UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA BIOMEDICA

"Venipuncture Robot"

Relatore: Prof. Roberto Lot

Laureando: Alessandro Baschiera
n° matricola: 2042624

ANNO ACCADEMICO 2023 – 2024

Data di laurea 23/09/2024

Indice dei contenuti

Abstract

La puntura venosa per somministrare fluidi o prelevare un campione di sangue è la procedura clinica più comune. I medici continuano a fare affidamento sulle tecniche manuali di venipuntura, ma i tassi di successo dipendono fortemente dall'abilità del medico e dalla fisiologia del paziente. Per migliorare il tasso di successo al primo tentativo, in questa tesi si discute lo sviluppo un dispositivo portatile di venipuntura autonoma che inserisce roboticamente un ago in una vena adatta. Il dispositivo opera in tempo reale, combinando immagini a infrarossi e a ultrasuoni, analisi delle immagini e un sistema robotico a 7 gradi di libertà (GdL) per eseguire la puntura venosa. Il robot è costituito da un manipolatore cartesiano a 3 GdL e da un braccio seriale a 4 GdL per guidare la cannula nella vena selezionata correggendo la traiettoria attraverso un sistema di controllo a catena chiusa. Viene poi presentato un prototipo del manipolatore descrivendo i materiali e i componenti elettronici utilizzati, i metodi di costruzione, i software di controllo e uno script MatLab per la simulazione del movimento del braccio robotico.

Introduzione

Il processo di puntura venosa - anche chiamato venipuntura - per il prelievo ematico o con lo scopo di erogare qualsiasi fluido intravenoso, è ad oggi la procedura clinica più comune praticata in medicina [1]. Tutt'ora, tale operazione è ancora svolta manualmente: in primo luogo, l'infermiere identifica la vena nella regione del cavo cubitale visivamente e attraverso palpazione; in secondo luogo, procede con l'inserimento manuale della cannula fino al centro del vaso. Spesso, tuttavia, questo metodo incontra problemi pratici in entrambi i passaggi: localizzare il sito di venipuntura può risultare difficile in pazienti obesi o dalla pelle scura, nonché nella popolazione pediatrica e geriatrica dove le vene sono solitamente piccole e fiacche; inoltre, può essere difficoltoso stimare la profondità della vena ed evitare l'overshooting (perforazione del vaso fino all'estremità opposta). Complessivamente, le percentuali di fallimento vengono segnalate essere tra il 30% e il 50% dei tentativi [2].

In alcuni casi, durante l'inserimento dell'ago, le forze meccaniche generate dalla punta possono causare la deformazione, la traslazione o la rotazione della vena nella regione target. Queste trasformazioni sono talvolta la causa dei mancati tentativi di puntura venosa e, per essere evitate, è richiesta un'ottima abilità ed esperienza da parte dell'infermiere nel performare delicati e precisi aggiustamenti all'orientazione dell'ago in tempo reale.
Di conseguenza, risulta chiara l'utilità di un dispositivo in grado di poter guidare l'ago attraverso tecnologie di imaging biomedico e un sistema di controllo con feedback a catena chiusa.

Negli anni recenti, varie tecnologie di imaging basate su ultrasuoni (US) [3], visible light (VIS) [4], o near-infrared light (NIR) [5] sono state introdotte per aiutare gli infermieri nel trovare la vena adatta. Ciononostante, tali tecnologie si basano ancora sull'inserimento manuale dell'ago svolto da una persona spesso qualificata ma non sempre esperta.
Complessivamente, la ricerca ha proposto risultati curiosi riguardanti l'efficacia dell'implementazione di dispositivi di imaging: diversi studi mostrano l'assenza di significativi miglioramenti nelle percentuali di successo al primo tentativo, nel numero di tentativi, o nei tempi della procedura rispetto alla venipuntura standard [6] [7]. Questi mancati miglioramenti potrebbero suggerire che la difficoltà della procedura stia principalmente nell'inserimento della

cannula, invece che nella localizzazione della vena, e che lo scarso controllo dell'ago sia la causa primaria di fallimento della puntura venosa.

Date le precedenti premesse, il progetto consiste nella prototipazione di un robot per l'automatizzazione della puntura venosa. A partire da un sistema di imaging biomedico, segue un algoritmo di intelligenza artificiale in grado di scegliere il miglior cluster di vene target. Successivamente, attraverso l'utilizzo della tecnologia ad ultrasuoni Doppler, viene verificato che sia sufficiente la portata ematica. L'algoritmo, quindi, prosegue nella decisione del miglior sito di venipuntura sia internamente, a livello dell'endotelio del vaso, sia esternamente, a livello cutaneo. Infine, un manipolatore meccanico a 7 gradi di libertà posiziona e guida l'ago durante tutto il processo di inserimento, infusione o prelievo, e rimozione. In questa tesi, si concentra l'attenzione su sintesi, produzione e simulazione di un prototipo del manipolatore meccanico.

Capitolo 1: Organi del Robot per Venipuntura

1.1 NIR Imaging

Un dispositivo per la rilevazione delle vene è composto principalmente da un LED NIR (Near Infra-Red) ad alta potenza, che funge da sorgente luminosa, una telecamera sensibile agli infrarossi, un sensore per acquisire e formattare in tempo reale l'immagine acquisita, e un filtro per la purificazione, che blocca le lunghezze d'onda indesiderate. Questa tecnologia è in grado di fornire un'immagine chiaramente visibile delle vene superficiali.

Esistono due principi di base per i rilevatori di vene: la luce riflessa e la transilluminazione. La modalità a luce riflessa è la più comunemente utilizzata: la luce della sorgente viene riflessa nel punto focalizzato e l'immagine è catturata da una telecamera sensibile alla luce ad una data lunghezza d'onda. Per la transilluminazione, la luce viene fatta penetrare nella pelle e nei tessuti del sito, seguita dall'acquisizione dell'immagine tramite una telecamera. Complessivamente, la sorgente luminosa può essere considerata il componente principale dello scanner.

Con un intervallo dello spettro elettromagnetico che va da 740 nm a 940 nm, la luce può penetrare fino a circa 5 mm di profondità nei tessuti cutanei, raggiungendo la vena sottocutanea insieme alle cellule adipose, alle arterie e ai nervi. Nella Figura 1 è illustrata la penetrazione della luce nella pelle con varie lunghezze d'onda.

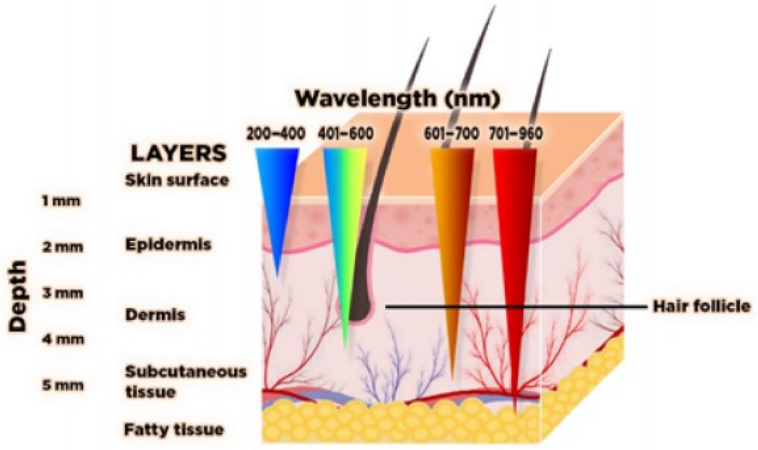

Figura 1. Penetrazione nella pelle della luce a diverse lunghezze d'onda (nm).

Le vene sono vasi sanguigni che hanno la funzione di trasportare il sangue deossigenato al cuore. Tale sangue deossigenato, contenente emoglobina, crea un forte contrasto con il tessuto cutaneo a seguito dell'assorbimento della luce NIR, rendendo le vene più visibili. La luce NIR è considerata una lunghezza d'onda a bassa energia e poco reattiva, ampiamente utilizzata in ambito clinico per la sua sicurezza [8].

Tutt'ora, sono già stati sviluppati diversi rilevatori basati sul principio della luce riflessa, e sembra che la tecnologia di Near Infra-Red imaging sia la più efficiente e popolare. Tuttavia, il problema maggiore, e quindi l'aspetto da migliorare di più in futuro, resta l'elevato costo di questi rilevatori: per i modelli portatili il prezzo si aggira attorno a circa 4.500 USD, per quelli non portatili può arrivare fino a circa 27.000 USD [9].

Nel 2021, MDPI pubblica l'articolo: *Competitive Real-Time Near Infrared (NIR) Vein Finder Imaging Device to Improve Peripheral Subcutaneous Vein Selection in Venipuncture for Clinical Laboratory Testing* [10] scritto da Mark D. Francisco, Wen-Fan Chen 4, Cheng-Tang Pan, Ming-Cheng Lin, Zhi-Hong Wen, Chien-Feng Liao e Yow-Ling Shiue. Al suo interno, viene presentato lo sviluppo di un rilevatore competitivo ma molto economico. I principali componenti utilizzati sono:

a. 3 NIR LED (Fongsam) 960 nm

b. IR (IMX238 Sony) CMOS camera con IR filter

c. Free software e laptop come processing unit delle immagini

Il dispositivo richiede un'alimentazione di 4.5V, ha un ingombro di 7 x 9 x 10 cm e pesa 0.4 kg. Offre un'ottima Reliability Rate del 94.21% sull'intero braccio, in confronto a circa 93.00% nei rilevatori commerciali. Tutto ciò risulta estremamente sorprendente quando viene comparato il prezzo stimato di 80-100 USD rispetto al limitante costo di 4000 USD dei rilevatori commerciali portatili precedentemente citato.

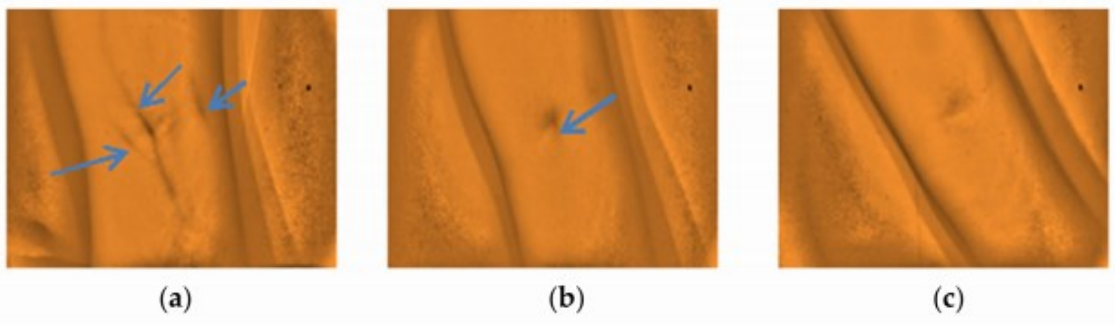

Figura 2. Esempi di immagini catturate dal rilevatore nella regione del braccio. (a) Highly visible, (b) Visible, (c) Non-visible

1.2 Doppler Ultrasound

In seguito alla rilevazione delle vene attraverso la tecnologia di imaging NIR, è necessario verificare che il flusso ematico all'interno delle possibili vene target sia sufficiente sia prima, sia durante tutta la procedura di prelievo ematico o di infusione. Tale misurazione viene condotta attraverso la tecnologia di imaging Doppler, che funziona ad ultrasuoni. Tenere sotto controllo questo parametro è fondamentale perché, se il flusso ematico non è sufficiente, significa che la vena può essere ostruita, contratta o collassata. Inoltre, visualizzare la direzione del flusso è importante per stabilire la direzione in cui inserire la cannula: è stato infatti dimostrato che esistono diversi vantaggi nel posizionare la cannula rivolta verso la periferia del sistema venoso piuttosto che il contrario [11].

L'imaging a ultrasuoni è una delle tecniche di imaging più comunemente utilizzate nella pratica clinica e nelle attività di ricerca. La combinazione dell'emissione di onde ultrasonore nei tessuti biologici, seguita dalla registrazione degli echi retro-riflessi, permette la ricostruzione di immagini anatomiche (B-Mode). Questo metodo è perfettamente adatto per l'imaging dei tessuti molli, come i tessuti biologici, che tipicamente permettono la penetrazione degli ultrasuoni per diversi centimetri, con una velocità di propagazione di circa 1540 m/s. A seconda della frequenza centrale della sonda a ultrasuoni, si ottengono immagini con una risoluzione che varia da 30 μm a 1 mm. Inoltre, è importante tenere conto che il movimento di una sorgente acustica influisce sulle caratteristiche fisiche delle onde associate. In particolare, il legame tra i cambiamenti di frequenza di un'onda rispetto alla velocità della sua sorgente è descritto come effetto Doppler, la cui manifestazione più semplice è il cambiamento di tono della sirena di un'ambulanza in movimento. L'imaging a ultrasuoni sfrutta questo effetto fisico per osservare i globuli rossi in movimento [12] e permette così la valutazione dei flussi sanguigni in applicazioni e organi molto diversi, come cervello, cuore, reni o arterie periferiche.
I principi di base dell'ultrasuono convenzionale sono i seguenti: un fascio acustico viene spostato lungo l'apertura del trasduttore a ultrasuoni e per ogni posizione del fascio, gli echi vengono registrati e convertiti in una linea di pixel dell'immagine finale. Spostando progressivamente il fascio lungo il trasduttore, l'intero campo visivo può essere acquisito linea per linea. Tuttavia, tale strategia presenta diversi svantaggi. Tra questi, la frequenza dei fotogrammi finale è limitata a poche centinaia di immagini al secondo dal processo di scansione del fascio. In termini di flusso sanguigno, questa frequenza relativamente bassa influisce sulle velocità massime di flusso che possono essere rilevate, dettate dai criteri di campionamento di Shannon-Nyquist. Inoltre, il Doppler convenzionale deve affrontare il seguente compromesso.

Per valutare la velocità del flusso sanguigno in una particolare regione di interesse, devono essere registrati successivamente diversi echi provenienti da quella stessa regione. Ciò implica che il fascio ultrasonoro sia temporaneamente mantenuto in una posizione fissa. Più lungo è l'insieme degli echi, migliore sarà la stima della velocità per quella regione. Tuttavia, per produrre un'immagine completa del campo visivo, il fascio deve scansionare il mezzo. Pertanto, si può notare il conflitto tra questi due vincoli: mantenere il fascio per valutare con precisione la velocità lungo una linea o spostare il fascio per produrre un'immagine.

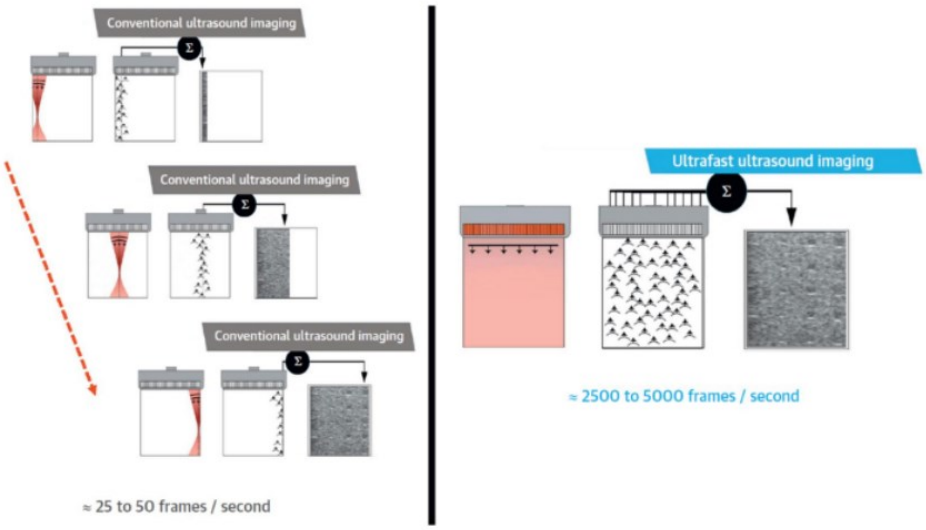

Figura 3. (Sinistra) Imaging convenzionale con emissioni focalizzate. (Destra) Imaging ad ultrasuoni ultraveloci con emissioni di onde piane.

Queste due limitazioni (bassa frequenza dei fotogrammi e compromesso tra localizzazione e quantificazione) sono superate con le nuove tecniche di imaging a ultrasuoni che permettono un'altissima frequenza dei fotogrammi. Un esempio è l'imaging a ultrasuoni ultraveloce: invece di utilizzare un fascio focalizzato in scansione, questo metodo utilizza onde piane o onde divergenti, in grado di pervadere l'intero campo visivo con una singola emissione. Dopo quella singola emissione, l'elettronica associata è anche in grado di ricevere e processare il gran numero di echi provenienti dall'intero campo visivo. Alla fine, un'immagine può essere ricostruita da un singolo schema di emissione/ricezione. Queste emissioni non focalizzate possono però avere un basso rapporto segnale/rumore (SNR) a causa della diffusione dell'energia acustica. Tale problema può essere affrontato emettendo diverse onde piane inclinate, oppure onde divergenti con sorgenti diverse, e sommando le immagini risultanti (la tecnica prende il nome di compunding coerente [13]). Ne derivano due conseguenze principali: in primo luogo, la frequenza dei fotogrammi dipende solo dal tempo di volo dell'ultrasuono e può raggiungere valori tipici da 1 a 10 kHz; in secondo luogo, garantisce la coerenza

spaziotemporale, ovvero la continuità dei dati sia nelle dimensioni spaziali che temporali. Questa combinazione di elevata frequenza dei fotogrammi e di coerenza spaziotemporale ha un impatto notevole sulla capacità di rilevare il flusso ematico con gli ultrasuoni: rispetto agli ultrasuoni convenzionali, gli ultrasuoni ultraveloci forniscono una caratterizzazione completa del flusso sanguigno. In pratica, l'utente (che nel progetto del Robot per Venipuntura è sostituito dall'algoritmo di scelta della vena target e dal sistema di controllo a catena chiusa, si veda sezione 1.3 e 1.4) ha accesso all'andamento temporale della velocità in ogni pixel dell'immagine, per tutta la durata dell'acquisizione. Inoltre, è stato dimostrato che la coerenza spaziotemporale migliora notevolmente la capacità di separare il flusso sanguigno lento dai tessuti in movimento di fondo, aumentando quindi la sensibilità al flusso microvascolare.

La regolazione dei parametri fisici della sequenza ultrasonora consente lo studio sia dei flussi lenti (fino a 1 mm/s) sia di quelli veloci (fino a diversi m/s). Esiste, tuttavia, un compromesso tra la risoluzione spaziale e la profondità di penetrazione: tipicamente, si può ottenere una risoluzione di 50 μm al costo di una penetrazione di circa 5 mm; al contrario, la penetrazione può essere estesa a 15-20 cm al costo di una risoluzione di 1 mm. Nel contesto di puntura venosa, vengono utilizzati usualmente aghi da 4 cm e va tenuto conto che il dolore percepito dal paziente aumenta all'aumentare della profondità di penetrazione dell'ago. In conclusione, è più che sufficiente che il sistema di imaging a ultrasuoni abbia una penetrazione di circa 5 cm.

1.3 Algoritmo Decisionale

Perché il Robot per Venipuntura possa funzionare autonomamente, ovvero con la sola supervisione di un operatore o magari senza nemmeno il bisogno di quest'ultimo (l'approvazione del paziente non è per nulla scontata, si veda Conclusioni), è necessaria la presenza di un algoritmo decisionale che coordini il sistema di imaging, il sistema di controllo e il manipolatore. In pratica, i compiti svolti dall'algoritmo sono:

1. Ricevere in input i dati del sistema di NIR imaging e restituire un cluster di vene target candidate a diventare il sito di venipuntura. Tali vene devono essere raggiungibili: se il manipolatore è dotato di un ago di 4 cm allora i vasi devono trovarsi a non più di 2 cm di profondità. Va tenuto conto, infatti, che il massimo angolo di inserimento accettabile è di 30 gradi: quando una vena viene perforata con un angolo superiore a 30 gradi, la possibilità di attraversare la vena e penetrare nelle strutture sottostanti aumenta il rischio di lesioni permanenti. Per cui la massima profondità risulta:
   4 * sin(30) = 2 cm.
2. Ricevere in input i dati del sistema di imaging a ultrasuoni, in particolare le caratteristiche del flusso ematico, e restituire un sottoinsieme del cluster prodotto al punto 1. All'interno di questo sottoinsieme devono essere presenti solo vene che presentano un flusso sanguigno sufficiente e con una direzione facilmente accessibile dal manipolatore.
3. A partire dal sottoinsieme generato al punto 2, l'algoritmo deve calcolare per ogni vena il path ottimale che l'ago deve compiere per raggiungere il miglior sito di puntura che offre tale vena. Dopodiché, verrà selezionata la miglior terna
   vena-sito-path che possa rendere l'intera procedura sicura, rapida e con la più alta probabilità di successo possibile.

Nella stesura, nell'apprendimento e nel testing dell'algoritmo è importante seguire alcune linee guida. Ad oggi, pochi articoli e studi sono stati pubblicati riguardo l'implementazione di algoritmi decisionali in ambito medico dato che si preferisce, e spesso soprattutto il paziente preferisce, affidare ogni scelta di tipo decisionale al medico. Nel 2023 BMJ Health Care Inform pubblica l'articolo: *Road map for clinicians to develop and evaluate AI predictive models to inform clinical decision-making* scritto da Nehal Hassan, Robert Slight, Graham Morgan, David W Bates, Suzy Gallier, Elizabeth Sapey, Sarah Slight. Al suo interno, vengono delineati nove step da seguire per sviluppare e valutare un algoritmo decisionale adibito all'ambito medico:

Step 1. Chiarire la domanda clinica o gli esiti di interesse (output).

Step 2. Identificare predictors appropriati (selezione delle features).

Step 3. Scegliere dataset pertinenti.

Step 4. Sviluppare il modello predittivo di AI.

Step 5. Validare e testare il modello sviluppato.

Step 6. Presentare e interpretare le predizioni del modello.

Step 7. Pubblicare licenze per il modello.

Step 8. Mantenere il modello.

Step 9. Fare una valutazione continua dell'impatto del modello.

Viene poi sottolineata la vitale importanza di richiedere l'input di un team multidisciplinare già durante i primi passi di sviluppo: oltre all'ovvio ruolo di ingegneri informatici che selezionino l'appropriato algoritmo, è necessaria la presenza di specialisti clinici che guidino il progetto, fornendo continui feedback, verso un prodotto che sia in grado di prendere la migliore decisione anche in situazioni non standard come pazienti pediatrici, geriatrici, o obesi. Inoltre, va riposta molta attenzione nell'includere queste categorie, e un buon numero di pazienti che presentano particolari necessità, nei dataset di apprendimento. In questo modo il robot potrà sostituire l'infermiere proprio nelle situazioni in cui la percentuale attuale di successo è minore.

1.4 Tracking e Sistema di Controllo a Catena Chiusa

Sebbene l'imaging intraoperatorio sia talvolta disponibile per assistere nella procedura di puntura venosa, il feedback basato sulle immagini in genere non viene utilizzato per regolare la traiettoria desiderata durante l'inserimento dell'ago, non tenendo conto così delle possibili deviazioni dell'ago o dello spostamento del target dovuto alla deformazione dei tessuti. Al contrario, un sistema di controllo a catena chiusa permette di compensare attivamente questo tipo di errori sotto la guida delle immagini fornite dal sistema di imaging NIR e ad ultrasuoni.
Per poter funzionare correttamente però, sono necessarie le dovute calibrazioni dei sistemi di imaging, da svolgere ad ogni accensione del robot. In più, il sistema di controllo deve avere accesso in ogni istante alle coordinate dell'ago e, di conseguenza, è fondamentale la presenza di un metodo di tracking della punta dell'ago e di ricostruzione 3D della forma della cannula.

Nel 2021 la rivista Int J Med Robot pubblica l'articolo: *Integrating robot-assisted ultrasound tracking and 3D needle shape prediction for real-time tracking of the needle tip in needle steering procedures* scritto da Bardia Konh, Blayton Padasdao, Zolboo Batsaikhan1, Seong Young Ko. Al suo interno, viene presentato un algoritmo, scritto in Python, in grado di muovere in tempo reale il trasduttore di ultrasuoni in concomitanza dell'ago, ricevere le immagini 2D, identificare la punta dell'ago e ricostruire la forma 3D della cannula.
Il trasduttore US viene spostato seguendo l'ago lungo l'asse di inserimento e le immagini 2D ottenute forniscono una sezione trasversale radiale dell'ago. Tuttavia, la forma circolare dell'ago è solitamente deformata da un artefatto noto come "comet tail artifact" (CTA). L'algoritmo elabora le immagini e, nella vista trasversale influenzata dalla CTA, la blob detection identifica i pixel associati all'ago. Viene calcolato il centroide di quest'area a cui vengono associate coordinate locali che, infine, vengono tradotte in coordinate globali per trovare la posizione della punta dell'ago.
Nella Figura 4 è rappresentato il diagramma di flusso del programma che permette il tracking real-time.

Grazie al sistema di tracking, il controllore del sistema di controllo a catena chiusa determina i parametri di compensazione a partire dalle trasformazioni dei frame trasmessi, dove sono rappresentati il target e la punta dell'ago. Il robot, quindi, aggiorna autonomamente la posizione della cannula in base a questi input per dirigere la punta dell'ago verso il bersaglio. A differenza di un controllo a ciclo aperto, la compensazione attiva è robusta rispetto a errori di registrazione, distorsioni delle immagini, deviazioni dell'ago e gonfiore o spostamento della vena target. Nella

Figura 5 viene presentato un esempio di sistema di controllo a catena chiusa basato su un controllore PID [14].

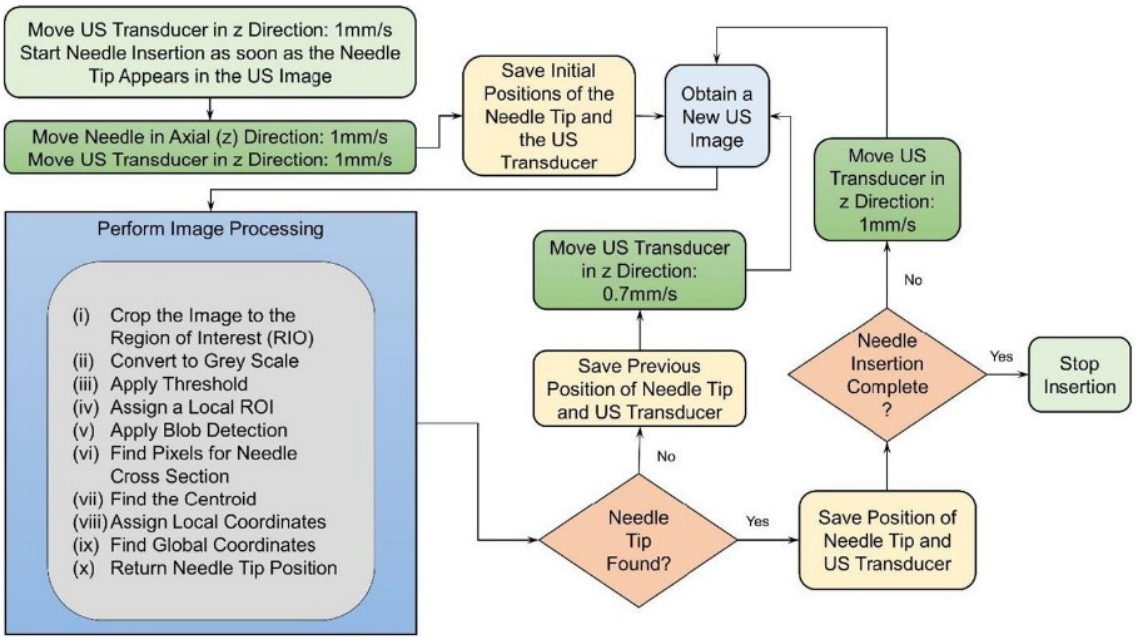

Figura 4.
Diagramma di flusso del programma che esegue l'elaborazione delle immagini e il tracciamento in tempo reale della punta dell'ago durante un task di inserimento dell'ago.

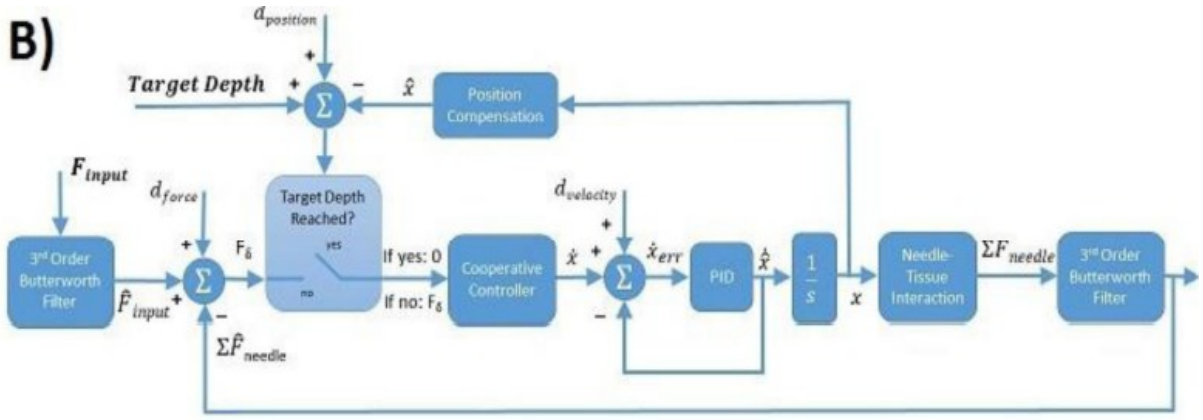

Figura 5.
Esempio di sistema di controllo a catena chiusa per implementare l'inserimento cooperativo controllato dell'ago. I due input sono una forza e una profondità target, mentre i due output sono le forze raccolte sull'ago e la posizione della punta dell'ago.

Capitolo 2: Sicurezza

2.1 End Effector

Per garantire sufficiente sicurezza nella gestione dell'ago, l'operatore sanitario, ipoteticamente incaricato di supervisionare la procedura di prelievo ematico o di infusione di liquido intravenoso, dovrebbe essere esposto il meno possibile al contatto con strumenti appuntiti esposti, in particolare durante la preparazione e la conclusione della procedura.

A tal fine, è necessario implementare un sistema quanto più sicuro di ricarica. L'end effector del manipolatore deve essere dotato di un elettromagnete che possa agganciare e sganciare una rondella in acciaio presente alla base di un nuovo ago. In questo modo, l'infermiere può caricare l'ago rimuovendo il cappuccio protettivo solo una volta che esso sia stato posizionato correttamente e agganciato saldamente. Al termine della venipuntura, il paziente ritrae il proprio braccio e il manipolatore posiziona l'end effector al centro del workspace, sollevato di una decina di centimetri dalla zona di appoggio del gomito. A questo punto, l'infermiere prende un piccolo contenitore, disattiva l'elettromagnete e l'ago viene sganciato cadendo direttamente nel contenitore evitando ogni contatto.

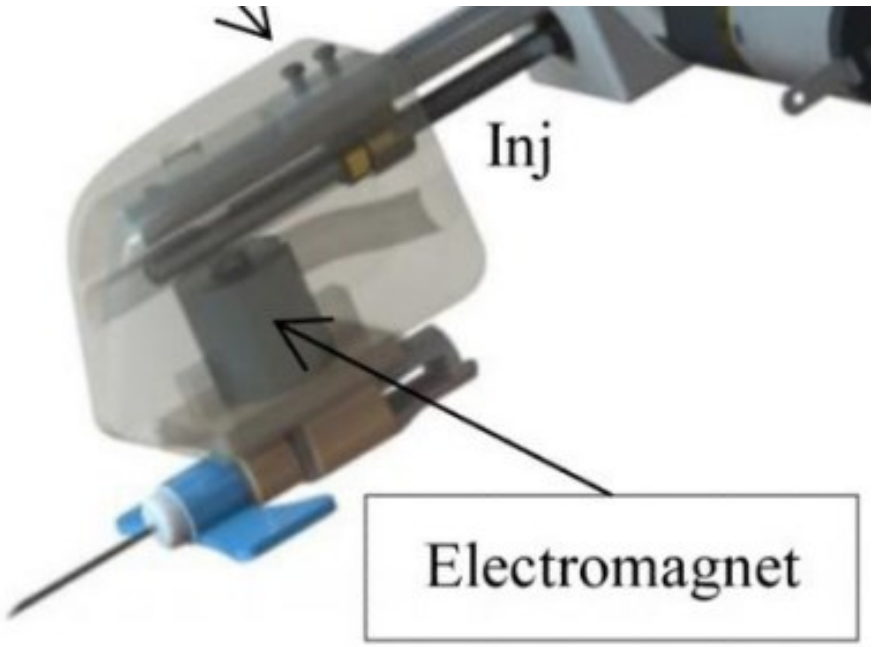

Figura 6.
End effector dotato di elettromagnete con ago agganciato.

2.2 Motori e Batteria di Emergenza

Tipicamente, per manipolatori che richiedono ottima precisione si sceglie di utilizzare motori passo-passo. Un motore passo-passo, o stepper motor, è un motore elettrico che ruota in una serie di piccoli e discreti passi angolari senza la necessità di un sensore di posizione per il feedback. La posizione del passo può essere rapidamente aumentata o diminuita per creare una rotazione continua, oppure al motore può essere ordinato di mantenere attivamente una posizione fissa con un buon rapporto coppia-peso. Uno stepper in modalità full step presenta 200 passi per rotazione completa e, perciò, un passo equivale a 1.8°. Per ottenere maggiore precisione nel controllo dell'ago, esistono driver che permettono al motore di lavorare in modalità microstepping, ovvero dividendo ogni passo in altri micropassi fino ad un rapporto di 1/128, rendendo quindi un micropasso equivalente ad una rotazione di 0.014°. Altri vantaggi che inducono a scegliere questi motori sono per esempio la caratteristica di fornire la massima coppia a velocità ridotte e l'eccellente affidabilità nelle fasi di avvio, arresto e inversione del moto.

Come tutti i motori elettrici, i motori passo-passo hanno una parte stazionaria (stator) e una parte mobile (rotor). Sulla parte stazionaria, sono presenti dei denti sui quali sono avvolte le bobine, mentre il rotore è costituito da un magnete permanente o da un nucleo di ferro a riluttanza variabile.

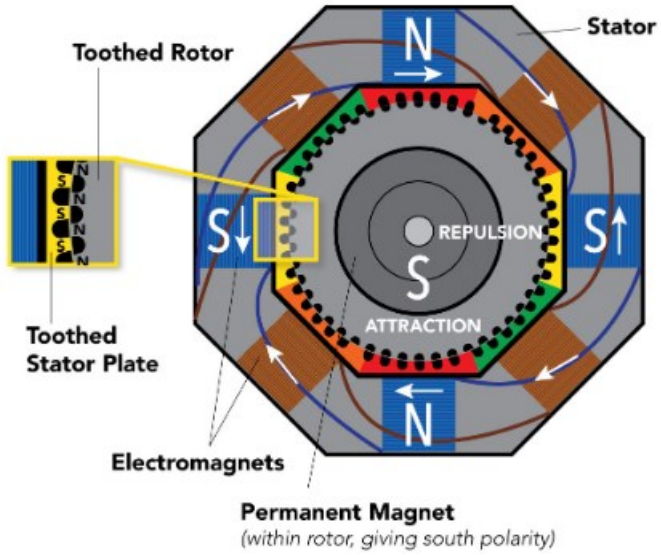

Figura 7.
Diagramma di un motore passo-passo.

Utilizzando motori elettrici, tuttavia, va tenuto conto che, se il voltaggio richiesto da uno stepper viene a mancare per qualsiasi motivo, la coppia generata dal motore crolla considerevolmente. Pertanto, in un'applicazione come il manipolatore del Robot per Venipuntura, un calo

improvviso di tensione implica la totale perdita di controllo dell'ago con conseguenze gravissime per il paziente. Per garantire assoluta sicurezza sotto questo aspetto, è necessario che i motori del manipolatore (e l'elettromagnete presente nell'end-effector) siano sempre collegati in parallelo ad una batteria d'emergenza. In questo modo, anche in caso di blackout, viene sempre fornita un'alimentazione indipendente.

Capitolo 3: Sintesi del Manipolatore

3.1 Architettura

La sintesi meccanica del manipolatore si pone i seguenti obiettivi:

1. Il manipolatore dovrebbe avere l'accuratezza necessaria per incannulare vene con un diametro minimo di ø2 mm (come quelle presenti nelle popolazioni pediatriche).
2. La forza di inserimento dell'ago dovrebbe essere superiore a 5 N (sufficiente per perforare il tessuto cutaneo umano [15]).
3. Il braccio robotico dovrebbe essere quanto più possibile compatto e leggero.
4. Il manipolatore dovrebbe avere uno spazio operativo superiore a 175 cm^3 (sufficiente per il task di inserimento dell'ago [16]).

Tenendo conto di tali obiettivi, il prototipo presentato nella Parte II consiste in un robot a 7 gradi di libertà (GdL) composto da un manipolatore cartesiano a 3 GdL e un braccio seriale a 4 GdL. Il prototipo, per motivi pratici, non comprende l'end effector precedentemente descritto (al suo posto è stato inserito un ago fittizio), né i dispositivi di imaging necessari al reale funzionamento del robot.

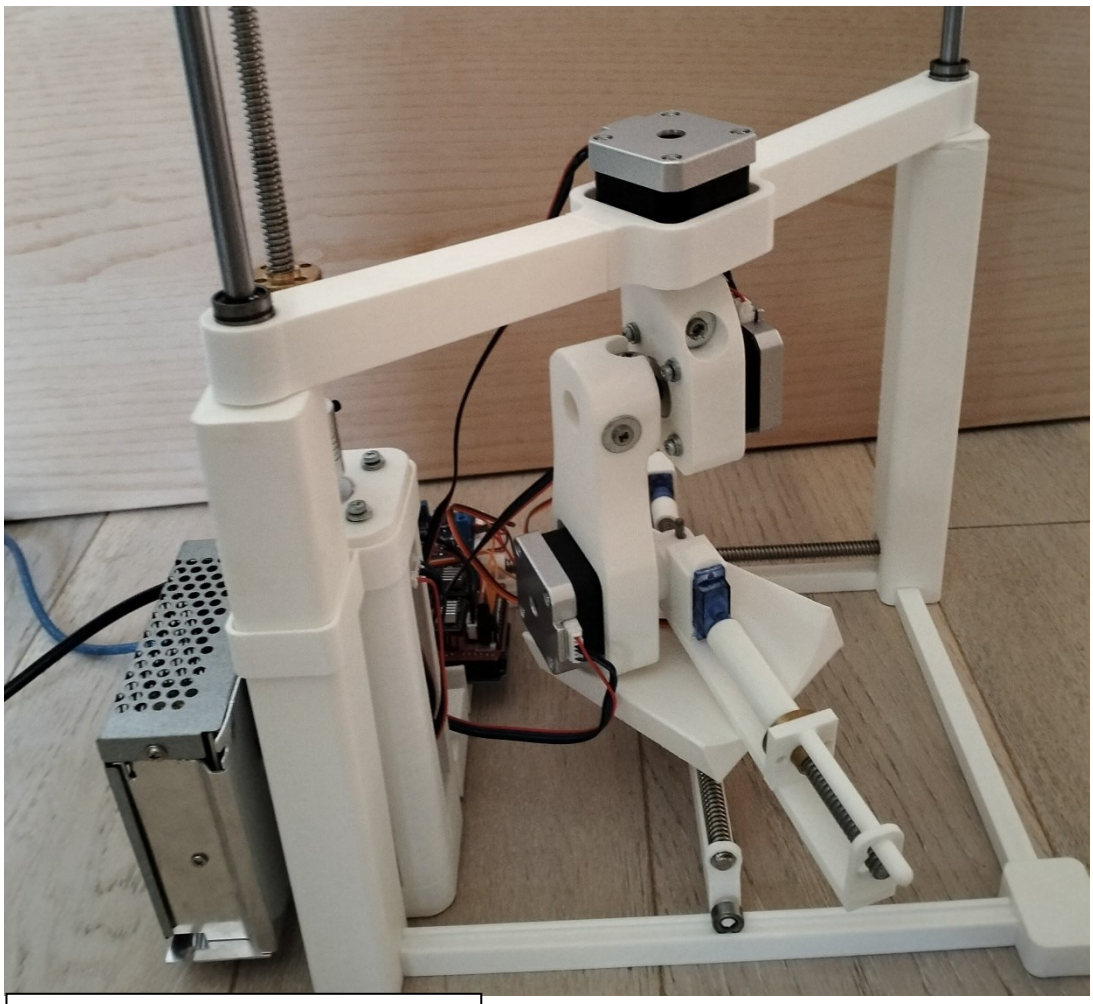

Figura 7. Prototipo del manipolatore.

3.2 Cinematica Diretta e Inversa

L'analisi cinematica diretta permette di calcolare le coordinate della posizione della punta dell'ago, date le posizioni dei moventi. L'analisi cinematica inversa, al contrario, permette di calcolare la posizione dei moventi date le coordinate della punta dell'ago. Nelle seguenti figure sono illustrati gli schizzi del manipolatore analizzando separatamente la fase cartesiana (3 GdL) e il braccio seriale (4 GdL).

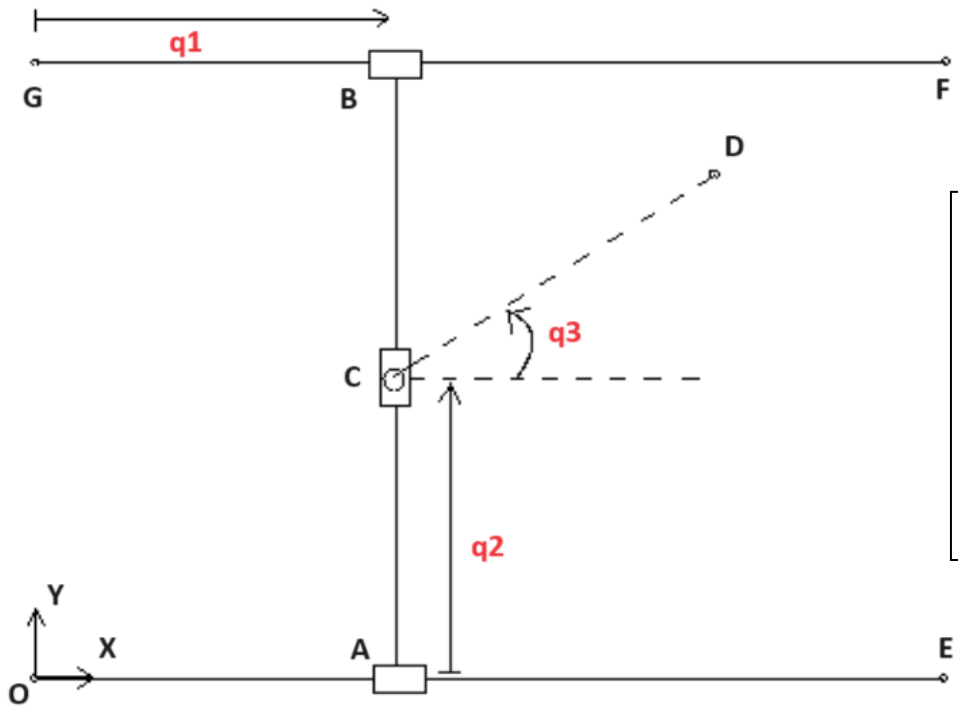

Figura 8. Schizzo del manipolatore cartesiano a 3 GdL (visto dall'alto). Il segmento CD rappresenta la proiezione del braccio seriale sul piano xy.

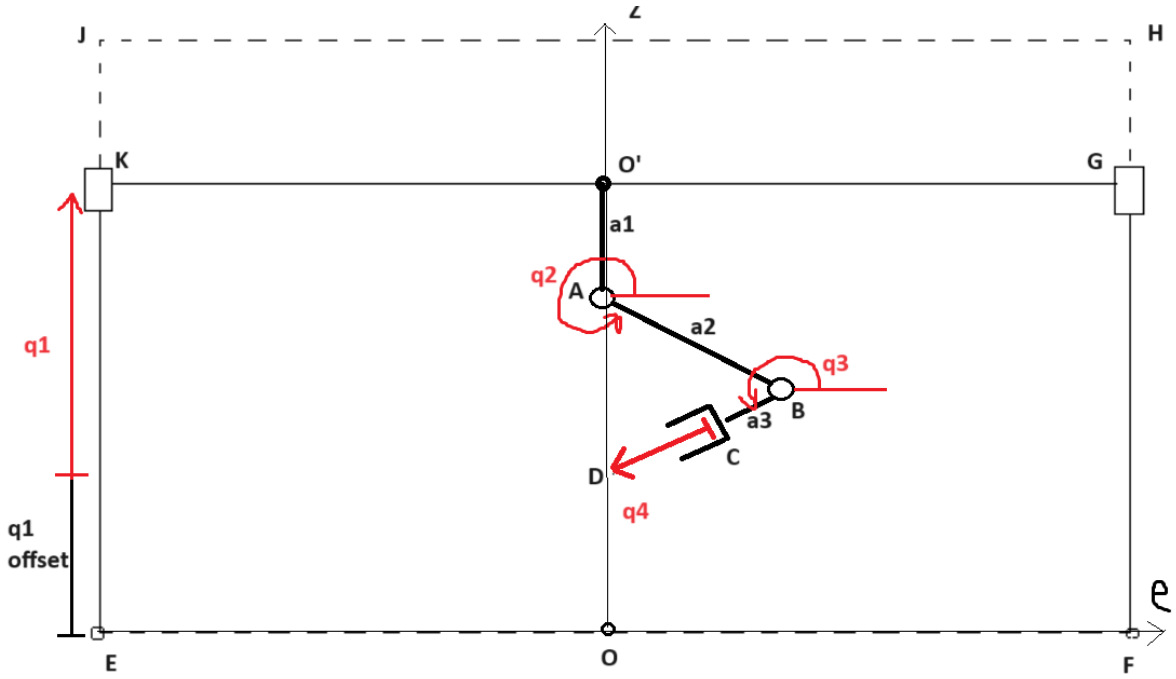

Figura 9. Schizzo del braccio seriale a 4 GdL (visto di profilo). Utilizzando coordinate cilindriche il braccio viene analizzato come un meccanismo piano giacente sul piano rho-z.

Per il manipolatore cartesiano, il sistema di riferimento *Oxy* è fisso. I moventi sono q1, q2, q3, rispettivamente asse x, asse y e giunto j1 nel prototipo. O, E, F e G sono perni a telaio. A e B sono coppie prismatiche e C è una coppia camma. Il segmento CD rappresenta la proiezione del braccio seriale sul piano *xy*.

Il braccio seriale viene analizzato come un meccanismo piano, giacente sul piano *ρz*, trasformando poi le coordinate nelle coordinate cilindriche del sistema di riferimento globale. Pertanto, il sistema di riferimento *Oρz* è mobile. I moventi sono q1, q2, q3, q4, rispettivamente asse z, giunto j2, giunto j3 ed estensore dell'ago (per semplicità, la distanza percorsa dalla punta dell'ago) nel prototipo. E ed F sono perni a telaio. K, G e C sono coppie prismatiche, A e B sono coppie rotoidali. Le misure dei membri a1, a2 e a3 sono fisse e note.

1. Analisi cinematica diretta del braccio seriale a 4 GdL:

Noti q1, q2, q3 e q4, calcolo coordinate del punto D (punta dell'ago).

$$\rho_D = a_2\cos(q_2) + (a_3 + q_4)\cos(q_3)$$
$$z_D = q_{1\ offset} + q_1 - a_1 + a_2\sin(q_2) + (a_3 + q_4)\sin(q_3)$$

2. Analisi cinematica inversa del braccio seriale a 4 GdL:

Il movente q3 rappresenta l'angolo di inserimento dell'ago (compreso tra 10° e 30°) e viene fissato a priori dall'algoritmo decisionale.

Il movente q4 rappresenta l'estensione dell'ago e quanto questo affonda nel braccio del paziente. Viene quindi fissato a priori dall'algoritmo decisionale e viene controllato attivamente dal sistema di controllo a catena chiusa per evitare la perforazione del vaso fino all'estremità opposta di quest'ultimo.

Noti q3, q4 e le coordinate del punto D (punta dell'ago), calcolo q1 e q2.

$$\rho_B = \rho_D - (a_3 + q_4)\cos(q_3)$$
$$z_B = z_D - (a_3 + q_4)\sin(q_3)$$

$$q_2 = \arctan\left(\frac{z_B}{\rho_B}\right)$$

$$q1 = z_B - a_2\sin(q_2) + a_1 - q_{1\ \text{offset}}$$

3.3 Simulazione MatLab

Utilizzando il software MatLab è possibile simulare il meccanismo del manipolatore controllato in cinematica inversa. Lo script presente nell'annex simula il movimento tridimensionale del braccio seriale, traccia la traiettoria del punto C e del punto O' (si veda schizzo in Figura 9), e plotta la posizione dei motori nel tempo. Dopodiché, premendo il tasto invio, simula l'estensione dell'ago. Il calcolo delle coordinate viene svolto analizzando il braccio seriale in 2 dimensioni, sul piano ρz, utilizzando la notazione complessa. Successivamente, viene aggiunta la coordinata θ che è uguale per tutti i punti. Infine, le coordinate cilindriche vengono trasformate in coordinate cartesiane tridimensionali per poter plottare la simulazione. Il main script utilizza 3 funzioni ausiliari: la funzione "cylindrical" trasforma le coordinate 2D in notazione complessa nelle coordinate cilindriche 3D; la funzione "create_trajectory" prende in input le coordinate del punto target e genera una traiettoria dal punto di partenza al punto di arrivo del punto C; la funzione "inverse_kinematics5" prende in input ciascun punto della traiettoria generata dalla funzione precedente e calcola la posizione istantanea dei motori q1 e q2.

Nelle seguenti figure viene presentato l'output ottenuto alla fine della simulazione.

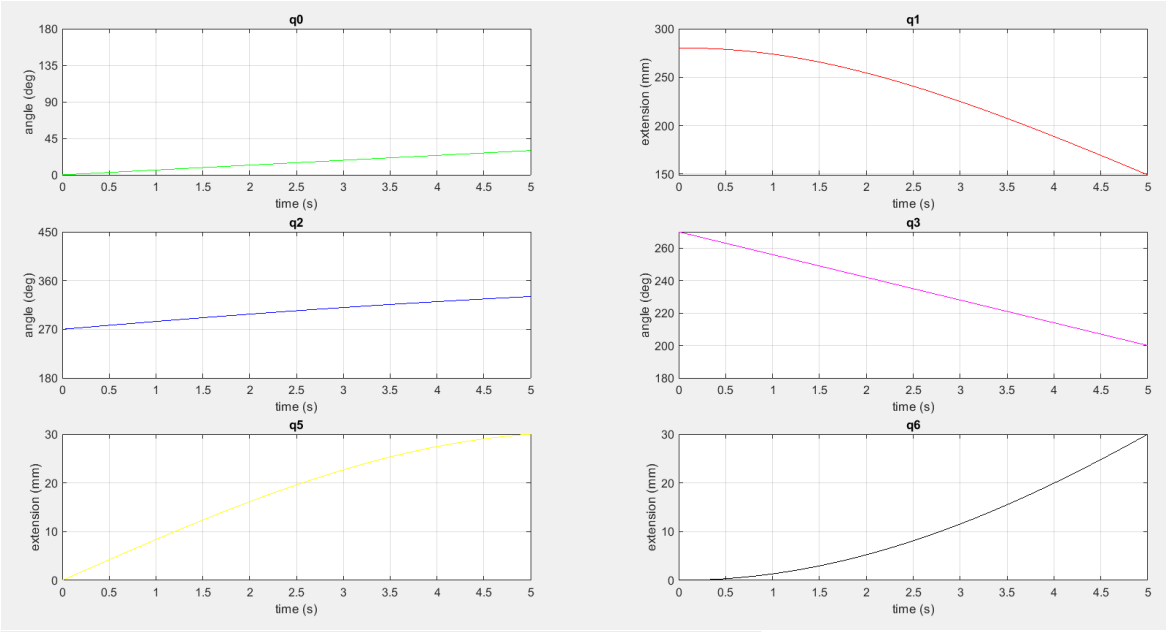

Figura 10. Posizione dei motori nel tempo durante la simulazione.

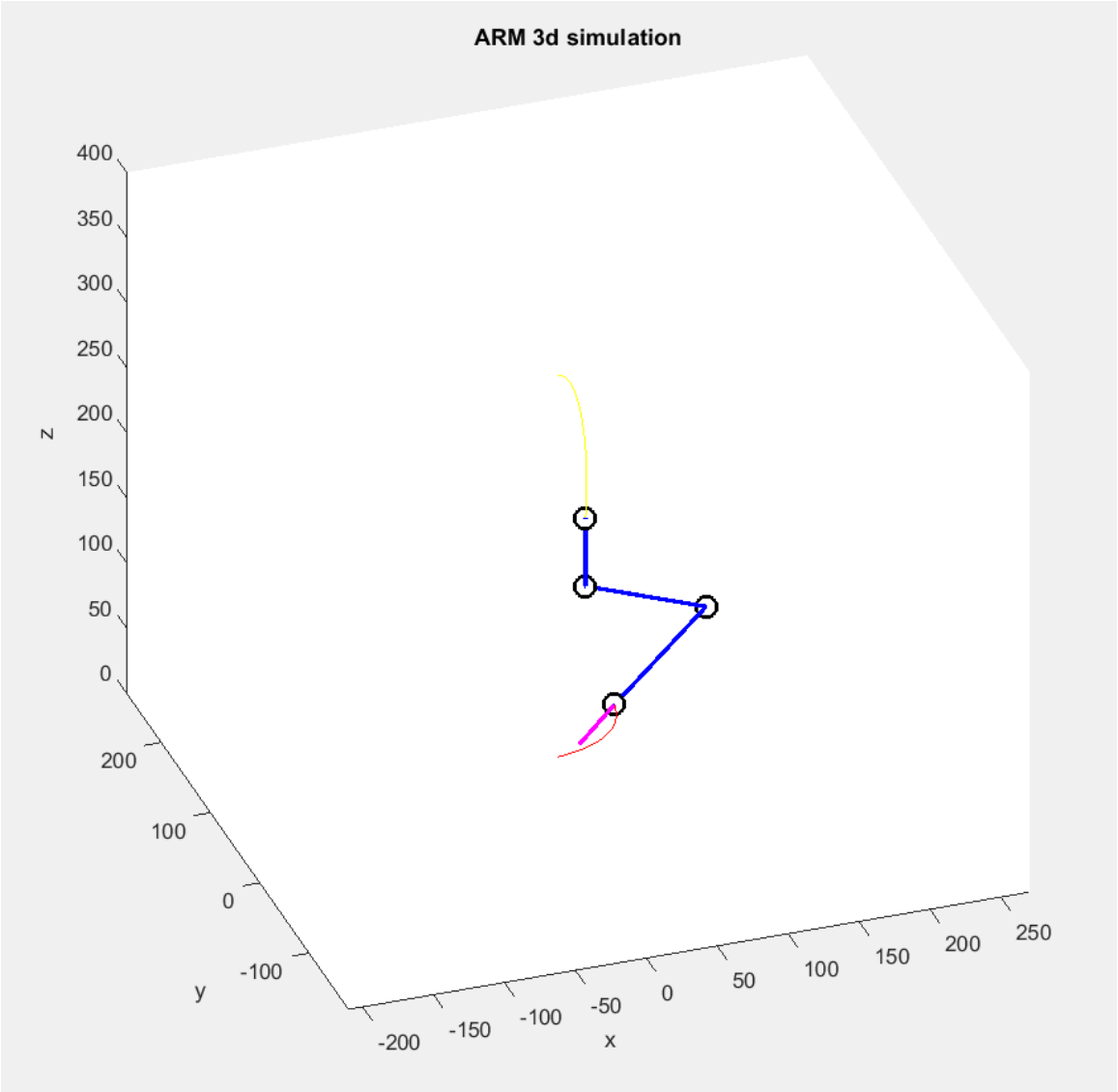

Figura 11. Posizione finale del braccio seriale. In giallo, traiettoria del punto O'. In rosso, traiettoria del punto C.

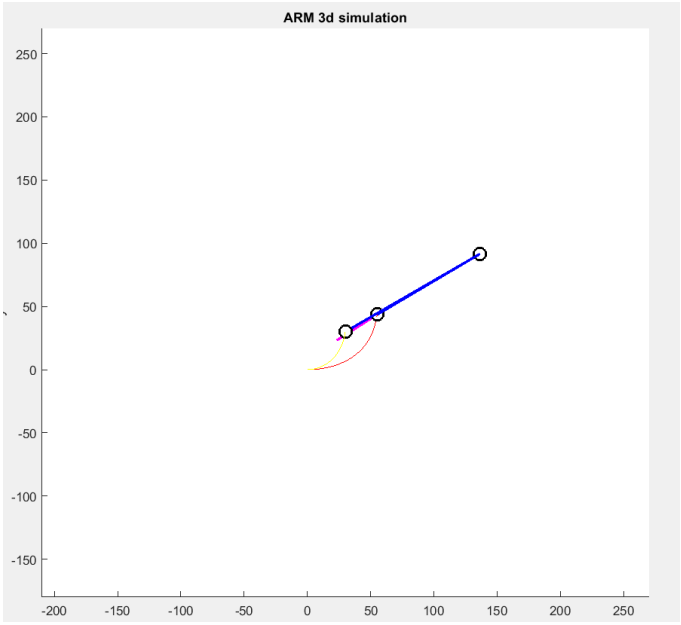

Figura 12.
Posizione finale del braccio (visto dall'alto). Si può notare come i 4 GdL (5 aggiungendo l'asse z), necessari al controllo dell'orientazione e all'estensione dell'ago, non bastano per permettere anche il movimento sul piano *xy*. Pertanto, risulta necessaria la sintesi meccanica del manipolatore cartesiano. Si arriva dunque ad un manipolatore a 7 gradi di libertà.

Capitolo 4: Prototipo del Manipolatore

4.1 Motori e Materiali Strutturali

I 7 gradi di libertà del prototipo sono controllati da 4 motori passo-passo Nema17 (modello 17HS4401) e 3 servomotori SG90 con 360° di rotazione libera.

I motori passo-passo sono stati utilizzati per i 3 giunti del braccio seriale e per l'asse z. Sono impostati in modalità microstepping con un rapporto di 1/16. Pertanto, un micropasso corrisponde ad una rotazione di 0.1125°. Necessitano di un'alimentazione a 12V in corrente continua e assorbono 1.7A ciascuno. Le dimensioni sono 42x42x34 mm, il peso è 225g e la massima coppia generata è 40 Ncm.

Figura 13. Stepper Nema17.

I servomotori sono stati utilizzati per l'asse x, l'asse y e l'estensore dell'ago. Necessitano di un'alimentazione di circa 5V in corrente continua e assorbono una corrente variabile che dipende dal carico. Le dimensioni sono 26x13x24 mm, il peso è 9g e la massima coppia generata è14,715 Ncm.

Figura 14. Servomotore SG90.

Il principale materiale strutturale del prototipo è PLA bianco (acido polilattico), un monomero termoplastico comunemente utilizzato come filamento per stampanti 3D. I principali vantaggi che offre sono: basso punto di fusione, elevata resistenza, bassa espansione termica e buona adesione tra gli strati.

L'asse x, l'asse y e l'asse z sono costituiti da viti senza fine lunghe 200mm con passo di 2 mm e lead, ovvero la distanza percorsa linearmente dal dado dopo una completa rotazione della vite, di 4 mm (2 mm per l'asse z).

4.2 Circuito Elettronico

Il microprocessore utilizzato per controllare tutti i motori è presente nella scheda Arduino Uno, alimentata a 5V tramite il cavo USB collegato al computer. Sopra la scheda è montato uno shield CNC v3 che contiene 4 slot per driver DRV8825. Questi driver generano le onde quadre che controllano i motori passo-passo e la corrente in output viene limitata attraverso un potenziometro (presente su ciascun driver) settato ad una tensione di riferimento di 0.68V data dalla formula:

$$V_{ref} = \frac{I_{lim}}{2}(1-10\%) = \frac{1.50A}{2}(1-\frac{10}{100}) = 0.68V$$

dove la corrente limite è la corrente per fase dei motori passo-passo.

L'alimentazione viene fornita direttamente allo shield da un alimentatore a 12V e corrente continua per un totale di 10A. L'alimentatore è inoltre collegato ad un regolatore di tensione che provvede a fornire 5V ai 3 servomotori.

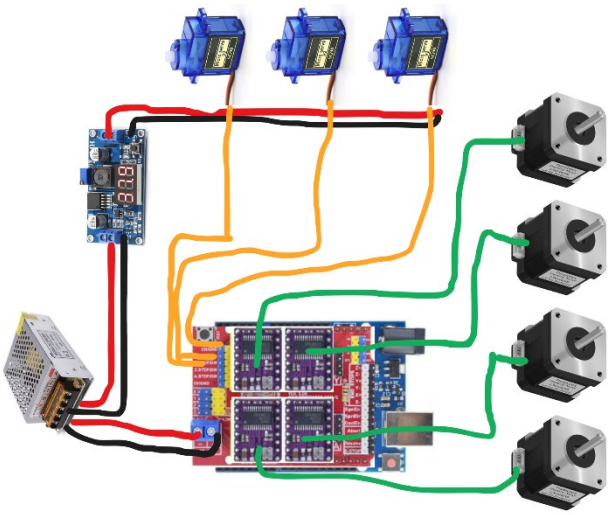

Figura 15. Diagramma del circuito.

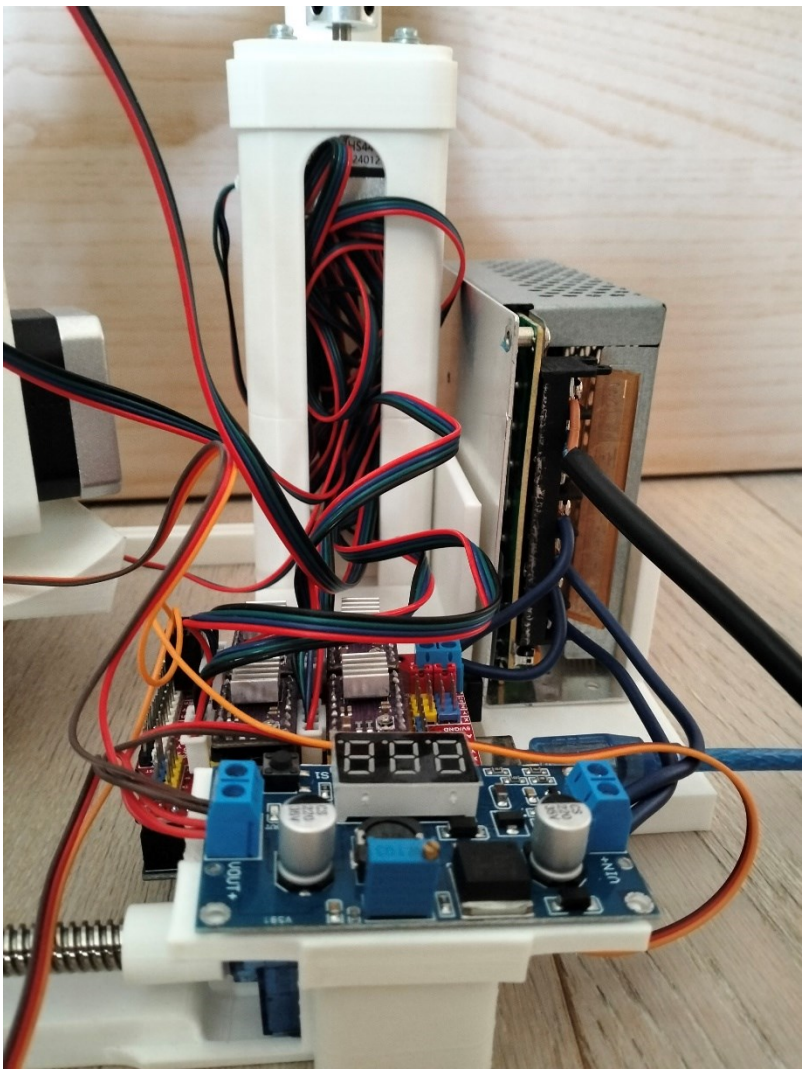

Figura 16. Circuito del prototipo.

4.3 Tecniche di Stampa 3D: Slicing, Bridging, Pause and Insert

Per la rapida prototipazione del manipolatore sono state utilizzate particolari tecniche nell'ambito della stampa 3D. La stampante utilizzata è il modello Ender 3 v2 di Creality, i pezzi da stampare sono stati progettati con il software Autodesk Fusion360 e successivamente trasformati in g-code tramite il software di slicing Cura.

1. Slicing esclusivo.

Per poter stampare in 3D un modello, è necessario che alla stampante venga fornito il codice che descrive il path che l'estrusore deve seguire. Questo compito viene eseguito dal sofware di slicing che taglia il modello in tante fette in modo che la stampante proceda a strati. Per eseguire lo slicing di una parte diagonale all'orizzontale, esistono 3 possibilità di tolleranza: slicing inclusivo, slicing nel mezzo e slicing esclusivo.

Usando l'opzione di slicing inclusivo, ogni layer conterrà *almeno* tutto il volume originale. Ciò significa che, quando la superficie è inclinata, gli strati si espanderanno leggermente oltre il modello rendendo il volume totale maggiore del volume dell'originale.

Usando l'opzione di slicing nel mezzo, ogni layer sarà il più possibile vicino alla superficie originale. Infatti, per le superfici inclinate, gli strati si espanderanno oltre il modello in alcune zone mentre in altre saranno leggermente schiacciati verso l'interno. In totale, il volume totale si avvicinerà il più possibile all'originale.

Infine, usando l'opzione di slicing esclusivo, ogni layer sarà sempre contenuto nel volume del modello. Pertanto, quando la superficie è inclinata, gli strati saranno sempre più piccoli del modello rendendo il volume totale minore del volume dell'originale.

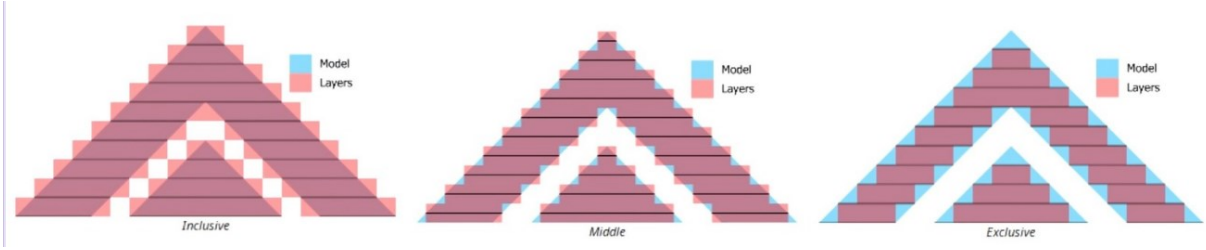

Figura 17. **Sinistra:** slicing inclusivo. **Centro**: slicing nel mezzo. **Destra**: slicing esclusivo.

La maggior parte dei componenti strutturali del prototipo sono stati assemblati senza l'uso di viti per motivi di rapidità. Sono spesso state utilizzate invece tecniche di incastro prendendo spunto dai giunti più comunemente usati nella lavorazione del legno. È qui che entra in gioco l'importanza delle 3 opzioni di tolleranza durante la fase di slicing. La procedura utilizzata per l'assemblamento del prototipo, infatti, sfrutta l'opzione di slicing esclusivo per stampare pezzi di volume leggermente minore così che possano formare un incastro solido e senza gioco. In questo modo, inoltre, si risparmia diverso tempo nella progettazione dei modelli dove non è necessario aggiungere manualmente una tolleranza di 0.1mm su ogni superficie del volume che fa parte dell'incastro.

2. Bridging

La tecnica bridging consiste nel permettere all'estrusore, attraverso specifiche impostazioni, di stampare espandendo il filamento orizzontalmente senza l'utilizzo di supporti, di fatto stampando "per aria". Questo permette, dove possibile, di ridurre drasticamente i tempi di stampa e (non dovendo stampare supporti).

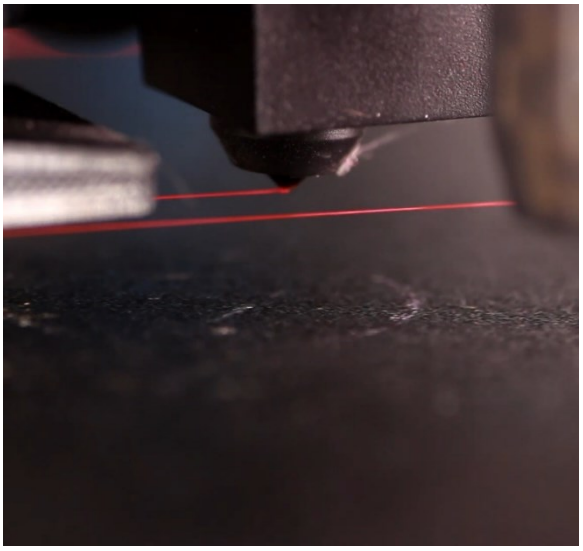

Figura 18.
L'estrusore crea il primo layer del ponte.

Per poter usare questa tecnica, l'estrusore deve stampare il primo layer ad una velocità ridotta, non superiore a 10 mm/s, con la massima attivazione delle ventole di raffreddamento. In questo modo il materiale si solidifica il prima possibile. Una volta completato il primo layer di base, l'estrusore continua con il secondo alle stesse condizioni, rendendo più spesso il ponte. Infine, dal terzo layer si può ricominciare a stampare sopra i precedenti layer a velocità standard.
Il software di slicing provvede a creare, per i primi layer del ponte, un path composto da traiettorie sempre rettilinee in modo da raggiungere il prima possibile un sostegno solido.

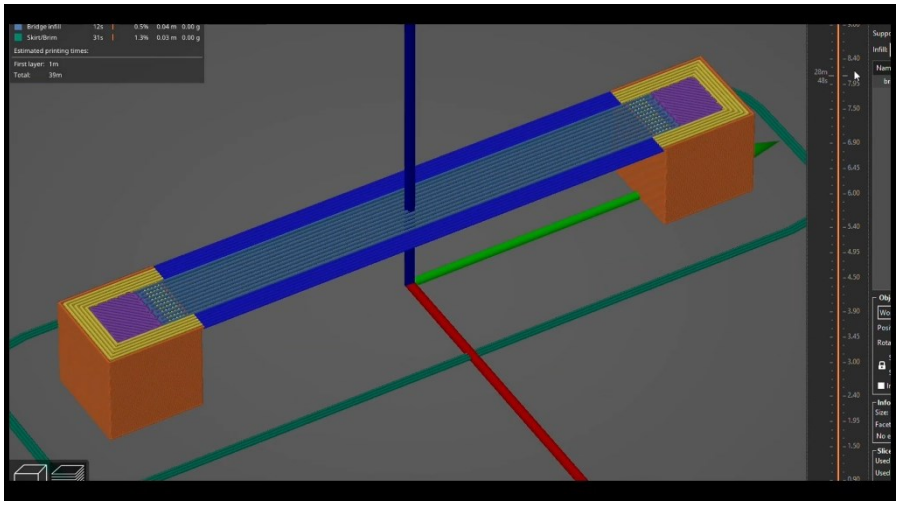

Figura 19.
Il path dell'estrusore, generato dal software di slicing, per creare il primo layer del ponte.

3. Pause and Insert

L'ultima tecnica di stampa 3D utilizzata nella produzione del prototipo viene chiamata "Pause and Insert". Il nome indica esattamente in cosa consiste: si interrompe la stampa al momento opportuno e si inserisce un oggetto così che a stampa completata si trovi all'interno del pezzo stampato. Nel caso del prototipo, per esempio, è stato inserito uno dei motori passo-passo all'interno di un pezzo dove, senza usare questa tecnica, non sarebbe potuto entrare.

Figura 20. Esempio della tecnica "Pause and Insert". Per stampare il resto dello stesso pezzo senza utilizzare supporti, che avrebbero ingombrato lo spazio dove è stato inserito il motore, è stata utilizzata anche la tecnica di Bridgring.

4.4. GUI Processing

Per controllare il prototipo in cinematica diretta, è stata sviluppata una graphical user interface (GUI). L'interfaccia presenta uno slider per il controllo di ciascun motore e visualizza in tempo reale la posizione della punta dell'ago calcolata con le formule della cinematica diretta.
Gli slider sono stati implementati utilizzando la libreria "ControlP5". Il codice è strutturato nella forma setup-loop dove la funzione di setup viene eseguita una volta all'avvio e la funzione loop viene eseguita ciclicamente.

```
130   updateData();
131
132   if (slider1Previous != j1Slider || slider2Previous != j2Slider ||
133     j1 = round(cp5.getController("j1Slider").getValue());
134     j2 = round(cp5.getController("j2Slider").getValue());
135     j3 = round(cp5.getController("j3Slider").getValue());
136     z = round(cp5.getController("zSlider").getValue());
137     n = round(cp5.getController("needleValue").getValue());
138     x = round(cp5.getController("xSlider").getValue());
139     y = round(cp5.getController("ySlider").getValue());
140
141     forwardKinematics();
142     myPort.write(data);            // Send data to Arduino
143   }
```

Attraverso queste righe di codice, all'interno della funzione loop, il programma controlla costantemente se avvengono modifiche (da parte dell'user) ai valori degli slider. Se questo succede, viene calcolata la nuova posizione della punta dell'ago tramite la funzione "forwardKinematics", e le nuove posizioni di ciascun motore vengono inviate allo script Arduino. Lo script completo è presente nell'Annex.

4.4 Sketch Arduino

Una volta ricevuti i dati di controllo dalla graphical user interface, lo sketch caricato sulla scheda Arduino Uno procede a comandare ciascun motore in modo opportuno. Per i motori passo-passo è stata utilizzata la libreria “AccelStepper”, per i servomotori la libreria “Servo”.

```
61    // move joint 1
62    int j1steps = round(j1 / 0.1125);   // 1 step = 1.8°/16 = 0.1125°
63    if (j1 > 1) {
64      stepper1.moveTo(-j1steps);
65      while (stepper1.distanceToGo() != 0) {
66        stepper1.run();
67      }
68    }
```

Con queste righe di codice, per esempio, Arduino riceve da Processing l’angolo target verso cui deve muovere lo stepper (“j1”). Quindi, calcola il numero di step corrispondenti a tale angolo (“j1steps”), tenendo conto della modalità microstepping con rapporto 1/16 settata sul driver. Infine, muove il motore di tanti step quanti sono necessari a raggiungere la posizione target.

Conclusioni

La progettazione e lo sviluppo di un Robot per Venipuntura che possa inserirsi nel mercato e sostituire quasi del tutto un operatore sanitario rappresenta un importante passo avanti nell'automatizzazione delle procedure cliniche più comuni. Tuttavia, è necessario che i pazienti forniscano approvazione e consenso all'utilizzo di tecnologie simili. Pertanto, non avendo ancora raggiunto a pieno questo obiettivo, la ricerca deve continuare lo sviluppo e ottenere risultati sempre più convincenti.

Il prototipo presentato presenta evidenti lacune per quanto riguarda la precisione e la completezza del progetto ma, nonostante ciò, può essere un buon punto di partenza nello sviluppo del manipolatore di un valido Robot per Venipuntura.

In conclusione, questa tesi fornisce comunque ottimi riferimenti per i vari organi del robot come il sistema di imaging infrarossi e ad ultrasuoni, i sistemi di sicurezza, e la sintesi meccanica del manipolatore a 7 gradi di libertà.

Bibliografia

1. McCann M, Einarsdottir H, Waeleghem JPV, Murphy F, Sedgewick J. Vascular access management, 1: an overview. J Ren Care. 2008; 34(2):77–84. [PubMed: 18498572]
2. Walsh G. Difficult peripheral venous access: Recognizing and managing the patient at risk. J. Assoc. Vascular Access. 2008; 13(4):198–203.
3. Maecken T, Grau T. Ultrasound imaging in vascular access. Critical Care Med. May; 2007 35(5 Suppl): S178–85. [PubMed: 17446777]
4. Katsogridakis Y, Seshadri R, Sullivan C, Waltzman ML. Veinlite transillumination in the pediatric emergency department: A therapeutic interventional trial. Pediatric Emergency Care. 2008; 24(2):83–88. [PubMed: 18277843]
5. Perry AM, Caviness AC, Hsu DC. Efficacy of a near-infrared light device in pediatric intravenous cannulation: A randomized controlled trial. Pediatric Emergency Care. 2011; 27(1):5–10. [PubMed: 21178814]
6. Juric S, Flis V, Debevc M, Holzinger A, Zalik B. Towards a low-cost mobile subcutaneous vein detection solution using near-infrared spectroscopy. Sci. World J. Jan.2014 2014:1–15.
7. Graaff JC, Cuper NJ, Mungra RA, Vlaardingerbroek K, Numan SC, Kalkman CJ. Near-infrared light to aid peripheral intravenous cannulation in children: A cluster randomised clinical trial of three devices. Anaesthesia. 2013; 68(8):835–845. [PubMed: 23763614]
8. Mela, C.A.; Lemmer, D.P.; Bao, F.S.; Papay, F.; Hicks, T.; Liu, Y. Real-time dual-modal vein imaging system. Int. J. Comput. Assist. Radiol. Surg. 2019, 14, 203–213.
9. Peled, G.; Halak, M.; Blechman, Z. Peripheral vein locating techniques. Imaging Med. 2016, 8
10. Mark D. Francisco, Wen-Fan Chen 4, Cheng-Tang Pan, Ming-Cheng Lin, Zhi-Hong Wen, Chien-Feng Liao e Yow-Ling Shiue. Competitive Real-Time Near Infrared (NIR) Vein Finder Imaging Device to Improve Peripheral Subcutaneous Vein Selection in Venipuncture for Clinical Laboratory Testing. Multidisciplinary Digital Publishing Institute, 2021. [PubMed: 33808493].

11. Keith L. Dorrington, Matthew C. Frise, Lessons of the month 1: Learning from Harvey; improving blood-taking by pointing the needle in the right direction. Clin Med (Lond), 2019. [PubMed: 31732596].
12. Bercoff, J. et al. Ultrafast compound doppler imaging: Providing full blood flow characterization. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. 58 (1), 134-147 (2011)
13. Montaldo, G., Tanter, M., Bercoff, J., Benech, N., Fink, M. Coherent plane-wave compounding for very high frame rate ultrasonography and transient elastography IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control. 56 (3), 489-506 (2009).
14. Marek Wartenberg, Joseph Schornak, Katie Gandomi, Paulo Carvalho, Chris Nycz, Niravkumar Patel, Iulian Iordachita, Clare Tempany, Nobuhiko Hata, Junichi Tokuda, Gregory S. Fischer. Closed-Loop Active Compensation for Needle Deflection and Target Shift During Cooperatively Controlled Robotic Needle Insertion. Ann Biomed Eng. 2018.
15. Boer TD, Steinbuch M, Neerken S, Kharin A. Laboratory study on needle–tissue interaction: Towards the development of an instrument for automatic venipuncture. J. Mech. Med. Biol. 2007; 7(3):325–335.
16. 21. Chen A, Nikitczuk K, Nikitczuk J, Maguire T, Yarmush ML. Portable robot for autonomous venipuncture using 3D near infrared image guidance. Technology. 2013; 01(01):1–16.

Annex

1. Script MatLab

###Main###

```
%% VENIPUNCTURE Robot, 4DoF Serial Arm - Inverse Kinematics Analysis
% 2024 by alessandro.baschiera@studenti.unipd.it
clc, clear all, close all

%% Inverse Kinematics Analysis

% geometric dimensions [mm]
max_height = 400;
framepinheight = 350; % in starting position
q1_offset = 70; % framepinheight with q1 = 0
a1 = 52.50;
a2 = 140;
a3 = 100;  % needle lenght in retracted state
O = [0, 0, framepinheight];  % frame pin (cylindrical coordinates)
S = [0, 0, (framepinheight - a1 - a2 - a3)]; % starting point (cylindrical coordinates)

% define end effector trajectory
xP_target = 30; % x target coordinate [mm]
yP_target = 30; % y target coordinate [mm]
zP_target = 50; % z target coordinate [mm]
q0_target = 30; % insertion angle with respect to x axis [deg]; theta coordinate for all points
q3_target = 20; % insertion angle with respect to rho axis (5°-30° depending on vein depth)
[deg]
q4_target = 40; % needle extension [mm]

q0_target = deg2rad(q0_target);
q3_target = pi + deg2rad(q3_target);
```

```
np = 50;
time = linspace(0, 5, np);

q0 = linspace(0, q0_target, length(time));  % [rad]
q3 = linspace(((3/2)*pi), q3_target, length(time)); % [rad]
q4 = linspace(0, q4_target, np); % [rad]

% target point (cylindrical coordinates)
[thetaP_target, rhoP_target, zP_target] = cart2pol(xP_target, yP_target, zP_target);
P_target = [thetaP_target, rhoP_target, zP_target];

% point C coordinates with respect to target point P
rhoC_P = -q4_target*cos(q3_target);
zC_P = -q4_target*sin(q3_target);
C_P = [q0_target, rhoC_P, zC_P]; % cylindrical coordinates
[xC_P, yC_P, zC_P] = pol2cart(q0_target, rhoC_P, zC_P); % cartesian coordinates
% point C coordinates with respect to origin
xC_target = xP_target + xC_P; % vector sum
yC_target = yP_target + yC_P;
zC_target = zP_target + zC_P;
[thetaC_target, rhoC_target, zC_target] = cart2pol(xC_target, yC_target, zC_target);
C_target = [thetaC_target, rhoC_target, zC_target]; % cylindrical coordinates

[xPv, yPv, zPv, thetaPv, rhoPv] = create_trajectory(np, S, P_target);
[xCv, yCv, zCv, thetaCv, rhoCv] = create_trajectory(np, S, C_target);



f1 = figure(1);
figure('OuterPosition',[1 200 600 600])
f2 = figure(2);

set(0, 'CurrentFigure', f2)
plot3(xPv,yPv,zPv, 'm')
hold on
axis equal
```

```
%% Inverse Kinematics solution

for k = 1:np

    % solving for C target, then needle will extend and reach P target
    [q1(k), q2(k)] = inverse_kinematics5(rhoCv(k), zCv(k), a1, a2, a3, q3(k), q1_offset);

    % theta coordinate for point B
    xBk(k) =  xPv(k) + (a2 * cos(q2(k))) * cos(q0(k)); % at instant k
    yBk(k) =  yPv(k) + (a2 * cos(q2(k))) * sin(q0(k));
    thetaBk(k) = atan2(yBk(k), xBk(k));

    % theta coordinate for point C
    xCk(k) =  xPv(k) + ((a2 - a3) * cos(q2(k))) * cos(q0(k));
    yCk(k) =  yPv(k) + ((a2 - a3) * cos(q2(k))) * sin(q0(k));
    thetaCk(k) = atan2(yCk(k), xCk(k));

    % theta coordinate for point P
    xPk(k) =  xP_target + ((a2 - a3 - q4(k)) * cos(q2(end))) * cos(q0_target);
    yPk(k) =  yP_target + ((a2 - a3 - q4(k)) * cos(q2(end))) * sin(q0_target);
    thetaPk(k) = atan2(yPk(k), xPk(k));

end

%% Animation

trajectory = [];

for k = 1:np

    set(0, 'CurrentFigure', f1)
    title('motors analysis')

    % motor q0 angle plot
```

```
subplot(3,2,1), cla
   plot(time(1:k), rad2deg(q0(1:k)), 'g');
   xlabel('time (s)'), xlim([time(1) time(end)])
   ylabel('angle (deg)')
   title('q0')
   grid on
   ylim([0 180]), yticks(0:45:180)

% motor q2 angle plot
subplot(3,2,3), cla
   plot(time(1:k), rad2deg(q2(1:k)), 'b');
   xlabel('time (s)'), xlim([time(1) time(end)])
   ylabel('angle (deg)')
   title('q2')
   grid on
   ylim([180 450]), yticks(180:90:540)

% motor q1 extension plot
subplot(3,2,2), cla
   plot(time(1:k), q1(1:k),'r')
   xlabel('time (s)'), xlim([time(1) time(end)])
   ylabel('extension (mm)')
   title('q1')
   grid on

% motor q3 angle plot
subplot(3,2,4), cla
   plot(time(1:k), rad2deg(q3(1:k)), 'm')
   xlabel('time (s)'), xlim([time(1) time(end)])
   ylabel('angle (deg)')
   title('q3')
   grid on
   ylim([180 270])

% motor q5 extension plot
```

```
    subplot(3,2,5), cla
        plot(time(1:k), xPv(1:k),'y')
        xlabel('time (s)'), xlim([time(1) time(end)])
        ylabel('extension (mm)')
        title('q5')
        grid on

    % motor q6 extension plot
    subplot(3,2,6), cla
        plot(time(1:k), yPv(1:k),'k')
        xlabel('time (s)'), xlim([time(1) time(end)])
        ylabel('extension (mm)')
        title('q6')
        grid on



    % ARM points
    set(0, 'CurrentFigure', f2)
    cla, axis equal,  hold on
    xlim([-(a2+a3)+xP_target a2+a3+xP_target]), ylim([-(a2+a3)+yP_target
a2+a3]+yP_target), zlim([0 max_height])
    xlabel('x'), ylabel('y'), zlabel('z')
    title('ARM 3d simulation')

        %forward kinematics (in rho and z coordinates 2D plane)

        O2 = complex(rhoPv(k), (q1_offset + q1(k)));
        [thetaO, rhoO, zO] = cylindrical(thetaPv(k), O2);
        O = [thetaO, rhoO, zO];

        A2 = O2 + a1 * exp(1i * (3/2) * pi);
        [thetaA, rhoA, zA] = cylindrical(thetaPv(k), A2);
        A = [thetaA, rhoA, zA];

        B2 = A2 + a2 * exp(1i * q2(k));
```

```
        [thetaB, rhoB, zB] = cylindrical(thetaBk(k), B2);
        B = [thetaB, rhoB, zB];

        C2 = B2 + a3 * exp(1i * q3(k));
        [thetaC, rhoC, zC] = cylindrical(thetaCk(k), C2);
        C = [thetaC, rhoC, zC];

        % switch to cartesian coordinates to plot
        [xO, yO, zO] = pol2cart(O(1), O(2), O(3));
        xOv(k) = xO; yOv(k) = yO; zOv(k) = zO;
        O = [xO; yO; zO];
        [xA, yA, zA] = pol2cart(A(1), A(2), A(3));
        A = [xA; yA; zA];
        [xB, yB, zB] = pol2cart(B(1), B(2), B(3));
        B = [xB; yB; zB];
        [xC, yC, zC] = pol2cart(C(1), C(2), C(3));
        C = [xC; yC; zC];

        % 3D plot
        ARM = [O A B C];
        plot3(ARM(1,:), ARM(2,:), ARM(3,:), '-o', 'Color','b','LineWidth',2, ...
           'MarkerEdgeColor','k', 'MarkerSize',10, 'MarkerFaceColor','auto')
        trajectory = [trajectory C];
        plot3(trajectory(1,:), trajectory(2,:), trajectory(3,:),'r')
        plot3(xOv,yOv,zOv, 'y')



        drawnow
        pause(0.01)
end

go = input("press enter to launch needle");

for k = 1:np
```

```
    % needle tip
    P2 = C2 + q4(k) * exp(1i * q3_target);
    [thetaP, rhoP, zP] = cylindrical(thetaPk(k), P2);
    [xP, yP, zP] = pol2cart(thetaP, rhoP, zP);
    P = [xP; yP; zP];

    cla
    plot3(ARM(1,:), ARM(2,:), ARM(3,:), '-o', 'Color','b','LineWidth',2, ...
        'MarkerEdgeColor','k', 'MarkerSize',10, 'MarkerFaceColor','auto')
    plot3(trajectory(1,:), trajectory(2,:), trajectory(3,:),'r')

    needle = [C P];
    plot3(needle(1,:), needle(2,:), needle(3,:), 'Color', 'm','LineWidth',2)
    plot3(xOv,yOv,zOv, 'y')



    drawnow
    pause(0.02)
end
###inverse_kinematics####
%% VENIPUNCTURE Robot, 5DoF end effector

% calculates motors angle in relation to point C (end effector) coordinates
% INPUT -------
%       rhoP, zP  - coordinates of point P, target point
%       a1, a2, a3  - geometric dimensions
%       q3 [rad]  -  needle insertion angle
%       q4 [mm]  -  needle extension
%       q1_offset [mm]  -  framepinheight with q1=0
% OUTPUT -------
%       q1 [mm] - framepinheght
%       q2 [rad] - motor angle

function [q1, q2] = inverse_kinematics5(rhoP, zP, a1, a2, a3, q3, q1_offset)
```

```
    rhoB = rhoP - a3*cos(q3);
    zB = zP - a3*sin(q3);

    q2 = atan2(zB, rhoB);
    if q2 < 0
        q2 = q2 + 2*pi;
    end
    q2 = 2*pi - q2;

    q1 = zB - a2*sin(q2) + a1 - q1_offset;
```

###cylindrical###

```
%% VENIPUNCTURE Robot, 5DoF end effector

% transfers 2D complex notation coordinates in
% 3D cylindrical coordinates
% INPUT -----
%     thetaA - first cylindrical coordinate
%     A - complex number representing point A in 2D
% OUTPUT: thetaA, rhoA, zA - cylindrical coordinates of point A

function [thetaA, rhoA, zA] = cylindrical(thetaA, A)
    rhoA = real(A);
    zA = imag(A);
```

###create_trajectory###

```
%% VENIPUNCTURE Robot, 5DoF end effector

% creates linear trajectory in cartesian and cylindrical coordinates from point A to point B
% INPUT------
%     time [linear space]
%     q0 [linear space] - theta coordinate for all points
%     A [cylindrical coordinates] - starting point
```

```
%     B [cylindrical coordinates] - target point
% OUTPUT ------
%     xPv [linear space] - x coordinate over time
%     yPv [linear space] - y coordinate over time
%     zPv [linear space] - z coordinate over time
%     thetaPv [linear space] - theta cylindrical coordinate over time
%     rhoPv [linear space] - rho cylindrical coordinate over time


function [xPv, yPv, zPv, thetaPv, rhoPv] = create_trajectory(np, A, B)

    thetaA = A(1);
    thetaB = B(1);
    rhoA = A(2);
    rhoB = B(2);
    zA = A(3);
    zB = B(3);

    thetaPv = linspace(thetaA, thetaB, np);
    rhoPv = linspace(rhoA, rhoB, np);
    zPv = linspace(zA, zB, np);

    [xPv, yPv, zPv] = pol2cart(thetaPv, rhoPv, zPv);
```

2. Script GUI Processing

```
import processing.serial.*;
import controlP5.*;

Serial myPort;
ControlP5 cp5;

int j1Slider = 0;
```

```
int j2Slider = 0;
int j3Slider = 0;
int zSlider = 0;
int needleValue = 0;
int xSlider = 0;
int ySlider = 0;


int slider1Previous = 0;
int slider2Previous = 0;
int slider3Previous = 0;
int sliderzPrevious = 0;
int needleValuePrevious = 0;
int sliderxPrevious = 0;
int slideryPrevious = 0;

float q1_offset = 100;  //[mm]
float q1, q4;           //[mm]
float q0, q2, q3;       //[deg]
float a1 = 52.5;        //[mm]
float a2 = 80;
float a3 = 104;
float j1, j2, j3, x, y, z, n;
float rhoP;
float xP = 0;
float yP = 0;
float zP = 0;

String data;

void setup() {
  size(960, 800);
  myPort = new Serial(this, "COM4", 115200);

  cp5 = new ControlP5(this);
```

```
PFont pfont = createFont("Arial", 25, true);
ControlFont font = new ControlFont(pfont, 22);

// Joint1 control
cp5.addSlider("j1Slider")
  .setPosition(110, 190)
  .setSize(270, 30)
  .setRange(-75, 75)       // [deg]
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("J1");

// Joint2 control
cp5.addSlider("j2Slider")
  .setPosition(110, 315)
  .setSize(270, 30)
  .setRange(0, 60)      // [deg]
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("J2");

// Joint3 control
cp5.addSlider("j3Slider")
  .setPosition(110, 440)
  .setSize(270, 30)
  .setRange(-45, 45)      // [deg]
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("J3");

// Z axis control
cp5.addSlider("zSlider")
  .setPosition(110, 565)
  .setSize(270, 30)
  .setRange(0, 170)         // [mm]
```

```
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("Z");

  // Needle control
  cp5.addSlider("needleValue")
    .setPosition(110, 690)
    .setSize(270, 30)
    .setRange(0, 40)        // [mm]
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("Needle");

  // X axis control
  cp5.addSlider("xSlider")
    .setPosition(500, 565)
    .setSize(270, 30)
    .setRange(-50, 50)      // [mm]
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("X");

  // Y axis control
  cp5.addSlider("ySlider")
    .setPosition(500, 690)
    .setSize(270, 30)
    .setRange(-50, 50)      // [mm]
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("Y");
}

void draw() {
  background(#F2F2F2);
  textSize(26);
```

```
  fill(33);
  text("Venipuncture Robot Control", 260, 60);
  textSize(22);
  text("Joint 1", 35, 250);
  text("Joint 2", 35, 375);
  text("Joint 3", 35, 500);
  text("Axis Z", 35, 625);
  text("Needle", 35, 750);
  text("Axis X", 35, 875);
  text("Axis Y", 35, 1000);

  fill(33);
  textSize(32);
  text("X: " + nf(xP, 0, 2), 500, 290);
  text("Y: " + nf(yP, 0, 2), 650, 290);
  text("Z: " + nf(zP, 0, 2), 800, 290);

  updateData();

  if (slider1Previous != j1Slider || slider2Previous != j2Slider || slider3Previous != j3Slider ||
sliderzPrevious != zSlider || needleValuePrevious != needleValue || sliderxPrevious != xSlider
|| slideryPrevious != ySlider) {
    j1 = round(cp5.getController("j1Slider").getValue());
    j2 = round(cp5.getController("j2Slider").getValue());
    j3 = round(cp5.getController("j3Slider").getValue());
    z = round(cp5.getController("zSlider").getValue());
    n = round(cp5.getController("needleValue").getValue());
    x = round(cp5.getController("xSlider").getValue());
    y = round(cp5.getController("ySlider").getValue());

    forwardKinematics();
    myPort.write(data);           // Send data to Arduino
  }

  slider1Previous = j1Slider;
```

```
  slider2Previous = j2Slider;
  slider3Previous = j3Slider;
  sliderzPrevious = zSlider;
  needleValuePrevious = needleValue;
  sliderxPrevious = xSlider;
  slideryPrevious = ySlider;
}

void forwardKinematics() {
  q0 = j1;
  if (q0 < 0) {
    q0 = q0 + 360;
  }
  q1 = z;
  q2 = j2 - 45;
  if (q2 < 0) {
    q2 = q2 + 360;
  }
  q3 = 180 - j3 + j2;
  q4 = needleValue;
  rhoP = a2*cos(q2) + (a3+q4)*cos(q3);
  zP = q1_offset + q1 - a1 + a2*sin(q2) + (a3+q4)*sin(q3);
  xP = rhoP*cos(q0) + x;
  yP = rhoP*sin(q0) + y;

}

void updateData() {
  data = String.format("%d,%d,%d,%d,%d,%d,%d\n", (j1Slider-slider1Previous), (j2Slider-
slider2Previous), (j3Slider-slider3Previous), (zSlider-sliderzPrevious), (needleValue-
needleValuePrevious), (xSlider-sliderxPrevious), (ySlider-slideryPrevious));
}

void serialEvent (Serial myPort) {
```

```
  String inString = myPort.readStringUntil('\n');
  if (inString != null) {
    print(inString);
  }
}
```

3. Script Arduino

```
#include <AccelStepper.h>
#include <Servo.h>

// Stepper NEMA17 controlled through CNC Shield V3 with DRV8825 drivers (1/16 microstepping)
AccelStepper stepper1(1, 2, 5);  // X module on shield
AccelStepper stepper2(1, 3, 6);  // Y module on shield
AccelStepper stepper3(1, 4, 7);  // Z module on shield
AccelStepper stepper4(1, 12, 13);  // A module on shield = Robot's Z axis

Servo needleServo;  // Servo360 to move Needle
Servo ServoX;       // Servo360 to move x axis
Servo ServoY;       // Servo360 to move y axis

int servospeedforward = 80;
int servospeedbackward = 100;
float servorevpersecond = 0.33;

void setup() {
  Serial.begin(115200);

  // enable pin to be be able to move steppers
  pinMode(8, OUTPUT);
  digitalWrite(8, LOW);

  // Config Stepper speed and acceleration
```

```
  stepper1.setMaxSpeed(200);
  stepper1.setAcceleration(100);
  stepper2.setMaxSpeed(200);
  stepper2.setAcceleration(100);
  stepper3.setMaxSpeed(200);
  stepper3.setAcceleration(100);
  stepper4.setMaxSpeed(4000);
  stepper4.setAcceleration(2000);

  needleServo.attach(A1);  // Attach Servo to pin A1 (Needle control)
  ServoX.attach(A2);  // Attach Servo to pin A2 (X axis control)
  ServoY.attach(A0);  // Attach Servo to pin A0 (Y axis control)
}

void loop() {

  // extract received data
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    int j1 = data.substring(0, data.indexOf(',')).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int j2 = data.substring(0, data.indexOf(',')).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int j3 = data.substring(0, data.indexOf(',')).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int z = data.substring(0, data.indexOf(',')).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int needle = data.substring(0, data.indexOf(',') + 1).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int servoXVal = data.substring(0, data.indexOf(',')).toInt();
    data = data.substring(data.indexOf(',') + 1);
    int servoYVal = data.toInt();


    // move joint 1
```

```
int j1steps = round(j1 / 0.1125);   // 1 step = 1.8°/16 = 0.1125°
if (j1 > 1) {
  stepper1.moveTo(-j1steps);
  while (stepper1.distanceToGo() != 0) {
    stepper1.run();
  }
}


// move joint 2
int j2steps = round(j2 / 0.1125);   // 1 step = 1.8°/16 = 0.1125°
if (j2 > 1) {
  stepper2.moveTo(-j2steps);
  while (stepper2.distanceToGo() != 0) {
    stepper2.run();
  }
}

// move joint 3
int j3steps = round(j3 / 0.1125);   // 1 step = 1.8°/16 = 0.1125°
if (j3 > 1) {
  stepper3.moveTo(-j3steps);
  while (stepper3.distanceToGo() != 0) {
    stepper3.run();
  }
}

// move Z axis
float zsteps = z * ((200 * 16) / 2);    // 200*16 steps = full rev = 2mm
stepper4.moveTo(-zsteps);
while (stepper4.distanceToGo() != 0) {
  stepper4.run();
}
```

```
  // Servo [mm] to [ms] (of running) conversion
  float needle_ms = ((abs(needle) / 4) / servorevpersecond) * 1000;         // 4 = screw lead =
4mm per revolution
  float servoXVal_ms = ((abs(servoXVal) / 4) / servorevpersecond) * 1000;
  float servoYVal_ms = ((abs(servoYVal) / 4) / servorevpersecond) * 1000;

  // Move servo motors
    if (needle > 0) {
      needleServo.write(servospeedforward);    // run forward
      Serial.println("servo needle running forward");
      delay(needle_ms);
      Serial.println("servo needle stopped");
      needleServo.write(90);                   // stop
    }
    if (needle < 0) {
      needleServo.write(servospeedbackward);    // run backward
      Serial.println("servo needle running backwards");
      delay(needle_ms);
      Serial.println("servo needle stopped");
      needleServo.write(90);                   // stop
    }

    if (servoXVal > 0) {
      ServoX.write(servospeedforward);    // run forward
      Serial.println("servo X running forward");
      delay(servoXVal_ms);
      Serial.println("servo X stopped");
      ServoX.write(90);                   // stop
    }
    if (servoXVal < 0) {
      ServoX.write(servospeedbackward);    // run backward
      Serial.println("servo X running backward");
      delay(servoXVal_ms);
      Serial.println("servo X stopped");
      ServoX.write(90);                   // stop
```

```
    }

    if (servoYVal > 0) {
      ServoY.write(servospeedforward);    // run forward
      Serial.println("servo Y running forward");
      delay(servoYVal_ms);
      Serial.println("servo Y stopped");
      ServoY.write(90);                   // stop
    }
    if (servoYVal < 0) {
      ServoY.write(servospeedbackward);    // run backward
      Serial.println("servo Y running backward");
      delay(servoYVal_ms);
      Serial.println("servo Y stopped");
      ServoY.write(90);                   // stop
    }

  }
}
```