un certo angolo definito, si può considerare il settore iperbolico di area A e parallelamente individuare un semi-settore corrispondente. Si rende possibile in questo modo individuare un punto qualsiasi lungo l'iperbole che possiede ascissa pari al coseno iperbolico (cosh) e ordinata pari al seno iperbolico (sinh).

Tralasciando le numerose dimostrazioni matematiche, l'equazione della catenaria deriva dalle funzioni iperboliche e per la rappresentazione di questa curva nello spazio bidimensionale è necessaria la conoscenza di alcuni parametri ovvero delle caratteristiche del cavo e le condizioni di tensione a cui questo è sottoposto. Queste caratteristiche determinano un diverso andamento della funzione.

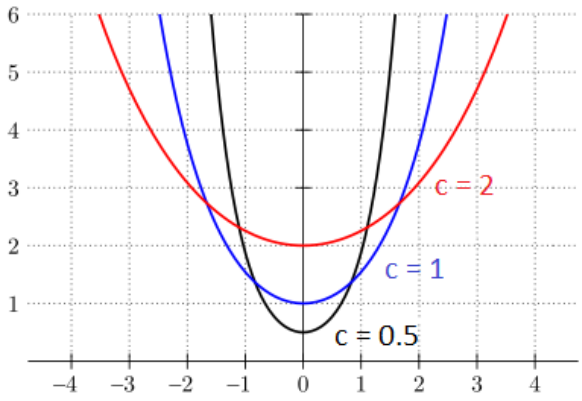

Figura 3.13 Equazione della catenaria

La forma canonica dell'equazione della catenaria è la seguente:

$$Y = c \times \cosh\left(\frac{x}{c}\right) \qquad [Eq.3.30]$$

Il grafico in figura 3.13 mostra come l'andamento della funzione dipenda sostanzialmente dal parametro indicato con la lettera c, definito in letteratura anche come parametro di tesatura della fune.

Per determinare con precisione la lunghezza corretta del cavo$_{(i)}$, dunque è necessario determinare in prima istanza il valore assunto da tale parametro che coincide anche con la distanza che esiste tra l'origine degli assi e il minimo valore della funzione.

Per il calcolo di questo parametro si può procedere nel seguente modo:

$$T_{o(i)} = T_{(i)} \times sin(\alpha_{interno}) \quad [N] \qquad [Eq.3.31]$$

$$P_{lineare\ cavo(i)} = M_{lineale} \times g \quad [N] \qquad [Eq.3.32]$$

$$c_{(i)} = \frac{T_{x\,(i)}}{P_{lineare\ cavo\ (i)}} \quad [m] \qquad [Eq.3.33]$$

Dopo aver calcolato il parametro c, utilizzando un sistema di riferimento bidimensionale con origine nel centro del segmento che collega i due punti come suggerito dal report tecnico per la modellazione di conduttori elettrici aerei (Zoppetti, 2003) è possibile trovare, con i dati a disposizione, la lunghezza del segmento di catenaria che separa due punti ovvero l'*exit point*$_{(i)}$ e il corrispondente ancoraggio a livello dell'*end effector*.

$$L\ Catenaria_{(i)} = \sqrt{4{c_{(i)}}^2 \times \sinh\left(\frac{a}{c_{(i)}}\right)^2 + 4b^2} \quad [m] \quad [Eq.\ 3.34]$$

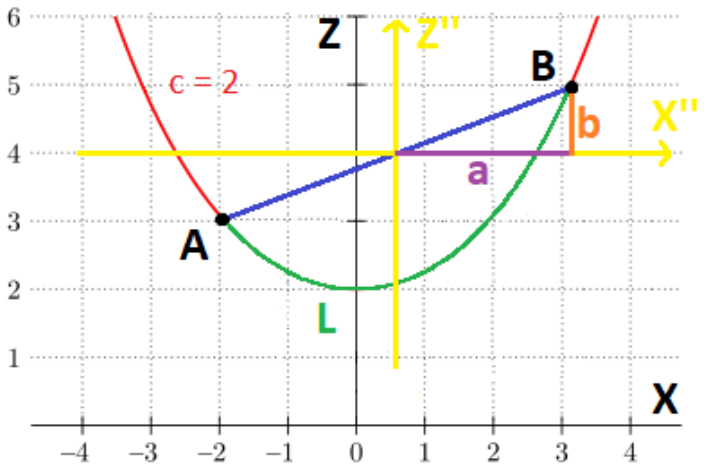

Figura 3.14 Parametri calcolo lunghezza della catenaria

3.6 Calcolo del target corretto

Il calcolo della correzione è necessario a limitare gli errori di posizionamento dell'*end effector* dovuti all'andamento non perfettamente rettilineo del cavo e a contrastare il fenomeno dell'eventuale elasticità del cavo.
Dopo aver spiegato come è stata determinata la funzione di elasticità del cavo, l'andamento secondo la funzione della catenaria e come sono state calcolate le tensioni, non rimane che calcolare il *target* corretto.
In questo modo è possibile calcolare la correzione da applicare alla lunghezza dettata dalla distanza che esiste tra l'***exit point***$_{(i)}$ e l'ancoraggio del cavo all'*end effector* per ottenere il valore del target corretto. Si può dire con certezza che tale distanza rettilinea sia la minima, su di essa agendo la tensione e per il fatto che il cavo è soggetto al proprio peso, il target corretto corrisponderà molto probabilmente ad un valore maggiore. Si può procedere allo scopo come segue:

$$L_{catenaria(i)} = L_{catenariaTarget(i)} + \Delta L_{tensione(i)} \quad [m] \quad [Eq.3.35]$$

Di conseguenza:

$$L_{catenariaTarget(i)} = L_{catenaria(i)} - \Delta L_{tensione(i)} \ [m] \quad [Eq.3.36]$$

$$Target_{corretto(i)} = \frac{L_{catenariaTarget(i)}}{step} \quad [step] \quad [Eq.3.37]$$

3.7 Simulazioni

Per effettuare il dimensionamento dei motori e per verificare gli algoritmi di correzione e controllo, sono state effettuate alcune simulazioni mediante apposito *script* in linguaggio *MATLAB*.
Le simulazioni sono state eseguite mediante la riproduzione del volume di lavoro per mezzo di matrici multidimensionali. Tali matrici sono state ricavate suddividendo lo spazio tridimensionale dello spazio di lavoro in una griglia costituita da nodi equidistanti 0.05 m l'uno dall'altro.
In ogni nodo sono stati poi effettuati i calcoli inerenti alla determinazione della tensione dei cavi nel caso in cui l'*end effector* si trovi in quella posizione del *workspace* determinando di conseguenza il momento resistente sviluppato a livello dei singoli motori. Tale tipo di analisi è stata eseguita per verificare l'intensità dell'intervento della correzione dovuto alla catenaria e, dato che il cavo utilizzato è considerato inestensibile, non per la correzione dovuta alla sua elasticità.
In figura 3.15 viene illustrata la composizione della griglia mettendone in evidenza i nodi.

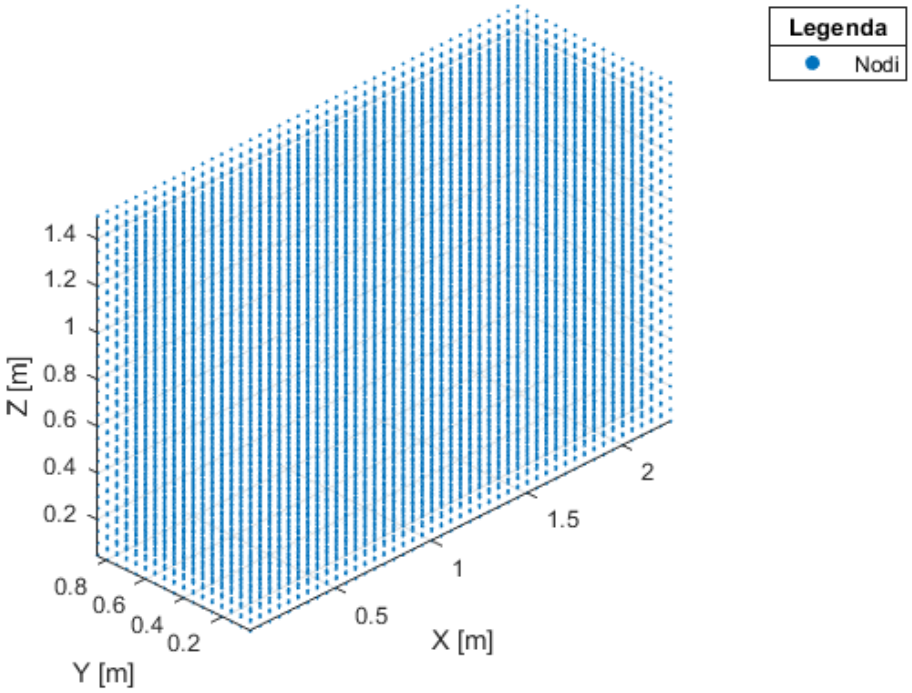

Figura 3.15 Determinazione dei nodi

Per ciò che attiene lo studio dello sfruttamento della coppia motrice all'interno del *workspace*, il procedimento di calcolo utilizzato è quello indicato al paragrafo 3.4.2 e prevede l'utilizzo di un ipotetico *end effector* di massa pari a 0,5 Kg.
Questo procedimento di calcolo è stato applicato a tutti i nodi e successivamente i risultati

ottenuti sono stati confrontati con i dati tecnici delle diverse alternative di motoriduttori presenti sul mercato per scegliere quelli più appropriati.
Il criterio di scelta è stato quello di poter garantire adeguate velocità di moto dell'*end effector* all'interno del *workspace* e garantire il più possibile la corrispondenza tra la velocità impartita a livello di *software* dall'algoritmo di controllo e la velocità angolare istantanea ottimale idealmente sviluppata dal motoriduttore.
Per ottenere questo è utile sfruttare la coppia motrice sviluppata dal motoriduttore per una quota inferiore al 40 % della coppia massima erogabile in un'area più estesa possibile del volume di lavoro in modo da non fare discostare significativamente la velocità di rotazione prevista da quella effettiva.

In figura 3.16 viene riportato un grafico che rappresenta l'andamento della curva di erogazione della coppia di un generico motore elettrico in corrente continua dato che nel *datasheet* del motoriduttore adottato non era fornito un grafico specifico da parte del costruttore, che comunque rappresenta un principio di massima utile in ogni caso in fase di progettazione.

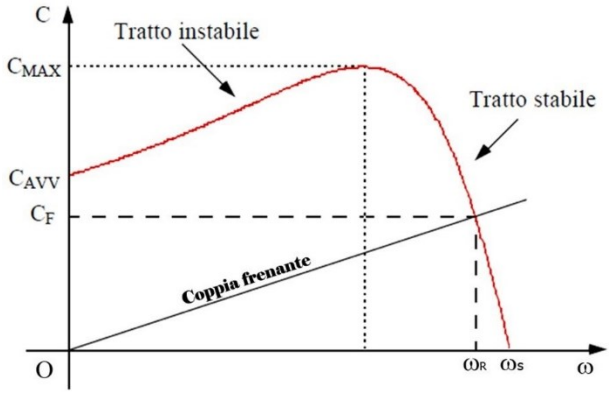

Figura 3.16 Curva di erogazione della coppia di un motore DC

I risultati ottenuti nella simulazione che prende in considerazione le caratteristiche tecniche del motoriduttore adottato nella realizzazione del prototipo sono riportati in figura 3.17 e sono stati utilizzati per determinare un limite di sicurezza alla quota massima raggiungibile dall'*end effector*.
Tale quota è stata fissata a 1,35 m perché si ritiene, dall'analisi dei dati, in grado di assicurare livelli di tensioni compatibili con il corretto funzionamento fornendo un adeguato livello di sicurezza utile a scongiurare tensioni eccessive per i cavi e la stabilità della struttura. Dai calcoli si è dedotto inoltre che l'altezza massima raggiungibile è di 1,45 m, che l'end effector è in grado di raggiungere potenzialmente il 96,7% dello spazio disponibile, e che l'utilizzo medio della coppia in tutte le posizioni raggiungibili è del 10%.

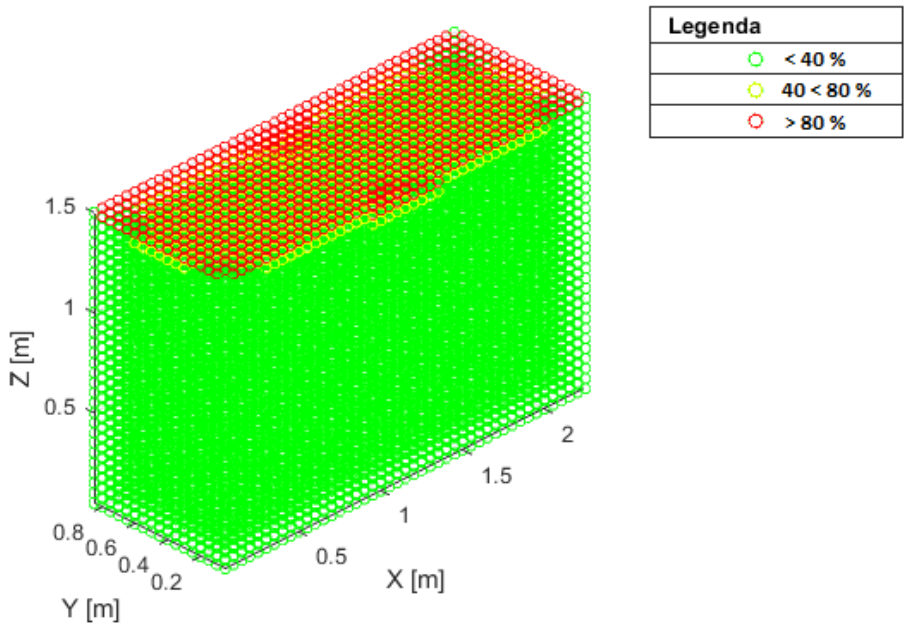

Figura 3.17 Percentuale di sfruttamento della coppia motrice nel workspace

Dall'analisi delle simulazioni inerenti all'utilizzo della correzione catenaria è emerso che l'algoritmo impiegato funziona correttamente ma l'utilizzo con questa configurazione, di peso dell'*end effector* e di peso lineare del cavo utilizzato, non contribuisce in maniera significativa ad aumentarne l'accuratezza.

I risultati delle simulazioni inerenti a questo meccanismo di correzione sono riportati in forma di quota relativa di intervento della correzione, sia come intervento a livello del singolo cavo (figura 3.18) che come combinazione di tutti i cavi (figura 3.19).

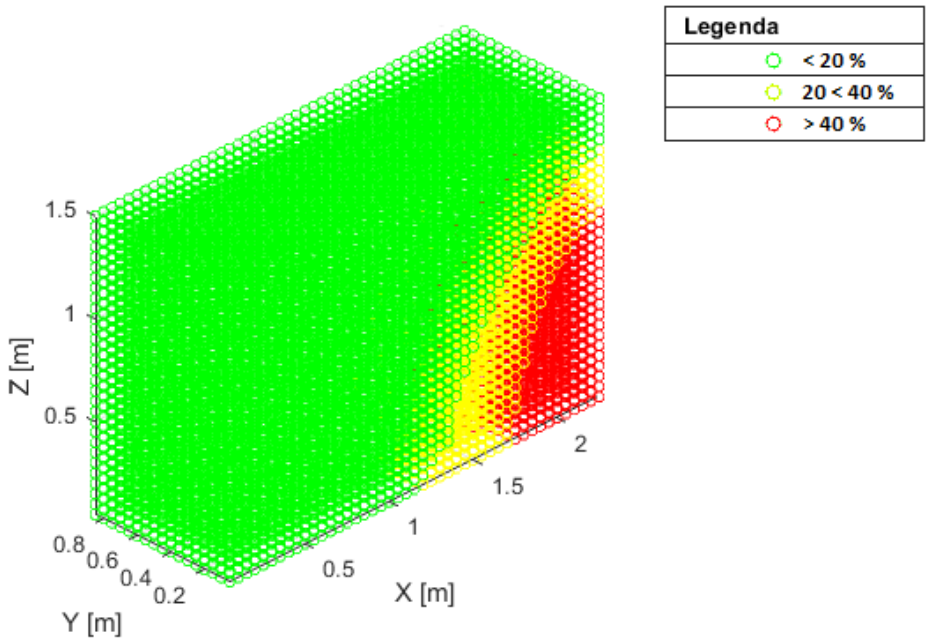

Figura 3.18 Quota di intervento relativa inerente al calcolo della correzione catenaria del singolo cavo A

Nel primo caso la quota di intervento è calcolata come rapporto tra la correzione calcolata nel singolo punto e il valore di correzione massimo calcolato all'interno del workspace mentre nel secondo caso la quota di intervento che viene considerata per ogni posizione è la maggiore tra tutti i cavi.

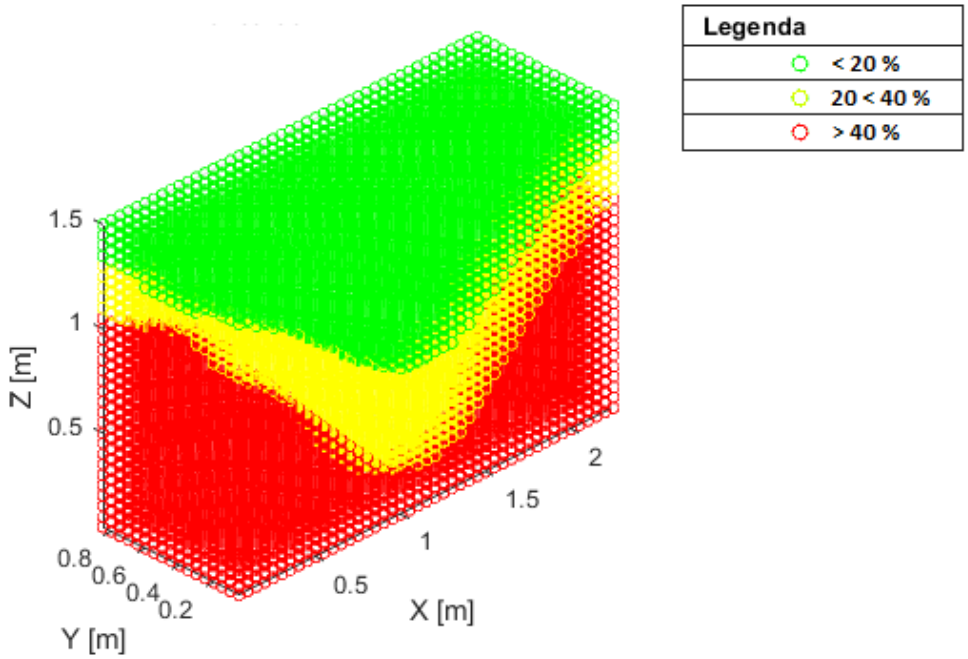

Figura 3.19 Quota di intervento relativa inerente al calcolo della correzione catenaria

In conclusione si può affermare dunque che i due meccanismi di correzione pensati in sede di progettazione non sono stati utili con una configurazione di questo tipo ma potrebbero esserlo sicuramente in altri progetti di scala più ampia o con diverse configurazioni di cavo e peso dell'*end effector* come analizzato in precedenza dall'analisi della letteratura inerente al tema.

3.8 Componenti *hardware*

Nella sezione che segue viene riportata una breve trattazione inerente alle caratteristiche e ai principi di funzionamento dei componenti che si sono rivelati necessari nella realizzazione del progetto.
Questa sezione ha lo scopo di individuare in maniera precisa i componenti e introdurne le caratteristiche tecniche e di funzionamento, prima di passare alla sezione che descrive nel dettaglio la composizione delle singole unità motore, di controllo, di tutte le componenti del *frame* e delle componenti accessorie.

3.8.1 Microcontrollore (Arduino Uno)

Arduino è una piattaforma hardware *open source* ideata e sviluppata dai membri dell'*Interaction Design Institute* di Ivrea e pensata come strumento utile per la prototipazione rapida utilizzabile a fini hobbistici, didattici e professionali. Con questo microcontrollore possono essere realizzati, in maniera relativamente semplice, progetti che possono comprendere la gestione automatizzata di sensori digitali, analogici, attuatori di vario genere e che prevedano l'eventuale comunicazione con altri dispositivi per mezzo di diversi protocolli di comunicazione (tra cui per esempio *I^2C e SPI*). Il *software* e gli schemi circuitali dell'*hardware* vengono forniti in maniera libera dagli ideatori.
L'ambiente di sviluppo integrato *(IDE)* multipiattaforma (disponibile per *Linux*, *Apple* e *Windows*) permette di lavorare con Arduino in maniera semplice, in quanto i programmi sono scritti in un linguaggio di programmazione semplice e intuitivo, chiamato *Wiring*, derivato dal *C* e dal *C++*, liberamente scaricabile e modificabile.
I programmi così realizzati vengono chiamati *sketch*.
Esistono diverse versioni dell'hardware (*Uno*, *Due*, *Mega2560*, *ZeroPro*, *Yun*, *Nano*, *Mini*, ecc..) che differiscono per dimensioni (ingombri, numerosità e tipologia delle porte), prestazioni e accessori e periferiche che possono essere integrati dall'origine o che vanno

Figura 3.20 Microcontrollore Arduino Uno

aggiunti assemblandoli successivamente, se necessari.
Per la realizzazione del prototipo sono state impiegate diverse schede sia Arduino Uno dotata di microcontrollore ATmega328P con memoria flash da 32 KB *EEPROM* da 1 KB e *SRAM* da 2 KB che Arduino Nano, dotate di uguali prestazioni ma caratterizzate da dimensioni inferiori.
Queste schede possiedono 14 pin digitali di cui 6 con possibilità di modalità di funzionamento *PWM* (tipologia di modulazione del segnale descritta nel paragrafo 3.3.6*)*, 6 pin analogici e un regolatore di tensione che ne permette l'alimentazione con diversi livelli di voltaggio.

3.8.2 Encoder

Un *encoder* rotativo, o trasduttore di posizione angolare, è un sensore che permette di codificare la rotazione di un albero in un segnale elettrico, fornendo quindi la possibilità di determinare il senso di rotazione, la velocità, il numero di giri compiuti o la posizione assoluta in cui si trova l'albero a cui esso è vincolato (a seconda del tipo di *encoder* cambiano le informazioni ottenibili).

Figura 3.21 Rotary encoder

L'*output* dell'*encoder* rappresenta dunque un segnale che può essere elaborato da un sistema di controllo ed essere usato come feedback di posizionamento per attuatori rotanti.
Il dispositivo è composto generalmente da una parte mobile solidalmente vincolata all'albero rotante di cui si vuole monitorare la posizione e da una parte fissa o corpo dove sono presenti le parti elettroniche che effettuano la trasformazione delle posizioni dell'albero mobile in segnali elettrici (trasduzione).
Gli *encoders* possono essere classificati a seconda del principio di trasduzione su cui si basa il suo funzionamento. Questi si possono classificare come:

- *encoders* capacitivi/induttivi, basati sulla capacità di un sensore di prossimità (*proximity sensor*) di leggere i denti di un ingranaggio o delle forature presenti su un disco metallico (ruota fonica) carpendo così il comportamento dell'albero mobile. L'ingranaggio o il disco costituiscono il rotore dell'*encoder*;
- encoders magnetici, basati sull'effetto *hall*, in questi un sensore è in grado di rilevare il campo magnetico prodotto da un magnete permanente, rilavando posizione e spostamento di un albero rotante a cui il magnete viene applicato;

- *encoders* potenziometrici, basati sulla caratteristica di un potenziometro di emettere un segnale elettrico proporzionale alla posizione che assume il suo rotore. Questi tipi di *encoder* sono pertanto solo di tipo assoluto.
- *encoders* ottici, dotati di sensori ottici (fotoresistenze o fotodiodi) in grado di leggere una matrice di aree trasparenti e opache, stampate sul rotore. In modo non dissimile dagli encoder capacitivi, la matrice è realizzata con un'alternanza di aree tali da codificare un settore di rotore pari alla sua risoluzione angolare.

Tali trasduttori, inoltre, possono essere classificati in base alla logica e al principio di funzionamento in due grandi gruppi principali, ovvero, *encoders* incrementali ed *encoders* assoluti.
Vengono definiti incrementali quando i segnali d'uscita sono proporzionali in modo incrementale allo spostamento effettuato. Segnalano dunque unicamente le variazioni rilevabili rispetto a un'altra posizione che viene assunta come riferimento.
Il rotore di questi *encoders* è collegato ad un disco la cui fattura dipende dal tipo di trasduzione utilizzato dal dispositivo. La rilevazione dello spostamento angolare avviene mediante il "conteggio" degli impulsi rilevati dal sensore che possono essere 0 logico o 1 logico a seconda che si trovi in stato basso o alto.
Il dispositivo digitale che rileva il numero di impulsi è un contatore il cui ingresso di conteggio risulta attivo sul fronte di salita (stato alto). Di conseguenza il numero degli impulsi contati è direttamente proporzionale allo spostamento angolare dell'*encoder*, e quindi dell'albero a cui è vincolato.
La risoluzione o il passo minimo misurabile dall'encoder dipende dal numero *(n)* di impulsi conteggiabili per ogni rivoluzione, tale passo misurato in gradi sarà uguale dunque a $360°/n$.
Per poter rilevare il verso di rotazione, l'*encoder* presenta una seconda linea di trasduzione *(B)* sfasata di metà passo. Effettuando un controllo dei fronti di salita degli impulsi in uscita da *A* e *B*, un sistema logico riesce a stabilire il verso di rotazione del disco oltre all'entità di tale spostamento (figura 3.22).

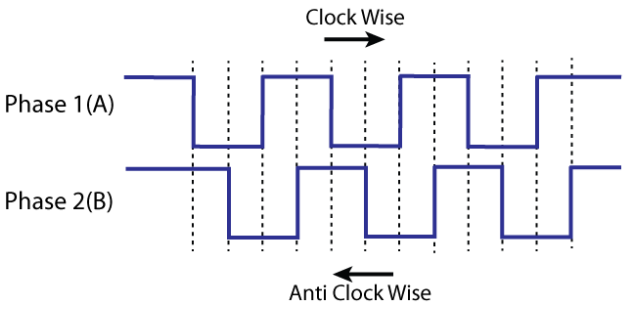

Figura 3.22 Segnale encoder incrementale Canale A e B

Per garantire il corretto funzionamento questo segnale in uscita dai due canali, ricevuto dal microcontrollore, genera opportunamente un *interrupt hardware* e attiva un *interrupt service routine (ISR)* ovvero il microcontrollore interrompe le azioni che stava svolgendo il processore per eseguire un preciso compito prima di riprendere l'azione che aveva

interrotto. Questa *routine* avviene ogni qual volta si verifica un'alterazione del segnale a livello di alcune porte digitali (solo *D2* e *D3*) impostabili come *input* e solo se lo *sketch* prevede l'attivazione di questa modalità di funzionamento per mezzo della funzione *attachInterrupt()* che individua la porta, la modalità di funzionamento e la *routine* associata.
Per modalità di funzionamento si intende la possibilità di scegliere se l'*ISR* sia attivata nel caso in cui la porta associata:

- passi dallo stato alto (1 logico) a quello basso (0 logico), definita *falling*;
- passi dallo stato basso a quello alto, definita *rising;*
- registri un qualsiasi cambiamento di stato, chiamata *change;*
- si trovi allo stato alto, *high*;
- si trovi allo stato basso, *low*.

Nel prototipo il contatore sfrutta la modalità *rising* sul pin *D2* per gestire il conteggio.

Un difetto di tali dispositivi è dato dal fatto che in mancanza di alimentazione per il microcontrollore che gestisce l'*encoder* venga perso il conteggio raggiunto.
Tale problema può essere ovviato accoppiandolo ad una memoria alimentata da batterie tampone o salvando il valore raggiunto dal conteggio.

Sono encoder assoluti, invece, quelli in cui il disco è suddiviso in settori che andranno a loro volta a comporre un codice (binario oppure *Gray*).

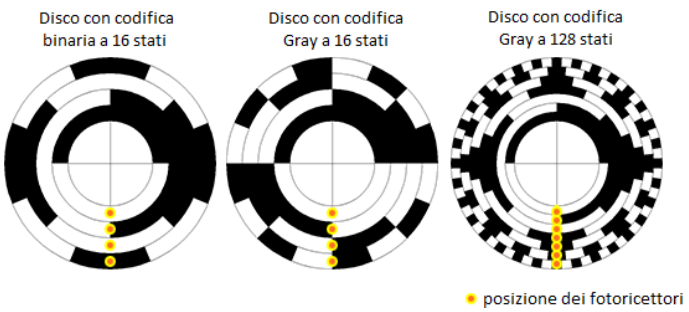

Figura 3.23 Dischi codificati di encoder assoluti

Il rotore in questo caso è diviso in *n* corone circolari e in 2*n* spicchi. Ogni settore (spicchio) avrà *n* piccole aree che a seconda che siano opacizzate o trasparenti corrisponderanno al dato 1 logico o 0 logico.
Ogni area avrà il valore di un *bit* e il *bit* meno significativo (*LSB*) sarà quello della corona più interna. Per evitare errori di lettura invece del codice binario puro vengono utilizzati altri codici, tra i quali il più importante è il codice *Gray*, questo prevede che il passaggio da un numero al successivo avvenga sempre variando un'unica cifra binaria, evitando così che nel passaggio tra la lettura di un numero e del successivo possano aversi delle letture errate, per questo motivo è uno tra i più utilizzati. In questo caso la trasduzione avviene con un fotoemettitore e un corrispondente foto-rilevatore per ogni corona circolare del disco.
L'*encoder* assoluto ha il vantaggio di dare informazioni che non vengono perdute in caso di

mancanza di alimentazione, è più costoso di quello incrementale e non è in grado di effettuare misure della velocità.
Gli e*ncoder* per la loro vastissima gamma di modelli, qualità e robustezza, sono validamente applicati in tutto il mondo in controlli di processo industriale in: robot industriali, macchine utensili, strumenti di misura, plotter, divisori, laminatoi, macchine per lamiera, bilance, antenne, telescopi, gru, carri ponte, presse, macchine da stampa e imballaggio.
Per la realizzazione di un prototipo di questo tipo, l'*encoder* è un elemento fondamentale perché necessario a garantire la possibilità di monitorare con continuità la posizione angolare effettivamente assunta dall'albero a cui esso è vincolato, fornendo la possibilità di determinare l'errore di posizionamento e di conseguenza utilizzare questo sensore come *feedback* di posizionamento per il microcontrollore che gestisce il motore. Tale componente andrà scelto dunque in maniera ragionata tenendo conto:

- delle dimensioni di massima che avrà il *workspace*;
- delle dimensioni del *drum*;
- della risoluzione necessaria per ottenere una sufficiente accuratezza del posizionamento;
- della logica di funzionamento.

Per concludere si può citare il fatto che esistono anche soluzioni tecniche che propongono motori elettrici già accoppiati ad *encoder*, ma questa soluzione non è stata adottata perché troppo rigida per via del fatto che spesso queste soluzioni garantiscono risoluzioni per rotazione molto più elevate del necessario e per il fatto che avrebbe vincolato la scelta del motore.
A livello di architettura generale una risoluzione immotivatamente troppo elevata avrebbe potuto penalizzare i tempi di esecuzione tenendo occupato il contatore con migliaia di *interrupt* ogni rotazione senza migliorare in maniera significativa l'accuratezza.

Figura 3.24 Encoder incrementale magnetico

Per la realizzazione del prototipo sono stati scelti *encoders* incrementali magnetici dotati di una risoluzione di 256 impulsi per rotazione e il dispositivo digitale che si occupa della sua gestione è un microcontrollore dedicato (Arduino Nano) che svolge funzione di contatore.
Se necessario, per mantenere il conteggio in caso di spegnimento è stata pensata una funzione di salvataggio del dato proveniente dal contatore.

Per consentire le operazioni di calibrazione, è stata pensata una funzione in grado di provocare l'azzeramento simultaneo di tutti i contatori via *software* mediante un apposito messaggio inviato tramite *CAN BUS*.

3.8.3 Motoriduttore

Per motoriduttore si intende un motore elettrico in corrente continua *(DC)* il cui albero motore è accoppiato ad un riduttore a ingranaggi. Il motore elettrico trasforma l'energia elettrica in energia meccanica. Il principio di funzionamento prevede che una bobina, situata nel rotore quando è attraversata da un flusso di corrente continua e inserita in un campo magnetico, dovuto alla presenza di magneti situati nella parte fissa del motore o statore, genera una coppia che fa ruotare il rotore.

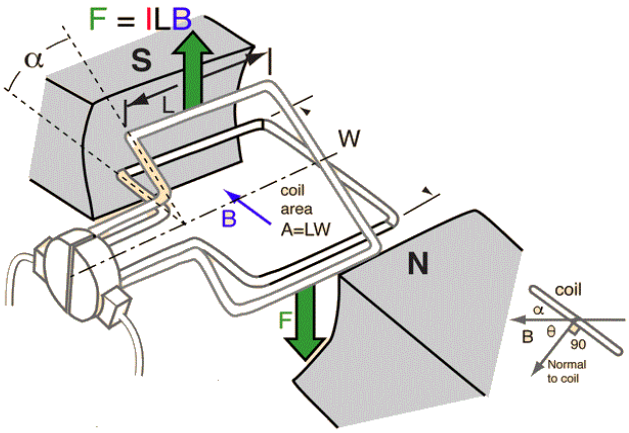

Figura 3.25 Principio di funzionamento e coppia motore DC

Per funzionare, ogni porzione di rotazione del rotore la direzione del flusso della corrente viene invertito grazie alla presenza delle spazzole e del collettore suddiviso in più parti permettendogli così di continuare a ruotare dato che il campo magnetico in cui è immerso il rotore generato dai magneti è orientato in modo da avere un lato Nord e uno Sud. La coppia erogata dal motore dipenderà dunque dalle sue caratteristiche costruttive e dalla corrente da cui è attraversato, la potenza è determinata a sua volta dal prodotto tra la coppia e la velocità angolare:

$$P = M \times \omega \quad [\text{W}] \quad [\text{Eq. } 3.39]$$

La potenza sarà erogata secondo caratteristiche proprie dei motori elettrici e spesso per necessità di utilizzo si rende necessario accoppiare il motore a un riduttore, o cambio, ovvero un congegno meccanico che ha lo scopo di consentire il più razionale sfruttamento della potenza sviluppata. Per semplicità, senza considerare gli attriti interni agli ingranaggi del riduttore che determinano la dissipazione di parte della potenza sviluppata dal motore, si può affermare che la potenza

Figura 3.26 Esempio di riduzione

sviluppata del motore entra mediante l'albero primario del riduttore, ed esce a livello dell'albero secondario subendo una trasformazione delle componenti che la caratterizzano, ovvero velocità angolare e coppia.
Per quantificare questo effetto si prende in considerazione il rapporto di trasmissione (τ) ovvero il rapporto matematico tra la velocità angolare della ruota conduttrice diviso quello della ruota condotta. Nel caso in cui, come in questo caso, si ha a che fare con ingranaggi sarà il rapporto tra il numero dei denti quando si ha a che fare con le pulegge si farà invece riferimento al raggio di queste.
Nel caso in cui la riduzione sia composta da più ingranaggi il τ finale sarà dato dal prodotto di tutti i singoli τ.

$$\tau = \frac{n\ denti\ conduttore}{n\ denti\ condotto} \qquad [\text{Eq. } 3.40]$$

$$\tau\,\text{tot} = \tau\,1 \times \tau\,2 \times \ldots \times \tau\,\text{n} \qquad [\text{Eq. } 3.41]$$

Si possono distinguere dunque trasmissioni:

- riduttrici, quando τ è maggiore di 1 e dunque l'albero secondario gira più lentamente e con una coppia più elevata;
- moltiplicatrici, con τ minore di 1 l'albero secondario gira più velocemente ma svilupperà meno coppia;
- imparziali se τ equivale a 1 e dunque non ci saranno variazioni di velocità e coppia.

Maggiore sarà l'aumento della coppia sviluppata a livello del suo albero di uscita maggiore sarà la riduzione del numero di giri rispetto all'albero in entrata e viceversa.
Per la realizzazione del progetto si è scelto, anche grazie alle simulazioni eseguite in via preliminare, di utilizzare quattro motoriduttori le quali caratteristiche sono riassunte in tabella 3.1.

Tabella 3.1 Dati tecnici motore elettrico

	Senza carico		Massima efficienza					Stallo
Tensione operativa	Velocità	assorbimento	Velocità	assorbimento	Coppia	Potenza	Efficienza	Coppia massima
Volt	giri/min	A	giri/min	A	Nm	W		Nm
6,0 - 12,0	7000	0,9	5700	5,5	0,07	41,3	0,63	0.43

Questo motore è accoppiato a un riduttore epicicloidale il cui rapporto di riduzione è pari a 104:1.
Tali motoriduttori presentano caratteristiche leggermente diverse a livello di riduttore

rispetto a quanto introdotto in precedenza perché dotati di meccanismo epicicloidale. Tale tipologia di riduzione prevede un ingranaggio primario definito solare, che si trova in posizione centrale, che ingrana e mette in rotazione tre o più ingranaggi definiti planetari che possono ruotare all'interno di un anello dentato esterno che risulta fisso. Di conseguenza gli assi dei planetari stessi, montati sul porta planetari si muovono lungo una traiettoria circolare. Il porta planetari è solidamente ancorato all'albero di uscita del moto (o secondario) ruota a velocità ridotta rispetto all'ingranaggio solare realizzando la riduzione. Per ottenere elevati rapporti di riduzione possono essere montati in serie più stadi ovvero più meccanismi di questo genere in serie.
I vantaggi nell'utilizzo di tali riduttori risiedono nel fatto che:

- In uno stadio di riduzione epicicloidale, rispetto ad una riduzione ad ingranaggi a parità di numero di denti fra pignone e corona si ottengono rapporti di riduzione più elevati;
- gli ingranaggi sono dotati di ingranamenti multipli contrariamente a quanto avviene in una riduzione ad ingranaggi paralleli dove l'ingranamento è singolo. Dunque a parità di coppia da trasmettere le sollecitazioni sui singoli denti saranno molto inferiori e quindi la riduzione epicicloidale necessita, a parità di forze da trasmettere, ingranaggi con dimensione più contenute.

Nella figura 3.27 viene proposto un confronto tra i meccanismi lineari classici e quelli epicicloidali.

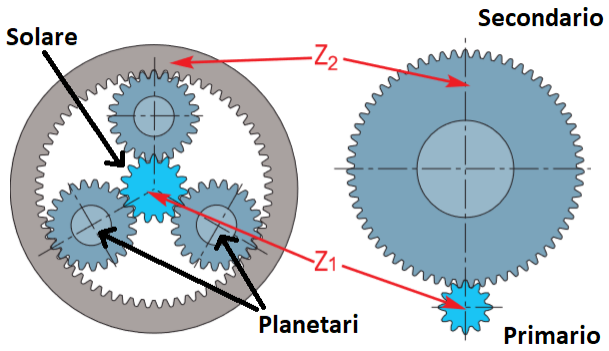

Figura 3.27 Riduttori epicicloidali e lineari

3.8.4 Driver motoriduttore

Il microcontrollore che deve gestire il motoriduttore non è in grado di erogare la potenza necessaria al suo funzionamento, per questo la sua azione si limita a pilotare attraverso alcune linee digitali una scheda predisposta.

Figura 3.28 Driver MD10C

La scheda o *driver* per motori *DC* è lo strumento *hardware* necessario a pilotare correttamente il motoriduttore. La scheda scelta è prodotta dall'azienda *Cytron technologies Mod. MD10C*, alimentata in questo caso da una tensione proveniente da un alimentatore stabilizzato che eroga una tensione pari a 12 V in corrente continua. La potenza erogata dipenderà dal carico richiesto dal motore. A tal proposito il *driver* è stato dimensionato in modo da poter garantire il funzionamento del motoriduttore a tutti i livelli possibili di carico e velocità, tale *driver* è infatti in grado di garantire una corrente di 13 A in maniera continua e resistere fino a picchi di 30 A per 10 secondi.

Il *driver* viene pilotato dal microcontrollore mediante due porte digitali, una di queste denominata *DIR* viene utilizzata per determinare il verso di rotazione del motore facendo assumere a quest'ultima il valore 0 o 1 logico tramite la funzione *digitalWrite()* di Arduino è possibile fare ruotare il motore in senso orario o antiorario. L'altra linea invece verrà pilotata con il metodo *PWM* per regolare la velocità di rotazione del motore tramite la funzione *analogWrite()* in grado di generare un'onda quadra ad ampiezza variabile.

3.8.5 Pulse Width Modulation (PWM)

Per poter regolare la velocità di rotazione di un motore elettrico in corrente continua, è necessario inviare al *driver* di alimentazione di quest'ultimo un segnale *PWM*.

Con *PWM* (*Pulse Width Modulation*) si intende una metodologia di modulazione a larghezza d'impulso variabile che permette di controllare l'assorbimento da parte di un utilizzatore emulando in questo modo diversi livelli di tensione.

Le possibilità di pilotaggio di una linea digitale sono solamente due: 0 e 1 logico. Nel caso in cui tale linea fosse utilizzata per alimentare un motore elettrico tali livelli corrisponderebbero al concetto di motore funzionante, perché alimentato alla tensione di utilizzo e motore non funzionante perché privo di tensione.

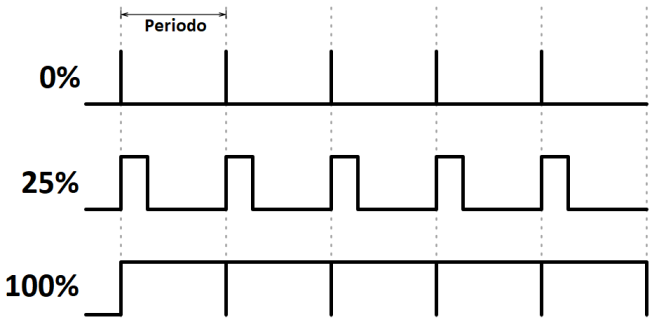

Figura 3.29 Pulse Width Modulation

Il *duty cycle* fornisce la possibilità di erogare all'utilizzatore diversi livelli di tensione di alimentazione medi che ne determinano diverse velocità di rotazione.
Utilizzando Arduino, è possibile ottenere tale segnale in alcune delle porte digitali (*D3*, *D5*, *D6*, *D9*, *D10* e *D11* contrassegnate con il simbolo ~) ottenendo così idealmente 256 velocità differenti mediante la funzione *analogWrite()* (risoluzione 8 *bit* da 0 a 255).
Matematicamente il *duty cycle* è il rapporto tra il tempo in cui la linea digitale così pilotata rimane a livello logico alto e la durata di un certo intervallo di tempo definito dalla frequenza del *timer* associato alla porta digitale, emulando livelli medi di tensione e quindi assorbimento diversi e quindi velocità di rotazione differenti.

3.8.6 CAN BUS shield

Con il termine *shield* si indica una scheda predisposta per essere utilizzata con un microcontrollore in particolare perché caratterizzata da dimensioni e configurazione delle porte che segue quella del microcontrollore per cui è dedicata. In questo caso per poter utilizzare con Arduino Uno il protocollo di comunicazione *CAN BUS* si è deciso di utilizzare lo *shield* commercializzato dalla *Sparkfun*, azienda che fornisce anche la "*library*" o *driver* necessari per poter utilizzare lo *shield*.

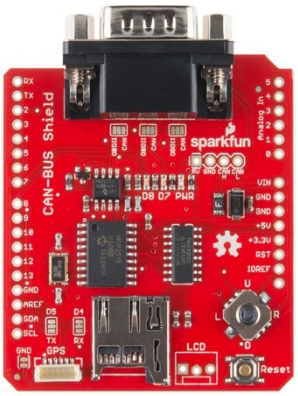

Figura 3.30 CAN BUS shield

Per poter gestire il protocollo *CAN BUS* la scheda presenta il chip *MCP_2515* con funzione di *CAN controller* che mette in comunicazione il microcontrollore e lo *shield* mediante protocollo *SPI* e il chip *MCP_2551* con funzione di *CAN transceiver* (trasmittente e ricevente) che gestisce il traffico da e per le linee *CAN HI* e *CAN LO* del *BUS* vero e proprio e interpreta la logica differenziale del segnale.
Lo *shield* presenta anche altre funzioni che però non vengono sfruttate come per esempio un lettore di memorie *SD*, un *joystick*, un interfaccia per un'antenna *GPS* consente di collegare un display *LCD* seriale.

3.8.7 Sensore ad ultrasuoni

Il sensore ad ultrasuoni è un dispositivo che è in grado di generare una serie di 8 impulsi sonori con una frequenza di 40 KHz inviando un segnale di livello logico alto con durata di 10 μs al *pin Trig* con un microcontrollore.
Questa serie di impulsi si propaga attraverso l'aria e il

Figura 3.31 Sensore ad ultrasuoni

sensore attende in ascolto il ritorno degli impulsi inviati sotto forma di onde riflesse. Il modulo produrrà poi in uscita un segnale di livello logico alto proporzionale al tempo di viaggio di queste onde e interpretabile dal microcontrollore con la funzione *pulseIn*().
La distanza viene calcolata prendendo come riferimento la metà del tempo di viaggio delle onde e moltiplicando questo tempo per la velocità di propagazione delle onde sonore nell'aria che è anche funzione della temperatura. La formula per il calcolo è la seguente:

$$Distanza_{ostacolo} = 331{,}4\ \frac{m}{s} + 0{,}62\ T° \quad [m] \quad [Eq.3.42]$$

Considerando una temperatura intermedia di funzionamento di 20°C il valore utilizzato nell'algoritmo di calcolo è 343 $\frac{m}{s}$.

3.8.8 Strumenti di input e output

Per la realizzazione dell'unità di controllo *master* sono stati necessari alcuni componenti utili a ricevere gli input dell'operatore e a fornire alcune informazioni necessarie per la calibrazione e il funzionamento del sistema.

3.8.8.1 Levetta analogica

Modulo che consente di leggere movimenti di una levetta lungo due assi. Il principio di funzionamento si basa sull'utilizzo di due canali distinti, canale x e y.
I movimenti lungo il singolo asse vengono rilevati per mezzo di un potenziometro resistivo da 10 KΩ che determina una variazione della tensione in uscita dal singolo canale in funzione della posizione della levetta. Conoscendo il valore del voltaggio in entrata (ovvero 5 V) è possibile ricavare la posizione della levetta analogica rispetto ai due assi di riferimento effettuando una lettura analogica del segnale proveniente dai due canali tramite la funzione *analogRead(),* disponibile per le porte analogiche di Arduino.

Figura 3.32 Levetta analogica

Di conseguenza, mediante una funzione di trasformazione, è possibile tradurre i movimenti di questa levetta in istruzioni utili per il software.
Premendo la levetta si può ottenere un *input* digitale equivalente alla pressione di un pulsante.

3.8.8.2 Pulsante

Modulo che consente di raccogliere *input* mediante la sua pressione. In questo caso la tensione in entrata quando il pulsante non è premuto, ovvero il contatto si trova in posizione aperta, esce inalterata dal *pin* dell'*output* generando un segnale per il microcontrollore di stato logico alto.

Figura 3.33 Modulo bottone

Quando il pulsante viene premuto il circuito viene chiuso a massa e l'*output* del modulo fornirà al microcontrollore un segnale nullo o 0 logico.

3.8.8.3 Schermo LCD

Per ottenere un interfaccia grafica utile a determinare le diverse modalità di funzionamento e regolazione è stato installato uno schermo a cristalli liquidi. Questo device è organizzato come una matrice composta da 80 elementi all'interno dei quali è possibile rappresentare numeri e caratteri.

Figura 3.34 Schermo LCD 20x4

Viene definito 20 X 4 per via delle dimensioni della matrice ovvero 20 colonne e 4 righe.
Lo schermo è dotato di un modulo che converte la metodologia nativa di comunicazione in parallelo con quella seriale tramite *BUS I^2C* che necessita di un numero inferiore di porte per funzionare.
Per riportare le informazioni necessarie basta puntare un elemento della matrice con la funzione *lcd.setCursor()* e con la funzione *lcd.print()* è possibile scrivere o riportare i valori delle variabili di interesse.
Per poter funzionare correttamente è necessario caricare nel microcontrollore le opportune *library*.

3.9 Componenti software

Il *software* che è stato scelto per realizzare lo script che gestisce le funzionalità del *cable robot* e che svolge la quasi totalità dei calcoli necessari è *MATLAB* (abbreviazione di *Matrix Laboratory*). Quest'ultimo è un ambiente per il calcolo numerico e l'analisi statistica scritto in *C*, che comprende anche l'omonimo linguaggio di programmazione creato dalla *MathWorks*.
MATLAB consente di manipolare matrici, visualizzare e analizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e lavorare insieme ad altri programmi e *hardware*.
E' stato scelto questo programma per la relativa semplicità d'uso e per le sue potenzialità di analisi e gestione dei dati rivelatasi utile oltre che allo svolgimento dei calcoli anche in fase di progettazione, dove è stata eseguita proficuamente l'analisi matematica per il dimensionamento preliminare. In prospettiva futura la scelta di utilizzare questo *software* consentirebbe anche l'implementazione di funzioni secondarie come per esempio quelle di raccolta dati per mezzo di immagini e la loro rielaborazione.
MATLAB fornisce inoltre la possibilità, grazie all'ausilio di appositi pacchetti *(Add-on)* dedicati ad Arduino, di configurare e utilizzare in maniera relativamente semplice tutte le componenti *hardware* interfacciabili con il microcontrollore oltre che ovviamente garantire la connettività di quest'ultimo con il *PC* mediante l'installazione di appositi *driver* di comunicazione sulla scheda per poter così gestirne la comunicazione seriale via *USB*.
Tale possibilità è stata adottata inizialmente nella fase di prototipazione ma è stata successivamente abbandonata perché riusciva a garantire tempi di esecuzione non compatibili con le esigenze di controllo *real-time*, escludendo di fatto la possibilità di effettuare un vero e proprio controllo distribuito accurato oltre che anche per il limite intrinseco della distanza raggiungibile mediante il protocollo seriale.
Per questi motivi si è deciso di trattare l'unico microcontrollore interfacciato con il *PC* come un oggetto seriale mentre tutti gli altri microcontrollori sono stati, per lo scopo, programmati con il linguaggio dell'ambiente di sviluppo di Arduino per renderli in grado di funzionare e svolgere i propri compiti in maniera parallela ed autonoma.
Il funzionamento del *cable robot* è gestito esclusivamente mediante la ricezione e l'invio di dati da parte del *master* esclusivamente in forma di *Byte* mediante *CAN BUS* sotto forma di istruzioni per gli *slave* con lo scopo di gestirne e coordinarne l'azione.

3.10 Protocolli di trasmissione dei dati

Per l'invio la ricezione di dati in forma binaria è necessario utilizzare un *BUS (Binary Unit System)* ovvero, in elettronica e informatica, un canale di comunicazione che permette a periferiche e componenti di un sistema elettronico di interfacciarsi tra loro scambiandosi informazioni attraverso la trasmissione di segnali all'interno di una rete.
Nella progettazione e realizzazione di sistemi automatici, specie quando si tratta di sistemi di controllo distribuito, la scelta del protocollo di comunicazione riveste un'importanza cruciale, al pari dell'algoritmo di controllo stesso. Il protocollo deve essere adeguato per poter permettere all'algoritmo di controllo di esprimere al massimo le sue potenzialità.
Per adeguatezza del protocollo si intende:

- la capacità di trasportare adeguati volumi di dati in un tempo più breve possibile *(bitrate)*;
- la distanza alla quale i dati possono essere veicolati senza incappare in errori;
- la robustezza di quest'ultimo, ovvero la capacità di trasportare i dati senza commettere errori.
  A questo proposito va fatto presente che spesso la velocità di trasmissione dei dati può rendere inferiore la distanza alla quale questi possono essere trasmessi correttamente a parità di protocollo, argomento che verrà approfondito successivamente.

La realizzazione del controllo distribuito ha reso necessario l'utilizzo di un protocollo robusto e caratterizzato da un'adeguata velocità di trasmissione. Allo scopo la scelta è ricaduta su diverse soluzioni a seconda del ruolo svolto dalla comunicazione:

- per il collegamento *PC → master* è stata adottata la soluzione della trasmissione via *USB*;
- per la comunicazione a breve raggio, ovvero quella che avviene tra i componenti di una stessa unità, per esigenze di progettazione e vincoli imposti dall'*hardware* è stato scelto il protocollo I^2C.
  Questo protocollo è veloce e affidabile ma caratterizzato da una limitata distanza raggiungibile (tale limite va precisato che è imposto dalle caratteristiche capacitive del *BUS,* il protocollo tollera fino a 400 pF di capacitanza, di solito raggiunta in qualche decina di centimetri a seconda della caratteristiche del cavo utilizzato);
- per la comunicazione a lunga distanza la scelta è ricaduta sul *CAN BUS*, protocollo molto robusto e veloce, nato e utilizzato come standard nel settore *automotive*.
  Il *CAN BUS* è stato scelto per la realizzazione della dorsale principale della rete,

ovvero, quella parte di comunicazione definibile come a lungo raggio che mette in comunicazione tra loro il *master* e gli *slave*.

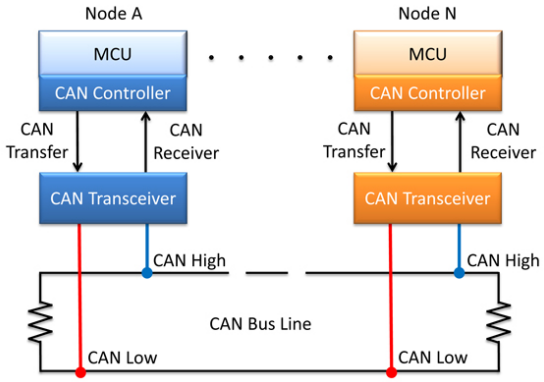

Figura 3.35 Struttura di una rete CAN BUS

Viene proposta di seguito la trattazione inerente ai protocolli di trasmissione e alla codifica utilizzata allo scopo di rendere più chiara la successiva sezione dove viene esposto nel dettaglio il funzionamento del prototipo.

3.10.1 Universal Serial Bus (USB)

La comunicazione seriale viene utilizzata per trasmettere dati tra dispositivi digitali. I dati, sotto forma di *bit*, vengono trasmessi uno dopo l'altro in successione attraverso la linea di comunicazione secondo la logica *TTL (transistor-transistor logic)*. Per la realizzazione del progetto è stato necessario utilizzare il protocollo *USB* (*Universal Serial Bus*) per mettere in comunicazione in maniera semplice il *PC* con il *master*, è necessario allo scopo unicamente regolare la velocità di comunicazione dei dispositivi sul medesimo livello di frequenza, in questo caso 115˙200 baud (bit/s), creando un oggetto seriale in *MATLAB* e inizializzando la comunicazione seriale mediante il comando *Serial.begin()* nello *sketch* di Arduino.
La scelta di questo sistema è dovuta alla sua perfetta compatibilità con le porte disponibili su *PC* e microcontrollore, quest'ultimo possiede infatti un interfaccia *USB* di tipo 1.0. Tale *standard*, dato che la distanza massima raggiungibile con esso va dai 3 ai 5 metri, è stato abbandonato per quanto riguarda la comunicazione *master-slave* anche se in linea di massima avrebbe avuto una portata sufficiente data la scala del progetto.

Questo abbandono è stato motivato dalla volontà di garantire la possibilità di ampliare in futuro la scala del progetto senza modificare l'architettura del *software* e perché in fase di *test* sono stati riscontrati problemi di sincronizzazione e tempi di esecuzione elevati per la difficoltà di gestire numerosi oggetti seriali da parte del *software MATLAB* specie quando si rende necessario l'utilizzo di *hub USB* come sarebbe stato necessario in questo caso data la presenza di *4 slave* e 1 *master*.

3.10.2 Controller Area Network BUS (CAN BUS)

Il *CAN BUS* è un protocollo di trasferimento di dati introdotto negli anni ottanta dalla *Robert Bosch GmbH* per collegare diverse unità di controllo elettronico *(ECU)* e si presta, date le sue caratteristiche, ad essere impiegato con elevata efficienza in sistemi di controllo distribuiti di tipo *real-time* garantendo un elevato livello di sicurezza e di integrità dei dati trasmessi. Inizialmente venne utilizzato in applicazioni automobilistiche per consentire la comunicazione fra i dispositivi elettronici intelligenti (ovvero in grado di interfacciarsi con il *CAN*, inviare messaggi, riceverli e filtrarli) montati su autoveicoli ma che successivamente, date le sue peculiarità, è stato adottato anche in molti settori dell'automazione.

Tipicamente il moderno settore automobilistico equipaggia i veicoli con diverse *ECU* utili a gestire i diversi sottosistemi presenti nel veicolo. Solitamente la *ECU* più importante e complicata è quella che gestisce i parametri relativi al funzionamento del motore, ma nell'automobile sono presenti numerose altre *ECU* che si occupano di gestire la trasmissione automatica, gli *airbag*, il sistema *ABS*, il *cruise control*, il servosterzo elettrico, i sistemi audio, la chiusura delle portiere e dei finestrini, la gestione del sistema di ricarica delle batterie dei veicoli ibridi, ecc. Alcuni di questi sottosistemi sono indipendenti e possono gestire autonomamente i dati raccolti dai sensori ed eventualmente agire su alcuni attuatori mentre altri, invece, hanno bisogno di informazioni provenienti da sensori o da altri sottosistemi per funzionare correttamente e per questo motivo la comunicazione tra diversi sottosistemi rappresenta un fattore essenziale che ha inoltre bisogno di avvenire in tempi ragionevolmente brevi e con elevati gradi di accuratezza.
Ogni singolo sottosistema può essere considerato quindi come un nodo facente parte di una complessiva rete di comunicazione (definito anche come ramo), collegato alla linea principale, o dorsale, costituita da un singolo canale bidirezionale che può essere di tipo differenziale o a cavo singolo e terra (meno usato del primo). Solitamente la dorsale viene realizzata utilizzando un doppino intrecciato, schermato o meno, a seconda della rumorosità elettrica e magnetica dell'ambiente in cui lavora.
I due trefoli intrecciati rappresentano i canali *CAN High* e *CAN Low* collegati tra loro agli estremi per mezzo di due resistenze da 120 Ω definite terminatori che hanno lo scopo di estinguere il segnale alle estremità.
Ogni nodo è composto da: un unità centrale di controllo (microcontrollore) che ha lo scopo di decidere cosa trasmettere e cosa fare dei dati in arrivo dal *BUS* alla quale possono essere collegati sensori e attuatori; un *controller CAN*, spesso integrato nel microcontrollore, che ha lo scopo di ricevere e immagazzinare i dati provenienti dal *BUS* fino a che un intero messaggio non è disponibile, oltre che quello di trasmettere i dati in

maniera seriale sul *BUS* quando esso è disponibile; un *transceiver*, componente utile a convertire il flusso di dati dalla logica del *CAN controller* a quella vera e propria del *BUS* e viceversa.
Ogni nodo è dunque capace di inviare e ricevere messaggi ma ciò non può avvenire simultaneamente e ogni messaggio deve attendere che la linea sia libera per essere trasmesso.
Questo protocollo è definito *message-based* perché non si può dire che i nodi siano dotati di un proprio indirizzo, a differenza dei protocolli *address-based* dove ogni nodo ha un indirizzo proprio.
Il messaggio *CAN*, definito come *frame*, è costituito principalmente da: un identificativo (*ID*) che oltre a renderlo riconoscibile, ne rappresenta anche la priorità di quest'ultimo; dal contenuto del messaggio stesso (tra 0 e 8 *byte* e fino a 64 *byte* con il protocollo *CAN FD*) e altri campi utili a capire la quantità di dati immessi, la correttezza del *frame* ricevuto e se il messaggio ricevuto rappresenta una richiesta di trasmissione di un messaggio in particolare.
Il sistema *CAN BUS* è nato dunque per andare incontro alle esigenze del settore automobilistico che nel tempo ha aumentato la complessità dei sistemi elettronici e il suo successo è dovuto dunque ai notevoli vantaggi tecnologici che offre e tra questi si possono principalmente individuare:

- tempi di risposta molto brevi che consentono di connettere un elevato numero di dispositivi mantenendo stringenti vincoli temporali e consentendo di implementare un controllo *real-time* di tipo distribuito;
- semplicità e flessibilità del cablaggio dato che il *CAN BUS* è implementato su un unico doppino intrecciato;
- capacità "*multi-master*", ovvero tutti i nodi della rete hanno la possibilità di trasmettere e più nodi della rete possono chiedere il canale trasmissivo contemporaneamente;
- i nodi non hanno un indirizzo che li identifichi univocamente e possono quindi essere aggiunti o rimossi senza dover riorganizzare totalmente il sistema. Tali nodi, inoltre, non essendo caratterizzati da indirizzi di rete "convenzionali", consentono ai messaggi di essere immessi nella rete in base a una criticità definita mediante un identificativo rappresentante la sua priorità.
  L'identificativo, inoltre, può essere utilizzato per decidere se processare i dati in arrivo o meno compiendo dunque funzione di filtro per i nodi che ricevono.
- la rilevazione degli errori e la richiesta di ritrasmissione viene gestita direttamente dall'*hardware* conferendo elevata affidabilità al protocollo. Il sistema ha

un'incredibile capacità di riconoscere gli errori e la probabilità che un messaggio sia corrotto e non riconosciuto come tale, è praticamente nulla.
E' stato calcolato che una rete basata su *CAN BUS* con velocità di trasmissione di 1 Mbit/s, con un'utilizzazione media del *BUS* del 50%, una lunghezza media dei messaggi di 80 *bit* e un tempo di lavorazione di 8 ore al giorno per 365 giorni all'anno, avrà un errore non rilevato ogni 1000 anni. Praticamente la rete non è soggetta ad errori per tutta la durata della sua vita e questo rappresenta il maggior punto di forza di questo *BUS*; (Galanti, materiale *CAN BUS*)

- ciascun nodo è in grado di rilevare il proprio malfunzionamento e di autoescludersi dal *BUS* se questo è permanente, impedendo che un solo nodo metta in crisi l'intero sistema;
- tale *standard* ha un elevato grado di maturità e la larga diffusione del protocollo negli ultimi anni ha determinato un'ampia disponibilità di chip ricetrasmettitori e di microcontrollori che integrano porte *CAN*, *tools* di sviluppo e inoltre ha portato ad una diminuzione del costo di questi sistemi.

Queste caratteristiche hanno concesso a questo protocollo di comunicazione di diventare uno *standard* oltre che nel settore dell'automobile anche in ambito industriale e sono in molti a vedere in tale sistema lo standard dominante nelle reti industriali del prossimo futuro.

3.10.2.1 Trasmissione dei dati

Il principio di funzionamento che regola la trasmissione dei dati si basa sulla priorità dei messaggi, dove messaggi a priorità più elevata, ovvero quelli caratterizzati da un *ID* con valore minore, hanno la precedenza ad occupare il *BUS* e ad essere trasmessi per primi. Il sistema di trasmissione utilizza un metodo per gestire le collisioni (ovvero la contesa dei diversi messaggi per ottenere la precedenza di trasmissione) basato su *bit* definiti come dominanti (0 logici) o recessivi (1 logici).
Se un nodo trasmette un *bit* dominante e un altro un *bit* recessivo, allora il *bit* dominante sarà quello che fra i due occuperà effettivamente il canale perché definito come di priorità più elevata. Per esempio, considerando un *ID CAN* di 11 *bit* e solo 2 nodi caratterizzati rispettivamente da *ID* 15 e 16 che in binario a 11 *bit* sono rappresentabili come:

000000**0**1111 (15)

000000**1**0000 (16)

Se questi due nodi iniziassero a trasmettere nello stesso momento, fino alla trasmissione del sesto *bit* (partendo da sinistra) la trasmissione avviene senza collisioni e dunque senza fare intervenire il sistema di arbitraggio.
Con la trasmissione del settimo *bit* (evidenziato sopra in grassetto) si nota come l'*ID* del nodo 15 presenta uno 0 che è considerato dominante e quindi tale nodo proseguirà senza ritardo nella trasmissione del resto del *frame* mentre il nodo con *ID* 16, perdendo la contesa, interrompe la trasmissione del *frame* che aveva iniziato a trasmettere. Lo stesso procedimento avviene con arbitraggi tra più di due nodi impegnati nella comunicazione, il concetto viene riproposto con una rappresentazione grafica in figura 3.35.

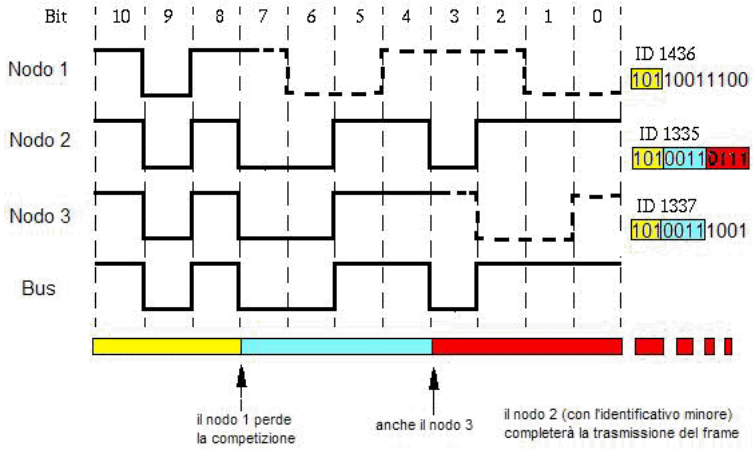

Figura 3.35 Funzionamento del sistema di arbitraggio

Avendo introdotto il concetto di *bit* dominante e recessivo, non resta che illustrare come vengano determinati i diversi livelli logici sulla linea *CAN High* e *CAN Low* per determinare il livello di stato del *BUS* a livello di singolo *bit* riportando un esempio (figura 3.36).

Al fine di assicurare un'elevata immunità ai disturbi, la singola lettura viene eseguita sulla differenza di potenziale esistente tra i conduttori *High* e *Low* eliminando di conseguenza gli eventuali componenti di disturbo che influenzerebbero in maniera molto simile entrambe i conduttori. Per tale ragione le linee di trasmissione vengono tenute ad una tensione comune di 2,5 *V* (1 logico recessivo $CAN\ High - CAN\ Low = 0$) e il livello logico di un *bit* dominante risulterà invece di 3,5 *V* nella linea *High* e di 1,5 *V* nella linea *Low* ottenendo così una differenza di potenziale pari a 2 V tra le due linee (0 logico dominante).

La velocità di trasmissione (*bitrate*) potenzialmente raggiungibile dal sistema di comunicazione è funzione della lunghezza complessiva della dorsale del *BUS*, di conseguenza maggiore sarà la sua lunghezza minore sarà la velocità massima di trasmissione supportabile. Per esempio tale velocità può raggiungere 1 Mbit/s quando la

dorsale ha una lunghezza inferiore a 40 m, per lunghezze superiori la velocità dovrà essere ridotta come mostrato in figura 3.37.

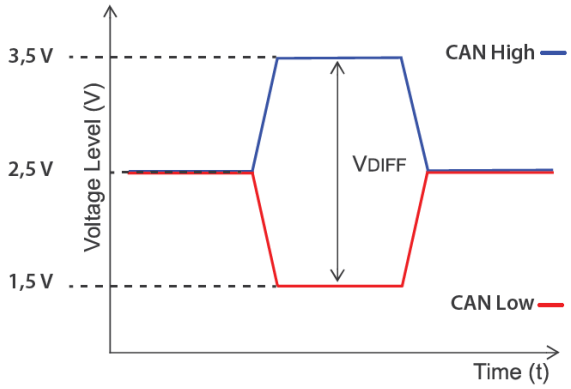

Figura 3.36 Trasmissione di bit dominanti e recessivi

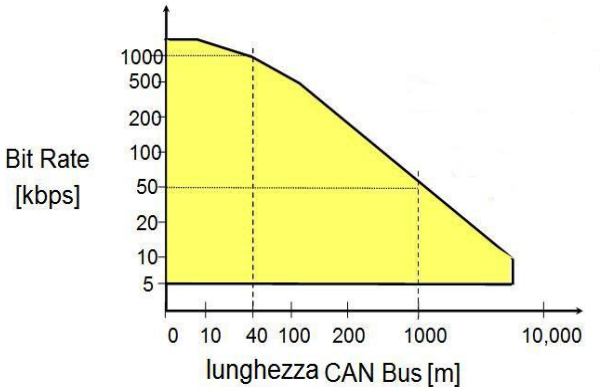

Figura 3.37 Bit rate in funzione della lunghezza della dorsale CAN

3.10.2.2 Struttura del messaggio

I dati che vengono inviati sul *CAN BUS* devono rispettare la formattazione corretta tipica del protocollo di comunicazione, utile a rendere interpretabile il messaggio da parte dei dispositivi interconnessi.
Il frame ha dunque una struttura propria e delle regole precise di composizione. Sulla base di questi vincoli possono essere individuati diversi tipi di *frame*, ognuno dei quali composto da alcuni campi.
Come primo elemento, prima di introdurre i *frame* verrà trattato l'*ID* che data la sua importanza e la sua presenza in tutti i tipi di *frame* è necessario affrontare il tema in maniera separata.

3.10.2.3 Identificativo

L'*ID* è un campo che ha lo scopo di rappresentare univocamente un particolare messaggio per consentirne l'identificazione certa e garantire il corretto funzionamento del *BUS*. Esso rappresenta la priorità di trasmissione del messaggio. La forma assunta dall'*ID* determina e caratterizza il tipo di *frame*. Tale campo può assumere due formati denominati rispettivamente *standard* (*CAN "base" frame*) ed *extended* (*CAN "extended" frame*).
La differenza tra i due frame sta nella dimensione della variabile destinata ad esso.
Per il formato standard sono previsti 11 *bit* che consentono 2^{11}(2˙048) tipi di messaggi diversi gestibili dal *BUS* mente per il formato *extended* 29 *bit* (11 *bit* di base più un estensione di 18 *bit*) ovvero 2^{29}(536˙870˙912) messaggi diversi.
La distinzione tra i due tipi di *frame* viene fatta con l'invio del *bit IDE*, campo utile a comunicare al ricevente la formattazione del *frame* e garantire così la corretta

interpretazione. Questo campo è trasmesso come *bit* dominante (0 logico) nel caso del *frame standard* e come *bit* recessivo (1 logico) nel caso del *frame extended*. I controllori *CAN* che supportano i *frame extended* sono in grado di utilizzare anche i *frame* a 11 *bit*, mentre quelli che supportano i *frame standard* sono in grado di gestire solamente questi ultimi.
Ogni *frame* inizia con un *bit* denominato *start of frame* (*SOF*) che apre il messaggio.
Possono essere distinti 4 tipi di *frame*:

- *data frame*, ovvero messaggi contenenti dati che devono essere inviati da parte del nodo che trasmette;
- *remote frame*, tipologia di messaggio caratterizzata da una richiesta di invio di dati identificati a loro volta da un certo *ID*;
- *error frame*, quando un nodo trasmette un messaggio di errore;
- *overload frame*, necessario ad introdurre un ritardo nella linea.

3.10.2.4 Data frame

Il *data frame* è la forma assunta dal messaggio quando lo scopo è l'invio di dati.
Può assumere il formato *standard* o *extended*. Tale frame viene trasmesso da un nodo nel momento in cui la linea è disponibile ad effettuare un trasferimento e viene ricevuto da tutti gli altri nodi che si comportano come ricevitori.
Successivamente ciascun nodo deciderà se ritenere utili o meno le informazioni ricevute mediante l'utilizzo di filtri. La struttura generale del data frame viene descritta in figura 3.38.

Il *data frame standard* ha una struttura può essere suddivisa in :

- *start of frame (SOF)* (1 *bit*): rappresenta l'inizio del messaggio;
- *identifier field (ID)* (11 *bit*): identificativo univoco del messaggio
- *remote request trasmission (RTR)* (1 *bit*): tale bit assume valore dominante (0 logico) nel *data frame*. *Nell'extended frame* viene definito come *SRR* (*substitute remote request* e assume valore recessivo);
- *identifier extension bit (IDE)* (1 *bit*): questo campo assume valore dominante nello *standard frame* con *ID* da 11 *bit*, mentre assume valore recessivo con *ID* da 29 *bit* e a cui segue il campo dell'indirizzo esteso che prevede i 18 *bit* in più rispetto allo *standard frame*;
- *reserved bit (1 bit)*: questo *bit* deve essere dominante ma viene accettato anche se recessivo, viene utilizzato come *bit* libero per futuri sviluppi del sistema;

- *data length code (DLC)* (4 *bit*): questo campo rappresenta il numero di *bit* contenuto nel campo dedicato ai dati da trasmettere veri e propri, solitamente assume valori tra 0 e 8;
- *data field* (0-64 *bit*): tale campo è destinato a contenere i dati oggetto del messaggio;
- *cyclic redundancy check (CRC)* (15 *bit*): è un metodo per il calcolo di somme di controllo (*checksum*), algoritmo utile ad individuare errori casuali nella trasmissione di dati dovuti ad interferenze, rumore di linea o distorsioni;
- *CRC delimiter* (1 *bit*): tale *bit* delimita il campo *CRC* e deve essere recessivo;
- *ACK slot* (1 *bit*): il nodo che ha trasmesso emette in questo campo un *bit* recessivo che gli altri nodi sovrascrivono apponendo un *bit* dominante. Questo metodo è utile a riconoscere la ricezione di un *frame CAN* valido. Ogni nodo che riceve il messaggio senza rilevare un errore trasmette un *bit* dominante nello *slot ACK* sostituendo quindi il livello recessivo del trasmettitore. Se il nodo che trasmette rileva un livello recessivo in questo campo vuol dire che nessun nodo ha ricevuto un *frame* valido. Il sistema in questo modo non riesce però a capire se, tra i numerosi possibili nodi, il messaggio non sia arrivato correttamente ad uno di essi o comunque basta anche che un solo nodo riceva correttamente il *frame* per fare mascherare con il suo *bit* dominante un eventuale errore di ricezione di un altro nodo.

  Spesso tale metodo porta alla ritrasmissione dei *frame* che non sono stati ricevuti correttamente più volte portando di conseguenza il *BUS* ad entrare in uno stato di errore passivo;
- *ACK Delimiter* (1 *bit*): bit che serve a delimitare il campo *ACK* e deve assumere valore recessivo;
- *end of frame (EOF)* (7 *bit*): tale campo identifica il termine del messaggio, esso assume valore recessivo per 7 *bit* consecutivi;
- *intermission* (3 *bit*): Fa parte dello spazio *interframe* e precede lo stato di *BUS* libero che assume valore recessivo. Questo campo assume valore recessivo per almeno 3 *bit* consecutivi. Dopo tale campo se compare un *bit* dominante rappresenta sicuramente lo *SOF* del messaggio seguente dato che i *frame* di *overload* e quelli di errore non sono preceduti da uno spazio di *interframe*.
  Tale campo è utile anche a sospendere la trasmissione a causa di uno stato di errore passivo che potrebbe essere stato trasmesso nel messaggio precedente. precedente.

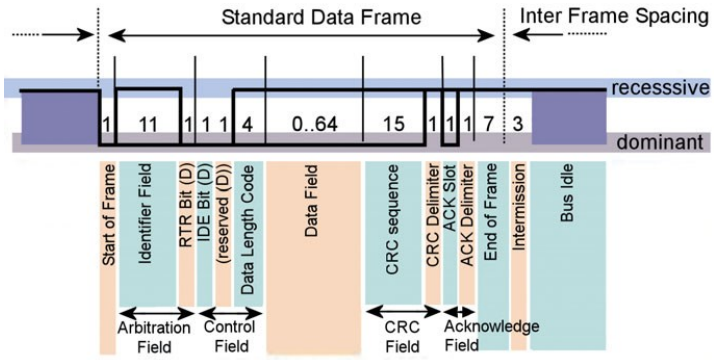

Figura 3.38 Struttura del data frame

3.10.2.5 Remote frame

In genere la trasmissione dei dati avviene, come visto in precedenza per il *data frame*, su base autonoma come episodio o ad intervalli regolari da parte di un nodo come per esempio un sensore. Tale sensore dunque se trova la linea disponibile e non incombono messaggi più urgenti riesce ad immettere i propri dati nel *BUS*.
Il *remote frame* come si può notare in figura 3.39, differisce dal *data frame* in alcuni suoi campi e nello scopo per cui viene inviato. Tale tipologia di messaggio funziona in maniera diversa perché esso viene emesso da un nodo che richiede un invio di particolari dati contenuti in un preciso messaggio identificato dal suo ID. Questa tipologia dunque è un interrogazione eseguita da un nodo a cui segue l'invio di quel particolare pacchetto di dati richiesto da parte del nodo interrogato.

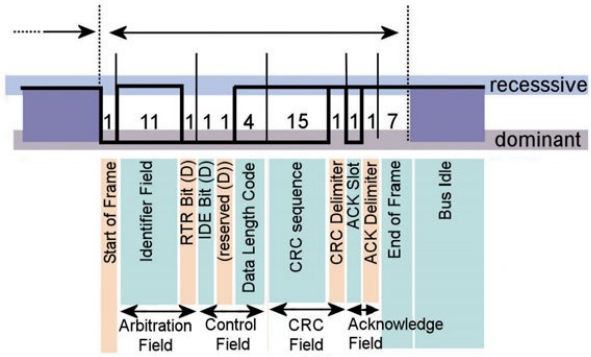

Figura 3.39 Struttura del remote frame

Le differenze che si possono trovare a livello di campi che compongono il messaggio tra il remote frame e il data frame sono:

- il campo *RTR* nel *remote frame* assume valore recessivo (per questo motivo nel caso in cui la trasmissione della richiesta di dati per mezzo del *remote frame* si scontrasse con l'invio del *data frame* contenente tali dati, il *data frame* vincerebbe

la contesa dato che presenta un *bit* dominante in questa posizione e ciò garantisce un risparmio di tempo nella trasmissione) ;

- non è presente il campo destinato a contenere i dati;
- il campo *DLC* non indica la quantità di dati presenti nel messaggio ma quella richiesta;

Gli altri campi presenti nel frame sono comuni e hanno funzioni uguali a quelle possedute nel *data frame*.

3.10.2.6 Error frame

Il *frame* che indica la presenza di errori nella trasmissione di un messaggio è l'*error frame*. Questo tipo di *frame* viene inviato da ogni nodo che rileva un errore nella trasmissione. Gli errori a cui il protocollo si riferisce sono:

- *Bit error:* viene monitorata la corrispondenza tra *bit* emesso e tra *bit* sul *BUS* da parte del nodo emettitore, escludendo l'*arbitration field* e il campo *ACK*;
- *Stuff error:* si verifica tra *SOF* e *CRC delimiter* quando viene violata la regola del *bit stuffing,* ovvero si hanno 6 *bit* consecutivi con stesso valore;
- *Form error*: errore di forma che si può verificare se si registrano *bit* dominanti nei campi *EOF*, *ACK delimiter* e *CRC delimiter*. Unica eccezione si applica quando riguarda l'ultimo *bit* dell'*EOF,* situazione interpretata come *overload frame*;
- *Ack error:* se assume valore dominante;

Questi tipi di errori provocano la trasmissione immediata del *frame* di errore da parte del nodo, mentre nel caso di *CRC error* il messaggio di errore viene inviato dopo l'*ACK delimiter*. Tale frame di fatto viola le regole di trasmissione (6 *bit* consecutivi senza *bit-stuffing*) e dunque l'errore viene così segnalato agli altri nodi. Dato che è sufficiente che un solo nodo segnali un errore per avere la trasmissione, il protocollo *CAN* prevede un sistema di confinamento degli errori che prevede che ciascun nodo monitori il proprio *status*, autoescludendosi in caso di tasso di errore elevato.
Tale tipologia di *frame* non interviene quando il *BUS* si trova in stato libero come avviene per gli altri frame visti fino ad adesso ma interviene all'interno del *frame* di dati o di richiesta.
Il procedimento di individuazione degli errori (*error-detecting*) è un punto fondamentale utile a conferire al protocollo *CAN BUS* elevati livelli di affidabilità. Questa procedura è suddivisa in 5 fasi:

- il controller *CAN* di un nodo rileva un errore (di ricezione o trasmissione);
- il nodo trasmette immediatamente nella rete un *error frame*;

- il messaggio contenente l'errore viene così ignorato da tutti i nodi della rete;
- il *CAN controller* aggiorna il suo stato;
- il messaggio viene ritrasmesso normalmente (secondo le priorità imposte dall'*ID*, quindi competendo per l'occupazione della linea).

Il protocollo *CAN* definisce 5 differenti tipi di errore, di cui 3 a livello di *bit* e 2 a livello di messaggio, rilevati rispettivamente attraverso le seguenti tecniche:

- monitoraggio: ciascun nodo confronta i *bit* che invia con quelli effettivamente presenti sul *BUS* e in caso di discordanza si ha un *bit error*;
- *bit-stuffing*: è una tecnica che consiste nel trasmettere dopo ogni sequenza di 5 *bit* uguali un sesto *bit* con polarità inversa, che viene automaticamente ignorato dai nodi ricevitori;
- controllo ciclico di ridondanza *(CRC)*: ciascun nodo ricevitore calcola la sequenza *CRC* corrispondente al messaggio ricevuto e la confronta con quella che il trasmettitore ha accodato al messaggio stesso;
- Controllo dei *frame*: se il ricevitore rileva che non sono state rispettate le 5 possibili strutture dei *frame CAN*;
- Invio dell'*acknowledgement bit (ACK)*: ciascun nodo che riceve correttamente un *data frame* o un *remote frame* è tenuto a inviare un bit dominante nel *bit-time* del campo *ACK* di questo *frame*.

Esistono due tipi di *error flags*:

- *active error flag*: campo che prevede l'invio di 6 *bit* dominanti da parte di un nodo che rileva un errore di trasmissione nella rete;
- *passive error flag*: campo composto da 6 *bit* recessivi trasmessi da un nodo *error passive* che rileva la presenza di un *frame* di errore attivo sul *BUS* trasmettendo di fatto un *frame* di 14 *bit* recessivi che non hanno effetto sugli altri nodi della rete;

L'*active error frame*, come si può intuire dalla figura 3.40, è composto da due campi principali:

- il primo campo prevede 6-12 *bit* di durata. La prima parte definita *active error flag* prevede che il nodo rilevante un errore nella trasmissione invii sul *BUS* 6 bit dominanti consecutivamente. La seconda parte del campo può prevedere altri *bit* (0-6 bit) come possibili sovrapposizione *(echo)* di *error flag* provenienti anche da altri nodi;

- il secondo campo è destinato all'*error delimiter*, utile a determinare la fine dell'errore e a ripristinare la linea. Questo campo è rappresentato con 8 *bit* recessivi consecutivi.

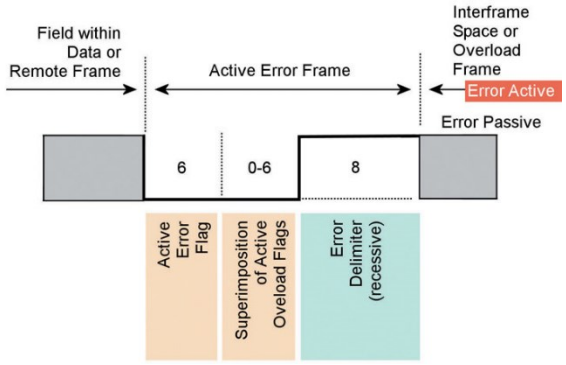

Figura 3.40 Struttura dell'error frame

Per quanto riguarda l'esclusione di un nodo malfunzionante (*fault confinement*), la rilevazione di un errore locale e la sua seguente globalizzazione non ha solo lo scopo di scartare il messaggio trasmesso per poter dare il via ad una nuova trasmissione ma è anche quello di poter andare a modificare alcuni registri di cui il *CAN* controller è dotato che funzionano come dei contatori dedicati a monitorare gli errori e denominati rispettivamente *transmit error counter (TEC)* e *receive error counter (REC)*.
Se il conteggio di *TEC* o quello di *REC* è inferiore a 128, in caso di errore viene trasmesso sul *BUS* un frame di errore attivo, mentre, quando invece il conteggio dei due contatori è maggiore di 127 e inferiore a 255 verrà trasmesso un frame di errore passivo e infine nel caso in cui i contatori superassero il valore di 255 il nodo in questione entra in modalità "*BUS off state*" dove è previsto che non possa più trasmettere.

3.10.2.7 Overload frame

L'*overload frame* ha una struttura simile a quella dell'*error frame* con la differenza che l'*overload* può intervenire solo all'interno dello spazio *interframe*, ovvero subito dopo la fine della trasmissione di un *frame*. In maniera uguale all'*error frame* questo è composto da due campi:

- *overload flag* che consiste di 6 bit dominanti consecutivi, La forma complessiva è come quella di un *active error flag*, che azzera i campi intervalli. Conseguentemente, anche gli altri nodi della rete rilevano una condizione di *overload* e trasmettono un *overload flag*;
- *overload delimiter* che prevede 8 *bit* recessivi consecutivi.

Lo scopo di questo *frame* è quello di ritardare l'istante in cui il *BUS* ritornerà libero a ricevere traffico dati.

Esistono due condizioni che possono portare alla trasmissione di un *overload flag*:

- il nodo per qualche motivo non è in grado di riabilitarsi in tempo per una nuova ricezione dopo che la precedente si è conclusa;
- la presenza di un *bit* dominante nel campo *intermission* presente nei *data frame*. In questo caso il *frame* inizia il *bit* successivo al *bit* dominante rilevato.

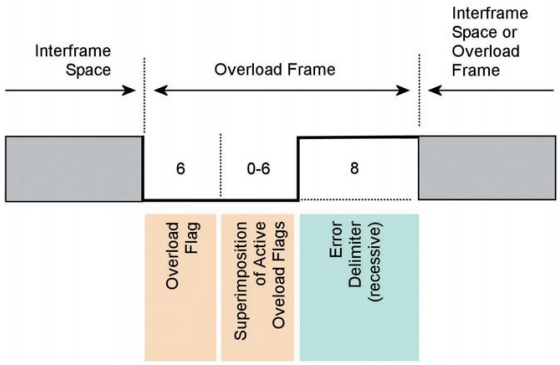

Figura 3.41 Struttura dell'overload frame

3.10.2.8 Bit-stuffing

Pur non essendo un *frame*, data la sua importanza, viene dedicato un paragrafo a questa metodologia utilizzata dal *CAN* per assicurare il mantenimento della sincronizzazione. A questo scopo viene inserito un *bit* di polarità opposta ogni volta che sono presenti 5 *bit* consecutivi con identica polarità. Tale pratica è necessaria per il tipo di codifica *Non Return to Zero (NRZ)* utilizzato nel protocollo: la linea viene pilotata con un valore di tensione alto per codificare il livello logico 1 e rimane in questo stato per un tempo pari al *bit-time* (durata temporale di un *bit*) moltiplicato per il numero di bit consecutivi in cui questo rimane allo stesso stato. Di conseguenza il segnale presenta fronti di salita e discesa che non sono in corrispondenza di ogni singolo bit trasmesso ma solo ogni volta che cambia lo stato del *BUS*. Per questo motivo per mantenere la sincronizzazione si utilizza il metodo del *bit-stuffing* che tende a non fare mantenere lo stesso stato alla linea per più di 6 *bit* consecutivi.

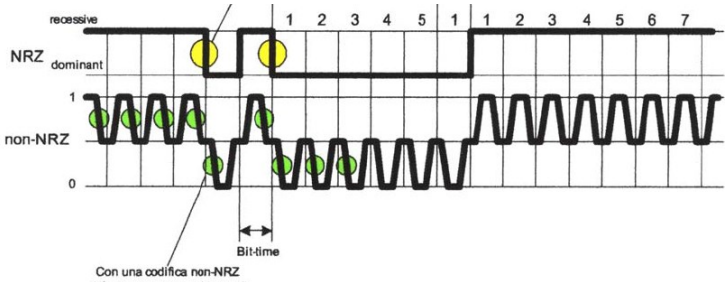

Figura 3.42 Codifica NRZ e non-NRZ

Il procedimento di *bit-stuffing* è applicato a tutti i campi del frame ad esclusione del *CRC delimiter*, *ACK* ed *EOF* perché hanno una dimensione fissa e non vengono per questo sottoposti al processo, i nodi che ricevono il *frame* svolgeranno un'azione di decodifica eliminando i *bit* aggiunti da questo processo per interpretarlo correttamente.
Nei campi in cui il metodo viene applicato 6 *bit* consecutivi con uguale polarità sono considerati un errore e un *active error flag* potrebbe essere inviato da un nodo. Il *bit-stuffing* può però aumentare dunque il volume di dati da trasmettere e presenta un effetto collaterale dovuto al fatto che se si possono verificare degli errori di riempimento dovuti al *bit-stuffing* che portano inevitabilmente a errori di interpretazione da parte del nodo ricevente durante l'operazione di eliminazione dei *bit* aggiunti. Questo problema è stato risolto con il protocollo *CAN FD* introducendo all'interno del frame oltre ai *bit* di *stuffing* un contatore che riporta il numero dei *bit* inseriti nel *frame con questo scopo*.

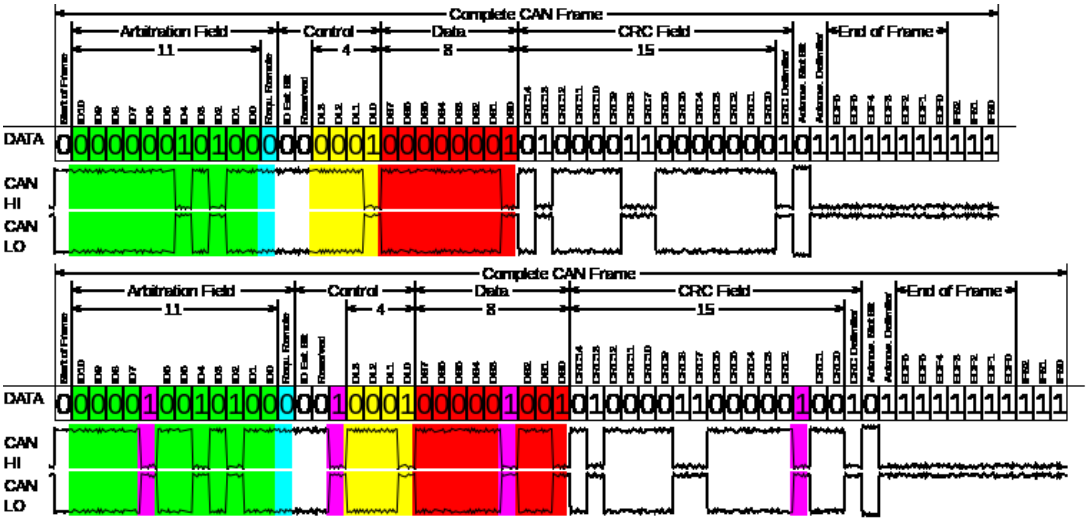

Figura 3.43 Frame CAN prima (sopra) e dopo (sotto) il bit-stuffing

3.10.3 *Inter Integrated Circuit BUS* (*I²C*)

Il *BUS I²C* è stato selezionato come metodo di trasferimento secondario con lo scopo di mettere in comunicazione:

- all'interno dello stesso nodo (o *slave*) il microcontrollore principale interfacciato con il *CAN BUS* con quello secondario utilizzato come contatore;
- i due microcontrollori utilizzati a livello di *master* rispettivamente interfacciati uno con il *CAN BUS* e l'altro per mezzo della comunicazione seriale con il *PC*;
- il microcontrollore presente nel *master* che dialoga con il *PC* con lo schermo *LCD* che è in grado di ricevere le informazioni per mezzo di tale *BUS*.

Questo protocollo è stato scelto per la sua semplicità di utilizzo, la perfetta compatibilità con i microcontrollori Arduino e per la breve distanza che, nel progetto, intercorre tra gli elementi sopra citati.
Questo protocollo si è rivelato utile per il fatto di essere gestito da un microcontrollore sotto forma di *master* e altri microcontrollori o strumenti definibili come *slave*. Il *master* in questo caso può inviare dati o richieste di invio di dati a un preciso elemento del *BUS* che partecipa come *slave*.
Il protocollo *I²C* è uno *standard* ideato dalla *Philips* per superare le difficoltà inerenti alla sincronizzazione e all'utilizzo dei *BUS* paralleli per la comunicazione tra un'unità di controllo e le varie periferiche.
La prima versione del bus *I²C* permette di trasmettere fino a 100 Kbit/s *(Standard mode)*. Questa velocità è stata portata a 400 Kbit/s *(Fast speed mode)* nelle modifiche apportate nel 1992 e dal 1998 la velocità è stata portata fino a 3,4 Mbit/s *(High speed mode)*.
L'*I²C* è un *BUS* seriale che necessita di sole due linee denominate rispettivamente *SDA (Serial Data)* e *SCL (Serial Clock)* più la linea di massa comune. La linea *SDA* è utilizzata per il transito dei dati in formato 8 *bit*, mentre la linea *SCL* è utilizzata per trasmettere il segnale di *clock* necessario per la sincronizzazione della trasmissione.
Questo protocollo ha bisogno della *library Wire* per essere utilizzato su Arduino ed è stato utilizzato.

3.10.3.1 Criticità

Le due linee che compongono il *BUS* sono implementate per mezzo di uscite *open drain* o *open collector*, questo significa che quando le linee *SDA* e *SCL* non sono utilizzate, si trovano a livello logico alto.

Questa caratteristica rende però necessaria una resistenza di *pull-up* per ogni linea, ovvero di una resistenza collegata tra la singola linea e +Vcc, come riportato in figura 3.44.
Valori tipici per le resistenze di *pull-up* sono compresi tra 2 KΩ e 10 KΩ. Il primo valore è utilizzato solitamente per *BUS* fino a 400 Kbit/s mentre il secondo per velocità fino a 100 Kbit. Il valore corretto comunque dipende, oltre che dalla frequenza di trasmissione, anche dalla capacità totale di linea.

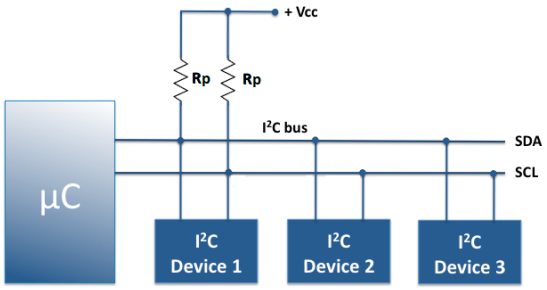

Figura 3.44 Bus I²C

Questo protocollo è definito come *address-based* e permette di connettere più periferiche su un unico *BUS* univocamente identificabili con il loro *ID*, permettendo la comunicazione tra due soli dispositivi alla volta.
L'indirizzo del dispositivo può essere fissato dal produttore in sede di fabbricazione (specie negli *slave* passivi come per esempio alcuni sensori o altre periferiche) o parzialmente fissato dal progettista.
Tale indirizzo è costituito da 7 *bit* (128 indirizzi) nelle versioni *standard* o da 10 *bit* (1˙024 indirizzi) nelle versioni estese (in realtà non tutti gli indirizzi possono essere utilizzati, alcuni sono riservati come quelli che vanno da 0 a 7).
Alla comunicazione partecipano un *master* e uno *slave*. Il protocollo è definito *multi-master* perché consente la presenza di più *master* contemporaneamente ma la comunicazione in ogni caso avviene sempre tra un *master,* che ha il compito di iniziare e terminare la comunicazione, e uno *slave* che si comporta da ricevitore o da trasmettitore quando interrogato. Dato che questo protocollo viene utilizzato molto non solo per la comunicazione tra microcontrollori ma anche per connettere microcontrollori con periferiche e altri dispositivi passivi, come per esempio schermi *LCD*, va precisato che non tutti i dispositivi possono ricoprire il ruolo di *master*; per esempio una memoria per l'immagazzinamento di dati dotata di interfaccia *I²C* non può comportarsi da *master*, al contrario di un microcontrollore.
Un vincolo stringente di questa tipologia di protocollo è imposto dalla capacità totale della linea che deve essere limitata a non più di 400 pF.
Il problema della capacità di linea è legato al tempo di salita con cui si riesce ad ottenere una variazione dal livello logico da basso ad alto definito come *rise-time*.

Il valore di questa capacità dipende dal numero di dispositivi e dalla lunghezza del *BUS* stesso. Potenzialmente, dal momento che una linea tipicamente introduce una capacità parassita di circa 80 pF/m, potrà essere lunga fino a 5 *m* senza contare la capacità introdotta dalla presenza di eventuali nodi sulla linea che ridurrebbero ulteriormente la distanza raggiungibile.
Abbassando la velocità di *clock* del *BUS* (trasmesso sulla linea *SCL*) si può aumentare la lunghezza della linea dato che il *rise-time* sarebbe meno stringente perché con frequenza minore ci sarebbe più tempo per consentire alla linea di tornare allo stato alto.
Altre soluzioni per evitare di rallentare la frequenza di *clock* prevedono lo spezzamento del *BUS*, in due o più parti, per mezzo di ripetitori che consentirebbero di avere 400 pF per ogni segmento del *BUS*.

3.10.3.2 Trasferimento dei dati

Il processo di trasferimento dei dati può essere schematizzato in 8 fasi principali:

- Il *master* controlla se le linee *SDA* ed *SCL* non siano già attive, ovvero siano poste ambedue a livello alto. Tale controllo ha una durata superiore al massimo periodo di trasmissione consentito da parte dell'*hardware*.
- Se il *BUS* risulta libero, il *master* invia la sequenza di *start* che serve a fare capire alle altre periferiche che il *BUS* è ora occupato. Tale sequenza consiste nel portare la linea *SDA* a livello basso quando *SCL* è a livello alto, come si può vedere in figura 3.45.

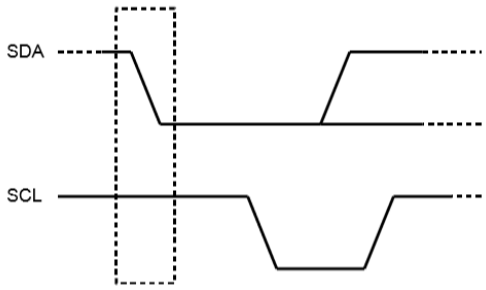

Figura 3.45 Sequenza di start sul BUS I²C

- Dopo l'invio della sequenza di *start* il *BUS* è considerato occupato e le altre periferiche si metteranno allora in ascolto per comprendere con chi il *master* ha intenzione di comunicare.
- Il *master* provvede poi all'invio del segnale di sincronizzazione sulla linea *SCL*, che sarà rappresentato da un onda quadra, non necessariamente periodica. A differenza della sequenza di *start* e di *stop* la linea *SDA* ha valore valido solo se la linea *SCL* è a livello basso. Non sono ammesse quindi transizioni di livello della linea *SDA*

durante il livello alto della linea *SCL*, se non da parte del *master* per inviare un nuovo *start* o uno *stop*.

- Il *master* compone l'indirizzo della periferica con la quale vuole parlare. Nel caso dell'indirizzo a 7 *bit*, questi vengono inviati sul *BUS* dal *bit* più significativo al *bit* meno significativo.
  In coda a questo indirizzo è previsto un *bit* per segnalare se il *master* voglia avviare con lo *slave* una comunicazione di scrittura *(W)* quando assume valore 0 o di lettura *(R)* nel caso in cui tale *bit* assume il valore 1. Il formato del byte (8 *bit*) che viene a questo scopo inviato è riportato in figura 3.46.

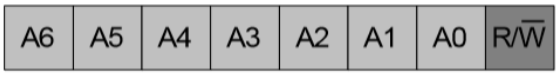

Figura 3.46 Struttura indirizzo a 7 bit I²C

Nel caso in cui l'indirizzo assuma il formato a 10 *bit* è necessario inviare un secondo *byte*. Per mantenere la compatibilità con il formato a 7 *bit* è però necessario un metodo che permetta alle periferiche in ricezione di capire che il primo *byte* non rappresenta in realtà il formato d'indirizzo a 7 *bit*. Questo è stato ottenuto proibendo alcuni indirizzi nel formato a 7 *bit*.
Sotto, in figura 3.47 viene presentato il formato dell'indirizzo a 10 *bit* composto da 2 *byte*.
A tale scopo i *bit* dell'indirizzo vengono inviati dal *bit* più significativo quello meno significativo.

Figura 3.47 Struttura indirizzo a 10 bit I²C

- Il *master* attende la risposta da parte della periferica che nella chiamata ha riconosciuto il suo indirizzo. L'invio dell'indirizzo a 7 bit e della modalità del colloquio (lettura/scrittura), avviene grazie ad otto transizioni, da livello alto a basso della linea *SCL*. Al nono impulso della linea *SCL* il Master si aspetta una risposta di un *bit* da parte della periferica che ha chiamato. La risposta della periferica chiamata consiste nel mantenere a livello basso la linea *SDA*, per la durata di un ciclo *SCL*, azione denominata *acknowledge* da parte della periferica chiamata. Una sola periferica risponderà alla chiamata del *master* e nel caso in cui nessuna periferica risponda, il *master* libererà il *BUS* permettendo così ad altri *master* di ottenere la linea per effettuare altre comunicazioni;

- Dopo l'avvenuto riconoscimento della periferica, il *master* inizia lo scambio dei dati inviando pacchetti composti da 8 *bit* e ad ogni pacchetto il *master* attende il segnale che avvisa dell'avvenuta ricezione.
- Quando la trasmissione è terminata il *master* libera il *BUS* inviando un segnale di *stop* che consiste nella transizione dal livello basso a quello alto della linea *SDA*, quando la linea *SCL* è alta.
  Se il *master* dovesse effettuare un'ulteriore comunicazione con un altro *slave*, piuttosto che liberare il *BUS*, rischiando di perderne il diritto del controllo, può inviare una nuova sequenza di *start*. Il vecchio *slave* comprenderà che la comunicazione con lui è terminata e ne comincerà una nuova con un altro *slave*.

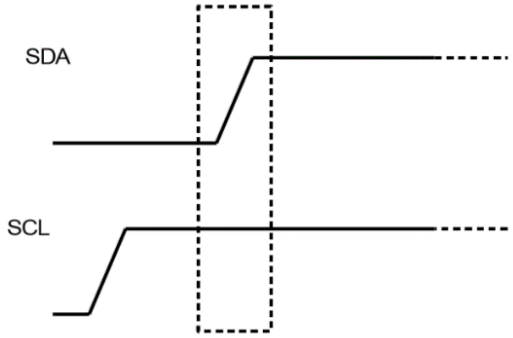

Figura 3.48 Sequenza di stop I^2C

L'interfaccia I^2C consente dunque di poter disporre di una larga varietà di dispositivi, individuabili singolarmente, all'interno di un *BUS*. Essi possiedono al loro interno l'*hardware* necessario per la gestione del protocollo rendendo di fatto agevole la costruzione di piccole reti di periferiche.

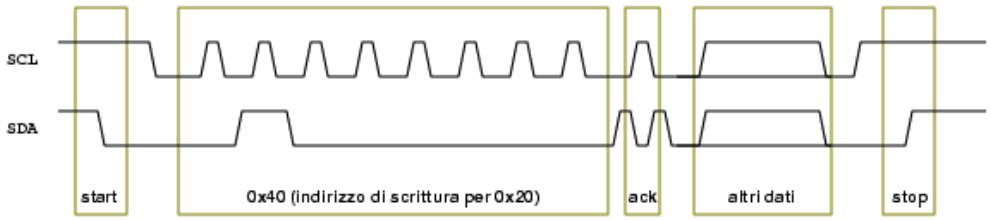

Figura 3.49 Esempio di trasmissione dati con il protocollo I^2C

3.10.4 Scomposizione in *byte*

I dati viaggiano attraverso i diversi *BUS* sottoforma di valori binari ovvero assumendo due diversi valori, 0 e 1, formattati secondo sequenze di 8 *bit* che rappresentano l'unità definita come *Byte* ovvero un elemento che può assumere 256 valori diversi compresi tra 0 e 255 ovvero 2^8 valori diversi.
I dati che devono essere trasferiti però possono assumere valori più elevati di 255 o comunque non per forza devono essere interi o positivi, è necessario dunque valutare attentamente il tipo di variabile appropriato a rappresentare e a immagazzinare correttamente nella memoria del calcolatore il dato.
In questo caso concreto le variabili che rientrano nella dimensione del *Byte* possono essere trasferite senza problemi mentre nel caso del valore *target* o del conteggio raggiunto dall'*encoder*, è necessario uno spazio di archiviazione e una codifica che supera il Byte, per questo motivo si rende necessaria una procedura di scomposizione in grado di riportare tale variabile all'unità minima del Byte.
Tenendo conto delle dimensioni del volume di lavoro (2,2 m x 0,8 m x 1,5 m), della risoluzione fornita dall'*encoder* (256 impulsi/giro) e delle dimensioni del rocchetto su cui viene avvolto il cavo (2 x 0,055 m x π =0,346 m) il valore approssimativo che può raggiungere è calcolabile nel seguente modo:

$$valore\ Max = \frac{\sqrt{\sqrt{(2{,}2\ m)^2 + (0{,}8\ m)^2} + 1{,}5\ m^2}}{0{,}346 \frac{m}{giro}} \times 256 \frac{impulsi}{giro} = 1586\ impulsi \qquad [Eq.3.43]$$

Dunque per rappresentare in maniera coerente una variabile di questo tipo e ricordando che questa può assumere esclusivamente valori interi è necessaria una variabile di tipo *integer* con dimensione 16 *bit* o 2 *Byte* che viene definita in ambiente *MATLAB* con il prefisso *int16* mentre in ambiente Arduino viene indicata con il prefisso *int*. Va ricordato che tale variabile assicura la possibilità di rappresentare numeri interi che vanno da -32˙768 (-2^{15}) a 32˙767 (2^{15}-1) sufficiente a rappresentare un *range* molto più ampio del necessario.
I valori delle distanze che rappresentano la lunghezza *target* del cavo necessaria ad ottenere il posizionamento voluto possono essere solo positivi, ma si è deciso di adottare un'ulteriore variabile in grado di esprimere il segno che in fase di scomposizione viene scorporato dalla variabile per evitare il rischio di *overflow* ed eventuali problemi di

codifica e per consentire dunque il funzionamento di una delle modalità di utilizzo che prevede il ripristino dell'attività anche a seguito di un'interruzione volontaria e completa dell'alimentazione, evento che porta ad un azzeramento degli *encoder*. Con questa funzione è possibile riprendere il funzionamento con l'*end effector* in qualunque posizione del *workspace* senza eseguire la procedura di calibrazione e settaggio iniziale.
A causa dell'eccessiva dimensione, per essere trasferite a livello di *BUS*, queste variabili devono prima essere scomposte in due parti dimezzandone in questo modo la loro dimensione da 16 a 8 *bit* senza però perderne l'informazione garantendo la possibilità di ricostruzione di quest'ultima una volta arrivata a destinazione.
Il procedimento di scomposizione prevede l'utilizzo di un apposito operando, utile ad ottenere il resto di una qualsiasi operazione aritmetica di divisione tra due numeri interi. In ambiente *MATLAB* viene eseguito con la funzione *mod*(dividendo, divisore) mentre nello *sketch* di Arduino viene definito con la funzione dividendo % divisore.
Tale resto rappresenta il *LSB* (*Less Significant Bit*) mentre l'intero rappresenta il *MSB* (*Most Significant Bit*). Inoltre per veicolare l'informazione del segno viene utilizzato, come anticipato in precedenza, un terzo *Byte* che assume valore 0 se il numero è negativo o 1 se il numero è maggiore o uguale a 0. Viene riportato di seguito un esempio numerico per spiegare meglio il concetto.

Ipotizzando che il numero da trasmettere sia 5˙000 (in binario è uguale a 0000001000111001):

- Il Byte riguardante il segno assumerà un valore pari a 1 (in binario 00000001);
- L'operazione di divisione fornisce un resto di 136 (in binario 00010001);
- L'intero è invece pari a 19 (in binario 00011001).

Di seguito vengono riassunti i passaggi matematici dell'operazione appena vista.

$$\frac{5\dot{}000}{256} = 19{,}53125$$

$$Intero_{MSB} = 19$$

$$Resto_{LSB} = 0{,}53125 \times 256 = 136$$

A seguito di questi passaggi il valore iniziale 5˙000 diventa una tripletta (1, 19, 136) di valori costituiti da interi rappresentabili con 8 *bit* e dunque facilmente trasmissibili attraverso i diversi *BUS* senza problemi.

Il destinatario di questi dati può ritrasmetterli in questa forma o può ricostruire il valore originario se necessario compiendo il procedimento matematico inverso:

- Partendo dal *MSB* si esegue una moltiplicazione per 256 che era il dividendo utilizzato nella scomposizione;
- Si somma il valore appena ottenuto con il resto della divisione;
- Si moltiplica il valore per 1 se il *Byte* del segno è pari a 1 o per -1 se quest'ultimo è pari a 0 e per ricapitolare:

$$19 \times 256 = 4˙864$$

$$4˙864 + 136 = 5˙000$$

$$5˙000 \times 1 = 5˙000$$

4 Dettaglio del prototipo

Questo capitolo ha lo scopo di descrivere nel dettaglio come è stato realizzato il prototipo in tutte le sue componenti fondamentali e di definirne la logica di funzionamento e di interpretazione dei comandi utilizzata.

4.1 Hardware

4.1.1 Unità motore (Slave)

Ogni unità motore, definibile anche come *slave,* è composta da diversi elementi i quali concorrono a rendere l'unità autonoma e in grado di eseguire istruzioni impartite mediante la rete *CAN* di cui ogni unità rappresenta un nodo. I principi di funzionamento e caratteristiche sono stati introdotti all'interno del paragrafo 3.8.
Come anticipato in precedenza i componenti fondamentali che compongono la singola unità *slave* sono:

- il motoriduttore accoppiato al proprio *driver* di gestione, utile a garantire la potenza necessaria a gestire la variabile della lunghezza del cavo e a contrastare i momenti resistenti causati dal peso dell'*end effector*;
- un *encoder* accoppiato direttamente all'albero del *drum*, montato in posizione contrapposta al motore. Quest'ultimo è gestito da un microcontrollore dedicato (Arduino Nano) che ha lo scopo di monitorare la posizione angolare assunta dal motoriduttore aggiornando una specifica variabile ogni qual volta l'encoder genera un *interrupt*.
  Questa informazione viene inviata mediante *BUS* I^2C solo su richiesta al microcontrollore che gestisce l'unità;
- un microcontrollore (Arduino Uno) dotato di uno *shield CAN BUS* che ha lo scopo di gestione generale dell'unità motore e di interfacciare quest'ultima con l'unità *master* mediante *CAN*.
  Questo componente è in grado dunque di:
  - richiedere informazioni riguardanti il valore raggiunto dal contatore;

- gestire mediante porte digitali il motoriduttore pilotando il *driver* secondo il *feedback* di posizione fornito dal contatore;
- scambiare informazioni con il *master* inviando il dato relativo al conteggio raggiunto e ricevendo informazioni aggiornate sul successivo *target* da inseguire tramite la rete *CAN*.

L'unità motore comprende anche il *drum* dove viene raccolto il proprio cavo di competenza.
Il *drum* ha un diametro di 0,11 m ed è costituito di materiale plastico.
I microcontrollori sono alimentati con un circuito di alimentazione indipendente dotato di un alimentatore da 12 V in corrente continua.
I motoriduttori sono alimentati con una linea dedicata anch'essa da 12 V in corrente continua.
La figura 4.1 mostra lo schema elettrico degli elementi che compongono la singola unità.

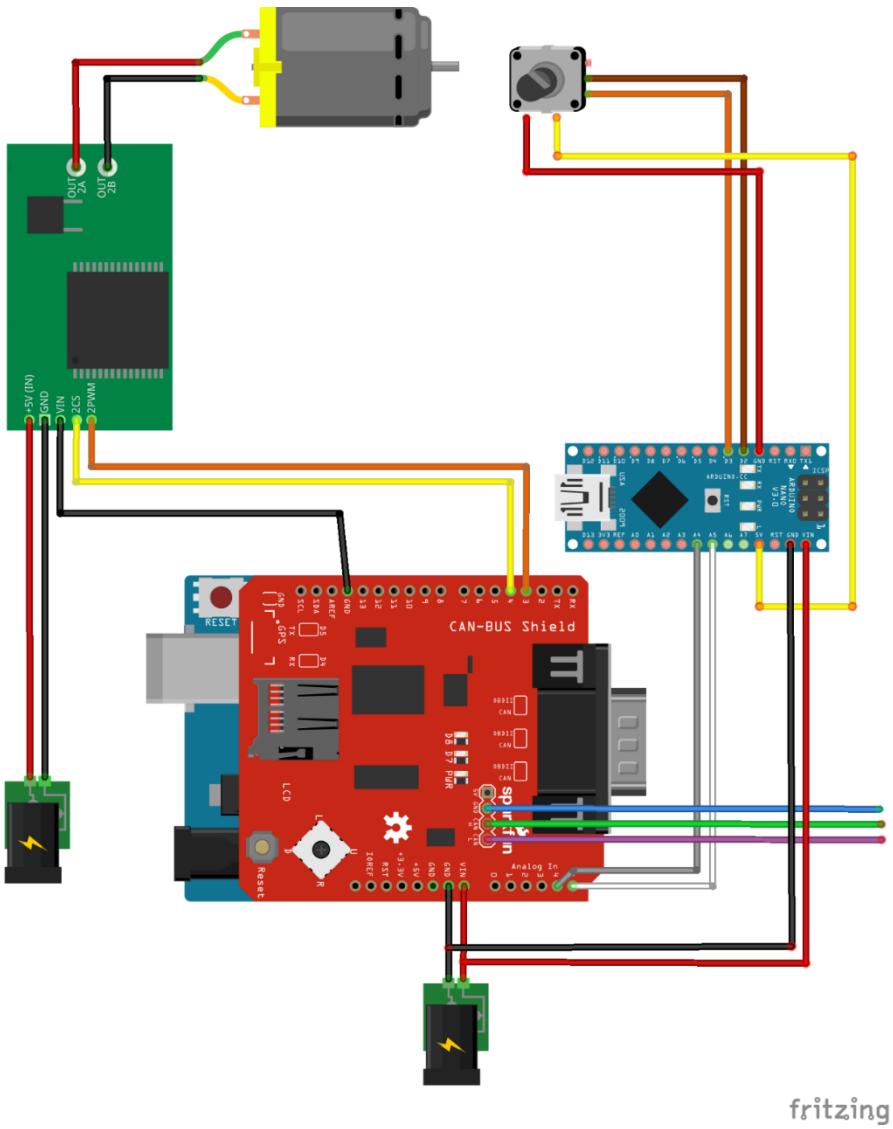

Figura 4.1 Schema elettrico dei componenti dell'unità motore (Slave)

4.1.2 Unità di controllo (Master)

L'unità di controllo ha il compito di gestire il funzionamento del sistema coordinando l'azione degli *slave* mediante l'invio dei dati necessari al loro funzionamento sul *CAN BUS* e scambiando dati via porta seriale con il *PC* che viene utilizzato per eseguire la maggior parte dei calcoli per mezzo del software *MATLAB*. Questa unità inoltre fornisce la possibilità di generare manualmente segnali di *input* utili a selezionare e gestire le diverse modalità di funzionamento del sistema oltre che a fornire un riscontro visivo mediante gli *output* riproposti su uno schermo *LCD*.
Nel dettaglio questa unità è composta da:

- due microcontrollori (Arduino Uno):
    - uno di questi è connesso via porta seriale con il *PC* e ha lo scopo di dialogare con l'algoritmo di controllo principale inviando i dati derivanti dagli *input*, utili a selezionare la modalità di funzionamento, e i dati inerenti ai valori raggiunti dai contatori dei singoli *slave*. Questo microcontrollore ha anche il compito di riceve anche i dati di prodotti da MATLAB riguardanti i singoli *target* da inviare agli *slave*. Tale microcontrollore gestisce inoltre in qualità di *master* il *BUS* I^2C locale utilizzato per dialogare con lo schermo *LCD* e per scambiare con il secondo microcontrollore i dati provenienti dagli *slave* in un senso e quelli destinati ad essi nell'altro;
    - Il secondo microcontrollore, dotato di *shield CAN BUS*, è in grado di consentire all'unità *master* di interfacciarsi con il *CAN BUS* per scambiare i dati sopra citati ed è programmato per comportandosi come *master* gestore della rete regolando il traffico dei dati.
- due levette analogiche di cui una utilizzata per la navigazione manuale (asse X e asse Y) e l'altra utilizzata per selezionare le opzioni e le modalità di funzionamento;
- due bottoni che forniscono gli input digitali utilizzati per la navigazione manuale (asse Z);
- uno schermo *LCD* per visualizzare le opzioni e le modalità di funzionamento direttamente dall'unità di controllo.

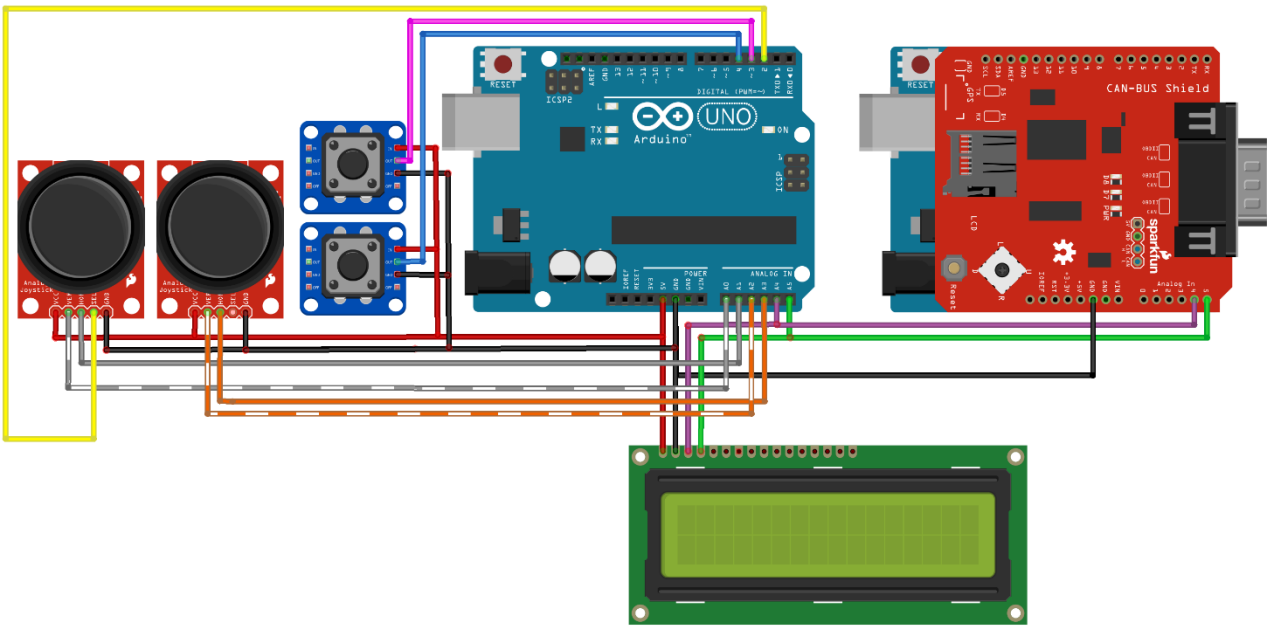

Figura 4.2 Schema elettrico dei componenti dell'unità di comando (Master)

4.1.3 Frame

Il *frame* rappresenta la struttura che racchiude il *workspace*, ad esso sono ancorate le unità motrici e gli *exit point* che sostengono i cavi in quota.

Per la realizzazione della struttura sono stati utilizzati profilati in alluminio anodizzato da 20 mm di lato, lunghi 3 m.

Dato il formato e i vincoli di spazio disponibile si è deciso di tagliare 2 profilati a metà ottenendo in questo modo uno sviluppo dello spazio di lavoro in altezza di 1,5 m e i restanti 4 profilati suddividendoli in 2,2 m e 0,8 m. Due profilati di questi sono poi stati ridotti a 0,76 m per poter configurare la struttura complessiva rappresentata in figura 4.4.

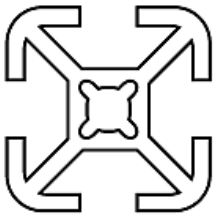

Figura 4.3 Sezione del profilato di alluminio

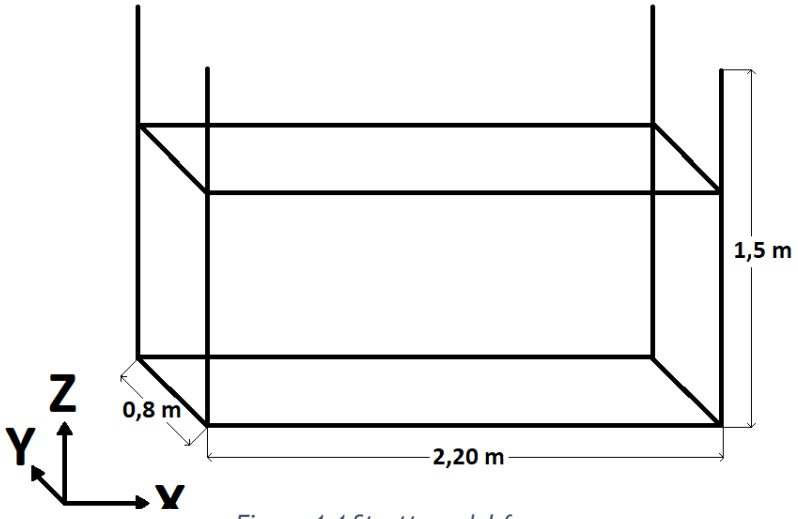

Figura 4.4 Struttura del frame

4.1.4 Exit point e definizione dello spazio utile

Gli *exit point* rappresentano un componente molto importante per il corretto funzionamento del prototipo perché, anche se sono componenti molto semplici, svolgono il ruolo importante di sostenere il cavo nel punto di quota massima ed è molto importante conoscerne la posizione.
Ogni cavo avrà dunque un suo *exit point* che rappresenta il preciso punto dello spazio dove il cavo entra all'interno del *workspace* e per questo è necessario ai fini della calibrazione conoscerne le sue coordinate nel modo più accurato possibile.
Dopo aver assemblato il prototipo è stato necessario procedere ad una prima calibrazione con lo scopo di trovare le coordinate di tali punti e di conseguenza definire le coordinate minime e massime del *workspace* effettivo che saranno diverse dalle dimensioni di massima del *frame* riportate in precedenza in figura 4.4.

Il procedimento di calibrazione può essere riassunto nelle seguenti fasi:

- Dopo aver caricato i cavi nei *drum* è stato applicato un peso al capo terminale;
- I cavi sono stati portati alla situazione in cui tale peso si trovi in corrispondenza dell'*exit point*;
- Successivamente i cavi sono stati allungati quasi fino a fare raggiungere quota zero in modo da sfruttarli come filo a piombo per determinare la proiezione dell'*exit point* sul piano di quota 0. In questo modo oltre a trovare le coordinate X e Y degli *exit point* si trovano le coordinate massime del volume di lavoro negli assi X e Y perché coincidono;
- Rilasciando ulteriormente il cavo fino a fare raggiungere la quota 0 al capo terminale del cavo si riesce a determinare la quota massima e la coordinata Z dell'*exit point*.

Il procedimento appena esposto viene ripreso graficamente nella pagina successiva in figura 4.5 mentre in tabella 4.1 vengono riportate le coordinate minime e massime del *workspace* e le coordinate spaziali dei singoli *exit point*.

Figura 4.5 Exit point e calibrazione della macchina

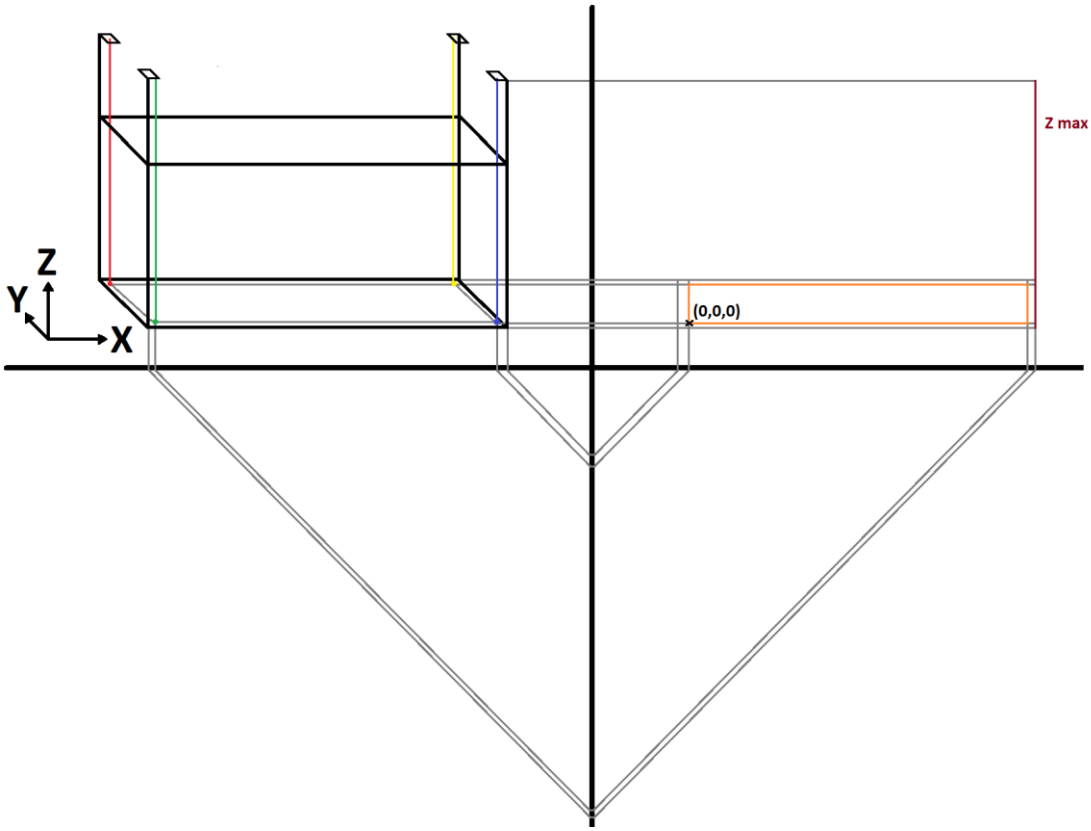

Tabella 4.1 Parametri dimensionali del workspace

Dimensioni massime work space	**X**	2.105	m
	Y	0.778	m
	Z	1.565	m

Coordinate exit point							
A	**X**	0	m	**C**	**X**	2.105	m
	Y	0	m		**Y**	0.778	m
	Z	1.565	m		**Z**	1.565	m
B	**X**	0	m	**D**	**X**	2.105	m
	Y	0.778	m		**Y**	0	m
	Z	1.565	m		**Z**	1.565	m

4.1.5 Assemblaggio

L'assemblaggio finale del prototipo prevede:

- il montaggio sulla struttura del *frame* delle 4 unità motore, dei rispettivi *exit point* e dei supporti guida cavo in prossimità del *drum* che hanno lo scopo di favorire la corretta gestione del cavo;
- Il cablaggio dei due circuiti separati di alimentazione (12 V *CC*) destinati rispettivamente ai motori e ai microcontrollori in modo da poter gestire separatamente i due circuiti ed escludere per esempio per ragioni di sicurezza la parte destinata ai motori senza causare il riavvio non voluto dei microcontrollori con conseguente perdita della posizione e evitare l'obbligo di calibrazione iniziale;
- Il montaggio di squadrette angolari per migliorare la rigidità del *frame*;
- Il cablaggio della dorsale del *CAN BUS* mediante l'utilizzo di un cavo idoneo e l'installazione delle resistenze terminali da 120 Ω.

4.1.6 End effector

L'*end effector* è il componente in grado di traslare all'interno del volume di lavoro ed è necessario per svolgere un particolare compito.
A livello di *software,* per quanto riguarda la cinematica, l'estensione di questo componente è considerata come puntiforme e capace di muoversi all'interno di una porzione del *workspace* definita come volume di lavoro effettivo.
L'ingombro effettivo verrà caratterizzato a seconda della tipologia di *end effector* e considerato nel caso della navigazione in presenza di ostacoli.
La differenza che esiste tra il *workspace* effettivo e quello potenziale sta nel fatto che alcune aree di quest'ultimo possono essere considerate come non raggiungibili per diversi motivi, ad esempio per:

- evitare zone che non assicurano adeguati livelli di tensione;
- evitare zone caratterizzate da livelli troppo elevati di tensione;
- negare la possibilità di intervenire in alcune aree potenzialmente dentro lo spazio utile ma che non devono essere raggiunte;
- considerare gli effettivi ingombri dell'*end effector*.

L'algoritmo di controllo prevede la possibilità di utilizzare diverse tipologie di *end effector* caratterizzabili per la tipologia di ancoraggio adottato, unico o singolo per ogni cavo.
Per questo motivo vanno inseriti i parametri dimensionali degli ancoraggi dei cavi e dell'ingombro dell'*end effector* secondo la logica indicata di seguito in figura 4.6.
A sinistra è rappresentato l'*end effector* dotato di un ancoraggio singolo centrale mentre a destra è riportato l'esempio in cui gli ancoraggi sono unici per ogni singolo cavo.
I parametri di correzione (K_{eeZ}, K_{eeX}, K_{eeY}) sono determinati secondo la logica descritta nella figura.

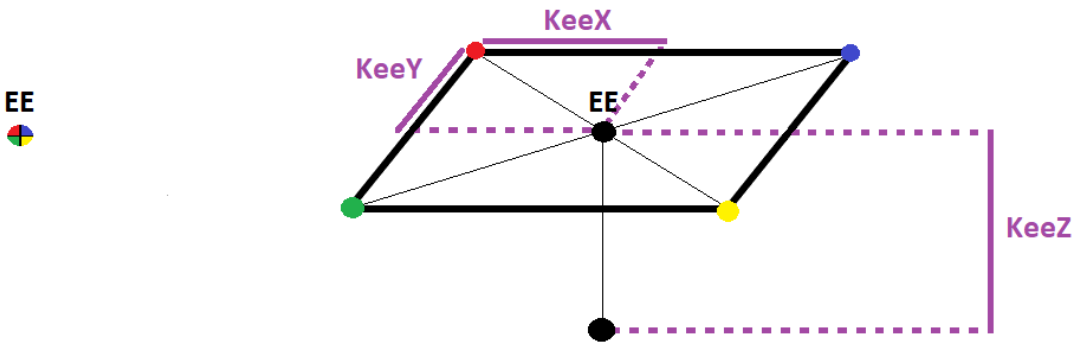

Figura 4.6 Coefficienti di correzione caratteristiche di ingombro e ancoraggio dell'end effector

Le piattaforme utilizzate per lo svolgimento delle funzioni secondarie prevedono, nello stadio di sviluppo del prototipo, l'adozione di un microcontrollore Arduino Uno interfacciato, per mezzo del *CAN BUS*, alla dorsale principale di trasmissione dati.
I dati e l'alimentazione dell'elettronica presente a bordo, sono veicolati mediante cavo di rete *LAN* modificato per lo scopo.
Nel caso dell'*end effector* utilizzato per effettuare le scansioni, a bordo della piattaforma è presente un sensore ad ultrasuoni posizionato al di sotto di quest'ultima per misurare la distanza che separa tale sensore dal suolo o da un eventuale ostacolo presente sotto di questo.
Per tale motivo il sensore è stato utilizzato per compiere alcune prove sperimentali di misurazione che hanno come scopo la scansione preliminare del volume di lavoro in modo da rendere possibile la realizzazione di una mappa digitale della conformazione dello spazio operativo effettivo per poter istruire l'algoritmo di controllo in merito alle coordinate possibili.

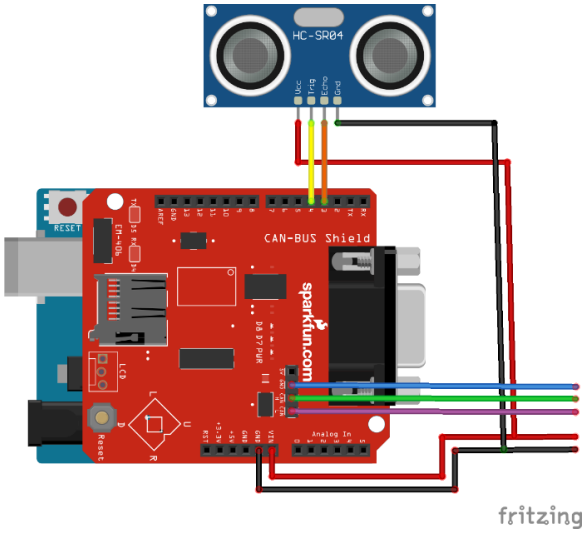

Figura 4.7 Schema elettrico end effector (scanner)

Nel caso dell'*end effector* utilizzato per effettuare le operazioni di intervento mediante fertirrigazione non è presente elettronica a bordo ma è stato realizzato per lo scopo un nodo aggiuntivo della rete *CAN* utile alla gestione dell'irrigazione per mezzo di una pompa collegata tramite tubazione in silicone direttamente all'*end effector*.
La logica di funzionamento prevede l'utilizzo di un relè per gestire l'attuazione del motore elettrico della pompa. Il microcontrollore pilota il relè secondo le istruzioni ricevute mediante CAN BUS provenienti dal master.
Lo schema elettrico del nodo è mostrato in figura 4.8.

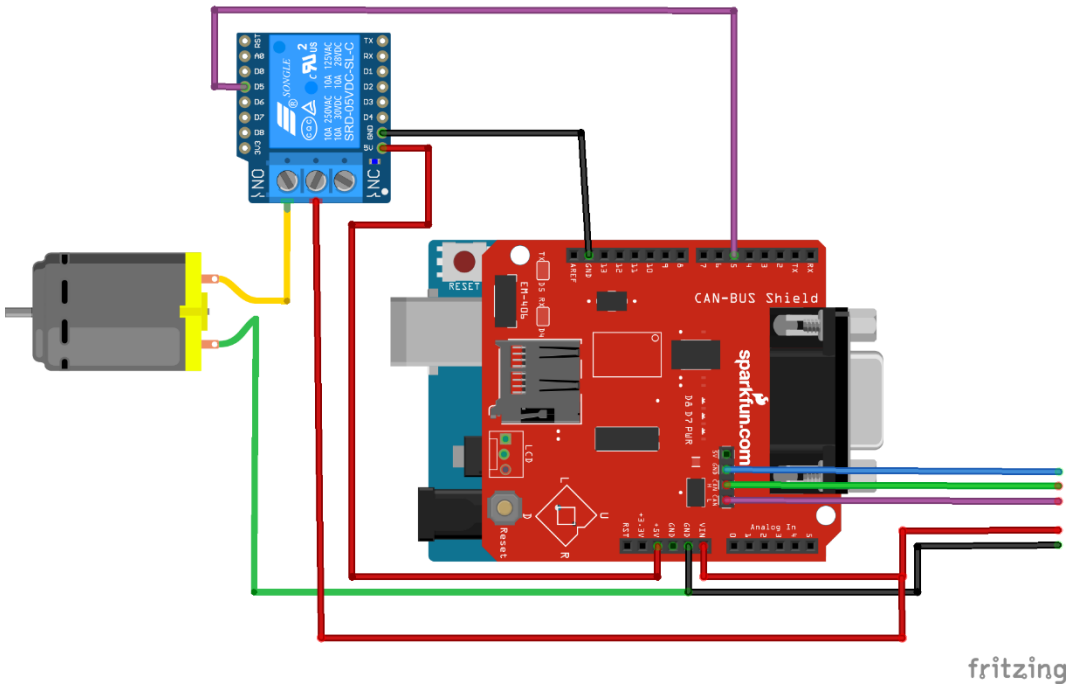

Figura 4.8 Schema elettrico nodo CAN di gestione della pompa di irrigazione

4.2 Software

Il paragrafo dedicato al *software* riporta una rassegna delle principali funzioni necessarie al funzionamento del prototipo introducendo l'architettura generale e la logica di alcune funzioni utilizzate per gestire le diverse modalità di funzionamento.

Come anticipato in precedenza la parte di *software* dedicata alla gestione dei microcontrollori è stata sviluppata con il linguaggio *Wiring* mentre la parte di calcolo in linguaggio *MATLAB*.

L'architettura generale prevede che ciclicamente, ogni 50 ms circa (che corrisponde a una frequenza di 20 Hz), il *master*:

- Invia un primo *Byte* per mezzo della porta seriale a *MATLAB,* utile a identificare la modalità di funzionamento e quindi selezionare la regione di codice che verrà eseguita e definire il quadro interpretativo che verrà data ai successivi *Byte* ricevuti;
- invia altri 3 Byte che contengono ulteriori istruzioni inviate a *MATLAB* per acquisire e interpretare gli *input* forniti dall'utente, utili a gestire le funzioni della modalità selezionata con il primo *Byte*;
- invia 12 *Byte* a *MATLAB* contenenti i valori dei conteggi raggiunti dai contatori degli *slave* scomposti in *Byte* e formattati nel seguente modo:
    - 1 *Byte* che esprime il segno del valore raggiunto dal conteggio del contatore dello *slave A*;
    - 2 *Byte* (*LSB -- MSB*) che esprimono il valore del conteggio del contatore dello *slave A* scomposto in 2 *Byte* secondo il procedimento esposto al paragrafo 3.10.4.
    - Queste due operazioni si ripetono in successione anche per gli *slave B,C* e *D*.
- Invia 1 Byte contenente il valore della distanza rilevato dal sensore a ultrasuoni

In questo modo *MATLAB* possiede tutti gli elementi per eseguire i calcoli e restituire i risultati inerenti ai *target* aggiornati che gli *slave* dovranno seguire durante l'esecuzione del *loop* successivo.

Il calcolo dei Δ attribuiti alle coordinate e che esprimono il moto dell'*end effector* in funzione del tempo sono calcolati come visto nel procedimento riportato nel paragrafo 3.3.3 e utilizzando come Δt il tempo di esecuzione dell'algoritmo di controllo del *loop* precedente.

Infine *MATLAB*:

- invia 12 *Byte* al *master* via porta seriale che rappresentano i dati riguardanti i *target* da inseguire scomposti in *Byte* con lo stesso principio visto in precedenza e secondo l'ordine:
    - 2 *Byte* (*LSB -- MSB*) che rappresentano il *target* scomposto in *Byte*;
    - 1 *Byte* per esprimere il segno del *target*.
- Invia 1 Byte contenente i comandi utili per gestire le funzioni degli attuatori presenti nell'*end effector* o in altri nodi della rete *CAN*.

In questo modo il *loop* principale termina e si ripete dall'inizio.
All'interno della tabella 4.2 viene riproposto schematicamente l'ordine di invio e ricezione dei dati da parte del *master* e successivamente in figura 4.9 e figura 4.10 vengono riproposti rispettivamente il diagramma che rappresenta lo schema complessivo di trasmissione dei dati e quello a livello delle singole unità, evidenziando il protocollo di trasmissione utilizzato e tra le parentesi quadre la dimensione in *Byte* del vettore contenente i dati.

Tabella 4.2 Definizione dei Byte di input e output

Vettore di input MATLAB			N Byte
Comandi		Mode	1
		Comando 1	2
		Comando 2	3
		Comando 3	4
Dati encoder	A	Segno	5
		LSB	6
		MSB	7
	B	Segno	8
		LSB	9
		MSB	10
	C	Segno	11
		LSB	12
		MSB	13
	D	Segno	14
		LSB	15
		MSB	16
	EE	Distanza	17

Vettore di output MATLAB			N Byte
Dati Target	A	LSB	1
		MSB	2
		Segno	3
	B	LSB	4
		MSB	5
		Segno	6
	C	LSB	7
		MSB	8
		Segno	9
	D	LSB	10
		MSB	11
		Segno	12
	EE	Comandi	13

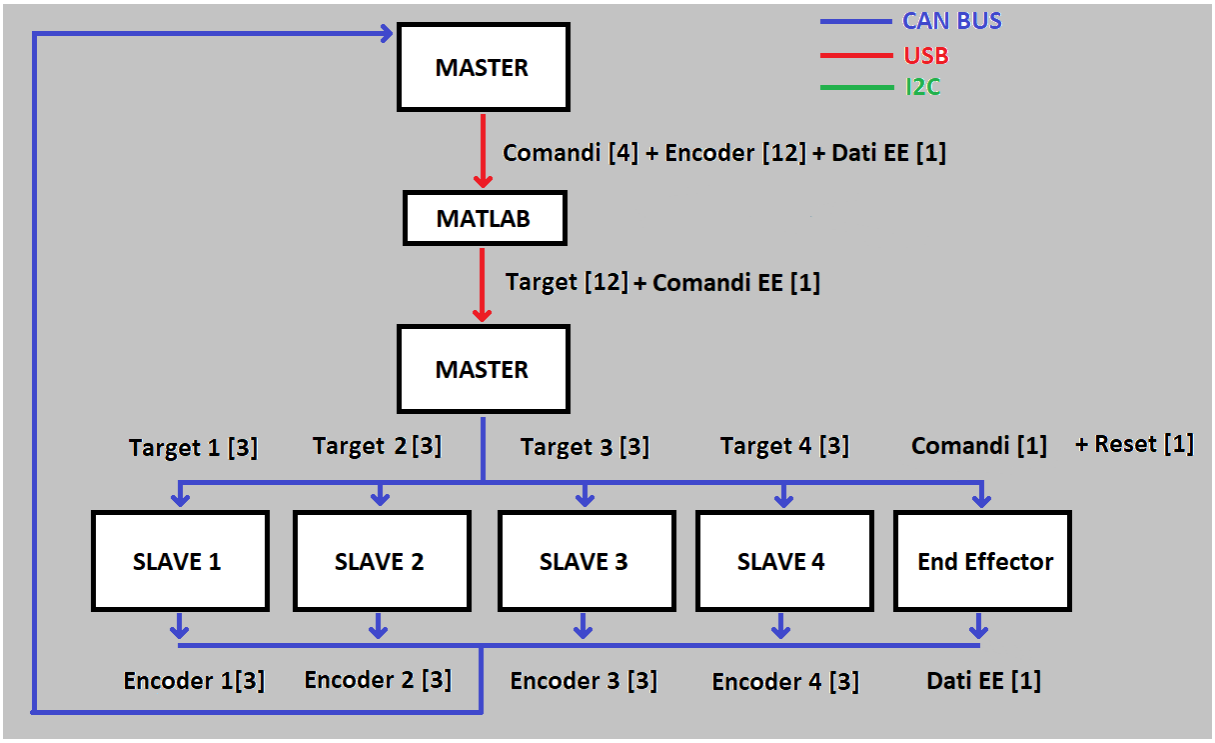

Figura 4.8 Diagramma globale di trasmissione dei dati

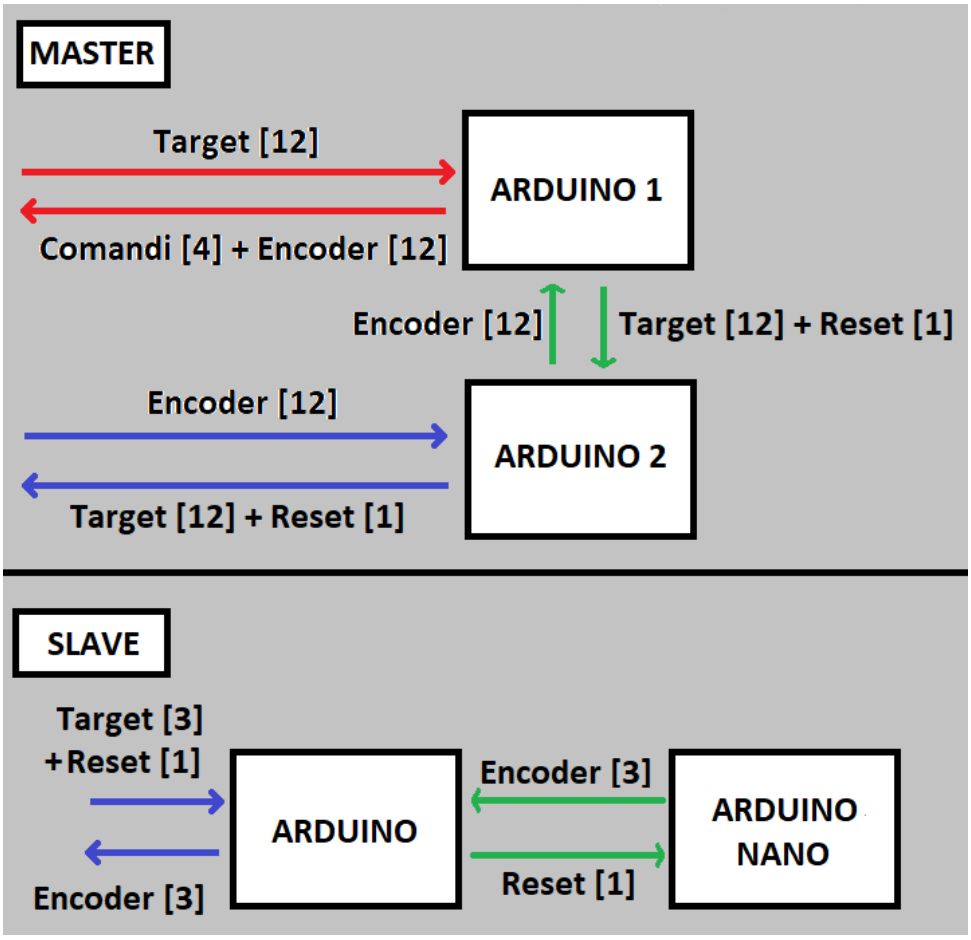

Figura 4.9 Diagramma locale di trasmissione dei dati

4.2.1 Gestione ID CAN BUS

La trasmissione dei dati mediante *CAN BUS* è gestita dal *master* che invia e riceve i dati necessari per coordinare l'azione dei motori secondo le modalità descritte nel paragrafo 3.10.2.
Ultimo punto fondamentale è definire quali siano i dati da trasmettere e con quale priorità. A tale scopo sono stati definiti gli *ID* dei messaggi e il contenuto, dando precedenza ai target che guidano l'azione dei motori, e alla funzione di reset simultaneo. Per i dati inerenti agli *encoder* è stato scelto di definire un doppio livello di priorità:

- basso per il funzionamento normale in modo da avere una priorità più bassa rispetto ai target;
- alto nel caso in cui l'aggiornamento del valore di un encoder non sia pervenuto entro una soglia di tempo inferiore a 0,5 s dall'ultimo aggiornamento. In questo caso il *master* invia un messaggio di richiesta di aggiornamento a priorità elevata che autorizza lo slave a utilizzare il messaggio di aggiornamento a priorità elevata.

Questo tipo di organizzazione degli *ID* è stato scelto per consentire allo *slave* di ricevere i dati aggiornati che del *target* da inseguire prima che il motore riesca a raggiunge il *target* precedentemente ricevuto consentendo una gestione fluida e ottimale del moto.

Tabella 4.3 Significati degli ID dei messaggi CAN

CAN ID	DESCRIZIONE
10	Reset conteggio encoder (A B C D)
11	Richiesta dati encoder A
12	Richiesta dati encoder B
13	Richiesta dati encoder C
14	Richiesta dati encoder D
21	Dati encoder A ad elevata priorità
22	Dati encoder B ad elevata priorità
23	Dati encoder C ad elevata priorità
24	Dati encoder D ad elevata priorità
25	Dati target slave A
26	Dati target slave B
27	Dati target slave C
28	Dati target slave D
31	Dati encoder A a bassa priorità
32	Dati encoder B a bassa priorità
33	Dati encoder C a bassa priorità
34	Dati encoder D a bassa priorità
100	Dati sensori end effector
101	Dati comandi end effector

In tabella 4.3 vengono elencati gli *ID* utilizzati e descritto il contenuto del messaggio che rappresentano.
Sono stati inoltre lasciati volutamente liberi *ID* con diversi gradi di priorità per consentire l'eventuale futura modifica o ampliamento delle funzioni senza stravolgere il protocollo di comunicazione già consolidato.

4.2.2 Opzioni generali

La prima funzione disponibile all'avvio del programma è una schermata di opzioni generali visualizzata sullo schermo *LCD* del *master*.
Questa schermata offre la possibilità di attivare o disattivare i due meccanismi di correzione della lunghezza del cavo utilizzati per :

- contrastare il fenomeno dell'elasticità del cavo, se presente;
- tenere in considerazione l'andamento del cavo che segue la forma della curva catenaria se significativo, specie quando si vuole lavorare con livelli bassi di tensione dei cavi, quindi con *end effector* dotati di peso limitato o se si vuole operare in regioni del *workspace* che prevedono livelli di tensione bassi per almeno uno dei cavi.
  Questa funzione si rivelerebbe utile specie nel caso in cui in un progetto di maggiore portata si debba usare un cavo dotato di massa lineare maggiore.

Questa schermata rende possibile il salvataggio nella memoria del *PC* del valore raggiunto da tutti i contatori in modo da poterlo poi richiamare tramite apposita funzione. Questa funzione è utile se si vuole riprendere il lavoro dopo che è stata interrotta l'alimentazione dei contatori e senza procedere a una calibrazione o azzeramento. Questa modalità di funzionamento è identificata come *Mode 0* e porta all'arresto del moto dell'*end effector* perché prevede che per ogni *slave* sia inviato il comando di equiparare il target al valore raggiunto dall'encoder. Il significato dei *Byte* di comando è illustrato in tabella 4.4.

Tabella 4.4 Codifica comandi opzioni generali

Mode	Comando 1	Comando 2	Comando 3	Effetto
0	1	0	0	Correzione catenaria disattivata
0	1	1	1	Correzione catenaria attivata
0	2	0	0	Correzione elasticità cavo disattivata
0	2	1	1	Correzione elasticità cavo attivata
0	3	0	0	Salvataggio posizioni correnti degli encoder
0	4	1	1	Recupero posizioni encoder salvate

4.2.3 Gestione dei singoli cavi

La modalità di gestione dei cavi singoli è identificata come *Mode 1* e ha lo scopo di consentire a livello di *master* di consentire la selezione e il comando dei motori singolarmente, essa possiede due scopi principali:

- accorciare o allungare manualmente i cavi singolarmente, selezionando lo *slave* è possibile variarne il *target* inseguito con velocità modulabile utilizzando la levetta analogica e ottenere un posizionamento di precisione utilizzando i bottoni;
- azzerare simultaneamente il conteggio di tutti gli *encoder*, funzione necessaria in sede di calibrazione.

Lo scopo dell'azzeramento nel momento in cui tutti i cavi sono stati portati manualmente a livello degli *exit* point è quello di consentire la calibrazione del sistema.
Per questo motivo è quindi necessario gestire manualmente la lunghezza dei cavi per portarne l'estremo terminale in prossimità dell'*exit point* e una volta effettuato l'azzeramento è dunque anche possibile portare manualmente i cavi in prossimità dei propri attacchi sull'*end effector* per facilitare le operazioni di settaggio iniziale.

Il dato della velocità ricevuto con il comando 3 viene tradotto da *MATLAB* in uno spostamento del *target* corrispondente secondo il calcolo:

$$\Delta Target_{(i)} = \frac{Comando\ 3}{255} \times \Delta t \times \frac{Vmax\ motore}{60} \times Risoluzione\ encoder \qquad [Eq. 4.1]$$

Tabella 4.5 Codifica comandi modalità di gestione dei singoli cavi

Mode	Comando 1 Selezione *slave*	Comando 2	Comando 3 Velocità	Effetto
1	1 - 4	0	0 - 255	Accorciamento cavo dello *slave* selezionato con velocità proporzionale a intensità *input*
1	1 - 4	1	0 - 255	Allungamento cavo dello *slave* selezionato con velocità proporzionale a intensità *input*
1	1 - 4	2	1	Allungamento cavo dello *slave* selezionato di uno *step*
1	1 - 4	3	1	Accorciamento cavo dello *slave* selezionato di uno *step*
1	9	9	9	Azzeramento simultaneo di tutti i contatori

4.2.4 Determinazione del punto di inizio del moto

Dopo aver effettuato l'azzeramento è possibile selezionare un punto di inizio da cui voler far partire l'*end effector* per mezzo della modalità di funzionamento definita *Mode 2*. In questa modalità i *Byte* di comando saranno utilizzati come *input* per selezionare le coordinate di inizio per poter utilizzare successivamente le modalità manuali ed automatiche.
Una volta che sono state inserite le coordinate di partenza, le lunghezze dei cavi vengono adeguate automaticamente per ottenere il posizionamento selezionato e verrà considerato dal *software* come *feedback* l'errore di posizionamento.

Dopo che tale errore sarà inferiore a una certa soglia il posizionamento iniziale sarà considerato andato a buon fine e sarà possibile sfruttare le funzioni di navigazione automatica o manuale per fare traslare l'*end effector* all'interno del *workspace*.

Tabella 4.6 Codifica comandi modalità di selezione del punto iniziale

Mode	Comando 1	Comando 2	Comando 3	Effetto
2	1	0	0	Aumenta coordinata X
2	2	0	0	Dimunuisce coordinata X
2	0	1	0	Aumenta coordinata Y
2	0	2	0	Dimunuisce coordinata Y
2	0	0	1	Aumenta coordinata Z
2	0	0	2	Dimunuisce coordinata Z
2	3	3	3	Conferma del posizionamento selezionato
2	4	4	4	Annullamento del posizionamento selezionato

4.2.5 Navigazione manuale

La modalità manuale, definita *Mode 3,* offre la possibilità di spostare l'*end effector* manualmente utilizzando la leva analogica e i pulsanti presenti sul *master*.
Questa modalità può essere utilizzata per eseguire manualmente delle traiettorie che possono essere poi riutilizzate nella modalità automatica grazie alle funzione di raccolta dati disponibile in questa modalità.
In questa modalità vengono monitorati e registrati, oltre alle coordinate occupate dall'*end effector* in funzione del tempo, altre variabili utili ai fini di possibili analisi di questi dati per verificare il funzionamento del prototipo e per ottimizzarlo di conseguenza.

Le variabili ritenute interessanti riguardano la posizione dell'*end effector*, gli errori di posizionamento dei singoli motori, gli angoli di navigazione e la velocità del moto.

I dati necessari al funzionamento di questa modalità sono gli angoli di navigazione e la velocità che vengono utilizzati, come visto nel paragrafo 3.3.3, e che sono ottenuti dall'interpretazione dei comandi secondo il criterio esposto in tabella 4.7.

Tabella 4.7 Codifica comandi modalità di navigazione manuale

			Conversione trigonometrica		
	Valori utilizzati	Significato	0 - 127	128	129 - 255
Mode	3				
Comando 1	0 - 255		-1 - 0	0	0 - 1
Comando 2	0 - 255		-1 - 0	0	0 - 1
Comando 3	0 - 255		-1	0	1

I comandi 1 e 2 per essere convertiti in valori idonei da usare per il calcolo degli angoli di navigazione vengono inseriti nel calcolo della seguente funzione di trasformazione:

$$K_{(i)} = \left(7{,}8 \times 10^{-3} \times Comando_{(i)}\right) - 1 \qquad [Eq.\,4.2]$$

$$azimut = \tan^{-1}\left(\frac{K_{(1)}}{K_{(2)}}\right) \qquad [Eq.\,4.3]$$

$$zenit = \tan^{-1}\left(\frac{K_{(3)}}{\sqrt{{K_{(1)}}^2 + {K_{(2)}}^2}}\right) \qquad [Eq.\,4.4]$$

4.2.6 Navigazione automatica

La navigazione automatica, o *Mode 4*, è una modalità che prevede la possibilità di fare seguire precise traiettorie all'*end effector* in maniera automatica.
Questa modalità prevede che le traiettorie da eseguire vengano calcolate recuperando i dati inseriti in un apposito foglio di calcolo. In questo foglio di calcolo vengono inserite le coordinate dei vertici delle traiettorie da seguire insieme alla velocità di spostamento dell'*end effector* in forma di indice unitario e altri dati riguardanti i compiti da eseguire,

le pause da effettuare in prossimità dei vertici (*via point*) e se l'*end effector* debba solo passare per il vertice senza fermarsi (*overfly*).

Dopo avere raggiunto la posizione iniziale inserita manualmente o durante il funzionamento in modalità manuale, è possibile passare alla modalità automatica. Quando viene attivata questa modalità, l'*end effector* viene portato automaticamente nelle coordinate corrispondenti al primo vertice della traiettoria inserita e successivamente segue la traiettoria impostata in precedenza passando per tutti i vertici indicati nel foglio di calcolo.
La logica di funzionamento prevede il calcolo degli angoli di navigazione con la differenza di posizione tra il punto occupato dall'*end effector* e il punto che deve essere raggiunto indicato come successivo vertice della traiettoria.
L'*end effector in questo modo* è in grado di muoversi nel volume di lavoro seguendo la traiettoria e mantenendo la velocità impostata considerando anche un valore di accelerazione positiva o negativa se la velocità di quest'ultimo varia nel tempo.
Il procedimento di calcolo delle principali variabili necessarie a calcolare gli angoli di navigazione determinando di conseguenza l'entità dello spostamento dell'*end effector* lungo gli assi in funzione del tempo è riportato di seguito:

$$\Delta X_{(i)} = X_{vertice\,(i)} - X_{EE} \quad [\mathrm{m}] \quad [Eq.4.5]$$

$$\Delta Y_{(i)} = Y_{vertice\,(i)} - Y_{EE} \quad [\mathrm{m}] \quad [Eq.4.6]$$

$$\Delta Z_{(i)} = Z_{vertice\,(i)} - Z_{EE} \quad [\mathrm{m}] \quad [Eq.4.7]$$

$$Dist_{azimut\,(i)} = \sqrt{\Delta X_{(i)}{}^2 + \Delta Y_{(i)}{}^2} \quad [\mathrm{m}] \quad [Eq.4.8]$$

$$Distanza_{(i)} = \sqrt{Dist_{azimut\,(i)}{}^2 + \Delta Z_{(i)}{}^2} \quad [\mathrm{m}] \quad [Eq.4.9]$$

$$Zenit_{(i)} = \tan^{-1}\left(\frac{\Delta Z_{(i)}}{Dist_{azimut\,(i)}}\right) \quad [^\circ] \quad [Eq.\,4.10]$$

$$Azimut_{(i)} = \tan^{-1}\left(\frac{\Delta Y_{(i)}}{\Delta X_{(i)}}\right) \quad [^\circ] \quad [Eq.\,4.11]$$

$$\Delta X_{EE} = Vel_{EE} \times \cos\left(Zenit_{(i)}\right) \times \cos\left(Azimut_{(i)}\right) \times \Delta t_{loop} \quad [\mathrm{m}] \quad [Eq.\,4.12]$$

$$\Delta Y_{EE} = Vel_{EE} \times \cos\left(Zenit_{(i)}\right) \times \sin\left(Azimut_{(i)}\right) \times \Delta t_{loop} \quad [\mathrm{m}] \quad [Eq.\,4.13]$$

$$\Delta Z_{EE} = Vel_{EE} \times \sin\left(Zenit_{(i)}\right) \times \Delta t_{loop} \quad [\mathrm{m}] \quad [Eq.\,4.14]$$

$$X_{EE} = X_{EE} + \Delta X_{EE\,(i)} \quad [\mathrm{m}] \quad [Eq.\,4.15]$$

$$Y_{EE} = Y_{EE} + \Delta Y_{EE\,(i)} \quad [\mathrm{m}] \quad [Eq.\,4.16]$$

$$Z_{EE} = Z_{EE} + \Delta Z_{EE\,(i)} \quad [\mathrm{m}] \quad [Eq.\,4.17]$$

4.2.7 Gestione dei motori

La gestione dei motori è lasciata al microcontrollore che coordina l'azione dell'unità *slave*. Il microcontrollore utilizza come *input*:

- il valore del conteggio raggiunto da parte del contatore che fornisce in questo modo il posizionamento attuale aggiornato;
- il valore del *target* ottimale che viene ricevuto per mezzo del *CAN BUS* proveniente dal *master*.

$$\varepsilon_{(i)} = Target_{(i)} - Conteggio\ encoder_{(i)} \quad [Eq. 4.18]$$

Sulla base di queste informazioni il microcontrollore calcola l'errore di posizionamento e pilota il motore di conseguenza.
I segnali digitali che vengono generati dal microcontrollore, e che vengono inviati alla scheda di alimentazione per pilotare il motore, sono 2:

- il segnale che determina il senso di rotazione del motore può essere uno 0 logico o un 1 logico. Il microcontrollore genera uno di questi segnali a seconda che l'errore di posizionamento sia positivo o negativo. Per esempio se il valore dell'errore è minore di 0 vuol dire che il cavo in questione è più corto di quanto dovrebbe essere e dunque il motore viene fatto ruotare nel senso che assicura un allungamento del cavo che dipende da come è stato cablato il motore e da come è stato arrotolato il cavo sul *drum*;
- il segnale che determina la velocità di rotazione del motore dipende direttamente dall'intensità dell'errore stesso ed è un segnale digitale di tipo *PWM* (Paragrafo 3.8.5).
  Il valore che esprime il *duty cycle* viene determinato dal microcontrollore secondo una funzione di trasformazione proporzionale che tiene conto dell'errore di posizionamento assoluto.

$$Duty\ Cycle = |\varepsilon| \times 4.25 \qquad con \qquad 0 \leq Duty\ Cycle \leq 255 \quad [Eq. 4.19]$$

5 TEST PRELIMINARI

I test iniziali si sono svolti in laboratorio e sono stati necessari a verificare le funzionalità del prototipo, ottimizzarne i parametri di funzionamento delle funzioni di gestione dei motori, degli algoritmi di controllo automatico e manuale, di trasmissione dei dati e di sincronizzazione generale del sistema oltre che a verificare la dinamica del moto dell'*end effector* all'interno del *workspace* con lo scopo di rendere idonea la macchina a poter svolgere i test preliminari.
Dopo aver ottimizzato i suddetti parametri sono state svolte alcune prove sperimentali per verificare le funzionalità necessarie allo svolgimento alle prove in serra.
Il test preliminare è articolato in due distinte fasi:

1. Scansione del volume di lavoro mediante sensore ad ultrasuoni con lo scopo di costruire una mappa tridimensionale della disposizione di 10 vasi inseriti all'interno di quest'ultimo. Riprodotta la mappa è possibile determinare da *software* il posizionamento dei vasi;
2. Ottenuta la posizione dei vasi è possibile raggiungerli con l'*end effector* per simulare un'operazione di intervento sulla singola pianta. Lo scopo di questa fase è dunque quello di valutare da un lato l'accuratezza della determinazione del posizionamento dei vasi e dall'altro la precisione del posizionamento della macchina.

5.1 Prova di scansione del workspace

La prova di scansione dello spazio è stata eseguita mediante l'esecuzione automatica di una traiettoria, rappresentata in figura 5.1, passante per 600 punti di acquisizione individuabili come punti medi o centroidi di un ipotetica matrice a maglia quadrata di 0,05 m per lato parallela al suolo ad una quota di 0,45 m riprodotta in figura 5.2.

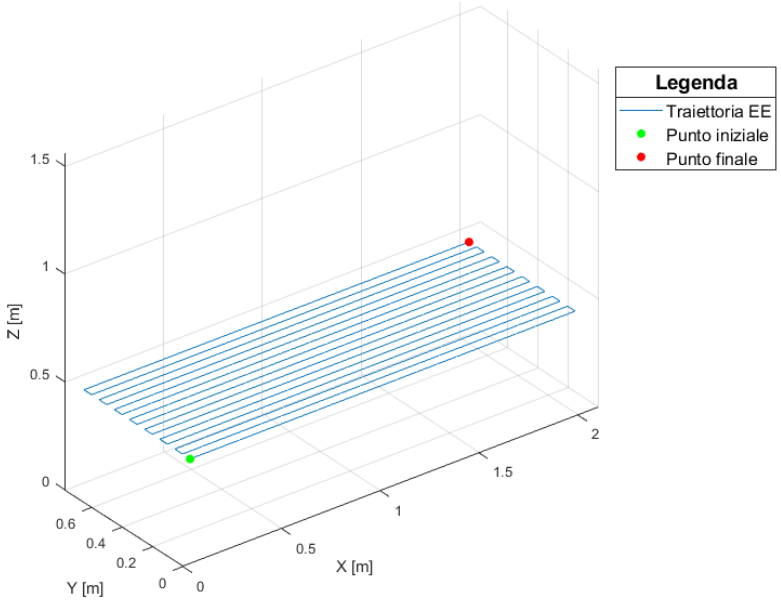

Figura 5.1 Traiettoria eseguita durante l'azione di scansione del workspace

Tale maglia viene trattata come una matrice che riporta i valori della quota degli ostacoli che si trovano al di sotto. Questo valore della quota viene determinato sottraendo, dal valore della quota Z a cui si trova l'*end effector* nell'istante di acquisizione, la distanza rilevata con il sensore a ultrasuoni presente a bordo.

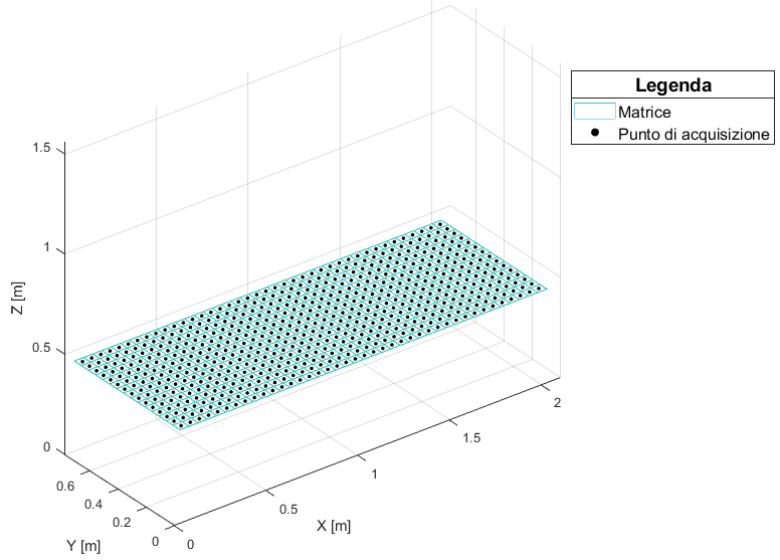

Figura 5.2 Matrice dei punti di acquisizione della distanza degli ostacoli

Lo scopo della scansione è quello di ottenere, mediante l'utilizzo del sensore, le quote degli eventuali ostacoli presenti nello spazio o in ogni caso ricavare la conformazione geometrica dello spazio sottostante.

$$Quota\ Ostacolo_{(x,y)} = \frac{\sum (Z_{EE\,(x,y)} - Distanza_{\,sensore})}{N_{misurazioni}} \quad [\mathrm{m}] \quad [\mathrm{Eq.5.1}]$$

Il rilievo prevede che l'*end effector* si arresti nelle coordinate che identificano il punto di acquisizione e che rimanga in tale posizione per un tempo pari a 2 s prima di spostarsi verso il punto di acquisizione successivo.
Mediamente in questo intervallo di tempo vengono effettuate e inviate al *master* mediante *CAN BUS* 22 misure della quota posseduta dall'ostacolo sottostante.
I dati ottenuti in ogni singolo punto di acquisizione sono stati poi analizzati calcolando media, mediana e deviazione standard e i risultati vengono riproposti nelle successive figure.
In figura 5.3 vengono riportati i valori medi delle quote rilevate durante la scansione.
La figura 5.4 riporta il dato della deviazione standard per mostrare la dispersione registrata durante le operazioni di misura.
Per ottenere l'immagine finale della mappa contenente i vasi è stata effettuata un'interpolazione tra i singoli punti rappresentanti le quote medie degli ostacoli e una maglia di 0.01 m di lato. Il risultato di questa operazione viene mostrato in figura 5.5.

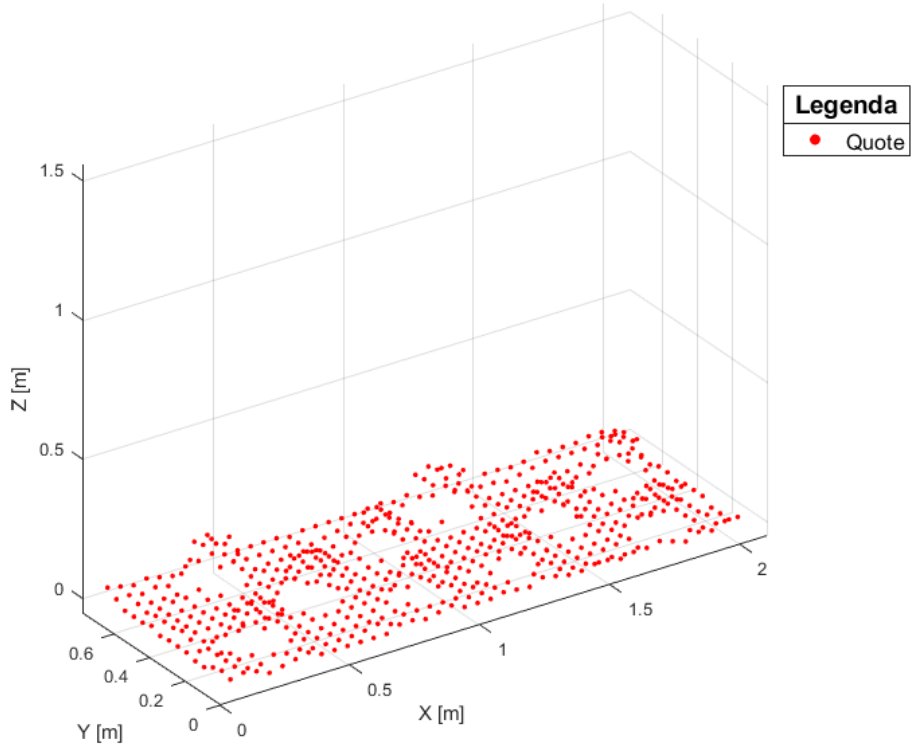

Figura 5.3 Quote medie rilevate durante le acquisizioni

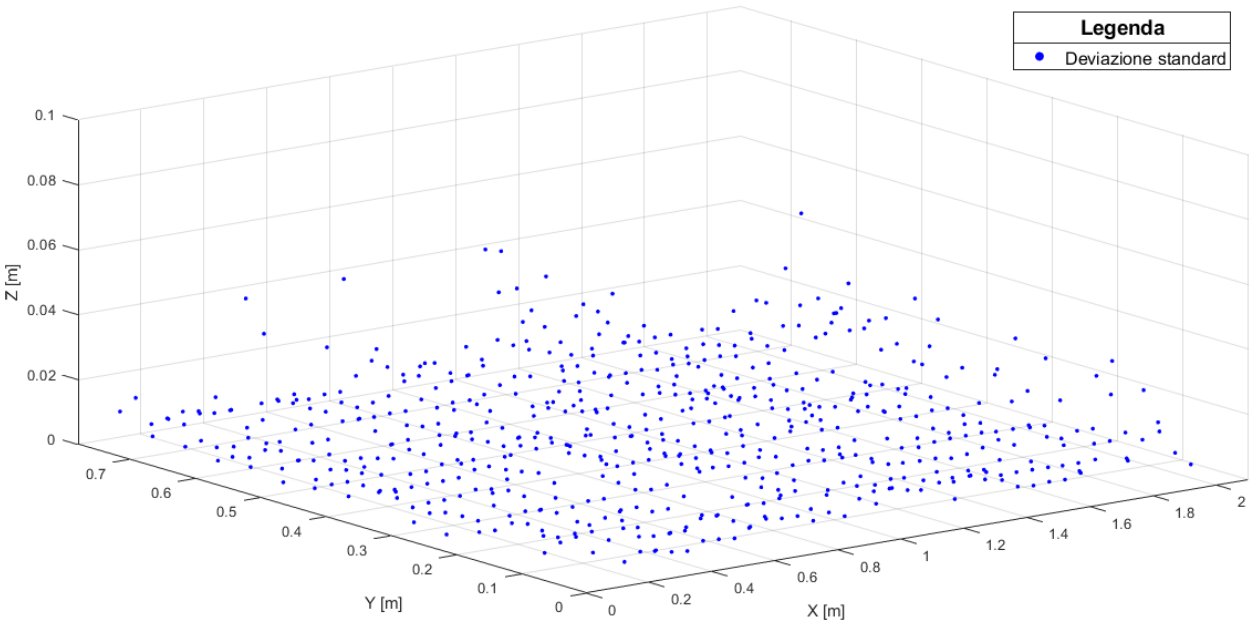

Figura 5.4 Deviazione standard delle quote rilevate

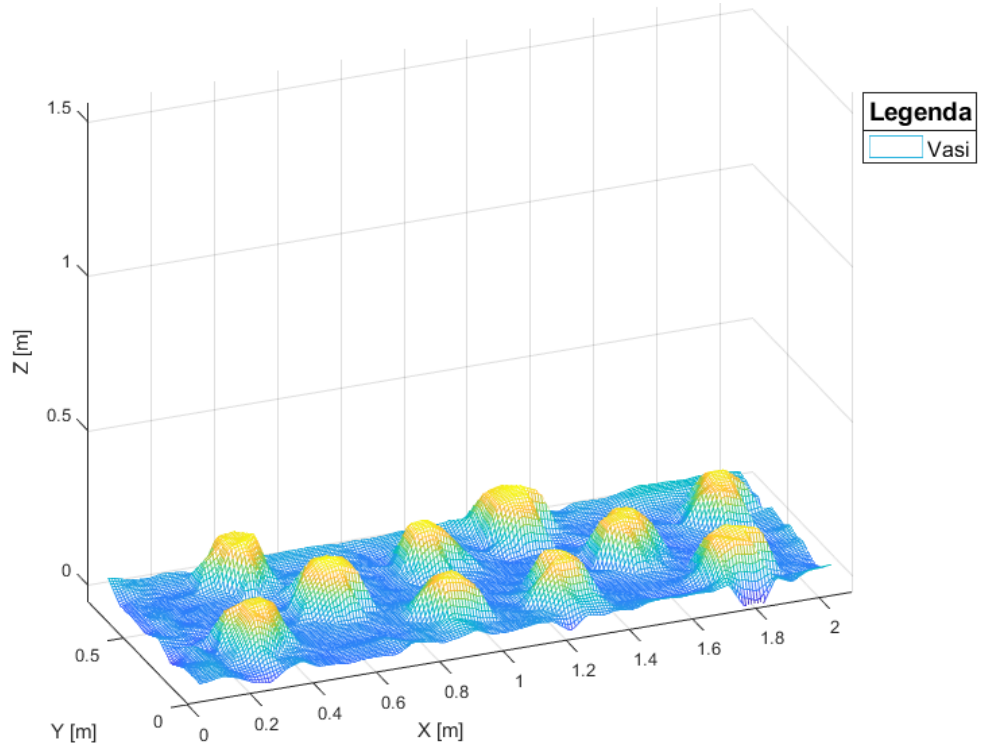

Figura 5.5 Modello della disposizione spaziale dei vasi nel workspace

Dalla figura 5.5 è intuibile la disposizione dei vasi e dunque è possibile, attraverso alcuni procedimenti, stimare la posizione che questi occupano all'interno del volume di lavoro in modo da poter essere raggiunti per compiere alcune operazioni.

Per cercare di determinarne le coordinate sono stati utilizzati due metodi differenti:

- il primo, definito geometrico, prevede la *clusterizzazione* dei punti rilevati mediante la scansione utilizzando un valore soglia per separare i punti appartenenti ai vasi da quelli che fanno parte dello sfondo e successivamente imponendo un altro valore soglia per determinare la distanza massima che possa esistere tra punti appartenenti allo stesso *cluster*.
  In questo modo è possibile determinare il centroide o punto medio dei punti appartenenti ad ogni singolo *cluster* e di conseguenza le coordinate di ogni singolo vaso;

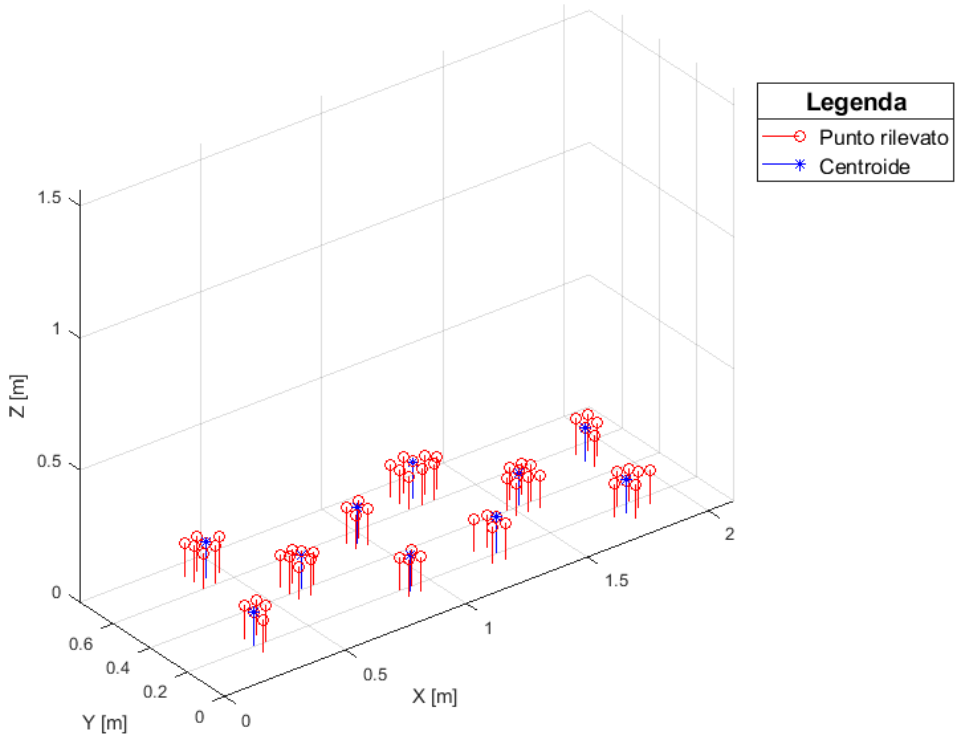

Figura 5.6 Determinazione del centroide tramite clusterizzazione

- il secondo, definito *vision blob analysis* prevede sempre l'applicazione di una soglia di quota minima che devono avere i punti per essere considerati.
  Successivamente viene ricostruita un'immagine compilando una matrice binaria con valore 0 se i punti sono inferiori alla soglia e 1 se questi sono maggiori o uguali.
  La matrice ottenuta in questo modo viene rappresentata come un'immagine e tramite il procedimento di *vision blob analysis* è possibile determinarne il centroide.

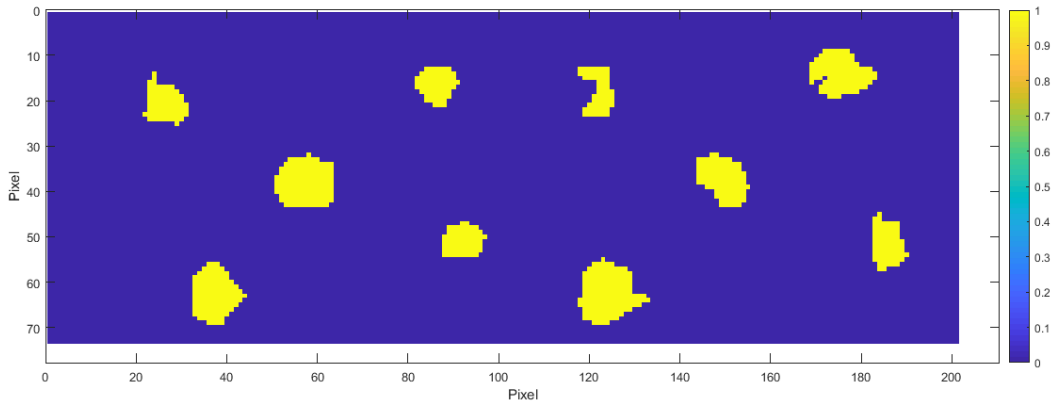

Figura 5.7 Matrice dell'immagine utilizzata per la vision blob analysis

I due metodi sono stati configurati inserendo gli stessi valori soglia e i risultati sono stati rappresentati in un unico grafico (figura 5.8) per renderli confrontabili.

Lo scopo della ricerca dei centroidi è quello di stimare il più accuratamente possibile le coordinate che identificano i singoli vasi, a tale proposito la prova contenuta nel paragrafo seguente servirà a valutare l'accuratezza dei due diversi metodi di calcolo.

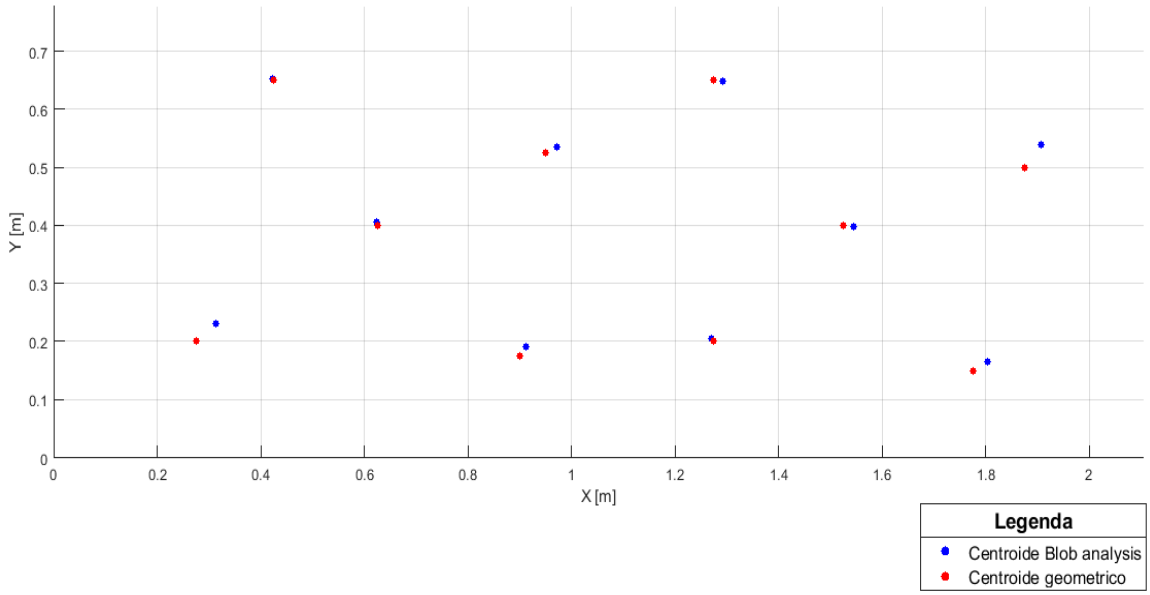

Figura 5.8 Confronto tra le coordinate dei centroidi determinate con il metodo geometrico e tramite la vision blob analysis

5.2 Prove di posizionamento

Lo scopo delle prove di posizionamento è duplice: da un lato la prima prova permette di determinare l'accuratezza dei metodi di calcolo esposti nel paragrafo precedente, utili a determinare le coordinate spaziali dei centroidi dei vasi disposti all'interno del *workspace*; dall'altro, la seconda prova permette di valutare l'accuratezza di posizionamento della macchina.

5.2.1 *Determinazione dell'accuratezza di individuazione dei centroidi*

La prova è stata allestita ricoprendo con dei dischetti di carta la parte superiore dei vasi, segnando al di sopra di questi il punto corrispondente al centro del vaso e determinandone le esatte coordinate spaziali mediante misura diretta.
Ai fini della valutazione dell'accuratezza dei due metodi, sono state ripetute 10 volte consecutive e per ognuno dei 2 metodi, le traiettorie passanti per i punti aventi le coordinate determinate in precedenza facendo arrestare l'*end effector* in corrispondenza di questi per un intervallo di tempo necessario ad appuntarne la posizione raggiunta. durante la sosta. In figura 5.9 viene rappresentata a titolo di esempio una singola traiettoria passante per i punti indicati.

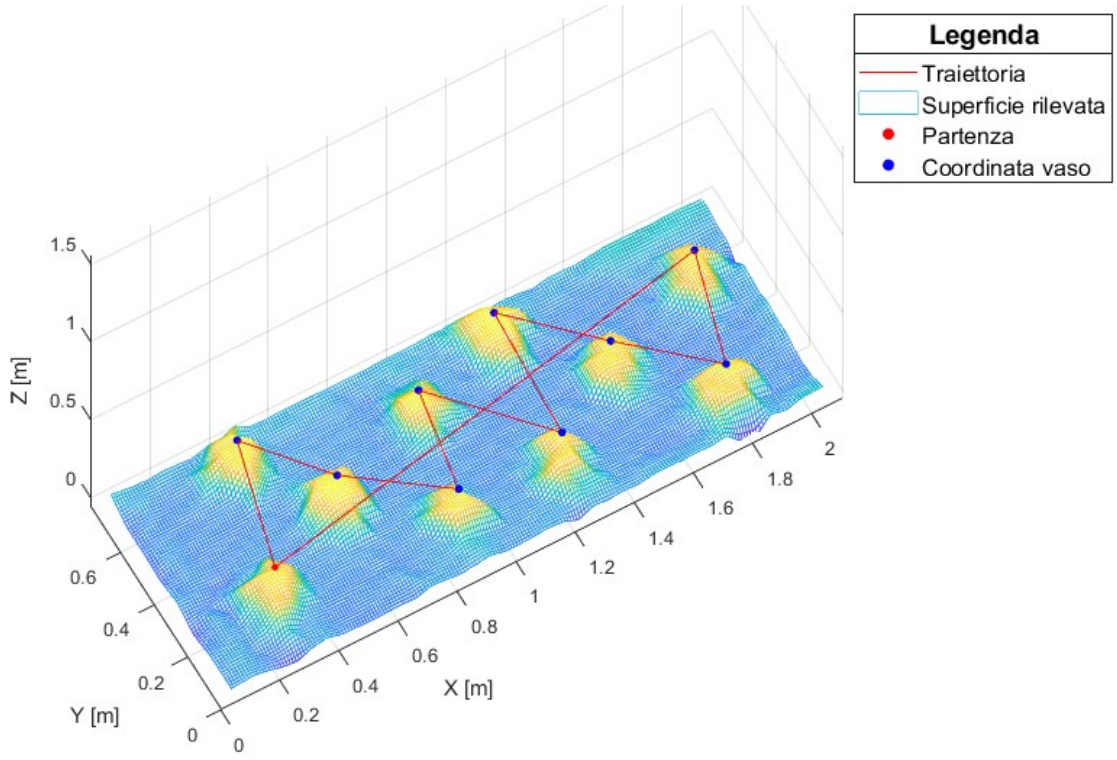

Figura 5.9 Esempio di traiettoria di avvicinamento ai singoli vasi

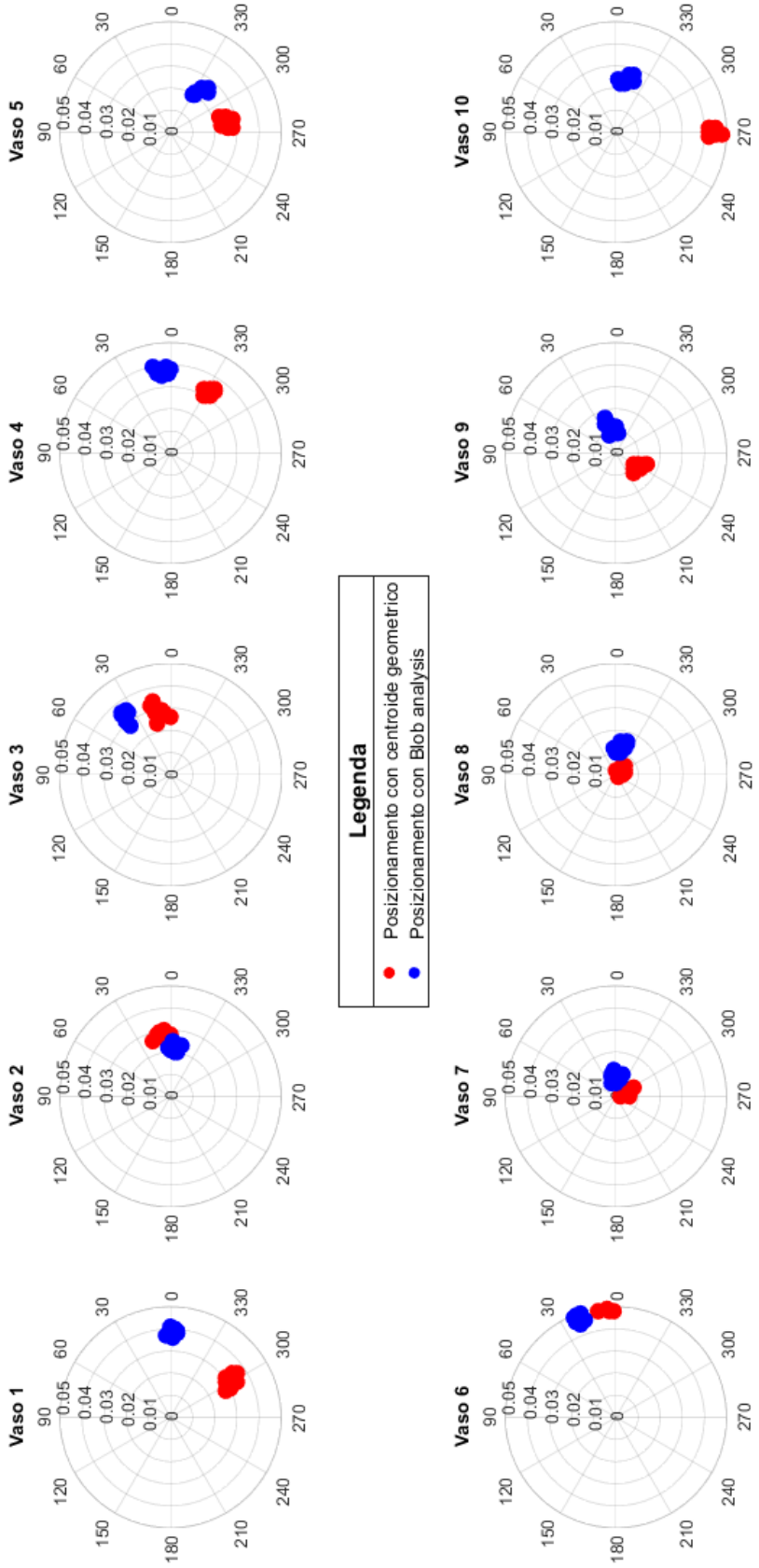

Figura 5.10 Risultati valutazione dell'accuratezza dei modelli di calcolo dei singoli centroidi

I risultati della prova di determinazione dell'accuratezza dei due metodi sono riportati in maniera sintetica in figura 5.10 dove, per ogni vaso, vengono raffigurati gli errori di posizionamento registrati durante i passaggi effettuati.
Il centro di ogni singolo *subplot* rappresenta la collocazione spaziale effettiva del centro del vaso considerato.
Inoltre, per rendere più facilmente interpretabili i risultati, i dati sono stati rappresentati tramite distribuzione di frequenza con gli istogrammi riportati in figura 5.11.

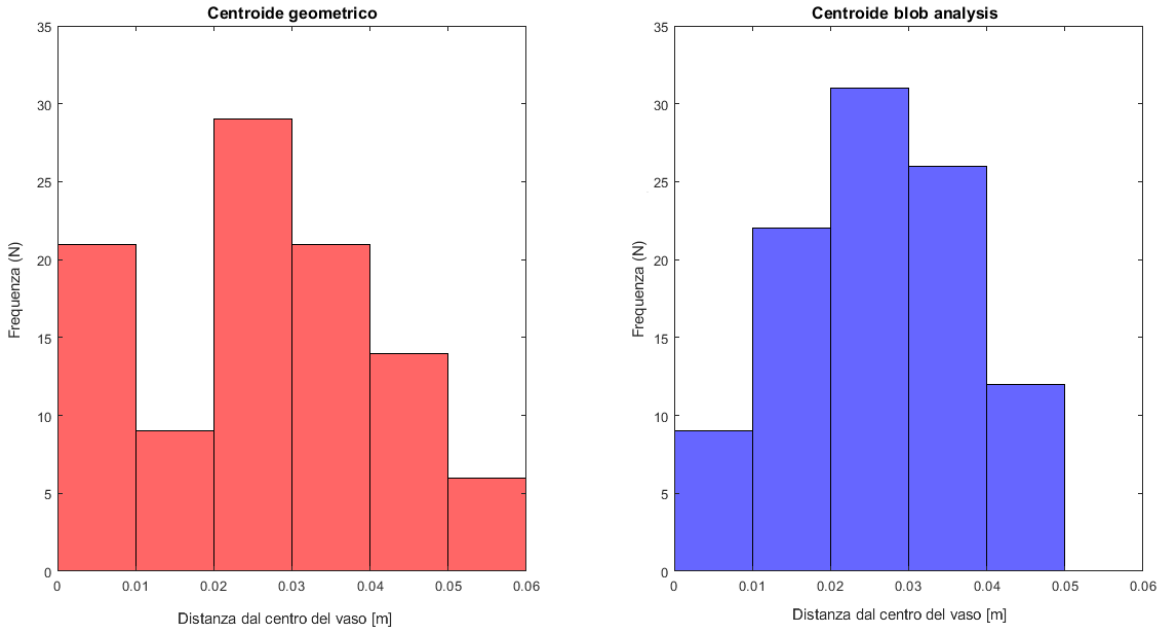

Figura 5.11 Distribuzione degli errori di posizionamento rispetto alla stima dei dei centroidi

Tabella 5.1 Risultati dell'accuratezza di determinazione del posizionamento dei vasi con metodo di calcolo geometrico e tramite vision blob analysis

	Distanza da centroide [m]	
Metodo geometrico	0.026 ± 0.015	N=100
Vision Blob analysis	0.026 ± 0.012	N=100

Da una prima analisi, considerando anche le informazioni presenti in tabella 5.1, si può dedurre che i due metodi di rilevazione, anche se hanno individuato le posizioni spaziali dei vasi in modo differente, hanno comunque fornito un grado di accuratezza medio equivalente con un indice di dispersione leggermente inferiore nel caso della *blob analysis*.
Il grado di accuratezza si può definire soddisfacente specialmente se si considera la qualità e le caratteristiche tecniche del sensore utilizzato per effettuare il rilievo, dispositivo impiegato molto spesso esclusivamente come sensore di prossimità.

5.2.1 *Determinazione dell'accuratezza di posizionamento dell'end effector*

La prova di determinazione dell'accuratezza di posizionamento è stata allestita ponendo dei dischetti di carta il cui centro si trova in posizioni con coordinate note.
Sono state effettuate due prove, definite esperimento 1 ed esperimento 2 che prevedono 100 posizionamenti ciascuna.
Ogni volta che l'*end effector* si è arrestato in una delle posizioni note è stata annotata la posizione raggiunta necessaria successivamente per determinare lo scostamento tra il punto noto e quello raggiunto.
I risultati, distinti per esperimento, sono riportati in figura 5.12.

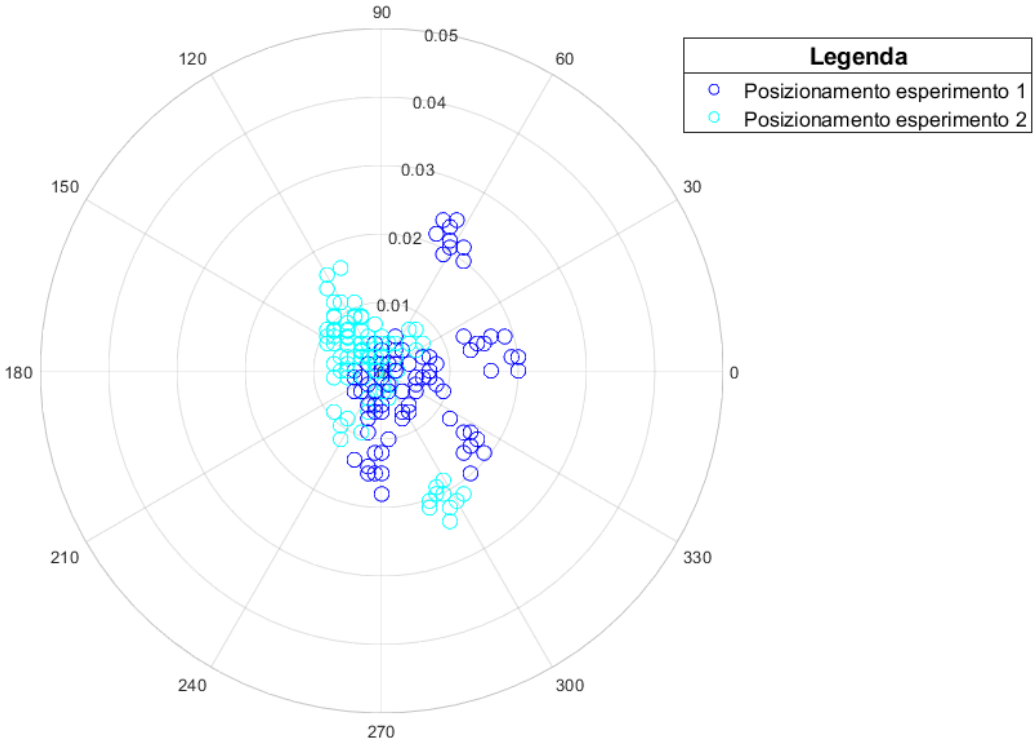

Figura 5.12 Risultati degli esperimenti per la determinazione dell'accuratezza di posizionamento

I dati ottenuti in queste prove sono stati poi rielaborati considerando le distribuzioni di frequenza per ogni esperimento singolarmente e globalmente come unico esperimento con lo scopo di verificare la distribuzione complessiva dei dati, i risultati sono riportati in figura 5.13.

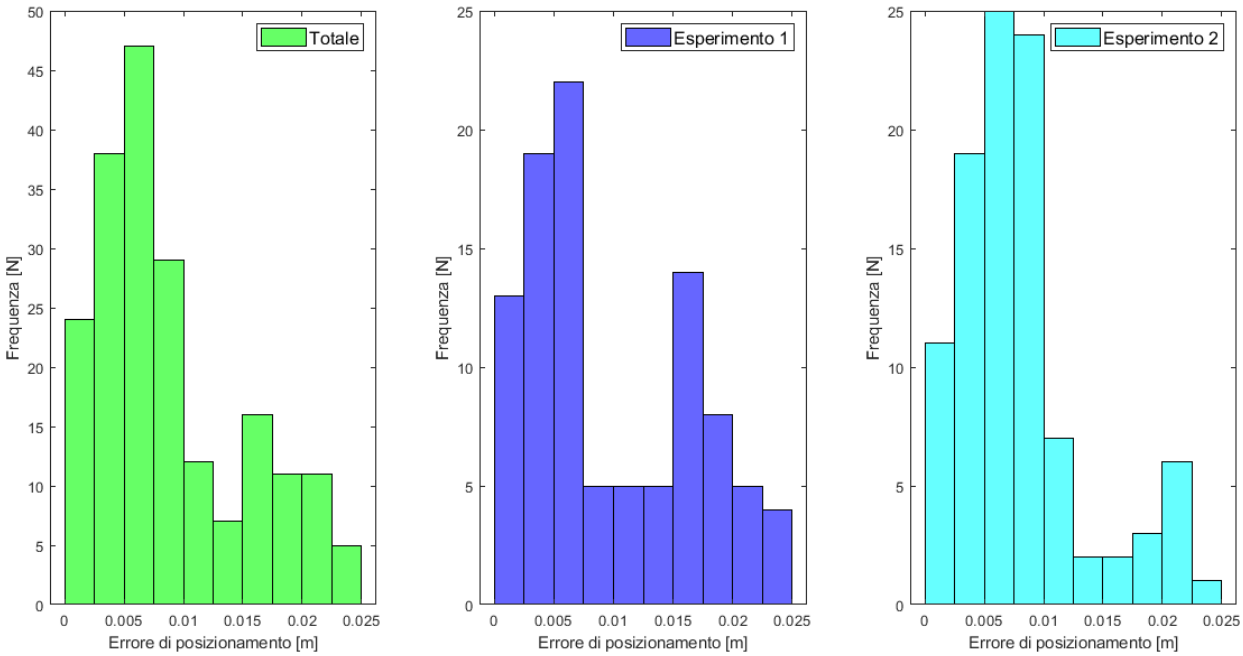

Figura 5.13 Distribuzione degli errori di posizionamento delle prove di posizionamento

Tabella 5.2 Risultati delle prove di accuratezza del posizionamento

	Distanza da posizione esatta [m]	
Esperimento 1	0.010 ± 0.007	N=100
Esperimento 2	0.008 ± 0.005	N=100
Globale	0.009 ± 0.009	N=200

Da un'analisi dei dati globali si evince che circa il 70% degli errori di posizionamento sono inferiori al centimetro.

L'andamento delle distribuzioni mostra che una parte dei posizionamenti sono caratterizzati da errori maggiori e dunque, per capire quale possa essere la ragione, è stato analizzato il grado di accuratezza di posizionamento rispetto al tempo di funzionamento della macchina.

Questo tipo di analisi permette di verificare se tale errore sia dovuto a una diluizione dell'accuratezza degli *encoders* rispetto al momento della calibrazione o se l'errore sia dovuto ad altre ragioni.

In figura 5.14 vengono riportati gli errori di posizionamento in funzione del tempo di utilizzo suddivisi per esperimento.

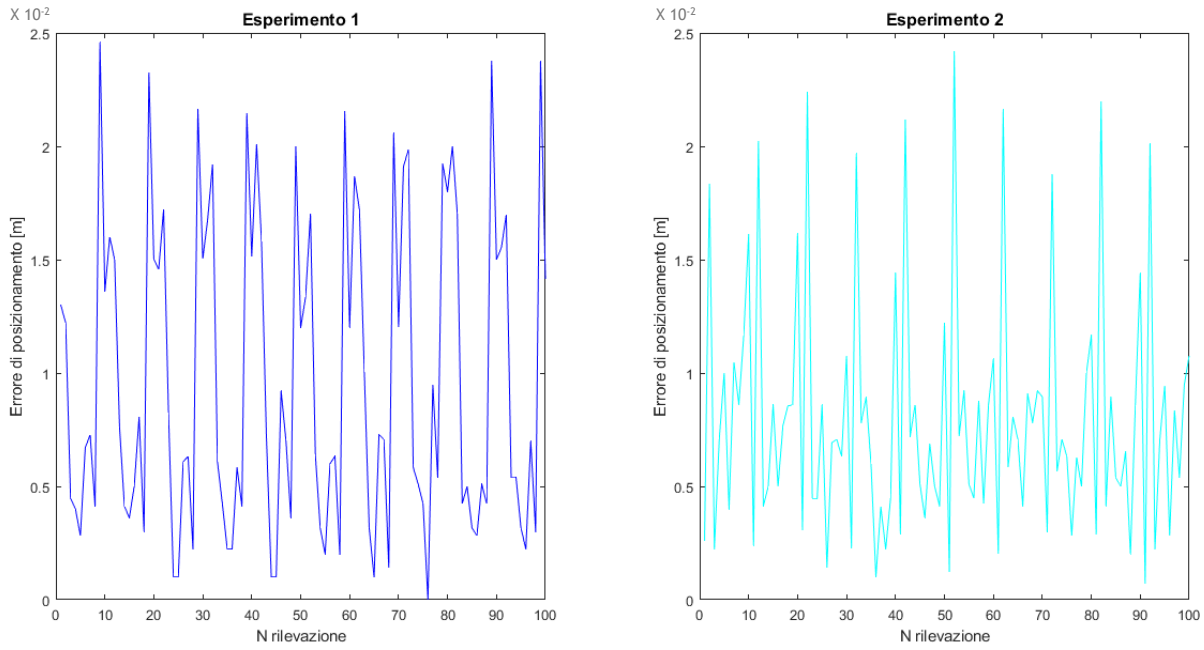

Figura 5.14 Errore di posizionamento in funzione del tempo di utilizzo

Analizzando i grafici in figura 5.14, si nota come l'errore di posizionamento non cambi significativamente in funzione del tempo di utilizzo ma che varia ciclicamente.
Questo fenomeno potrebbe essere legato alla regione di *workspace* dove si trova il *target* da raggiungere e dunque la verifica successiva ha reso necessario considerare l'errore medio spazialmente correlato ad ogni punto dove l'*end effector* è stato fatto arrestare.
Il risultato di questa analisi è riportato in figura 5.15 ed è stato ottenuto mediante la realizzazione di un grafico che rappresenta una superficie di interpolazione degli errori medi di posizionamento spaziali.
Si nota in questo modo come l'errore di posizionamento dipenda dalla regione di *workspace* dove si trova il punto da raggiungere, infatti, le zone critiche o caratterizzate da errori maggiori sono quelle maggiormente vicini ai bordi del volume di lavoro.
Questo "effetto bordo" è sempre presente ed è maggiore in questi esperimenti per via del fatto che l'*end effector* utilizzato per le prove fosse dotato di una massa molto ridotta (0.03 Kg) e che la quota a cui sono stati posti i punti da raggiungere fosse di soli 0.18 m ovvero in una regione di *workspace* caratterizzata da posizioni che assicurano ridotti livelli di tensione nei cavi e quindi maggiori problemi per l'accuratezza di posizionamento e navigazione specie se non viene utilizzato, come in questo caso, l'algoritmo di correzione della lunghezza del cavo dovuto all'effetto catenaria.
In questo caso, considerata la massa ridotta dell'*end effector* utilizzato per la prova, sarebbe stato probabilmente necessario utilizzare tale metodo di correzione per avere un grado di accuratezza di posizionamento maggiore.
Nonostante le condizioni potenzialmente critiche, il grado di accuratezza del

posizionamento è da definirsi comunque soddisfacente in buona parte del volume di lavoro specie nelle aree centrali dove si ottiene un'accuratezza media inferiore al centimetro. Nelle aree prossime ai bordi l'accuratezza diminuisce e l'errore assume una dimensione mediamente doppia rispetto alle aree centrali.

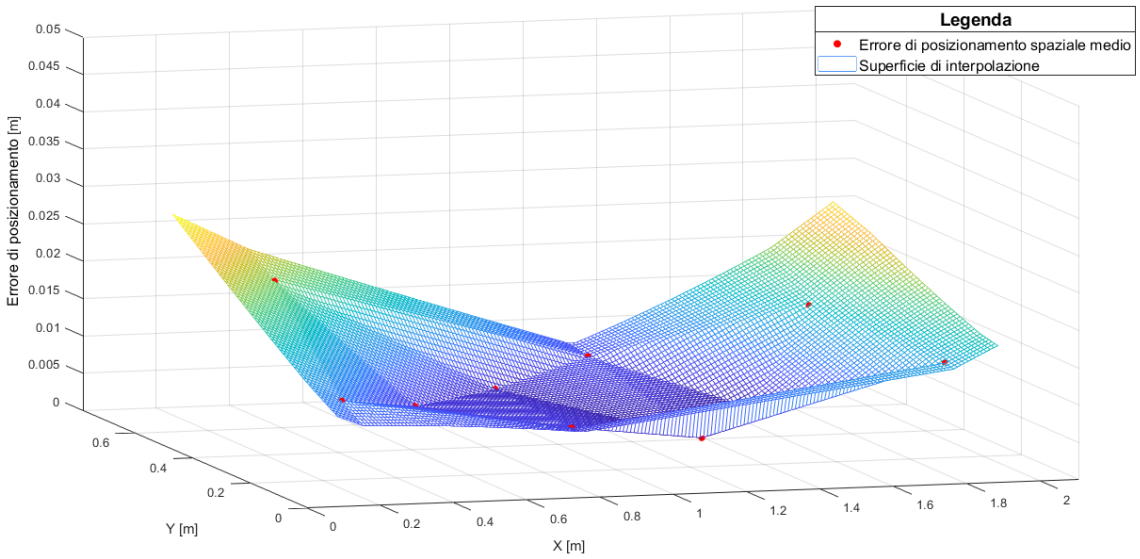

Figura 5.15 Errore di posizionamento medio spaziale

6 PROVE DI SERRA

6.1 Obiettivi delle prove

Le prove in serra hanno lo scopo di verificare l'effettivo funzionamento dell'algoritmo di controllo della cinematica dell'*end effector* e, allo stesso tempo, delle funzioni implementate utili ad eseguire diversi *task*.
Le prove si articolano in due parti principali che consistono:

- nella scansione dello spazio interno al *workspace*, con lo scopo di determinare la conformazione del volume interno di lavoro e a rilevare eventuali ostacoli presenti oltre che, in questo esempio, a stimare le coordinate spaziali dei vasi;
- nell'utilizzo delle informazioni acquisite per impostare un intervento mirato di fertirrigazione sulle piante presenti all'interno del volume di lavoro.

Per lo svolgimento di ogni prova è stato realizzato un *end effector* dedicato allo svolgimento di tale funzione.
Vengono riportate, allo scopo di poter illustrare il *setup* finale del prototipo, le immagini della macchina ottenute durante lo svolgimento delle prove e alcuni dettagli delle variabili fondamentali per il funzionamento del *cable robot*.
La figura 6.1 riporta una vista frontale della macchina, nella quale sono visibili le unità di gestione dei motori e il *master*.
Il *cable robot* è stato inserito all'interno di una delle vasche presenti in serra con alcune plantule di cetriolo in vaso

Figura 6.1 Vista frontale del cable robot con unità motore e master

6.2 Configurazione generale

Le configurazione iniziale prevede l'assemblaggio della macchina e il cablaggio dei componenti principali.
A scopo illustrativo, in figura 6.2 e in figura 6.3, vengono riportati per la prima volta nel testo alcuni componenti fondamentali, necessari al funzionamento del *cable robot*.

Osservando la figura 6.2, partendo da sinistra si possono distinguere: l'*encoder* magnetico; il rocchetto su cui viene avvolto il cavo flessibile; il motoriduttore; il *driver* di alimentazione del motoriduttore e i due microcontrollori con l'interfaccia *CAN*.

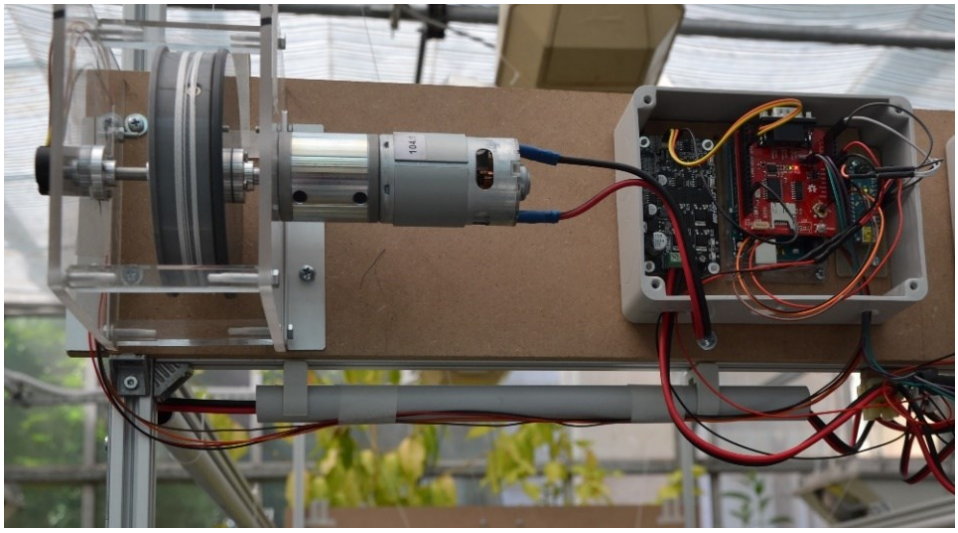

Figura 6.2 Dettaglio dell'unità di gestione del motore

Nella figura 6.3 viene mostrato l'interno del *device* di comando che permette di inserire gli *input* manuali per la selezione delle modalità di funzionamento.
Al suo interno contiene due microcontrollori che garantiscono la connessione e la sincronizzazione delle informazioni che vengono scambiate tra il controllo centrale, su *PC* per mezzo della porta seriale, e la macchina che utilizza come sistema di comunicazione il *CAN*.

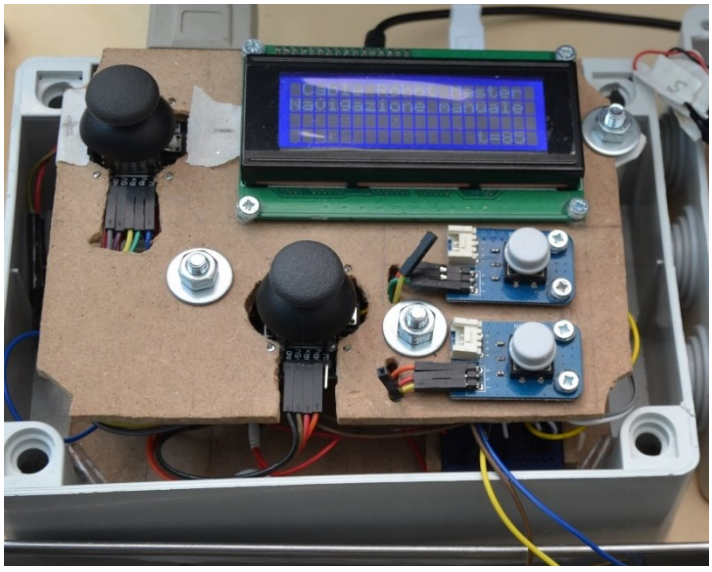

Figura 6.3 Dettaglio del device di controllo generale

La sequenza che avviene a livello del *device* prevede diversi passaggi:

Arduino(1) → *Seriale* → *PC* → *Seriale* → Arduino (1) → I^2C → Arduino(2) → *CAN BUS*

6.3 Scansione del *workspace*

6.3.1 Obiettivo

L'obiettivo della prova di scansione è quello di ottenere una ricostruzione della conformazione dello spazio di lavoro in modo da poter identificare gli ostacoli al moto dell'*end effector* e parallelamente localizzare con precisione eventuali elementi utili.
Lo scopo di questa fase è dunque quello di determinare la posizione delle piante disposte all'interno del volume di lavoro.

6.3.2 Configurazione

Per effettuare la scansione del workspace è stato necessario realizzare un *end effector* dotato di sensore ad ultrasuoni e di un microcontrollore capace di gestire tale sensore oltre che di interfacciare l'unità con il sistema generale mediante *CAN BUS*, descritto maggiormente nel dettaglio nel paragrafo 4.1.6.
La figura 6.4 riporta le immagini dell'*end effector* catturate durante le operazioni di scansione.
La traiettoria necessaria ad effettuare le scansioni è la medesima vista nel capitolo 5 e più precisamente in figura 5.1, che permette di acquisire i dati inerenti alle quote degli ostacoli nei punti mediani della maglia di acquisizione riportata in figura 5.2.

Figura 6.4 Dettaglio dell'end effector durante le scansioni in serra

6.3.3 Risultati

L'operazione di scansione del *workspace* ha reso possibile identificare in maniera puntuale la disposizione spaziale delle piante in vaso, informazioni utili per pianificare la successiva prova di fertirrigazione.
Le coordinate che identificano le singole piante sono state stimate applicando il metodo geometrico visto nel capitolo precedente. In questo caso prevede la clusterizzazione dei punti con quote superiori alla soglia di 0.25 m e viene considerato il parametro di 0.15 m come soglia per la clusterizzazione.
Questi parametri sono stati ipotizzati tenendo conto della taglia e del portamento delle piante presenti nel volume di lavoro.

In figura 6.5 sono riportati graficamente i risultati delle operazioni di scansione e clusterizzazione del *workspace* che hanno individuato con successo le posizioni delle 9 piante in vaso disposte nel volume di lavoro.

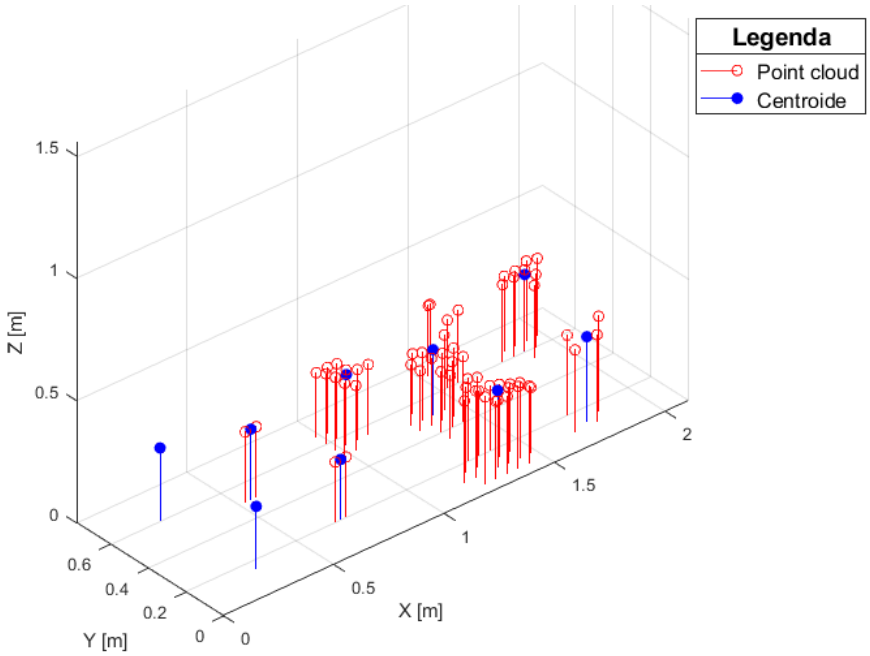

Figura 6.5 Risultato della stima dei centroidi delle piante in vaso con l'utilizzo del metodo geometrico

Per concludere il paragrafo è necessario precisare che l'esito della scansione, anche se complessivamente è da ritenersi più che soddisfacente, è stato ottenuto con un sensore che generalmente non viene utilizzato per tale scopo ma solo come sensore di prossimità date le caratteristiche tecnologiche proprie. Per questo motivo, utilizzando un sensore con performance superiori e in grado di effettuare misure puntuali sarebbe possibile ottenere un'accuratezza maggiore del rilievo oltre che ulteriori livelli informativi.

6.4 Intervento di fertirrigazione di precisione

6.4.1 Obiettivo

L'obiettivo dell'intervento di fertirrigazione è quello di utilizzare l'*end effector* preposto a tale operazione e le informazioni ottenute per mezzo della scansione per raggiungere le piante ed erogare loro una quantità di soluzione che possa essere differente a seconda delle esigenze delle singole piante.

6.4.2 Configurazione

Partendo dalle informazioni provenienti dalla stima dei centroidi delle piante si è potuto definire la traiettoria utile ad effettuare l'intervento di irrigazione di precisione.
La traiettoria impostata segue l'ordine di posizionamento delle piante lungo l'asse X.
Per quanto riguarda la determinazione del punto di arresto dell'*end effector* in prossimità della singola pianta viene utilizzato il dato del centroide per localizzare le singole piante sul piano identificato dagli assi X e Y, mentre per la quota Z viene utilizzato il dato della quota massima rilevata in tale punto maggiorata di una certa soglia di sicurezza (0.05 m in questo caso). per il posizionamento lungo viene riportata in figura 6.6.

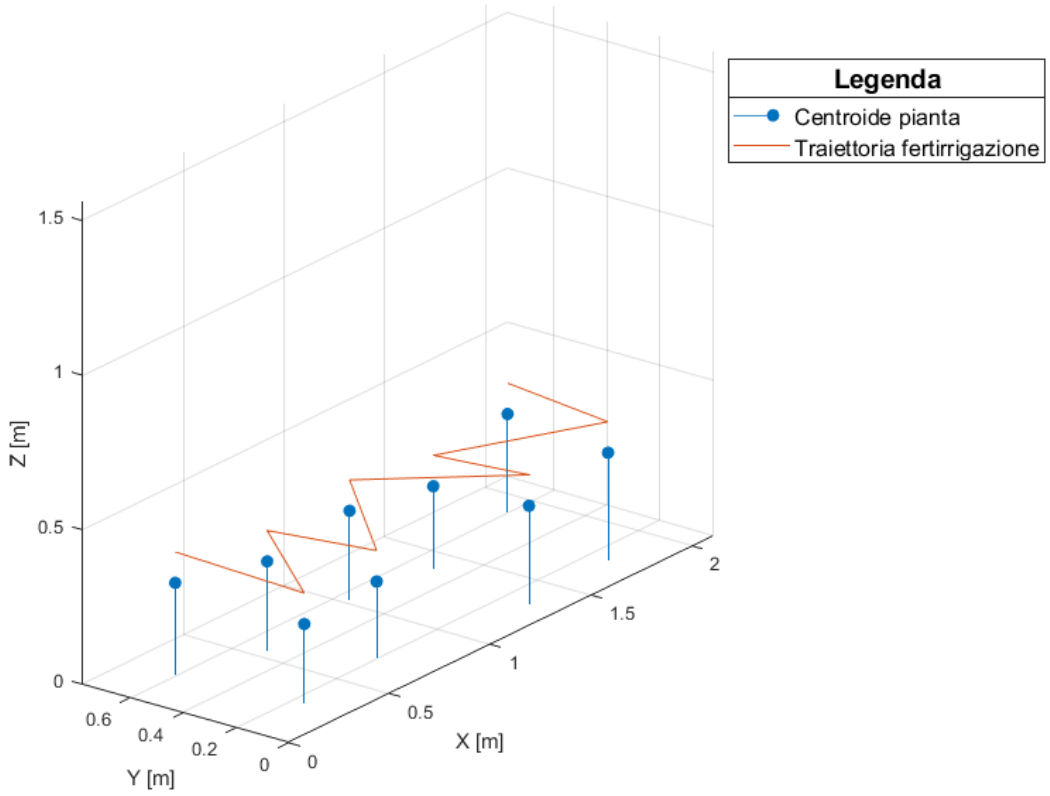

Figura 6.6 Traiettoria intervento di fertirrigazione di precisione

L’azione di erogazione della soluzione prevede dunque una sosta dell’*end effector* in prossimità della pianta con una durata proporzionale alla quantità di liquido da erogare, determinata considerando la portata della pompa. In questo caso si è scelto di somministrare un volume di 0.1 L (1×10^{-4} m^3) che considerata la portata della pompa necessita di un tempo di sosta pari a circa 60 s.

Nella figura 6.7 viene mostrato il microcontrollore, anch’esso nodo della rete *CAN*, che si occupa di gestire l’irrigazione comandando la pompa secondo le istruzioni derivanti dall’algoritmo centrale.

La figura 6.8 e 6.9 riportano le immagini dell’*end effector* utilizzato durante la prova di fertirrigazione che consiste di una piattaforma stabilizzata e di una tubazione in silicone che collega l’ugello terminale alla pompa peristaltica. Nella figura 6.9 è riportato l’intervento di fertirrigazione delle piante rilevate con la scansione riportata in figura 6.6.

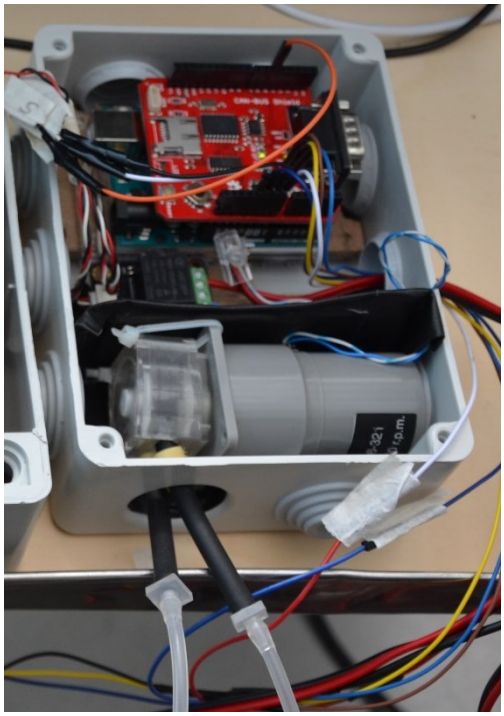

Figura 6.7 Dettaglio del nodo CAN che gestisce la pompa per l’irrigazione

Figura 6.8 Operazione di fertirrigazione su piante di cetriolo in vaso

6.4.3 *Risultati*

L'intervento di fertirrigazione ha avuto esito positivo, l'*end effector* è riuscito a raggiungere le piante e a erogare correttamente la quantità di soluzione prescritta. Nella figura 6.9 viene mostrato l'*end effector* utilizzato durante lo svolgimento dell'operazione di fertirrigazione pianificata.

Figura 6.9 Intervento di fertirrigazione sulle piante di cetriolo in vaso

6.5 *Datalog*

Per completezza, è stato ritenuto appropriato allegare all'esposizione delle prove al fine di fornire ulteriori informazioni riguardanti il funzionamento, alcuni grafici che riportano variabili chiave che provengono dalle acquisizioni dei dati durante il funzionamento.
I dati si riferiscono all'operazione di scansione del *workspace* presente in questo capitolo.

Nella figura 6.10 viene riportato un particolare del segnale, proveniente dall'*end effector* per mezzo del *CAN BUS*, che rappresenta la distanza rilevata mediante il sensore ad ultrasuoni.
Lo scopo di tale figura è di fornire la possibilità di poter visualizzare la sensibilità e la risoluzione temporale del segnale.
In questo caso la variabile della distanza è rappresentata con una risoluzione di 1 *Byte* in grado di rappresentare valori che vanno da 0 a 2,55 m (0 → 255).

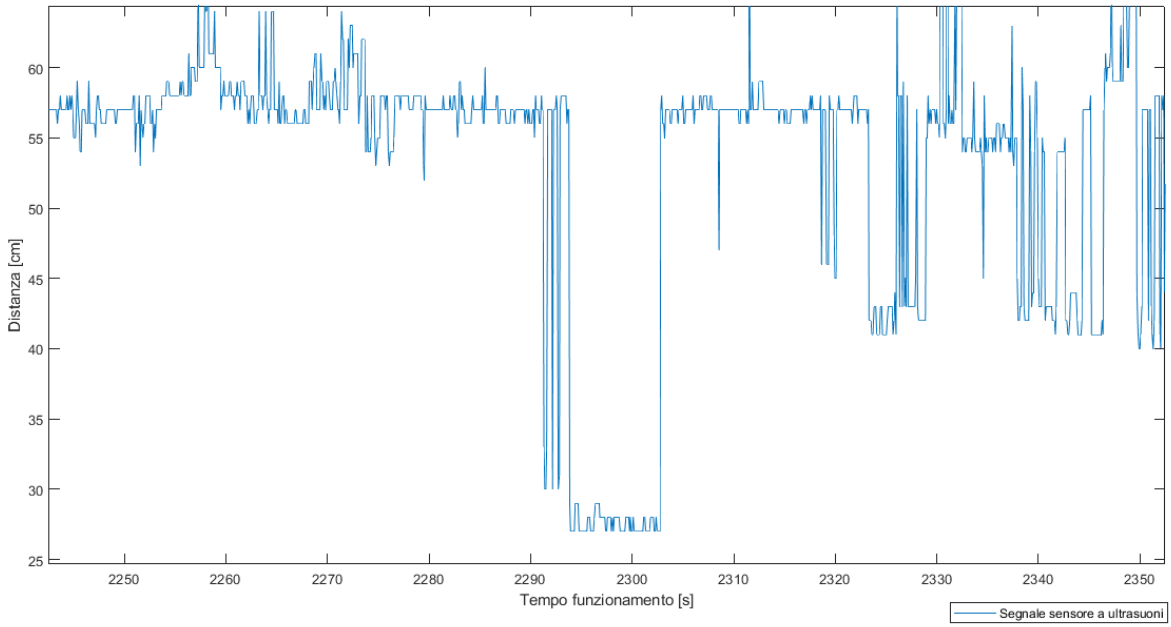

Figura 6.10 Segnale CAN che rappresenta la distanza rilevata dal sensore a infrarossi.

La figura 6.11 riporta, invece, il dato dell'errore di posizionamento calcolato per differenza tra il valore ottimale del *target* e il valore che deriva dal *feedback* degli *encoder*.
Va ricordato che l'errore riportato durante le fasi di movimento dei motori potrebbe essere sovrastimato per via del ritardo che ha il segnale di *feedback* derivante dagli *encoders* causato della minore frequenza di aggiornamento.
La minore frequenza di aggiornamento è dovuta principalmente a due motivi, ovvero: la minore priorità di trasmissione che hanno i messaggi contenenti questi dati sul *CAN BUS* e per il fatto che gli *slave* fanno un tentativo di trasmissione ogni 50 ms.

Questo ritardo non influisce sul corretto funzionamento perché il dato in ritardo riguarda solo il datalog su PC mentre a livello dell'unità *slave* l'aggiornamento del conteggio dell'*encoder* è pari al tempo di esecuzione del loop di controllo interno allo *slave* che è nell'ordine del millisecondo.

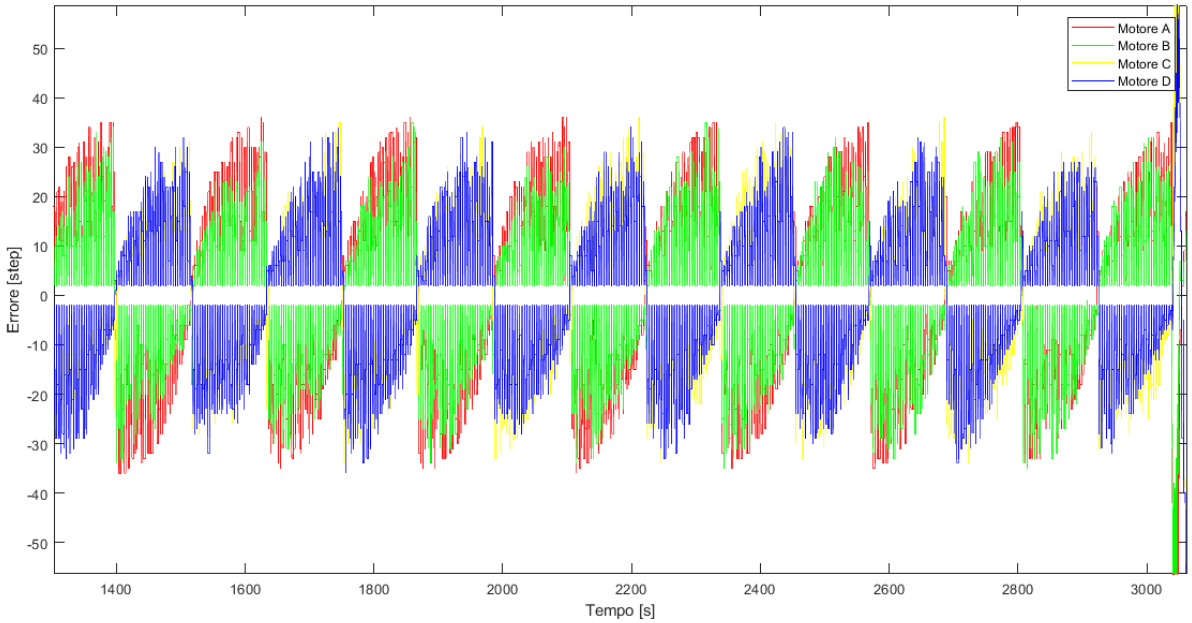

Figura 6.11 Datalog dell'errore di posizionamento

Il dato appena presentato deriva dal *datalog* della scansione e lo si evince dalla ripetitività ciclica e dal fatto che per lungo tempo il dato assume il valore di +2 o -2.
Il numero 2 come è stato usato come soglia per il posizionamento dei motori e si nota come il motore si arresti su un valore positivo o negativo a seconda del senso di rotazione di provenienza e che tale valore si inverta con il senso di rotazione e di marcia durante l'esecuzione delle traiettorie tipiche della scansione.
Da questo dato si può affermare che a livello dei motori l'accuratezza di posizionamento statico è pari a 2 *step* di *encoder* o 0,27 $x10^{-2}$ m.
Si potrebbe pensare dunque di ridurre ulteriormente tale margine abbassando la soglia.

Nelle figure 6.12 e 6.13 vengono riportati, infine, i grafici inerenti al del valore del *target* ottimale e un ingrandimento dello stesso.
Questo valore rappresenta l'*output* dell'algoritmo di controllo principale proveniente da MATLAB e che scomposto in *Byte* viene inviato agli *slave*.
La figura 6.12 riporta una visione di insieme della variabile inviata tramite *CAN BUS* durante tutta l'operazione di scansione dello spazio di lavoro.

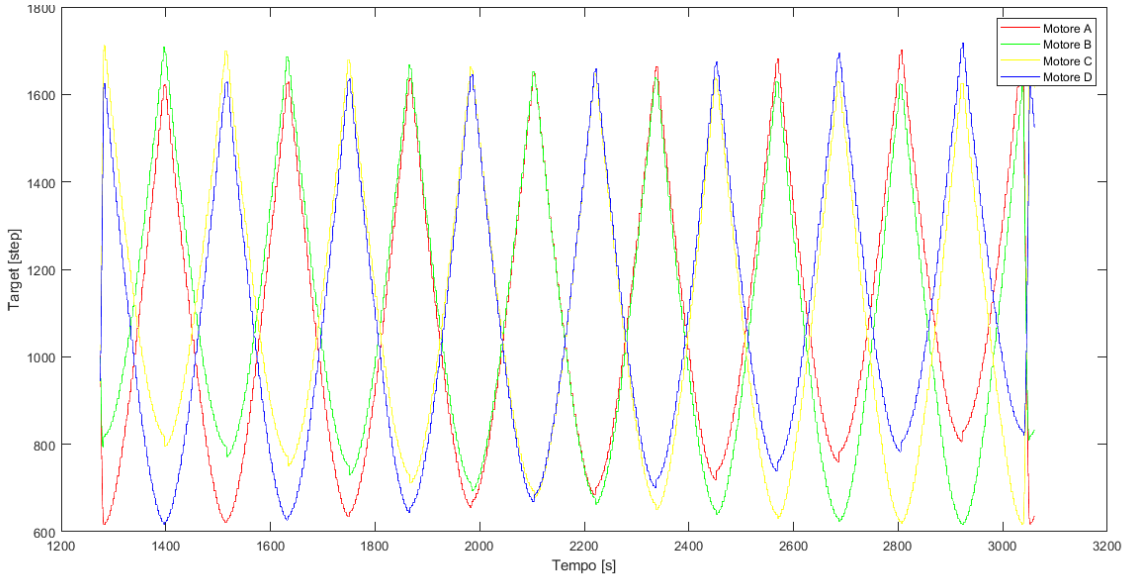

Figura 6.12 Datalog del valore target ottimale

La figura 6.13 riporta un dettaglio della variabile valore del *target* ottimale. Si può notare da questa figura l'andamento segmentato delle curve. Tale andamento è dovuto ai periodi di 2 s dove l'*end effector* si ferma per effettuare le acquisizioni delle quote degli ostacoli. Tale periodo di sosta genera le linee orizzontali simultaneamente in tutti i motori mentre le linee con andamento verticale rappresentanogli spostamenti tra un punto di acquisizione e il successivo.

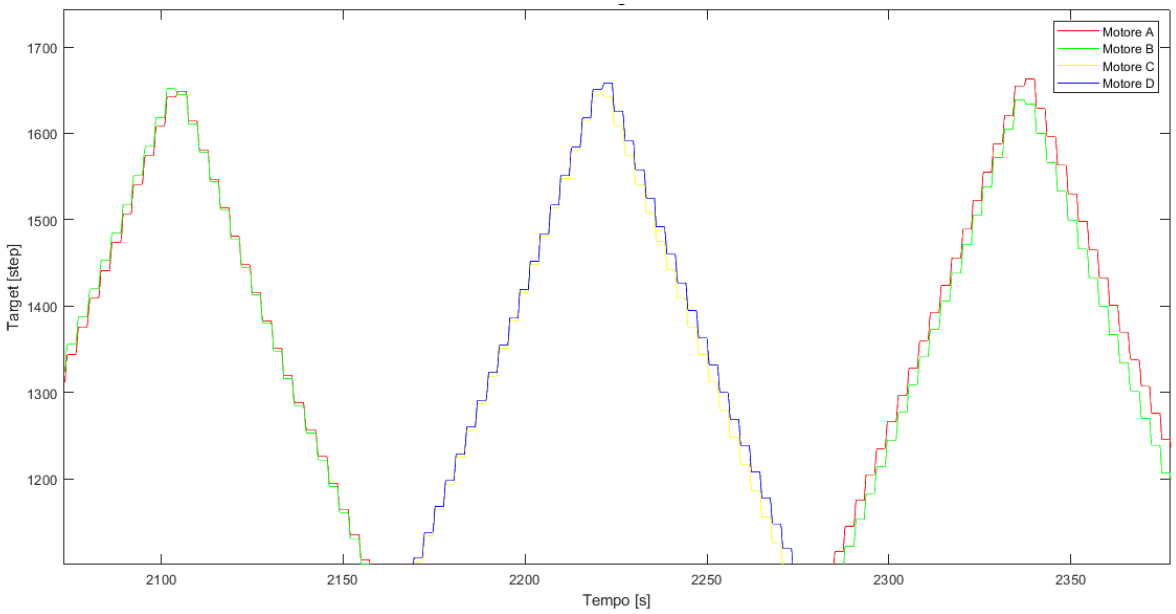

Figura 6.13 Dettaglio del datalog del valore target ottimale

7 CONCLUSIONI E SVILUPPI FUTURI

Il prototipo realizzato è caratterizzato, allo stadio di sviluppo raggiunto, da prestazioni che complessivamente possono essere considerate soddisfacenti in termini di accuratezza di posizionamento, di gestione della dinamica dell'*end effector*, di stabilità e di sincronizzazione generale del sistema.

Complessivamente il sistema, sia nella parte *hardware* che *software*, ha un margine di sviluppo elevato ottenibile con alcune strategie di ottimizzazione emerse durante i test:

- l'architettura potrebbe essere ottimizzata secondo due strategie principali: interfacciando direttamente l'*output* dell'algoritmo centrale in *MATLAB* con la rete *CAN* consentendo la trasmissione diretta dei dati da e per l'algoritmo principale eliminando di fatto la comunicazione mediante porta seriale, che è quella che si è rivelata critica sia per la sincronizzazione che per la stabilità del sistema; oppure trasferire direttamente l'algoritmo di controllo centrale della cinematica su un microcontrollore con prestazioni superiori ad Arduino Uno e utilizzando *MATLAB* per la gestione degli *end effector*, la raccolta e la rielaborazione dei dati rilevanti;
- dotare l'*end effector* di: batteria di alimentazione, modulo di comunicazione *wireless* (*XBee)* per consentire il funzionamento senza il vincolo di avere un cavo per alimentazione e trasferimento dei dati; installare un giroscopio a bordo per stabilizzare l'*end effector* o garantirne la stabilità per mezzo di servomotori;
- provare diversi arrangiamenti dei cavi per valutarne l'effetto in termini di stabilità e accuratezza di moto dell'*end effector*;
- ampliare le dimensioni di massima del *workspace*;
- ottimizzare e sviluppare il *software* ampliando il numero di funzioni possibili tra cui la possibilità di evitare eventuali possibili ostacoli in *real-time* durante la modalità di navigazione automatica;
- aumentare la qualità di alcuni sensori;
- utilizzare celle di carico per determinare la tensione effettiva dei cavi e applicare le dovute correzioni alle lunghezze dei cavi;
- progettare un sistema di ricalibrazione dinamica.

8 RINGRAZIAMENTI

Dopo lunghi e intensi mesi di lavoro che sono stati necessari per la progettazione, la realizzazione e l'ottimizzazione del funzionamento del *cable robot* è arrivato il momento di chiudere, insieme alla tesi, questo periodo di apprendimento personale e scientifico. E' necessario anche spendere due parole di ringraziamento nei confronti di tutte le persone che mi hanno sostenuto e aiutato durante questo periodo.

Un ringraziamento particolare va al relatore, il professore Oberti, che ha creduto da subito nella possibilità di realizzare l'idea di questo progetto nato durante il corso di *farm automation* e che ha svolto, durante tutto questo tempo, il ruolo di guida e ispirazione anche nell'indicare le metodologie di lavoro più appropriate da seguire.

Ringrazio l'Ing. Naldi che ha dispensato numerosi consigli per quanto riguarda la parte informatica ed elettronica del progetto e specialmente per la parte della scomposizione in byte.

Ringrazio anche in maniera speciale Anna e i miei genitori che mi hanno sostenuto e sopportato in questi anni di studio.

9 ELENCO DELLE FIGURE

10 ELENCO DELLE TABELLE

11 ELENCO DELLE EQUAZIONI

12 BIBLIOGRAFIA

12.1 Fonti bibliografiche

- Abdolshah S. and Barjuei E. S., "Linear quadratic optimal controller for cable-driven parallel robots." Frontiers of Mechanical Engineering 10.4 (2015): 344-351.
- Babaghasabha R., Khosravi M. A. and Taghirad H.D., "Adaptive robust control of fully-constrained cable driven parallel robots." Mechatronics 25 (2015): 27-36.
- Barbazza L. et al., "Trajectory planning of a suspended cable driven parallel robot with reconfigurable end effector." Robotics and Computer-Integrated Manufacturing 48 (2017): 1-11.
- Bocus S.,"Ponti strallati. Problemi di progettazione degli stralli.", PhD Thesis (2012).
- Borgstrom P. H. et al., "Rapid computation of optimally safe tension distributions for parallel cable-driven robots." IEEE Transactions on Robotics 25.6 (2009): 1271-1281.
- Boschetti G. and Trevisani A., "Cable robot performance evaluation by wrench exertion capability." Robotics 7.2 (2018): 15.
- De Clercq M., Anshu V. and Alvaro B., "Agriculture 4.0: The Future of Farming Technology." Proceedings of the World Government Summit, Dubai, UAE (2018): 11-13.
- Dragoljub S. and Bernhardt R., "STRING-MAN: a new wire robot for gait rehabilitation." IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. Vol. 2. IEEE, 2004.
- Duan Q. J. et al., "Modeling of variable length cable driven parallel robot." Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. IEEE, 2010.
- Gosselin C. and Foucault S., "Dynamic point-to-point trajectory planning of a two-DOF cable-suspended parallel robot." IEEE Transactions on Robotics 30.3 (2014): 728-736.
- Izard J. B. et al., "A reconfigurable robot for cable-driven parallel robotic research and industrial scenario proofing." Cable-driven parallel robots. Springer, Berlin, Heidelberg, 2013. 135-148.

- Khosravi M. A. and Taghirad H. D., "Dynamic modeling and control of parallel robots with elastic cables: singular perturbation approach." IEEE Transactions on Robotics 30.3 (2014): 694-704.
- Kirchgessner N. et al., "The ETH field phenotyping platform FIP: a cable-suspended multi-sensor system." Functional Plant Biology 44.1 (2017): 154-168.
- Kraus W. et al., "Hybrid position/force control of a cable-driven parallel robot with experimental evaluation." New Trends in Mechanism and Machine Science. Springer, Cham, 2015. 553-561.
- Lahouar S. et al., "Collision free path-planning for cable-driven parallel robots." Robotics and Autonomous Systems 57.11 (2009): 1083-1093.
- Mroz G. and Leila N., "Design and prototype of parallel, wire-actuated robots with a constraining linkage." Journal of Robotic Systems 21.12 (2004): 677-687.
- Oh S.R. and Agrawal S. K., "Generation of feasible set points and control of a cable robot." IEEE Transactions on Robotics 22.3 (2006): 551-558.
- Ottaviano E., "Progettazione ottimizzata di manipolatori paralleli". PhD Thesis. 2001.
- Otto S., and Seifried R., "Real-time Trajectory Tracking of a Cable-driven Parallel Robot Using Servo-constraints." PAMM 17.1 (2017): 161-162.
- Ouyang B. and Weiwei S., "Wrench-feasible workspace based optimization of the fixed and moving platforms for cable-driven parallel manipulators." Robotics and Computer-Integrated Manufacturing 30.6 (2014): 629-635.
- Piao J. et al., "Open-loop position control of a polymer cable–driven parallel robot via a viscoelastic cable model for high payload workspaces." Advances in Mechanical Engineering 9.12 (2017): 1687814017737199.
- Pidatella, Ferrari, Aggradi, “Corso di meccanica, macchine ed energia”, approfondimento sull’equilibrio delle funi, Zanichelli 2012.
- Pott A. et al., "Cable-driven parallel robots for industrial applications: The IPAnema system family." IEEE ISR 2013. IEEE, 2013.
- Pott A. et al., "IPAnema: a family of cable-driven parallel robots for industrial applications." Cable-Driven Parallel Robots. Springer, Berlin, Heidelberg, 2013. 119-134.
- Reicherts S. et al., "Sensitivity Analysis of the Design Parameters for the Calibration of Cable-driven Parallel Robots." PAMM 16.1 (2016): 859-860.
- Riehl N. et al., "On the determination of cable characteristics for large dimension cable-driven parallel mechanisms." 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010.

- Schrijver R., Poppe K. and Daheim C., "Precision agriculture and the future of farming in Europe.". [online]http://www.europarl.europa.eu/RegData/etudes/STUD/2016/581892/EPRS_STU(2016)581892_EN.pdf (2016). Download 01/10/2018.
- Seriani S., Gallina P. and Wedler A., "A modular cable robot for inspection and light manipulation on celestial bodies." Acta Astronautica 123 (2016): 145-153.
- Sito FarmBot visitato il 15/05/2019 [online] https://farm.bot/
- Tadokoro S. et al., "A portable parallel manipulator for search and rescue at large-scale urban earthquakes and an identification algorithm for the installation in unstructured environments." Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289). Vol. 2. IEEE, 1999.
- Tang L. et al., "Dynamic trajectory planning study of planar two-dof redundantly actuated cable-suspended parallel robots." Mechatronics 30 (2015): 187-197.
- Li T., Tang X. and Tang. L., "Algebraic expression and characteristics of static wrench-closure workspace boundary for planar cable-driven parallel robots." Advances in Mechanical Engineering 8.3 (2016): 1687814016638217.
- Tempel P., Miermeister P. and Pott A.. "Kinematics and dynamics modeling for real-time simulation of the cable-driven parallel robot IPAnema 3." Proceedings of the 14th IFToMM World Congress, 2015.
- Trevisani A., Gallina P. and Williams R.L., "Cable-direct-driven robot (CDDR) with passive SCARA support: theory and simulation." Journal of Intelligent and Robotic Systems 46.1 (2006): 73-94.
- Trevisani A., "Underconstrained planar cable-direct-driven robots: A trajectory planning method ensuring positive and bounded cable tensions." Mechatronics 20.1 (2010): 113-127.
- Wei H., Yuanying Q. and, Yu S., "Motion control strategy and stability analysis for high-speed cable-driven camera robots with cable inertia effects." International Journal of Advanced Robotic Systems 13.5 (2016): 1729881416663374.
- Weltzien C., "Digital Agriculture or Why Agriculture 4.0 Still Offers Only Modest Returns." Landtechnik 71.2 (2016): 66-68.
- Zhang N. and Shang W., "Dynamic trajectory planning of a 3-DOF under-constrained cable-driven parallel robot." Mechanism and Machine Theory 98 (2016): 21-35.
- Zi B. et al., "Dynamic modeling and active control of a cable-suspended parallel robot." Mechatronics 18.1 (2008): 1-12.

- Zoppetti N. and Andreuccetti D., Consiglio Nazionale delle Ricerche, Florence (Italy). Istituto di Fisica Applicata 'Nello Carrara'; "Modellazione dei conduttori di un elettrodotto aereo: la catenaria." Report Tecnico IFAC N. TR/AEL/08.03 ISSN (2003): 1120-2823.

11.2 Datasheet

- CANBUS: Microchip MCP2515 “Stand-Alone CAN Controller With SPI Interface” (http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf)
- CANBUS: Microchip MCP2551 “High-Speed CAN Transceiver” (http://ww1.microchip.com/downloads/en/DeviceDoc/21667E.pdf)
- CANBUS: Microchip AN754 “Understanding Microchip’s CAN Module Bit Timing” (http://ww1.microchip.com/downloads/en/AppNotes/00754.pdf)
- CANBUS:http://www.dia.uniroma3.it/autom/Reti_e_Sistemi_Automazione/PDF/CAN%20(Applicazioni%20 Automobilistiche)%20Di%20Galanti%20Antonello.pdf
- Cavo: Dyneema (https://www.dsm.com/products/dyneema/en_GB/home.html Visitato il 03/05/2018)
- Driver Motore: Cytron Technologies MD10C R 3.0 (https://docs.google.com/document/d/1rgQzn-nWn-qcWNnHjDZvIYqUrdCeBQQxXA-TU3BF0AQ/view)
- Encoder: Broadcom Avago Technologies AEAT-601B Incremental Magnetic Encoder (https://docs.broadcom.com/wcs-public/products/data-sheets--technical-specifications/data-sheet/616/15/av02-2176en.pdf)
- I^2C: NXP Semiconductors UM10204 “I^2C-bus specification and user manual” (https://www.nxp.com/docs/en/user-guide/UM10204.pdf)
- Motoriduttore: MFA Como Drills 975D Series 42mm (45mm motor) Planetary (Epicyclic) metal gear box 104:1 (https://www.mfacomodrills.com/pdfs/975D-series.pdf)
- Sensore ultrasuoni: http://elecfreaks.com/store/download/HC-SR04.pdf