Guida all'utilizzo del robot Panda

Giulio Luongo

Novembre 2021

Indice

1 Introduzione

Il presente documento è una revisione del pdf "*Guida operativa sintetica all'utilizzo del robot Panda - Franka Emika*" realizzato dal collega Luigi Vetrella durante il suo lavoro di tesi. Tale documento raccoglie le informazioni utili per il corretto utilizzo del cobot Panda della Franka Emika. Esso non è da considerarsi in alcun modo sostituibile al manuale di utilizzo originale fornito dalla casa costruttrice del robot, disponibile in formato cartaceo o comunque reperibile sulla piattaforma Franka World nella sezione Hub/Resources. È suggerita, quindi, un'attenta ed approfondita lettura dell'handbook allegato al manipolatore nella confezione di consegna prima di intraprendere qualsiasi azione con il robot. All'interno del documento, tutti i riferimenti espliciti ad eventuali indirizzi IP, user-id, password e dettagli simili sono da riferirsi al set-up del robot Panda attualmente (Novembre 2021) presente nel laboratorio di robotica dell'Università degli Studi della Campania Luigi Vanvitelli.

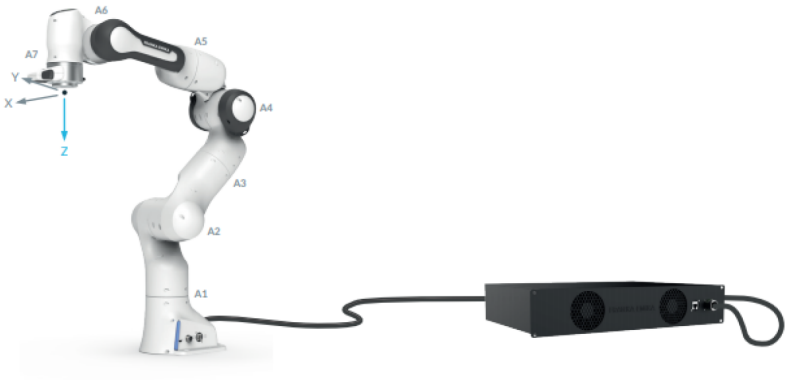

Figure 1: Cobot Panda - Franka Emika

2 Setup workstation

Prima di poter programmare e mettere in movimento il robot Panda è necessario configurare la workstation con cui questo si interfaccerà. La workstation comunica con il robot attraverso un Controller intermedio, mostrato in figura 2, a cui è collegata per mezzo di un cavo ethernet. La Franka Emika richiede alla workstation due requisiti minimi:

1. **Sistema Operativo**: Linux con patch PREEMPT_RT o Windows 10[1].
2. **Scheda di rete**: 100BASE-TX. Il soddisfacimento di tale richiesta permette di fare affidamento su una connessione dati a 100 Mbps tra il Controller del robot e la workstation.

Tali requisiti minimi sono dovuti al fatto che la libreria software *libfranka*, sviluppata da Franka Emika e utilizzata per programmare il robot, richiede imperativamente che l'intero loop di controllo sia eseguito con una frequenza di 1 kHz. Per assicurare il rispetto di tale vincolo il processo deve essere schedulato con priorità **realtime** sotto kernel *PREEMPT_RT*. Dire che l'intero loop di controllo deve essere eseguito con una frequenza di 1 kHz è equivalente ad affermare che la somma dei seguenti intervalli di tempo debba essere minore di 1ms:

- Round trip time (RTT) dei messaggi scambiati tra workstation e Controller.
- Tempo di calcolo della legge di controllo.
- Tempo necessario al Controller per processare i dati inviati dalla workstation e inviare il comando al robot.

Se la somma di questi intervalli di tempo non dovesse rispettare il vincolo allora il pacchetto viene scartato dal controller. A seguito di 20 pacchetti scartati il robot viene fermato con errore communication_constraints_violation. Al momento della stesura di questo documento (Novembre 2021) la workstation **#####**[2] del laboratorio ha una scheda di rete adeguata, ha già installata la versione patchata del kernel Linux e la connessione con il Controller è già stata configurata. Ciò nonostante, per completezza, nella seguente sezione ver-ranno spiegate le procedure per installare la patch realtime del kernel Linux e configurare la connessione con il Controller.

2.1 Setup del kernel realtime

La libreria libfranka è progettata per funzionare su tutte le distribuzioni Linux ma il supporto è assicurato solo per le distribuzioni Ubuntu 16.04 LTS Xenial Xerus, Ubuntu 18.04 LTS Bionic Beaver e Ubuntu 20.04 LTS Focal Fossa.

[1] E' in fase sperimentale
[2] Coppia user-password: ##### #####.

Dando per scontato che sulla workstation sia già presente una distribuzione di Linux, in dual-boot o meno, i passi da seguire per installare la patch realtime sono i seguenti:

1. Aprire un terminale e installare le dipendenze necessarie:

   ```
   sudo apt-get install build-essential bc curl ca-certificates gnupg2
   libssl-dev lsb-release libelf-dev bison flex dwarves vim
   ```

2. Aggiornare `apt-get update`

3. Successivamente, decidere quale versione della patch kernel utilizzare. È consigliato utilizzare una patch quanto più vicina possibile alla versione del kernel attualmente installata [3]. Le patch realtime sono disponibili solo per alcune versioni del kernel Linux, per scoprire quali andare sul sito ufficiale. Una volta scelta la versione, utilizzare il comando curl per scaricare i source files (possono essere anche scaricati manualmente dal sito ufficiale: https://mirrors.edge.kernel.org/pub/linux/kernel/):

   ```
   curl -SLO https://www.kernel.org/pub/linux/kernel/v5.x/
   linux-5.4.19.tar.xz

   curl -SLO https://www.kernel.org/pub/linux/kernel/projects/
   rt/5.4/older/patch-5.4.19-rt10.patch.xz
   ```

   Ovviamente il nome delle versioni deve essere cambiato in funzione della propria versione del kernel Linux.

4. Una volta che i file sono stati scaricati bisogna decomprimerli:
   `xz -d *.xz` Ed estrarre il source code e applicare la patch:

   ```
   tar xf linux-*.tar
   cd linux-*/
   patch -p1 < ../patch-*.patch
   ```

5. Successivamente, copiare la configurazione del kernel attualmente attiva come default configuration per il realtime kernel:
   `cp -v /boot/config-$(uname -r) .config`

6. Lanciare quindi i comandi:
   ```
   make olddefconfig
   make menuconfig
   ```
   Il secondo comando aprirà un interfaccia navigabile con le freccette della tastiera. La prima cosa da fare è modificare il *preemption model*, ovvero il modello utilizzato dal sistema real time per gestire il rilascio delle risorse da parte dei processi. Navigando con le frecce della tastiera andare in

[3] Individuabile con il comando uname -r .

General Setup > *Preemption Model* e selezionare *Fully Preemptible Kernel (Real-Time)*. Dopo aver fatto ciò, tornare indietro[4] e navigare fino all'opzione *Cryptographic API* > *Certificates for signature checking*[5] > *Provide system-wide ring of trusted keys* > *Additional X.509 keys for default system keyring*, cancellare la stringa "debian/canonical-certs.pem" e premere invio.

7. Salvare la configurazione nel file .config (basta uscire dall'interfaccia).

8. Compilare il kernel:
   `make -j$(nproc) deb-pkg`
   Quest'operazione è piuttosto lunga quindi si consiglia di settare l'opzione multithread -j al massimo numero di CPU cores.

9. Finalmente, installare il package appena creato. Per installare:
   `sudo dpkg -i ../linux-headers-*.deb ../linux-image-*.deb`
   I nomi esatti dei file dipenderanno dall'environment su cui si sta lavorando ma in generale serve trovare gli headers e le images **senza il suffisso dbg**.

10. Riavviare il sistema. Il Grub boot menu dovrebbe ora consentire all'utente di scegliere la versione del kernel appena installata. Selezionare la versione che contiene RT nel nome. Per confermare la corretta installazione, una volta dentro digitare il comando *uname -a*: l'output del comando dovrebbe contenere la stringa PREEMPT RT e il numero di versione che è stato scelto. In più, come conferma aggiuntiva dovrebbe esistere il file /sys/kernel/realtime e dovrebbe contenere il numero 1.

11. Se il tutto funziona correttamente non resta che creare un gruppo a cui sia concesso settare priorità realtime ai propri processi.
    ```
    sudo addgroup realtime
    sudo usermod -a -G realtime $(whoami)
    ```
    Dopodiché, aggiungere i seguenti limiti al gruppo realtime in /etc/security/limits.conf:

    ```
    @realtime soft rtprio 99
    @realtime soft priority 99
    @realtime soft memlock 102400
    @realtime hard rtprio 99
    @realtime hard priority 99
    @realtime hard memlock 102400
    ```

    Una volta riavviata la workstation, questa sarà configurata correttamente per funzionare con la libreria libfranka di Franka Emika.

[4] cliccando due volte il tasto ESC
[5] Si trova alla fine della lista.

2.2 Network Configuration

Il secondo passo è configurare la connessione tra la workstation e il robot Panda. Buone performance di rete sono cruciali quando si controlla il robot attraverso il Controller FCI. Questa sottosezione descrive come configurare la rete per garantire buone performance. Il controller e la workstation devono essere configurate in modo da apparire sulla stessa sottorete. Il modo più semplice per ottenere questo risultato è assegnare indirizzi IP statici manualmente al controller e alla workstation. L'indirizzo IP del controller è stato assegnato, attraverso la sezione NETWORK dell'interfaccia *App Desk* descritta nella sezione 4, al valore 10.224.20.198\24. Alla workstation è stato assegnato l'indirizzo 10.224.20.197\24 attraverso l'interfaccia grafica di Ubuntu.

3 Avviamento e spegnimento del Robot

3.1 Avviamento

Supponendo che il manipolatore sia stato collegato correttamente all'alimentazione e saldamente installato sul banco di lavoro, per avviare il manipolatore è sufficiente attivare il Controller agendo sull'interruttore presente sulla parte anteriore (Power Switch in figura 2). La fase di avviamento è caratterizzata dal fatto che le luci di stato (vedere figura 3) sono di colore giallo e lampeggianti. La fine del del processo di avviamento viene notificata, invece, da luci di stato di colore giallo e fisse.

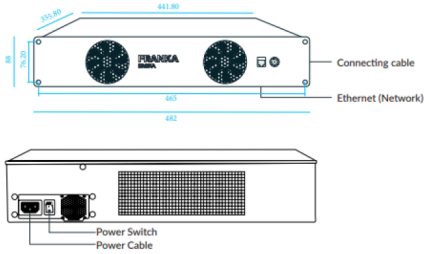

Figure 2: Controller FCI del robot Panda

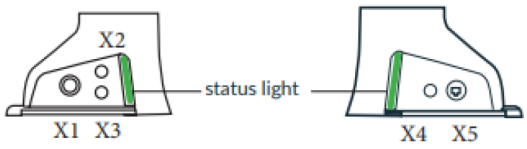

Figure 3: Luci di stato del robot Panda

Durante l'attività del robot, i colori che possono assumere le luci di stato sono i seguenti:

- Bianco *(Interazione)*: il manipolatore non è frenato ed è possibile un'interazione sicura con esso; In questo stato è possibile eseguire le modalità *Guiding Mode* e *Pilot Mode* che saranno illustrate nel capitolo successivo.
- Blu *(Attenzione)*: il manipolatore è abilitato al movimento e potrebbe muoversi da un momento all'altro. È necessario portarsi in tale stato quando si vuole far eseguire al manipolatore un algoritmo di controllo. Per portarsi in tale stato è sufficiente azionare uno dei due dispositivi di attivazione mostrati in figura 9.

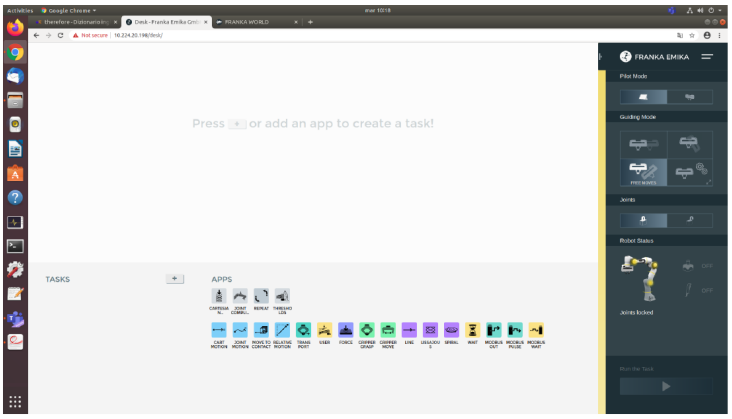

Figure 4: Interfaccia App Desk

- Verde *(Esecuzione automatica)*: il manipolatore sta eseguendo un task programmato indipendentemente;
- Giallo *(Locked)*: il freni del manipolatore sono bloccati;
- Rosa *(Conflitto)*: il manipolatore ha ricevuto segnali di attivazione contrastanti; Questo accade se vengono utilizzati contemporaneamente i due dispositivi di attivazione;
- Rosso *(Error)*: il manipolatore si trova in stato di errore. Provare a bloccare e sbloccare i freni di giunto oppure, alla peggio, riavviar e il sistema agendo sull'interruttore del Controller.

3.2 Spegnimento

Per spegnere il robot è necessario bloccare i freni di giunto del robot e successivamente eseguire lo *Shut Down*. Sia il bottone per il lock dei freni di giunto che il bottone di *Shut Down* si trovano nel menù della pagina App Desk descritta nella sezione 4. In particolare, il pulsante di blocco freni, raffigurato da un lucchetto, si trova nel menù sulla destra della homepage di App Desk (figura 4) mentre il pulsante di *Shut Down* è situato nel menù a tendina che compare cliccando sulle due linee orizzontali in alto a destra. Durante la fase di spegnimento le luci di stato alla base del manipolatore torneranno ad essere gialle lampeggianti per poi spegnersi, ma soltanto quando a video compare l'apposito messaggio la fase di spegnimento potrà dirsi completata e si potrà procedere con l'abbassamento dell'interruttore del Controller.

4 Descrizione App Desk

La casa produttrice del robot Panda, ovvero Franka Emika, ha realizzato una comoda interfaccia web chiamata App Desk (figura 4) dalla quale è possibile amministrare e programmare il robot. Supponendo che il Controller del manipolatore sia collegato alla workstation attraverso cavo Ethernet, per accedere a tale interfaccia web è sufficiente avviare la workstation scegliendo dalla GNU GRUB il kernel Linux **realtime**, aprire un qualsiasi browser e inserire nella barra di ricerca l'indirizzo IP statico del manipolatore[6]. Eventualmente fossero richieste, le credenziali dell'account per entrare nell'applicazione App Desk, a Novembre 2021, sono: panda - pandarobot. É importante sottolineare che, qualora il cavo Ethernet non sia collegato direttamente al Controller del robot bensì alla base del braccio, l'indirizzo IP a cui è possibile accedere all'interfaccia web non sarà più quello statico assegnato a piacere dall'amministratore del robot ma quello assegnato dinamicamente dal server DHCP presente alla base del robot stesso. In ogni caso è possibile accedere all'interfaccia web inserendo nella barra degli indirizzi del browser l'URL **robot.franka.de**.

Come anticipato, in questa schermata di interfaccia è possibile amministrare e programmare il robot. Nel dettaglio, la programmazione del robot attraverso App Desk avviene per mezzo di un linguaggio di programmazione basato sull'interconnessione di blocchi elementari estremamente intuitivo che non è stato approfondito; mentre l'amministrazione avviene per mezzo del menù principale. Dalla schermata principale di figura 4 è possibile:

- Avviare un task programmato con il linguaggio a blocchi attraverso il tasto Start;
- Verificare lo stato dei dispositivi di attivazione di cui si parlerà in seguito;
- Bloccare/Sbloccare i freni di giunto. il pulsante di blocco freni, raffigurato da un lucchetto, si trova nel menù sulla destra della homepage di App Desk;
- Osservare lo stato corrente del robot (equivalente al colore delle luci di stato);
- Scegliere la modalità di pilotaggio (*Pilot Mode*) e la modalità di guida del manipolatore (*Guiding Mode*) che saranno trattate nella sezione 6.

Attraverso le due linee orizzontali presenti sulla parte in alto a destra della schermata di figura 4 si accede alle impostazioni dell'App Desk ed è possibile navigare nelle diverse sezioni di settaggio di cui dispone quest'interfaccia web:

1. Sezione Dashboard;
2. Sezione Network;

[6] È necessario proseguire con la navigazione nonostante il sito web non sia protetto da certificazione SSL.

3. Sezione End-Effector;

4. Sezione Franka World App Desk.

In particolare, nella sezione DASHBOARD, rappresentata in figura 5, è possibile visualizzare una panoramica dello stato di collegamento del robot, la versione del sistema operativo installato nel Controller, la sigla del Firmware e gli indirizzi IP, con relative maschere di sotto-rete, sia del manipolatore che del Controller.
Nella sezione NETWORK, invece, è possibile scegliere se attribuire un indirizzo

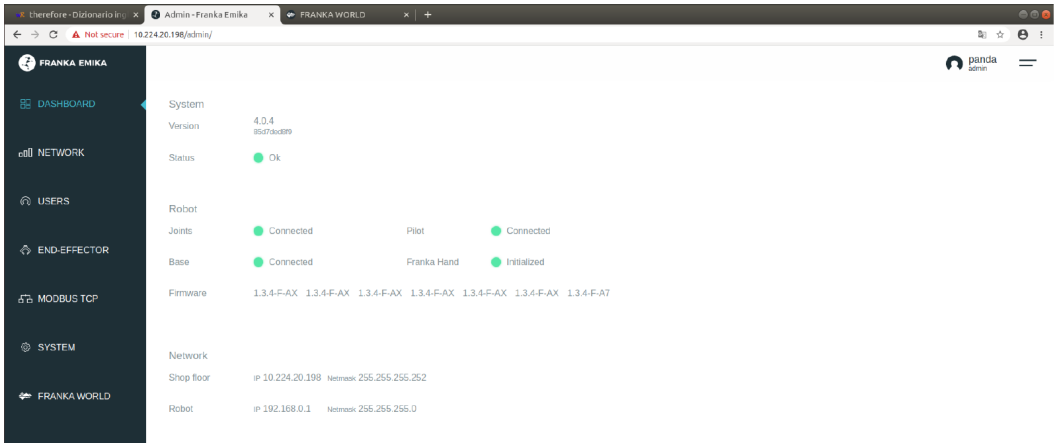

Figure 5: Sezione Dashboard App Desk

IP statico al robot oppure procedere con l'assegnazione automatica attraverso il server DHCP presente alla base del braccio. Nel primo caso, selezionato in figura 6, è possibile anche scegliere un'opportuna maschera di sotto-rete,l'indirizzo del gateway e quella del DNS. La sezione END-EFFECTOR (EE), visibile in figura 7 a pagina 12, può essere utilizzata per effettuare l'inizializzazione o modificare i dettagli meccanici del gripper montato. Ad esempio si può specificare la massa dell'EE, il centro di massa dell'EE, il tensore di inerzia dell'EE e la matrice di trasformazione omogenea tra la flangia e l'EE.
Di particolare interesse risulta essere anche la sezione FRANKAWORLD, rappresentata in figura 8 a pagina 12, realizzata in seguito al rilascio della versione 4:0:0 del sistema operativo da parte della Franka Emika.
Sebbene la piattaforma Franka World sia ancora in fase di sviluppo e aggiornamento, attraverso questa sezione è possibile controllare lo stato di connessione del manipolatore alla piattaforma online dalla quale è possibile scaricare automaticamente eventuali aggiornamenti, se disponibili, agendo sulla funzione SYNCHRONIZE. La pagina di Franka World è accessibile solo in seguito a registrazione come specificato al seguente link. Il robot Panda presente nel laboratorio è già stato registrato a nome del professore *Ciro Natale*. L'App Desk presenta anche altre sezioni di minore importanza quali per esempio: "USERS" per abilitare un utilizzo multi-utente del robot, "SYSTEM" che contiene in-

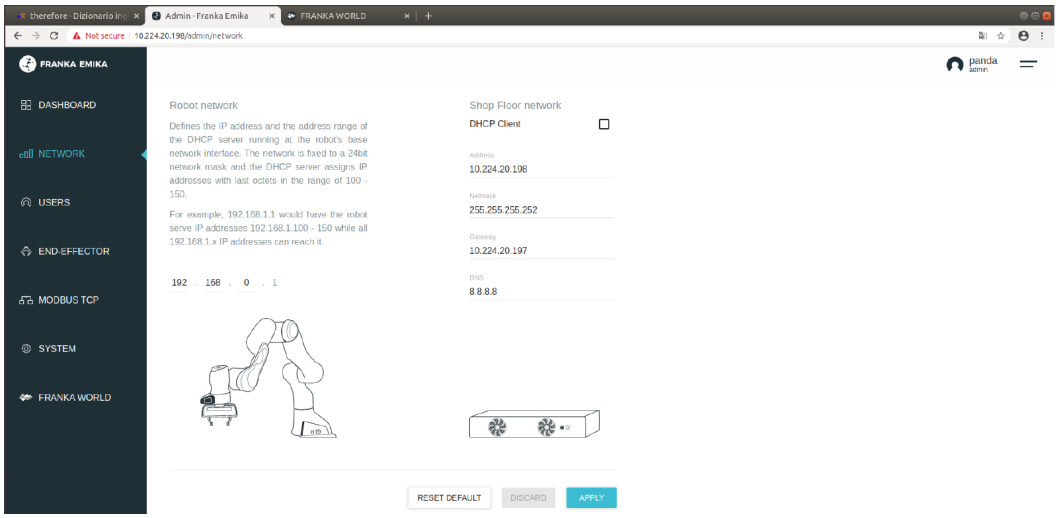

Figure 6: Sezione Network App Desk

formazioni di dettaglio sul sistema e "MODBUSTCP" che conserva i settaggi relativi al protocollo di comunicazione utilizzato dal robot.

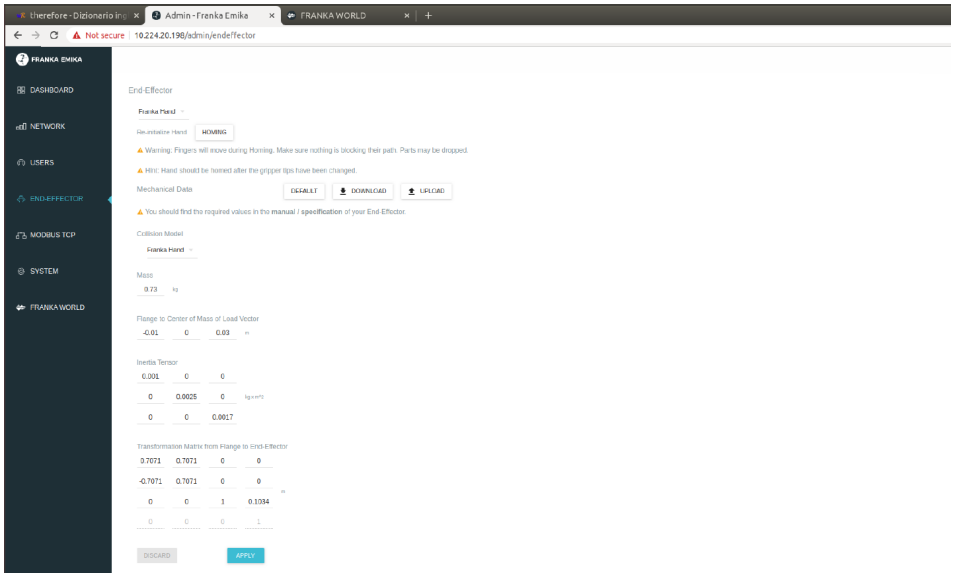

Figure 7: Sezione End-Effector App Desk

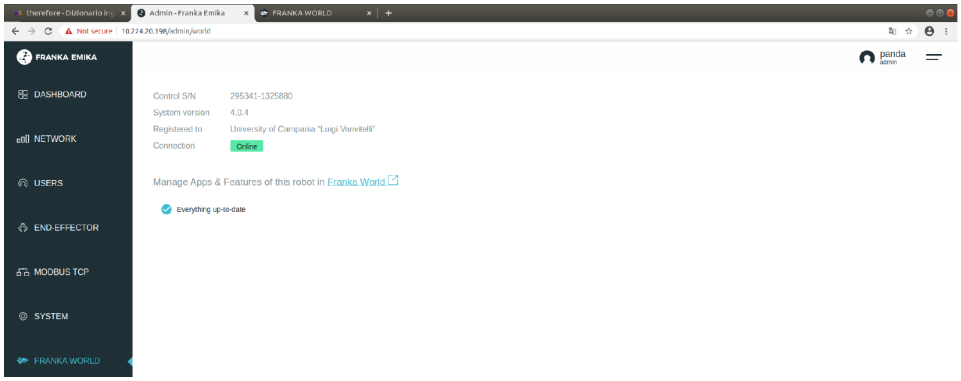

Figure 8: Sezione Franka World App Desk

5 Dispositivi di Attivazione ed Emergency Stop Device

Il robot Panda è dotato di due dispositivi di attivazione e di un dispositivo di emergenza. Tali dispositivi sono:

- **External enabling device**: noto anche come dispositivo dell'uomo morto, consiste di un unico pulsante che va tenuto premuto a metà per attivare il dispositivo e va rilasciato o premuto con forza per abilitare il dispositivo.
- **External activation device**: dispositivo a fungo nero che richiede di svitare il fungo per essere attivato e schiacciare il fungo per essere disabilitato;
- **Emergency stop device**: dispositivo a fungo rosso che interrompe completamente l'alimentazione al manipolatore se viene schiacchiato il fungo rosso. Per riattaccare l'alimentazione è necessario svitare il fungo (il momento torcente da applicare per svitare il fungo dell'Emergency stop device è di gran lunga maggiore di quello richiesto per svitare il fungo dell'External activation device).

Figure 9: Dispositivi di attivazione del robot Panda

Si consiglia di lavorare con un solo dispositivo di attivazione: anche se è già stato anticipato in sezione 3, è utile ricordare che il robot entra in stato di conflitto (indicato dalle luci di stato rosa) se si prova ad utilizzarli contemporaneamente. Ipotizzando di aver messo da parte l'External enabling decive (uomo morto), che quindi sarà di default disabilitato, e di lavorare unicamente con l'*External activation device* (fungo nero):

- Schiacciando il fungo nero, quindi disabilitandolo, il robot entra nello stato *Interazione* (luci di stato bianche fisse) in cui è possibile utilizzare il robot in *Pilot Mode* e *Guiding Mode*. Queste due modalità di controllo sono spiegate nella sezione 6.
- Svitando il fungo nero, quindi attivandolo, il robot entra nello stato *Attenzione* (luci di stato blu fisse) in cui è possibile eseguire un task. Come scrivere il codice per realizzare un task è spiegato nella sezione 7.

Se l'*External activation device* viene schiacciato, quindi disabilitato, durante l'esecuzione di un task il robot arresta il proprio movimento e genera un'eccezione software. Tale comportamento suggerisce di utilizzare l'*External activation device* (fungo nero) in luogo dell'*Emergency stop device* (fungo rosso) se si vuole unicamente fermare il robot e non vi sono emergenze che richiedono di interrompere l'alimentazione.

6 Guiding Mode e Pilot Mode

Il robot Panda può essere fin da subito utilizzato in *Guiding Mode* e in *Pilot Mode*. Prima di poter attivare una delle due modalità è necessario che il robot sia nello stato *Interazione* indicato da luci di stato bianche e fisse. Per portarsi nello stato interazione:

1. Rilasciare i freni di giunto (vedere come fare nella sezione 4)
2. Disabilitare i due dispositivi di attivazione (vedere come fare nella sezione sezione 5.[7]

Nei seguenti sottoparagrafi verranno illustrate le modalità *Guiding Mode* e *Pilot Mode*.

6.1 Guiding Mode

Il controllo *Guiding Mode* è equivalente ad un controllo di forza diretto in cui forze e coppie desiderate in punta all'organo terminale sono scelte nulle. In altre parole, sotto l'azione del controllo *Guiding Mode* l'operatore può manualmente applicare forze e momenti per spostare l'EE del manipolatore all'interno dello spazio di lavoro. Tale modalità di controllo è sicuramente utile per memorizzare pose dell'EE o configurazioni adatte all'esecuzione di un task.
Dando per scontato che il robot si trovi nello stato *Interazione*, il manipolatore può essere guidato manualmente dall'utente premendo il *Guiding Button* e contemporaneamente premendo **a metà** l'*Enabling Button* (figura 11). Qualora l'*Enabling Button* venga premuto completamente il braccio si fermerà immediatamente alla stessa stregua di ciò che accadrebbe se questo non venisse premuto affatto (panic function del tasto di abilitazione).

Per permettere all'utente di personalizzare il controllo in *Guiding Mode*, la Franka Emika ha realizzato la sottosezione *Guiding Mode* nel menù principale di App Desk (mostrata in figura 10). Tale sottosezione permette di selezionare uno dei seguenti controlli di tipo *Guiding Mode*:

- **Translation**: il manipolatore può essere spostato esclusivamente per cambiare la posizione Cartesiana dell'organo terminale. Eventuali momenti applicati all'EE vengono compensati in modo da mantenere l'orientamento costante.
- **Rotation**: il manipolatore può essere spostato esclusivamente per cambiare l'orientamento dell'organo terminale. Eventuali forze applicate all'EE vengono compensate in modo da mantenere la posizione costante.
- **User**: in questa modalità di guida l'utente può definire il comportamento di guida, il ché significa che è possibile definire per ogni asse Cartesiano se il manipolatore deve essere mobile o immobile (compensare o meno forze e momenti) sia in posizione che in orientamento.

[7] Se viene utilizzato unicamente l'External activation device come suggerito allora l'External enabling device è già disabilitato.

- **Free**: il braccio può essere mosso liberamente, quindi tutti e 7 i giunti possono essere movimentati;

Chiaramente se vengono applicati forze e/o momenti eccessive lungo direzioni in cui il robot non ammette spostamenti e/o rotazioni allora il robot Panda potrebbe danneggiarsi.

6.2 Pilot Mode

Insieme alla *Guiding Mode*, sempre nel menù principale dell'interfaccia web App Desk in figura 10, è presente la sottosezione *Pilot Mode*. Attivare la *Pilot Mode* attraverso App Desk è equivalente a cliccare il bottone situato sul robot *Pilot Mode* mostrato in figura 11. La Pilot Mode consente di comandare l'apertura e la chiusura delle dita del gripper. In particolare, facendo riferimento ai bottoni situati sull'ultimo link del Panda, è possibile:

- Tenere premuto il tasto "sinistra" per aprire le dita lentamente;
- Tenere premuto il tasto "destra" per chiudere le dita lentamente;
- Premere una volta il tasto "giù" per aprire le dita alla massima larghezza;
- Premere una volta il tasto "su" per chiudere le dita del gripper. Un eventuale oggetto afferrato viene trattenuto con la forza di tenuta.

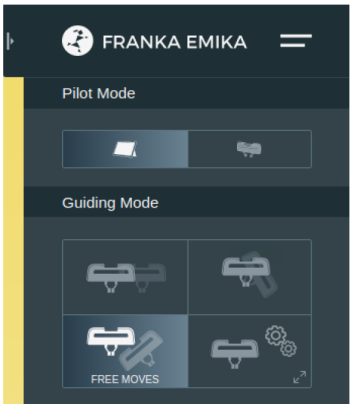

Figure 10: *Pilot and Guiding mode*

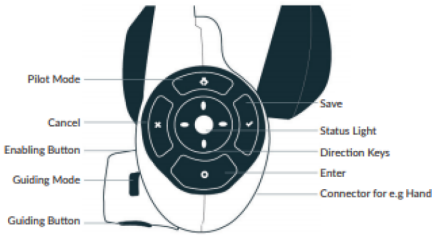

Figure 11: Tasti pilotaggio locali

7 Programmazione del robot ed esecuzione dei programmi

7.1 Programmazione del robot

Il robot Panda viene programmato in C++ attraverso la libreria open-source ***libfranka***[8]. Questa libreria è stata sviluppata dalla casa produttrice del robot Panda, ovvero Franka Emika, e mette a disposizione:

- Metodi per accedere al modello cinematico del manipolatore;
- Metodi per accedere ai parametri cinematici e dinamici del manipolatore;
- Metodi di comunicazione veloce, diretta e bidirezionale di basso livello tra il manipolatore e la workstation. Permette infatti di inviare al robot segnali di controllo e ricevere misure dai sensori in real-time alla frequenza di 1 kHz;

Utilizzare la libreria libfranka non è complesso e la documentazione ufficiale è ben scritta. Come però è noto dai corsi di programmazione, il modo più semplice e veloce per imparare ad utilizzare una nuova libreria è partire degli esempi. Non a caso, il documento che si sta leggendo in questo momento è previsto che sia allegato alla repository ***libfranka_ demo***[9] in cui sono stati aggiunti diversi esempi commentati e documentati. Per non rubare il merito agli sviluppatori della Franka Emika, l'autore ci tiene a precisare che la repository *libfranka* nasce da una clonazione della repository Github ufficiale *libfranka*[10] della Franka Emika. Facendo riferimento alla repository ufficiale di Novembre 2021, *libfranka_ demo* si differenzia da *libfranka* per i seguenti aspetti:

1. È presente il documento che si sta leggendo.
2. È stata aggiunta la directory CustomPrograms pensata per contenere i file C++ realizzati dagli studenti che programmeranno il robot Panda e per separare tali programmi dagli esempi di libfranka contenuti nella directory *examples*.
3. È stato aggiunto e configurato il CMakeLists.txt della directory CustomPrograms per compilare i programmi C++ realizzati dallo studente. Ogni programma C++ che si desidera compilare va dichiarato nel CMakeLists.txt.
4. È stato modificato il root CMakeLists.txt (CMakeLists.txt contenuto nella directory *libfranka_ demo*) in modo da includere CustomPrograms tra le subdirectory da buildare.

[8] La documentazione delle API della libreria è consultabile al seguente URL.
[9] Realizzata durante il lavoro di tirocinio dell'autore.
[10] Reperibile al seguente URL.

5. È stata aggiunta la subdirectory CustomLibrary pensata per contenere eventuali librerie C++ scritte dagli studenti. All'interno di tale directory è stato inserito un banale esempio di libreria utile a capire come configurare il CMakeLists.txt (libreria *StateSaver*).

6. È stato aggiunto e configurato il file CMakeLists.txt della directory CustomLibrary per compilare le librerie C++. Ogni volta che si desidera creare una libreria questa va dichiarata in tale file CMakeLists.txt.

La repository *libfranka_ demo*, in quanto repository clonata, presenta tutti i codici d'esempio della directory *examples* della repository ufficiale *libfranka*. Nonostante tale directory sia un'ottima fonte di esempi, nella directory CustomPrograms sono stati aggiunti i seguenti codici d'esempio:

1. TorqueControlGeneric.cpp: programma che rappresenta lo scheletro da impiegare per implementare un generico algoritmo di controllo in coppia del robot Panda. Il file è pensato come punto di partenza per sviluppare il proprio algoritmo di controllo. Il codice è stato commentato e documentato adeguatamente.

2. Cartesian_Impedance_Control.cpp: programma di esempio che mostra come implementare un controllo di cedevolezza attiva in spazio operativo con rappresentazione dell'orientamento mediante quaternione unitario. L'esempio è stato realizzato a partire dall'omonimo programma presente nella cartella examples della repository libfranka ufficiale. A tale programma sono state effettuate le seguenti modifiche:

   (a) È stato aggiunto un timer per terminare l'esecuzione del programma dopo un certo intervallo di tempo configurabile a piacere. Il timer non è un elemento indispensabile ai fini del paritcolare algoritmo di controllo ma è imporante che si conosca come utilizzarlo.

   (b) È stata utilizzata la classe StateSaver presente nella libreria custom realizzata come esempio. Anche in questo caso, la classe StateSaver non è indispensabile all'algoritmo di controllo ma, come suggerisce il nome della classe, serve unicamente a memorizzare lo stato del robot.

   (c) Il codice è stato opportunamente commentato.

3. UtilizzoStateSaver.cpp: programma C++ che mostra come utilizzare la classe StateSaver realizzata come libreria d'esempio.

4. Cartesian_Force_Impedance_Control.cpp: il programma è frutto della tesi del collega Luigi Vetrella, motivo per cui nella directory root è stato incluso anche il pdf della sua tesi. Tra i programmi di esempio questo è sicuramente il più complesso e completo.

Per scrivere i propri programmi si consiglia di utilizzare VScode in quanto le estensioni (ex CMake,C++ IntelliSense) dedicate al linguaggio C++ semplificano il lavoro.

7.2 Compilazione

Una volta clonata la repository *libfranka_ demo* da Github[11] è necessario compilare i file di esempio per poterli eseguire. Per farlo è sufficiente:

1. Installare le dipendenze:
   `sudo apt install build-essential cmake libpoco-dev libeigen3-dev`
2. Nella directory libfranka_demo creare la directory build:
   ```
   cd /path/to/libfranka_demo
   mkdir build
   cd build
   ```
3. Creare i makefile:
   `cmake ..`
4. Compilare:
   `make`

Da questo momento in poi è possibile eseguire i programmi come sarà spiegato nel sottoparagrafo 7.3. Può sorgere la domanda: come compilare un nuovo programma? Molto semplicemente:

1. Scrivere[12] e salvare il programma C++ nella directory CustomPrograms. Nel caso in cui si ritenga necessario aggiungere una propria libreria scrivere questa nella subdirectory CustomLibrary.
2. Aggiungere il file da compilare al set CUSTOMPROGRAMS nel CMakeLists.txt della directory CustomPrograms.
   Se nella directory CustomLibrary è stata scritta una libreria che si vuole utilizzare allora serve:
   (a) Modificare il CMakeLists.txt della directory CustomLibrary in modo che la libreria sia compilata.
   (b) Specificare nel CMakeLists.txt della directory CustomPrograms a che programma linkare la libreria;

   Attenzione: i documenti CMakeLists.txt sono stati commentati dove necessario ma è comunque consigliata la visione della documentazione ufficiale di CMake.
3. Una volta modificati i file CMakeLists.txt è sufficiente eseguire:
   ```
   cd /path/to/libfranka_demo/build/CustomPrograms
   make
   ```

Così facendo i programmi eseguibili saranno caricati automaticamente nella directory /path/to/libfranka_demo/build/CustomPrograms.

[11] git clone `https://github.com/Vanvitelli-Robotics/libfranka_demo.git`

[12] Prima di scrivere il proprio algoritmo C++ sfruttando la libreria libfranka è suggerito dare uno sguardo agli esempi inseriti nella directory CustomPrograms e agli esempi della directory examples

7.3 Esecuzione

Affinché sia possibile lanciare un qualsiasi algoritmo di controllo il robot possa eseguire il task codificato nel codice C++ questo deve trovarsi nello stato *Attenzione* indicato da luci di stato blu e fisse. Inoltre, diversamente dall'esecuzione di un task codificato nel linguaggio a blocchi avviabile dall'interfaccia web, in questo caso è strettamente necessario che la workstation sia collegata al Controller e non alla base del braccio (il manipolatore deve avere un indirizzo IP di tipo statico). Per portarsi nello stato attenzione:

1. Rilasciare i freni di giunto (vedere come fare nella sezione 4);
2. Attivare uno dei dispositivi di attivazione (vedere come nella sezione 5);

Fermo restando le premesse fatte sul set-up del manipolatore ad inizio documento, è inoltre necessario riavviare la workstation ed entrare nel sistema scegliendo la patch real-time del kernel Linux. Supponendo di voler eseguire uno dei codici di esempio presenti nella directory CustomPrograms della repository *libfranka_ demo*, diciamo Cartesian_Impedence_Control.cpp, la prima cosa da fare è spostarsi nella sotto-directory build della cartella libfranka digitando il seguente comando:
`cd /path/to/libfranka_demo/build/CustomPrograms`
E poi lanciare il programma:
`./Cartesian_Impedence_Control <ip-fci>`

A questo punto è sufficiente seguire le indicazioni che vengono stampate a schermo (che consistono tipicamente nel premere invio da tastiera quando si è pronti) per far partire il movimento (in questo caso il manipolatore si porterà in una configurazione predeterminata ed eseguirà un test sulla qualità di connessione con la workstation stampando i risultati a video).