

POLITECNICO DI BARI



DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

RELAZIONE DI PROGETTO

in

EMBEDDED CONTROL

***SRAP: Prototipazione e controllo di una
protesi transradiale robotica basata su sEMG***

Docente:

Prof. Ing. De Cicco Luca

Studente:

Nicolò Santilio

Anno Accademico 2021/2022

Abstract. Con il continuo miglioramento della tecnologia di controllo e l'aumento dell'accessibilità di sensori ad alta complessità e precisioni, pur rimanendo in fasce considerabili "low-cost", le possibilità di fornire alle persone disabili protesi di alta qualità e a basso costo sono diventate sempre più forti. Viene proposto un metodo di controllo di arti robotici basato sul segnale elettromiografico di superficie (SEMG) dell'avambraccio, denominato SRAP (Simple Robotic foreArm Prosthetic), per la produzione di protesi robotiche transradiali.

In primo luogo, i dati EMG ad un canale dell'avambraccio vengono ottenuti tramite un sensore specificatamente adibito all'acquisizione di segnali elettromiografici (Myoware v1), posto sull'avambraccio del paziente. Il segnale pre-filtrato acquisito viene elaborato attraverso una scheda di prototipazione e sviluppo, la Nucleo-F446RE, attraverso cui è implementato un loop di controllo su di un Servo motore attuatore AD-002 per la simulazione del movimento di un'articolazione protesica implementata da un plant (impalcatura con artiglio robotico, montata sul motore). Il sistema prevede inoltre la possibilità di personalizzare l'esperienza utente con la protesi robotica, attraverso un protocollo di calibrazione e configurazione. Il progetto ha dunque come obiettivo quello di emulare una protesi reale attraverso una articolazione controllata digitalmente mediante segnali elettrici trasmessi dall'avambraccio, consistente in una procedura non-invasiva, a bassa complessità sensoristica e basso costo.

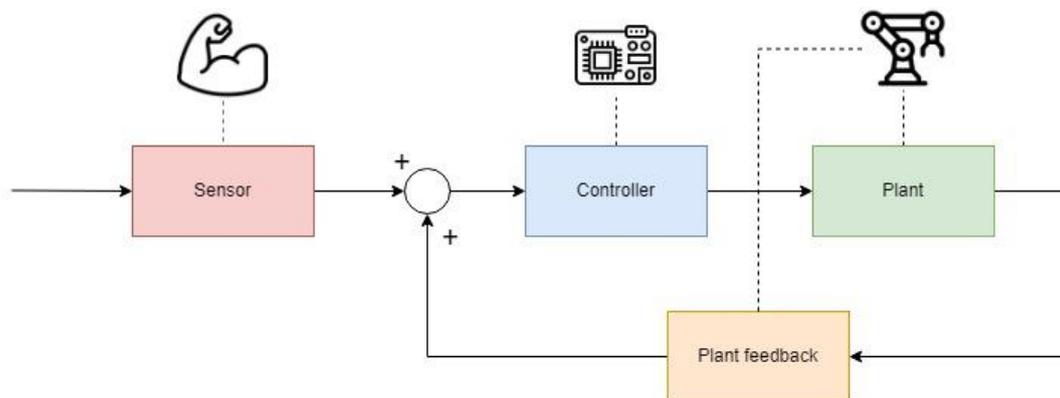


fig 1. Loop di controllo schematico

Note. A causa di un malfunzionamento del sensore Myoware v1, e della non reperibilità del sensore stesso sul mercato per via della crisi dei semiconduttori, nonché della pausa delle vendite del Myoware v2 come annunciato ufficialmente dalla casa produttrice, il progetto non ha potuto coprire tutti gli obiettivi prefissati. La conclusione della relazione illustrerà i punti non coperti, accompagnati da commenti su miglioramenti.

Sommario

1. Ricerche preliminari	4
1.1. Che cos'è l'elettromiografia	4
1.2. Tipologie di amputazioni e protesi	7
1.3. Anatomia dell'avambraccio e scelta del gruppo muscolare	9
1.4. Stato dell'arte sulla tecnologia	12
2. Prototipazione del plant	13
2.1. Descrizione dell'hardware	13
2.3. Descrizione del firmware	29
2.3.4. Routine del firmware	44
3. Analisi dei risultati e conclusione	45
Bibliografia.....	46

1. Ricerche preliminari

1.1. Che cos'è l'elettromiografia

L'elettromiografia è un esame utilizzato per diagnosticare problemi a livello nervoso periferico e dell'apparato muscolare.

L'esame si basa sulla rilevazione del Potenziale d'azione muscolare, ovvero quell'evento di breve periodo durante il quale l'energia di una cellula subisce una rapida variazione in termini di differenza di potenziale elettrico a livello di membrana cellulare.

Ogni cellula possiede un proprio potenziale di riposo naturalmente negativo (-70mV per le cellule nervose, -90mV per le cellule muscolari), variabile a causa di un evento che produce una depolarizzazione della membrana oltre un livello di soglia. Raggiunto questo livello, si innescano una serie di processi consistenti in scambi di ioni attraverso la membrana che generano un ciclo di fasi di polarizzazione e depolarizzazione.

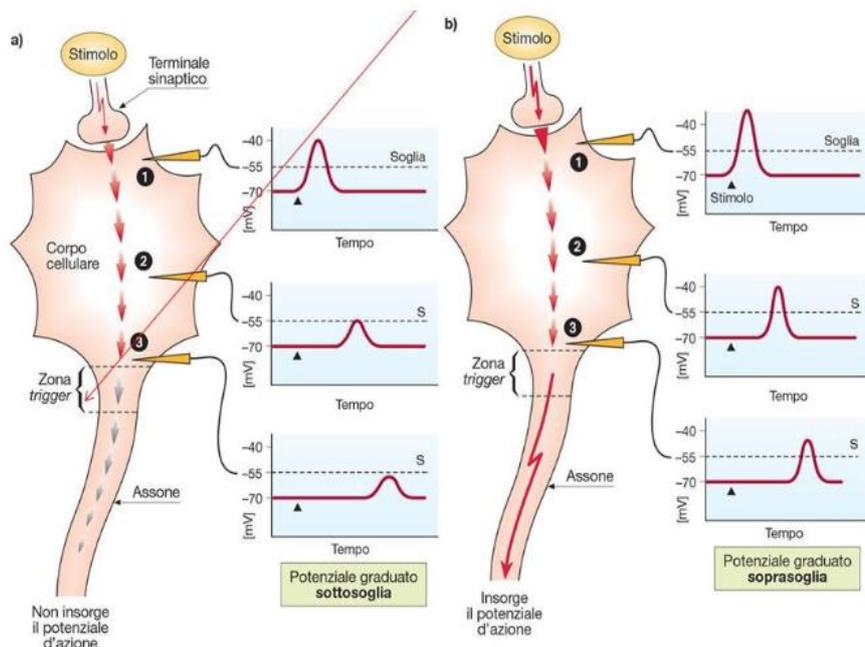


fig 2. Potenziale generato da un motoneurone

Per quanto riguarda le cellule muscolari, e in particolare le cellule dei muscoli scheletrici, il potenziale d'azione è simile a quello dei neuroni, con la differenza di un potenziale a riposo molto più alto (in segno negativo) rispetto a quello di tutte le altre cellule.

La depolarizzazione del sarcolemma (la membrana che ricopre le fibre del tessuto muscolare) apre i canali sodio tensione-dipendenti. Quando questi diventano inattivi, la membrana raggiunge un livello di potenziale positivo per l'azione della corrente esterna degli ioni potassio. La durata del potenziale d'azione del muscolo scheletrico è di circa 2-4ms, il Periodo Refrattario Assoluto dura circa 1-3ms, mentre la velocità di conduzione è di appena 5m/s.

Ciò che permette la polarizzazione e depolarizzazione della membrana muscolare è l'arrivo di uno stimolo neurale pre-sinaptico alle giunzioni muscolari attraverso la trasmissione

dell'acetilcolina dalle sinapsi cerebrali, che genera appunto il potenziale d'azione, innescando il meccanismo di contrazione nei miociti (fibrocellule nel tessuto muscolare). Tale trasmissione è guidata dai neurotrasmettitori, messaggeri chimici endogeni, di cui si avvalgono le cellule del sistema nervoso (i cosiddetti neuroni) per comunicare tra loro o per stimolare cellule di tipo muscolare o ghiandolare.

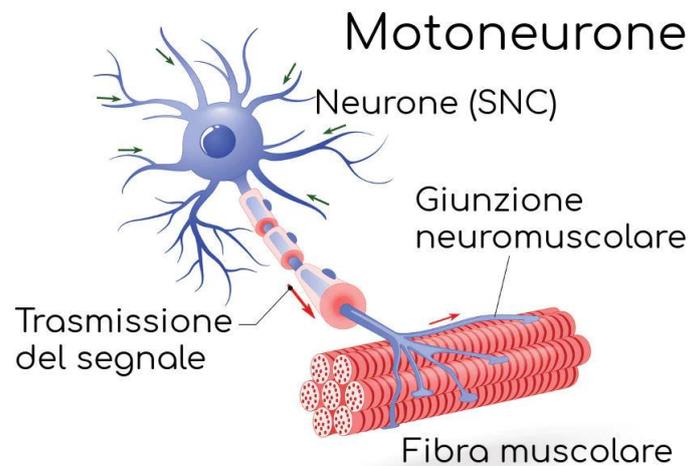


fig 3. Un motoneurone

Dunque, i dispositivi e i sensori di EMG permettono la rilevazione e l'amplificazione di questi segnali, reperibili non solo a livello extra e intra muscolare, ma anche a livello superficiale attraverso elettrodi applicati sulla pelle. Mentre queste apparecchiature però, sono principalmente utilizzate per l'amplificazione di questi potenziali con lo scopo di rilevare problemi neuro-muscolari relativi a diversi tipi di muscoli (da quelli cardiaci a quelli scheletrici), hanno anche un'efficienza tale da poter rilevare qualsiasi tipo di segnale elettrico (di giusta intensità) generato da un'attività muscolare.

A seconda di come il segnale è registrato, in generale, si parla di elettromiografia di superficie (sEMG) o elettromiografia "ad ago" (detta anche elettroneurografia, ENG). La sEMG prevede il prelievo del segnale tramite degli elettrodi posti sulla pelle (elettrodi superficiali), mentre l'EMG ad ago prevede l'utilizzo di aghi sottocutanei posti in diretto contatto con il muscolo di interesse (l'ago sarà tanto più lungo quanto più il muscolo è profondo). I potenziali registrati andranno quindi ad evidenziare un gruppo di unità motorie (nervi) e la loro velocità di conduzione con gli elettrodi superficiali, mentre con gli elettrodi ad ago si potrà analizzare una singola unità motoria (l'EMG permette di "guardare" direttamente nel muscolo).

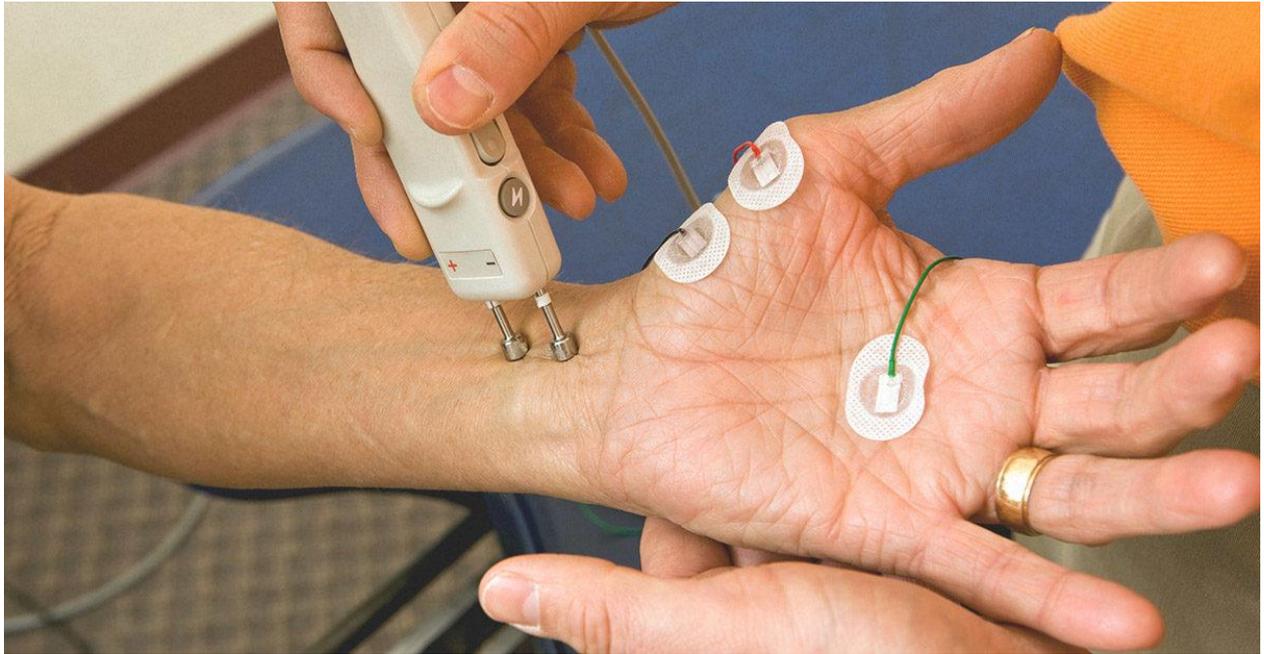


fig 4. Esempio di elettromiografia

Allontanando la posizione degli elettrodi dal cuore sarà sperimentalmente osservabile una rilevazione meno accurata del potenziale elettrico. Nonostante ciò esistono punti specifici in diverse zone delle braccia e delle gambe dove è ancora possibile rilevare e dunque amplificare il potenziale elettrico con valori di rumore accettabili.

Per quanto riguarda il braccio umano - oggetto sperimentale di questo progetto - il posizionamento degli elettrodi (rilevazione a due canali) è strettamente dipendente dal tipo di muscolo di cui si vuole rilevare il potenziale.

In principio, la rilevazione dell'attività muscolare avviene trovando e scegliendo la coppia antagonista di muscoli.

Si definisce muscolo "antagonista" quel muscolo che, relativamente ad un muscolo "agonista", svolge la funzione opposta di contrazione od estensione.

La coppia bicipite-tricipite ne è il perfetto esempio.

Durante la contrazione del bicipite dunque, l'azione di riduzione dell'angolo di inclinazione tra l'avambraccio e il bicipite stesso, il tricipite svolge la funzione di "antagonista": si rilassa ed estendendosi permette la contrazione dell'agonista bicipite.

Durante il rilassamento del bicipite, invece, questo risulta essere antagonista del tricipite, poiché è quello che si sta contraendo, svolgendo la funzione di agonista.

Sarà dunque possibile rilevare il potenziale elettrico di due muscoli durante la loro azione agonista-antagonista, quindi contrazione-estensione.

È però anche possibile una rilevazione ad un canale, dunque con l'utilizzo di soli tre elettrodi, scegliendo solo uno dei due muscoli nella coppia agonista-antagonista.

Utilizzare un potenziale d'azione per tradurlo in un movimento meccanico infatti, non richiede necessariamente l'esistenza di un muscolo antagonista che ne rilevi la contrazione-estensione.

Di per sé è la contrazione stessa di un muscolo che innalza il potenziale elettrico. Nel ritorno alla fase di riposo, il potenziale d'azione raggiunge livelli distinguibili dal valore attivo.

Un dispositivo a un canale possiede in generale tre elettrodi: due verranno utilizzati per rilevare la differenza di potenziale sulla superficie cutanea lungo un tratto, mentre il terzo verrà

utilizzato come messa a terra, dunque posizionato in un punto a potenziale nullo (una zona muscolare non attiva, generalmente sulle ossa es. gomito).

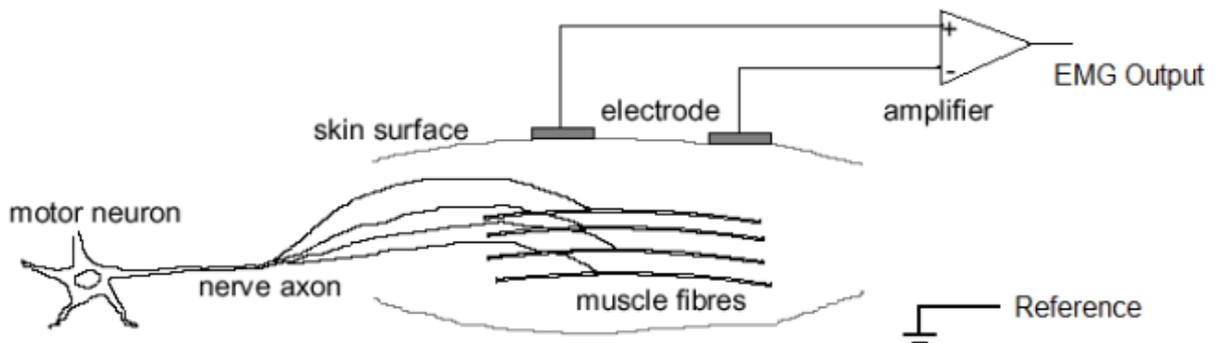


fig 5. Posizionamento degli elettrodi per un sensore elettromiografico

La posizione e la distanza degli elettrodi che rilevano il potenziale d'azione è molto importante:

1. per il posizionamento, i due elettrodi dovranno essere posti sul muscolo di cui ci interessa leggerne il potenziale, lontani dalle estremità di un muscolo (per non rilevare l'azione di muscoli adiacenti)
2. gli elettrodi devono essere lontani dal punto motore del muscolo, poiché la sua rilevazione comporterebbe la sottrazione e l'addizione di frequenze elevate erranee
3. gli elettrodi devono essere posti sullo stesso asse
4. la distanza degli elettrodi per ottenerne una rilevazione il più efficiente possibile è di circa 1-2 cm, dal momento che il potenziale si distribuisce e propaga per tutto il muscolo

L'elettrodo neutro viene posto in un punto a potenziale nullo (o neutro) sia per ottenere un valore di riferimento neutro durante la rilevazione, sia per abbassare il livello del rumore e le interferenze. I punti principali generalmente sono le parti ossee o a poca/nulla attività muscolare.

L'applicazione degli elettrodi è preceduta da una preparazione dei punti di applicazione. La presenza di cellule morte, peli e polvere inciderebbe sulla rilevazione del potenziale, si consiglia dunque di pulire e rendere asciutta la superficie prima dell'applicazione.

1.2. Tipologie di amputazioni e protesi

Ogni anno nel mondo vengono segnalate un milione di amputazioni di arti. E nel 2017, 57.7 milioni di persone in tutto il mondo hanno vissuto con amputazioni traumatiche.

I dati di Stanford Healthcare mostrano un aumento del 49% del numero totale di amputazioni durante il periodo della pandemia di COVID-19, da marzo 2020 a febbraio 2021.

Ci sono diverse condizioni che possono portare all'amputazione:

- Grave infezione con esteso danno tissutale
- Cancro
- Traumi derivanti da incidenti o lesioni, come schiacciamento o ferita da esplosione
- Deficit congenito/ pediatrico dell'arto sottoposto ad amputazione di conversione
- Deformità congenite delle dita o degli arti

- Dita o arti extra congeniti
- Necrosi o fascite necrotizzante
- Cellulite
- Malattia arteriosa periferica
- Congelamento
- Tumore maligno/canceroso nell'osso o nel muscolo dell'arto, ad es . Osteosarcoma
- Condizioni che influenzano il flusso sanguigno, ad esempio il diabete

Inoltre, esistono diverse tipologie di amputazione, in base all'arto di interesse. In particolare, per gli arti superiori, oggetto di studio in analisi:

- Quarto anteriore
- Disarticolazione della spalla (DS)
- Transomerale (sopra il gomito AE)
- Disarticolazione del gomito (DE)
- Transradiale (sotto il gomito BE)
- Disarticolazione mano/polso
- Transcarpale (PH parziale della mano)
- Transmetacarpale

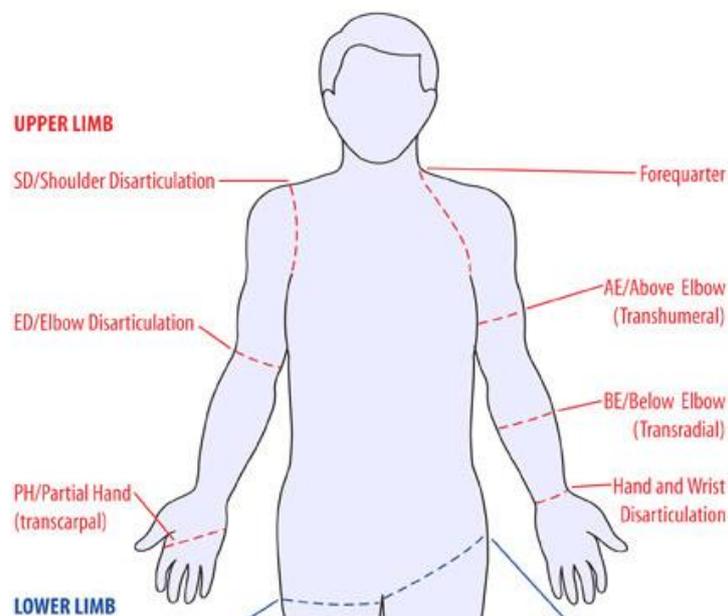


fig 6. Schema delle amputazioni a livello di arti superiori

Il radio è una delle due ossa lunghe che compongono l'avambraccio, che va dal polso al gomito. L'amputazione transradiale è l'amputazione parziale del braccio sotto il gomito, in un punto lungo l'osso radiale. Questo tipo di intervento chirurgico lascia intatti il gomito e la maggior parte del braccio, il che rende più facile il recupero e rende più probabile la possibilità di continuare a usare il braccio anche dopo l'amputazione.

Esistono tre tipi fondamentali di protesi transradiali:

- Protesi estetica, solo per apparenza e non ha capacità motorie;

- Protesi convenzionale (o alimentata dal corpo), è collegata al corpo tramite una serie di cavi. Muovendo il corpo in diversi modi, è possibile muovere la protesi e persino aprire e chiudere la mano artificiale;
- Protesi mioelettrica, è la forma più recente e avanzata di protesi transradiale. Collega una mano elettronica ai muscoli del braccio. Quando i muscoli si contraggono, gli elettrodi inviano un segnale all'arto artificiale, facendolo muovere più o meno allo stesso modo di una mano reale.

Il progetto si è dunque concentrato sulla riproduzione di una protesi mioelettrica transradiale.

1.3. Anatomia dell'avambraccio e scelta del gruppo muscolare

Al fine di creare una articolazione protesica che, seppur semplificata, riuscisse a rispecchiare e tradurre il basilare movimento dell'avambraccio (contrazione e rilassamento) nel movimento di un attuttore elettronico (Servomotore controllato digitalmente), proprio come farebbe una protesi transradiale robotica, è stato effettuato uno studio anatomico incentrato sulla composizione muscolare dell'avambraccio, per una scelta accurata del gruppo muscolare di riferimento su cui sarebbero stati posti gli elettrodi del sensore elettromiografico per l'analisi ad un canale.

L'avambraccio e la mano sono collegati dal polso, formando insieme un'unità funzionale in cui sono presenti circa 30 muscoli che lavorano di concerto in modo molto complesso, consentendo di svolgere un'ampia gamma di attività, molte delle quali con un alto livello di precisione. Lo strato superficiale dell'avambraccio posteriore è costituito da sette muscoli (Figura 7A), mentre quello anteriore ha quattro muscoli diversi (Figura 7B).

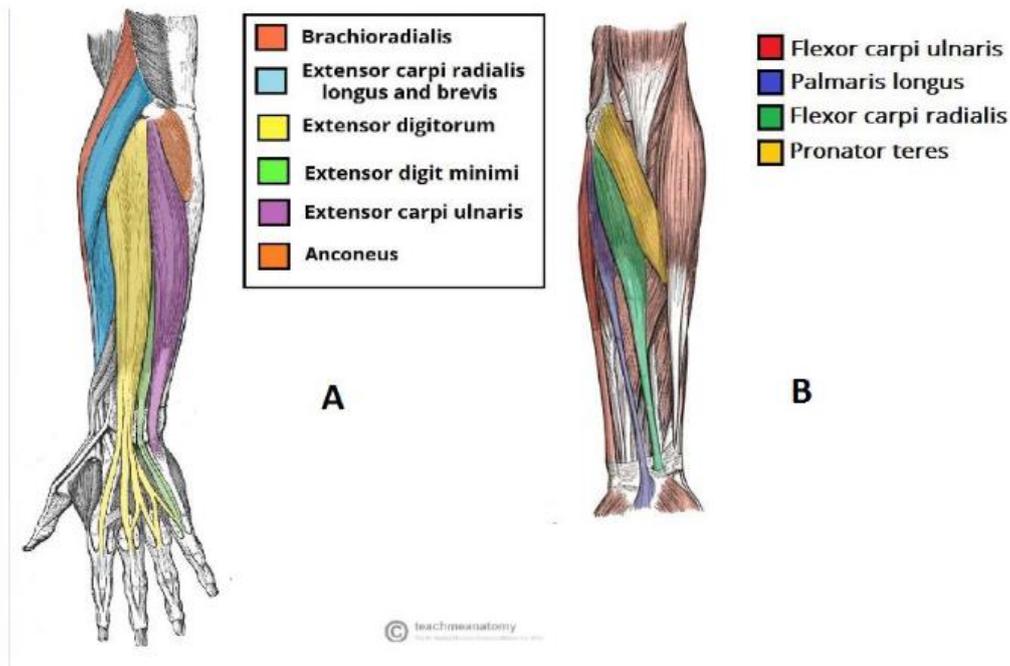


fig 7. Fasci muscolari su arti superiori

Come mostrato nella figura 8, le dita non hanno muscoli. Tutti i muscoli che controllano le dita si trovano nel palmo e nell'avambraccio. Questi muscoli si connettono al dito attraverso i tendini. I muscoli che aiutano a controllare la mano iniziano dal gomito o dall'avambraccio correndo giù per incrociare il polso e la mano. Pochi di questi muscoli posizionano e stabilizzano il polso e le mani mentre altri aiutano a consentire al pollice e alle dita di afferrare oggetti e svolgere una funzione motoria. Come mostrato in figura 8, i principali muscoli che muovono il dito su e giù sono il flessore delle dita e l'estensore delle dita, mentre il muscolo interosseo nella mano lascia che le dita si muovano da un lato all'altro.

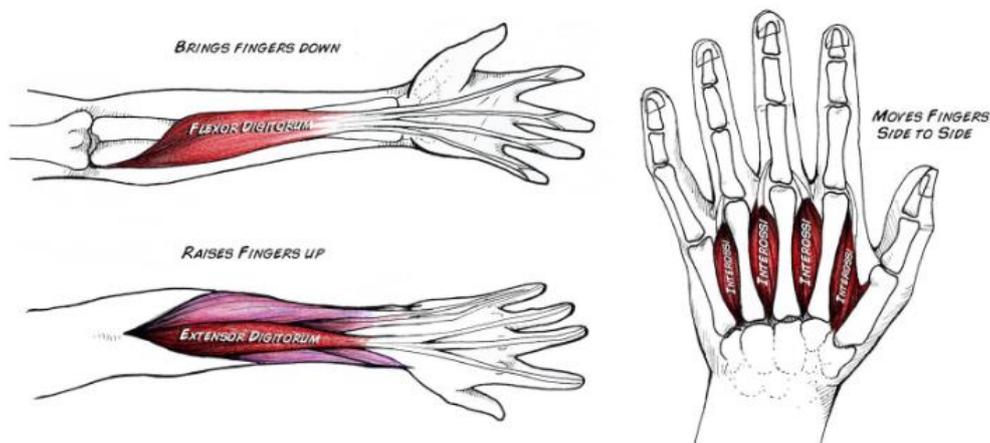


fig 8. Muscoli legati al principale movimento della mano

Dunque, se attraverso un sensore EMG si volesse controllare un servomotore in grado di simulare l'apertura e la chiusura di una mano, mediante la rilevazione dell'attività neuro-muscolare dell'avambraccio, risulterebbe molto difficile capire quale gruppo muscolare sarebbe più opportuno scegliere per il posizionamento dei due elettrodi, positivo e negativo, del sensore.

Secondo uno studio pubblicato sul *Journal of NeuroEngineering and Rehabilitation* nel 2018 [\[1\]](#), è stato possibile effettuare un'accurata classificazione degli spot accomunati da simili attività neuro-muscolari, mediante l'analisi dei segnali EMG trasmessi dai vari gruppi muscolari dell'avambraccio, attraverso un approfondimento nello studio dell'attività dei muscoli dell'avambraccio durante la simulazione di attività vita quotidiana (ADL).

In particolare, dallo studio è stato rilevato che è possibile classificare i movimenti dell'avambraccio in sette gruppi muscolo-anatomicamente coerenti: (1) flessione del polso e deviazione ulnare; (2) flessione del polso e deviazione radiale; (3) flessione delle dita; (4) estensione del pollice e abduzione/adduzione; (5) estensione del dito; (6) estensione del polso e deviazione ulnare; e (7) estensione del polso e deviazione radiale.

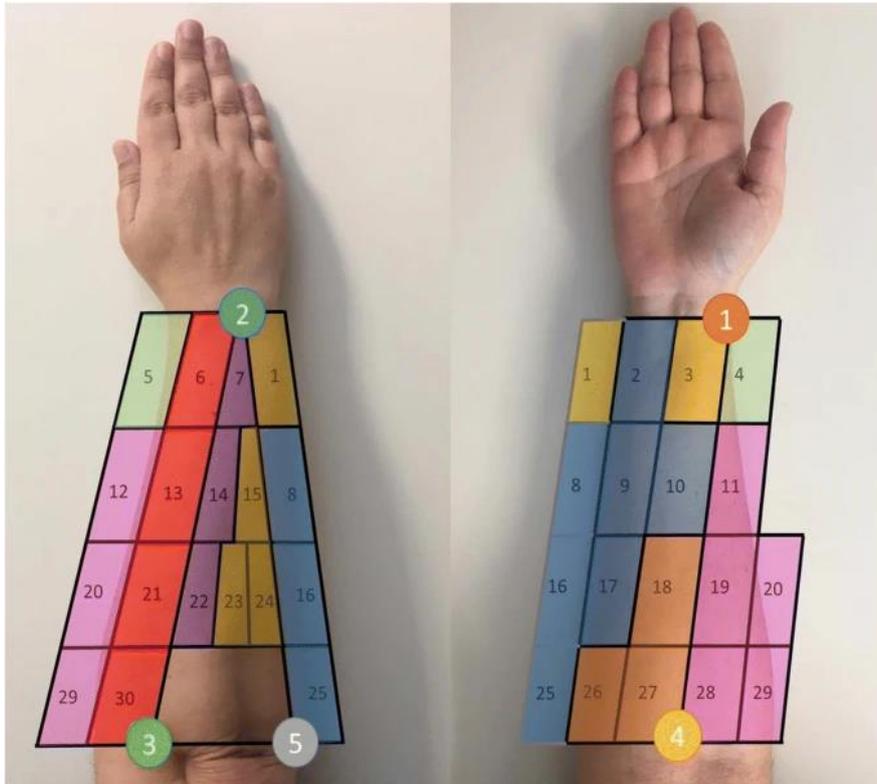


fig 10. Gruppi muscolari sull'avambraccio, classificati secondo lo studio dell'ADL

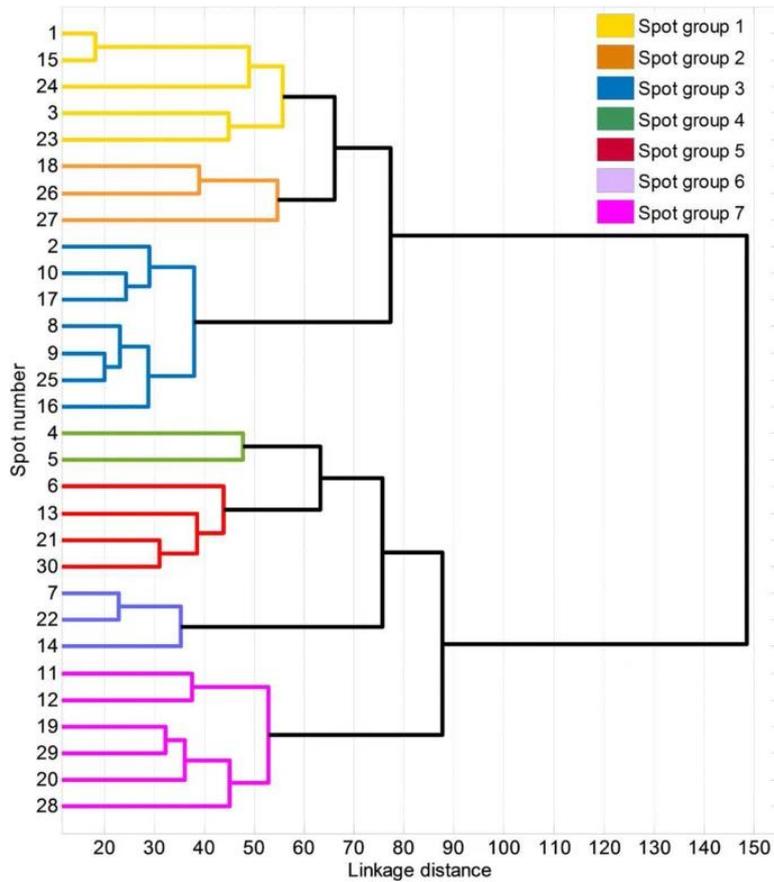


fig 11. Classificazione dei fasci muscolari in gruppi per funzionalità

Secondo lo studio, osservando il dendrogramma ottenuto dall'analisi dei cluster, è possibile scegliere numeri diversi di cluster. Si possono osservare due livelli gerarchici principali: uno coinvolge i muscoli flessori del polso e delle dita (gruppi di punti da 1 a 3) e l'altro include i loro muscoli estensori (gruppi di punti da 4 a 7). Nel livello gerarchico successivo, i flessori si biforcano in flessori del polso (gruppi 1 e 2) e flessori delle dita (gruppo 3); è noto che anche i flessori delle dita contribuiscono alla flessione del polso. E qualcosa di simile si verifica per gli estensori, che si biforcano in estensori delle dita (gruppi 4 e 5) ed estensori del polso (gruppi 6 e 7). Pertanto, nel caso di utilizzo di due segnali per il controllo della protesi, sarebbe logico considerare i due livelli gerarchici principali per un controllo intuitivo nell'esecuzione dell'ADL, associando i segnali sEMG all'esecuzione dei movimenti di flessione ed estensione della protesi. Nel caso di utilizzo di quattro segnali di controllo, potremmo utilizzare il livello gerarchico successivo descritto, differenziando tra flessori ed estensori della mano e del polso. Tuttavia, maggiore è il numero di segnali, più difficile diventerà separarli e l'utilizzo di raggruppamenti diversi da quelli risultanti dal dendrogramma non sarà così intuitivo.

Poiché l'obiettivo della prototipazione è, come affermato precedentemente, quello di voler simulare la chiusura di una mano, o più in generale il movimento contrattivo dell'avambraccio legato a tale azione, è stato identificato come gruppo muscolare sperimentale il gruppo 3, poiché legato ai muscoli flessori delle dita e del polso.

Nello specifico, il gruppo 3 è definito dai punti 2, 8, 9, 10, 16, 17 e 25 e potrebbe registrare l'attività muscolare dei muscoli flessori delle dita: flessore superficiale delle dita (FDS) e profondo (FDP) e flessore lungo del pollice (FPL). I muscoli FDS e FDP sono flessori delle dita e si trovano sulla stessa area dell'avambraccio, ma a profondità diverse. FPL è un flessore di tutte e tre le articolazioni del pollice ed è l'unico flessore dell'articolazione interfalangea (IP) del pollice.

Infine, per la scelta dello spot specifico per il posizionamento del sensore EMG (ricordando che per una accurata rilevazione la distanza degli elettrodi non deve superare gli 1-2 centimetri), è stato scelto quello che nel gruppo presentasse il più alto valore di RMS, indicatore rappresentativo per tutto il gruppo di appartenenza di uno spot. In particolare per il gruppo 3, lo spot a più alto RMS risulta essere il 16.

1.4. Stato dell'arte sulla tecnologia

Al giorno d'oggi, ci sono molte diverse protesi transradiali commerciali disponibili e anche molti dei prodotti protesici per le mani in fase di sviluppo. Questa sezione ne menziona alcune.

Il "Rehabilitative Robotic Glove" è basato sull'utilizzo di un attuatore basato su cavo durevole. Il guanto è stato progettato dal laboratorio AIM del Politecnico di Worcester che aveva lo scopo di aiutare la riabilitazione di coloro che avevano subito di recente un infarto. Secondo il rapporto del laboratorio AIM, i guanti utilizzano un sistema di cavi per aprire e chiudere la mano del paziente, con servomotori che azionano i cavi. Il gruppo di ricerca ha scelto i cavi Kevlar k49 per la loro resistenza alla trazione e bassa elasticità. I cavi sono posizionati su ciascun lato del dito e attaccati a un servo e una bobina. Come risultato di questo design, in base alla direzione in cui si muovono i servi, i guanti tirano sempre le dita verso la posizione desiderata.

La “Bebionic Hand” è stata progettata da Ottobock, un'azienda tecnologica tedesca. Finora, la mano artificiale Bebionic è una delle più realistiche, funzionali e facili da usare. Questa mano protesica non solo può offrire 14 diversi modelli di presa ma anche varie opzioni di polso come disconnessione rapida, multi-flex, flessione e polso corto. L'azienda ha anche sviluppato un sensore integrato per consentire all'utente di avere una posizione del pollice selezionabile. La mano Bebionic può sopportare fino a 45 chili grazie alla sua resistenza di costruzione e materiali avanzati. I pazienti possono usare con sicurezza questa mano artificiale per trasportare oggetti pesanti o sollevarsi dalla posizione di seduta.

e-NABLE è una community per tutte le persone di tutto il mondo che utilizzano stampanti 3D per creare mani o braccia stampate in 3D per chi ha bisogno di un arto superiore artificiale. Il community è stata fondata nel 2013 e finora ha fornito oltre duemila design per chi ha bisogno di un braccio protesico. Al momento, la maggior parte dei design lo sono alimentato meccanicamente da un polso o gomito funzionale. Il primo progetto mioelettrico di e-NABLE è “Limbless Arm” che è un design opensource con Arduino Micro come microcontrollore, un unico servo in grado di produrre coppia, sensori muscolari e un cordino in Kevlar per muovere le dita. La caratteristica meno vantaggiosa di questo design rispetto al precedente è quella di essere dotata di un solo servomotore, dunque la mano può solo aprirsi e chiudersi. D'altra parte, il “Limbless Arm” è molto economico e comodo per l'utente.

2. Prototipazione del plant

2.1. Descrizione dell'hardware

Innanzitutto sono state effettuate delle ricerche per identificare il miglior sensore candidato per il rilevamento del segnale EMG di superficie.

Al momento dell'inizio del progetto, erano presenti poche alternative per la scelta del sensore adatto, in quanto tutti i sensori a basso costo sul mercato che affermavano di poter essere utilizzati per EMG sono dotati di elettrodi a cavo, nella letteratura sconsigliati per questa specifica applicazione, poiché altamente più soggetti a rumore. Questa tipologia di sensori EMG è generalmente più consigliata per effettuare altre tipologie di analisi, con una sensibilità elettromiografica inferiore.

L'unica alternativa accettabile è stata dunque identificata dal sensore muscolare MyoWare (AT-04-001) sviluppato dall'azienda Advancer Technologies. Questo tipo di sensore presenta infatti la caratteristica di essere “wearable”, garantendo così non solo la compattezza stessa dell'hardware, possibile da montare direttamente sull'avambraccio, ma soprattutto garantendo una buona sensibilità essendo dotata di slot per gli elettrodi direttamente saldati sulla PCB ad una breve distanza, con l'elettrodo di riferimento collegato attraverso un unico cavo.

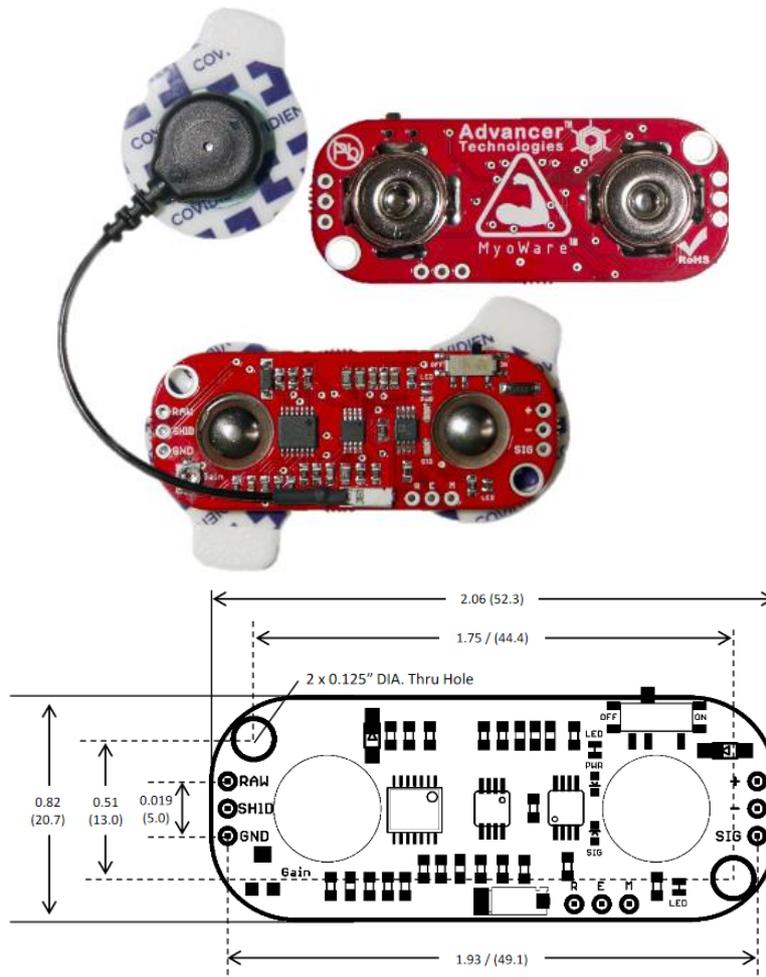


fig 12 & 13. Immagine e dimensioni board Myoware v1

Il segnale del sensore muscolare Myoware viene raccolto dagli elettrodi che sono direttamente attaccati alla pelle dell'utilizzatore. Nel mercato attuale, ci sono molti diversi tipi di elettrodi; tuttavia, questo dispositivo si concentrerà su elettrodi medici multiuso e tessuto conduttivo. Gli elettrodi di superficie più comunemente utilizzati sono quelli di tipo Ag-AgCl, in gel, adesivi e usa e getta. Presentano tipicamente una forma circolare a singolo bottone o duali, a seconda della distanza a cui si vogliono posizionare. L'area conduttiva, costituita dal disco argentato ricoperto di AgCl, deve essere di circa 1 cm o minore. Per accompagnare l'utilizzo del Myoware sono dunque stati utilizzati degli elettrodi passivi a bottone generalmente utilizzati in ambito medico e sportivo, i FITOP.



fig 14. Elettrodo a bottone FITOP

Per avere un buon controllo della mano protesica, è meglio che gli elettrodi ricevano un segnale forte e chiaro. In realtà ci sono vari modi per ottenerlo, come mantenere il contatto pelle-superficie, ricevere un rumore e un'interferenza minimi e avere un'elevata conduttività. Come affermato da Advancer Technologies, i sensori muscolari Myoware sono progettati per essere utilizzati direttamente con un microcontrollore. Pertanto, l'uscita primaria del segnale del sensore non è un segnale EMG "raw" ma piuttosto un segnale amplificato, rettificato e segnale integrato (AKA l'involuppo dell'EMG) che funziona bene con un qualsiasi microcontrollore dotato di convertitore analogico-digitale (ADC).

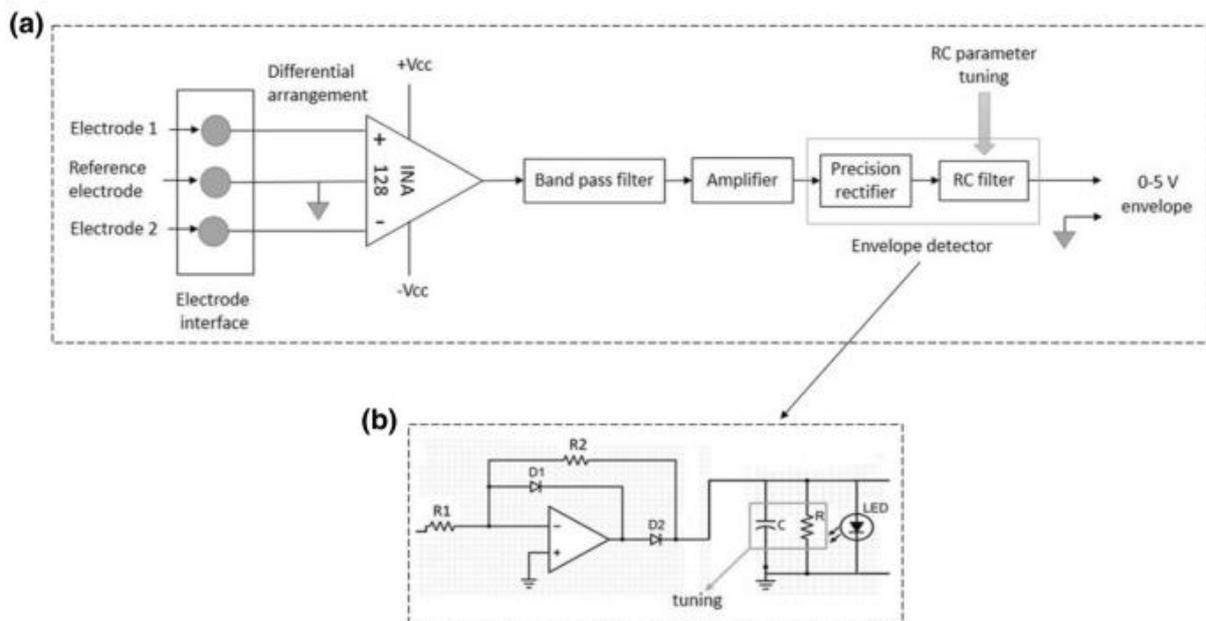


fig 15. Schema a blocchi del funzionamento elettrico-digitale di un sensore EMG

Per quanto concerne l'attuatore invece, la scelta era principalmente orientata da due requisiti primari:

1. la possibilità di controllare con precisione la posizione/rotazione del dispositivo, in maniera tale da mappare l'intensità del segnale EMG (dipendente dalla contrazione/rilassamento del muscolo dell'avambraccio) con un puntuale spostamento del dispositivo stesso

2. la possibilità di verificare in retroazione la posizione effettiva del dispositivo rispetto a quella che ci si aspetta di avere di seguito all'input proveniente dal microcontrollore, per verificare che l'attuatore non sia soggetto a malfunzionamenti o ad impedimenti, e per verificare continuamente che ad ogni azione proveniente dall'utilizzatore corrisponda una reazione adeguata

In base a questi punti la scelta è ricaduta su un attuatore rotativo, comunemente un motore, controllato in corrente continua e che potesse applicare un momento meccanico adeguato per lo spostamento di una minimale impalcatura, consistente in una pinza meccanica, per simulare l'apertura e la chiusura di una mano.

La scelta è stata dunque fatta prendendo in considerazione le due principali alternative di attuatore rotativo, date le circostanze: un semplice motore a corrente diretta (DC) dotato di encoder, oppure un servomotore.



fig 16. Esempi di motori DC (A) motore semplice (B) servomotori

Di seguito, una tabella riassuntiva per i pro e i contro nell'utilizzo dei servomotori e dei motori lineari in applicazioni generali:

	Motori DC	Servomotori
Pro	<ul style="list-style-type: none"> - più adatti per le applicazioni ad alta velocità; - utilizzati nell'applicazione dove è richiesta la rotazione continua; - non richiedono alcun circuito esterno per il controllo della velocità (controllo in tensione di corrente); - disponibili in diversi livelli di tensione e velocità; - possono funzionare per un periodo di tempo più lungo; 	<ul style="list-style-type: none"> - la posizione può essere controllata in modo più preciso - sono più adatti per le applicazioni in cui il movimento angolare dovrebbe essere accurato - sono disponibili per un'ampia gamma di coppie e rotazioni angolari. - possono essere facilmente interfacciati con il microcontrollore per ottenere una rotazione angolare precisa;
Contro	<ul style="list-style-type: none"> - non adatti per i casi dove è richiesto un preciso movimento angolare; - forniscono una coppia inferiore 	<ul style="list-style-type: none"> - nella maggior parte dei casi possono ruotare solo da 0 a 180 gradi - necessitano di un circuito esterno per controllare la rotazione angolare, nonché

	senza il sistema di ingranaggi esterni collegato ad esso;	di una libreria software adatta poiché usano PWM; - possono essere ruotati solo elettricamente;
--	---	--

In entrambi i casi, la dimensione della componente non ha suscitato problematiche nella scelta, in quanto entrambe le tecnologie presentano attualmente dimensioni del tutto accettabili in confronto alla coppia che riescono a sprigionare, considerando anche l'eventualità di rendere l'attuatore stesso indossabile, montato sull'avambraccio di una persona.

Mentre nel caso di un motore a corrente diretta, data la necessità di controllare con precisione l'angolo di rotazione dell'albero, nonché di ricevere in feedback l'attuale posizione, sarebbe stato necessario accoppiare un encoder (aumentando la complessità dell'attuatore nella sua interezza), nel caso di un servomotore non è generalmente possibile ottenere direttamente la posizione dell'albero stesso, nonostante sia comunque possibile controllare la rotazione con precisione, sia in termini di velocità che di angolo.

Tale limite può essere però facilmente superato attraverso una modifica hardware, sperimentata in questo progetto, che consentirebbe di utilizzare un qualsiasi servomotore non solo come attuatore ma addirittura come sensore di posizione angolare.

Ricordiamo che un servomotore è una specifica implementazione di un generico motore ad alimentazione diretta, dotato di ingranaggi che consentono di sviluppare maggiore coppia e di una unità di controllo che consente di convertire un impulso elettrico, trasmesso attraverso la tecnica PWM (pulse width modulation), in un preciso angolo di rotazione su 360 gradi.

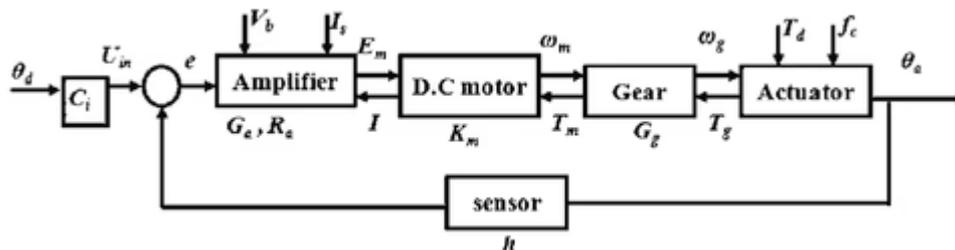


fig 17. Schema a blocchi del funzionamento interno di un servomotore

La maniera con cui però un servomotore stesso è in grado di definire uno specifico angolo di rotazione per uno specifico valore di impulso, deriva dalla natura del circuito interno dei servomotori: l'albero del servomotore è connesso ad un potenziometro che, in base alla rotazione stessa dell'albero, fornisce all'unità di controllo interna informazioni in merito alla rotazione, come segnale di feedback per consentirne il confronto con la posizione target desiderata, definita dal segnale in input.

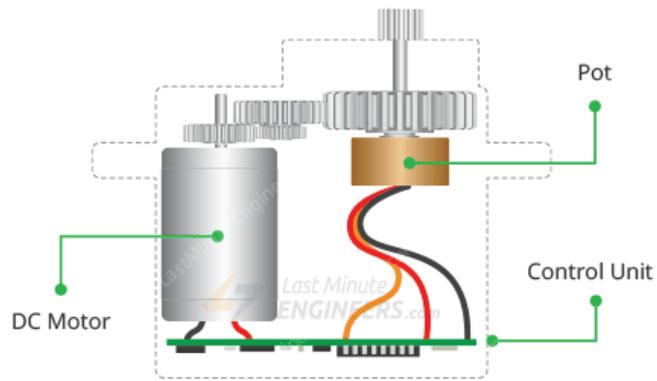


fig 18. Schema rappresentativo dei meccanismi interni di un servomotore

La modifica sperimentata consiste dunque nel consentirci di poter leggere il valore fornito dal potenziometro interno al motore per ottenere un valore analogico, corrispondente alla posizione reale dell'albero del servomotore.

Per fare ciò è stato aperto l'involucro del servomotore utilizzato per il progetto, è stato identificato sulla PCB dell'unità di controllo del servomotore il potenziometro, ed è stato saldato un filo aggiuntivo al pin del potenziometro corrispondente al cursore interno scorrevole, da cui poter leggere il valore di resistività del potenziometro. È stato infine allargato il foro di uscita dei tre cavi del servomotore, in maniera tale da permettere l'alloggio del quarto filo aggiuntivo.

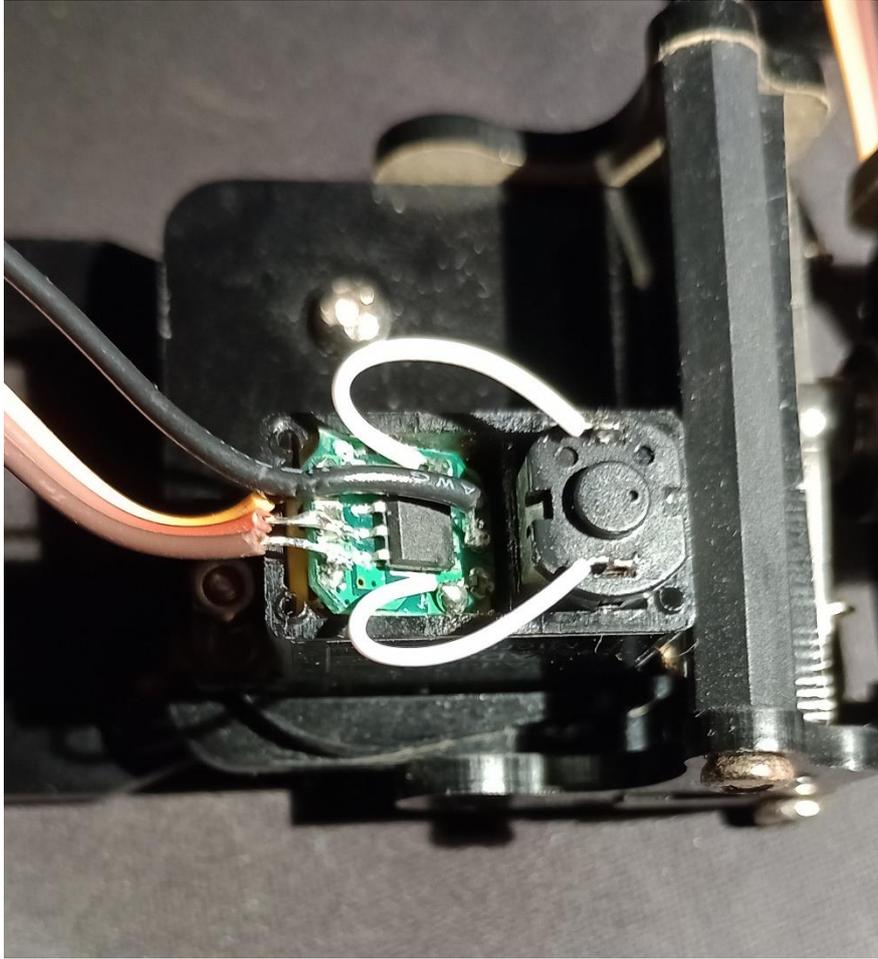


fig 19. Modifica sul servomotore per la lettura del potenziometro

Per verificare l'effettivo funzionamento di tale modifica, è stato progettato un semplice circuito di test leggendo il valore in uscita dal cavo collegato al potenziometro attraverso un Arduino Nano.

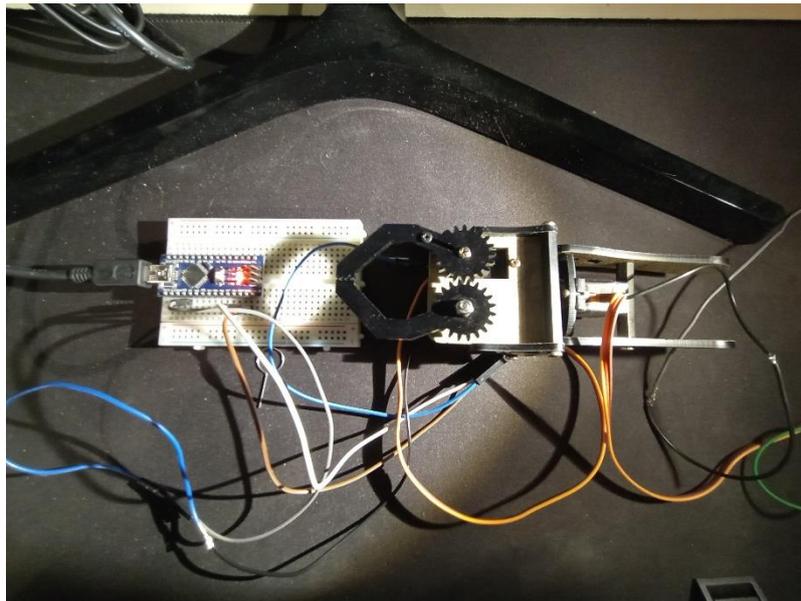
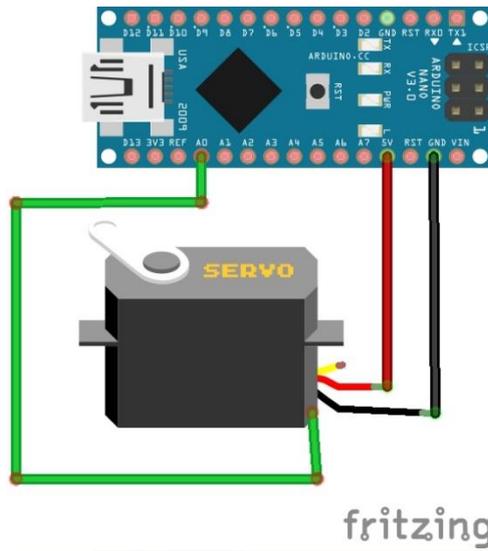
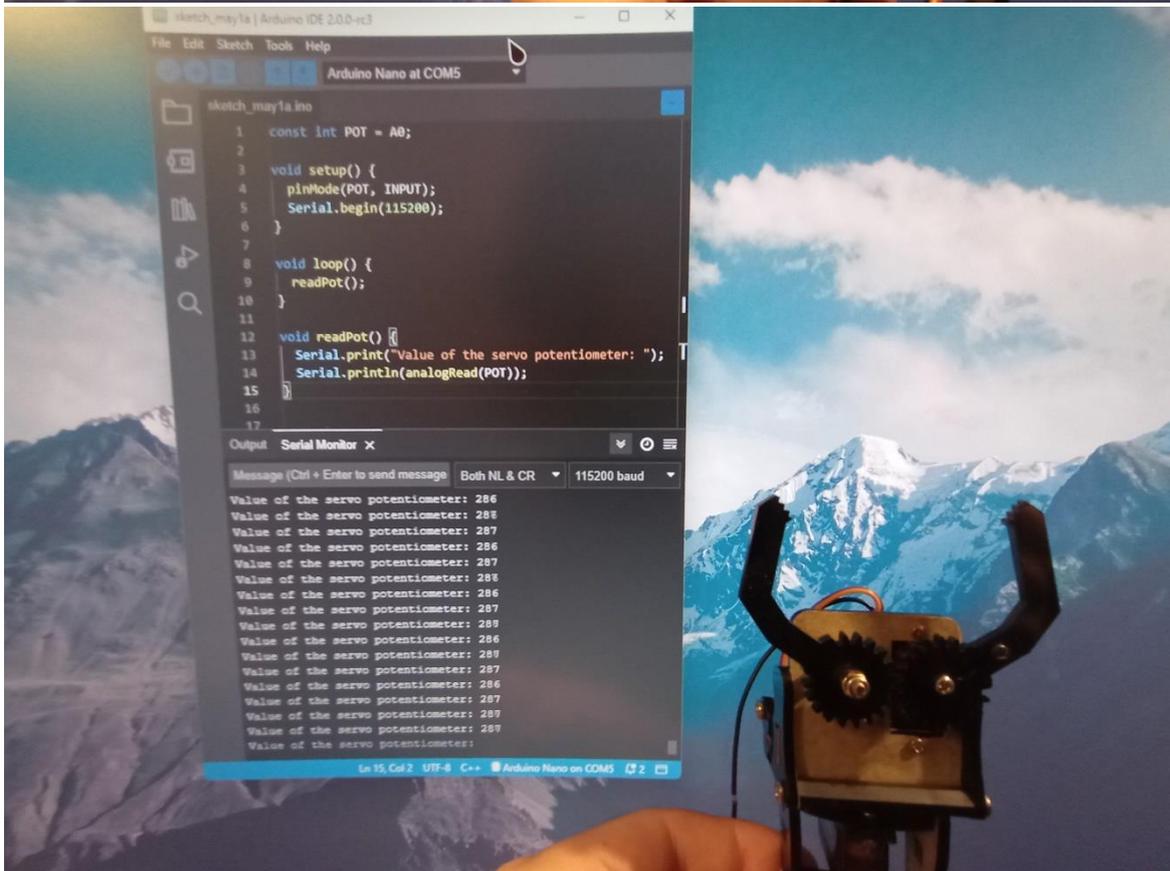
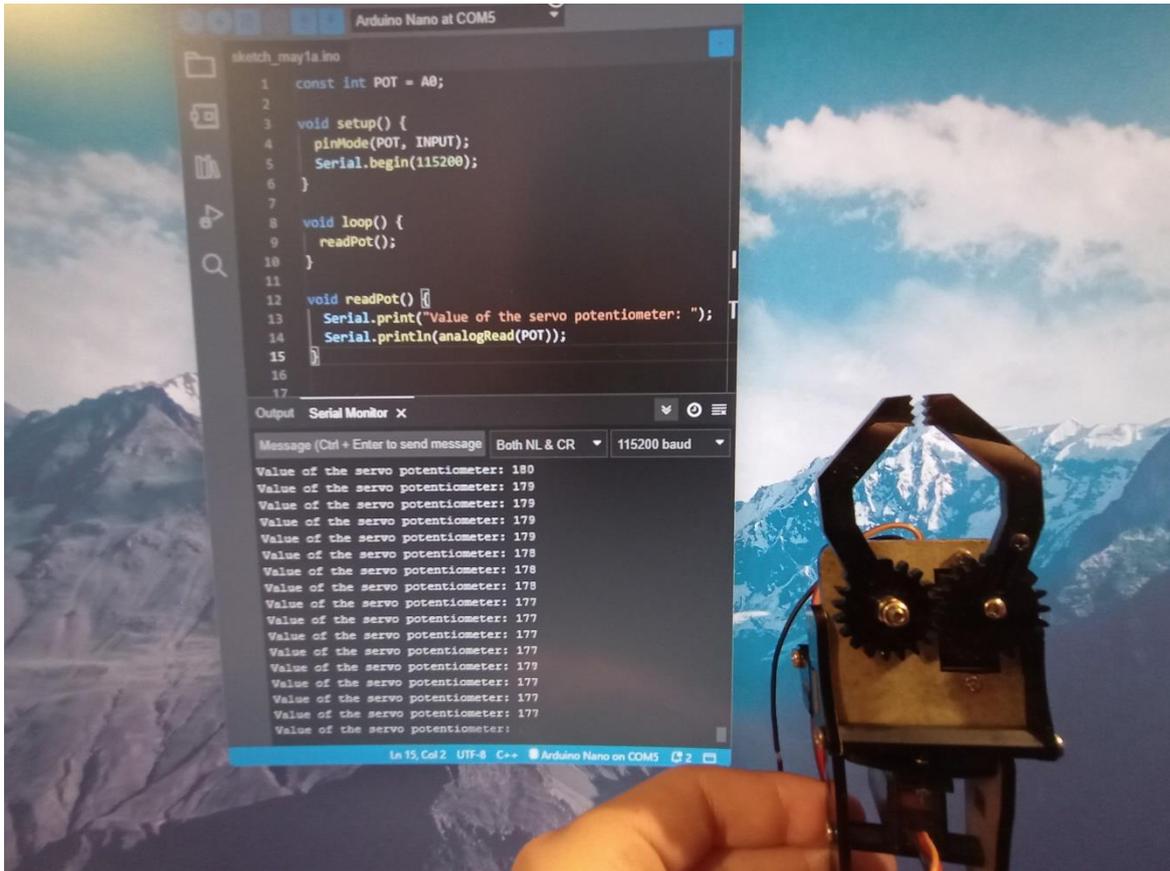


fig 20 & 21. Prototipo di circuito con Arduino Nano per la lettura del feedback dal potenziometro

Come è possibile vedere dalle seguenti immagini, la modifica ha effettivamente consentito di poter leggere la posizione reale del servomotore attraverso un valore analogico.



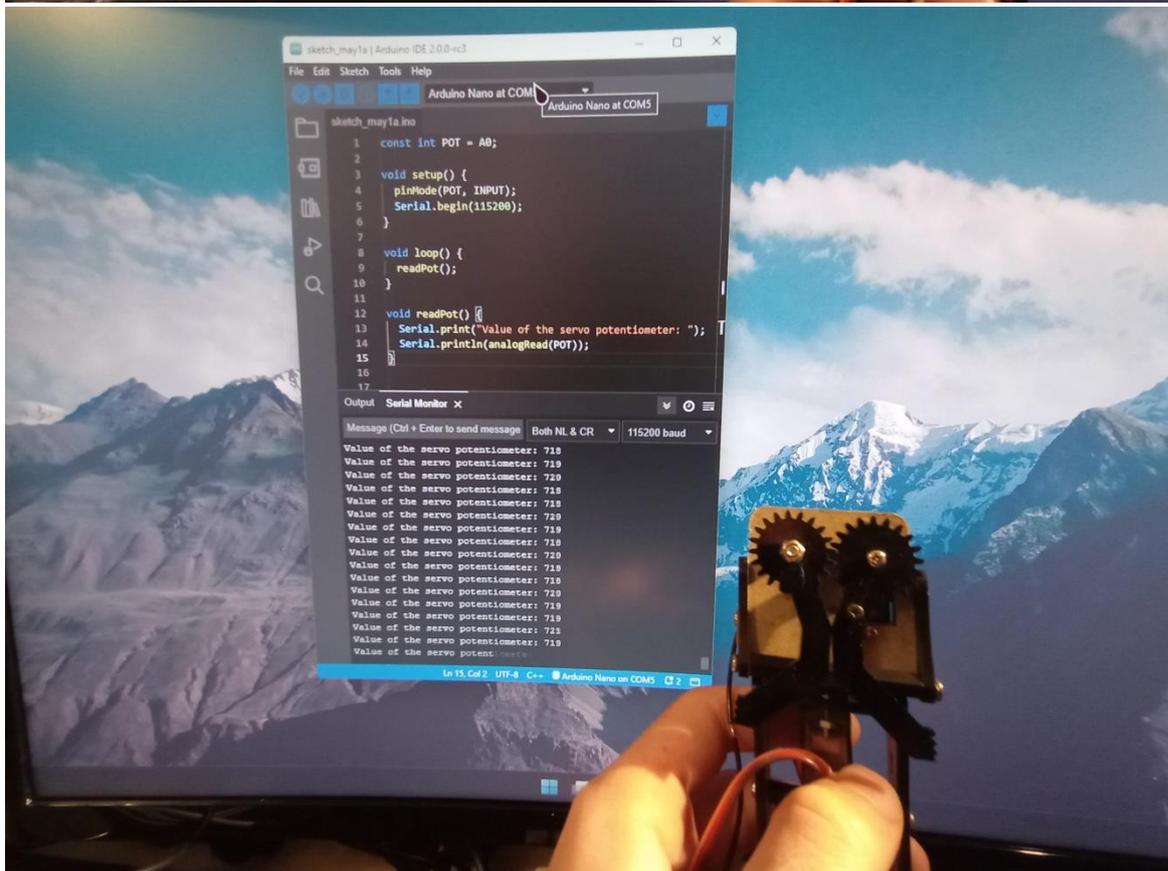
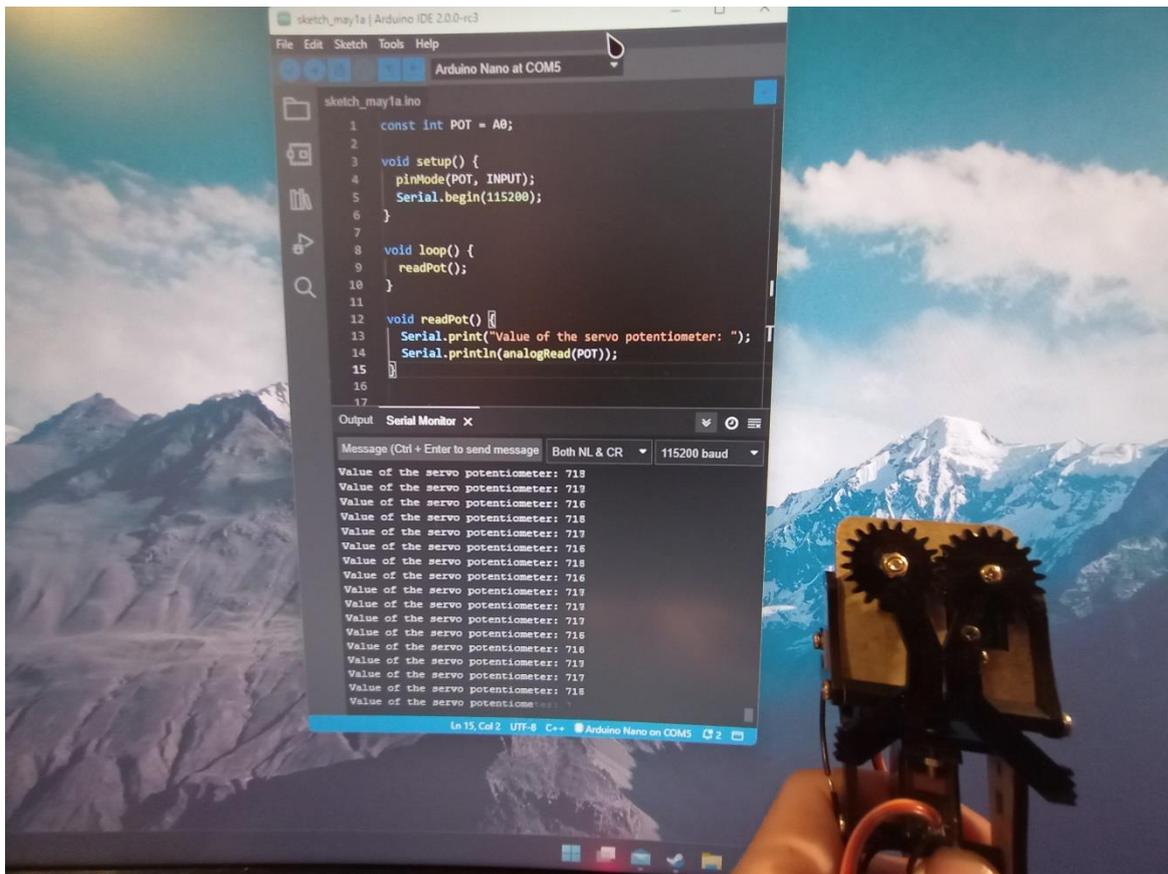


fig 22 ... 25. Valori letti dal potenziometro del servomotore attraverso codice Arduino

Come è possibile osservare dalle immagini precedenti, i valori letti dal potenziometro si aggirano nell'intorno [169, 719], principalmente per via delle braccia della pinza che limitano sia la chiusura che l'apertura massima. Per poter ottenere un valore in gradi equivalente al valore analogico ottenuto, che ci aspettiamo di essere realmente compreso tra 0 e 1023 per via della risoluzione a 10bit dell'ADC di Arduino Nano ed una tensione di alimentazione a 5v, è stato necessario innanzitutto capire ogni valore analogico ottenuto in feedback a quanti gradi di rotazione dell'albero corrispondessero.

È stato dunque scritto uno script di calibrazione per ottenere i valori analogici restituiti dal potenziometro rispetto ai gradi 0° e 180° del servomotore.

Lo script in questione è il seguente:

```
1. #include <Servo.h>
2.
3. Servo myservo;
4. const int feedback = A0;
5. unsigned int analog0deg, analog180deg;
6.
7. void setup() {
8.   Serial.begin(115200);
9.   pinMode(feedback, INPUT);
10.  calibration();
11. }
12.
13. void loop() {
14.  Serial.println(map(analogRead(feedback), analog0deg, analog180deg, 0, 180));
15.  delay(500);
16. }
17.
18. void calibration(){
19.  myservo.attach(9);
20.
21.  delay(500);
22.  myservo.write(0);
23.  delay(2000);
24.  analog0deg = analogRead(feedback);
25.
26.
27.  delay(500);
28.  myservo.write(180);
29.  delay(2000);
30.  analog180deg = analogRead(feedback);
31.
32.  Serial.println("Found 0° = " + String(analog0deg) + ", 180° = " + String(analog180deg));
33.
34.  myservo.detach();
35.
36. }
```

Attraverso questo script è stato possibile identificare dei valori analogici di riferimento per il segnale di feedback. La sua esecuzione ha riportato, mediamente, i seguenti risultati:

```
12:40:57.235 -> Found 0° = 641, 180° = 87
```

Ottenendo dunque valori compresi tra 641 e 87 nel movimento libero del servomotore, è stato poi utilizzata la funzione *map* nel loop di controllo per verificare in maniera sperimentale l'intervallo di valori in gradi relativo all'apertura *reale* della pinza, una volta rimontate le braccia, pari a -25° per l'apertura massima (a quanto pare il motore dell'albero è in grado di

girare per più di 180* effettivi, e si ottiene un corrispettivo valore dal potenziometro) e 152* per la chiusura completa.

Il servomotore scelto, AD002 MicroServo della Adept, è stato recuperato da un kit di sviluppo composto da un braccio meccanico dotato di cinque gradi di libertà ed una pinza all'estremità. In questo modo non solo è stato possibile utilizzare più servomotori differenti per sperimentare diverse tipologie di movimento controllate con EMG, ma è stato possibile anche riutilizzare la pinza per avere un esempio visivo e fisico di utilizzo del prototipo di protesi.



fig 26. Braccio robotico Adept

Infine, come microcontrollore del plant è stata utilizzata la scheda di sviluppo Nucleo-F446RE

2.2. Configurazione del plant

Il seguente circuito mostra il plant completo, costituito da sensori, attuatori e scheda di controllo, nonché dispositivi aggiuntivi che successivamente verranno descritti:

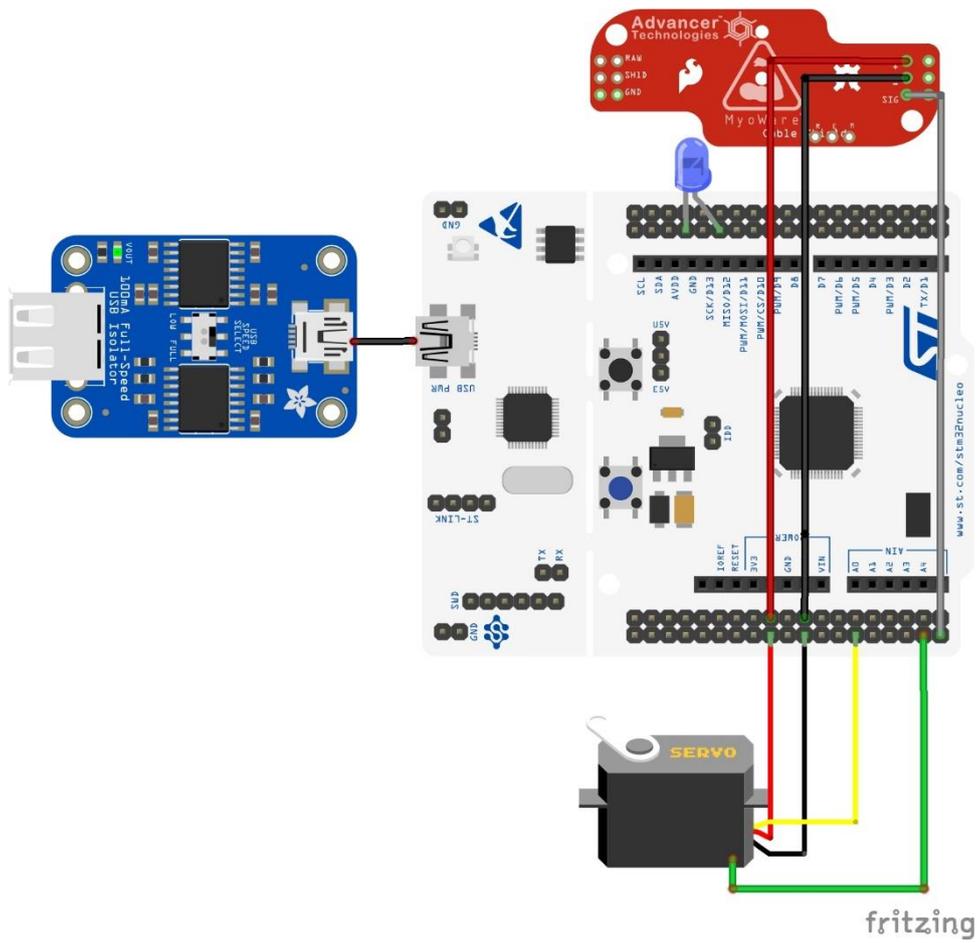


fig 27. Circuito dei collegamenti

d) Grid powered. **Warning: No isolation.**

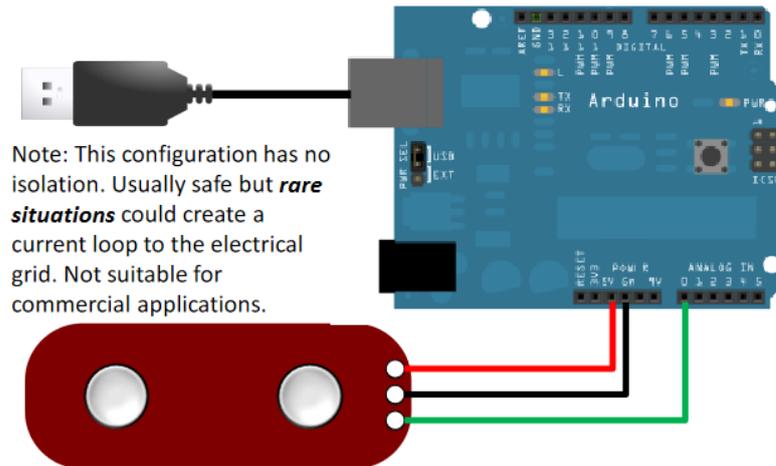


fig 30. Nota sul collegamento di Myoware alla scheda di sviluppo

Proprio per questo motivo è stato acquistato un modulo per l'isolamento dell'alimentazione USB, il 2107 di Adafruit



fig 31. Modulo Adafruit 2107 per l'isolamento

Infine, per ridurre il più possibile l'influenza del rumore nel segnale proveniente dal sensore Myoware, sono stati utilizzati dei cavi schermati per il collegamento del sensore alla scheda di controllo.

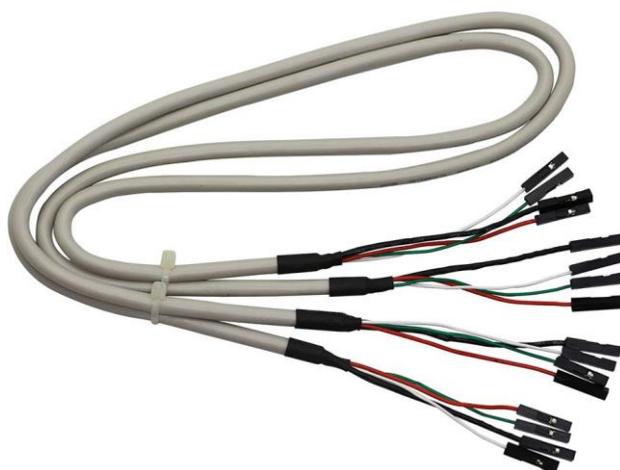


fig 32. Cavi schermati per il collegamento del sensore Myoware alla scheda di sviluppo

Di seguito, alcune specifiche elettriche delle varie componenti:

Parameter	Min	TYP	Max
Supply Voltage	+2.9V	+3.3V or +5V	+5.7V
Adjustable Gain Potentiometer	0.01 Ω	50 k Ω	100 k Ω
Output Signal Voltage EMG Envelope Raw EMG (centered about +Vs/2)	0V 0V	-- --	+Vs +Vs
Input Impedance	--	110 G Ω	--
Supply Current	--	9 mA	14 mA
Common Mode Rejection Ratio (CMRR)	--	110	--
Input Bias	--	1 pA	--

fig 33. Tabella delle specifiche elettriche del sensore Myoware

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 μ s
- Temperature range: 0 $^{\circ}$ C – 55 $^{\circ}$ C

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is all the way to the left. ms pulse) is all the way to the right, ""-90" (~1ms pulse) is all the way to the left.

Additional Specifications

Rotational Range: 180 $^{\circ}$
Pulse Cycle: ca. 20 ms
Pulse Width: 500-2400 μ s

fig 34. Specifiche elettriche e meccaniche del Servomotore

2.3. Descrizione del firmware

Il firmware del progetto è stato progettato e sviluppato utilizzando l'ambiente Stm32CubeIDE v1.0.9. Il codice sorgente è disponibile al seguente link, su GitHub:

<https://github.com/fenix-hub/srap>

Per prototipare un firmware funzionante, permettendo di sperimentare le basiche funzioni di lettura dal sensore Myoware e scrittura sul servomotore, è stato anzitutto configurato l'intero ambiente utilizzando i driver HAL (Hardware Abstraction Level) offerti da Stm32CubeIDE. In questa maniera è stato possibile preparare la base del progetto, prima di applicare ottimizzazioni passando ad una programmazione di più basso livello delle routine di controllo del plant.

Nella figura seguente è mostrata la configurazione effettuata attraverso l'editor visuale dei pinout per l'STM32F446RE.

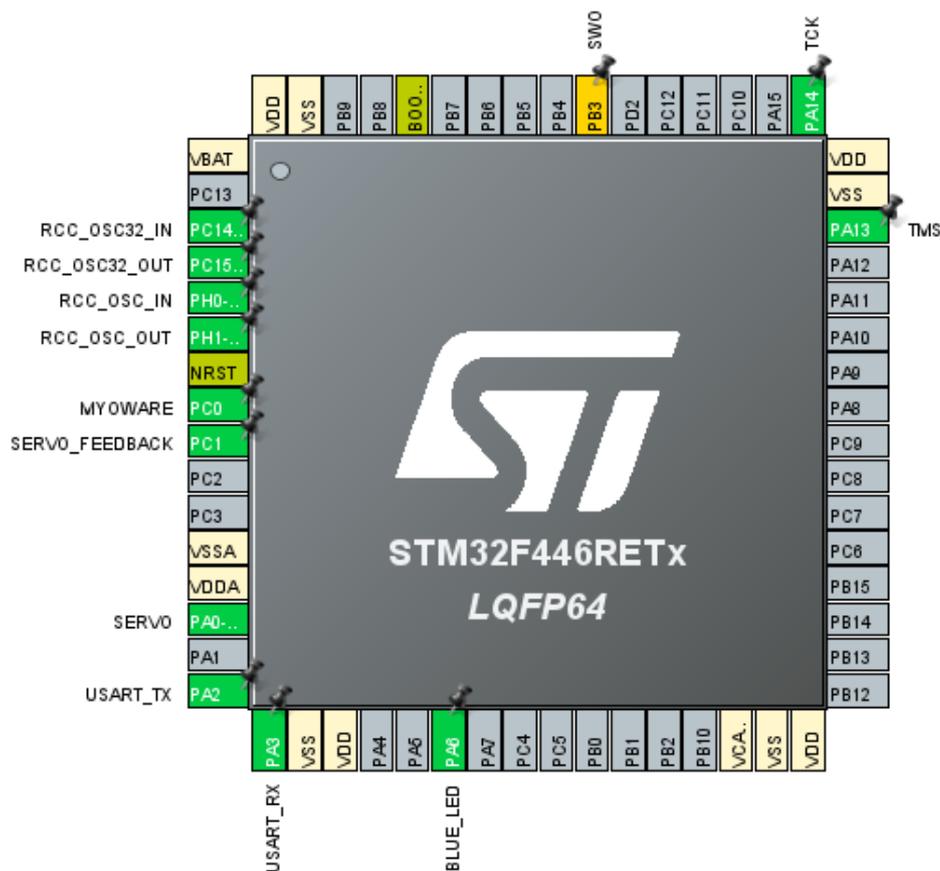


Fig 35. Configurazione dei PIN di STM32F446RE

Nella seguente tabella sono invece illustrate le funzionalità associate ad ogni PIN, rinominati con opportuni Label per renderli comprensibili:

Label	Pin / Funzione	Utilizzo
MYOWARE	PC0 (ADC1_IN10)	Lettura dal sensore Myoware attraverso ADC
SERVO	PA0 (TIM2_CH1)	Scrittura sull'attuatore AD002 servomotore attraverso PWM
SERVO_FEEDBACK	PC1 (ADC2_IN11)	Lettura dall'attuatore AD002 servomotore attraverso ADC
BLUE_LED	PA6 (GPIO_PIN6)	Diodo LED blu collegato per debug visivo attraverso GPIO
UART_TX	PA2 (USART2_TX)	Scrittura su porta seriale per lettura mediante monitor su COM3, Baud Rate 115200

2.3.1. Lettura dal Myoware

Per leggere il segnale prodotto dal sensore EMG, è stata utilizzata l'unità DMA (Direct Memory Access) per rendere efficiente la lettura di un buffer di dati analogici e la loro elaborazione, utilizzando il pin PC0 per la lettura mediante ADC (Convertitore Analogico/Digitale).

La potenza del segnale EMG grezzo è molto debole (~ 2 mV) e rumorosa. Per migliorare la qualità del segnale, questo deve essere amplificato, rettificato e filtrato accuratamente. Fortunatamente, il sensore Myoware è stato progettato per essere utilizzato direttamente con un microcontrollore. Quindi, l'uscita principale del sensore Myoware non è un segnale EMG grezzo ma piuttosto un segnale amplificato, segnale rettificato e integrato. Il segnale di uscita dal sensore funziona bene con un convertitore analogico-digitale (ADC) del microcontrollore.

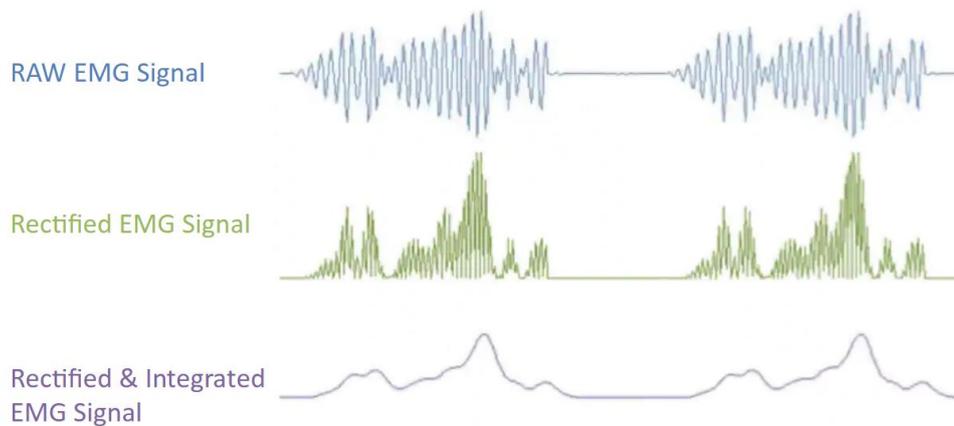


fig 36. Grafico esemplificativo dei segnali grezzo, rettificato, rettificato e integrato da specifiche Myoware

L'ADC è stato configurato nella seguente maniera:

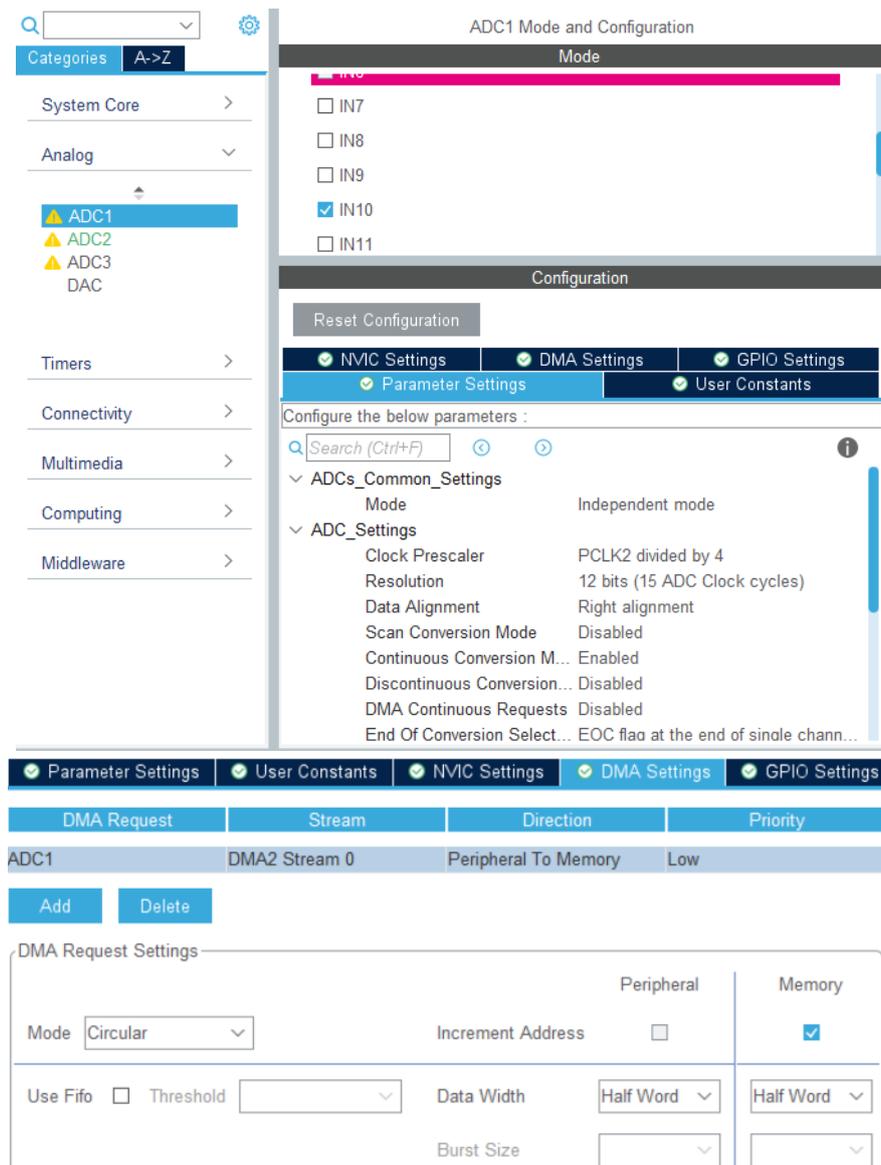


Fig 37. Configurazione dell'ADC1 per la lettura del sensore Myoware attraverso PC0 mediante DMA

Come è possibile notare, il DMA è stato configurato in modalità circolare per ottenere una continua scrittura circolare all'interno del buffer di riferimento, ma l'effettivo accesso mediante DMA alla periferica è stata configurata per avvenire attraverso specifica richiesta da parte del processore, e non attraverso una richiesta automatica continua da parte di HAL.

In questa maniera viene effettuata una rapida lettura di una serie di campioni attraverso l'ADC in maniera intermittente e controllata, per poi gestire all'interno del loop di controllo logiche successive per l'elaborazione dei dati e l'esecuzione delle logiche sul plant.

Attraverso prove sperimentali, infatti è stato osservato che i valori trasmessi dal sensore Myoware v1, nonostante le precauzioni precedentemente illustrate, non sono lineari, ma assumono sensibili variazioni intorno ad un punto fisso di riferimento.

Ad esempio, è stato osservato che il valore ottenuto dalla lettura del sensore con il muscolo del braccio a riposo si aggira intorno al valore 210 (spesso variabile tra 200 e 230), mentre la contrazione del muscolo porta ad un valore di picco fino a 2100 per poi stabilizzarsi intorno al

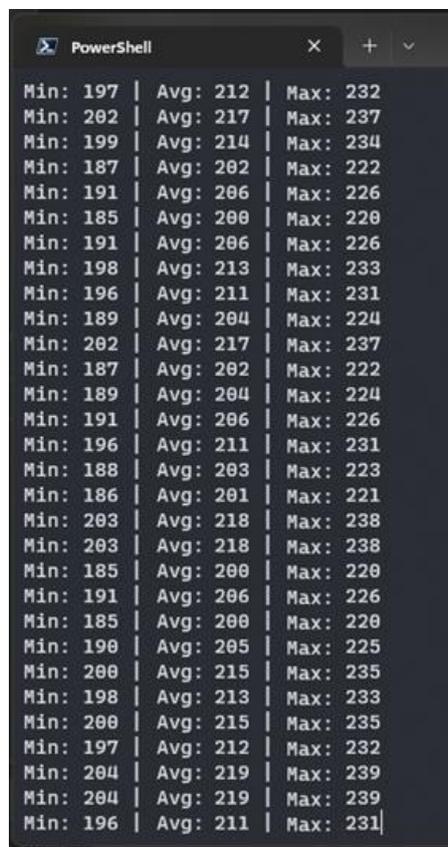
valore 1300 (variabile tra 900 e 1500).

Questi valori, ottenibili solo sperimentalmente poichè non illustrati dalle specifiche, possono assumere grosse variazioni a causa delle seguenti supposizioni:

- Essendo gli elettrodi passivi (ovvero si comportano come delle vere e proprie antenne per qualsiasi segnale elettrico, poichè per la lettura dell'impulso trasmesso dai neurotrasmettitori non generano nessun segnale attivo) sono soggetti inevitabilmente a qualsiasi rumore magnetico esterno;
- Essendo il sensore estremamente sensibile (poichè il segnale deve essere amplificato) non solo i minimi rumori di fondo vengono amplificati, ma soprattutto qualsiasi movimento fisico del sensore può comportare alla generazione di rumore addizionale (poichè il braccio, soprattutto nella fase di contrazione, è inevitabilmente soggetto a tremolio)

Per avere un'idea più precisa dei valori assumibili come "soglie di riferimento", sono state scritte delle basilari funzioni *min/max* all'interno del firmware, che estraggono il valore minimo ed il valore massimo tra tutti i campioni presenti in un buffer dopo una lettura.

La seguente immagine mostra un esempio:



```
PowerShell
Min: 197 | Avg: 212 | Max: 232
Min: 202 | Avg: 217 | Max: 237
Min: 199 | Avg: 214 | Max: 234
Min: 187 | Avg: 202 | Max: 222
Min: 191 | Avg: 206 | Max: 226
Min: 185 | Avg: 200 | Max: 220
Min: 191 | Avg: 206 | Max: 226
Min: 198 | Avg: 213 | Max: 233
Min: 196 | Avg: 211 | Max: 231
Min: 189 | Avg: 204 | Max: 224
Min: 202 | Avg: 217 | Max: 237
Min: 187 | Avg: 202 | Max: 222
Min: 189 | Avg: 204 | Max: 224
Min: 191 | Avg: 206 | Max: 226
Min: 196 | Avg: 211 | Max: 231
Min: 188 | Avg: 203 | Max: 223
Min: 186 | Avg: 201 | Max: 221
Min: 203 | Avg: 218 | Max: 238
Min: 203 | Avg: 218 | Max: 238
Min: 185 | Avg: 200 | Max: 220
Min: 191 | Avg: 206 | Max: 226
Min: 185 | Avg: 200 | Max: 220
Min: 190 | Avg: 205 | Max: 225
Min: 200 | Avg: 215 | Max: 235
Min: 198 | Avg: 213 | Max: 233
Min: 200 | Avg: 215 | Max: 235
Min: 197 | Avg: 212 | Max: 232
Min: 204 | Avg: 219 | Max: 239
Min: 204 | Avg: 219 | Max: 239
Min: 196 | Avg: 211 | Max: 231
```

Fig 38. Valori di min/max per un buffer di 2555 campioni estratti dall'ADC con muscolo a riposo

Proprio per questo motivo, la lettura di campioni dall'ADC attraverso un buffer alimentato dal DMA è stata sfruttata per calcolare un valore unico di EMG ad ogni ciclo iterativo, applicando differenti logiche di filtraggio.

Filters.c

```
1. float relative_average(const float* samples, size_t length, uint8_t moving_point) {
2.     float sum = 0.0f;
3.     uint16_t half_interval = (moving_point - 1) / 2;
4.
5.     for (uint16_t i = half_interval; i < length - half_interval; ++i) {
6.         sum += samples[i];
7.     }
8.     return sum / (length - moving_point - 1) ;
9. }
10.
11. void moving_average_filter(const uint16_t* samples, float* filtered, size_t length,
12.     uint16_t moving_point) {
13.     uint16_t half_interval = (moving_point - 1) / 2;
14.
15.     for (uint16_t i = half_interval; i < length - half_interval; i++) {
16.         filtered[i] = 0.0f;
17.         for (int16_t j = -half_interval; j < half_interval; j++) {
18.             filtered[i] = filtered[i] + samples[i + j];
19.         }
20.         filtered[i] = (float) filtered[i] / moving_point;
21.     }
22. }
```

In particolare, dapprima viene applicato un filtro a media mobile sul set di campioni nel buffer, utilizzando un moving point pari a 25, ottenendo un set di nuovi campioni che ci aspettiamo essere più lineare rispetto al precedente, ma con meno campioni disponibili.

Successivamente per ottenere un valore “istantaneo” di EMG utilizzabile come valore di riferimento per l’applicazione del controllo sul plant, è stato calcolato il valore medio del nuovo set di campioni.

Infine, il valore medio è stato diviso per 10 in maniera tale da ottenere un valore di ordine più piccolo, così da poter gestire più facilmente il valore di attivazione del servomotore. Attraverso questa breve routine di filtraggio è stato possibile ottenere valori relativamente più stabili rispetto ai valori che si otterrebbero considerando istantaneamente un unico campione.

Il motivo per cui il servomotore viene mosso con un *valore di attivazione*, e non attraverso una mappatura continua tra un valore di EMG ed un angolo, deriva dalla natura stessa del segnale letto dal Myoware, e dalla non possibilità di applicare filtri più efficienti al momento dello sviluppo del progetto.

Dunque, il valore definitivamente ottenuto dal sensore Myoware, a valle di filtraggio ed adattamento, risulta essere circa 21.80 per quando il muscolo dell’avambraccio posto in analisi è totalmente rilassato, e circa 150.50 quando il muscolo è stabile in fase di contrazione.

Tali valori sono stati inizialmente considerati come soglie di riferimento assolute per il calcolo del valore di attivazione, posto a 80.0.

Successivamente tali valori verranno calcolati in maniera procedurale attraverso l’implementazione di un ciclo di calibrazione del sistema, con conseguente calcolo del valore di attivazione stesso.

Di seguito, un riassunto delle porzioni di codice inerenti alla parte di lettura del sensore Myoware.

Main.c

```
1. void emg_read_loop() {
2.     HAL_ADC_Start_DMA(&hadc1, (uint32_t*) adc1_buf, ADC1_BUF_LEN);
3.     while(adc1_conv_complete == 0) {
4.
5.     }
6.     adc1_conv_complete = 0;
7. }

1. float normalize_emg(uint16_t* buffer) {
2.     float myoware_filtered[ADC1_BUF_LEN] = { 0.0f };
3.     moving_average_filter(buffer, myoware_filtered, ADC1_BUF_LEN, MOVING_POINT);
4.     float myoware_avg = relative_average(myoware_filtered, ADC1_BUF_LEN,
MOVING_POINT) / 10;
5.
6.     sprintf(
7.         msg,
8.         "Myoware = %.2f, min = %hu, max = %hu \r\n",
9.         myoware_avg,
10.        min(buffer, ADC1_BUF_LEN),
11.        max(buffer, ADC1_BUF_LEN)
12.    );
13.    console_log(msg);
14.
15.    return myoware_avg;
16. }
```

2.3.2. Controllo Servomotore

Per interagire con il servomotore AD002 sono stati scelti i pin PA0 per la lettura e PC1 per la lettura.

Per quanto concerne la scrittura, il driver dell'AD002 come qualsiasi altro servomotore necessita di essere azionato mediante PWM.

Tra le varie funzionalità implementabili attraverso i timer sicuramente una delle più importanti per i timer General Purpose è quella di poter generare segnali PWM.

I segnali PWM (pulse-width modulation) sono una alternativa alla generazione di valori analogici di tensione, a partire da valori digitali, che consistono nella generazione di un'onda quadrata ad una specifica frequenza, e con un duty cycle prestabilito. È dunque una tecnica attraverso cui, mediante un segnale digitale, è possibile generare un'onda quadra contenente una informazione variabile. Per *duty cycle* si intende la quota parte di segnale in cui lo stesso è attivo rispetto al suo intero periodo, calcolabile dunque come:

$$DC = \frac{T_{on}}{T_{tot} \cdot 100}$$

Nel nostro caso, i timer possono essere usati per generare delle PWM con un duty cycle configurabile e il cui valore di tensione oscilla tra 0 e il valore di tensione con cui alimentiamo il servomotore, ovvero 5V.

Ogni timer ha una serie di canali attraverso i quali la PWM può essere generata, e ciascun canale è collegato ad un pin elettrico. Una volta configurata la frequenza del timer, che sarà la frequenza della PWM che vogliamo generare ed il cui valore dipende dal valore assegnato al registro TIMx_ARR (Auto-reload Register), possiamo scegliere il duty cycle del nostro segnale cambiando il valore del "pulse" nel registro TIMx_CCR (Capture/Compare Register). Come facciamo però a sapere quale valore settare nel registro CCR per muovere il servomotore secondo i valori dati dalle specifiche, e soprattutto quale valore assegnare all'ARR ?

Innanzitutto sappiamo che per definizione un timer è un contatore con una frequenza di conto, che è una frazione dell'orologio sorgente.

La componente hardware che frazione l'orologio sorgente in una sotto-frequenza è il prescaler. In base al valore assegnato al registro TIMx_PSC si otterrà una frequenza che è:

$$f = \frac{f_{clk}}{(PSC + 1)}$$

Dopo il prescaler abbiamo un timer counter, ovvero un registro (TIMx_CNT) in cui viene incrementato il valore per contare alla frequenza del prescaler. Il valore nel CNT verrà resettato in base al threshold scelto, il cui valore è nel registro ARR.

Ad ogni conto del timer counter fino al valore ARR verrà sollevato un interrupt e si resetterà il CNT, con un periodo di conto (*counting period*) pari a:

$$CP = \frac{(ARR + 1)}{f_{clk.cnt}}$$

Si può dunque osservare che per generare un'onda con un periodo f_c frazione di f_{clk} (entrambi noti, il primo dalle specifiche del servomotore, l'altro dalla configurazione del clock) sarà necessario stabilire il valore del prescaler e del timecounter.

$$\frac{f_{clk}}{f_c} = (ARR + 1) \cdot (PSC + 1)$$

Nel nostro caso specifico, noto $f_{clk} = 84MHz$ e avendo un pulse cycle di 20ms per il servomotore, dunque $f_c = 1/20ms = 50Hz$, si ottiene:

$$(168 * 10^4 = (ARR + 1) \cdot (PSC + 1))$$

Ricordando che PSC è un registro a 16bit, dunque con valore massimo accettabile 65'535, e che tanto più e basso il prescaler, tanto più sarà alta la risoluzione che si traduce in più bassi valori per il calcolo del duty cycle (ma allo stesso tempo maggiore sarà il consumo di energia); è facile trovare che il più piccolo valore accettabile di PSC+1 tale che sia massimo il valore a 32bit di (ARR+1) è 1.

$$168 * 10^4 = 1680000_{(ARR+1)} \cdot 1_{(PSC+1)}$$

Di seguito, vengono illustrate le configurazioni del timer per la generazione della PWM:

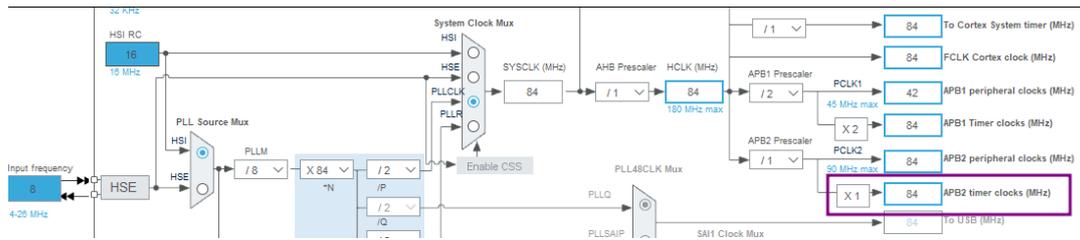


fig 36. Configurazione del Clock per generazione del segnale con PWM

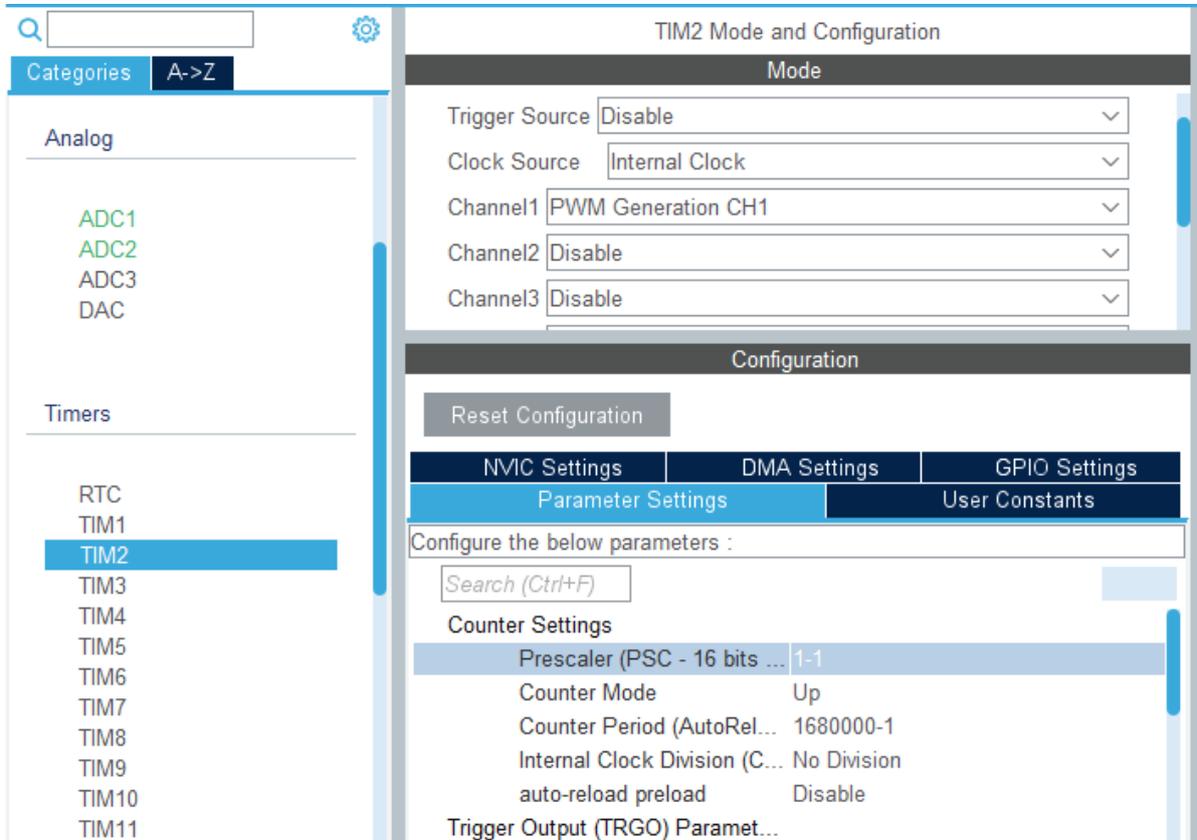


Fig 37. Configurazione del Timer 2 per la generazione del segnale con PWM

Tornando alla generazione della PWM, capiamo ora come calcolare il valore di duty cycle da assegnare al registro CCR.

Il duty cycle sappiamo essere una frazione del periodo dell'onda.

Sapendo che il periodo dipende da ARR, possiamo immaginare che anche il duty cycle dipenderà da esso. Infatti si dimostra facilmente che:

$$DC_{\%} = \frac{CCR}{(ARR + 1)}$$

Dove CCR è il valore dell'impulso che può variare tra 0 e ARR. Fin tanto che $CCR < ARR$ (in up-counting), il canale rimarrà attivo.

Si ottiene dunque che:

$$CCR = \frac{DC_n}{100} \cdot (ARR + 1)$$

Dove DCn non è altro che un numero intero, da 0 a 100, che rappresenta la quota parte percentuale di onda quadra che si vuole mantenere attiva.

In definitiva per trovare dei valori programmatici da assegnare al registro CCR in codice, non è bastato altro che calcolare a quanta percentuale di periodo totale corrispondessero i valori forniti dalle specifiche del servomotore, e convertirli in valori di DCn.

Come fornito dalle specifiche, si può ricavare la seguente tabella:

Gradi di rotazione	Millisecondi	DCn % (di 20ms)
-90*	1ms	5%
0*	1.5ms	7,5%
90*	2ms	10%

Attraverso i dati sperimentali è stato possibile verificare che il servomotore non rispetta esattamente le specifiche, ma presenta un offset costante di disallineamento rispetto ai valori attesi di rotazione. È stato possibile correggere tale disallineamento quantificando l'errore in termini percentuali all'interno del codice, in maniera sempre sperimentale, con un valore pari a +0.8%.

Inoltre, è stato possibile verificare che il valore massimo di apertura (-90*) coincide con un duty cycle dell'1%, mentre il valore di chiusura massimo (90*) coincide con un valore di duty cycle pari all'11%.

Per il controllo del servomotore è stato scritto un driver personalizzato, consistente in un file header ed un file .c, che consente di trattare una qualsiasi periferica servomotore come un oggetto C (seguendo la metodologia di programmazione OOP).

Servo.h

```
1. #ifndef INC_SERVO_H_
2. #define INC_SERVO_H_
3.
4. #include "stm32f4xx_hal.h"
5. #include <math.h>
6.
7. // The Number OF Servo Motors To Be Used In The Project
8. #define SERVO_NUM 1
9.
10. typedef struct Servo
11. {
12.     TIM_HandleTypeDef* TIM_HandleInstance;
13.     uint16_t          TIM_Channel;
14.     float             min_angle;
15.     float             max_angle;
16.     float             min_dc;
17.     float             max_dc;
18.     float             dc_offset;
19. } Servo;
20.
21. /*-----[ Prototypes For All Functions ]-----*/
22.
23. Servo new_servo(TIM_HandleTypeDef* TIM_HandleInstance, uint16_t TIM_Channel, float
    min_angle, float max_angle, float min_dc, float max_dc, float dc_offset);
24. void start_servo(Servo servo);
25. void stop_servo(Servo servo);
```

```

26. void servo_write_deg(Servo servo, float deg);
27. void servo_write_dc(Servo servo, float dc);
28. void servo_write_ccr(Servo servo, uint32_t CCR);
29.
30. #endif /* INC_SERVO_H */

```

Servo.c

```

1. #include <Servo.h>
2.
3. Servo new_servo(TIM_HandleTypeDef* TIM_HandleInstance, uint16_t TIM_Channel, float
   min_angle, float max_angle, float min_dc, float max_dc, float dc_offset) {
4.     Servo s;
5.     s.TIM_HandleInstance = TIM_HandleInstance;
6.     s.TIM_Channel = TIM_Channel;
7.     s.max_angle = max_angle;
8.     s.min_angle = min_angle;
9.     s.max_dc = max_dc;
10.    s.min_dc = min_dc;
11.    s.dc_offset = dc_offset;
12.    return s;
13. }
14.
15. void start_servo(Servo servo) {
16.     HAL_TIM_PWM_Start(servo.TIM_HandleInstance, servo.TIM_Channel);
17. }
18.
19. void stop_servo(Servo servo) {
20.     HAL_TIM_PWM_Stop(servo.TIM_HandleInstance, servo.TIM_Channel);
21.     servo_write_dc(servo, servo.max_dc);
22. }
23.
24. void servo_write_deg(Servo servo, float deg) {
25.     float dc = (deg - servo.min_angle) * (servo.max_dc - servo.min_dc) / (servo.max
       _angle - servo.min_angle) + servo.min_dc;
26.     servo_write_dc(servo, dc);
27. }
28.
29. void servo_write_dc(Servo servo, float dc) {
30.     uint32_t ccr = (uint32_t) roundf(((float) (servo.TIM_HandleInstance->Instance-
       >ARR + 1.0) / 100.0) * (dc + servo.dc_offset) );
31.     servo_write_ccr(servo, ccr);
32. }
33.
34. void servo_write_ccr(Servo servo, uint32_t CCR) {
35.     __HAL_TIM_SET_COMPARE(servo.TIM_HandleInstance, servo.TIM_Channel, CCR);
36.     servo.TIM_HandleInstance->Instance->EGR = TIM_EGR_UG;
37. }

```

Come è possibile notare dal file di codice, il metodo `servo_write_deg` è utilizzato per muovere il servomotore fornendo come argomento del metodo direttamente un valore in gradi. Questo metodo è risultato molto comodo, poichè in questo modo è stato possibile muovere e fare varie sperimentazioni del servomotore utilizzando valori molto più vicini al funzionamento osservabile del motore stesso.

Tale valore è calcolato attraverso una semplice funzione “map”, che calcola il valore in *duty cycle* del servomotore a partire da un valore in gradi, in base ai valori con cui è stato configurato il servomotore stesso.

Dopodichè il metodo chiama internamente la funzione `servo_write_dc`, utilizzando la formula precedentemente illustrata per il calcolo del CCR a partire dal valore DC.

È infine possibile osservare che per settare l'effettivo valore nel registro CCR viene dapprima chiamata la macro `_HAL_TIM_SET_COMPARE`, dopodichè viene settato il valore del registro EGR a `TIM_EGR_UG`.

Come definito nelle specifiche STM32F446xx:

“An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register”.

Infine, in merito al controllo del servomotore sulla base del segnale EMG, il valore filtrato e mediato proveniente dal campionamento dell'ADC viene convertito in un valore in gradi centigradi per la chiusura/apertura della pinza meccanica.

```
1. float emg_to_angle(float emg) {
2.     float target_angle = min_angle;
3.     if (emg > TRESHOLD) {
4.         target_angle = max_angle;
5.     }
6.     return target_angle;
7. }
```

Come affermato nel capitolo precedente, al momento dello sviluppo del progetto non è stato possibile verificare la fattibilità di un controllo continuo del sensore (ovvero di una mappatura continua del valore di EMG estratto a valore in angolo) a causa del malfunzionamento del sensore Myoware e della sua non reperibilità. Il servomotore dunque, precedentemente all'integrazione di un controllore PID, è stato mosso direttamente con il valore estratto dalla funzione di conversione emg ad angolo.

```
1. /* USER CODE BEGIN PV */
2. Servo servo;
3. Float emg_value = 0.0f;
4. Float setpoint_value = 0.0f;
5.
6. /* USER CODE END PV */
7.
8. [...]
9.
10. int main(void) {
11.
12.     servo = new_servo(&htim2, TIM_CHANNEL_1, 0.0, 180.0, 1.0, 11.0, 0.8);
13.     start_servo(&servo);
14.
15.     while (1) {
16.         [...]
17.
18.         setpoint_value = emg_to_angle(emg_value);
19.         servo_write_deg(servo, setpoint_value);
20.
21.         // LIGHT the LED for DEBUG
22.         light_led(emg_value > TRESHOLD);
23.
24.         [...]
25.     }
26.
27. }
```

2.3.3. Feedback del Servomotore

Attraverso la modifica effettuata sul servomotore, è possibile leggere per qualsiasi istante di tempo il valore reale del servomotore e confrontarlo rispetto al valore utilizzato per muovere il servomotore da firmware, ovvero la variabile *setpoint*.

Tale caratteristica è stata molto importante ai fini dello sviluppo del loop di controllo per questo caso di studio, per l'introduzione di due importantissime funzionalità:

1. Possibilità di calibrare e configurare la protesi robotica
2. Verifica della posizione reale dell'albero rispetto al valore atteso (setpoint) per il rilevamento di ostacoli e limiti di funzionamento del servomotore

In merito alla prima funzionalità, bisogna ricordare che il meccanismo di funzionamento di una protesi robotica si basa interamente sull'utente che ne fa utilizzo: sia i valori analogici rilevati attraverso EMG, sia i limiti di apertura e chiusura della pinza meccanica, sono stati assunti come valori *assoluti* per qualsiasi applicazione, ma una protesi assoluta è tale solo se adattiva, ovvero se consente di adattarsi alle specifiche caratteristiche dell'utilizzatore.

Un utente infatti, potrebbe generare diversi valori di EMG durante la fase di rilassamento e contrazione del muscolo. In questo caso, il sistema deve essere in grado di configurare il valore di *threshold* per l'attivazione della pinza meccanica dopo la calibrazione da parte dell'utente.

Inoltre la stessa chiusura della pinza deve poter essere configurabile: non è detto che il servomotore impiegato per la protesi sia sempre lo stesso e abbia gli stessi valori di funzionamento. Anche in questo caso è dunque necessario effettuare una calibrazione dell'attuatore, ed utilizzare poi i valori rilevati per la configurazione.

Proprio per questo motivo, sono state introdotte due procedure, chiamate *calibrazione* e *configurazione* per distinguere quelle funzionalità rispettivamente una automatica, l'altra interattiva, che consentono di personalizzare il funzionamento della protesi.

In particolare, il meccanismo relativo alla configurazione del servomotore, si basa interamente sul segnale di feedback ottenute dal potenziometro.

In merito alla seconda funzionalità, anche questa richiede l'utilizzo del feedback.

Come illustrato precedentemente, i servomotori hanno l'utilità di poter essere controllati con precisione rispetto ai normali motori DC, avendo la possibilità di definire esattamente quale sia l'angolo verso il quale il rotore deve girare e fermarsi.

Nel caso di controllo fino ad ora illustrato, ovvero quello relativo alla rotazione di un servomotore attraverso la definizione di precisi step di movimento attraverso un threshold di EMG, applicare il segnale di feedback per il movimento (ad esempio mediante un controllore Proporzionale-Integrativo-Derivativo) sarebbe contro intuitivo: i servomotori sono appositamente costruiti e configurati per poter definire esattamente dove l'albero si deve trovare ad un determinato istante di tempo, e si può dire che un controllore PID è praticamente già presente nel driver del servomotore stesso.

Al contrario, l'utilizzo del feedback proveniente dal potenziometro sarebbe invece molto più utile a stabilizzare il movimento del servomotore nei seguenti casi:

- Presenza di ostacoli nel raggio di rotazione del servomotore, che può comportare non solo il non raggiungimento del setpoint, ma eventualmente affaticamento ed usura degli ingranaggi del rotore, o se la coppia esercitata dall'albero è troppo forte, rottura

dell'ostacolo (che non sempre è da considerarsi come un ostacolo "negativo", ma può essere anche un oggetto che si desidera afferrare o spostare)

- Rilevazione di posizioni non accettabili (secondo i limiti imposti dalla fase di calibrazione e configurazione) della pinza rispetto i valori di riferimento, come eventuale indicatore per malfunzionamento o manomissione del dispositivo.

Per questa funzionalità, a causa del malfunzionamento del sensore Myoware e della sua non reperibilità, non è stato possibile investigare un'effettiva implementazione.

Per quanto concerne la prima funzionalità, dapprima è stata scritta la procedura di *calibrazione*. In particolare, tale calibrazione si suddivide in due sotto-procedure: calibrazione del sensore e calibrazione dell'attuatore.

Per quanto concerne la calibrazione del sensore, non è stato possibile codificare tale procedura a causa della indisposizione del sensore Myoware, e di conseguenza la mancanza di poter valutare che valori di

Infine per quanto concerne la parte interessata dall'utilizzo del feedback del servomotore, troviamo sia la sotto-procedura di calibrazione, sia la procedura di configurazione.

Per la procedura di calibrazione automatica del servomotore, il firmware non fa altro che muovere ad intervalli regolari, nel range di movimenti ammessi dal servomotore, l'angolo minimo e quello massimo dell'albero per ottenere il corrispettivo valore analogico dal potenziometro.

```
1. void calibrate_actuator(Servo servo, uint16_t *min_calibration, uint16_t *max_calib
   ration, uint16_t *mid_calibration) {
2.     start_servo(servo);
3.
4.     for (int i = 0.0; i < 180.0 ; i+=15.0) {
5.         servo_write_deg(servo, i);
6.         HAL_Delay(100);
7.     }
8.
9.     servo_write_deg(servo, 0.0);
10.    HAL_Delay(1000);
11.    *min_calibration = read_feedback();
12.
13.    servo_write_deg(servo, 180.0);
14.    HAL_Delay(1000);
15.    *max_calibration = read_feedback();
16.
17.    servo_write_deg(servo, 90.0);
18.    HAL_Delay(1000);
19.    *mid_calibration = read_feedback();
20.
21.    HAL_ADC_Stop(&hadc2);
22.
23.    // debug
24.    sprintf(
25.        msg,
26.        "Calibration completed: 0* = %u; 90* = %u; 180* = %u \r\n",
27.        *min_calibration,
28.        *mid_calibration,
29.        *max_calibration
30.    );
31.    console_log(msg);
32. }
```

Una volta ottenuti questi valori analogici, che rappresentano la reale estensione di movimento del servomotore, viene iniziata la fase interattiva di configurazione, durante la quale l'utente può liberamente stabilire di quanto la pinza si deve aprire (valore massimo di apertura) e di quanto si deve chiudere (massimo valore di chiusura).

```
1. void configure_actuator(Servo servo, uint16_t min_calibration, uint16_t max_calibra
   tion, uint16_t mid_calibration) {
2.     stop_servo(servo);
3.     uint16_t t_feedback, min_feedback, max_feedback = mid_calibration;
4.
5.     // configuration 1
6.     while (system_state == CONFIGURATION_MIN) {
7.         t_feedback = read_feedback();
8.         if (t_feedback < min_calibration && t_feedback >= mid_calibration) {
9.             min_feedback = t_feedback;
10.        }
11.    }
12.
13.    start_servo(servo);
14.    servo_write_deg(90.0);
15.    stop_servo(servo);
16.
17.    // configuration 2
18.    while (system_state == CONFIGURATION_MAX) {
19.        t_feedback = read_feedback();
20.        if (t_feedback > max_calibration && t_feedback <= mid_calibration) {
21.            max_feedback = t_feedback;
22.        }
23.    }
24.
25.    HAL_ADC_Stop(&hadc2);
26.
27.    min_angle = map(min_feedback, (float) min_calibration, (float) max_calibration,
   0.0, 180.0);
28.    max_angle = map(max_feedback, (float) min_calibration, (float) max_calibration,
   0.0, 180.0);
29.
30.    // debug
31.    sprintf(
32.        msg,
33.        "Configuration completed: min angle = %.2f; max angle = %.2f \r\n",
34.        min_angle, max_angle
35.    );
36.    console_log(msg);
37. }
```

Affinché l'utente possa accettare la configurazione, e dunque passare nel loop di controllo, è stato predisposto lo USER_BUTTON dell'stm32 F446RE per poter cambiare, attraverso il toggle, il valore della variabile "system_state" a "CONTROL", uscendo dunque dal loop.

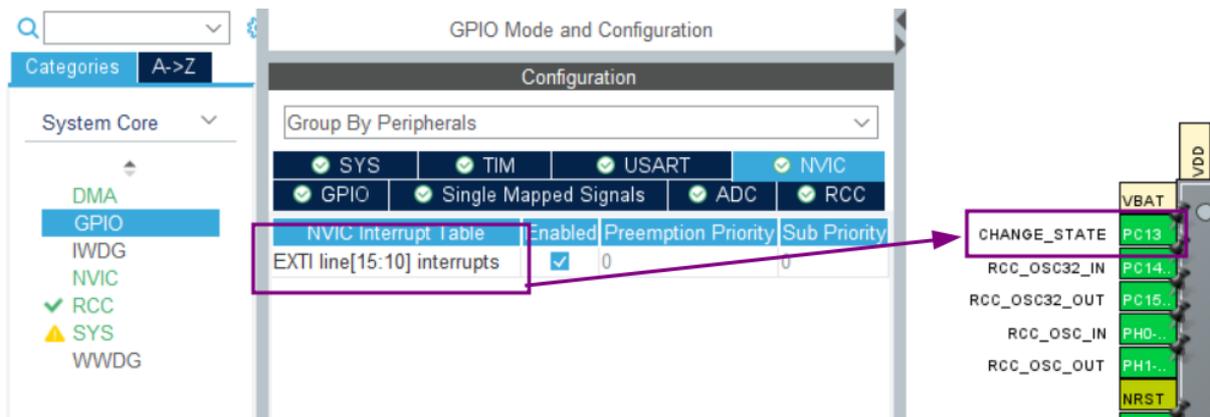


fig 38. Configurazione dello User Button come interrupt per il Change State

Dunque, i metodi *calibrate_actuator* e *configure_actuator* sono stati inseriti all'interno del main del firmware.

2.3.4. Routine del firmware

A questo punto, il progetto risulta essere completo.

Di seguito viene illustrata una schematica macchina a stati del firmware.

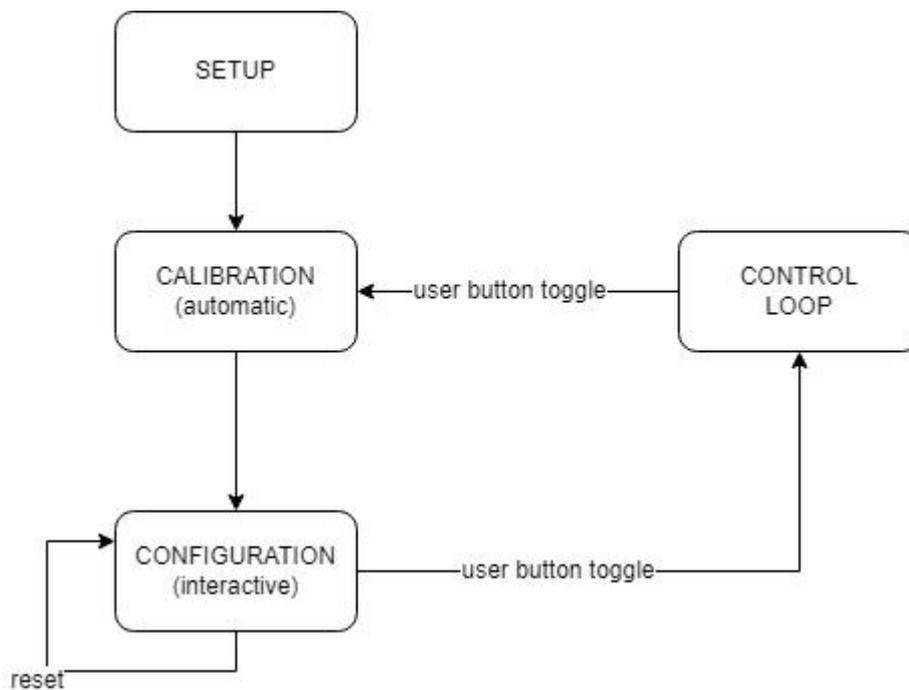


fig 39. Schematica macchina a stati

E' possibile confrontare il *main* completo del firmware al seguente link:

<https://github.com/fenix-hub/srap/blob/main/Core/Src/main.c>

3. Analisi dei risultati e conclusione

Attraverso il progetto SRAP è stato possibile verificare la fattibilità di una protesi robotica home-made, con tecnologie low cost e orientate sul *wearable*.

Ovviamente implementazione illustrata rappresenta un prototipo, e allo stato attuale e con le tecnologie impiegate non può sostituire una reale protesi transradiale robotica, come quelle già presenti in commercio.

Allo stesso tempo, il progetto ha avuto come obiettivo quello di sperimentare ed esplorare diverse tecniche di lettura e controllo all'interno di un sistema complesso e altamente interattivo e allo stesso tempo non invasivo, poichè basato su segnali provenienti direttamente dall'azione dell'utente.

Come più volte affermato all'interno della relazione, l'intero processo sperimentale non può considerarsi concluso a causa del non corretto funzionamento del sensore Myoware, e della sua non reperibilità nel mercato, a causa di diversi fattori, quali la non disponibilità di componenti elettroniche a causa dei recenti accadimenti geo-politici, la precedente ed ancora attuale crisi dei microprocessori, e l'ufficiale pausa da parte della casa produttrice del sensore Myoware nella sua produzione e distribuzione.

Ad ogni modo, alcune delle parti che non hanno potuto subire il processo di sperimentazione, sono state comunque implementate nel firmware del prototipo, come la procedura di calibrazione del sensore Myoware e lo sviluppo di un controllore PID.

Proprio per questo motivo, si spera che il progetto possa essere rivalutato in un futuro prossimo, poichè sono ancora molte le possibilità di miglioramento. Alcune di esse:

- Integrazione e sperimentazioni di filtri analogici e digitali per il miglioramento del campionamento del segnale EMG per un controllo continuo e più accurato dell'attuatore, come la famiglia di filtri di Kalman Steady-State o Transient Savitzky-Golay;
- Sostituzione del servomotore con pinza con un attuatore più completo, eventualmente con una riproduzione meccanica di una mano;
- Sperimentazione di una routine di calibrazione del sensore sul braccio dell'utente, per ottenere in maniera procedurale valori massimi e minimi associabili alla chiusura ed apertura della mano;
- Integrazione di un controllore PID per il controllo di uno o più attuatori in maniera continua a partire dal segnale EMG (senza utilizzo di un valore di treshold, ma attraverso una mappatura diretta in tempo reale);
- Sperimentazione nell'utilizzo di parte della memoria flash di STM32F466RE per la simulazione di una EEPROM in cui salvare i dati di calibrazione, per poterli utilizzare automaticamente in un nuovo loop di controllo ogni qual volta il prototipo venga alimentato;

Bibliografia

[1] Jarque-Bou, N.J., Vergara, M., Sancho-Bru, J.L. et al. Identification of forearm skin zones with similar muscle activation patterns during activities of daily living. *J NeuroEngineering Rehabil* 15, 91 (2018). <https://doi.org/10.1186/s12984-018-0437-0>