POLITECNICO DI MILANO

Facoltà di Ingegneria dei Sistemi

Corso di Laurea Specialistica in Ingegneria Biomedica

DEFINIZIONE E TESTING DI UN PROTOCOLLO DI COMUNICAZIONE E GESTIONE PER UN APPARATO DI POSIZIONAMENTO DI TIPO MODULARE CON GRADI DI LIBERTÀ MULTIPLI PER CHIRURGIA MINI-INVASIVA A SINGOLA PORTA

Relatore: Chiar.mo Prof. Pietro Cerveri

Tesi di laurea di:

Alberto Giovanni Pansini

Matr. 740731

Anno accademico 2010/2011

Sintesi

La robotica ha consentito lo sviluppo di strumenti per la chirurgia mini-invasiva con caratteristiche funzionali non ottenibili con strumenti puramente manuali, favorendo il passaggio dalla chirurgia laparoscopica alla chirurgia robotica mini-invasiva. Il passo successivo è stato quello di eliminare l'incisione, e di sfruttare gli orifizi naturali. Una tecnica rivoluzionaria che potrebbe eliminare il rischio di complicanze associate all'incisione della parete addominale è la Chirurgia Transluminare Endoscopica attraverso Orifizi Naturali (NOTES Natural Orifice Transluminal Endoscopic Surgery).

In questo contesto si inserisce il progetto Snake, seguito dal prof. Cerveri del Dipartimento di Bioingegneria del Politecnico di Milano. L'obbiettivo del progetto è lo sviluppo del prototipo di un endoscopio chirurgico robotizzato per NOTES.

La tesi si inserisce in questo contesto con lo scopo di fornire uno studio del sistema di comunicazione, del controllo e della scelta della componentistica del robot endoscopico "snake". Viene fornita una panoramica dei sistemi robotici endoscopici per chirurgia NOTES attualmente in uso. Viene accennata una descrizione dell'intero robot per poi scendere del dettaglio di ogni componente. Ognuno di questi viene scelto cercando di rispettare le specifiche di progetto.

Viene poi analizzato il sistema di comunicazione. I protocolli utilizzati per lo scopo sono il CAN (Controller Area Network) e l'SPI (Serial Peripheral Interface). Essi vengono utilizzati per fini diversi a seconda dei vantaggi che ognuno propone. Il primo è usato per la comunicazione tra esterno ed interno del robot (PC-microcontrollori) mentre il secondo per la comunicazione tra i microcontrollori e tra i sensori e i micro.

La tesi illustra poi il protocollo di comunicazione da me ideato, e gli schematici da me realizzati.
Viene affrontato il problema del controllo dei motori e dei sensori e come esso è stato risolto utilizzando i due microcontrollori presenti su ogni modulo. Vengono quindi spiegati i punti principali del firmware dei suddetti micro.
Infine vengono descritti i test che sono stati effettuati.

.

Abstract

Robotics has allowed the development of tools for minimally invasive surgery with functional characteristics not obtainable with purely manual tools, supporting the transition from laparoscopic surgery to minimally invasive robotic surgery. The next step was to eliminate the incision, and make use of the natural orifices. A revolutionary technique that could eliminate the risk of complications associated to incision of the abdominal wall is the Natural Orifice Transluminal Endoscopic Surgery (NOTES).

The Snake project was conceived in this background in effort to develop a first prototype of a robotic platform for NOTES. The project is followed by the prof. Cerveri of the Politecnico di Milano.

The thesis fits into this context with the aim of defining a study of the communication system, control and choice of components of the endoscopic robot "snake". The thesis provides an overview of robotic systems for endoscopic surgery NOTES currently in use. It is mentioned a description of the entire robot and then to details of each component. Each of these is chosen trying to observe the design specifications.

The communication system is then analyzed. The protocols used for this purpose are the CAN (Controller Area Network) and the SPI (Serial Peripheral Interface). They are used for different purposes depending on the advantages that each offers. The first is used for communication between the exterior part and interior part of the robot (PC-microcontroller) and the second for communication between the microcontroller and between sensors and micro.

The thesis then describes the communication protocol designed by me, and the schematic I made.

Finally, the problem of motor control and sensors are discussed and then how it was solved using two microcontrollers on each form. The main points of these micro's firmware are explained. At last the thesis describes the tests that were performed.

Indice dei contenuti

Indice delle figure

Indice delle tabelle

Capitolo 1
INTRODUZIONE

1.1 **Contesto clinico**

L'avvento della "chirurgia mini-invasiva" (MIS) ha rappresentato un importante progresso nel campo chirurgico. L'ipotesi tradizionale, che sostiene che i problemi di grandi dimensioni richiedono grandi incisioni chirurgiche per un trattamento adeguato, non è più valida. Uno degli obiettivi del MIS è quello di ridurre la grandezza del trauma chirurgico rispetto alla tradizionale tecnica chirurgica. La riduzione della risposta locale e sistemica al danno fisiologico produce dei benefici immunologici e clinici [1]. I vantaggi sono:

- mancanza di incisione addominale;
- minore dolore nella fase post-operatoria;
- minore lunghezza della degenza ospedaliera;
- basso tasso di complicanze;
- tempi di recupero ridotti;
- migliore effetto estetico rispetto alla corrispondente chirurgia aperta.

Recentemente, la ricerca, volta a ridurre ulteriormente il trauma chirurgico, ha portato la chirurgia a voler utilizzare solamente gli orifizi naturali.

Il concetto di operazione senza cicatrici ha sempre affascinato i chirurghi di tutto il mondo: è questo l'obiettivo che la "chirurgia endoscopica transluminare attraverso orifizi naturali" (NOTES) si prefigge di raggiungere[2]. Questo metodo designa una

procedura chirurgica che utilizza uno o più orifizi naturali del corpo con l'intenzione di perforare visceri cavi in modo da entrare in cavità del corpo altrimenti inaccessibili. Gli orifizi naturali di solito includono la bocca, lo stomaco, il colon e la vagina[3]. Teoricamente i vantaggi di una procedura NOTES rispetto all'approccio laparoscopico sono:

- assenza di incisione addominale;
- procedura meno invasiva;
- minimizzazione di anestesia;
- riduzione del dolore della parete addominale nel fase postoperatoria;
- minore probabilità di infezione della ferita;
- minore probabilità di formazione di ernia e aderenze.

A causa dell' originalità della procedura e del processo medico, la connotazione e la classificazione della tecnica NOTES non sono ancora definite e sono a volte controverse. Questo può impedire un efficace progresso scientifico e la diffusione di questa nuova tecnica. Secondo lo stato attuale degli studi le procedure NOTES sono principalmente divise in due categorie: "puro" e "ibrido". Il NOTES "puro" fa riferimento ad una procedura che è stata completata senza l'utilizzo di alcuna porta transaddominale, incluso l'ombelico. Il NOTES "ibrido" si riferisce ad una tecnologia "mista" che utilizza una strumentazione transaddominale per facilitare la procedura NOTES[4]. Fino ad oggi, la maggior parte delle procedure NOTES pubblicate ha utilizzato almeno uno strumento transaddominale per mantenere l'orientamento spaziale, la triangolazione o il ritiro di un campione. Allo stato attuale, le tecniche ibride NOTES sono particolarmente importanti per l'esperienza clinica iniziale, dove la sicurezza deve essere tenuta nella più alta considerazione possibile. L'utilizzo di una porta transaddominale deve essere considerato nello sviluppo della tecnica NOTES invece di essere incompatibile con la NOTES.

La maggior parte dell'esperienza con NOTES viene da esperimenti su animali. Le vie più studiate e popolari sono la via orale transgastrica, transcolonica e transvaginale. Per esempio: la tecnologia NOTES transgatrica può essere brevemente riassunta come segue:

1) Cura pre-operatoria e anestesia: tutti gli animali sperimentali vengono nutriti con formula liquida e poi digiuno 12-24 ore. Gli antibiotici sono di solito somministrati per via endovenosa (o intramuscolare) e lo stomaco è irrigato con acqua sterile e una soluzione antibiotica. L'anestesia totale e la ventilazione meccanica sono necessarie per garantire che la procedura NOTES avvenga senza problemi.
2) Gastrotomia: la parete anteriore dello stomaco è, di solito, il sito di incisione ideale per accedere alla cavità peritoneale, mentre la parete posteriore può essere selezionata per esplorare il reteroperitoneum. Successivamente un endoscopio sterilizzato a doppio canale è inserito nello stomaco attraverso un tubo sterile. Un bisturi endoscopico viene poi utilizzato per creare una incisione di 2-4 mm con elettro-cauterizzazione. Un palloncino di dilatazione è spinto all'interno di un catetere e l'incisione è radialmente dilatata per garantire il libero accesso dell'endoscopio alla cavità peritoneale. Un filo guida posizionato attraverso la stessa incisione gastrica è necessario per l'orientamento dell'incisione in caso di rimozione accidentale dell'endoscopio.
3) Pneumoperitoneum: l'obiettivo è quello di ottenere uno spazio adeguato al funzionamento e alla visualizzazione, e questa pratica può essere ottenuta o da un insufflatore endoscopico on-demand, o da un insufflatore laparoscopico autoregolato. Alcuni ricercatori utilizzano un catetere percutaneo attraverso la parete addominale per aspirare il gas insufflato[5].
4) Chiusura gastrica: con il ritiro dell'endoscopio inflessibile, gli endoclips, o altri dispositivi di sutura, vengono utilizzati per chiudere l'incisione gastrica al fine di eliminare o diminuire le possibili infezioni intra-addominali causate da perdite del contenuto gastrico[6].

A causa degli ostacoli tecnici esistenti e delle preoccupazioni di sicurezza, nella maggior parte del mondo, il NOTES era stato confinato a studi su animali fino a quando non è stato pubblicato il primo rapporto di una colecistectomia NOTES in un essere umano realizzato da Marescaux. A partire dal luglio 2008, un totale di 16

studi, su 49 soggetti umani, sono stati effettuati con procedure NOTES su appendicectomia transvaginale, appendicectomia transgastrica, peritoneoscopia, biopsia epatica, sigmoidectomia e legatura delle tube. Nel 2010 Xiu-li Zhang et al. hanno confrontato gli studi pubblicati legati alla procedura NOTES negli ultimi anni al fine di documentare il grande entusiasmo e interesse per il nuovo campo[3]. Da questo confronto è emerso che il numero di pubblicazioni riguardanti la NOTES tra il 2006 e il 2009 è passato da 6 a 160.

I potenziali benefici della NOTES alla chirurgia addominale rappresentano una nuova frontiera della chirurgia. Tuttavia, diversi ostacoli al momento impediscono l'adozione diffusa per l'applicazione clinica. Nel 2005, il Consorzio per la Valutazione e la Ricerca per la chirurgia attraverso orifizi naturali (NOSCAR) è stato istituito per studiare e valutare la NOTES. Gli ostacoli alla pratica clinica sono stati affrontati nel dettaglio, compreso l'accesso al sito ottimale, la chiusura gastro-intestinale, la prevenzione delle infezioni intraperitoneali, l'orientamento spaziale all'interno della cavità peritoneale e lo sviluppo di nuovi strumenti. Nonostante il notevole lavoro nel campo NOTES, è chiaro che c'è molta strada da percorrere relativamente agli esperimenti su animali all'utilizzo clinico e non poche barriere da superare.

La NOTES è una procedura promettente che potrebbe essere un'altra alternativa favorevole alle attuali tipologie di interventi chirurgici. Molte operazioni comuni eseguite oggi in laparoscopia potrebbero essere convertite in un approccio NOTES e molti pazienti potrebbero trarne beneficio. Attualmente, un elevato numero di ricerche sperimentali sugli animali deve essere effettuato, dispositivi endoscopici più adatti devono essere sviluppati e il coordinamento tra endoscopisti e chirurghi rafforzato, prima che la NOTES sia adottata come una tecnologia praticabile[3].

1.2 **Motivazioni**

Una procedura NOTES viene spesso effettuata con le attuali tecniche endoscopiche e una serie di accessori endoscopici, quali lacci endoscopici, pinze da biopsia endoscopica, pinze da presa endoscopiche, endoloops e clip endoscopiche. Nonostante i limiti di endoscopi attualmente utilizzati nella procedura NOTES, l'endoscopio terapeutico a doppio canale è ancora il tipo di endoscopio principale[3]. Questo introduce due importanti problemi:

- **Problema della movimentazione**: molte manovre importanti per la manipolazione dei tessuti, come l'afferramento, la trazione, la compressione o la divisione delle strutture, sono difficili da eseguire anche con un endoscopio flessibile a due canali. La flessibilità dell'endoscopio, vantaggiosa per attraversare il lume intestinale, è uno svantaggio quando si necessita stabilizzare l'estremità distale per effettuare interventi precisi e sicuri. Risulta quindi evidente il bisogno di fissare ed irrigidire l'endoscopio.
- **Problema della visione**: uno dei principi fondamentali della laparoscopia è la triangolazione dell'ottica e della strumentazione, che prevede un controllo visivo su degli strumenti disposti ai lati del apparecchio di visione. Le attuali versioni di endoscopi precludono una triangolazione nella tecnica NOTES. I canali di lavoro sono in linea con il sistema di visione, limitando l'ampiezza di movimento degli strumenti ed impedendo di ottenere una visione della profondità della scena (effetto di parallasse).

Gli strumenti utilizzati in laparoscopia consentono di avere la rigidità necessaria per effettuare interventi chirurgici, ma non possono essere adoperate per applicazioni NOTES in quanto la rigidezza non permette di compiere traiettorie curvilinee che un endoscopio flessibile riesce a percorrere.

A fronte delle problematiche esposte, si evince la necessità di sviluppare una piattaforma appositamente studiata per applicazioni NOTES. Questa necessità, però, non è ancora stata soddisfatta da nessuna delle grandi aziende di apparecchiature biomedicali, le quali non riescono a proporre altro che endoscopi modificati. Nel paragrafo 2.1 verranno illustrati alcuni di questi dispositivi.

Il motivo per cui è molto difficile sviluppare tale strumentazione può dipendere dalla scarsa disponibilità di componenti e tecnologie. Non è facile reperire sul mercato componenti che soddisfino dei requisiti specifici. Il limite principale di componenti altamente miniaturizzati, come quelli richiesti nell'ambito della chirurgia mini invasiva, riguarda soprattutto le loro basse prestazioni. Se, ad esempio, si prendono in considerazione gli attuatori miniaturizzati, essi spesso non riescono a sviluppare delle coppie o delle forze sufficienti. Le telecamere miniaturizzate disponibili in commercio hanno delle risoluzioni basse o delle ottiche non adatte . Ulteriori limiti vengono introdotti dai circuiti di pilotaggio di tali componenti che sono anch'essi difficili da miniaturizzare.

Se le grandi aziende non sembrano cogliere la forte necessità di produrre nuove piattaforme, un'altra direzione invece è stata presa dai laboratori di ricerca. Sono numerosi gli studi che vengono effettuati proprio in questo ambito.

In questo scenario si inserisce il progetto Snake, all'interno del quale si colloca questa tesi, che si occupa di sviluppare una piattaforma robotizzata per NOTES. Il progetto, finanziato dal Ministero dell'Istruzione, dell'Università e della Ricerca, è già in corso da un paio di anni e viene seguito dal prof. Pietro Cerveri del Dipartimento di Bioingegneria del Politecnico di Milano.

Al progetto collabora anche l'AIMS (Advanced Mini-Invasive Surgery) Academy dell'Ospedale Niguarda Cà Grande di Milano insieme al quale sono stati definiti i requisiti clinici e tecnologici e che fornisce strumenti e informazioni utili per la progettazione.

Da un anno circa si è aggiunta anche la ZD Mechatronics srl, un'azienda che si occupa di prototipazione e che sta aiutando a sviluppare un primo prototipo del robot.

1.3 **Requisiti tecnologici**

La piattaforma deve essere sviluppata in modo da permettere al chirurgo di operare nelle condizioni più confortevoli e sicure per il paziente. Questa necessità si traduce nei seguenti requisiti tecnici:

- Fornire al chirurgo uno strumento di facile manovrabilità e destrezza, in modo da permettergli di lavorare in condizioni idealmente paragonabili a quelle che possiede in chirurgia tradizionale. Questo generalmente si ottiene introducendo dei gradi di libertà interni allo spazio di lavoro;
- Mantenere le dimensioni degli strumenti che verranno inseriti nel corpo più ridotte possibili (diametro al di sotto di 10-12 mm);
- Permettere di cambiare lo strumentario chirurgico velocemente ed automaticamente senza dover sfilare l'intero dispositivo inserito nel paziente;
- Utilizzare un'architettura modulare per aumentare i gradi di libertà del sistema senza perdere prestazioni in termini di accuratezza e rigidezza;
- Prevedere un numero ridondante di componenti hardware, ad esempio sensori, e software in modo da fornire un numero ridondante di informazioni e avere un sistema più affidabile;
- Gli strumenti che interagiscono con il paziente devono essere biocompatibili e sterilizzabili;
- L'interfaccia utente ed il sistema di controllo devono essere facili da usare. Il sistema di controllo deve permettere al chirurgo di operare agevolmente.

1.4 **Invenzione**

1.4.1 **Descrizione**

L'invenzione consiste in un apparato robotico miniaturizzato di tipo modulare per il posizionamento di strumenti chirurgici con superiore destrezza, flessibilità/rigidezza e accuratezza rispetto ai manipolatori convenzionali basati su endoscopi flessibili. L'apparato è costituito da unità identiche, internamente attuate,

interconnesse tra loro in cascata in modo tale da realizzare una catena cinematica di tipo seriale. Ciascuna unità, di forma cilindrica (∅ 1cm, lunghezza 4-5cm), esprime due gradi di libertà indipendenti tramite un giunto flessionale ad una estremità e un giunto assiale all'altra estremità. La capacità di movimento dell'apparato è dipendente dal numero di unità interconnesse. Con tre unità si ottiene un posizionamento a 6 gradi di libertà tipico dei sistemi robotici convenzionali. L'attuazione è implementata direttamente tramite micro-motori elettrici a bordo dell'unità di tipo brushless, che sono più precisi e sviluppano meno calore dei normali motori DC. Ciascun giunto è dotato di un encoder di tipo rotativo per la misura angolare della sua posizione. Ogni unità è equipaggiata con un sensore di forza miniaturizzato mono-assiale impiegato per registrare le interazioni meccaniche agenti sull'unità. Complessivamente, l'apparato può misurare le forze che si trasferiscono dalla parte distale verso la parte prossimale (interazioni ambientali). Ciascuna unità è dotata di un modulo elettronico basato su microprocessore che esplicita le seguenti funzionalità: 1) comunicazione IO (input/output) con le unità di attuazione adiacenti e con un sistema esterno master tramite modalità bus CAN; 2) attuazione e controllo dei motori; 3) monitoraggio dei sensori (encoder rotativi, sensore di forza). La connessione tra unità adiacenti è implementata tramite un'interfaccia innovativa meccanico/elettrica di tipo *plug-and-play* che consente contemporaneamente la cinematica relativa tra le unità e lo scambio dei segnali elettrici (alimentazione, segnali Input/Output di controllo e di riferimento per gli attuatori, segnali output dei sensori). Nell'insieme, l'apparato può funzionare attraverso un controllo in tempo reale per il posizionamento del punto terminale sia in anello aperto che in anello chiuso. Si ipotizza che un sistema di supervisione esterno all'apparato sia in grado di pianificare e generare i comandi di alto livello per ogni giunto. Questi vengono poi distribuiti a ciascuna unità attraverso la linea di comunicazione CAN che autonomamente gestisce la generazione dei comandi di attuazione a basso livello per i driver (stadi di potenza) dei motori. La modalità in anello chiuso è abilitata tramite la rilevazione della posizione angolare dei giunti equipaggiati con encoder rotativi e l'implementazione a bordo di un meccanismo di servo assistenza per i motori.

Il potenziale scenario di utilizzo dell'invenzione proposta è il seguente: la parte prossimale dell'apparato è connessa a un macro-posizionatore che porta l'apparato robotico in prossimità del paziente; sulla parte distale dell'apparato è agganciato un modulo operativo per la manipolazione chirurgica o l'ispezione che può essere dotato di propria attuazione e sensorizzazione (consideriamo moduli terminali di tipo lavoro quali pinze, ablatori, forbici, sistemi di sutura, e di tipo sensore quali visione, temperatura, suoni, ultrasuoni, palpazione meccanica); operativamente, l'apparato robotico viene introdotto all'interno della cavità addominale attraverso una singola incisione sulla parete esterna dell'addome o attraverso una incisione sulle pareti interne di organi cavi (vagina, ano, stomaco).

1.4.2 **Applicazione industriale**

L'invenzione proposta ambisce a definire una struttura di manipolazione chirurgica dotata di destrezza sufficiente per applicazioni di chirurgia mini-invasiva dove non è possibile utilizzare strumenti rigidi e le attuali tecnologie basate su endoscopi flessibili presentano i limiti enunciati nel paragrafo 1.2. Ad esempio, per la NOTES con accesso trans-vaginale o trans-gastrico, il chirurgo si trova a dover protendere l'endoscopio per distanze maggiori di quelle richieste dalla laparoscopia e allo stesso tempo deve eseguire movimenti più complessi per seguire l'andamento dei dotti anatomici e superare punti con aderenze o rigonfiamenti.

Altre limitazioni, dovute all'uso di uno strumento come l'endoscopio, possono essere evidenziate nella perdita di orientamento spaziale, aspetto ricorrente in tutta la chirurgia mini-invasiva, e nella ridotta coordinazione occhio-mano che limita la triangolazione. Quest'ultimo limite è caratteristico della chirurgia a singola porta nella quale la visione e lo strumentario sono coassiali poiché costretti a passare attraverso un unico strumento endoscopico. Inoltre, l'endoscopio flessibile non garantisce un giusto compromesso tra flessibilità e rigidezza. L'endoscopio necessita infatti di entrambi: da un lato deve essere snodabile per giungere in prossimità del sito da operare, ma dall'altro deve essere in grado di esercitare la forza richiesta dalla procedura garantendo la necessaria rigidezza. L'invenzione proposta integra la

possibilità di definire configurazioni multiple e controllabili di posizionamento, di esercitare le forze necessarie e di presentare la rigidezza sufficiente tipiche della manipolazione dei tessuti in chirurgia addominale e toracica. L'invenzione si candida, quindi, ad essere integrata in un sistema chirurgico robotizzato per interventi addominali e toracici.

1.4.3 **Limitazioni**

Uno degli aspetti importanti della presente invenzione riguarda la miniaturizzazione. Il requisito di dimensionare a 1cm il diametro massimo della singola unità di attuazione vincola tutti gli stadi dello sviluppo a partire dalla fase di disegno meccatronico e di selezione dei componenti e dei materiali, fino alla fase di realizzazione meccanica e integrazione dei componenti.
A livello di attuatori e sensori non sussistono particolari limitazioni poiché sono già stati individuati fornitori per i micro-motori brushless del diametro inferiore a 4mm, lunghezza di 20mm con coppie all'albero sufficienti, per encoder rotativi a effetto Hall di dimensione di 4mmx4mm con risoluzione a 12bit, per sensori di forza monoassiale cilindrici di diametro inferiore a 2mm. Per quanto riguarda l'elettronica, la scala metrica di microcontrollori commerciali adatti allo scopo raggiunge i pochi millimetri e procedure di stampa di circuiti a multi-strato garantiscono il sufficiente livello di integrazione in dimensioni contenute inferiori a 1cm. Inoltre i processi di fabbricazione a scala millimetrica con accuratezze micrometriche e l'impiego di nuovi materiali con elevate proprietà meccaniche (rigidezza e leggerezza) fa prevedere che la fabbricazione, l'assemblaggio e l'integrazione siano procedure a basso rischio. Per applicazione nel campo dei dispositivi biomedici, l'elevata biocompatibilità di particolari materiali metallici (titanio), polimerici o fibre di carbonio, con buon grado di lavorabilità, limita i problemi derivati dal contatto con fluidi biologici.

1.4.4 **Principi attuativi**

L’apparato proposto si compone di unità cinematiche identiche di forma cilindrica, connesse in cascata, aventi ciascuna 2 gradi di libertà di movimento. Connettendo tre unità di ottiene un manipolatore a 6 DoF (Figura 1.1).

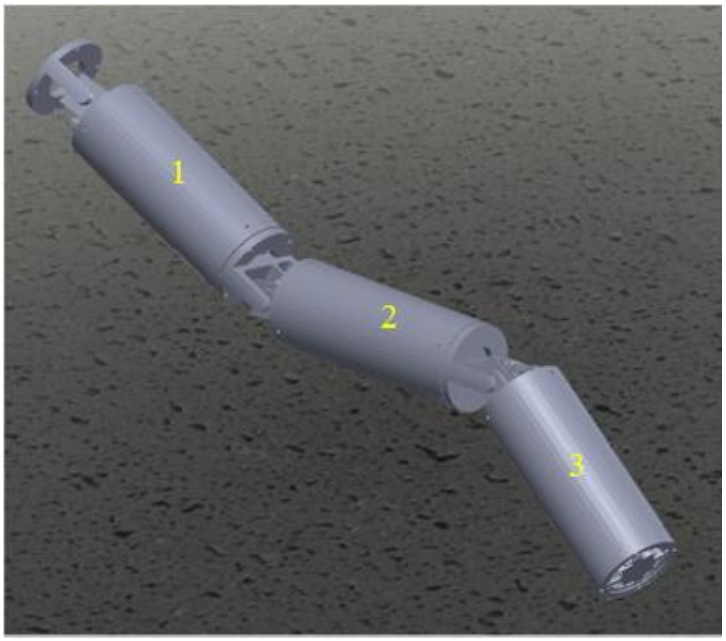


Figura 1.1 Connessione di 3 unità cinematiche

All’estremità distale dell’apparato può essere connesso un generico dispositivo o strumento di lavoro chirurgico (es.: camera, pinza, forbice,..). Operativamente si possono ottenere manipolatori ridondanti connettendo 4 o più unità di cinematiche. Un’unità cinematica è costituita da un corpo di ∅ 1cm e include alle sue estremità un giunto per il movimento assiale e un giunto per il movimento di flessione, entrambi attuati tramite un motore rotativo e appositi meccanismi (Figura 1.2).

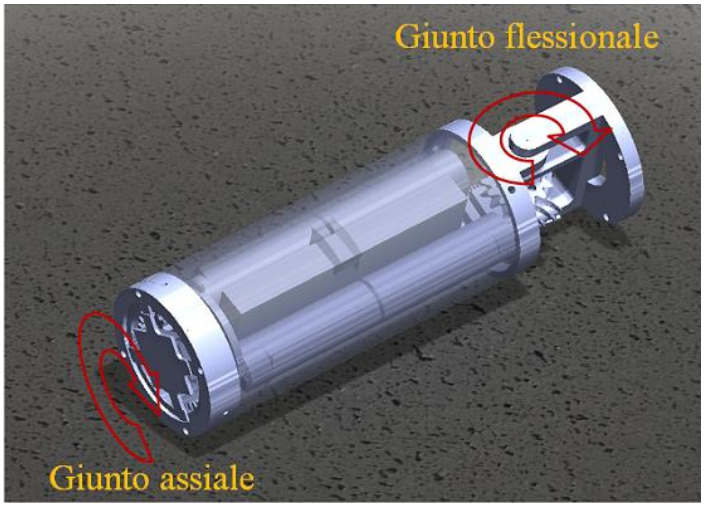


Figura 1.2 Singola unità. Giunto flessionale e assiale

Il giunto assiale è implementato tramite un rotismo epicicloidale mentre il giunto flessionale è implementato tramite un rotismo accoppiato di tipo vite/ruota (Figura 1.3). Il rotismo di tipo vite/ruota ha il vantaggio di avere una sola direzione di rotazione (positiva nel senso di rotazione del motore) mentre la direzione opposta è impedita al movimento.

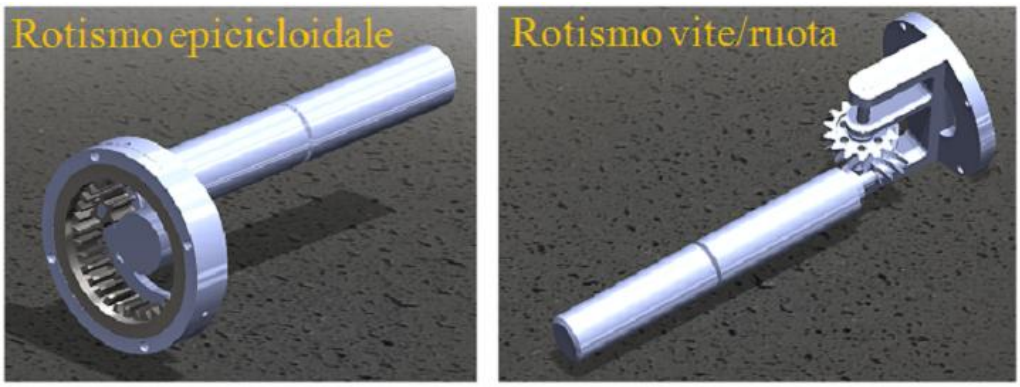


Figura 1.3 Rotismo epicicloidale e rotismo vite/ruota

Questa soluzione fornisce un meccanismo interno di coppia resistente importante per concorrere a controbilanciare, ad esempio, l'effetto della gravità. Per evitare che gli sforzi agenti sull'albero del giunto e sull'accoppiamento vite/ruota in caso di contro-rotazione producano danni meccanici, il giunto è opportunamente concepito in

termini di resistenza meccanica, precisione dell'accoppiamento includendo anche micro-cuscinetti rotativi. Date le basse velocità di movimento richieste (180°/s), è possibile utilizzare alti rapporti di riduzione tra la velocità dell'albero del motore e la velocità dell'albero del giunto.

A bordo di ciascuna unità cinematica, sono disposti i seguenti dispositivi (nel Capitolo 3 verranno trattati singolarmente):

- Due microcontrollori
- Due driver
- Sei transistori MOSFET
- Due motori
- Due encoder
- Un tranceiver
- Un sensore di forza

Si ipotizza che lo scafo dell'unità cinematica possa essere realizzato in materiale metallico (leghe di titanio o alluminio) e realizzato attraverso microlavorazioni con macchine a controllo numerico. Per le componenti meccaniche di rotismo si considera di utilizzare acciaio il cui auto-accoppiamento cinematico presenta un basso coefficiente di attrito (0.1). Per quanto concerne l'albero del giunto di flessione, si considera di utilizzare l'acciaio 40NiCrMo7, particolarmente resistente e quindi adeguato per le dimensioni estremamente ridotte del pezzo. Includendo il sistema di controllo e le connessioni elettriche, complessivamente si stima il peso per la singola unità nell'ordine di 15g.

In termini dinamici, l'apparato (es. 3 unità connesse) deve essere in grado di sostenere il proprio peso (~0.45N), esercitare adeguate forze di trazione durante la manipolazione dei tessuti biologici (0.25-5N), produrre delicate e precise forze di spinta durante task di inserimento di aghi per biopsie (~5mN). In aggiunta, durante qualsiasi task l'apparato deve garantire una sufficiente rigidezza. Esistono mini-

motori brushless commerciali della NAMIKI (JP) in grado di produrre coppie all'albero nell'ordine di 0.66mNm a 200rpm e 0.19mNm a 900rpm (∅: 2mm, L: 15mm) o 5mNm a 85rpm e 1.5mNm a 350rpm (∅: 4mm, L: 20mm). Per opportuni rapporti di trasmissione in unione al meccanismo di blocco del movimento retrogrado del giunto flessionale è possibile rispondere all'esigenze meccaniche sopra citate. Compatibilmente con le dimensioni richieste, i rotismi dei due giunti sono disegnati per fornire tali rapporti di riduzione e consentire di generare coppie risultanti di 100mNm (giunto flessione) e 10mNm (giunto assiale), quindi in grado di esercitare forze all'estremità dell'apparato con 3 moduli connessi (L=10cm) pari a 1N.

L'interfaccia meccanica tra le unità cinematiche è concepita per garantire un'elevata accuratezza dell'accoppiamento e il trasferimento trasparente della cablatura elettrica tra unità adiacenti. L'interfaccia connette fisicamente il giunto assiale di una unità con il giunto flessionale della successiva unità. A causa del movimento relativo presente non è possibile trasferire la cablatura elettrica attraverso cavi convenzionali rigidi. Per superare questo limite si concepisce un sistema bus flessibile di connessione. Il bus si configura come un cavo piatto arrotolato a spirale, di spessore ridotto e aderente alla superficie dell'interfaccia, e sviluppato attorno al giunto flessionale. Nel moto relativo del giunto il bus segue coerentemente il movimento deformandosi in modo elastico. I terminali del bus sono connessi al corpo distale della unità precedente e al corpo prossimale dell'unità successiva. Per l'involucro del bus, che incapsula le linee elettriche metalliche, consideriamo materiali polimerici o siliconici di tipo gommoso (elastomeri) già comunemente usati per apparecchiature medicali.

1.5 **Scopo della tesi**

Lo scopo della tesi è quello di fornire una descrizione del sistema robotico. In particolare focalizzare l'attenzione sul sistema di comunicazione dell'intera piattaforma, e del singolo modulo. Per far ciò è necessario prima effettuare un attento

studio dei protocolli di comunicazione attualmente in uso. Di questi scegliere quelli che si adattano meglio alle singole funzioni. È inoltre necessario definire un protocollo di comunicazione interno, ovvero adattato alla nostra piattaforma. La tesi deve essere in grado anche di fornire una descrizione dei componenti scelti. Di ogni componente bisognerà studiarne le caratteristiche funzionali e se queste sono adattabili oppure no ai nostri scopi. I singoli componenti non dovranno soltanto adattarsi ai requisiti funzionali ma anche ai requisiti meccanici ed elettrici (diametro, dimensione, potenza, alimentazione, ecc..). Tutti questi componenti dovranno poi essere montati su una board che verrà successivamente inserita all'interno di ogni unità. La tesi dovrà anche descrivere come è stata realizzata questa scheda.

La gestione del movimento, il controllo dei sensori, la gestione della comunicazione spetta a due microcontrollori assemblati sulle board. È opportuno quindi descrivere come vengono svolte queste funzioni e il firmware utilizzato.

A tale scopo la tesi è suddivisa nei seguenti capitoli:

- Capitolo 2: Fornisce un quadro generale delle tecnologie utilizzate fino ad oggi nel settore della chirurgia endoscopica.
- Capitolo 3: Descrive le caratteristiche generali dei componenti utilizzabili per ogni singolo modulo. Di ognuno verrà fornita la sigla del componente scelto e le motivazione che hanno portato alla sua scelta.
- Capitolo 4: Definisce il Controller Area Network (CAN). Protocollo di comunicazione utilizzato per lo scambio di dati dell'intera piattaforma. Verrà poi illustrato come questo è stato adattato ai nostri scopi.
- Capitolo 5: Descrive il protocollo di comunicazione Serial Peripheral Interface(SPI) e come è stato implementato per la comunicazione tra micro e encoder.
- Capitolo 6: Illustra il protocollo di comunicazione ideato.
- Capitolo 7: Mostra gli schematici elettronici realizzati.
- Capitolo8: Fornisce una descrizione della modalità di controllo dei motori e dei sensori.
- Capitolo 9: Descrive i firmware realizzati appositamente per i due microcontrollori.

- Capitolo 10 : Descrive i test effettuati.

Capitolo 2
STATO DELL'ARTE

In questo capitolo verranno descritti i principali dispositivi biomedici che costituiscono lo stato dell'arte tecnologico. In seguito sarà discusso lo stato dell'arte brevettuale, ovvero le invenzioni brevettate che più somigliano al nostro robot snake. Infine viene presentato un elenco dei principali miglioramenti e vantaggi della nostra invenzione rispetto allo stato dell'arte.

2.1 **Stato dell'arte tecnologico**

A partire dalla metà degli anni '80 i sistemi robotici, che prima venivano utilizzati solo in ambito industriale nelle catene di montaggio, trovano il loro impiego anche nel settore medico e chirurgico. Risale infatti al 1985 il primo utilizzo di un robot antropomorfo in medicina. Si tratta del PUMA 560. Questo venne utilizzato per posizionare un ago durante una biopsia al cervello con l'assistenza di una CT (Computer Tomografy). In seguito sempre più società hanno compreso i notevoli vantaggi che un sistema robotico può apportare alla medicina. Nel 1988, il PROBOT, sviluppato dall'Imperial College di Londra, è stato utilizzato per effettuare un intervento chirurgico alla prostata. Il ROBODOC della Integrated Surgical Systems è stato impiegato nel 1992 per interventi di chirurgia ortopedica. Nel 1997 un

intervento di inversione di legatura delle tube è stato eseguito con successo a Cleveland utilizzando Zeus della Intuitive Inc. La stessa azienda negli anni successivi lancia il Da Vinci che nel 2000 riceve l'autorizzazione dalla FDA (Food and Drug Administration) per essere utilizzato per interventi di chirurgia urologica, interventi di laparoscopia generale, laparoscopia ginecologica, interventi di chirurgia toracica non cardiaca.

I sistemi robotici per applicazioni NOTES (Natural Orifice Transluminal Endoscopic Surgery) sono ancora in fase sperimentale. Le piattaforme attualmente disponibili si limitano a sfruttare le caratteristiche di un endoscopio flessibile, modificando solamente l'end effector per renderlo più funzionale agli scopi chirurgici. Inoltre esistono in commercio dei cateteri robotici studiati per procedure di elettrofisiologia. Le tecnologie incluse in questi strumenti sono di grande interesse, dato che potrebbero essere applicate anche in una piattaforma per chirurgia transluminare. Numerose ricerche sono rivolte alla progettazione di un innovativo apparecchio robotico in grado di soddisfare la richiesta dei clinici di poter utilizzare uno strumento apposito per NOTES. Inoltre parallelamente a queste ricerche, ne vengono condotte molte altre in diversi ambiti scientifici che potrebbero fornire le tecnologie necessarie allo sviluppo di una piattaforma robotica adatta al nostro scopo.

Un endoscopio può essere adoperato sia per controlli diagnostici sia per l'esecuzione di interventi terapeutici (endoscopia operativa), sia ancora come strumento di controllo durante un'operazione chirurgica. Essi sono costituiti da una telecamera per l'ispezione visiva e da dei piccoli strumenti per prelevare eventualmente del tessuto per biopsie. Con la nascita della tecnica NOTES gli endoscopi sono stati utilizzati per interventi chirurgici che normalmente vengono effettuati mediante chirurgia aperta o laparoscopica. Per adempiere a tutti i requisiti imposti dagli interventi NOTES, i costruttori di endoscopi hanno apportato modifiche agli endoscopi normali in modo da renderli più adatti alla chirurgia. In particolare si sono soffermati sulla modifica dell'end effector, più che alla manovrabilità e al controllo dell'intera piattaforma. Riporto di seguito esempi di endoscopi modificati per applicazioni NOTES.

L'Anubis della Karl Storz (Figura 2.1) è un endoscopio il cui end effector è provvisto di tre canali per gli strumenti chirurgici ed una telecamera. Durante l'inserimento dell'endoscopio, gli strumenti sono chiusi all'interno di un cappuccio conico. Una volta che l'endoscopio è arrivato nel sito dell'intervento, esso si apre e gli strumenti si allungano per uscire dai loro canali [7]. In questo modo si ottiene solo un leggero effetto di triangolazione degli strumenti con il sistema visivo. I limiti principali di questo strumento sono una bassa illuminazione e campo di vista, un'interfaccia meccanica non ergonomica e la completa assenza di robotizzazione.

Figura 2.1 Anubis della Karl Storz

La USGI Medical (Figura 2.2) ha sviluppato i sistemi TransPort e Cobra che rappresentano delle evoluzioni del precedente modello ShapeLock, il quale è costituito da un tubo in cui può essere inserito un endoscopio e gli strumenti chirurgici. Il tubo può essere introdotto in uno stato flessibile e poi essere bloccato in una configurazione rigida. In questo modo costituisce una piattaforma stabile per la chirurgia. Il TransPort ha un diametro di 16 mm, un movimento indipendente della punta (180° di retroflessione e movimento laterale) e quattro canali per gli strumenti e per l'endoscopio. Siccome il controllo è manuale, i movimenti degli strumenti non sono molto precisi. Gli strumenti sono in linea con l'endoscopio e quindi non si ha un effetto di triangolazione [8].

Figura 2.2 TransPort della USGI medical

Per risolvere questo problema è stato sviluppato il Cobra (vedi Figura 2.3), che ha tre canali indipendenti [9]. Il controllo è effettuato mediante l'uso di cavi e la movimentazione risulta rigida e imprecisa. Un altro svantaggio di questi sistemi è che gli strumenti chirurgici non possono essere cambiati se non estraendo l'intero tubo.

Figura 2.3 Cobra della USGI Medical

L'R-scope della Olympus (Figura 2.4) ha due canali per altrettanti strumenti che si muovono uno in direzione verticale e l'altro orizzontale. E' previsto di un sistema visivo e di un canale per il fluido di pulizia dell'end effector. E' una piattaforma abbastanza stabile sia per la ritrazione che per la dissezione. Lo svantaggio è il non facile orientamento visivo spaziale, la sua dimensione e la sua eccessiva flessibilità [10].

Figura 2.4 R-scope della Olympus

L'Endosamurai, mostrato in Figura 2.5, è un prototipo della Olympus costituito da due canali più grandi di lavoro e uno più piccolo per la ritrazione. E' provvisto di un sistema visivo e di uno per l'illuminazione. E' il primo strumento con cui si ottiene una buona triangolazione tra il canale visivo e la strumentazione.

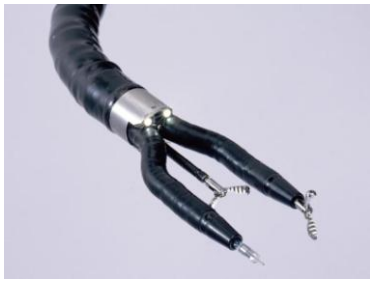


Figura 2.5 Endosamurai delle Olympus

Un altro esempio molto chiaro di endoscopio con bracci robotici è il Direct Drive Endoscopic System (Figura 2.6) formato da una guida avente rigidità flessibile in cui vengono inseriti un endoscopio e due strumenti da 4mm. Gli strumenti vengono comandati dall'esterno mediante due manopole. L'intero sistema è fissato solidalmente al lettino operatorio in modo da permettere una maggiore stabilità dell'intero sistema. L'end effector è dotato di 5 gradi di libertà. Altri due gradi di libertà sono dati dalla guida.

Figura 2.6 Direct Drive Endoscopic System

Basandosi su un'attuazione mediante leghe a memoria di forma, la Columbia University ha proposto l' IREP (Insertable Robotic Effector Platform) illustrato in Figura 2.7 per applicazioni a singola porta SPA (Single Port Access) [11]. Una volta che il robot è entrato nella cavità addominale, l'end effector si apre e fuoriescono due braccia robotiche ed una telecamera stereoscopica mobile con tre gradi di libertà. Le braccia robotiche sono divise in due segmenti, costituiti da leghe a memoria di forma, che aggiungono quattro gradi di libertà al sistema. In cima alle braccia ci sono gli strumenti chirurgici che possono ruotare su se stessi.

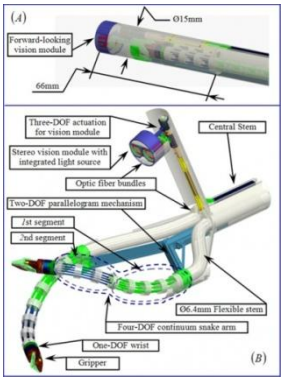


Figura 2.7 IREP (Insertable Robotic Effector Platform)

Un esempio di robot flessibile usato in ambito clinico è il Sensei della Hansen Medical System (Figura 2.8). Questo robot raggiunge il sito di interesse mediante un catetere. Il catetere viene introdotto attraverso un vaso sanguigno e spinto fino a raggiungere l'organo di interesse, solitamente il cuore, per procedure di

elettrofisiologia. Raggiunto il sito di interesse, dalla punta del catetere, esce il robot avente sei gradi di libertà. Questa parte del catetere è formata da una calza metallica percorsa in lunghezza da quattro fili paralleli. L'attuazione di questi fili mediante dei motori permette di muovere il robot. Esso è controllato esternamente dal chirurgo mediante un joystick tridimensionale. L'interazione con i tessuti viene continuamente monitorata, attraverso dei sensori di forza, e mostrata su un display. Anche da un punto di vista della visualizzazione lo strumento è molto innovativo, in quanto fornisce tre diverse visualizzazioni scelte dal chirurgo.

Figura 2.8 Sensei della Hansen Medical System

Di tutt'altra specie sono i robot MAGS (magnetic anchoring and guidance system). Essi sfruttano i principi del magnetismo. L'idea alla base è quella di introdurre attraverso un orifizio naturale, nel caso NOTES, oppure attraverso un'incisione, nel caso LESS (Laparo-Endoscopic Single Side), tutti gli strumenti necessari per effettuare un intervento chirurgico, ancorarli mediante magneti alla parete addominale e muoverli e manovrarli dall'esterno.

Un prototipo interessante è l'AB1 sviluppato dall'Università del Nebraska (Figura 2.9). Il robot è formato da una barra centrale, provvista di due telecamere per la stereovisione ed un led per l'illuminazione, collegata ai suoi estremi a due braccia

robotiche che recano in punta gli strumenti chirurgici. Il braccio destro è provvisto di un cauterio mentre quello sinistro di una pinza. Lo strumento viene inserito interamente nella cavità addominale attraverso un orifizio naturale e poi fissato tramite due magneti, uno interno ed uno esterno, alla parete addominale. Dall'esterno il chirurgo si aggancia con una console e controlla il movimento del robot. La console è formata da uno schermo con due joystick ai lati, uno per ogni manipolatore. Sono state effettuate colecistectomie su maiali che hanno avuto un esito positivo. Lo strumento si è dimostrato capace di manovrare i tessuti e dissezionarli [12].

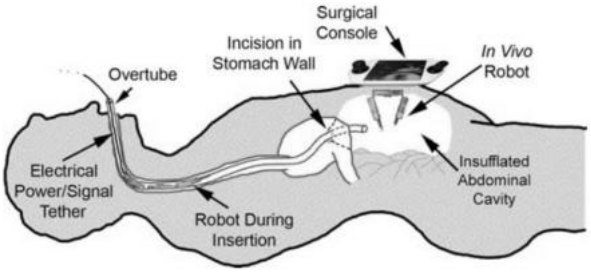


Figura 2.9 L'AB1 sviluppato dall'Università del Nebraska

I robot finora descritti sono tutti del tipo "Fixed-Base". Esistono altri robot che invece vengono inseriti completamente nel paziente e da lì si muovono indipendentemente. Sono stati sviluppati e testati robot mobili in grado di navigare all'interno della cavità addominale (Figura 2.10). Essi possono essere previsti di una piattaforma esterna per la visione e l'assistenza alla procedura chirurgica. Il robot è una sorta di telecamera su ruote che viene inserito attraverso un'incisione. Le sue ruote, che si muovono indipendentemente l'una dall'altra, hanno dei profili a vite. In questo modo riesce a muoversi senza lacerare il tessuto. La telecamera è capace di effettuare uno zoom. Recentemente è stato provvisto anche di una piccola pinza per biopsia. In studi su animali questa telecamera mobile è stata utilizzata per provvedere alla visualizzazione durante un'esplorazione addominale ed una colecistectomia. In un altro studio la piattaforma robotica è stato in grado di effettuare con successo una biopsia di tessuti epatico [13].

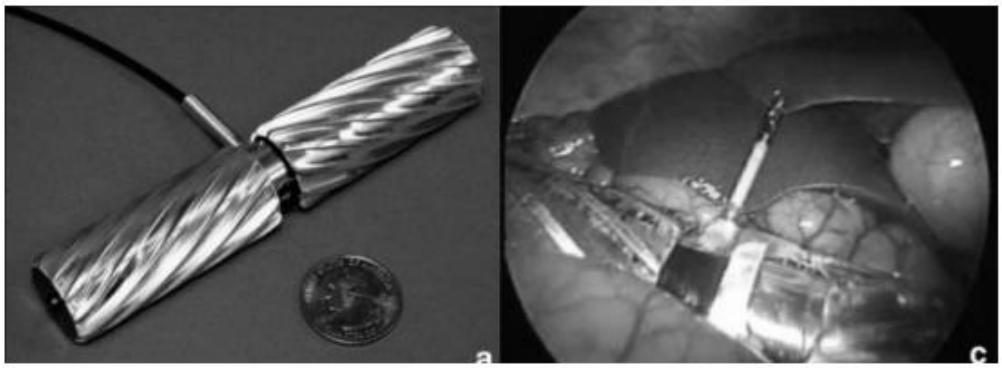


Figura 2.10 Robot mobile

Numerosi campi di ricerca sono interessati nello sviluppare strumenti robotici che mimano i movimenti di determinate specie animali. I robot antropomorfi sono ad esempio ispirati all'essere umano, ma esistono numerose specie da cui ci si potrebbe ispirare per costruire sistemi robotici flessibili in grado ad esempio di soccorrere e svolgere tutti quei compiti in cui è richiesto un livello di destrezza che l'uomo non possiede. La letteratura è piena di esempi di “Snake like robots” e lo sviluppo di questi robot che riescono a mimare il movimento dei serpenti è ad uno stato molto avanzato. Il serpente è un rettile in grado di compiere dei movimenti molto agili sia quando è appoggiato per terra, che quando nuota o si arrampica sugli alberi. Il suo movimento è caratterizzato dalla sua ampia disponibilità di gradi di libertà. Inoltre è la testa che decide la traiettoria, infatti ogni punto della traiettoria percorso dalla testa verrà percorso dal resto del corpo. Questo permette al serpente di attraversare percorsi molto articolati senza che il corpo urti gli ostacoli che la testa ha precedentemente evitato. Quindi esso non sta semplicemente raggiungendo un oggetto, ma sta compiendo un percorso per raggiungerlo. Riuscire ad imitare questa capacità in ambito medico permetterebbe, ad esempio, al chirurgo di percorrere la traiettoria desiderata con l'end effector senza che il resto del corpo urti le pareti dei tessuti. I sistemi robotici aventi un numero elevato di gradi di libertà, la cui morfologia è simile a quella di serpenti, sono chiamati iperridondanti [14].
La cinematica dei robot iperridondanti è stata modellizzata attraverso differenti tecniche: la convenzione di Denavit-Hartenberg, backbone curve.

La convenzione di Denavit-Hartenberg è un metodo ben noto per stabilire la posizione e l'orientamento dei link di un manipolatore rispetto alla base. Le distanze tra un link ed un altro e gli angoli dei giunti costituiscono dei parametri di una matrice di trasformazione che esprime la posizione di ogni link in funzione della posizione della base. E' un metodo che solitamente viene impiegato per robot aventi un numero limitato di link e gradi di libertà[15].
La backbone curve, invece di descrivere la posizione e l'orientamento di tutti i segmenti del robot, descrive la sua configurazione come una curva che corrisponde alla sua spina dorsale. E' un modello indipendente dall'implementazione meccanica del robot. In questo senso, se il robot si trova in una posizione in cui il corpo è piegato in prossimità di 10 giunti diversi, il fatto che questi giunti abbiano per esempio 2 gradi di libertà, non impone di risolvere un sistema con 20 gradi di libertà ma solo con 10 gradi di libertà. Con questo metodo si riesce a risparmiare sul calcolo computazionale [16,17].
L'ultimo tipo di robot descritto (snake) è quello che più si avvicina a quello sviluppato in questo lavoro di tesi. Tuttavia la definizione e lo sviluppo di un controllo vero e proprio della cinematica dei vari giunti del robot non verranno trattati in questo lavoro di tesi in quanto non ancora stato sviluppato.

2.2 **Stato dell'arte brevettuale**

Di seguito riporto le invenzioni brevettate che più si avvicinano alla nostra. Questa ricerca è stata effettuata in seguito alla scelta di brevettare il lavoro. Nel foglio di brevetto, è necessario fornire una lista di tutti i brevetti, più attinenti all'invenzione, con l'obiettivo di evidenziare le differenze della nostra invenzione rispetto allo stato dell'arte brevettuale, onde esaltarne la novità e l'originalità. Riporto i brevetti trovati effettuando la ricerca sul sito "Espacenet.com". Per ogni invenzione fornisco una breve descrizione, le somiglianze e differenze con il nostro robot.

2.2.1 **Modular robot manipulator apparatus**

Inventore: ZHU HAI HONG [SG] XIE MING [SG]

Applicant: ZHU HAI HONG [SG] XIE MING [SG]

Codice del brevetto:

WO0151259(A2)-2001-07-19

WO0151259(A3)-2001-11-08

WO0151259 (B1)-2002-05-30

Data di pubblicazione: 2000-01-11

L'invenzione (Figura 2.11) rappresenta un' apparato per trasportare vari tipi di strumenti allo scopo di saldatura, pittura, manipolazione, assemblaggio, movimentazione materiali, controllo e qualsiasi movimento desiderato. L'apparecchio comprende una struttura di base la quale presenta un'interfaccia meccanica ed elettrica standardizzata. La base è in grado di fornire la potenza necessaria per effettuare i movimenti. Una serie di moduli collegati insieme uno di seguito all'altro consente all'apparato di eseguire vari movimenti.

Ogni modulo ha due gradi di libertà (DOF): uno per la rotazione di torsione e l'altro per la rotazione girevole. Ogni unità di trasmissione del moto comprende due alberi e uno speciale meccanismo di ingranaggi conici.

Somiglianze:

- sistema modulare,
- ogni modulo presenta 2 gradi di libertà.

Differenze:

- non applicato in medicina,
- utilizza un unico motore,
- non vengono utilizzati sensori (encoder, forza),
- non è presente un sistema di controllo a bordo di ogni modulo.

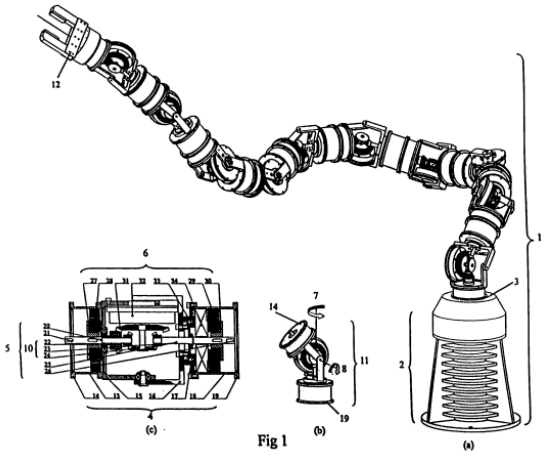


Figura 2.11 Modular robot manipulator apparatus

2.2.2 **Robotic snake**

Inventore: RENNEX BRIAN G [US]
Applicant: RENNEX, BRIAN G [US]
Codice del brevetto: US19930072853 19930607
Data di pubblicazione: 1995-02-07

Questa invenzione mostrata in Figura 2.12 presenta dei miglioramenti rispetto al precedente in quanto costituita da un arto flessibile robotizzato che può funzionare come un serpente robotico. Questi miglioramenti sono particolarmente pertinenti alle applicazioni di miniaturizzazione quali cateteri o posizionatori per microchirurgia, micro-assemblaggio, micro-manipolazione, o micro-esplorazione. Questa invenzione unisce la struttura di base con sistemi di sensing, di controllo e di trasmissione del segnale. Questi sistemi comprendono sensori di pressione, sensori di lunghezza, skin protettivo, imaging ad ultrasuoni, utensili da taglio, e schemi di multiplexing.

Somiglianze:

- applicabile in chirurgia,
- flessibilità dello strumento.

Differenze:

- non sono presenti motori,

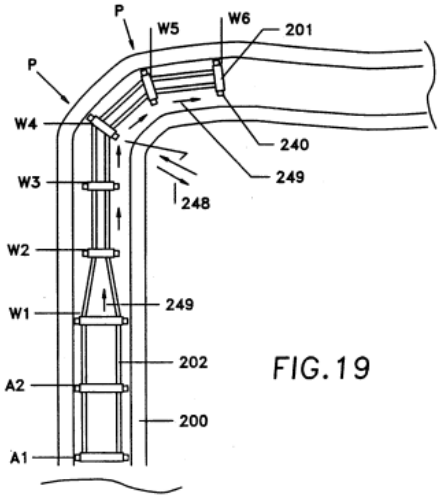


Figura 2.12 Robot snake

2.2.3 **Modular manipulator for robotic surgery**

Inventori: COOPER THOMAS [US]; BLUMENKRANZ STEPHEN J [US]; GUTHART GARY S [US]; ROSA DAVID J [US]
Applicant: INTUITIVE SURGICAL OPERATIONS INC
Codice del brevetto: WO2006US02628 20060124
Data di pubblicazione: 2011-06-08

La suddetta invenzione (Figura 2.13), altro non è che il robot DaVinci accennato nel paragrafo precedente, è generalmente associata ai dispositivi medici, chirurgici.
L‘ invenzione rappresenta un sistema di chirurgia mini-invasiva. Il compito è quello di migliorare le strutture per il sostegno e l'allineamento dei manipolatori robotici, come manipolatori per lo spostamento di uno strumento chirurgico, un endoscopio o altro dispositivo di acquisizione immagini. Migliorare il supporto del manipolatore modulare può fornire diversi vantaggi: maggiore manovrabilità; migliore utilizzazione dello spazio in una sala operatoria; un più semplice e facile set-up;

riduzione delle collisioni tra i vari dispositivi robotici durante l'operazione; ridotta complessità meccanica; ridotte dimensioni di questi nuovi sistemi chirurgici. Tali vantaggi, a sua volta migliorano l'efficienza e la facilità d'uso di tali sistemi di chirurgia robotica.

Il sistema di chirurgia robotica comprende una base di montaggio, una pluralità di strumenti chirurgici, e un gruppo di supporto articolato. Ogni strumento è inseribile in un desiderato sito chirurgico interno attraverso un'apertura minimamente invasiva. La base di montaggio comprende una struttura di sostegno dal soffitto in modo da consentire al gruppo di bracci articolati di estendersi in generale dalla base verso il basso.

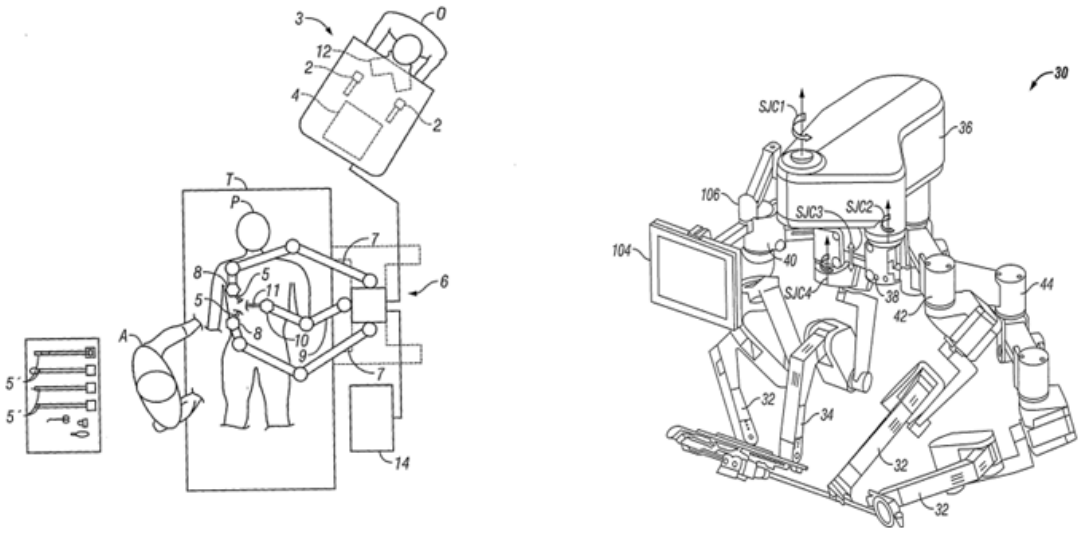


Figura 2.13 Robot DaVinci

Somiglianze:

- modularità,
- applicazione di chirurgia mini-invasiva.

Differenze:

- possibilità di inserire diversi strumenti,
- non utilizzabile per endoscopia ma soltanto per laparoscopia.

2.2.4 **Endoluminal robotic system**

Inventori: DARIO PAOLO [IT] CUSCHIERI ALFRED [IT]
Applicant: SCUOLA SUPERIORE DI STUDI UNI [IT] DARIO PAOLO [IT]
Codice del brevetto: WO2010046823
Data di pubblicazione: 2010-04-29

Il sistema (Figura 2.14) è costituito da una micro-piattaforma robotica che comprende due robot chirurgici ciascuno dotato di uno strumento chirurgico e configurato in modo tale da essere attaccato alla parete interna del tessuto. Entrambi i robot chirurgici sono costituiti da due unità, il che lo rende simile ad un serpente. Il primo robot è formato da una prima unità centrale e una seconda parte articolata che si estende dall' unità centrale alla parete della cavità. Il secondo comprende una seconda unità centrale e un braccio articolato che ha come end-effector lo strumento chirurgico. Il sistema è in grado di muoversi mediante dei motori brushless calettati all'interno ed è dotato di sensori encoder esterni per la retroazione.

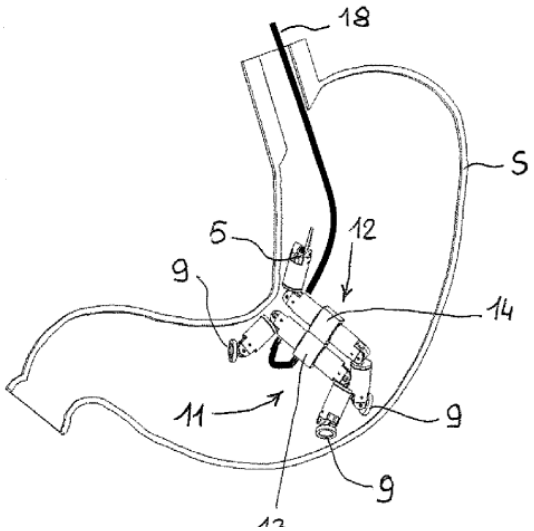


Figura 2.14 Endoluminal robotic system

Somiglianze:

- sistema modulare,
- utilizza motori brushless,
- utilizza encoder.

Differenze:

- il sistema incide sulla parete del lume per potersi muovere all'interno di esso,
- sistema costituito da due soli moduli,
- mancanza di sensori di forza a bordo,
- uso di batteria invece di un'alimentazione fornita dall'esterno.

2.3 **Miglioramenti e vantaggi rispetto alle tecnologie attuali**

In questo paragrafo elenco le principali ottimizzazioni che il robot modulare discusso in questo lavoro di tesi offre rispetto alla tecnologia descritta nei paragrafi precedenti:

- **Aumento del grado di manipolabilità di strumenti chirurgici:** grazie alla connessione di almeno tre moduli (ognuno ha 2 gradi di libertà) è possibile ottenere un sistema a 6 gradi di libertà. Questo garantisce la possibilità di porre l'end effector in una qualsiasi posizione e orientamento all'interno dello spazio operativo. Aumenta anche il numero di possibilità di movimenti e traiettorie per il raggiungimento di tali posizione e orientamento.

- **Miniaturizzazione:** tutti i componenti (motori, microcontrollori, scafo, sensori) sono stati scelti in modo da ridurre il più possibile le dimensioni del modulo per soddisfare le specifiche di progetto.

- **Elevato livello di integrazione:** attuazione e sensori a bordo, elettronica di comando, controllo e diagnostica a bordo, cablatura di interconnessione tra le unità cinematiche adiacenti.

- **Modularità:** moduli identici connessi in cascata per produrre un sistema cinematico di manipolazione.

- **Sensorizzazione:** Ogni modulo è sensorizzato. In ogni modulo sono presenti due encoder ad effetto Hall per rilevare lo spostamento angolare durante la flessione e la rotazione del modulo. È presente anche un sensore di forza in grado di fornire informazioni sull'interazione robot-tessuto.

- **Capacità di esercitare forze sufficienti:** grazie alla presenza di un sensore di forza in grado di rilevare le forze assiali esercitate dal robot sul tessuto è possibile fornire un feedback di forza al chirurgo. Si potrebbe rendere il sistema ancora più all'avanguardia utilizzando un interfaccia aptica con retroazione di forza. Il sensore di forza fornisce informazioni sulla forza che si sta imprimendo sulla struttura e quindi sul tessuto. Questa informazione può essere utilizzata per irrigidire o meno l'interfaccia aptica in modo che il chirurgo abbia una retroazione tattile.

- **Cablatura flessibile:** Il cavo di comunicazione si configura come un cavo piatto arrotolato a spirale di spessore ridotto e aderente alla superficie dell'interfaccia e sviluppato attorno al giunto flessionale. Nel moto relativo del giunto il bus segue coerentemente il movimento deformandosi in modo elastico.

- **Protocollo di comunicazione definito:** Poiché la comunicazione tra esterno e microcontrollore (che controlla il modulo) è basata sul protocollo CAN è possibile sfruttare quest'ultimo per definire un protocollo di controllo

dell’intero sistema. Ad ogni buffer di dati inviati al robot corrisponde un determinato comando per un singolo componente di ogni singolo modulo.

- **Riduzione dei costi di produzione:** Poiché il CAN richiede l’utilizzo di solo due linee di comunicazione, questo riduce la quantità di cavo richiesta, con conseguente riduzione della difficoltà di cablaggio, e tutto questo porta quindi a una riduzione dei costi.

Capitolo 3
MATERIALI E COMPONENTI

Il capitolo che segue vuole fornire una descrizione approfondita dei dispositivi implementati su ogni modulo: microcontrollori, driver e mosfet, motori, encoder, transceiver, sensori di forza. Per ogni componente viene fornita una descrizione di tutti le famiglie di dispositivi presenti sul mercato. In base alla specifiche di progetto verrà individuato un unico componente. Verranno anche illustrate le motivazioni che hanno portato alla scelta.

3.1 **Schema generale**

L'elettronica presente a bordo (schematizzata in Figura 3.1) di ciascun modulo è costituita da:

- Due microcontrollori
- Due driver
- Sei transistori MOSFET
- Due motori
- Due encoder
- Un tranceiver
- Un sensore di forza

Analizziamo in dettaglio ogni singolo componente.

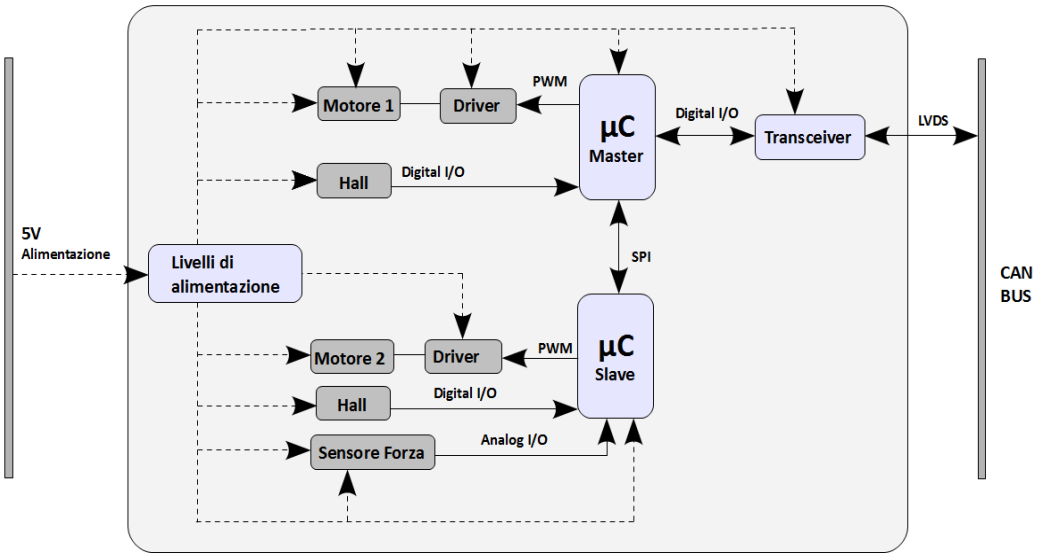


Figura 3.1 Schema generale

3.2 **Microcontrollori**

I microcontrollori sono l'elemento fondamentale del modulo. Essi devono essere in grado di adempiere a tre macro-compiti:

- Controllo del movimento del robot;
- Comunicazione con pc, con altri micro e con sensori;
- Controllo dell' interazione con l' ambiente esterno.

Queste tre funzioni saranno indispensabili nella scelta del micro.

Il controllo del movimento avviene mandando un segnale PWM, pulse width modulation (vedi paragrafo 8.1), ai driver, i quali condizioneranno opportunamente il segnale (vedi paragrafo driver) e lo manderanno ai motori. Esistono due modi per generare segnali PWM: il primo (emulazione PWM software) consiste nello scrivere un codice che generi un'onda quadra, di frequenza e duty cycle assegnata, e mandi questa su un certo piedino di uscita; il secondo (PWM hardware) consiste nell'utilizzare delle periferiche apposite dei microcontrollori, i moduli PWM, per

generare l'onda. Risulta ovvio che il secondo metodo è il migliore (con il primo la generazione dell'onda quadra può essere difficile e comunque toglie preziosi cicli di clock al resto dell'applicazione) tuttavia non tutti i microcontrollori ne sono dotati [18]. È quindi necessario utilizzare un controllore che presenta tali moduli PWM.

Una volta ricevuta l'onda di alimentazione, il motore comincerà a ruotare. Poiché non saranno utilizzati motori con sensori calettati sull'albero-motore, è necessario controllare l'avvenuta rotazione mediante encoder esterni (vedi paragrafo Encoder). Questi inviano il dato dell'angolo in formato digitale mediante comunicazione SPI al micro. Altro modulo richiesto al micro è quindi quello SPI.

La comunicazione tra PC e micro segue il protocollo CAN (controller area network). Per poter effettuare questo tipo di comunicazione è necessario quindi che il micro sia dotato della periferica CAN. La comunicazione invece con il micro-slave avverrà mediante SPI. Saranno quindi necessario almeno due moduli SPI: uno per la comunicazione micro-master con micro-slave e una per la comunicazione micro con encoder.

Le interazione del robot con l'ambiente esterno saranno rilevate dal sensore di forza (vedi paragrafo sensore di forza). Esso genera un segnale analogico. Questa caratteristica impone che il micro sia dotato anche di porte analogiche.

Come detto nell'introduzione alla tesi, il sistema totale deve avere un diametro di massimo 1 cm. Questo richiede che il micro abbia dimensioni quanto più ridotte possibili. Dopo numerose ricerche, il micro che soddisfa tutte le caratteristiche sopra citate, è il dsPIC33FJ128MC802 della Microchip. Il package migliore è il QFN (6x6 mm) a 28 pin.

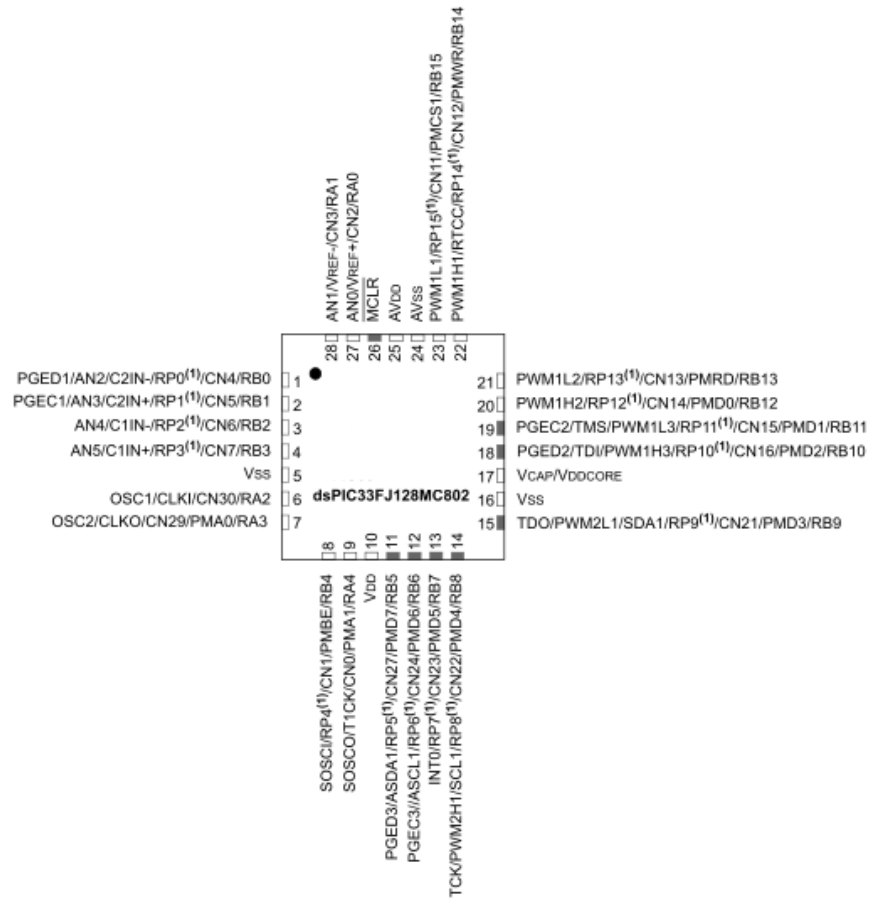


Figura 3.2 Descrizione pin dsPIC33FJ128MC802

Riporto di seguito le caratteristiche principali:

Parameter Name	Value
Architecture	16-bit
CPU Speed (MIPS)	40
Memory Type	Flash
Program Memory (KB)	128
RAM Bytes	16,384
Temperature Range C	-40 to 150
Operating Voltage Range (V)	3 to 3.6
I/O Pins	21
Pin Count	28
System Management Features	PBOR
POR	Yes
WDT	Yes
Internal Oscillator	7.37 MHz, 512 kHz
nanoWatt Features	Fast Wake/Fast Control
Digital Communication Peripherals	2-UART, 2-SPI, 1-I2C
Codec Interface	NO
Analog Peripherals	1-A/D 6x10-bit @ 1100(ksps)
Op Amp	NO
Comparators	2
CAN (#, type)	1 ECAN
Capture/Compare/PWM Peripherals	4/4
PWM Resolution bits	16
Motor Control PWM Channels	8
Quadrature Encoder Interface (QEI)	2
Timers	5 x 16-bit 2 x 32-bit
Parallel Port	PMP
Hardware RTCC	Yes
DMA	8

Figura 3.3 Caratteristiche dsPIC33FJ128MC802

3.3 **Driver e Mosfet**

Il segnale uscente dal micro non è in grado mettere in movimento i motori in quanto non sono in grado di fornire tensione e correnti adatte allo scopo. Sono necessari dei driver e dei transistori mosfet. Questi hanno il compito di innalzare il valore di corrente e tensione. I driver inoltre adempiono anche ad altri scopi: mantengono la Vgs (tensione tra gate e sourse) che serve ad attivare i MOS, ad un livello costante; fanno i modo che la tensione al source non sia flottante. I mosfet sono usati come interruttori. Si richiede che questi componenti presentino dimensioni ridotte.

Per quanto riguarda i driver si è optato per il componente della International Rectifier IRS2001MPBF (mostrato in Figura 3.4), il quale presenta dimensione 4x4 mm. Sebbene siano presenti 16 pin, i pin configurabili sono soltanto 8.

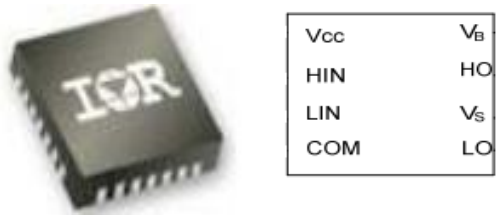


Figura 3.4 International Rectifier IRS2001MPBF

Per i MOSFET invece ho scelto l'integrato della Panasonic FC69430 mostrato in Figura 3.5. L'integrato presenta al proprio interno due transistori NMOS. In questo modo saranno necessari solamente 3 di questi componenti. Le dimensioni sono 1.6x1.6 mm.

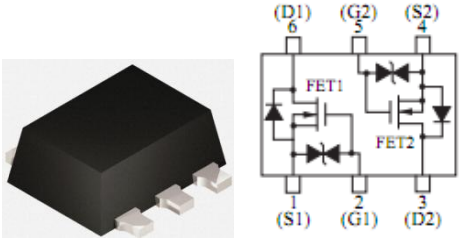


Figura 3.5 Panasonic FC69430

3.4 **Motori**

I motori sono necessari per effettuare i movimenti di bending e rotazione del giunto. Una prima decisione da prendere è stata quello sulla tipologia di motori da utilizzare. Le due grandi famiglie di motori elettrici sono: i motori in corrente continua (CC); i motori in corrente alternata (AC). Il controllo dei motori proviene dai driver collegati a mosfet a loro volta controllati dai microcontrollori. Quindi poiché non è possibile generare una corrente alternata con questa modalità di controllo, e soprattutto vista la pericolosità generata dalla corrente alternata ho deciso

di utilizzare i motori CC. Questa famiglia può essere nuovamente suddivisa in 3 categorie: motori a spazzole (brushed); motori passo-passo; motori senza spazzole (brushless);

I primi hanno una parte che gira detta rotore o armatura, e una parte che genera un campo magnetico fisso detta statore (Figura 3.6). Un interruttore rotante detto commutatore o collettore a spazzole inverte due volte ad ogni giro la direzione della corrente elettrica che percorre i due avvolgimenti generando un campo magnetico che entra ed esce dalle parti arrotondate dell'armatura. Nascono forze di attrazione e repulsione con i magneti permanenti fissi [19].

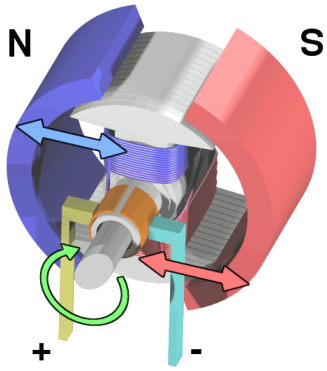


Figura 3.6 Motore con spazzole

Questo gruppo di motori però presenta alcuni svantaggi che hanno portato ad escluderli dalla scelta:

- Le spazzole pongono un limite alla massima velocità di rotazione: maggiore è la velocità e più forte è la pressione che bisogna esercitare su di esse per mantenere un buon contatto;
- Tra spazzole e collettore, nei momenti di commutazione, si hanno transitori di apertura degli avvolgimenti induttivi e quindi scintillio, non accettabili in una sonda endoscopica che si trova a stretto contatto con tessuti umani;
- Si hanno dei problemi di smaltimento del calore (gli avvolgimenti si riscaldano per effetto Joule e il campo magnetico alternato nel nucleo del

rotore genera altre perdite, causate da isteresi magnetica e correnti parassite nel nucleo stesso, e quindi altro calore;

- Gli avvolgimenti appesantiscono il rotore (aumenta il momento d'inerzia): se il motore deve rispondere con rapidità e precisione (come è necessario che avvenga nel nostro snake) il controllo diventa più complesso.

I motori passo-passo sono motori che, a differenza di tutti gli altri, hanno come scopo quello di mantenere fermo l'albero in una posizione di equilibrio: se alimentati si limitano infatti a bloccarsi in una ben precisa posizione angolare (Figura 3.7).
Solo indirettamente è possibile ottenerne la rotazione: occorre inviare al motore una serie di impulsi di corrente, secondo un'opportuna sequenza, in modo tale da far spostare, per scatti successivi, la posizione di equilibrio.
È così possibile far ruotare l'albero nella posizione e alla velocità voluta semplicemente contando gli impulsi ed impostando la loro frequenza, visto che le posizioni di equilibrio dell'albero sono determinate meccanicamente con estrema precisione.

Figura 3.7 Motore passo-passo

La rinuncia all’utilizzo di questi motori invece è dovuta ai seguenti difetti che essi presentano:

- Hanno un funzionamento a scatti e producono vibrazioni, soprattutto ai bassi regimi e se si adottano le tecniche di pilotaggio più semplici.
- Producono molto calore anche dopo pochi minuti.

Il motore brushless (Figura 3.8) è un motore elettrico a magneti permanenti. A differenza di un motore a spazzole, non ha bisogno di contatti elettrici striscianti sull'albero-motore per funzionare. La commutazione della corrente circolante negli avvolgimenti, infatti, non avviene più per via meccanica, tramite i contatti striscianti, ma elettronicamente. L'inversione di corrente è ottenuta tramite un banco di transistor di potenza comandati da un microcontrollore che controlla la commutazione della corrente [20].

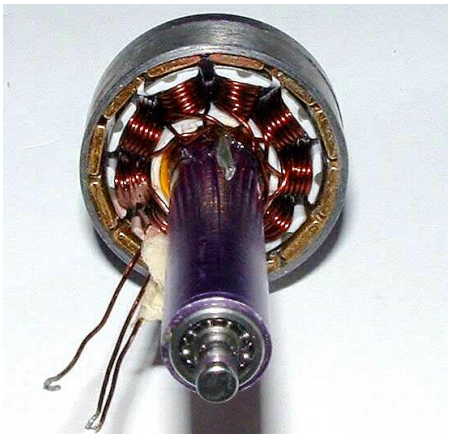


Figura 3.8 Motore senza spazzole

I vantaggi che hanno portato alla scelta di quest'ultima tipologia di motori sono i seguenti:

- minore resistenza meccanica;
- la mancanza di spazzole elimina la possibilità che si formino scintille al crescere della velocità di rotazione;
- non necessita di manutenzione continua;
- vita del motore maggiore rispetto a quella del motore brushed;

- l'assenza di spazzole elimina anche la principale fonte di rumore elettromagnetico presente negli altri motori in continua;
- ingombro limitato;
- sviluppano meno calore;
- i magneti permanenti sono posizionati sul rotore, questo permette di avere un'inerzia molto bassa, cosa che permette di avere un controllo estremamente preciso sia in velocità che in accelerazione.

Dopo una ricerca approfondita sul web, nei vari siti di aziende produttrici di motori brushless, tenendo conto delle specifiche richieste, il motore che meglio le soddisfa è il modello trifase SBL04-0829PG04-337 della Namiki.

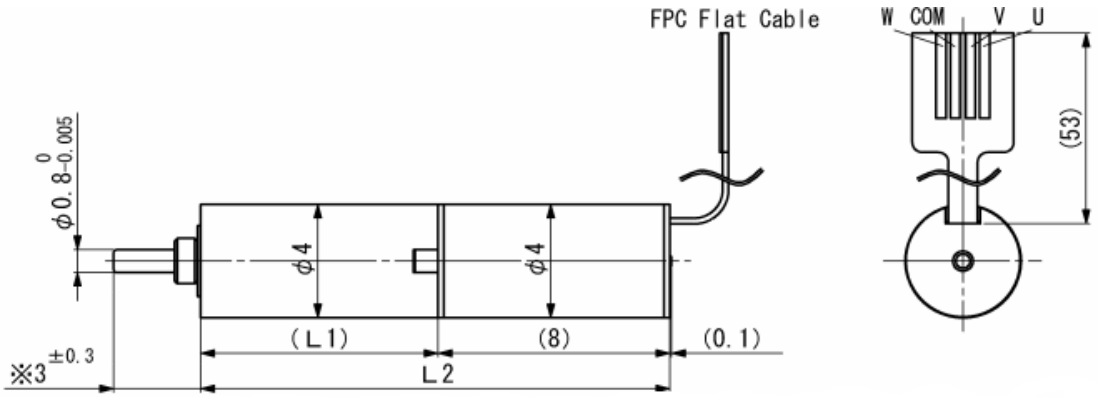


Figura 3.9 Motore Namiki SBL04-0829PG04-337

In Figura 3.10 è rappresentata una tabella con tutte le caratteristiche dei motori della famiglia SBL04-0829 :

	Ordering number	SBL04-0829 PG04-4.3	SBL04-0829 PG04-18	SBL04-0829 PG04-79	SBL04-0829 PG04-337	
1	Nominal voltage	3	3	3	3	V
2	Terminal resistance	29	29	29	29	Ω
3	Gearhead reduction ratio	4.3	18	79	337	:1
4	Gearhead number of stage	1	2	3	4	
5	Max. output power, gearhead output	0.02	0.02	0.01	0.01	W
6	Max. efficiency, motor only	12	11	11	9	%
7	No-load speed, gearhead output	7000	1500	350	85	rpm
8	No-load current	44	45	46	50	mA
9	Stall torque (2-points measurement), gearhead output	0.13	0.43	1.5	5.7	mNm
10	Friction torque	0.10	0.35	1.3	5.6	mNm
11	Back-EMF constant	0.24	1.1	4.7	17.8	mV/rpm
12	Torque constant	2.3	7.7	27.3	111.8	mNm/A
13	Coil inductance	74	74	74	74	µH
14	Mechanical time constant, motor only	5	5	5	5	ms
15	Gearhead length, L1	5.8	7	8.2	9.4	mm
16	Total length, L2	13.8	15	16.2	17.4	mm
17	Rotor inertia, motor only	9×10^{-4}	9×10^{-4}	9×10^{-4}	9×10^{-4}	gcm^2
18	Total weight	810	990	1010	1030	mg
	General data					
19	Operating temperature range	-20 ... +70				°C
20	Max. temperature raise at stall	55				°C
21	Bearing type	Sintered sleeves				
22	Max. permissible press fit force, gearhead output	5				N
23	Max. radial play, gearhead output	0.044				mm
24	Max. axial play, gearhead output	0.15				mm

Figura 3.10 Caratteristiche motori Namiki

Le caratteristiche che hanno portato alla scelta del modello PG04-337 sono:

- Diametro: 4 mm;
- Stall torque: 5.7 mNm;
- Tensione nominale: 3 V;
- Peso: 1.03 g;
- Lunghezza totale: 20.4 mm;
- Rapporto di riduzione 337:1;

3.5 **Encoder**

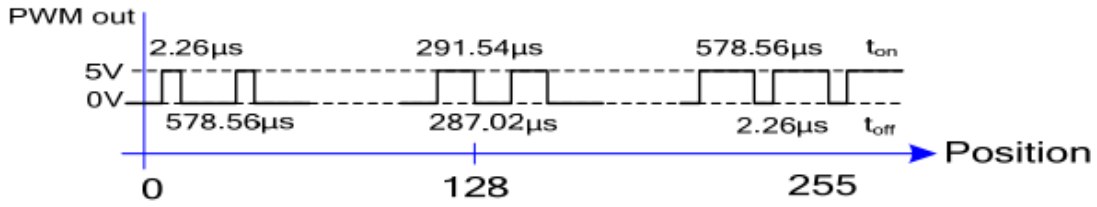
I motori scelti non presentano sensori interni in grado di verificare l'effettivo spostamento dell'albero motore. Poiché è necessario, per la nostra applicazione, avere un feedback è conveniente utilizzare degli encoder rotativi esterni. Gli encoder

rotativi esterni possono essere di due tipi: encoder ottici ed encoder magnetici (ad effetto Hall). Nell'encoder di tipo ottico si ha un raggio di luce che viene rilevato da un sensore ottico e che può essere schermato o meno dal movimento di una superficie rotante su cui sono state praticate delle fessure equidistanti. Il sensore ottico rilevando il raggio luminoso commuta e provvede un voltaggio in uscita, tali variazioni di tensione possono essere processate e quindi contate [21]. Nell'encoder ad effetto Hall invece si misura la variazione di campo magnetico generata da un magnete in movimento. Il magnete dovrà essere di tipo diametrale. L'encoder deve essere posto di fronte al magnete. Tra entrambe le tipologie di sensore della famiglia degli encoder si sceglie la seconda perché, nel secondo caso, a fronte di una risoluzione leggermente inferiore si ha una migliore risposta in frequenza, una banda passante maggiore, dimensioni e pesi inferiori e soprattutto è possibile montare il chip e il magnete su due pezzi distinti, il che aumenta notevolmente la potenzialità di utilizzo di questa tipologia di encoder. Normalmente infatti gli encoder vengono montati con calettamento all'albero motore, ovvero dalla parte opposta alla trasmissione. Avendo dovuto scegliere dei motori con dimensioni minime (e quindi coppia bassissima), applicare un'inerzia direttamente al motore e non dopo la trasmissione potrebbe risultare gravoso a tal punto che la coppia motore potrebbe essere inferiore alla coppia inerziale. Nel caso si voglia utilizzare encoder del primo tipo, collegarli dopo la trasmissione complicherebbe di molto il progetto e sicuramente porterebbe ad un aumento generale delle dimensioni. Per gli encoder magnetici, invece, essendo i due pezzi (chip e magnete) svincolati tra loro, facilitano molto il progetto e permettono di ridurre al massimo le dimensioni.

Inizialmente è stato scelto l'encoder magnetico della Austriamicrosystem AS5030. Questo componente fornisce in uscita un segnale PWM. Questo segnale opportunamente decodificato, dalla formula riportata di seguito, indica la posizione angolare del magnete con una risoluzione di 8 bit:

$$angle[8-bit] = \left(257 \frac{t_{ON}}{t_{ON} + t_{OFF}} \right) - 1$$

dove ton e toff sono rispettivamente il tempo nel quale il segnale PWM è alto e basso.



Questo componente è stato inizialmente scelto perché necessita solo di una linea per la comunicazione tra micro e encoder. Purtroppo la risoluzione fornita non è adatta ai nostri scopi, in quanto troppo bassa, ed inoltre il segnale è soggetto a disturbi. Un disturbo sulla linea, quindi, può portare alla ricezione di un valore errato di PWM.
Si è quindi optato per un altro encoder magnetico, sempre della Austriamicrosystem: l'encoder AS5055. Questo componente oltre ad avere una risoluzione di 12 bit, ha dimensione minori rispetto al precedente. Il formato QFN infatti ha dimensioni 4x4x0.85 mm. Altra caratteristica degna di nota riguarda la comunicazione tra micro ed encoder. Questa infatti è di tipo SPI (Serial Peripheral Interface) vedi Capitolo 5.
L'alimentazione dell'encoder è un altro elemento che ha portato alla scelta di quest'ultimo componente rispetto al primo. L'AS5055 richiede 3.3 V anziché i 5 V dell'AS5030. Questo è molto importante poiché è lo stesso livello di tensione usato per alimentare il microcontrollore. Non necessita quindi di un regolatore di tensione.

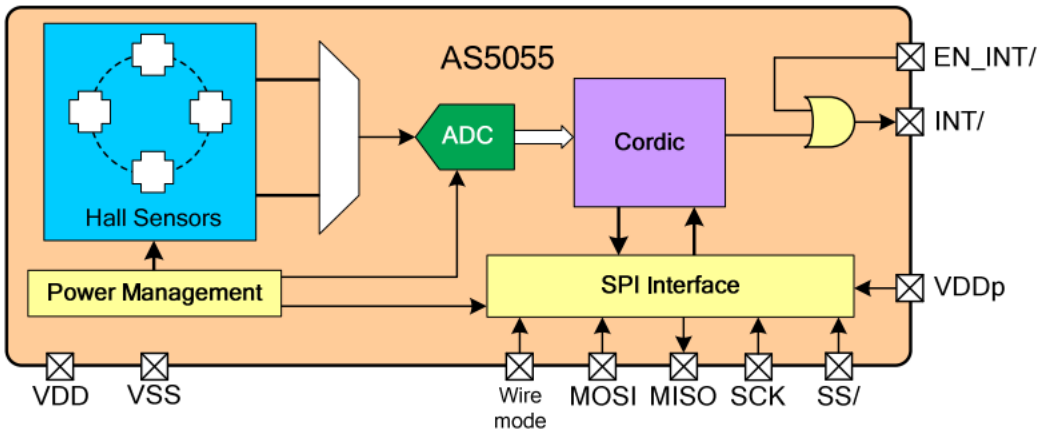


Figura 3.11 Diagramma a blocchi dell'encoder ad effetto Hall AS5055

3.6 **Transceiver**

Nelle reti informatiche, il transceiver o ricetrasmettitore viene tipicamente utilizzato per collegare un bus a un dispositivo che può trasmettere i dati nelle due direzioni (tipicamente un dispositivo che legge e scrive su un registro) come può essere un microcontrollore o una memoria [22]. Nel nostro caso si interporrà tra il modulo CAN del microcontrollore e il PC. Il vantaggio dell'interposizione del transceiver sta nel fatto che i segnali elettrici, pur mantenendo sempre due livelli logici (0 e 1) sono amplificati con una corrente maggiore.
Le caratteristiche richieste per questo componente sono:

- dimensioni ridotte,
- velocità di trasmissione fino a 1 Mbps;
- tensione di alimentazione di 3.3 V.

Il ricetrasmettitore che soddisfa tutte queste caratteristiche è il MAX3051 della Maxim. Le dimensioni sono 2.8x2.6mm.

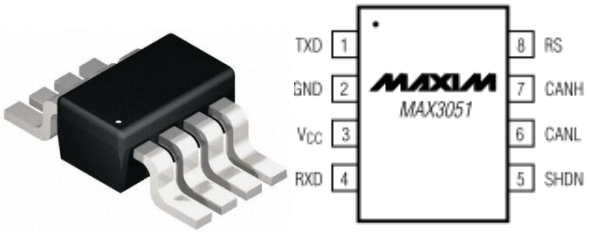


Figura 3.12 Transceiver CAN, MAX3051

3.7 **Sensori di forza**

I sensori di forza sono necessari in quanto è molto importante sapere se stiamo imprimendo una forza eccessiva sulle pareti del lume durante l'inserimento o durante il movimento della sonda. Si cerca quindi di evitare lacerazioni del tessuto. Il sensore di forza è posizionato in modo tale da poter rilevare con estrema precisione le forze assiali agenti sullo scheletro esterno di ogni singola unità. Il segnale dovrà quindi essere decodificato dal micro, e nel caso in cui il valore di forza risultasse maggiore

di un valore limite (ottenuto mediante test), il micro dovrà arrestare immediatamente il movimento e muoversi in direzione opposta.
Il sensore dovrà avere dimensioni molto ridotte in quanto dovrà essere calettato nello scheletro del modulo. Il sensore scelto per tale scopo è l'AIFP della Microstrain (Figura 3.13). Esso presenta dimensioni molto ridotte 5x1.8x1.4. E' costituito da due strain gauges montati su una struttura di forma ellittica. Posizionato opportunamente, non appena la struttura si deforma sottoposta ad una forza esterna, i due strain gauges rileveranno lo spostamento e invieranno il segnale al micro. Nel paragrafo 8.2 verrà spiegato meglio il principio di funzionamento del sensore.

Figura 3.13 Sensore AIFP

Le caratteristiche elettriche e meccaniche del sensore sono rappresentate in Figura 3.14.

ELECTRICAL SPECIFICATIONS

Excitation	*14 milliamps D.C.*
Sensitivity	*30 mV/ma/N*
Non-Linearity	*+/- 1.5% over 1/2 FS* *+/- 2.5% over FS*
Hysteresis	*0.3% FS*
Repeatability	*0.3% FS*
Temp. Coeff. - Offset	*+/- .013% FS/C deg.*
- Span	*+ .0125%/N/C deg.*

MECHANICAL SPECIFICATIONS

Full scale (FS)	*16 N **
Length	*5 mm*
Cross section	*elliptical* *1.4 mm by* *1.8 mm*
Leads	*Polyimide and copper tape cable, 20 cm long*
Transverse stiffness	*Approx. 1.5 N/micron*

Figura 3.14 Caratteristiche elettriche e meccaniche del sensore di forza AIFP

Capitolo 4
CAN-CONTROLLER AREA NETWORK

Il capitolo tratta del protocollo di comunicazione CAN. Verrà inizialmente fornita la definizione e la modalità di trasmissione dei dati sul bus. Successivamente verranno descritti i tipi di dato: data, remote, error, overload. Il protocollo CAN viene utilizzato per mettere in comunicazione il pc con il microcontrollore. Nel paragrafo 4 verrà descritto come ciò avviene. Verranno descritti i programmi utilizzati per comunicare sia da parte del microcontrollore che da parte del pc.

4.1 **Definizione**

Il Controller Area Network, noto anche come CAN-bus, è uno standard seriale per bus di campo (principalmente in ambiente automotive), di tipo multicast, introdotto negli anni ottanta dalla Robert Bosch GmbH, per collegare diverse unità di controllo elettronico (ECU). Il CAN è stato espressamente progettato per funzionare senza problemi anche in ambienti fortemente disturbati dalla presenza di onde elettromagnetiche e può utilizzare come mezzo trasmissivo una linea a differenza di potenziale bilanciata come la RS-485 [23].
La scelta di utilizzo del CAN-bus è dovuta ai notevoli vantaggi tecnologici che offre, tra cui:

- **tempi di risposta rigidi**: la tecnologia CAN prevede molti strumenti hardware e software e sistemi di sviluppo per protocolli ad alto livello (il bus CAN implementa solo i primi due livelli della pila ISO-OSI) che consentono di connettere un elevato numero di dispositivi mantenendo stringenti vincoli temporali.
- **semplicità e flessibilità del cablaggio**: CAN è un bus seriale tipicamente implementato su un doppino intrecciato (schermato o meno a seconda delle esigenze). Questa tipologia di rete di comunicazione viene anche utilizzata per aumentare l'immunità ai disturbi EMC. I nodi non hanno un indirizzo che li identifichi e possono quindi essere aggiunti o rimossi senza dover riorganizzare il sistema o una sua parte.
- **alta immunità ai disturbi**: lo standard CAN raccomanda che i chips di interfaccia possano continuare a comunicare anche in condizioni estreme, come l'interruzione di uno dei due fili o il cortocircuito di uno di essi con massa o con l'alimentazione.
- **elevata affidabilità**: il bus CAN ha un' incredibile capacità di riconoscere gli errori. 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 a 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 l'anno, avrà un errore non rilevato ogni 1000 anni. Praticamente la rete non è soggetta ad errori per tutta la durata della sua vita. Questo è il maggior punto di forza di questo bus. La rilevazione degli errori e la richiesta di ritrasmissione viene gestita direttamente dall'hardware con cinque diversi metodi (due a livello di bit e tre a livello di messaggio).
- **confinamento degli errori**: ciascun nodo è in grado di rilevare il proprio malfunzionamento e di autoescludersi dal bus se questo è permanente. Questo è uno dei meccanismi che consentono alla tecnologia CAN di mantenere la rigidità delle temporizzazioni, impedendo che un solo nodo metta in crisi l'intero sistema.

- **maturità dello standard**: la larga diffusione del protocollo CAN in questi venti anni ha determinato un'ampia disponibilità di chip rice-trasmettitori, di microcontrollori che integrano porte CAN (come ad esempio il dsPIC33FJ128MC802 da me utilizzato), di tools di sviluppo, oltre che una sensibile diminuzione del costo di questi sistemi. Questo è molto importante per far sì che uno standard si affermi nell'ambito industriale.

Sebbene inizialmente applicata in ambito automotive, come bus per autoveicoli, attualmente è usata in molte applicazioni industriali di tipo embedded come nel caso del robot da noi considerato, dove è richiesto un alto livello di immunità ai disturbi. Il bit rate può raggiungere 1 Mbit/s per reti lunghe meno di 40 m. Velocità inferiori consentono di raggiungere distanze maggiori (ad es. 125 kbit/s per 500 m). Il protocollo di comunicazione del CAN è standardizzato come ISO 11898-1 (2003). Questo standard descrive principalmente lo strato (layer) di scambio dati (data link layer), composto dallo strato sottostante (sublayer) "logico" (Logical Link Control, LLC) e dallo strato sottostante del Media Access Control, (MAC) e da alcuni aspetti dello strato "fisico" (physical layer) descritto dal modello ISO/OSI (ISO/OSI Reference Model). Il layer fisico è composto da due linee di tensione su cui viaggia il segnale, CAN_H e CAN_L ai cui estremi sono collegate due resistenze che costituiscono l'impedenza della linea.

4.2 **Trasmissione dati**

Il CAN trasmette dati secondo un modello basato su bit "dominanti" e "recessivi", in cui i bit dominanti sono gli 0 logici ed i bit recessivi sono gli 1 logici. Se un nodo trasmette un bit dominante ed un altro un bit recessivo, allora il bit dominante "vince" fra i due (realizzando una combinazione AND logico).
Stato del bus quando due nodi trasmettono:

	dominante	recessivo
dominante	dominante	dominante
recessivo	dominante	recessivo

AND logico:

	0	1
0	0	0
1	0	1

Con questa tecnica, quando viene trasmesso un bit recessivo, e contemporaneamente un altro dispositivo trasmette un bit dominante, si ha una collisione, e solo il bit dominante è visibile in rete (tutte le altre collisioni sono invisibili). Se i due bit coincidono il nodo continua la trasmissione. Quando il livello associato al bit è recessivo, mentre sul canale si riscontra un livello dominante, l'unità interrompe immediatamente la trasmissione. In pratica avviene che un bit dominante è "asserito" dalla generazione di una tensione fra i conduttori, mentre un bit recessivo è semplicemente ignorato (Figura 4.1). Si è così sicuri che ogni volta che si impone una differenza di potenziale, tutta la rete la rileva, e quindi "sa" che si tratta di un bit dominante. Il Data Link Layer, ha l'importante compito dell'arbitraggio nella competizione per la contesa del canale trasmissivo, da parte dei vari nodi CAN che, contemporaneamente, ne richiedono l'utilizzo.

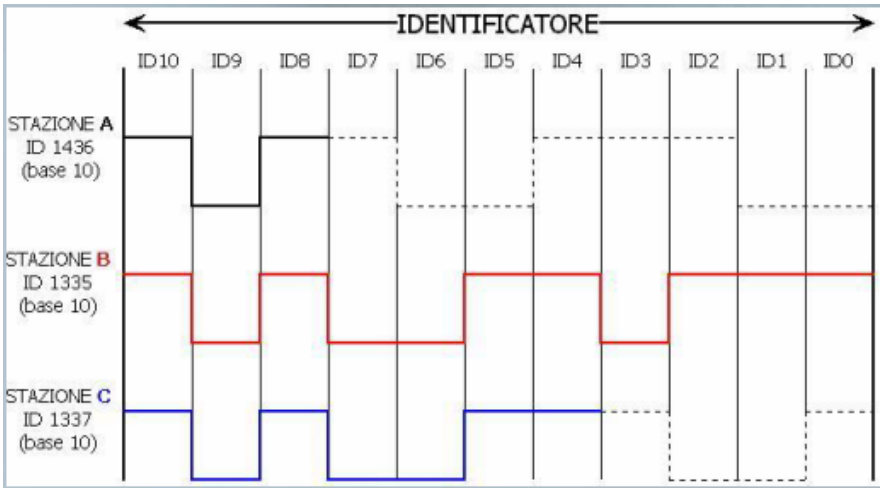


Figura 4.1 Esempio di comunicazione mediante protocollo CAN su bus condiviso.

4.3 **Formato dei messaggi**

Il protocollo CAN supporta quattro tipi di messaggi (anche detti “frame”):

- Data frame: frame contenente i dati che il nodo trasmette.
- Remote frame: frame che richiede la trasmissione di un determinato identificatore.
- Error frame: frame trasmesso da un qualsiasi nodo che ha rilevato un errore.
- Overload frame: frame che introduce un ritardo fra data frame e/o remote frame.

4.3.1 **Data Frame**

Sono i frame che eseguono l'effettiva trasmissione dei dati. I messaggi possono avere due formati:

- Base frame format: con 11 bit di identificazione (Figura 4.2).
- Extended frame format: con 29 bit di identificazione (Figura 4.3).

Lo standard CAN deve obbligatoriamente riconoscere il formato base frame e può opzionalmente riconoscere il formato extended frame format (che, tuttavia, deve essere tollerato).

Il CAN base permette $2^{\wedge 11}$ = 2048 tipi di messaggi diversi. Nella versione extended si possono avere fino a $2^{\wedge 29}$ = 536 870 912 tipi di messaggi.

Formato Base frame

Il formato base frame ha la seguente struttura [24]:

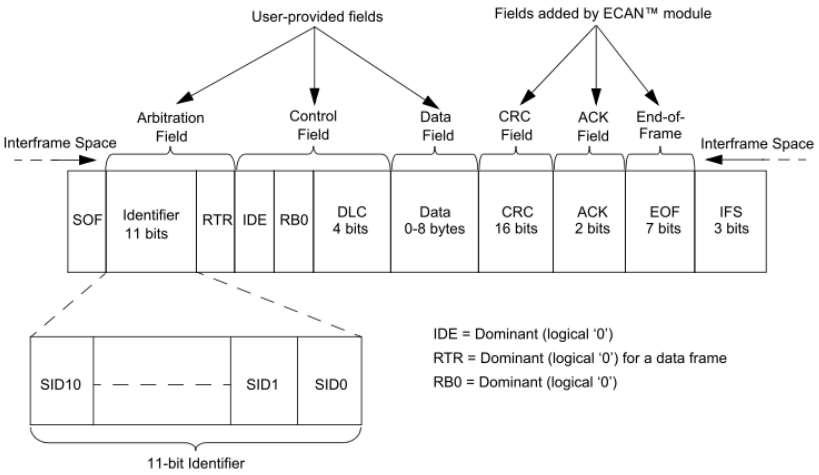


Figura 4.2 Messaggio dati standard

Il messaggio dati standard comincia con un SOF (start of frame) ovvero un bit che identifica l'inizio del pacchetto. Questo è seguito da 12 bit che costituiscono il campo arbitrario. Il campo è formato da 11 bit di identificazione e da un bit RTR (Remote Transmit Request). I primi identificano informazioni contenute nel messaggio ed è usato dal nodo ricevitore per comprendere se il messaggio è "indirizzato" a lui oppure no. Il bit RTR invece serve per distinguere un data frame da un remote frame. Per il messaggio dati di tipo standard questo bit deve essere posto a zero.

Dopo il campo arbitrario troviamo il campo di controllo, il quale fornisce maggiori informazioni riguardo il messaggio. Il primo bit del campo è un identificatore dell'estensione (IDE), il quale indica se il messaggio è di tipo base oppure di tipo esteso. Nel caso considerato, ovvero il messaggio di tipo base, questo bit è fissato dominante quindi posto a zero. Il secondo bit del campo di controllo è il bit RB0, un bit riservato e deve essere dominante. Gli ultimi quattro bit del campo rappresentano

il DLC (Data Length Code), il quale specifica il numero di byte di dati presenti nel messaggio.
Successivamente troviamo il campo dati. Questo contiene i dati veri e propri del messaggio. Questo campo può variare da un minimo di 0 byte a un massimo di 8 byte. Questo numero, come detto precedentemente, viene inserito nel DLC.
Di seguito al campo dati troviamo il CRC (Cyclic Redundacy Check), ovvero il controllo di parità a ridondanza ciclica. Questo serve a determinare se il messaggio è stato alterato da qualche disturbo lungo la linea.
Il bit di acknowledgement (ACK) è inviato come bit recessivo (livello logico '1') ed è sovrascritto come un bit dominante dal nodo ricevitore che ha ricevuto correttamente il messaggio. Il bit viene sovrascritto solo dopo che il messaggio ha superato il filtraggio di accettazione.
Infine troviamo il campo di fine messaggio, il quale è costituito da 7 bit recessivi che indicano la fine del messaggio.
Dopo quest'ultimo campo troviamo lo spazio interframe IFS (ovvero uno spazio tra i vari messaggi). Questo spazio separa due frame successivi trasmessi sul canale. E' costituito da almeno 3 bit recessivi ed ha il compito di dare il tempo necessario ai nodi per processare i precedenti messaggi ricevuti, prima dell'inizio di un nuovo frame.

Formato estended frame

Il formato dell'Extended Frame ha la seguente struttura:

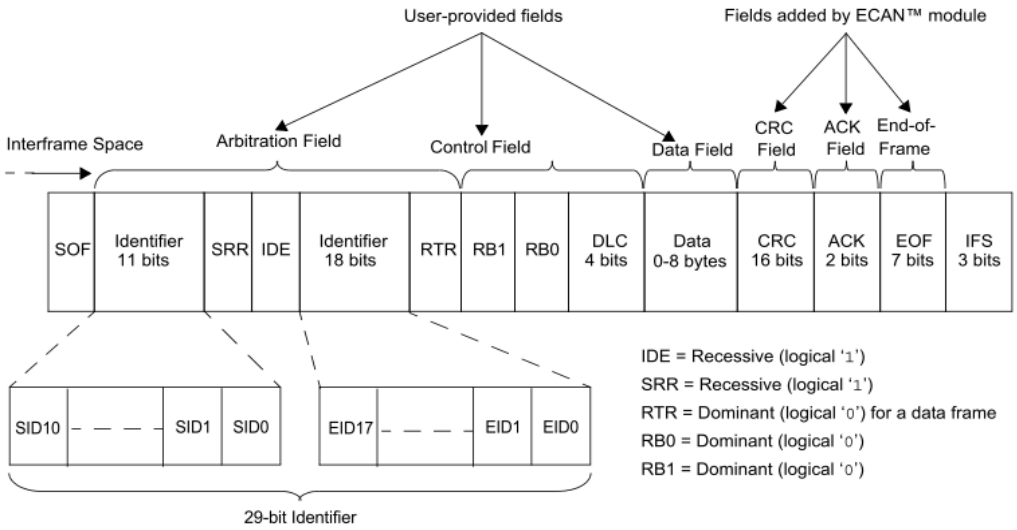


Figura 4.3 Messaggio dati esteso.

Anche il formato esteso comincia con un bit SOF seguito dal campo arbitrario. Questa volta però quest'ultimo è costituito da 29 bit di identificazione suddivisi in due parti separate: una prima parte costituita, come nel caso del messaggio base, da 11 bit; una seconda parte costituita da 18 bit. In questo campo è presente anche un bit SRR (Substitute Remote Request) che determina se il messaggio è di tipo remote, ed anche un bit IDE che indica il tipo di messaggio dati. In questo caso essendo di tipo esteso questo bit deve essere posto a livello logico '1'.

I rimanenti campi sono identici a quelli dello standard data frame discusso precedentemente.

4.3.2 **Remote frame**

Un nodo che si aspetta di ricevere dati da un altro nodo può iniziare la trasmissione verso il nodo "sorgente" mediante l'invio di un remote frame. Il remote frame può essere di tipo standard o di tipo esteso. Esso è simile ai messaggi in formato dati con alcune eccezioni:

- Il bit RTR è recessivo (RTR=1)
- Non è presente un campo dati (quindi DLC=0)

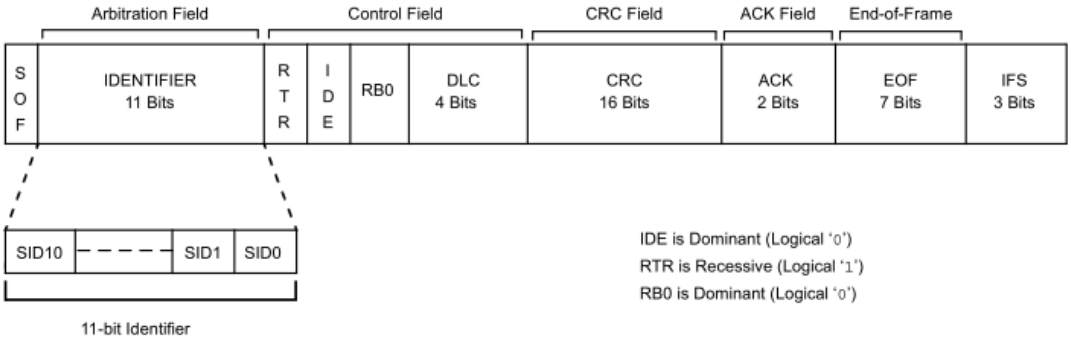


Figura 4.4 Messaggio remoto

4.3.3 **Error Frame**

Questo tipo di messaggio è generato da ogni nodo quando viene rilevato un errore sul bus di comunicazione. Un messaggio errore è costituito da un campo Error Flag seguito un campo delimitatore di errore. Quest'ultimo è composto da 8 bit recessivi e permette ai nodi del bus di poter ricominciare la comunicazione dopo che è stato rilevato l'errore.

4.3.4 **Overload Frame**

Un messaggio di sovraccarico può essere generato da un nodo sia quando un bit dominante è rilevato durante l'interframe space (vedi sopra) sia quando un nodo non è ancora pronto per la ricezione del successivo messaggio (per esempio se sta ancora

leggendo il precedente messaggio ricevuto). L'overload frame ha lo stesso formato del messaggio di errore, ma può essere generato soltanto durante l'interframe space. E' costituito da un campo Flag con 6 bit dominanti seguito da un campo delimitatore con 8 bit recessivi. Un nodo può generare al massimo 2 messaggi di sovraccarico consecutivamente per ritardare l'invio del messaggio successivo.

4.4 **Comunicazione microcontrollore-PC**

Dopo aver definito e spiegato la struttura del protocollo di comunicazione CAN, espongo ora come esso è stato utilizzato per i miei scopi. La comunicazione tra micro e PC è resa possibile tramite il connettore USB-seriale mostrato in Figura 4.5.
Il connettore RS-232 è collegato alla development board sulla quale è montato il micro. Mentre il connettore USB va collegato alla porta USB del computer.

Figura 4.5 Connettore seriale-USB

4.4.1 **Comunicazione lato micro**

La comunicazione da parte del microcontrollore è resa possibile grazie al modulo ECAN presente al suo interno. Il modulo ECAN consta di tre parti: un protocol engine, uno o più filtri per la corretta acquisizione del messaggio, un interfaccia DMA (Direct Memory Access) di trasmissione e ricezione. Il protocol engine trasmette e riceve messaggi sul e dal CAN bus. I filtri, configurabili dall'utente, sono usati dal modulo per esaminare i messaggi ricevuti e quindi determinare se questi messaggi devono essere salvati nel registro messaggi del DMA oppure se devono essere scartati (Figura 4.6).

Per i messaggi ricevuti, l'interfaccia del DMA che si occupa di esso genera un interrupt non appena i messaggi vengono salvati nel buffer. Dopo aver fatto ciò comincia il ciclo del Direct Memory Access. Quest'ultimo legge i dati dal registro CiRXD e li scrive nel buffer presente nella memoria RAM dedicato al DMA.

Per i messaggi trasmessi, l'interfaccia DMA anche in questo caso genera un interrupt e inizia un ciclo di istruzioni, ma in questo caso il DMA legge i dati all'interno del buffer della DMA RAM e li scrive nel registro CiTXD per la trasmissione sul canale. In questo caso ovviamente non è presente alcun filtro in quanto i messaggi provengono dal micro stesso e vanno verso il canale di comunicazione.

Per quanto riguarda i message buffers della DMA possiamo dire che esso supporta fino a 32 registri per l'immagazzinamento dei dati di trasmissione e di ricezione. I buffer numerati da zero a sette posso essere configurati sia per operazioni di trasmissione che di ricezione. I buffer dall'8 al 32 invece possono essere utilizzati invece soltanto per la ricezione e non per la trasmissione [25].

Il DMA agisce come un interfaccia tra i message buffer e il modulo ECAN per la trasmissione e ricezione dei dati senza che sia necessario l'intervento della CPU. Questo permette un migliore utilizzo della CPU, la quale può essere utilizzata per gestire altri dati e istruzioni. Permette inoltre di poter lavorare a frequenze molto elevate in quanto non viene interrotto il processo principale. Il controllore DMA supporta fino a 8 canali (vedi Figura 4.7) per il trasferimento dei dati dalla RAM alle varie interfacce del dsPIC, nel nostro caso al modulo ECAN. Ogni canale è in grado

di manipolare messaggi di 8 o 16 bit di lunghezza. E' possibile variare il livello di priorità di ogni singolo canale in modo tale da poter gestire in modo intelligente vari tipi di messaggio contemporaneamente. Due canali separati sono necessari per supportare la comunicazione dei messaggi CAN: uno per la trasmissione e uno per la ricezione. Ogni canale inoltre presenta un registro (DMAREQ), il quale è usato per assegnare l'interrupt all'evento di ricezione o trasmissione del dato.

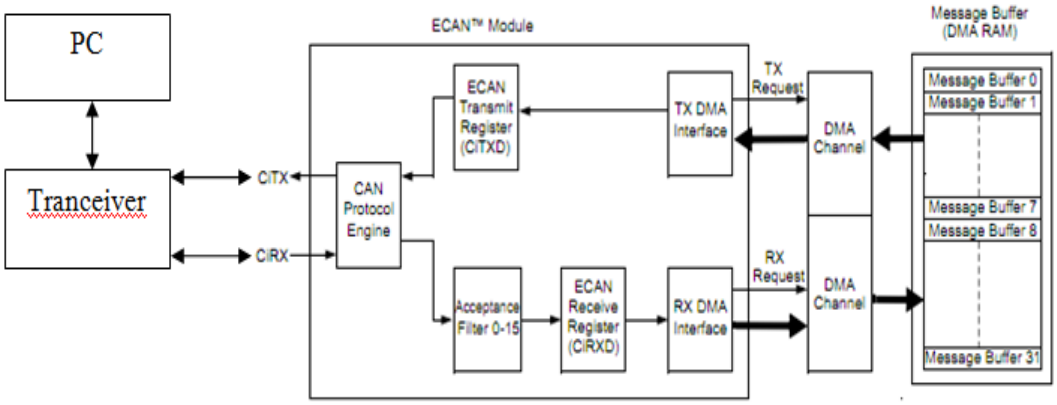


Figura 4.6 Sistema di comunicazione

Trasmissione messaggio

Un nodo che è in grado di generare e inviare un messaggio viene detto trasmettitore. Un nodo rimane un trasmettitore fino a quando il canale non diventa inattivo oppure fino a che una unità non perde la priorità. Un possibile processo di trasmissione del messaggio è illustrato in Figura 4.7.

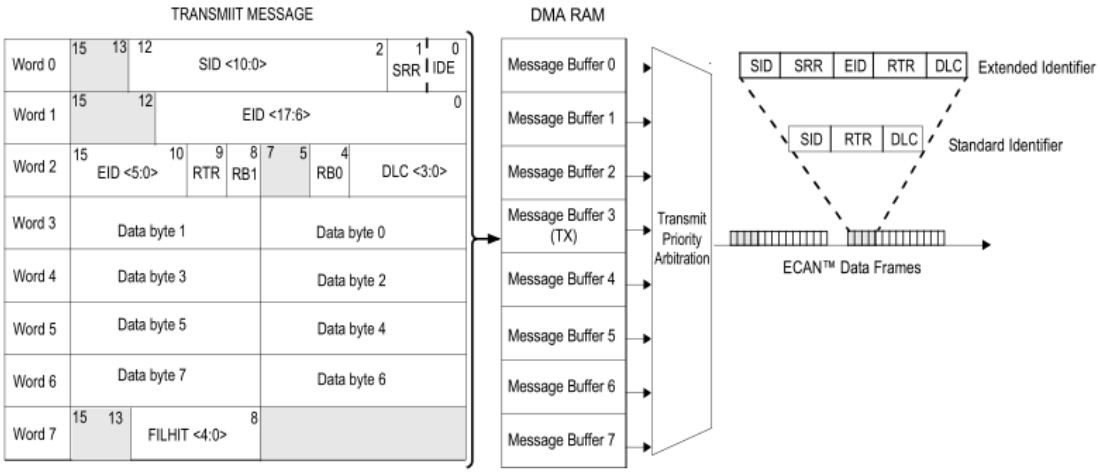


Figura 4.7 Esempio di trasmissione

Come si nota dalla figura il messaggio è suddiviso in 8 campi da 16 bit ciascuno. Le "parole" 3, 4, 5, 6, che rappresentano i dati veri e propri, possono essere di lunghezza variabile. Questi verranno o no riempiti dall'utente a seconda sia che si tratti di messaggio dati oppure che si tratti di messaggio remoto e così via.

Per trasmettere il messaggio sul bus CAN, è necessario compiere i seguenti passaggi:

- Configurare un buffer per la trasmissione ed assegnare un valore di priorità ad esso.
- Scrivere il messaggio CAN nel buffer allocato nella memoria DMA RAM.
- Settare il bit di richiesta di trasmissione (TXREQ) per iniziare la trasmissione del messaggio. Questo bit verrà poi resettato automaticamente dopo che il messaggio è stato inviato.

Ricezione messaggio

Come mostrato in Figura 4.8 ogni messaggio in ingresso proveniente dal bus viene immagazzinato in un buffer di 8 "parole" ciascuna costituita da 8 bit come impostato durante la fase di invio del messaggio (vedi paragrafo precedente). La parte di identificazione (ID) del pacchetto viene confrontata con uno dei 16 filtri di accettazione opportunamente scelto dal'utente.

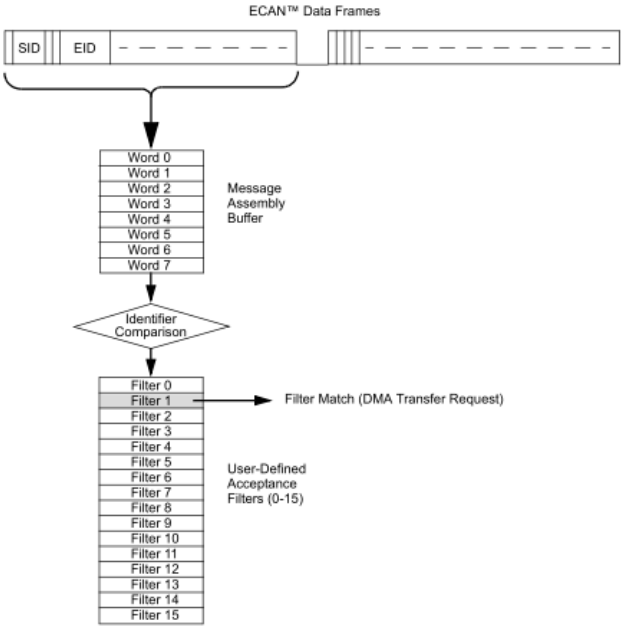


Figura 4.8 Ricezione del messaggio e filtro di accettazione

Sia nel caso di messaggio base che nel caso di messaggio esteso verrà fatto un AND logico tra l'identificativo del messaggio in ingresso e l'identificativo del filtro di accettazione (vedi Figura 4.9 e Figura 4.10). Quest'ultimo può essere impostato settando opportunamente i bit del registro CiRXFnSID (e CiRXFnEID nel caso di exteded frame). Se tutti i bit dell'identificativo del messaggio in ingresso coincidono completamente con quelli del filtro, allora il modulo ECAN genera una richiesta di trasferimento al controllore del DMA in modo tale che il messaggio venga salvato in un appropriato buffer della DMA RAM.

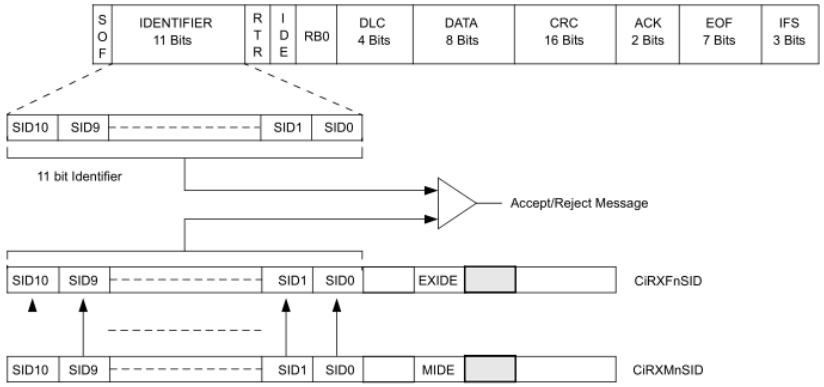


Figura 4.9 Filtraggio di un messaggio base.

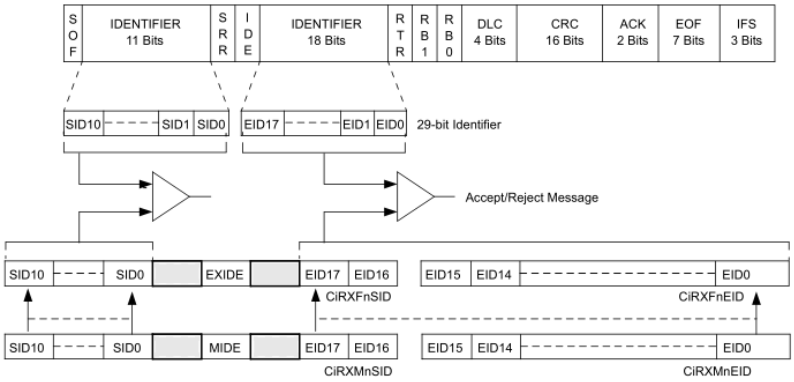


Figura 4.10 Filtraggio di un messaggio esteso.

4.4.2 **Comunicazione lato PC**

Il PC è in grado trasmettere e ricevere dati, da e verso, i vari moduli del robot ed in particolare con i microcontrollori presenti all'interno di essi. L'invio e la ricezione dei dati è resa possibile grazie all'uso del programma Labview. In particolare è stata utilizzata l'interfaccia rappresentata in Figura 4.11.

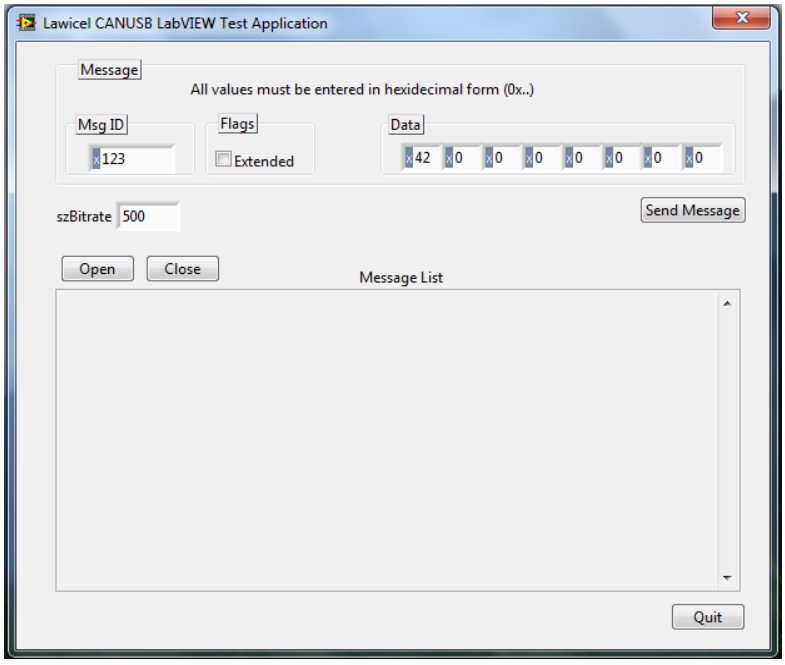


Figura 4.11 Interfaccia Labview per la comunicazione CAN

Mediante questa interfaccia è possibile settare gli elementi principali del protocollo di comunicazione CAN (vedi paragrafo precedente):

- **Msg ID:** Viene impostato il campo identificativo del messaggio. Questi valori andranno a riempire i campi SID (11 bit) e EID (18 bit) a seconda se il messaggio è di tipo base oppure di tipo esteso. L'identificativo deve essere identico all'identificativo dell'elemento al quale vogliamo inviare i dati. Come accennato nel paragrafo precedente, infatti, verrà fatto un AND logico e nel caso in cui l'identificativo non risultasse identico a quello dell'elemento, il messaggio verrà scartato. E' necessario quindi conoscere preventivamente il SID e EID del componente desiderato. Si rimanda al Capitolo 6 la spiegazione del protocollo utilizzato.
- **Flags:** Spuntando la casella viene impostato il messaggio come esteso. Questo corrisponde ad impostare a valore logico '1' il bit IDE.
- **szBitrate:** In questa sezione viene modificata la velocità di trasmissione di invio dei dati. Nel nostro caso sarà di 500 Kbit/s. Anche in questo caso è necessario conoscere in anticipo questo valore e settarlo nel firmware di ogni singolo elemento che si "affaccia" sul bus. Valori di bitrate differenti tra PC e micro non permettono la comunicazione.
- **Data:** Come accennato nel paragrafo 4.3 sia lo standard frame che l'extended frame possono contenere dati da un minimo di 0 byte a un massimo di 8 byte. In questa campo dell'interfaccia Labview è possibile modificare opportunamente i valori dei byte. I dati inviati vengono codificati in modo tale che ad ogni dato inviato corrisponda un comando (anch'esso fissato in precedenza dall'utente).
- **Open-Close:** Una volta settati tutti i parametri necessari per la comunicazione, mediante questi due tasti è possibile "aprire" e "chiudere" il canale di comunicazione.
- **Send Message:** Dopo aver attivato la comunicazione, premendo questo pulsante, viene inviato il messaggio sul canale. A seconda del SID e del EID questo verrà ricevuto solamente dal componente voluto.
- **Message List:** Questa finestra verrà riempita dai dati provenienti dai moduli.

Capitolo 5
SPI-SERIAL PERIPHERAL INTERFACE

Il serial peripheral interface è il secondo protocollo di comunicazione utilizzato. Come nel capitolo precedente verrà inizialmente fornita la definizione di tale protocollo e come esso funziona. Negli ultimi due paragrafi del capitolo verrà descritto come l'SPI è usato per nostri scopi. In particolare sarà trattata la comunicazione tra micro-master e micro-slave e tra microcontrollori ed encoder.

5.1 **Definizione**

L'SPI è un sistema di comunicazione seriale di tipo sincrono che opera in modalità di trasmissione full duplex, ovvero è in grado di trasmettere e ricevere dati digitali o analogici su un unico canale di comunicazione. Poiché la comunicazione è di tipo sincrono sarà necessario adibire una linea per il segnale di clock. I dispositivi connessi mediante SPI possono essere di due tipi: master o slave. Di solito esiste un unico master e uno o più slave. La comunicazione avviene mediante 4 linee:

- MOSI (Master Output, Slave Input): è la linea mediante la quale i dati viaggiano dal master verso lo slave.
- MISO (Master Input, Slave Output): è la linea mediante la quale i dati viaggiano dallo slave verso il master.
- SCLK (serial clock): su questa linea il master invia il clock.

- SS (Slave Select): nel caso in cui siano presenti più slave, questa linea viene utilizzata dal master per scegliere lo slave al quale inviare i dati. E' una linea di tipo active low ovvero per abilitare lo slave desiderato è necessario porre a livello logico '0' questa linea.

Vediamo ora alcuni vantaggi e svantaggi del SPI:

Vantaggi:

- Comunicazione di tipo full duplex.
- Comunicazione sincrona.
- Capacità di trasmissione maggiore rispetto a I^2C.
- Possibilità di scelta della dimensione del messaggio (8 o 16 bit).
- Non necessita di circuiti di pullup come nel caso dell'I^2C.
- Gli slave non hanno la necessità di utilizzare oscillatori ad alta precisione in quanto è il master stesso che decide la frequenza di clock e lo invia sull'apposita linea.
- Non sono necessari tranceiver.

Svantaggi:

- Sono necessarie almeno 3 linee (MISO, MOSI, SCLK), quindi occorre utilizzare almeno 3 pin del microcontrollore.
- E' possibile l'uso di massimo un componente master.
- Nessun ack di tipo hardware da parte dello slave (sarà necessario impostare l'invio di un ack nel firmware dello slave).
- Non è definito nessun protocollo riguardo l'error-checking.

5.2 **Comunicazione SPI: MicroMaster-MicroSlave**

Analizzo ora il caso particolare della comunicazione SPI tra i due microcontrollori scelti per la gestione del modulo. Come detto nel paragrafo 3.2 i micro sono della famiglia dsPIC33F. Per entrambi è possibile schematizzare il modulo SPI mediante lo schema a blocchi rappresentato in Figura 5.1.

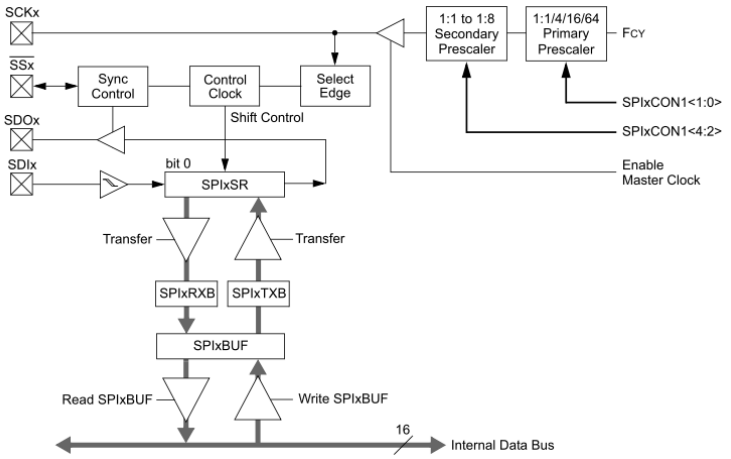


Figura 5.1 Schema a blocchi del modulo SPI dei micro dsPIC33F.

I quattro pin utilizzati sono SCKx, SSx, SDOx, SDIx (la x può assumere valore 1 o 2, in quanto sulla famiglia di micro utilizzata sono presenti due moduli SPI). Essi rappresentano rispettivamente: il pin sul quale mandare o ricevere il clock (dipende se il micro considerato è master oppure slave); il pin per mandare o ricevere lo slave select; il pin sul quale inviare i dati; il pin sul quale ricevere i dati.[26]

Prima di cominciare la comunicazione è necessario che il master configuri il clock. La frequenza di quest'ultimo deve essere minore o uguale alla massima frequenza dello slave. La frequenza verrà impostata settando opportunamente il prescaler primario e secondario. In questo modo verrà diminuita la frequenza del clock interno al master (Fcy).

Com'è possibile notare in Figura 5.1 sia i dati in ingresso che quelli in uscita vengono gestiti dal medesimo registro (shift register SPIxSR). Una volta che un nuovo messaggio è presente al pin SDI, esso verrà salvato nello shift register, trasferito al SPIBUF e scatenata l'interrupt di ricezione del messaggio. Viceversa,

quando il microcontrollore vuole inviare un messaggio all'altro micro, basterà scrivere il messaggio sul buffer SPIBUF. Successivamente il pacchetto verrà mandato allo shift register che si occuperà di inoltrarlo sul canale attraverso il pin di uscita SDO. Sia in trasmissione che in ricezione, l'utente ha la sola possibilità di interagire con il buffer SPIBUF e non con gli altri registri. Da un lato questo potrebbe costituire un problema, in quanto pone l'utente ad un più alto livello di astrazione, quindi minor controllo. Dall'altro lato però un vantaggio sta nel fatto che, mentre lo shift register sta inviando il dato sul bus, l'utente può scrivere il nuovo messaggio sul SPIBUF. In questo modo non si viene a creare nessun problema di sovrascrizione, e quindi perdita, del dato. [27]

5.3 **Comunicazione SPI: Micro-Encoder**

I pacchetti di dati scambiati tra micro ed encoder sono costituito sempre da 16 bit. A seconda del tipo di pacchetto, essi sono costituiti da diversi campi. Le varie tipologie di pacchetto sono: comando, lettura, scrittura.

Nel pacchetto di comando il bit più significativo RWn indica se il comando è di lettura oppure di scrittura. I successivi 14 bit rappresentano il codice indirizzo. Infine il bit meno significativo PAR è un bit di parità ed è ottenuto mediante un algoritmo sui primi 15 bit del pacchetto. Nel caso in cui il comando sia di lettura, il bit RWn verrà posto a livello logico alto.

Bit	MSB	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
	RWn	Address<13:0>														PAR

Figura 5.2 Pacchetto di comando.

Nel pacchetto di lettura (lettura dall'encoder), invece, i primi 14 bit rappresentano il dato vero e proprio. Il penultimo bit è un error flag. Questo bit indica se nella precedente trasmissione si è verificato un errore. Infine il bit meno significativo è ancora il bit di parità.

Bit	MSB	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
	Data <13:0>														EF	PAR

Figura 5.3 Pacchetto di lettura.

La terza tipologia di pacchetto che il micro e l'encoder possono scambiare è quella di scrittura (scrittura verso encoder). La struttura è molto simile alla precedente tipologia, con i primi 14 bit di dati e il bit meno significativo di parità, con la differenza che non è presente un bit di errore.

Bit	MSB	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
	Data <13:0>														Don't care	PAR

Figura 5.4 Pacchetto di scrittura.

Descrivo ora le modalità di interconnessione tra encoder e micro. Come detto all'inizio del capitolo vi sono 4 pin che devono essere collegati per poter effettuare una comunicazione utilizzando l'SPI : MOSI, MISO, SCK, SS. Ovviamente nella comunicazione considerata il master è il microcontrollore e lo slave è l'encoder. Come è possibile osservare in Figura 5.5 esistono 3 modalità di interconnessione: modalità a 4 fili, modalità a 3 fili di sola lettura, modalità a 3 fili bidirezionale.
Nella prima modalità tutti è quattro i pin per la comunicazione SPI devono essere collegati. Avremo quindi una comunicazione dal pin MOSI del master al pin MOSI dello slave; una dal MISO dallo slave al MISO del master; una per la linea di clock e per la linea slave select dal master verso lo slave. In questa modalità quando il micro desidera conoscere l'angolo registrato dall'encoder (e quindi lo spostamento angolare del giunto) invia un comando (di lettura) e allo stesso tempo pone a livello logico basso lo slave select. Dopo aver fatto ciò ripone a livello logico alto l'SS per un determinato intervallo di tempo predefinito. Alla trasmissione successiva l'encoder invierà sulla linea MISO l'angolo richiesto precedentemente. Contemporaneamente a ciò il master può richiedere nuovamente l'angolo registrato dall'encoder.

Nella modalità a 3 linee (sola lettura) ad ogni ciclo dello slave select l'encoder invierà informazioni riguardo l'angolo letto. In questa modalità quindi il master ha la possibilità di decidere solamente l'istante in cui ricevere informazioni dall'encoder, ovvero quando viene disabilitato l'SS. Da notare che in questa modalità non è necessario collegare i pin MOSI dei due componenti.

Nell'ultima modalità i pin MOSI e MISO dell'AS5055 sono entrambi collegati ad un unico pin del micro. Si alternano invii di richieste di lettura da parte del micro a invii di angoli letti da parte dell'encoder. Tutto questo sulla stessa linea. Ovviamente bisogna fare molta attenzione nel sincronizzare al meglio la comunicazione. Uno svantaggio di questa modalità riguarda ovviamente la frequenza di comunicazione dell'angolo che è il doppio rispetto alle modalità precedenti.

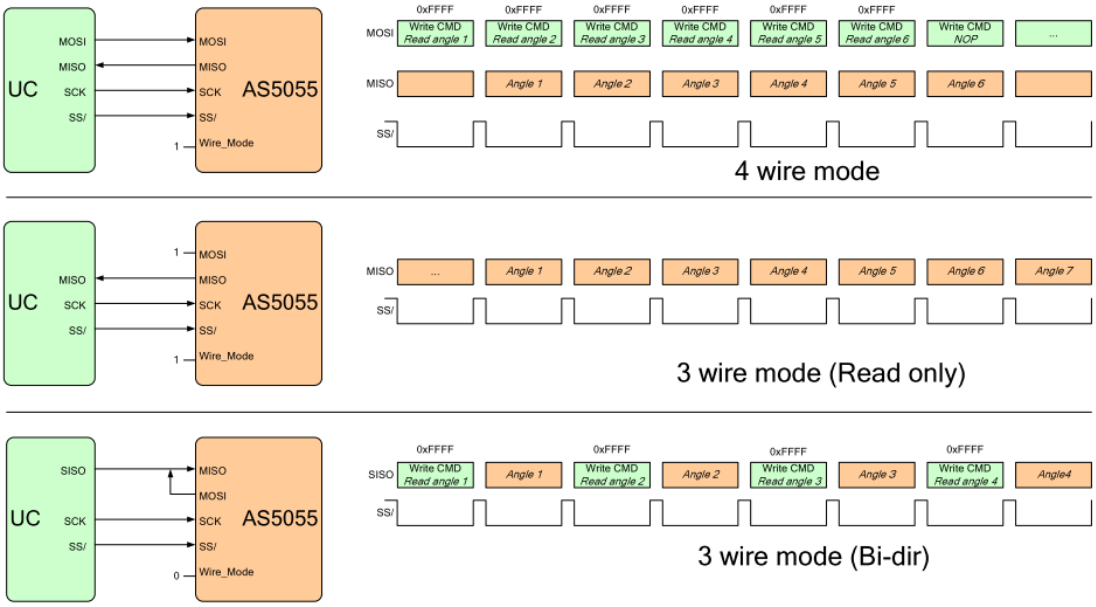


Figura 5.5 Modalità di connessione.

Capitolo 6
PROTOCOLLO DI COMUNICAZIONE

In questo capitolo verrà descritto il protocollo di comunicazione strutturato per il robot snake. Viene valutato il numero di componenti totali, i possibili comandi e dati da inviare e su questo verrà costruito il protocollo.

6.1 **Definizione protocollo**

Prima di cominciare a definire il protocollo è necessario riassumere lo schema generale del robot. Vi sono i moduli iniziali, uguali l'uno con l'altro e poi vi è il modulo finale nel quale è presente la telecamera e i sensori ad essa associati. Supponendo di considerare quindi 3 unità per il corpo del robot e 1 unità per la telecamera avremo un totale di 4 macro-unità. A queste corrispondono altrettanti microcontrollori-master.
Analizziamo più in dettaglio le due tipologie di modulo. Quella del corpo del robot è costituita da: un micro-master al quale sono associati un motore e un encoder; un micro-slave che gestisce un motore, un encoder, un sensore di forza. Avremo quindi un totale di 5 componenti.
L'unità-telecamera invece è caratterizzata da un unico micro che comanda un CMOS per lo zoom, 2 motori piezo elettrici per lo spostamento delle lenti, 2 encoder lineari per rilevare lo spostamento delle lenti, un LED per l'illuminazione dell'area d'esame, un fotosensore che rileva questa luminosità. Totale: 7 componenti.

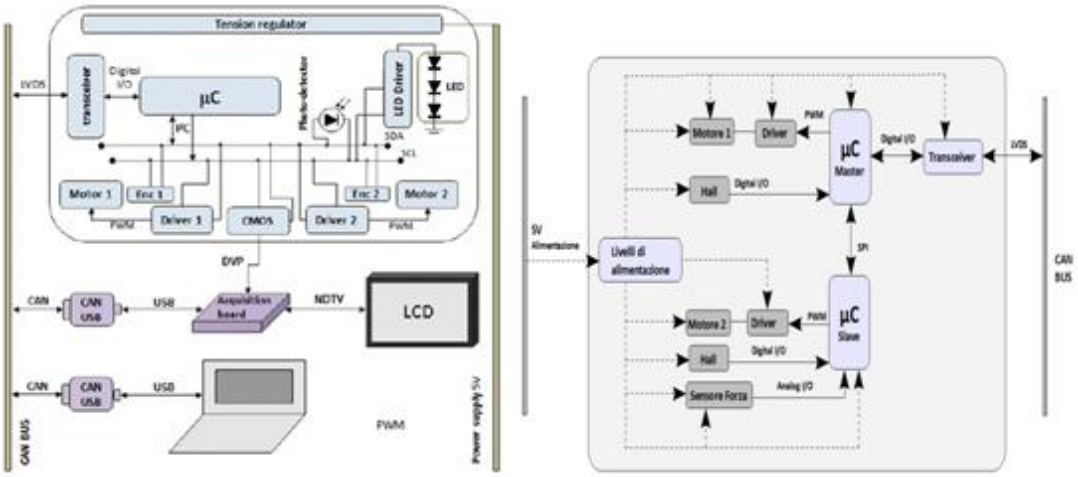


Figura 6.1 A sinistra: schema a blocchi del modulo della telecamera.
A destra: schema a blocchi del modulo corpo del robot.

Il totale quindi dei componenti dell'intero apparato è quindi 12. Ad ogni componente è possibile associare un comando e per alcuni anche uno o più dati (riassunti in Tabella 6.1): il motore deve ruotare in verso orario o antiorario di un determinato angolo e con un a determinata velocità; gli encoder angolari dovranno fornire l'angolo di rotazione del giunto; i sensori di forza dovranno fornire la forza di interazione del modulo con l'ambiente; lo zoom deve essere settato ad un determinata percentuale tra 0% e 100%; l'encoder lineare deve fornire lo spostamento lineare delle lenti; il LED deve essere settato a un valore compreso tra 0% e 100%; il fotosensore deve fornire la quantità di illuminazione osservata; i motori piezoelettrici devono spostarsi in avanti o indietro di micrometri con una certa velocità. Per ogni componente è definito anche un comando di verifica della connessione, in modo tale da poter controllare, anche ciclicamente, lo stato del sistema.

Componente	Comando	Dato
Motore (1,2)	- Ruota verso orario di un angolo x con velocità y - Ruota verso antiorario di	- Angolo - Velocità

	un angolo x con velocità y - Verifica connessione	
Encoder (1,2)	- Fornire angolo - Verifica connessione	
Sensore di forza	- Fornire forza - Verifica connessione	
Zoom	- Setti zoom a x % - Verifica connessione	- Percentuale zoom tra 0% ÷ 100%
Encoder lineare(1,2)	- Fornire spostamento - Verifica connessione	
Fotosensore	- Fornire illuminazione - Verifica connessione	
LED	- Setta luminosità - Verifica connessione	- Luminosità 0% ÷ 100%
Motore piezo(1,2)	- Muovi avanti di x µm con velocità y - Muovi indietro di x µm con velocità y - Verifica connessione	- Spostamento - Velocità

Tabella 6.1 Componenti, comando e dati dell'intero robot

Riporto in Figura 6.2 lo schema del messaggio standard inviato sul CAN-bus. Ricordo che gli unici bit modificabili sono quelli dell'identificativo e dai campo dati. Su questa osservazione ho avuto l'idea di come definire il protocollo. In particolare il campo identificativo specificherà il modulo, quindi il micro-master, al quale inviare il dato. Nel caso sopracitato con 4 macro-unità, questo campo assumerà valori 01, 02, 03, 04. Ricordo che ogni micro leggerà il frame inviato sul canale, nel caso in cui l'ID del micro coincide con quello del pacchetto allora si procederà con la decodifica, altrimenti il frame verrà ignorato.

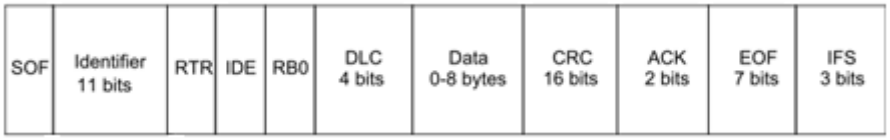


Figura 6.2 Standard frame

Nel campo dati, costituito da 8 byte, inserisco invece l'informazione legata al componente, al comando e ai dati (Tabella 6.2).

Byte n°	
0	Componente
1	Comando
2-3	Dato 1: angolo, zoom, luminosità, spostamento
4	Dato 2 : velocità
5	
6	
7	

Tabella 6.2 Numero byte con relativa informazione ad esso associato

Il protocollo sarà quindi impostato in questo modo:

Campo identificativo:

- ID ------ N° modulo

Campo dati :

Byte 0:

- 0x01 ------ Motore 1
- 0x02 ------ Motore 2
- 0x03 ------ Encoder 1
- 0x04 ------ Encoder 2
- 0x05 ------ Sensore forza

- 0x06 ------ Zoom
- 0x07 ------ Encoder lineare 1
- 0x08 ------ Encoder lineare 2
- 0x09 ------ Fotosensore
- 0x10 ------ LED
- 0x11 ------ Motore piezo 1
- 0x12 ------ Motore piezo 2

Byte 1:

- 0x01 ------ Ruota verso orario
- 0x02 ------ Ruota verso antiorario
- 0x03 ------ Fornire angolo
- 0x04 ------ Fornire forza
- 0x05 ------ Setta zoom
- 0x06 ------ Fornire spostamento
- 0x07 ------ Fornire illuminazione
- 0x08 ------ Setta illuminazione
- 0x09 ------ Muovi avanti
- 0x10 ------ Muovi indietro
- 0x11 ------ Verifica connessioni

Byte 2-3:

- 0x0000 ÷ 0xFFFF ------ Angolo, zoom, luminosità, spostamento

Byte 4:

- 0x00 ÷ 0xFF ------ Velocità

Capitolo 7
SCHEMATICI ELETTRONICI

Nel seguente capitolo verrà riportata una breve descrizione dei circuiti di condizionamento e delle connessioni dei vari componenti che caratterizzeranno la mia board e infine lo schematico totale realizzato col programma Eagle della CADSOFT. Per il PCB occorre conoscere lo spazio disponibile all'interno del modulo e poiché il CAD del modulo è ancora in fase di progetto, bisognerà attendere.

7.1 **Condizionamento componenti**

I componenti per il condizionamento dei micro sono condensatori di disaccoppiamento tra l'alimentazione e la massa. Questi capacitori devono essere messi quanto più vicino possibile ai pin di alimentazione e sullo stesso layer dove è saldato il micro. E' consigliato sul datasheet del componente di posizionarli ad una distanza massima di 6 mm. I condensatori devono avere tutti valore di 0.1 uF e devono essere di tipo ceramico. Per ridurre al massimo lo spazio occupato dai condensatori ho scelto il package 0201, corrispondenti a 0.6x0.3mm. Il pin MCLR utilizzato per la programmazione e il debugging richiede una resistenza di 10 kΩ tra il pin e VDD.

I driver necessitano di un condensatore di disaccoppiamento tra Vcc e COM (corrispondente a massa), ed anche tra i pin VB e VS. Tra i piedini VCC e VB è

richiesto il posizionamento di un diodo. Tra i pin di uscita dei driver e i gate dei mosfet interpongo due resistenze di valore 33Ω. Queste resistenze, una per HO e una per LO, sono usate per rallentare il funzionamento dell'interruttore ad un valore di/dt fisso, ed inoltre per ridurre oscillazioni in RF indesiderate [24].
Il transceiver per la comunicazione CAN presenta anch'esso il condensatore di disaccoppiamento tra alimentazione e massa. Se il nodo (modulo) considerato è il primo o l'ultimo connesso al bus di comunicazione CAN è necessario porre una resistenza di terminazione tra i pin CANH e CANL di valore 120 Ω. È opportuno collocare queste resistenze poiché come già detto nei capitoli precedenti il bus è composto da un doppino che connette tutti i nodi (moduli). Essendo un bus occorre che ai suoi estremi vi sia un adattatore di impedenza chiamato appunto terminatore realizzato dalle due resistenze [25]. L'impedenza di linea sarà quindi di 60 Ω. Ponendo inoltre una resistenza tra il pin RS e massa è possibile decidere, a seconda del valore, se il transceiver lavori in high-speed mode (connettendo direttamente a massa), in control slope mode (valore tra 25 kΩ e 200 kΩ) oppure in standby mode (connettendo il pin direttamente a VDD).
Il pin di alimentazione VDD dell'encoder effetto hall non deve essere collegato direttamente all'alimentazione, ma è consigliato collegarlo ad essa attraverso un filtro RC. In questo modo si evita che fluttuazioni del segnale di alimentazione causino delle variazione del segnale in uscita all'encoder. I valori di resistenza e capacità sono: R=15 Ω e C=4.7uF.

7.2 **Connessione tra i componenti**

Sia il micro master che per il micro slave devono essere programmati. Connetto quindi pin PGED3, PGEC3, MCLR del micro a dei PAD sui quali verrà poi collegato il programmatore. Quest'ultimo necessita anche di un pin di alimentazione e di un pin di massa.
Il movimento dei motori viene effettuato mandando un segnale PWM ai driver. Poiché i motori sono trifase è necessario utilizzare 3 moduli PWM del micro e quindi

bisogna utilizzare 6 pin di uscita: PWM1L1, PWM1H1, PWM1L2, PWM1H2, PWM1L3, PWM1H3. Questi devono essere opportunamente collegati ai pin HIN, LIN dei driver. Le uscite HO e LO, di ogni singolo driver, sono collegate rispettivamente ai gate G1 e G2 dei due MOSFET. Il drain del primo transistor è collegato alla tensione di alimentazione 3.3 V mentre il source del secondo direttamente a massa. Il source del primo mosfet e il drain del secondo sono collegati tra di loro ed anche all'uscita Vs del driver. Questa linea andrà poi ad un' avvolgimento del motore. Le stesse medesime connessioni vanno fatte per gli altri due driver collegati al micro master e per gli altri tre del micro slave.

I pin RB7 ed RB9 del dsPIC master sono stati scelti per la comunicazione CAN. Il primo per la ricezione e il secondo per la trasmissione. È opportuno quindi tracciare una linea che colleghi il piedino RB7 del micro con il pin RXD del tranceiver ed una linea che unisca RB9 con TXD.

Come già descritto nel paragrafo 5.3, la comunicazione tra micro ed encoder è di tipo SPI. Collego i pin MOSI, MISO, SCK, SS dell'encoder con i pin scelti per la comunicazione SPI del micro: RP4 per il master output, RP5 per il master input, RP6 per lo slave select output, RP8 per il clock output. Tutto quanto appena detto vale per entrambi i microcontrollori.

Il sensore di forza è posizionato esternamente alla board. Metto quindi sulla board un pad per il collegamento col sensore. Collego con una pista il pad e il pin RA0 del dsPIC slave scelto per la lettura del segnale analogico.

Le ultime connessioni da fare sono quelle tra i due dsPIC. La comunicazione è ancora di tipo SPI, quindi sono necessari 4 pin di ogni micro. Per il micro-master imposto: RB0 come SPI data input; RB1 come SPI data output; RB2 come SPI slave select output; RB3 come SPI clock output. Per il micro slave imposto: RB0 come SPI data input; RB1 come SPI data outpu; RB2 come SPI slave select input; RB3 come SPI clock input. Connetto quindi i pin RB0, RB1, RB2, RB3 del controllore master rispettivamente con RB1, RB0, RB2, RB3 dello slave.

7.3 **Schematico**

Riporto di seguito gli schematici realizzati. Poiché non è stato possibile inserirli tutti in un'unica figura, li divido in 3 figure.

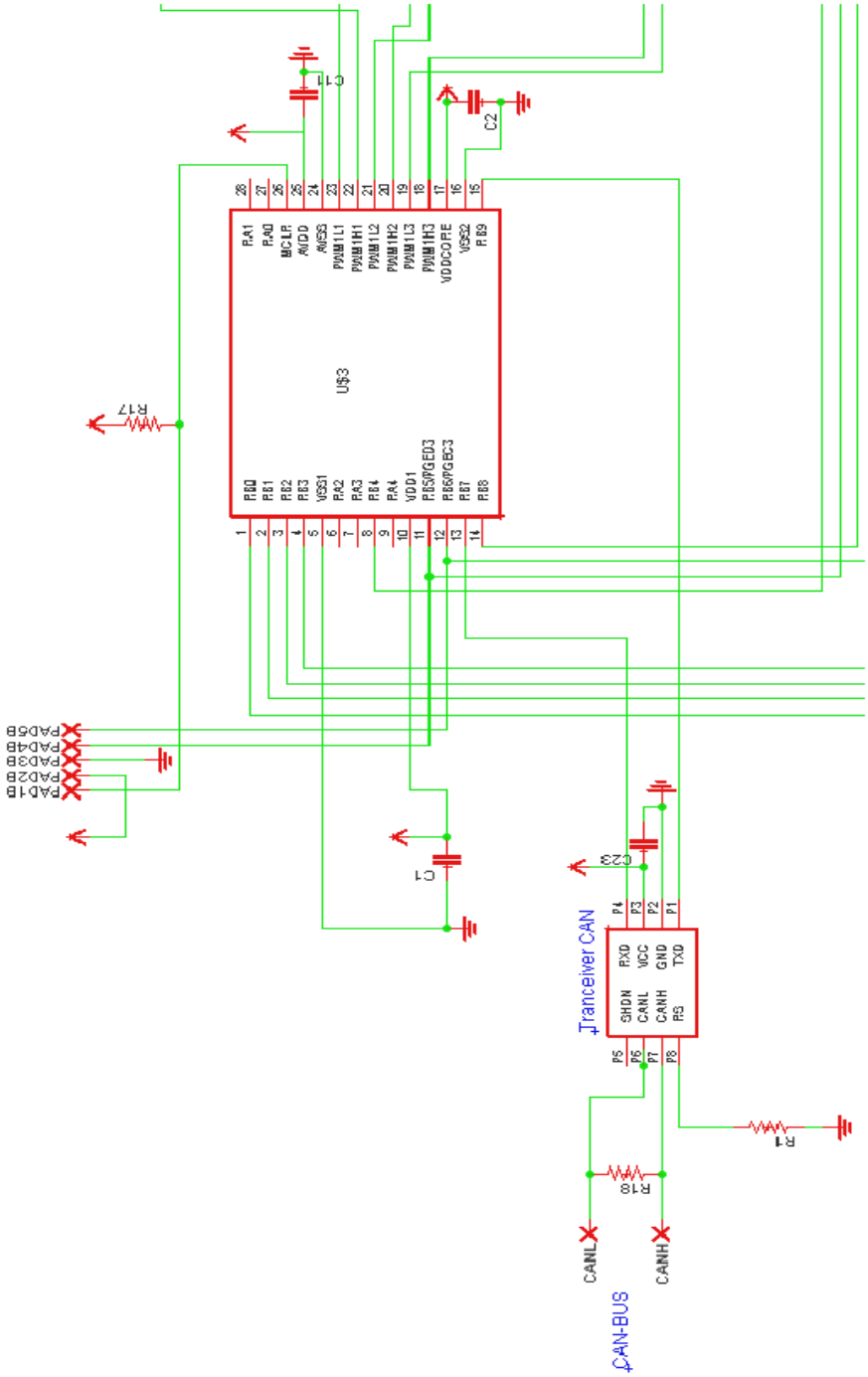


Figura 7.1 Micro-master, tranceiver, connettore per programmare.

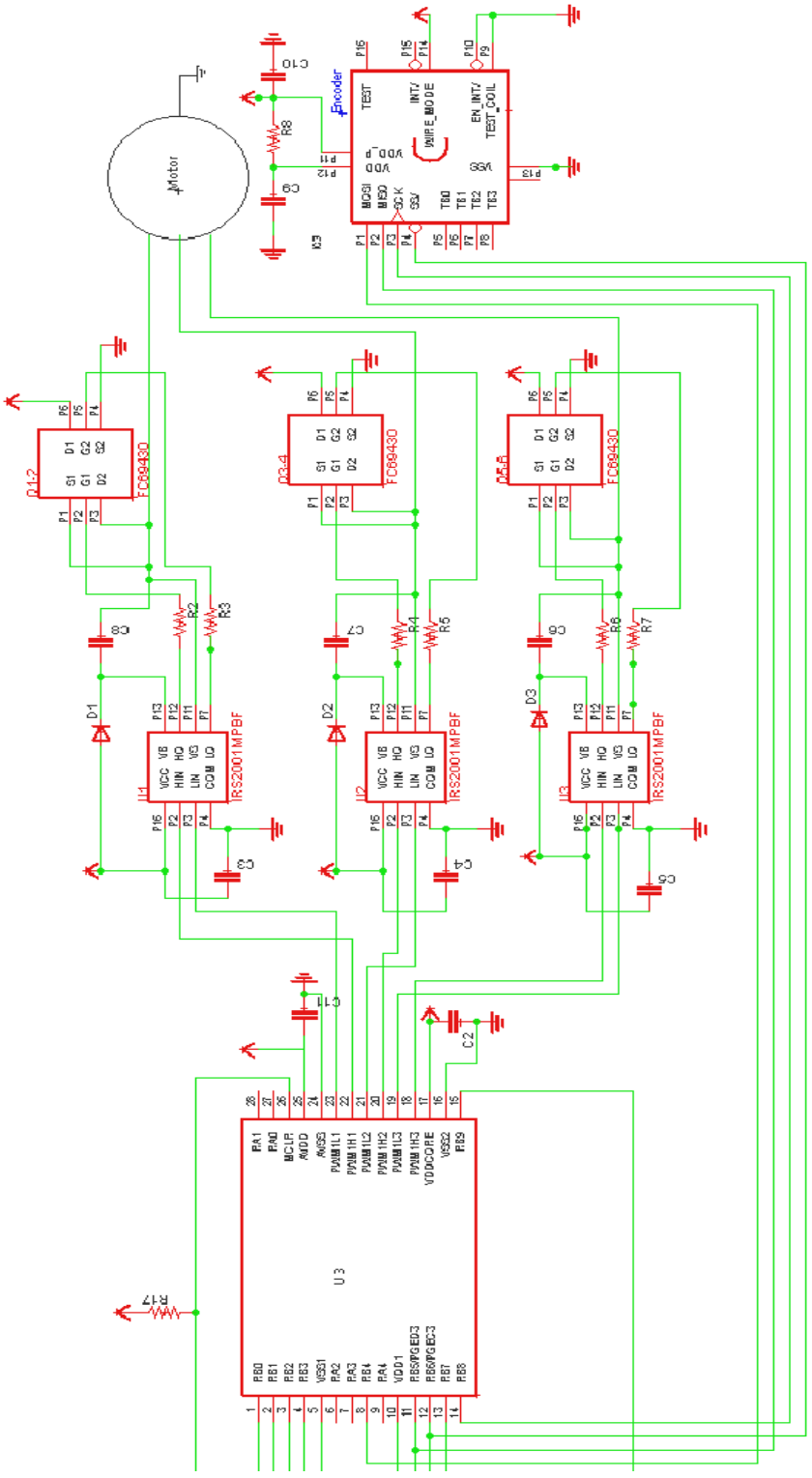


Figura 7.2 Micro, driver, mosfet, encoder effetto Hall

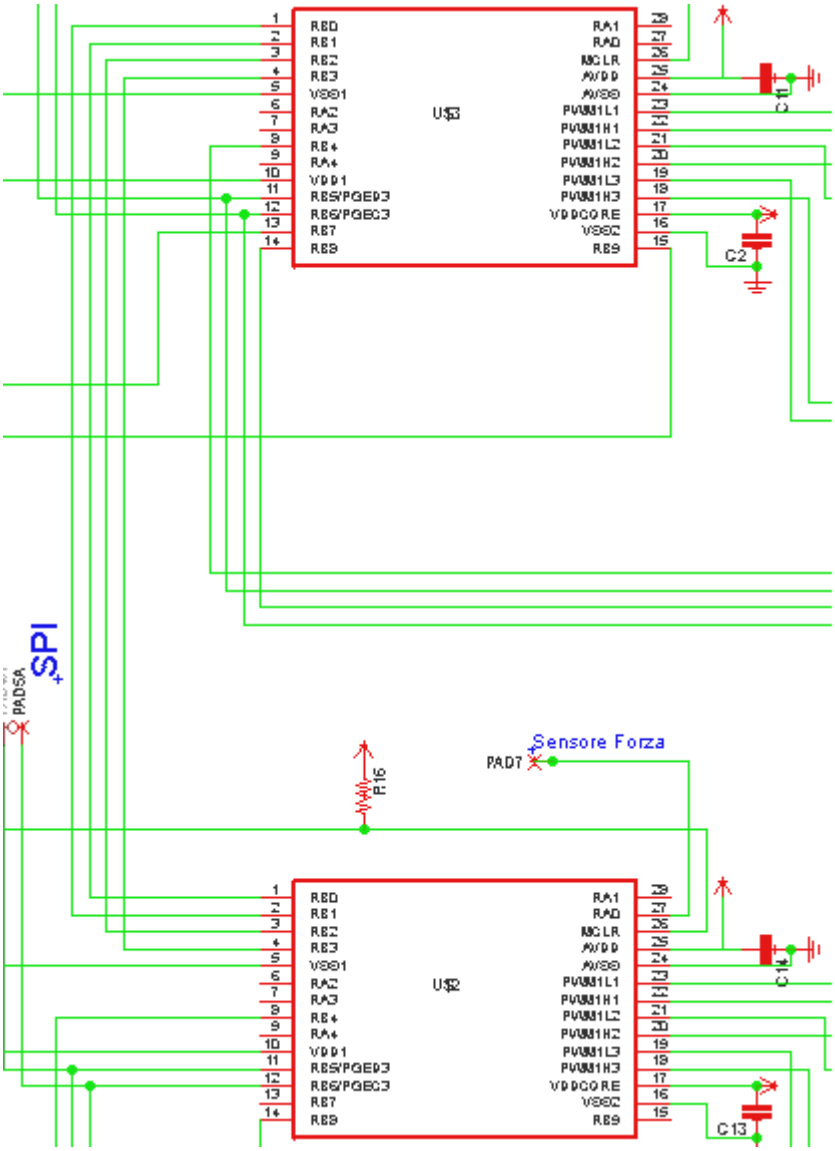


Figura 7.3 Micro-master, Micro-slave

Capitolo 8
CONTROLLO MOTORI E SENSORI

Il capitolo descrive la modalità di controllo dei motori ed dei sensori a bordo. Per i motori il controllo avverrà inviando un segnale PWM e la retroazione potrà avvenire in due modi: mediante la Back Electromotive Force oppure più semplicemente mediante encoder. Verrà trattato alla fine del capitolo la procedura di controllo del sensore di forza.

8.1 **Controllo motori ed encoder**

Il metodo per fornire energia agli avvolgimenti del motore sensorless, da noi utilizzato in questa applicazione, è quello della commutazione a sei stati detto anche “120° commutation”. La Figura 8.1 illustra il principio di funzionamento.

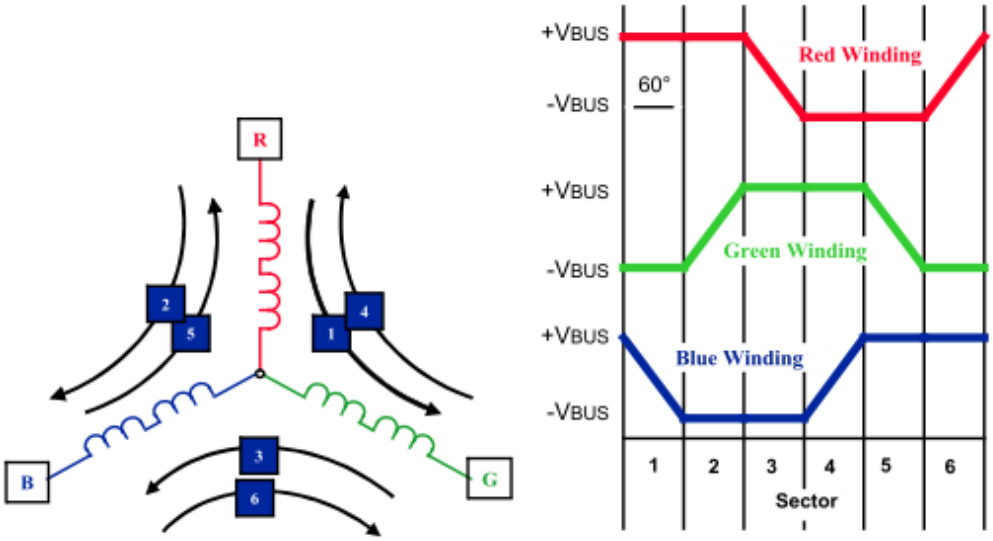


Figura 8.1 Commutazione a sei stati

Ogni step o settore è equivalente a 60 gradi *elettrici*. Sei settori compongono quindi 360 gradi *elettrici* ovvero una intera rivoluzione *elettrica.* Le frecce nella figura di sinistra mostrano la direzione nella quale scorre la corrente attraverso gli avvolgimenti del motore in ognuno dei sei settori. Il grafico sulla destra mostra il potenziale medio applicato ad ogni bobina del motore durante i vari step. Un' intera sequenza di questi sei passi muove l'albero motore di un'intera rivoluzione. Si parla di potenziale medio in quanto il segnale inviato al motore è un segnale ti tipo PWM. Un segnale PWM (Pulse Width Modulation ovvero modulazione a variazione della larghezza d'impulso) è un' onda quadra di duty cycle variabile che permette di controllare l'assorbimento (la potenza assorbita) di un carico elettrico(nel nostro caso il motore DC), variando (modulando) il duty cycle. Un segnale PWM è caratterizzato dalla frequenza (fissa) e dal duty cycle (variabile) ovvero il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T (l'inverso della frequenza: T=1/f). Ne segue che un duty cycle del 50% corrisponde ad un'onda quadra che assume valore alto per il 50% del tempo, un duty cycle dell'80% corrisponde ad un'onda quadra che assume valore alto per l'80% del tempo e basso per il restante 20%, un duty cycle del 100% corrisponde ad un segnale sempre alto e un duty cycle dello 0% ad un segnale sempre basso[8].

Analizzando più in dettaglio i sei step.

- Step 1
    - Avvolgimento rosso è settato positivo
    - Avvolgimento verde è settato negativo
    - Avvolgimento blu non è settato
- Step 2
    - Avvolgimento rosso è settato positivo
    - Avvolgimento blu è settato negativo
    - Avvolgimento verde non è settato
- Step 3
    - Avvolgimento verde è settato positivo
    - Avvolgimento blu è settato negativo
    - Avvolgimento rosso non è settato
- Step 4
    - Avvolgimento verde è settato positivo
    - Avvolgimento rosso è settato negativo
    - Avvolgimento blu non è settato
- Step 5
    - Avvolgimento blu è settato positivo
    - Avvolgimento rosso è settato negativo
    - Avvolgimento verde non è settato
- Step 6
    - Avvolgimento blu è settato positivo
    - Avvolgimento verde è settato negativo
    - Avvolgimento rosso non è settato

Per ogni settore, due avvolgimenti sono "energizzati" mentre uno non lo è. Il fatto che una delle bobine non è energizzata durante ogni step è una caratteristica molto importante del controllo a sei stadi che permette di poter utilizzare un metodo di retroazione di basato BEMF, back electro motive force (vedi prossimo paragrafo).

Entrambi i micro controllano un motore. Questo avviene attraverso il modulo PWM presente al proprio interno. Nel firmware del micro settiamo la frequenza del segnale PWM pari a 20 kHz . Sarà poi il microcontrollore stesso a far variare il valore del duty cycle settando opportunamente un registro. Più è alto è il valore del duty cycle più alto sarà il valore medio del segnale PWM in uscita, quindi i motori gireranno più velocemente. Viceversa più sarà basso il valore, più i motori ruoteranno lentamente. I segnali PWM in uscita sono 3 in quanto i motori da noi utilizzati sono trifase. Quindi è stato anche necessario sincronizzare opportunamente i tre segnali. Per controllare ogni singola fase del motore sono necessari 2 piedini del micro. Infatti è necessario un piedino PWMH (PWM high) e uno PWML (PWM low). Quindi servono un totale di 6 pin. La mancanza di pin a disposizione per controllare due motori contemporaneamente da un singolo micro, è stato il motivo principale della scelta di 2 microcontrollori.

8.1.1 **Metodo di retroazione basato sulla BEMF-Back Electro Motive Force**

Quando un motore brushless in corrente continua (BLDC) ruota, ogni avvolgimento genera BEMF la quale si oppone alla tensione di alimentazione applicata agli avvolgimenti secondo la famosa legge di Lenz. La polarità della BEMF è nella direzione opposta alla tensione energizzante [28]. La BEMF dipende principalmente da tre parametri del motore:

- Numero di avvolgimenti nello statore
- Velocità angolare del rotore
- Campo magnetico generato dal rotore magnetico

La BEMF può essere calcolata mediante la formula:

$$BEMF = NlrB\omega$$

dove:

N = numero di avvolgimenti per ogni fase

l = lunghezza del rotore

r = raggio interno del rotore

B = campo magnetico generato dal rotore

ω = velocità angolare

Della formula precedente l'unico termine variabile è la velocità angolare. La BEMF, quindi è proporzionale alla velocità del rotore.

La BEMF viene calcolata nell'istante in cui la tensione di alimentazione di ogni singola fase passa per lo zero (zero crossing). Per poter rilevare lo zero crossing è necessario utilizzare il centro stella dei tre avvolgimenti che si trova sempre alla tensione nulla. Il segnale BEMF è un segnale analogico. È quindi opportuno collegare in retroazione ogni singola fase ad un pin analogico del microcontrollore (vedi Figura 8.2) . Il segnale sarà poi opportunamente filtrato e mediante la formula sopracitata sarà possibile calcolare la velocità angolare e quindi anche lo spostamento angolare. Quest'ultimo segnale sarà confrontato (sottratto) con lo spostamento desiderato. Questo valore verrà poi inviato ad un controllore PI (proporzionale-integrale). Il segnale in uscita verrà poi gestito dal micro che muoverà di conseguenza il motore in una direzione o nell'altra oppure nel caso in cui lo spostamento reale e quello desiderato coincidono non manderà segnali in uscita.

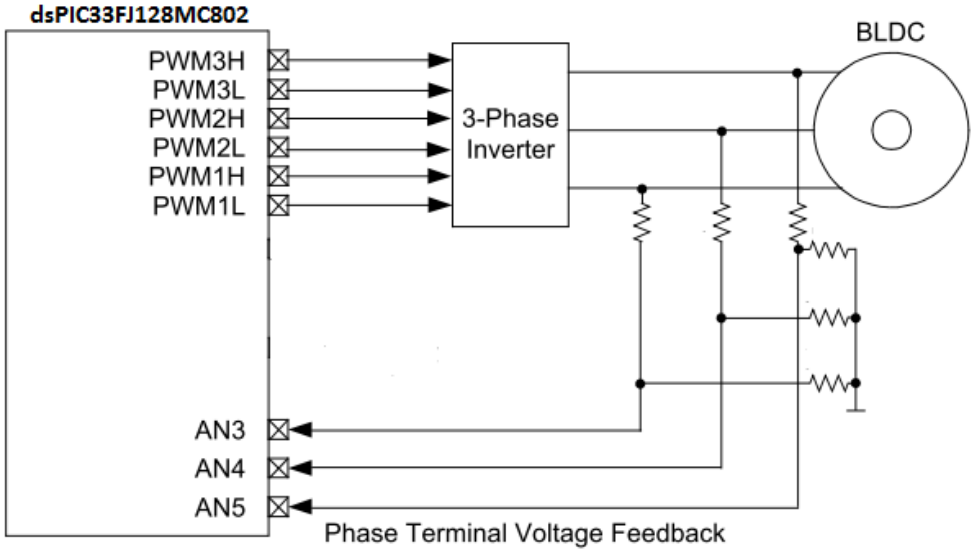


Figura 8.2 Connessione microcontrollore-motore con retroazione analogica

8.1.2 **Metodo di retroazione basato su Encoder**

Sebbene la retroazione basata su BEMF sia molto valida, ho deciso di adottare anche la retroazione classica per i sistemi robotici: ovvero la retroazione microcontrollore-encoder. Ho scelto di utilizzare anche questo metodo per i seguenti motivi:

- Il motore deve muoversi ad una velocità sufficientemente elevata per generare un segnale BEMF rilevabile dal micro;
- Bruschi cambiamenti al carico del motore possono generare un segnale BEMF non rilevabile;
- Il segnale BEMF è soggetto a disturbi elettromagnetici quindi una lettura errata di tale valore potrebbe portare ad un valore errato di spostamento angolare con relative conseguenze;
- Se è presente dell'offset sul centro stella non verrà rilevato con precisione il punto in cui la fase passa per lo zero e quindi non è possibile ricevere il segnale BEMF.

Il metodo di retroazione mediante encoder è molto semplice. L'encoder fornisce il valore dell'angolo del giunto che sta ruotando e come nel metodo precedente questo valore viene confrontato con uno spostamento desiderato e in questo modo calcolato un valore errore. Questo verrà opportunamente gestito dal controllore PI che invierà il risultato al micro che poi deciderà se continuare a fornire o no, energia al motore.
Nel paragrago 5.3 si è discusso delle varie modalità di connessione tra microcontrollore e encoder: 4 fili, 3 fili (sola lettura), 3 fili (bidirezionale). Ho scelto di utilizzare la prima tipologia.

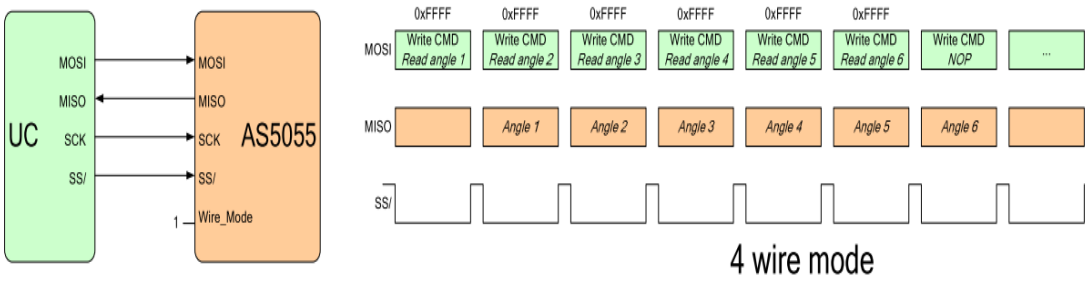


Figura 8.3 Modalità di connessione a 4 fili

Questa tipologia richiede l'utilizzo di 4 connessioni (MOSI master con MOSI slave, MISO slave con MISO master, SCK master con SCK slave, SS/ master con SS/ slave), il che implica 4 pin del micro. Inizialmente avevo deciso di adottare la 3 wire mode (sola lettura) per risparmiare così l'utilizzo del micro MOSI. In questa modalità però l'encoder invia a ripetizione i valori dell'angolo senza alcun controllo da parte del micro. In questo modo verrà riempito costantemente il buffer SPIxBUF del micro e quindi verrà generata costantemente l'interrupt. Questo ovviamente non è un bene in quanto viene interrotta costantemente la sequenza di comandi della CPU.
Come è possibile notare in Figura 8.3, con questa tipologia di connessione, il micro richiede, inviando il comando di lettura, l'angolo letto dall'encoder. Al ciclo successivo l'encoder invierà l'angolo letto e nello stesso istante il micro può inviare un nuovo comando di lettura. Quando il micro non ha più bisogno di ricevere informazioni dall'encoder, invia il comando NOP (no operation command) e

l’encoder smetterà di inviare dati. In questo modo verrà scatenato l’interrupt di ricezione SPI solamente quando necessario e quando voluto dal micro.

8.2 **Controllo sensore di forza**

Il sensore di forza utilizzato è costituito da una struttura ellittica in acciaio di dimensioni molto ridotte, la quale si deforma elasticamente se viene applicata un forza. Due strain-gauges termicamente saldati alla struttura forniscono un’uscita lineare, proporzionale alla forza applicata[29]. Quando viene impressa una forza trasversale alla struttura (vedi Figura 8.4), le pareti si deformano. Questo crea una flessione elastica che a sua volta varia il valore di resistenza degli estensimetri. Una calibrazione a priori della sonda consente di misurare la forza in Newton[30].

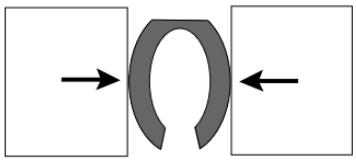


Figura 8.4 Applicazione di una forza trasversale alla struttura

Gli strain gauges vengono posti in configurazione a ponte di Wheatstone con altre due resistenze di valore costante e uguale tra di loro :

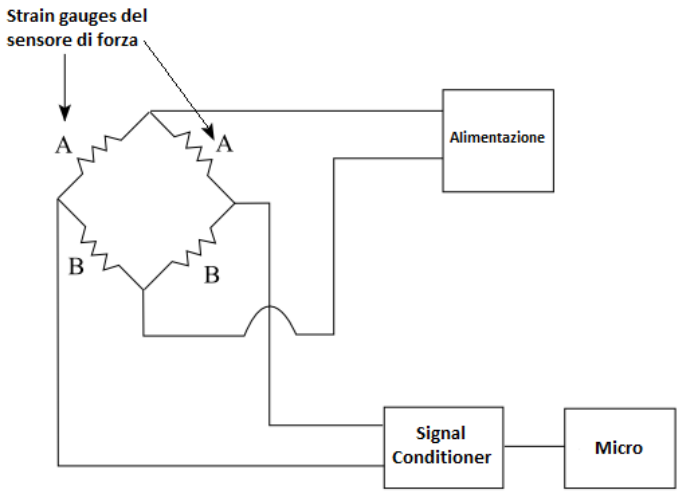


Figura 8.5 Schematico del circuito di condizionamento del sensore di forza

A due nodi del ponte di Wheatstone viene fornita una alimentazione costante. Sugli altri due nodi, invece viene rilevata la variazione di resistenza. Nel caso in cui non viene impressa nessuna forza alla struttura il valore di tensione tra questi due nodi è zero. Nell'istante in cui la sonda urta un tessuto, viene impressa una forza alla struttura in acciaio, la struttura si modifica, viene modificato il valore di resistenza degli estensimetri che modificheranno la condizione di equilibrio del ponte modificando il valore di tensione ai nodi. Questo segnale verrà opportunamente condizionato. Verrà in primo luogo amplificato e successivamente filtrato. Il valore di tensione a questo punto è pronto per poter essere inviato ad un pin analogico del micro. I pin analogici del dsPIC sono tutti collegati ad un convertitore analogico/digitale. Il valore di tensione analogico verrà quindi convertito in un valore di tensione digitale opportunamente codificato. Questo valore verrà confrontato ciclicamente con un valore massimo. Questo valore massimo dovrà essere calcolato mediante test di contatto della sonda con tessuti. Allo stato attuale dello sviluppo del lavoro di tesi non è stato ancora possibile effettuare questo calcolo in quanto non si è ancora in possesso del sensore. Si rimanda a lavori futuri il calcolo di questo valore ed anche la calibrazione del sensore di forza stesso.

Nel caso in cui il valore di forza risultasse maggiore del valore massimo, vuol dire che stiamo imprimendo una forza eccessiva sul tessuto e quindi necessario comandare i motori in direzione opposta in modo da evitare la rottura del tessuto.
Un possibile ulteriore impiego del sensore di forza riguarda la retroazione della forza. Si potrebbe pensare di utilizzare un interfaccia aptica. Un'interfaccia aptica è un dispositivo che permette di manovrare un robot, reale o virtuale, e di riceverne delle sensazioni tattili in risposta (retroazione o feedback). Un esempio potrebbe essere un joystick con ritorno di forza (force feedback) [31]. Nel caso quindi in cui il chirurgo stia imprimendo una forza troppo elevata con la sonda sul tessuto, questo potrebbe tradursi in una irrigidimento dell'interfaccia aptica (joystick che comanda la sonda). In questo modo si da una retroazione di forza al chirurgo.

Capitolo 9
FIRMWARE

Nel seguente capitolo non verrà descritto il firmware completo perché sarebbe troppo tedioso ma verranno analizzate solo le caratteristiche più importanti e degne di nota in quanto dettate da una riflessione e da scelte necessarie e obbligate. Questo capitolo si pone l'obiettivo di facilitare la comprensione del firmware. Bisogna seguire questo capitolo confrontandolo con l'appendice A dove è presente il codice completo.

9.1 **Impostazioni comuni**

In questo paragrafo illustro le impostazioni generali presenti nel firmware sia del master che dello slave.

```
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF);
_FWDT(FWDTEN_OFF);
_FICD(JTAGEN_OFF & ICS_PGD3);
```

Questi 4 comandi vengono posti subito dopo gli "include" delle varie librerie del microcontrollore, del can e dell'SPI (vedi appendice A). Il primo comando impone che l'oscillatore utilizzato non sia esterno (collegato ai pin OSC1 e OSC2) bensì

quello interno al micro da 7.37 Mhz. Non verrà quindi utilizzato un oscillatore esterno, il quale può essere sicuramente molto più preciso e può raggiungere frequenze maggiori. Si è optato verso la scelta dell'oscillatore interno per due motivi: lo spazio occupato sulla schedina da un oscillatore esterno è un ostacolo quando si parla di sistemi miniaturizzati come nel nostro caso; il secondo motivo riguarda invece la necessità di impiegare due pin del micro per ricevere il segnale dall'oscillatore che potrebbero essere utilizzati invece per altri scopi.
Con il secondo comando viene disabilitata la funzione di output del clock, ovvero non viene abilitato il pin OSC2 all'invio del segnale di clock. In questo modo il pin OSC2 diventa un normale pin di input/output, adoperabile per altri scopi.
Gli ultimi due comandi descritti sopra servono a disabilitare il watchdog timer, per evitare il reset del dispositivo dopo un certo intervallo di tempo, e il jtag. Viene anche impostato il debugging mediante il PGD3. Questa scelta è obbligata dal fatto che i pin del PGD1 e PGD2 servono per altri usi.
Successivamente vengono settate le variabili globali, in particolare i buffer canTxMessage e canRxMessage di tipo mID e definito il buffer per i messaggi ECAN allocati nella memoria ram del dma (vedi Capitolo 4).

```
mID canTxMessage;
mID canRxMessage;
ECAN1MSGBUF ecan1msgBuf __attribute__((space(dma),aligned
(ECAN1_MSG_BU F_LENGTH*16)));
```

Con quest'ultima impostazione tutti i messaggi proveniente dal modulo CAN del micro che arrivano al controllore DMA e successivamente alla sua memoria RAM verranno memorizzati nel buffer ecan1msgBuf. In questo modo quando verrà scatenato l'interrupt di ricezione di un messaggio CAN, saremo in grado di leggere il messaggio proprio in questo buffer.
Dopo questi settaggi si entrerà nel main. Viene lanciata la funzione oscConfig() per il settaggio dell'oscillatore interno il quale presenta i seguenti comandi:

```
PLLFBD=41;                  // M=43
CLKDIVbits.PLLPOST=0;       // N1=2
CLKDIVbits.PLLPRE=0;        // N2=2
OSCTUN=3;
```

Settando questi parametri viene aumentata la frequenza dell'oscillatore interno grazie al PLL (phase-locked loop) secondo la formula:

$$\mathrm{Fcy} = \frac{\mathrm{Fin} \times \mathrm{M}}{\mathrm{N1} \times \mathrm{N2} \times 2}$$

Nel mio caso Fin=7.37 MHz, quindi impostando M=43, N1=2, N2=2 otteniamo una frequenza di clock di circa 40 Mhz. Dopo aver settato opportunamente l'oscillatore, questo verrà abilitato, si attenderà poi un colpo di clock in modo tale da essere sicuri del corretto funzionamento (while (OSCCONbits.COSC != 0b001);) e successivamente si tornerà nel main.

9.2 **Firmware Master per la comunicazione**

9.2.1 **Inizializzazione porte**

Arrivati a questo punto del programma è necessario fare una distinzione tra firmware del microcontrollore master e dello slave.
Cominciamo dal settaggio delle porte di input/output mediante la funzione initport().
Per prima cosa imposto i pin di ingresso e di uscita mediante il registro TRISB. In particolare setto come pin di output i pin RB1, RB3, RB8, RB9 mentre come pin di input RB4 e RB7:

```
TRISBbits.TRISB1 = 0;
TRISBbits.TRISB3 = 0;
TRISBbits.TRISB4 = 1;
TRISBbits.TRISB7= 1;
TRISBbits.TRISB8 = 0;
```

```
TRISBbits.TRISB9= 0;
```

Ho scelto appositamente questi pin in quanto essi sono pin "rimappabili", ovvero possono essere associati a un particolare modulo o funzione presente sul micro (CAN, SPI, I^2C, digital, analog, ecc…). Per essere "mappabili" e associati ad una funzione particolare bisogna settare i seguenti registri:

```
RPINR26bits.C1RXR= 0b00111;
RPOR4bits.RP9R = 0b01010;
RPOR0bits.RP1R = 0b00111;
RPOR1bits.RP3R = 0b01000;
RPOR4bits.RP8R = 0b01001;
RPINR20bits.SDI1R = 0b0100;
```

Il primo e il secondo commando impostano rispettivamente RB7 come pin per la ricezione mediante modulo CAN e RB9 come pin per la trasmissione. Gli ultimi 4 invece servono per impostare l'SPI: RP1 come Data output; RP3 come clock output, RP8 come Slave Select output; RP4 come Data input.

9.2.2 **Inizializzazione SPI**

Per poter settare il modulo SPI è necessario innanzitutto disattivarlo, resettare la interrupt flag e disabilitare l'interrupt dell'SPI. Tutto ciò viene fatto per non avere problemi di crash del micro. I comandi sono i seguenti:

```
SPI1STATbits.SPIEN = 0;
IFS0bits.SPI1IF = 0;
IEC0bits.SPI1IE = 0;
```

Viene quindi impostato il registro di controllo principale del modulo SPI, ovvero l'SPI1CON, settando opportunamente i singoli bit. Per prima cosa viene abilitato il clock sul pin SCK. Poiché stiamo considerando il firmware del master si impone che

il pin SDO sia controllato dal modulo e non dall'esterno ed anche che la modalità master sia abilitata. Una riflessione è stata fatta per la lunghezza dei dati inviati. Si è deciso di utilizzare una comunicazione a 16 bit poiché l'encoder ad effetto Hall invia e riceve pacchetti a 16 bit. Successivamente vengono settati altri bit del registro. Non li riporto qui di seguito poiché non essenziali ai fini della trattazione, ma comunque visibili nell'appendice.

```
SPI1CON1bits.DISSCK = 0;
SPI1CON1bits.DISSDO = 0;
SPI1CON1bits.MSTEN = 1;
SPI1CON1bits.MODE16 = 1;
```

Ora il modulo SPI è pronto ad essere utilizzato, quindi viene riabilitato il modulo stesso, resettata ancora la flag dell'interrupt e abilitato l'interrupt:

```
SPI1STATbits.SPIEN = 1;
IFS0bits.SPI1IF = 0;
IEC0bits.SPI1IE = 1;
```

9.2.3 **Inizializzazione CAN**

L'inizializzazione del modulo in realtà è formata da due parti. Una prima parte dove si setta il CAN vero e proprio mediante la funzione initECAN; una seconda parte dove si inizializza il DMA mediante la funzione initDMAECAN. Entrambe le funzioni vengono richiamate nel main prima di entrare nel ciclo infinito while(1).

InitECAN()

Occorre innanzitutto porre il modulo in modalità di configurazione e attendere che il modulo invii la conferma:

```
C1CTRL1bits.REQOP=4;
while(C1CTRL1bits.OPMODE != 4);
```

La frequenza del modulo l'ho imposta pari a quella dell'oscillatore interno del micro, cioè Fcy che come visto nel paragrafo precedente è pari a 40 MHz:

```
C1CTRL1bits.CANCKS = 0x1;
```

Come detto in precedenza il controllore del DMA ha la possibilità di accedere ad un massimo di 32 buffer di ricezione e trasmissione presenti nella memoria RAM. Non verranno utilizzati tutti e 32 i buffer ma soltanto 4: ne vengono utilizzati 3 per la ricezione e 1 per la trasmissione. Dei 3 buffer per la ricezione il primo verrà riempito solamente se il messaggio in ingresso è di tipo standard mentre gli altri due se il messaggio è di tipo esteso. Preferisco utilizzare 2 buffer per il tipo esteso per una questione di sicurezza poiché utilizzando un unico buffer vi è il rischio di sovrascrizione di esso se i messaggi in input arrivano ad una frequenza troppo elevata.
Occorre prima di tutto settare i filtri di accettazione dei messaggi in ricezione (vedi Capitolo 4). Per far ciò abilito la modalità di configurazione dei registri dei filtri con il comando : `C1CTRL1bits.WIN=0b1`. Al buffer 1 associo il filtro di accettazione 0. Decido di prendere tutti i bit del SID del filtro per fare l'AND logico con il SID del messaggio in ingresso. Imposto poi il SID vero e proprio. Poiché è ancora una versione di prova ho impostato il SID pari a 123 in esadecimale. Dopodiché abilito il filtro.

```
C1FMSKSEL1bits.F0MSK=0;
C1RXM0SID=CAN_FILTERMASK2REG_SID(0x7FF);
C1RXF0SID=CAN_FILTERMASK2REG_SID(0x123);
C1FEN1bits.FLTEN0=1;
```

Discorso analogo può essere fatto per i buffer 2 e 3 associati ai messaggi extended al quale ho associato i filtri 1 e 2. Per entrambi decido di considerare tutti i bit del SID e dell'EID per effettuare l'AND logico. L'identificativo che permette al messaggio si "superare" il filtraggio è 12345678. Bisogna dire ancora che questo è un ID di prova a titolo puramente esplicativo. Come è stato trattato nel Capitolo 6 ogni modulo e

ogni componente avrà un ID preciso che segue un protocollo ben definito. Verrà quindi poi cambiato l'ID del filtraggio per adattarsi al protocollo. Riporto di seguito il codice del solo buffer 2 e del buffer 3:

```
C1FMSKSEL1bits.F1MSK=0b01;
C1RXM1EID=CAN_FILTERMASK2REG_EID0(0xFFFF);
C1RXM1SID=CAN_FILTERMASK2REG_EID1(0x1FFF);
C1RXF1EID=CAN_FILTERMASK2REG_EID0(0x5678);
C1RXF1SID=CAN_FILTERMASK2REG_EID1(0x1234);
C1FEN1bits.FLTEN1=1;

C1FMSKSEL1bits.F2MSK=0b01;
C1RXM2EID=CAN_FILTERMASK2REG_EID0(0xFFFF);
C1RXM2SID=CAN_FILTERMASK2REG_EID1(0x1FFF);
C1RXF2EID=CAN_FILTERMASK2REG_EID0(0x5678);
C1RXF2SID=CAN_FILTERMASK2REG_EID1(0x1234);
C1FEN1bits.FLTEN2=1;
```

Pongo infine il modulo in modalità "normal mode" e impongo che il buffer 0 sia un buffer di trasmissione mentre i buffer 1,2,3 siano di ricezione:

```
C1TR01CONbits.TXEN0=1;
C1TR01CONbits.TXEN1=0;
C1TR23CONbits.TXEN2=0;
C1TR23CONbits.TXEN3=0;
```

InitDMAECAN()

Il Direct Memory Access (DMA) ha la possibilità di gestire dati in ingresso e uscita sia del modulo CAN, ma anche di altri componenti (ad esempio il convertitore analogico/digitale, l'input capture, timer, uart e altri). E' necessario quindi settare correttamente i canali del DMA adibiti al modulo CAN. Ho deciso di usare in particolare il canale 0 per la trasmissione e il canale 2 per la ricezione:

```
DMA0CON=0x2020;
DMA0PAD=0x0442;
DMA2CON=0x0020;
DMA2PAD=0x0440;
```

Occorre fornire al DMA l'indirizzo nella memoria RAM nel quale memorizzare i dati provenienti dal modulo CAN e l'indirizzo dal quale prendere i dati da inviare al CAN-bus. Questo è l'indirizzo del buffer ecan1msgBuf, il quale è stato precedentemente inizializzato nel main. Vengono quindi abilitati i canali.

```
DMA0STA=__builtin_dmaoffset(&ecan1msgBuf);
DMA0CONbits.CHEN=1;
DMA2STA=__builtin_dmaoffset(&ecan1msgBuf);
DMA2CONbits.CHEN=1;
```

9.2.4 **Ciclo infinito**

All'interno del ciclo infinito while(1) è possibile sia inviare messaggi verso il pc sia ricevere messaggi dal pc. Analizziamo i due casi separatamente.

Ricezione da bus CAN e trasmissione SPI

Per poter accedere alla funzione di ricezione rxECAN è necessario che sia settato il buffer status ovvero che venga verificata la condizione:
`if(canRxMessage.buffer_status==CAN_BUF_FULL)`

E' necessario che venga ricevuto un messaggio, che il messaggio passi il filtraggio di accettazione e che quindi venga scatenata l'interrupt. Non appena viene settata la flag viene subito interrotto il ciclo principale della CPU e viene caricato l'indirizzo della prima istruzione dell' Interrupt Service Routine del CAN 1. Si andrà quindi alla funzione
`void __attribute__((interrupt,no_auto_psv))_C1Interrupt(void).`

Per prima cosa verrà controllato se l'interrupt è stato causato da una ricezione di un messaggio:

```
if(C1INTFbits.RBIF)
```

In caso positivo si controllerà quindi quale tra il buffer 1, il buffer 2 o il buffer 3 è stato riempito dal DMA. Viene impostato il canRxMessage.buffer_status come CAN_BUF_FULL e viene indicato il buffer:

```
if(C1RXFUL1bits.RXFUL1)
 {
            canRxMessage.buffer_status=CAN_BUF_FULL;
            canRxMessage.buffer=1;
}

else if(C1RXFUL1bits.RXFUL2)
{
      canRxMessage.buffer_status=CAN_BUF_FULL;
      canRxMessage.buffer=2;
}
else if(C1RXFUL1bits.RXFUL3)
{
      canRxMessage.buffer_status=CAN_BUF_FULL;
      canRxMessage.buffer=3;
}
```

Dopo aver resettato la flag si esce dall'interrupt service routine e si ritorna al main. Essendo verificata la condizione

```
if(canRxMessage.buffer_status==CAN_BUF_FULL)
```

viene lanciata subito la funzione rxECAN che necessita come parametro l'indirizzo del buffer dove si vuole memorizzare il messaggio (&canRxMessage).

Il programma è in grado di riconoscere se il messaggio è di tipo standard o di tipo estended. Questo lo fa prendendo il primo bit della word 0. Se questo è pari a zero allora il messaggio è di tipo standard altrimenti è di tipo esteso. Viene poi controllato se il frame è del tipo un RTR (request to remote) (vedi Capitolo 4) oppure è un

messaggio dati. Nel secondo caso vengono riempiti gli 8 byte di dati disponibili nel buffer con i dati presenti nel messaggio ricevuto. Viene infine resettata la flag di ricezione.

```
ide=ecan1msgBuf[message->buffer][0] & 0x0001;
if(ide==0){
        message->id=(ecan1msgBuf[message->buffer][0] & 0x1FFC) >> 2;

        message->frame_type=CAN_FRAME_STD;
        rtr=ecan1msgBuf[message->buffer][0] & 0x0002;
}
else{
        id=ecan1msgBuf[message->buffer][0] & 0x1FFC;
        message->id=id << 16;
        id=ecan1msgBuf[message->buffer][1] & 0x0FFF;
        message->id=message->id+(id << 6);
        id=(ecan1msgBuf[message->buffer][2] & 0xFC00) >> 10;
        message->id=message->id+id;
        message->frame_type=CAN_FRAME_EXT;
        rtr=ecan1msgBuf[message->buffer][2] & 0x0200;
}

if(rtr==1){
message->message_type=CAN_MSG_RTR;
}
else{
message->message_type=CAN_MSG_DATA;
message->data[0]=(unsigned char)ecan1msgBuf[message->buffer][3];
message->data[1]=(unsigned char)((ecan1msgBuf[message->buffer][3] &
0xFF00) >> 8);
message->data[2]=(unsigned char)ecan1msgBuf[message->buffer][4];
message->data[3]=(unsigned char)((ecan1msgBuf[message->buffer][4] &
0xFF00) >> 8);
message->data[4]=(unsigned char)ecan1msgBuf[message->buffer][5];
```

```
message->data[5]=(unsigned char)((ecan1msgBuf[message->buffer][5] &
0xFF00) >> 8);
message->data[6]=(unsigned char)ecan1msgBuf[message->buffer][6];
message->data[7]=(unsigned char)((ecan1msgBuf[message->buffer][6] &
0xFF00) >> 8);
}
clearRxFlags(message->buffer);
```

I dati sono ora memorizzati nel buffer canRxMessage. Occorre quindi decodificare i dati. Questo avviene nella funzione control(), la quale viene richiamata subito dopo la funzione rxECAN e vuole come variabile l'indirizzo del buffer.
Descrivo di seguito la funzione control() utilizzata per effettuare un semplice test di comunicazione CAN, di controllo del dato e di successiva comunicazione SPI. In questo test verrà inviato da pc, nel campo dati, i valori 0x41, 0x42, 0x43. In ognuno dei casi verrà illuminato un LED per assicurare la corretta ricezione del dato e verrà inviato il valore 10, 20, 30 al micro slave mediante la funzione write_SPI(). Da ricordare che questi valori sono del tutto casuali e non sottoposti al protocollo discusso nel Capitolo 6.

```
switch (message->data[0]) {
        case 0x41 :
        TRISBbits.TRISB12=0;
                PORTBbits.RB12=1;
                write_SPI(10);
        break;
        case 0x42 :
                TRISBbits.TRISB13=0;
                PORTBbits.RB13=1;
                write_SPI(20);
        break;
        case 0x43:
                TRISBbits.TRISB11=0;
                PORTBbits.RB11=1;
                write_SPI(30);
```

```
        break;
        default:
        break;
```

La funzione write_SPI pone subito a zero il pin collegato alla linea Slave Select in modo tale che lo slave venga attivato. Viene poi scritto nel buffer SPI1BUF il valore da mandare. Attendo poi che il dato venga inviato e infine pongo nuovamente alto lo slave select:

```
void write_SPI(int command)
{
        PORTBbits.RB8 = 0;
 SPI1BUF = command;
        while (SPI1STATbits.SPITBF);
        PORTBbits.RB8 = 1;
}
```

Ricezione SPI

I dati inviati mediante il protocollo SPI possono provenire: dal micro slave il quale invia informazione sullo stato dei propri sensori in base a una richiesta del master; dal sensore encoder dello stesso micro-master. Per ognuno dei due dispositivi viene utilizzato un modulo SPI diverso. In particolare per il micro slave viene usato il modulo SPI1 mentre per l'encoder il modulo SPI2. Di seguito riporto solo il codice del SPI1 in quanto quello del SPI2 è identico.

Come già osservato nel paragrafo 5.2, non appena un messaggio riempie il buffer SPIxBUF, viene settata la flag e scatenata l'interrupt. Occorre quindi impostare i comandi desiderati all'interno dell'interrupt service routine del modulo SPI. Per prima cosa viene resettata la flag. Si attende poi che il buffer di ricezione sia completamente riempito. Il messaggio presente nel buffer SPIxBUF viene poi memorizzato in un variabile buffer. Il prossimo comando è uno switch che decide quali istruzioni eseguire a seconda del messaggio ricevuto. Questo seguirà il protocollo descritto nel Capitolo 6. Per una semplice trattazione di seguito riporto il

codice usato inizialmente, nel quale ad ogni case dello switch si imponeva l'accensione di un LED della board "alzando" i pin collegati ad essi.

```
void __attribute__((interrupt,auto_psv))_SPI1Interrupt(void){
IFS0bits.SPI1IF = 0;

while (!SPI1STATbits.SPIRBF);
buffer=SPI1BUF;
SPI1BUF=0;

switch (buffer) {
        case 10 :
                TRISBbits.TRISB12=0;
          PORTBbits.RB12=1;
                break;
        case 20 :
                TRISBbits.TRISB13=0;
                PORTBbits.RB13=1;
                break;
        case 30 :
                TRISBbits.TRISB11=0;
                PORTBbits.RB11=1;
                break;
        default :
        break;
}
```

Lo stesso codice può essere usato per il modulo SPI2 avendo cura di sostituire il numero "2" con "1" nel testo del codice.

Trasmissione su bus CAN

La trasmissione avviene riempiendo il vettore canTxMessage nei vari campi quali: il tipo di dati inviati (dati, rtr,ecc.); il tipo di messaggio inviato (esteso,

standard); il buffer sul quale inviare il dato; l'ID; gli 8 byte dedicati nel caso si tratti di messaggio dati; DLC (data lenght code). Nel codice riportato di seguito ipotizzo si tratti di un messaggio dati, di tipo esteso, sul buffer 0 e riempio i byte dati con valori 12, 34, 56, 78, 11, 22, 33, 44, con ID pari a 123. Come nel caso della ricezione questi sono valori del tutto casuali utilizzati solo per testare il CAN.

```
canTxMessage.message_type=CAN_MSG_DATA;
canTxMessage.frame_type=CAN_FRAME_EXT;
canTxMessage.buffer=0;
canTxMessage.id=0x123;
canTxMessage.data[0]=0x12;
canTxMessage.data[1]=0x34;
canTxMessage.data[2]=0x56;
canTxMessage.data[3]=0x78;
canTxMessage.data[4]=0x11;
canTxMessage.data[5]=0x22;
canTxMessage.data[6]=0x33;
canTxMessage.data[7]=0x44;
canTxMessage.data_length=8;
```

Dopo aver quindi riempito il vettore viene lanciata la funzione sendECAN che accetta come parametro l'indirizzo del vettore.
La funzione ha il compito di riempire il buffer ecan1msgBuf dedicato alla trasmissione dei dati all'interno della memoria RAM del DMA. Poi sarà il DMA che si occuperà, senza dover interrompere la CPU, dell'invio del pacchetto al modulo CAN, il quale lo manderà poi al PC tramite il bus di comunicazione.

```
ecan1msgBuf[message->buffer][0]= message->id;
ecan1msgBuf[message->buffer][1]=word1;
ecan1msgBuf[message->buffer][2]=word2;
ecan1msgBuf[message->buffer][3]=((message->data[1] << 8) + message->data[0]);
```

```
ecan1msgBuf[message->buffer][4]=((message->data[3] << 8) + message->data[2]);
ecan1msgBuf[message->buffer][5]=((message->data[5] << 8) + message->data[4]);
ecan1msgBuf[message->buffer][6]=((message->data[7] << 8) + message->data[6]
```

9.3 **Firmware Slave per la comunicazione**

9.3.1 **Inizializzazione porte**

Specularmente al master, lo slave dovrà configurare 3 pin come input e uno come output. In particolare: il pin RB3 come clock input(SCK1); il pin RB8 come slave select input (SS1); il pin RP2 come data input (SDI1); il pin RB1 come data output. Come sappiamo anche se il messaggio viaggia dallo slave verso il master (per esempio una risposta ad una richiesta di controllo dell'encoder, da parte del master) il clock che controlla la comunicazione è sempre quello del master.

```
RPOR1bits.RP2R = 0b00111;
RPINR20bits.SCK1R = 0b00011;
RPINR21bits.SS1R = 0b01000;
RPINR20bits.SDI1R = 0b0010;
TRISBbits.TRISB2 = 1;
TRISBbits.TRISB3 = 1;
TRISBbits.TRISB8 = 1;
TRISBbits.TRISB9 = 0;
```

9.3.2 **Inizializzazione SPI**

La funzione di inizializzazione SPI è identica a quella del master, vista nel precedente capitolo, con alcune modifiche. Per prima cosa è necessario disabilitare la modalità master. Occorre anche impostare il pin di slave select in modo che esso sia

in funzione slave. Questo significa che quando il livello di tensione al pin passa da un valore alto a zero, il microcontrollore dovrà scatenare l'interrupt dell'SPI e abilitare il pin del clock e del data input. Alla fine della funzione Init_SPI abilito il modulo SPI, resetto la flag dell'interrupt, in modo tale che sia possibile scatenare l'interrupt, e abilito quest'ultimo:

```
SPI1CON1bits.MSTEN = 0;
SPI1CON1bits.SSEN = 1;
SPI1STATbits.SPIEN = 1;
IFS0bits.SPI1IF = 0;
IEC0bits.SPI1IE = 1;
```

9.3.3 **Ciclo infinito**

Nel while(1) si attende soltanto lo scatenarsi dell'interrupt dell'SPI, ovvero si attende l'arrivo di un nuovo messaggio.
Entrati nell'interrupt service routine, la prima cosa da fare è resettare la flag. Attendo poi che tutto il messaggio venga ricevuto, ovvero che il buffer SPI1BUF venga riempito completamente, e assegno subito questo valore ad una variabile "buffer". Ho deciso di utilizzare un'altra variabile poiché il buffer di ricezione viene riempito ogni volta che il pin SS va a zero. Nel caso in cui le operazioni da effettuare su SPI1BUF fossero troppo lente e un nuovo dato è pronto per essere ricevuto, il dato precedente andrà perso e il nuovo dato sovrascriverà il vecchio con perdita ovvia di informazioni.

```
void __attribute__((interrupt,auto_psv))_SPI1Interrupt(void){
IFS0bits.SPI1IF = 0;
while (!SPI1STATbits.SPIRBF);
buffer=SPI1BUF;
```

A seconda del valore presente nella variabile "buffer", mediante uno switch, decido quale comando effettuare. Di seguito riporto solo il codice usato inizialmente per

controllare la comunicazione SPI tra master e slave. Lo switch controlla il valore presente in "buffer". Se è pari a 10 allora porrò a livello alto il pin RB12, se 20 il pin RB13, se 30 il pin RB11. Questo è solo un semplice codice per illustrare il sistema di funzionamento. Nel Capitolo 6 è stato illustrato il protocollo completo.

```
switch (buffer) {
        case 10 :
         TRISBbits.TRISB12=0; // accendo led d13 per vedere se funziona
         PORTBbits.RB12=1;
        break;
        case 20 :
                TRISBbits.TRISB13=0; // accendo led d12 per vedere se funziona
                PORTBbits.RB13=1;
        break;
        case 30 :
                TRISBbits.TRISB11=0; // accendo led d14 per vedere se funziona
                PORTBbits.RB11=1;
        break;
        default :
        break;
```

9.4 **Firmware per il controllo motori**

Per il controllo motori non ho fatto distinzione tra firmware master e firmware slave in quanto entrambi seguono gli stessi comandi. L'unica differenza è legata al modo in cui arriva al componente il valore della velocità desiderata. Al master arriverà dal pc mediante CAN bus. La velocità desiderata del motore controllato dallo slave verrà inviata dal PC al master tramite CAN, verrà decodificato il messaggio e poi inviato il valore della velocità mediante SPI allo slave.
Come già discusso nel Capitolo 8 per poter generare un segnale PWM è necessario impostare il buffer P1TPER. Esso viene settato utilizzando una formula dove compaiono delle costanti definite all'inizio del codice. Queste sono la frequenza del clock del micro (Fcy=40 MHz) e la frequenza del PWM (FPWM=20KHz):

```
P1TPER = ((FCY/FPWM)/2 - 1)
```

Da questa formula otteniamo il valore 999 che rappresenta il 50% del Duty Cycle. Ricordando il principio di funzionamento del modulo PWM del micro, verrà aumentato il valore della variabile P1TMR fino a quando non raggiungerà il valore P1TPER. Successivamente verrà fatto decrescere fino a zero. Non appena verrà raggiunto lo zero, verrà scatenato l'interrupt.

Setto il prescale e il postscale al valore 1:1 in modo tale che la frequenza con la quale cresce P1TMR è la stessa del clock interno. Setto il PTMOD in modo da avere l'interrupt quando il valore di P1TMR torna di nuovo a zero. Imposto a valore 4 la priorità dell'interrupt del PWM (il valore massimo è 7), resetto la flag dell'interrput, abilito l'inerrupt e abilito il modulo PWM.

```
P1TPER = ((FCY/FPWM)/2 - 1);
P1TCONbits.PTOPS = 0;
P1TCONbits.PTCKPS = 0;
P1TCONbits.PTMOD = 2;
IPC14bits.PWM1IP = 4;
IFS3bits.PWM1IF=0;
IEC3bits.PWM1IE=1;
P1TCONbits.PTEN = 1;
```

Come descritto ampiamente nel Capitolo 8 per permettere la rotazione completa dell'albero motore è necessario che vengano attivati alternativamente i sei segnali in uscita. Per far ciò basta modificare ad ogni ingresso nell'interrupt service routine il valore del registro P1OVDCON secondo la Tabella 9.1.

State	PxOVDCON<15:8>	PxOVDCON<7:0>
1	00000000b	00100100b
2	00000000b	00100001b
3	00000000b	00001001b
4	00000000b	00011000b
5	00000000b	00010010b
6	00000000b	00000110b

Tabella 9.1 Valori del registro P1OVDCON per ogni stato della rotazione

Ad ogni valore del registro corrisponde un determinato stato illustrato in Figura 9.1.

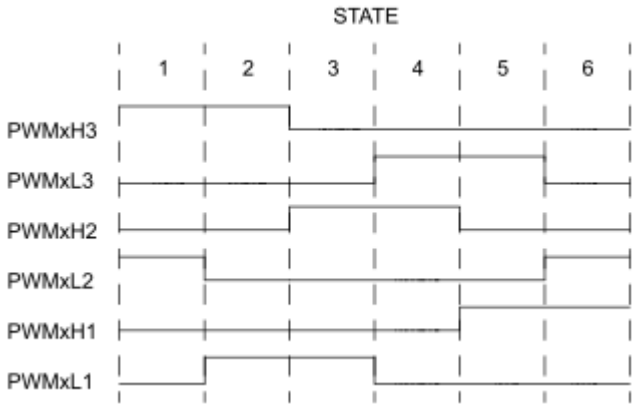


Figura 9.1 Output dei pin PWM a seconda dello stato.

La modifica del registro P1OVDCON verrà fatta andando considerare tutti e sei i valori del vettore PWM_STATE. Questo vettore sarà impostato nella parte iniziale del codice dove vengono definite le costanti e le variabili globali. Esso è costituito da tutti i valore presenti nella Tabella 9.1.

Successivamente controllo che il valore di velocità richiesto non sia troppo basso o troppo elevato. Faccio cioè un confronto con delle costanti calcolate a priori e settate che corrispondono ad un minimo valore di duty cycle sotto il quale non conviene andare e un valore massimo imposto dalla formula discussa sopra (ovvero il doppio di P1TPER). Questi valori sono MIN_DUTY_CYCLE = 73 e MAX_DUTY_CYCLE =1998. Nel caso in cui, per un errore di comunicazione, il duty cycle sia troppo basso o troppo alto, questo verrà impostato pari ai valori limite.

Nel caso invece in cui il valore sia corretto, questo verrà assegnato ai tre buffer che impostano i duty cycle dei tre segnali PWM. Verrà infine resettata la flag.

```
void __attribute__((__interrupt__,auto_psv)) _MPWM1Interrupt (void)
{
        if (++ADCCommState>6) {
                ADCCommState=1;
}       P1OVDCON=PWM_STATE[ADCCommState];
```

```
        if (u16DesiredPWMDutyCycle < MIN_DUTY_CYCLE)
                {u16DesiredPWMDutyCycle = MIN_DUTY_CYCLE;}
        if (u16DesiredPWMDutyCycle > MAX_DUTY_CYCLE)
                {u16DesiredPWMDutyCycle = MAX_DUTY_CYCLE;}

        P1DC1 = u16DesiredPWMDutyCycle;
        P1DC2 = u16DesiredPWMDutyCycle;
        P1DC3 = u16DesiredPWMDutyCycle;
        IFS3bits.PWM1IF = 0;
}
```

Capitolo 10
TEST

In questo capitolo verranno illustrati i test svolti in laboratorio durante il tirocinio. Sono state svolti tre test. Il primo test ha riguardato la comunicazione CAN, il secondo entrambe le comunicazioni insieme (CAN+SPI). Infine sono stati provati i motori.

10.1 **Strumenti**

Per tutti i test svolti è stata utilizzata la evaluation board dsPICDEM MCLV mostrata in Figura 10.1. Essa presenta le seguenti caratteristiche principali: è dotata un connettore per la comunicazione CAN, uno stadio per il condizionamento (driver e mosfet) del segnale PWM da inviare al motore, connettori per la connessione con motore trifase, connessioni feedback per poter effettuare la BLDC, connettore per la programmazione e debugging, porte di input e output. Per la programmazione dei microcontrollori è stato utilizzato il programma MPLABIDE v8.50. Per effettuare il test di comunicazione SPI tra i due microcontrollori è stata utilizzata una scheda millefori sulla quale è stato saldato il secondo dsPIC. Per l'invio dei dati da PC è stato utilizzato l'interfaccia LABVIEW descritta nel paragrafo 4.4.2.

Figura 10.1 dsPICDEM MCLV

10.2 **Test comunicazione CAN**

Il primo test consiste nell'inviare da pc un messaggio di tipo extended con ID 12345678 e campo dati 11 22 33 44 55 66 77 88. L'ID è quello del microcontrollore montato sulla scheda, e il dato corrisponde al comando di accenzione di un LED presente sulla board. Non appena viene cliccato il pulsante Send message (vedi paragrafo 4.4.2), sul bus viene inviata l'onda mostrata in Figura 10.2. E'stato utilizzato un particolare oscilloscopio in grado di rilevare e decodificare il segnale CAN.

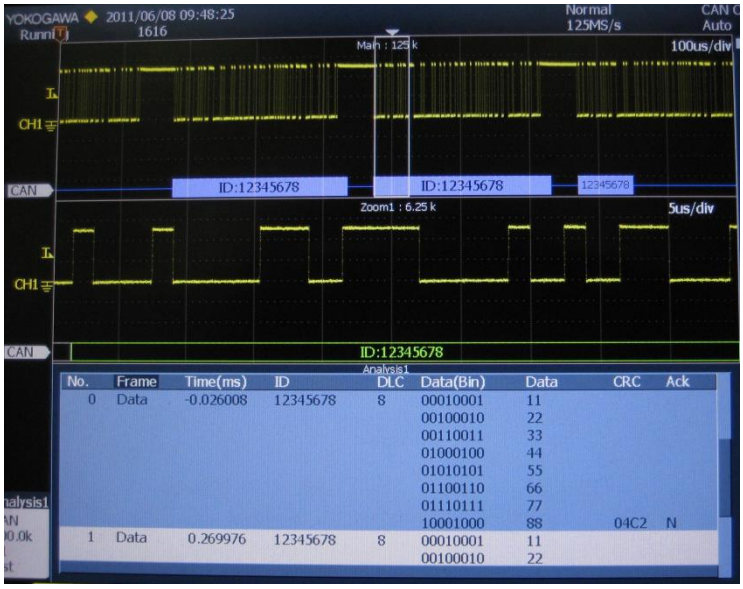


Figura 10.2 Onda segnale CAN

Oltre quindi a un risultato positivo visivo dato dall'accenzione del LED sulla board, è stato possibile osservare che effettivamente il dato inviato è corretto in ogni suo campo.

Il secondo test sulla comunicazione CAN riguarda invece la trasmissione da micro verso PC. Nel firmware del micro è stato comandato di inviare un messaggio con ID pari a 0x123(ID del programma Labview) e campo dati pari a 12 34 56 78 11 22 33 44. In Figura 10.3 è mostrato il risultato.

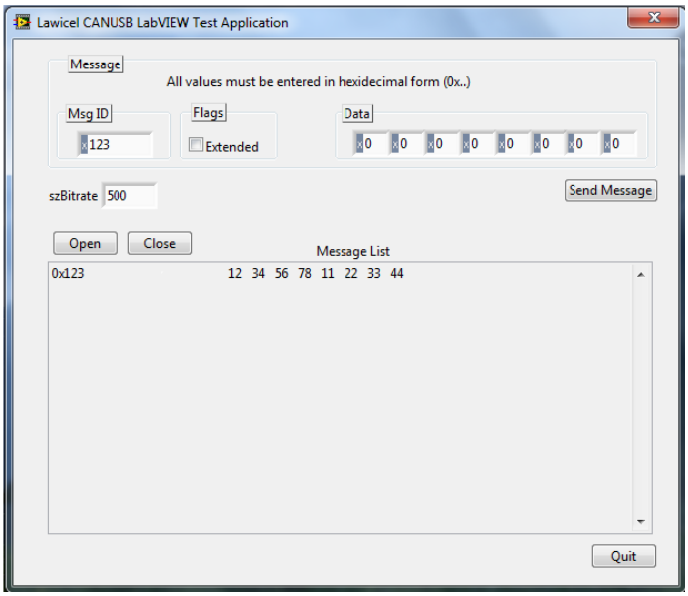


Figura 10.3 Messaggio inviato dal micro verso il PC

10.3 **Test comunicazione CAN+SPI**

In questo test il micro-master riceve il dato 0x41 da pc (mediante CANbus), a questo dato corrisponde un determinato comando, ovvero inviare al micro slave il dato 0x20 sul bus SPI. Per lo slave a questo valore è associato un determinato comando: poni alto pin RB12. Il risultato è osservabile andando a misurare la tensione sul pin RB12 mediante oscilloscopio. In Figura 10.4 è possibile osservare le forme d'onda del segnale dati, segnale di clock e slave select inviati dal micro master al micro slave.

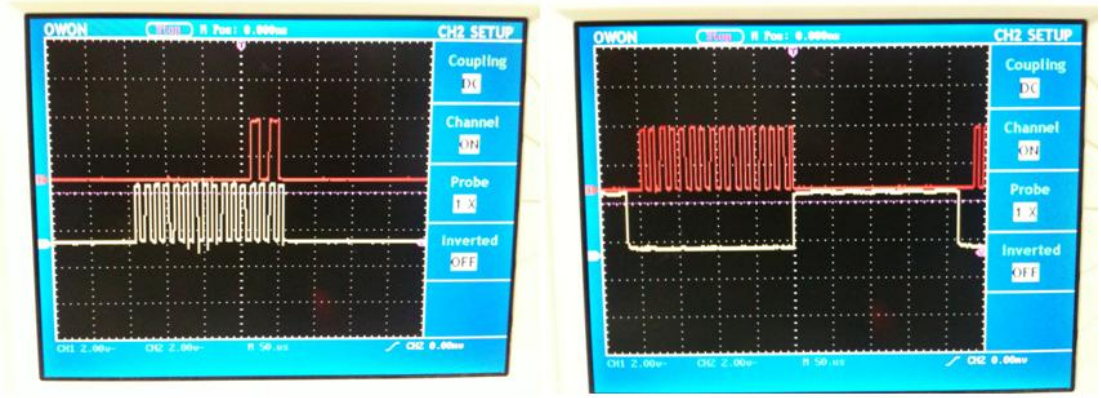


Figura 10.4 A sinistra: in alto segnale dati, in basso clock.
A destra: in alto segnale clock, in basso Slave Select

10.4 **Test controllo motori**

Durante questa prova è stato valutato il corretto funzionamento del modulo PWM del microcontrollore. È stato impostato il firmware del micro in modo tale da inviare tre segnali PWM sfasati l'uno dall'altro, ai tre avvolgimenti del motore. La buona riuscita del test è stata valutata osservando la rotazione dell'albero motore. In Figura 10.5 sono mostrati i segnali PWM all'uscita del micro. E' stato necessario utilizzare due foto in quanto l'oscilloscopio utilizzato presenta solo due uscite per le sonde. E' possibile comunque notare lo sfasamento dei segnali PWM.

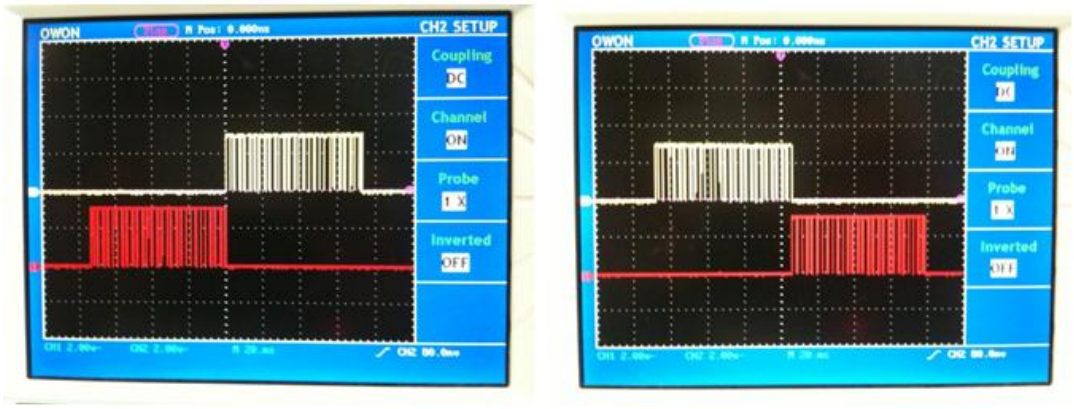


Figura 10.5 Segnali PWM all'uscita del micro

In Figura 10.6 rappresento invece i segnali PWM amplificati in uscita dai driver.

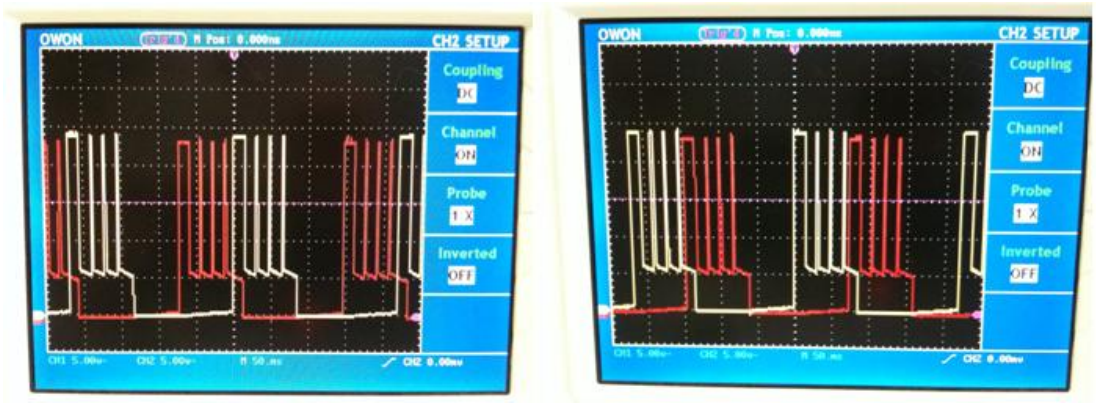


Figura 10.6 Segnali PWM in uscita dai driver.

Capitolo 11
DISCUSSIONE E CONCLUSIONI

La tesi ha fornito uno studio del sistema di comunicazione, del controllo e della componentistica del robot snake. E' stato fornito un ampio panorama sui dispositivi biomedici per endoscopia attualmente in uso nel mondo e sulla modalità di chirurgia endoscopica transluminare attraverso orifizi naturali (NOTES). È stato descritto uno stato dell'arte tecnologico per inquadrare il nostro robot in una gruppo ben preciso e poi di uno stato dell'arte brevettuale in modo da illustrare somiglianze differenze con i brevetti esistenti.

La scelta dei componenti è stato uno scoglio abbastanza arduo in quanto non è stato sempre facile trovare dei componenti che soddisfino le specifiche di progetto. Per ogni componente è stato necessario valutare vari campi: dimensione, alimentazione, potenza, modalità di comunicazione, ecc...

È stato fatto uno studio approfondito dei due protocolli di comunicazione scelti per l'applicazione: CAN, SPI. Il primo è stato utilizzato per la comunicazione tra l'esterno e l'interno del robot. In particolare tra il PC e i microcontrollori. Sono stati esposti i principali vantaggi che il protocollo possiede e che hanno portato al suo impiego: tempi di risposta rigidi, semplicità e flessibilità del cablaggio, alta immunità ai disturbi, elevata affidabilità, confinamento degli errori, maturità dello standard. Sono stati analizzati tutti i tipi di messaggio che il protocollo propone. Sono stati illustrati anche gli strumenti utilizzati per la comunicazione CAN del

microcontrollore e del PC. Questi strumenti sono serviti per effettuare dei test di comunicazione.

Il protocollo di comunicazione SPI invece è stato utilizzato per la comunicazione tra micro-master e micro-slave e tra micro ed encoder.

Vista la numerosità dei componenti e la complessità dell'intero sistema è stato necessario definire un protocollo per la gestione dei comandi ai vari componenti. Questo sfrutta il protocollo CAN e in particolare il frame di trasmissione dei dati.

I componenti scelti verranno poi saldati su una schedina, la quale verrà poi inserita all'interno del modulo. La tesi ha fornito una descrizione dello schematico di tale board. Poiché ancora non è stato possibile valutare le dimensioni effettive dello spazio all'interno del modulo, ancora non è stato possibile realizzare il PCB. Sono stati anche realizzati in Eagle i package di tutti i componenti non presenti nella libreria del software. E' stato necessario anche valutare le modalità di controllo dei motori e dei sensori (encoder e sensori di forza). È stato commentato, in modo da renderlo quanto più semplice possibile, il firmware presente su ognuno dei due micro.

Sono stati effettuati anche dei test: un test per il controllo dei motori, un test per la comunicazione CAN, uno per la comunicazione SPI e infine un test che unisce tutto il sistema di comunicazione (CAN+SPI).

Gli sviluppi futuri riguardano soprattutto la meccanica del robot. Occorrerà realizzare, con un software CAD come Solidworks, uno disegno del modulo del robot. Dopo aver fatto ciò sarà possibile realizzare una PCB da inserire all'interno. Prima di far ciò però, converrebbe costruire una PCB anche di dimensioni maggiori, saldare sopra tutti i componenti scelti e verificare il corretto funzionamento del robot: controllo motori, comunicazione, gestione delle informazioni dei sensori.

Occorrerà calibrare i sensori di forza. Occorrerà trovare il valore che rappresenta la massima forza applicabile sui tessuti. Un possibile ulteriore impiego del sensore di forza riguarda la retroazione della forza. Si potrebbe pensare di utilizzare un interfaccia aptica. Un'interfaccia aptica è un dispositivo che permette di manovrare un robot, reale o virtuale, e di riceverne delle sensazioni tattili in risposta (retroazione o feedback). Un esempio potrebbe essere un joystick con ritorno di forza (force

feedback). Nel caso quindi in cui il chirurgo stia imprimendo una forza troppo elevata con la sonda sul tessuto, questo potrebbe tradursi in una irrigidimento dell'interfaccia aptica (joystick che comanda la sonda). In questo modo si da una retroazione di forza al chirurgo.

Un altro aspetto che bisognerà sicuramente trattare è la cinematica del robot. Sarà necessario effettuare uno studio approfondito sulla cinematica diretta e inversa del robot, nonché dei sui punti di singolarità.

Inoltre non sono ancora stati definiti gli strumenti chirurgici per la manipolazione dei tessuti. Un'idea, che ancora non è stata sufficientemente elaborata, prevede l'utilizzo di due strumenti, uno a destra e uno a sinistra della telecamera. Essi saranno saldamente attaccati al corpo del robot durante l'inserimento della sonda e si schiuderanno solo nel momento in cui è stato raggiunto il sito operatorio. L'obbiettivo è di imitare gli endoscopi già esistenti che prevedono due strumenti chirurgici.

Riferimenti bibliografici

[1] Paggi Claus C., Bonin E., Torres M., Ligocki Campos A., Cury A., Uili Coelho J., "Liver and peritoneal biopsy by laparoscopy or notes in pigs: comparison of operative parameters and postoperative evolution", 2011.

[2] Wagh M.S., Thompson C.C., "Surgery insight: natural orifice transluminal endoscopic surgery-an analysis of work to date"2007.

[3] Zhang X., Yang Y., Sun G., Guo M., "Natural orifice translumenal endoscopic surgery (NOTES)：current status and challenges", 2010.

[4] Geoffrey B., Timothy A., Jeffrey C., Edward C., Ralph C., Mihir D., et al., "Nomenclature of Natural Orifice Translumenal Endoscopic Surgery (NOTES™) and Laparoendoscopic Single-Site Surgery (LESS) Procedures in Urology", 2008.

[5] Wagh M., Merrifield B., Thompson C., "Survival studies after endoscopic transgastric oophorectomy and tubectomy in a porcine model, Gastrointest Endosc 2006, pag 473-478.

[6] Gumbs A., Fowler D., Milone L., Evanko J., Ude A., Stevens P., et al., "Transvaginal natural orifice translumenal endoscopic surgery cholecystectomy: early evolution of the technique", 2009, pag: 908-912.

[7] http://www.karlstorz.de/cps/rde/xchg/SID-83BCB9C4-468AAC48/karlstorz-

en/hs.xsl/8818.htm.

[8] http://www.usgimedical.com/eos/usgi_transport_description.pdf.

[9] Vahe Karimyana et al., " Navigation systems and platforms in natural orifice translumenal endoscopic surgery (NOTES)".

[10] Cerveri P, Forgione A., Marchente M. "Self propelled Instrument Carrier for surgery through natural orifices- Relevant technologies and trends", TEC-MMG/2009/8.

[11] Kai Xu, Roger E. Goldman, Jienan Ding, Peter K. Allen, Dennis L. Fowler and Nabil Simaan, "System Design of an Insertable Robotic Effector Platform for Single Port Access (SPA) Surgery".

[12] Amy C. Lehman ,Jason Dumpert, Nathan A. Wood , Lee Redden , Abigail Q. Visty, Shane Farritor ,Brandon Varnell ,Dmitry Oleynikov, "Natural orifice cholecystectomy using a miniature robot".

[13] Bhavin C. Shah, Shelby L. Buettner, Amy C. Lehman, ShaneM. Farritor, Dmitry Oleynikov, "Miniature InVivo Robotics and Novel Robotic Surgical Platforms".

[14] Chirikjian, G.S. & Burdick, J.W, "Kinematics of hyper-redundant robot locomotion with applications to grasping", Proc. IEEE International Conference on Robotics and Automation, pp. 720-725, 1991.

[15] Poi G., Scarabeo C. & Allotta B, "Traveling wave locomotion hyper-redundant mobile robot", Proc. IEEE International Conference on Robotics and Automation, Vol. 1, pp. 418-423, 1998.

[16] http://it.edaboard.com/topic-4590520.0.html

[17] La Mantia D., “Linea CAN”, 2004.

[18] Giannetti G., “PWM Tutorial”, 2004.

[19] http://it.wikipedia.org/wiki/Motore_in_corrente_continua

[20] http://it.wikipedia.org/wiki/Motore_brushless

[21] Ranzato F., “Encoder Ottico, Principio di Funzionamento e Applicazioni”, 2009.

[22] http://it.wikipedia.org/wiki/Ricetrasmettitore

[23] Galanti A., “BOSCH‘s CONTROLLER AREA NETWORK”, 2005.

[24] dsPIC33F Family Reference Manual. Section 21. Disponibile su : http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532302

[25] Application note AN1249 : “ECAN™ Operation with DMA on dsPIC33F and PIC24H Devices”. Disponibile su : http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532302

[26] dsPIC33F Family Reference Manual. Section 18. Disponibile su: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532302

[27] Fulchiero R. “Serial Communication using the dsPIC30F SPI Module”, 2005.

[28] Torres D., “Sensorless BLDC Control with Back-EMF Filtering Using a Majority Function”, 2008.

[29] AIFP® Product Datasheet. Disponibile su:
http://microstrain.s3.amazonaws.com/pdf/AIFP_flier_rev2.pdf

[30] AIFP® User Manual. Disponibile su:
http://microstrain.s3.amazonaws.com/pdf/manuals/AIFP_usermanual.pdf

[31] Di Martino A., "Applicazioni di Interfacce aptiche e Realtà Aumentata in ambiti di manutenzione industriale, formazione, medicina e beni culturali", 2010.

APPENDICE A
FIRMWARE COMPLETO

Main.c :

```
#include "p33FJ128MC802.h"
#if defined(__dsPIC33F__)
#include "p33fxxxx.h"
#elif defined(__PIC24H__)
#include "p24hxxxx.h"
#endif
#include "ecan.h"
#include "delay.h"
#include "spi.h"
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF);
_FWDT(FWDTEN_OFF);
_FICD(JTAGEN_OFF & ICS_PGD3);

/******************************************************************************
Function Prototypes
*******************/
void oscConfig(void);
void initport(void);
void delay(void);
void write_SPI(int command);
void Init_SPI(void);
void DelayNmSec(unsigned int N);
void delay_SPI(void);
/******************************************************************************
```

```
Globals
******************/
mID canTxMessage;
mID canRxMessage;
/* Define ECAN Message Buffers */
ECAN1MSGBUF ecan1msgBuf __attribute__((space(dma),aligned(ECAN1_MSG_BUF_LENGTH*16)));
#define FCY  40000000
#defineMILLISECFCY/40000

/*****************************************************************************
main() function
******************/
int main ( void )
{
/* Configure Oscillator Clock Source */
oscConfig();
/* initialise other modules */
initport();
initECAN();
initDMAECAN();
Init_SPI();
// Enable ECAN1 Interrupt
IEC2bits.C1IE=1;
// enable Transmit interrupt
C1INTEbits.TBIE=1;
// Enable Receive interrupt
C1INTEbits.RBIE=1;
//configure and send a message
canTxMessage.message_type=CAN_MSG_DATA;
//canTxMessage.message_type=CAN_MSG_RTR;
canTxMessage.frame_type=CAN_FRAME_EXT;
//canTxMessage.frame_type=CAN_FRAME_STD;
canTxMessage.buffer=0;
canTxMessage.id=0x123;
canTxMessage.data[0]=0x12;
```

```
canTxMessage.data[1]=0x34;
canTxMessage.data[2]=0x56;
canTxMessage.data[3]=0x78;
canTxMessage.data[4]=0x11;
canTxMessage.data[5]=0x22;
canTxMessage.data[6]=0x33;
canTxMessage.data[7]=0x44;
canTxMessage.data_length=8;


// Delay for a second
Delay(Delay_1S_Cnt);
/*// send a CAN message
sendECAN(&canTxMessage);
*/
while(1)
{
/* check to see when a message is received and move the message
into RAM and parse the message */
if(canRxMessage.buffer_status==CAN_BUF_FULL)
{
rxECAN(&canRxMessage);
control(&canRxMessage);
// reset the flag when done
canRxMessage.buffer_status=CAN_BUF_EMPTY;
}
else
{// delay for one second
Delay(1);
// send another message
// canTxMessage.id++;
sendECAN(&canTxMessage);
}
```

```
void oscConfig(void){
// Configure Oscillator to operate the device at 40 Mhz
// Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
PLLFBD=41;                                                    // M=43
CLKDIVbits.PLLPOST=0;                                         // N1=2
CLKDIVbits.PLLPRE=0;                                          // N2=2
OSCTUN=3;// Tune FRC oscillator, if FRC is used
// Initiate Clock Switch to Internal FRC with PLL (NOSC = 0b001)
__builtin_write_OSCCONH(0x01);
__builtin_write_OSCCONL(0x01);
// Wait for Clock switch to occur
while (OSCCONbits.COSC != 0b001);
// Wait for PLL to lock
while(OSCCONbits.LOCK != 1) {};
}
void initport(void) {
PORTB = 0x0000;
TRISBbits.TRISB7= 0x1;  // imposto porta RP7 come ingresso
TRISBbits.TRISB9= 0x0;  // imposto porta RP9 come uscita
RPOR4bits.RP9R = 0x10;   // imposto porta RP9 come C1TX per la trasmissione attraverso CAN
RPINR26bits.C1RXR= 0b00111;  // imposto porta RP7 come C1RX per la ricezione attraverso CAN
//TRISBbits.TRISB1 = 0;
TRISBbits.TRISB2 = 0;
TRISBbits.TRISB3 = 0;
TRISBbits.TRISB8 = 0;
TRISBbits.TRISB4 = 1;
RPOR0bits.RP1R = 0b00111;   // RP1 tied to SPI1 Data Output
//RPOR1bits.RP2R = 0b00111;   // RP2 tied to SPI1 Data Output
RPOR1bits.RP3R = 0b01000; // RP3 tied to SPI1 Clock Output
RPOR4bits.RP8R = 0b01001;  // RP8 tied to SPI1 Slave Select Output
RPINR20bits.SDI1R = 0b0100; // Assign SPI1 Data Input (SDI1) to RP4 pin
AD1PCFGLbits.PCFG3 = 1;// set the RB1 pin to Digital Mode (all others to analog)
//AD1PCFGLbits.PCFG4 = 1;// set the RB2 pin to Digital Mode (all others to analog)
AD1PCFGLbits.PCFG5 = 1;// set the RB3 pin to Digital Mode (all others to analog)
/*PWMs are outputs*/
```

```
LATBbits.LATB9 = 1;//PWM_EN, FIRE_EN in the MC Pictail Duaghter Board
TRISBbits.TRISB9 = 0;
LATBbits.LATB10 = 0;//PWM1H3
TRISBbits.TRISB10 = 0;
LATBbits.LATB11 = 0;//PWM1L3
TRISBbits.TRISB11 = 0;
LATBbits.LATB12 = 0;//PWM1H2
TRISBbits.TRISB12 = 0;
LATBbits.LATB13 = 0;//PWM1L2
TRISBbits.TRISB13 = 0;
LATBbits.LATB14 = 0;//PWM1H1
TRISBbits.TRISB14 = 0;
LATBbits.LATB15 = 0;//PWM1L1
TRISBbits.TRISB15 = 0;
INTCON1bits.NSTDIS = 0; // Enabling nested interrupts
InitMCPWM();
LATBbits.LATB9 = 0;//PWM_EN, FIRE_EN in the MC Pictail Duaghter Board
DesiredSpeed = 0;
ActualSpeed = 0;
SpeedError = 0;
SpeedIntegral = 0;
PILoopControllerOutput = 0;
timer3value = 0;
 timer3avg = 0;
}
void __attribute__((interrupt,no_auto_psv))_C1Interrupt(void)
{
/* check to see if the interrupt is caused by receive */
  if(C1INTFbits.RBIF)
  {  /* check to see if buffer 1 is full */
  if(C1RXFUL1bits.RXFUL1)
  {
/* set the buffer full flag and the buffer received flag */
canRxMessage.buffer_status=CAN_BUF_FULL;
canRxMessage.buffer=1;
```

```
}
/* check to see if buffer 2 is full */
else
if(C1RXFUL1bits.RXFUL2)
{/* set the buffer full flag and the buffer received flag */
canRxMessage.buffer_status=CAN_BUF_FULL;
canRxMessage.buffer=2;}
/* check to see if buffer 3 is full */
else
if(C1RXFUL1bits.RXFUL3)
{/* set the buffer full flag and the buffer received flag */
canRxMessage.buffer_status=CAN_BUF_FULL;
canRxMessage.buffer=3;}
else;
/* clear flag */
C1INTFbits.RBIF = 0;
}
else if(C1INTFbits.TBIF)
 {  /* clear flag */
C1INTFbits.TBIF = 0;
}
else;
/* clear interrupt flag */
IFS2bits.C1IF=0;
}

void __attribute__((__interrupt__,auto_psv)) _MPWM1Interrupt (void)
{
IFS3bits.PWM1IF = 0;
Commutate();
if (u16DesiredPWMDutyCycle < MIN_DUTY_CYCLE)
u16DesiredPWMDutyCycle = MIN_DUTY_CYCLE;
if (u16DesiredPWMDutyCycle > MAX_DUTY_CYCLE)
u16DesiredPWMDutyCycle = MAX_DUTY_CYCLE;
P1DC1 = u16DesiredPWMDutyCycle;
```

```
P1DC2 = u16DesiredPWMDutyCycle;
P1DC3 = u16DesiredPWMDutyCycle;



}

void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
 IFS0bits.DMA0IF = 0;    // Clear the DMA0 Interrupt Flag;
}
void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
{
 IFS0bits.DMA1IF = 0;    // Clear the DMA1 Interrupt Flag;
}
void __attribute__((interrupt, no_auto_psv)) _DMA2Interrupt(void)
{
 IFS1bits.DMA2IF = 0;    // Clear the DMA2 Interrupt Flag;
}

void __attribute__((interrupt, no_auto_psv)) _DMA3Interrupt(void)
{
 IFS2bits.DMA3IF = 0;    // Clear the DMA3 Interrupt Flag;
}
void control (mID *message) {
if(message->data[0]== 'A') { //cioé se è 0x41
TRISBbits.TRISB12=0;  // accendo led d13 per vedere se funziona
PORTBbits.RB12=1;
SPI1STATbits.SPIEN = 1; //abilita SPI
write_SPI(10);
//delay_SPI();
}
if(message->data[0]== 'B') { //cioé se è 0x42
TRISBbits.TRISB13=0;  // accendo led d13 per vedere se funziona
PORTBbits.RB13=1;
SPI1STATbits.SPIEN = 1; //abilita SPI
```

```
write_SPI(20);
 delay_SPI();
}
if(message->data[0]== 'C') { //cioé se è 0x43
TRISBbits.TRISB11=0;  // accendo led d13 per vedere se funziona
PORTBbits.RB11=1;
SPI1STATbits.SPIEN = 1; //abilita SPI
write_SPI(30);
delay_SPI();
}
//delay_SPI();
 //SPI1STATbits.SPIEN = 0; //disabilita SPI
}

void DelayNmSec(unsigned int N)
{
unsigned int j;
while(N--){
 for(j=0;j < MILLISEC;j++){};
}
}

void InitMCPWM(void)
{P1TPER = ((FCY/FPWM)/2 - 1); // formula per calcolare P1TPER (dal reference manual)
//FCY  29491200...FRC w/PLL x16
//FPWM 20KHz PWM Freq
// MAX_DUTY_CYCLE = 1469
// 50% duty cycle = 734
P1TCONbits.PTSIDL = 1;  // PWM time base halted in CPU IDLE mode
P1TCONbits.PTOPS = 0; // PWM time base 1:1 postscale
P1TCONbits.PTCKPS = 0;  // PWM time base 1:1 prescale
P1TCONbits.PTMOD = 2;// Center Aligned with single interrupt mode per PWM period (comincia a
contare, arriva a P1TPER,scende e appena arriva a zero -> interrupt)
PWM1CON1 = 0x0700;// disable PWMs
P1OVDCON = 0x0000;// allow control using OVD
```

```
P1SECMPbits.SEVTDIR = 0; // trgger ADC when PWM counter is in upwards dir
//....Tad=84.77, Tpwm=67.816
P1SECMPbits.SEVTCMP = 0;// generates a trigger event for the ADC
 PWM1CON2 = 0x0000;// 1:1 postscale values
IPC14bits.PWM1IP = 4;// PWM Interrupt Priority 4
IFS3bits.PWM1IF=0;// clear
IEC3bits.PWM1IE=1;
P1TCONbits.PTEN = 1;}


void Commutate(void)
{//Commutation
if (++ADCCommState>6)
ADCCommState=1;
P1OVDCON=PWM_STATE[ADCCommState];
}

void SpeedPILoopController(void)
{//PWM duty cycle = potenziometer value *2
u16DesiredPWMDutyCycle = DesiredSpeed<<1;
u16CurrentPWMDutyCycle = u16DesiredPWMDutyCycle;
if(u16CurrentPWMDutyCycle != u16DesiredPWMDutyCycle)
{if(u16CurrentPWMDutyCycle < u16DesiredPWMDutyCycle)
u16CurrentPWMDutyCycle++;
if(u16CurrentPWMDutyCycle > u16DesiredPWMDutyCycle)
u16CurrentPWMDutyCycle--;
}
if (u16CurrentPWMDutyCycle < MIN_DUTY_CYCLE)
u16CurrentPWMDutyCycle = MIN_DUTY_CYCLE;
if (u16CurrentPWMDutyCycle > MAX_DUTY_CYCLE)
u16CurrentPWMDutyCycle = MAX_DUTY_CYCLE;
P1DC1 = u16CurrentPWMDutyCycle; // decido quanto tempo deve stare alzato
P1DC2 = u16CurrentPWMDutyCycle;
P1DC3 = u16CurrentPWMDutyCycle;
}
```