



# *Fondamenti di robotica:*

Programmazione Tiago per  
operazioni di pick and place

## GRUPPO 2





## Partecipanti :

- *Gianluca Barnaba*
- Federico Bini
- Beatrice Bussone
- Federico Principi
- Luca Scorza



## Svolgimento dei task:

- tutti i componenti hanno partecipato attivamente alla gestione del carico di lavoro
- Nessuna particolare suddivisione

```
src
└── task_1
    ├── resource
    └── task_1
        ├── action_client.py
        ├── action_server.py
        ├── aruco_handler.py
        └── grasp_publisher.py
    └── test
└── task_2
    ├── resource
    └── task_2
        ├── state_machine.py
        └── planner_ik.py
    └── test
└── task_3
    ├── resource
    └── task_3
        ├── LbD.py
        ├── state_DMP.py
        └── planner_DMP.py
    └── test
└── personal_interfaces
    ├── action
    │   ├── MoveHead.action
    │   └── PlanAndExecute.action
    └── srv
        └── Attach.srv
└── myBringUp
    ├── action_app.launch.py
    └── launch_task_3.launch.py
```

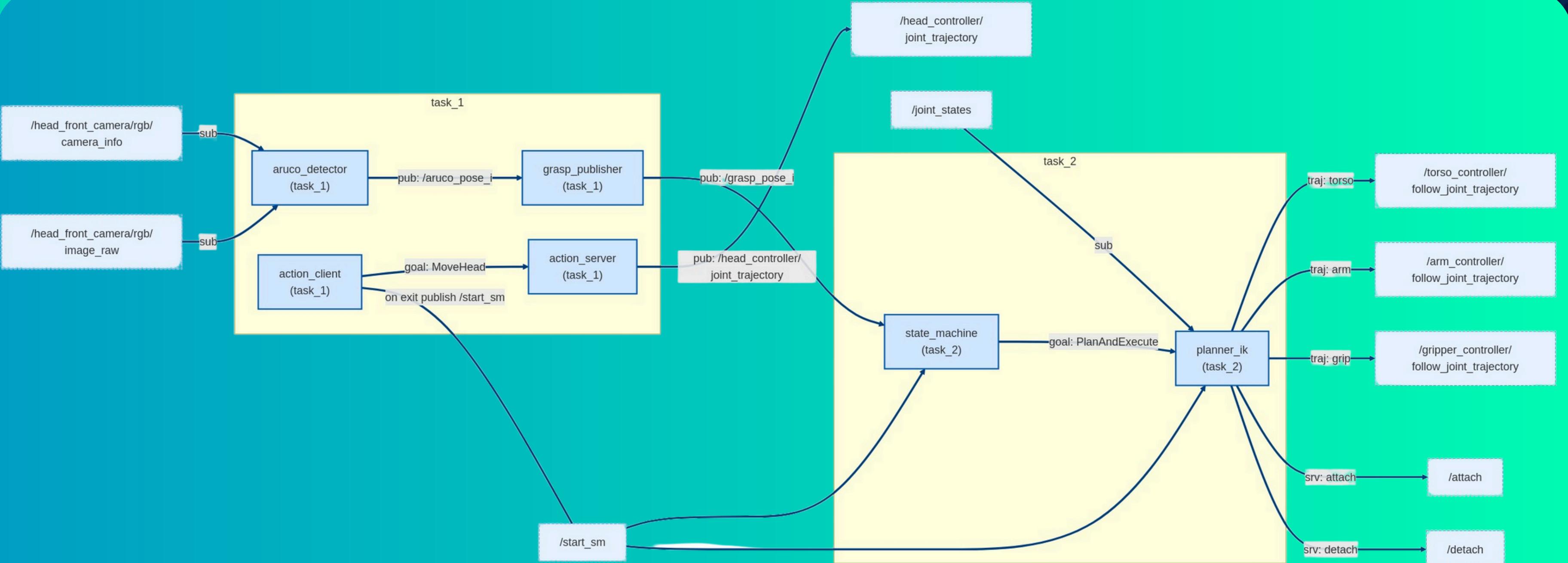
# Gerarchia delle cartelle:

**Blocco responsabile della rilevazione e pubblicazione delle pose di interesse.**

**Sezione incaricata di pianificare le traiettorie, calcolare la cinematica inversa e inviare comandi ai giunti (in due modi diversi in task\_2 e task\_3).**

**Blocco contenete tutti i service e le action utilizzate per permettere la comunicazione tra i vari nodi.**

# Diagramma di funzionamento dell'applicazione:



Per motivi grafici è stato omesso il task\_3 e i suoi nodi, che sono da considerarsi analoghi a quelli del task\_2 ai fini della rappresentazione

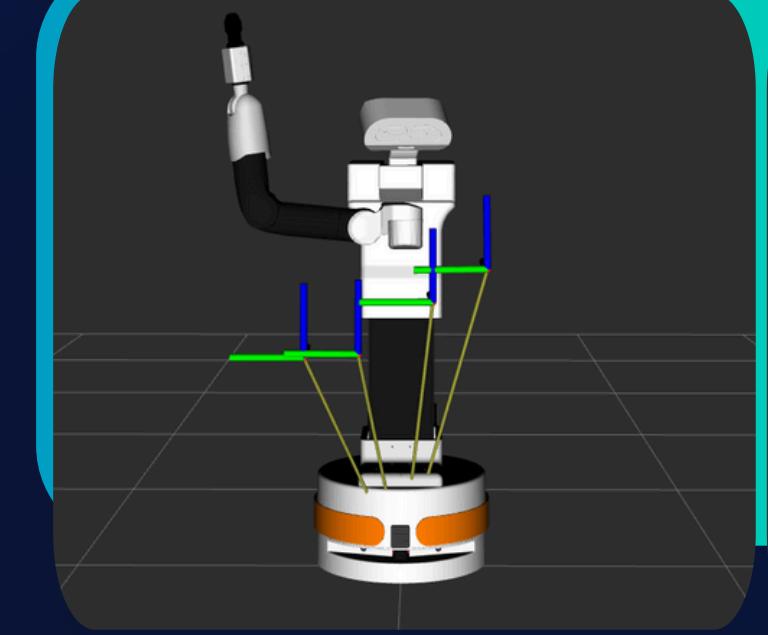


# TASK 1

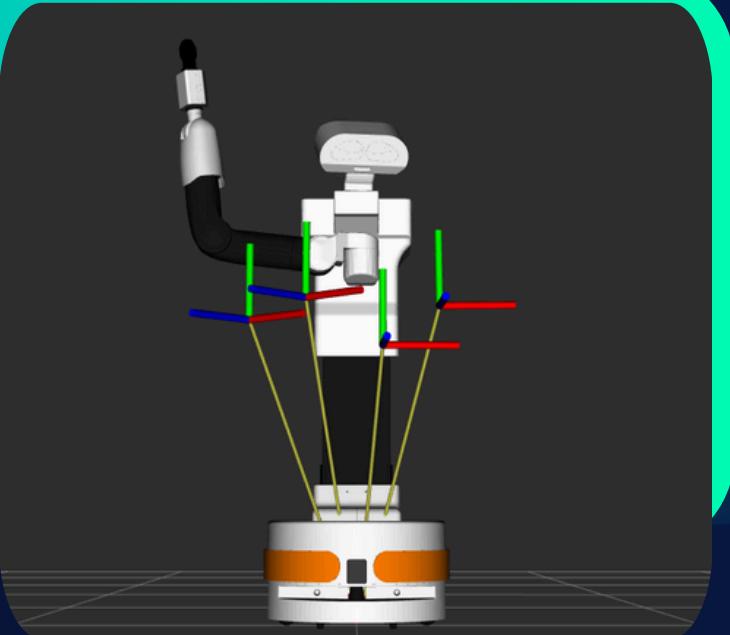
## ***Step principali:***

- Setting iniziale per la scansione dell'ambiente e definizione della posa di partenza.
- Sviluppo dei codici per la rilevazione degli ArUco marker.
- Acquisizione delle coordinate dei marker e successiva trasformazione in terna base.

**Aruco\_pose**



**Grasp\_pose**



## ***Specifiche dei codici:***

Le pose dei marker vengono pubblicate durante la scansione, nei vari ArUco-pose topic. Al termine della scansione le ultime pose vengono salvate nei grasp-pose topic.

In questo modo una volta spenta la camera Tiago è comunque in grado di raggiungere tali pose.



# TASK 1: SVILUPPO

## Action Server - Client:

Per il movimento della testa del robot è stata usata una action interface invece di un semplice topic, così da poter:

- inviare feedback
- interrompere l'esecuzione



### Struttura Action

```
# Goal  
float64 left_limit  
float64 right_limit  
float64 step_angle  
---  
# Result  
bool success  
---  
# Feedback  
float64 current_angle
```

## Lettura pose e pubblicazione:

- Acquisizione della posa dei marker con riferimento alla terna della camera.
- Trasformazione per riportare le pose in terna base.
- Pubblicazione delle nuove pose su topic appositi.

## Correzione offset:

### Nodi *grasp\_pose*

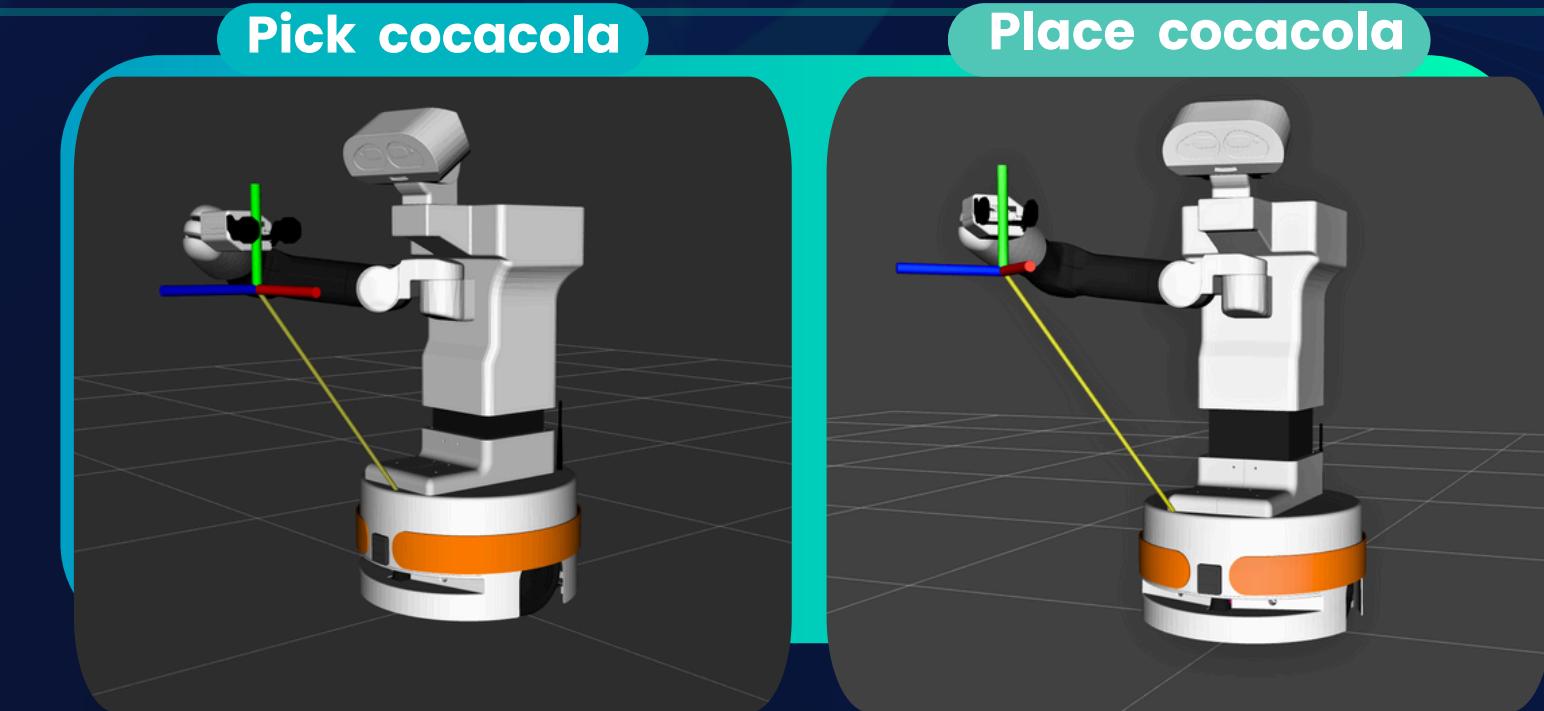
- leggono le pose dai topic *aruco\_pose*
- trasformazione TF
- offset di posizione → esecuzione più precisa di *pick and place*.



# TASK 2

## Step principali:

- Pianificazione delle traiettorie per lo spostamento dell'end-effector da posa iniziale a pose da raggiungere.
- Visualizzazione terna end-effector e terne degli ArUco da raggiungere, in ambiente Rviz.
- Inversione cinematica.



## Nodi implementati:

Il nodo *state\_machine* dopo aver ricevuto l'avvio dall'action client procede a leggere le pose pubblicate sul *grasp\_topic* e a costruire e inviare goal in base al tipo di task al nodo *planner\_node*.



Il nodo *planner* esegue i vari task, tra cui inviare le traiettorie da seguire ai vari giunti del braccio, in modo da raggiungere le pose desiderate



# TASK 2: SVILUPPO

## ***Tipi di task:***

- Move
- Place
- Lift
- Open/Close Gripper
- Back to Home
- Take/Put



### ***Move:***

- Input: geometry\_msgs/Pose
- Output: valore booleano

### ***Place/Lift:***

- Input: delta z (float)
- Output: valore booleano

### ***Back to home***

- input: None
- output: valore booleano

### ***Open/Close gripper:***

- input: string, "open\_gripper" o "close gripper"
- output: valore booleano

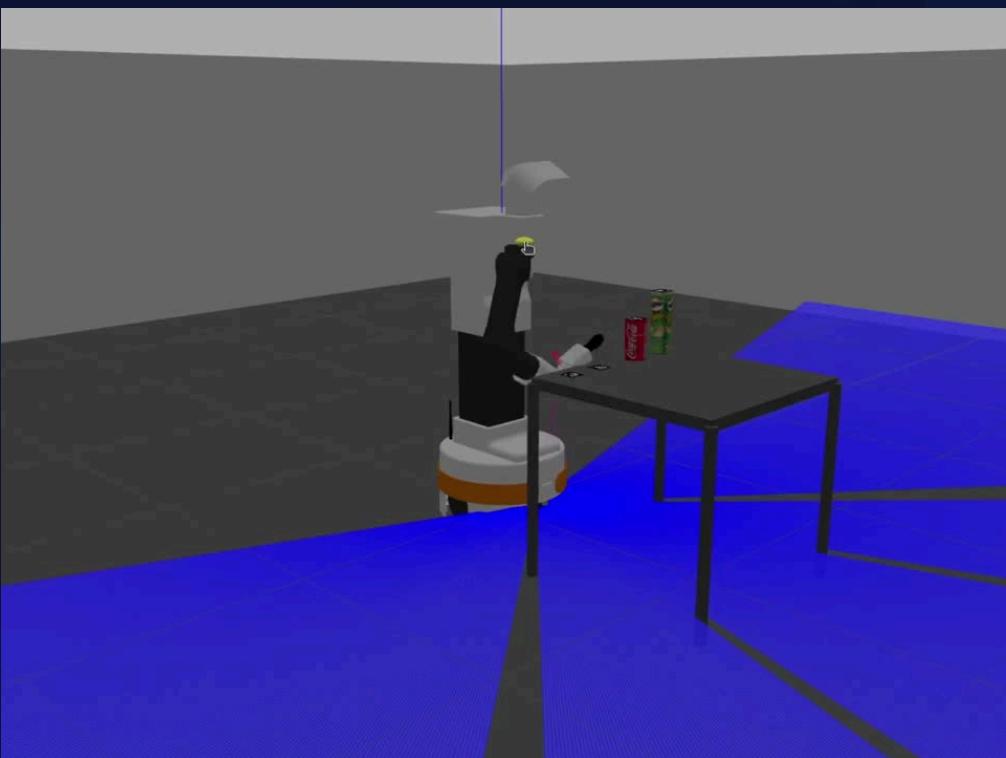
### ***Struttura Action***

```
# Goal  
geometry_msgs/PoseStamped target_pose  
    string task_type  
    string model_name_2  
    string link_name_2  
    int32 marker_id  
    ---  
# Result  
bool success  
    ---  
# Feedback  
string status
```

### ***Take/Put:***

- input: model\_name, link\_name
- output: valore booleano

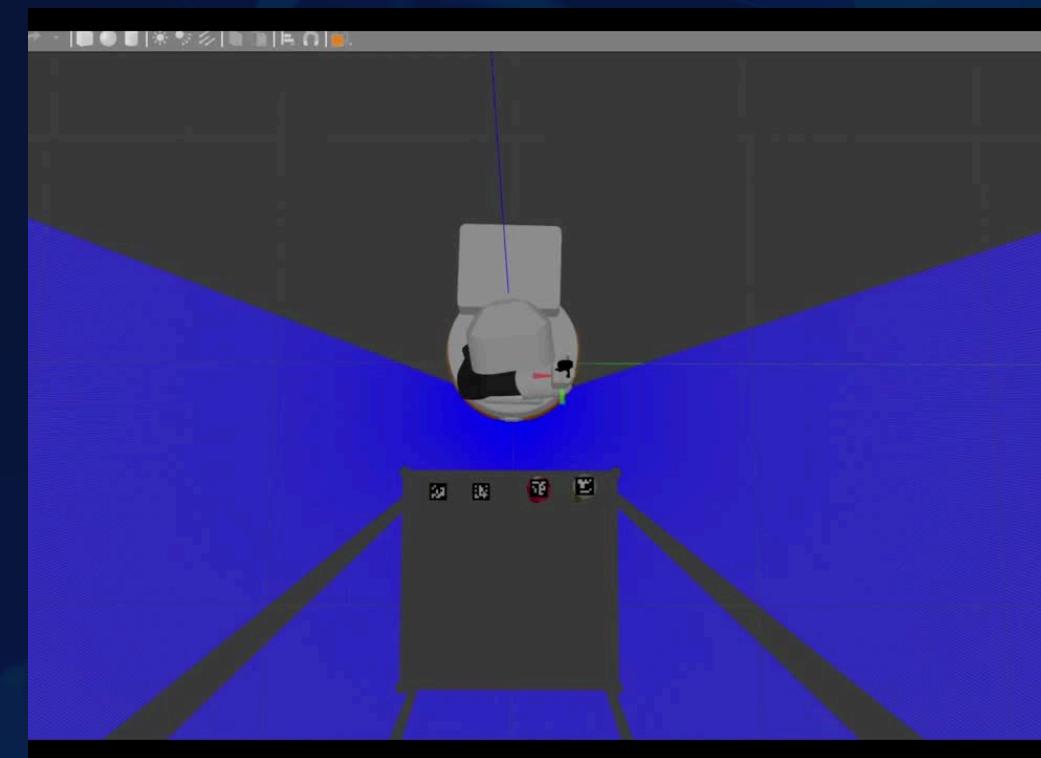
# TASK 2: ESECUZIONE



Vista laterale



Vista frontale

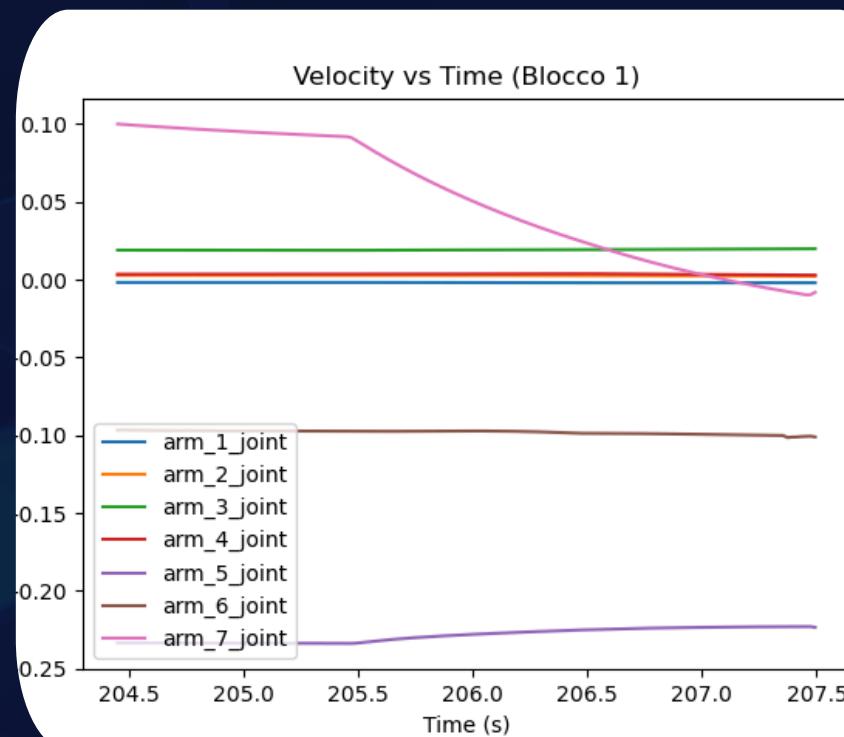
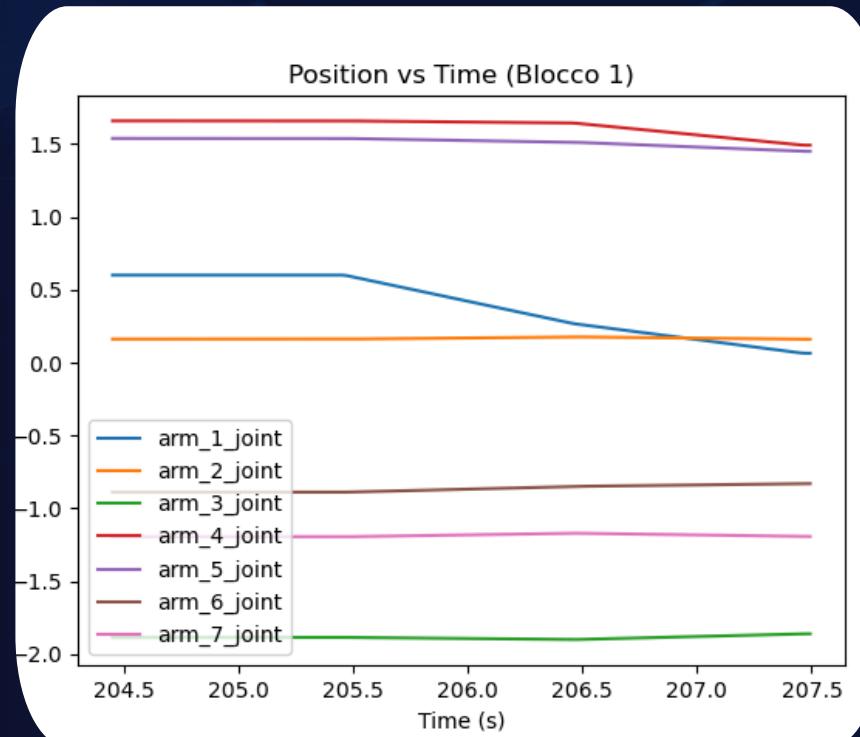
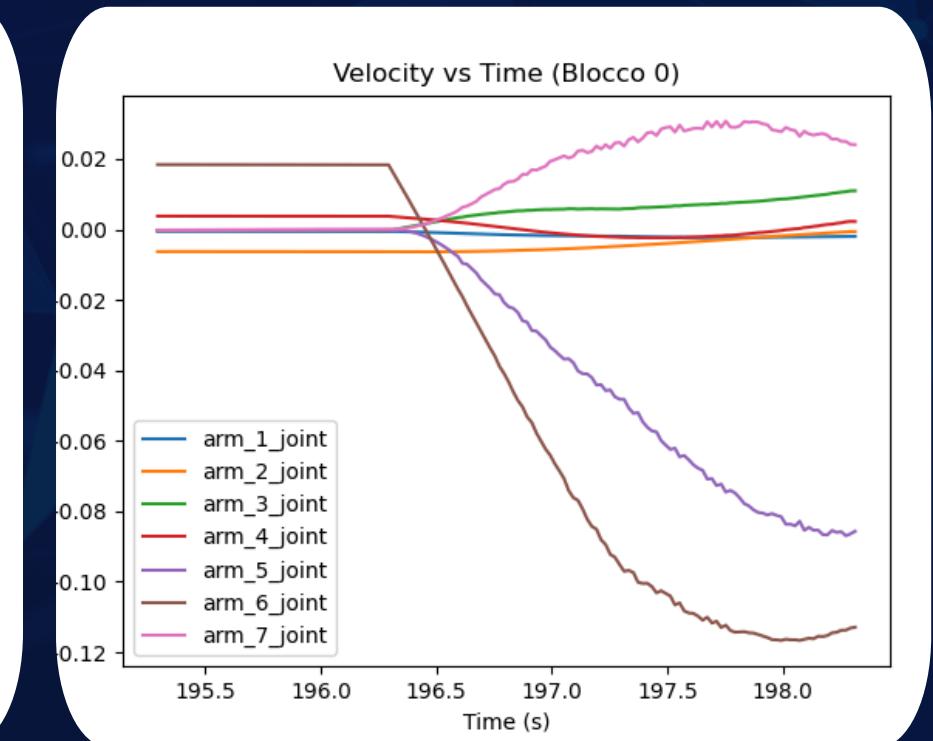
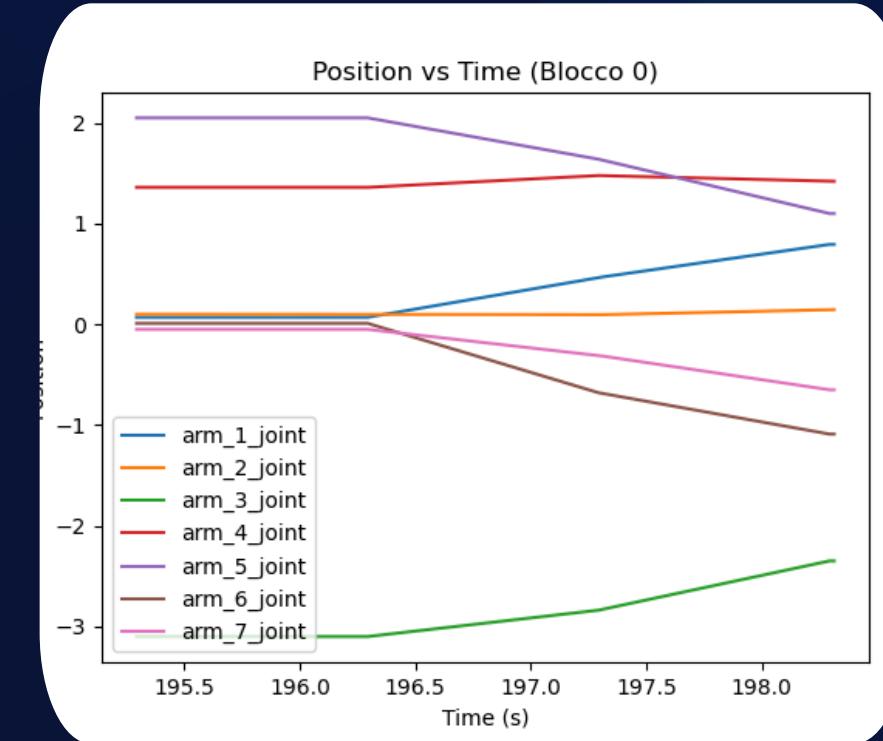


Vista dall'alto



# TASK 2: ANALISI

Analisi degli andamenti di **posizione** e **velocità** nel tempo durante il task move.  
(primo movimento)



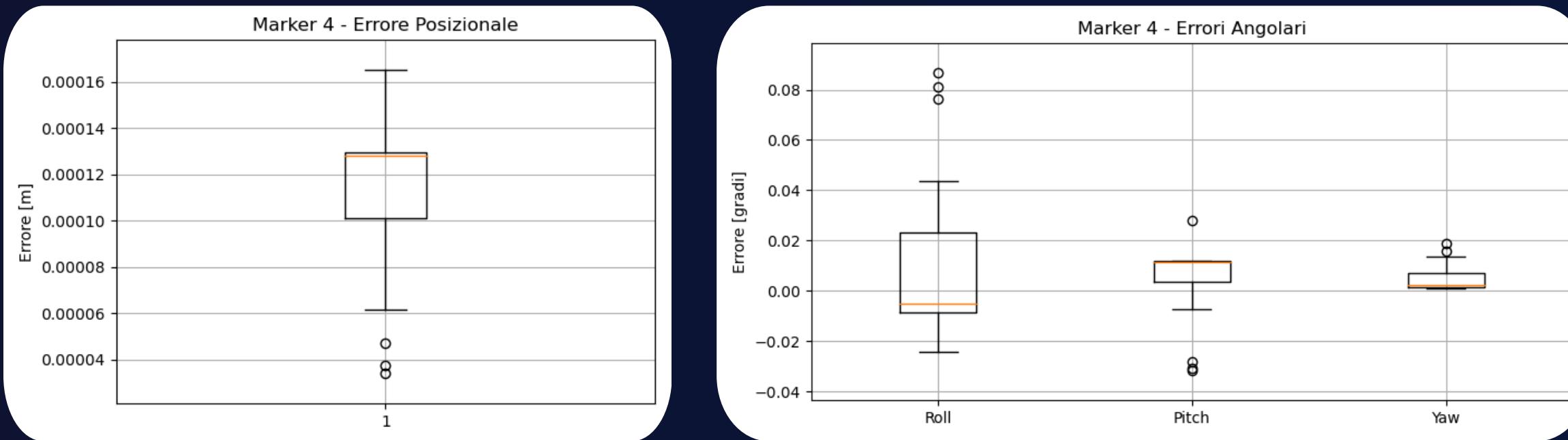
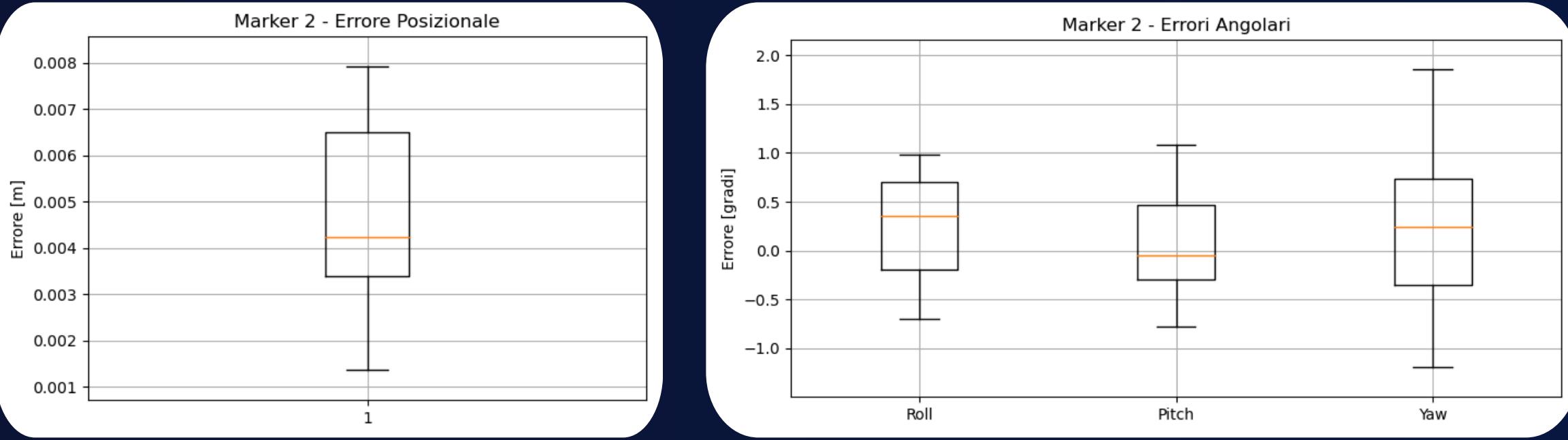
Analisi degli andamenti di **posizione** e **velocità** nel tempo durante il task move.  
(secondo movimento)

realizzate tramite 2 nuovi task type activate e deactivate e tramite il nodo analisi.py



# TASK 2: ANALISI

Analisi dell'**errore** tra  
**posa desiderata** e  
**posa raggiunta**  
dall'end effector  
durante il task move  
(primo movimento)



Analisi dell'**errore** tra  
**posa desiderata** e  
**posa raggiunta**  
dall'end effector  
durante il task move  
(secondo  
movimento)

realizzate tramite il task analyze



# TASK 3

## Step principali:

→ **modulo LbD** per la pianificazione del movimento:

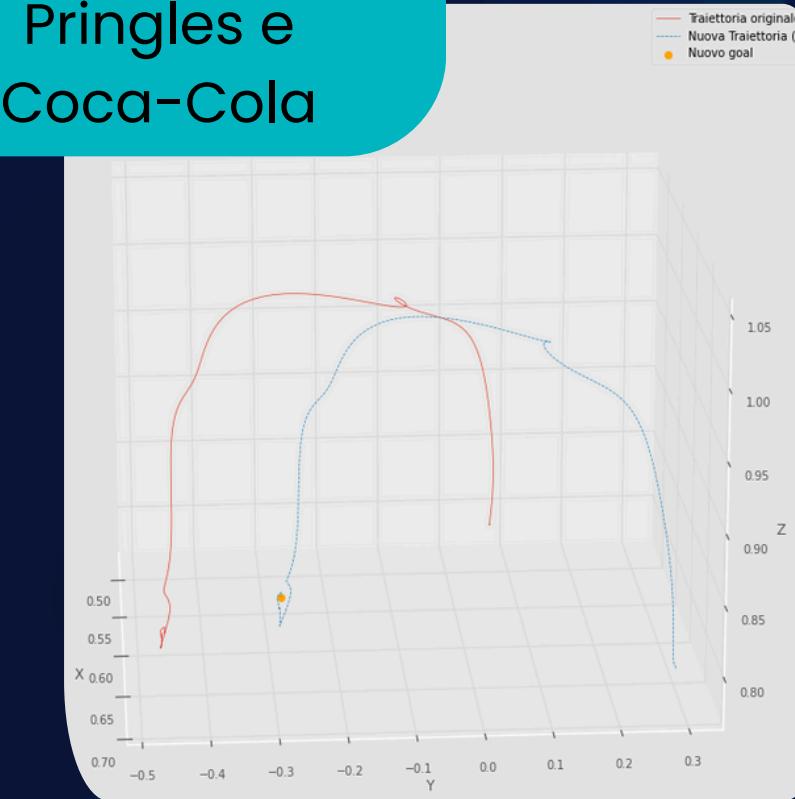
- utilizzo DMP per interpolare (in SO) le acquisizioni prese dai dataset (in SG)
- visualizzazione e scelta delle traiettorie acquisite per il pick e per il place
- adattamento delle traiettorie alle pose target
- inversione cinematica → ingresso ai controlli

## LbD con approccio DMP

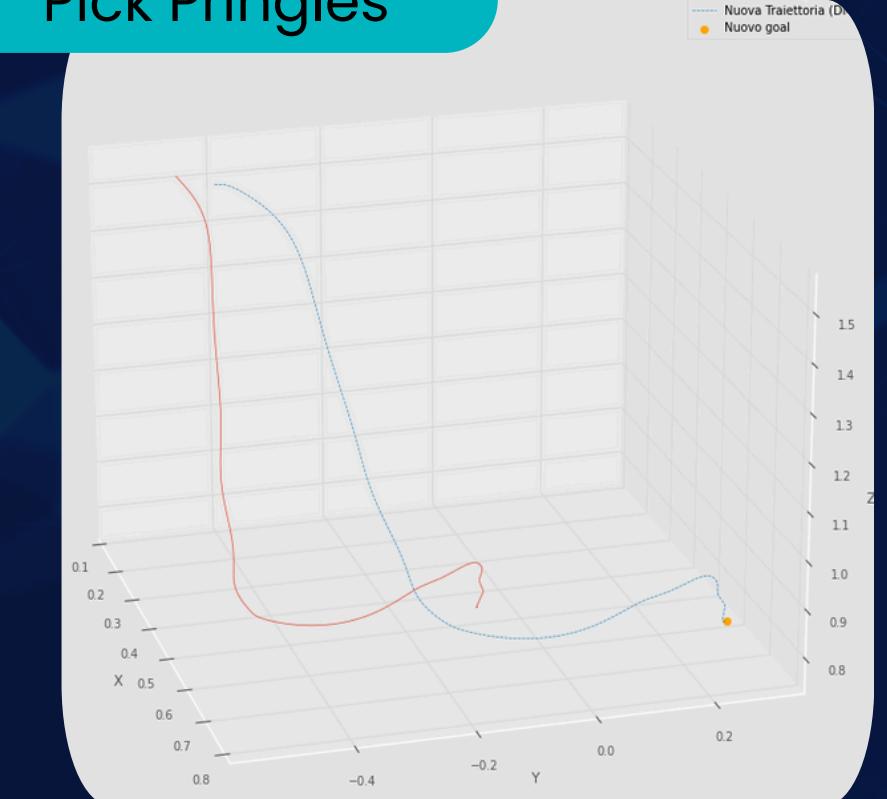
Modello generativo per:

- Apprendere una traiettoria da dimostrazione
- Generalizzare la traiettoria verso nuovi goal
- Movimento fluido

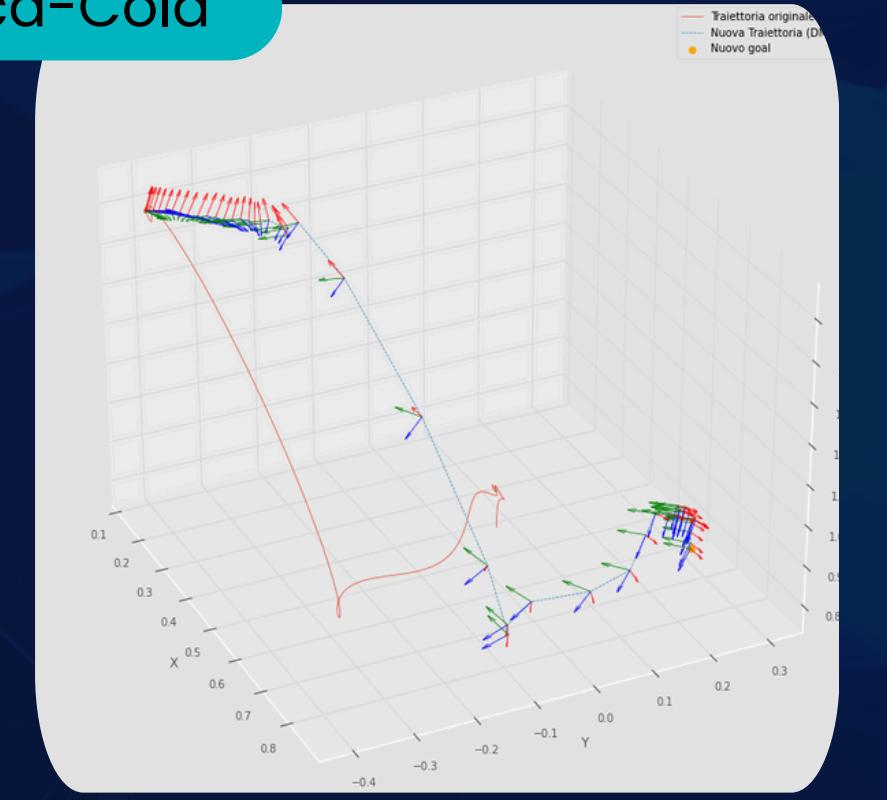
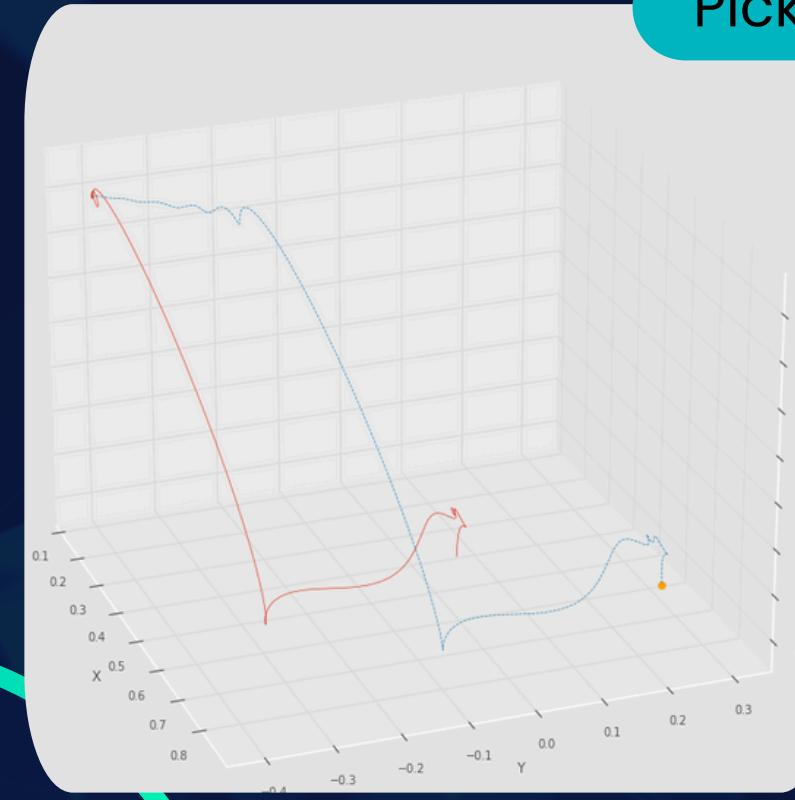
Place di  
Pringles e  
Coca-Cola



Pick Pringles



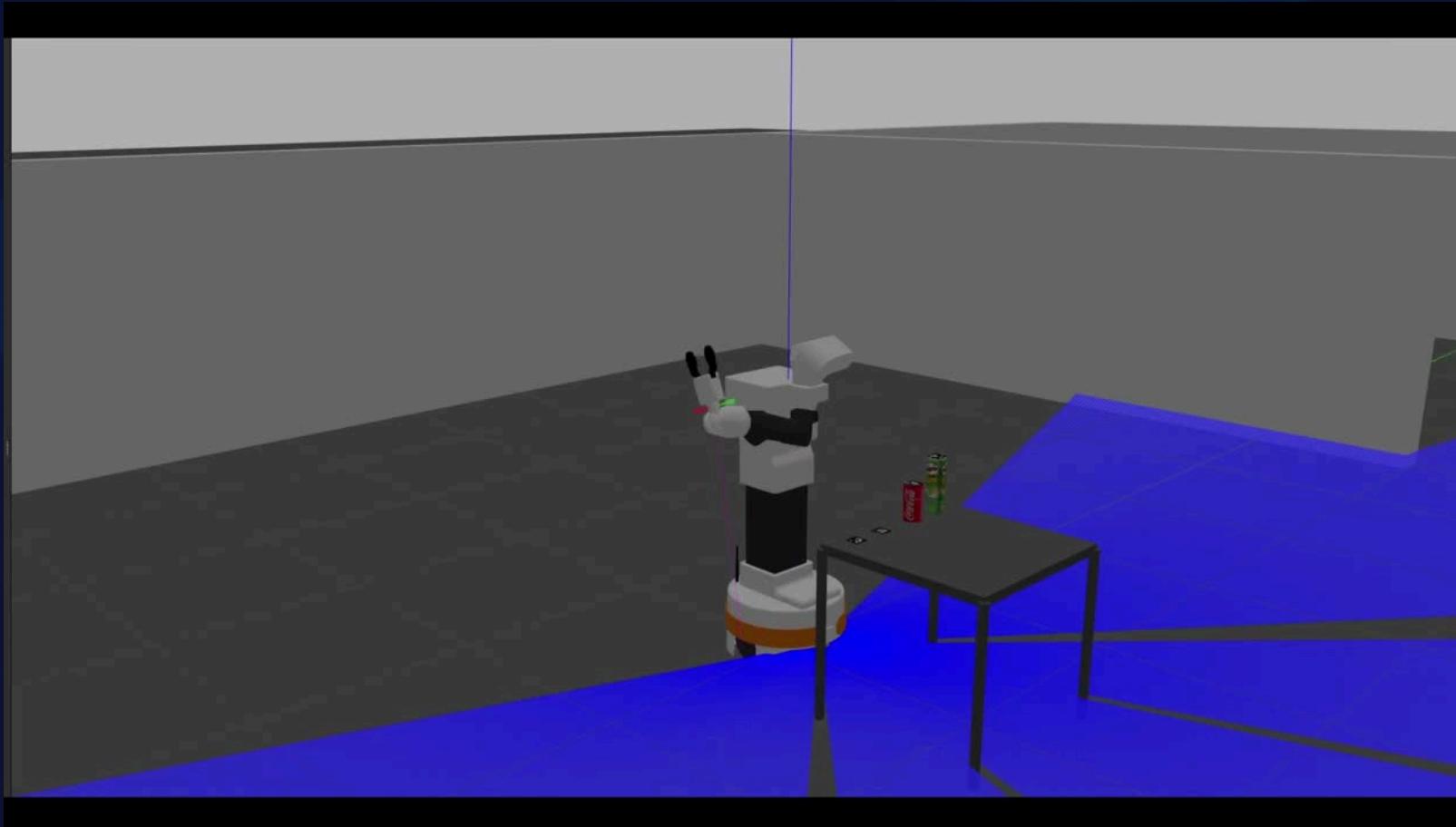
Pick Coca-Cola



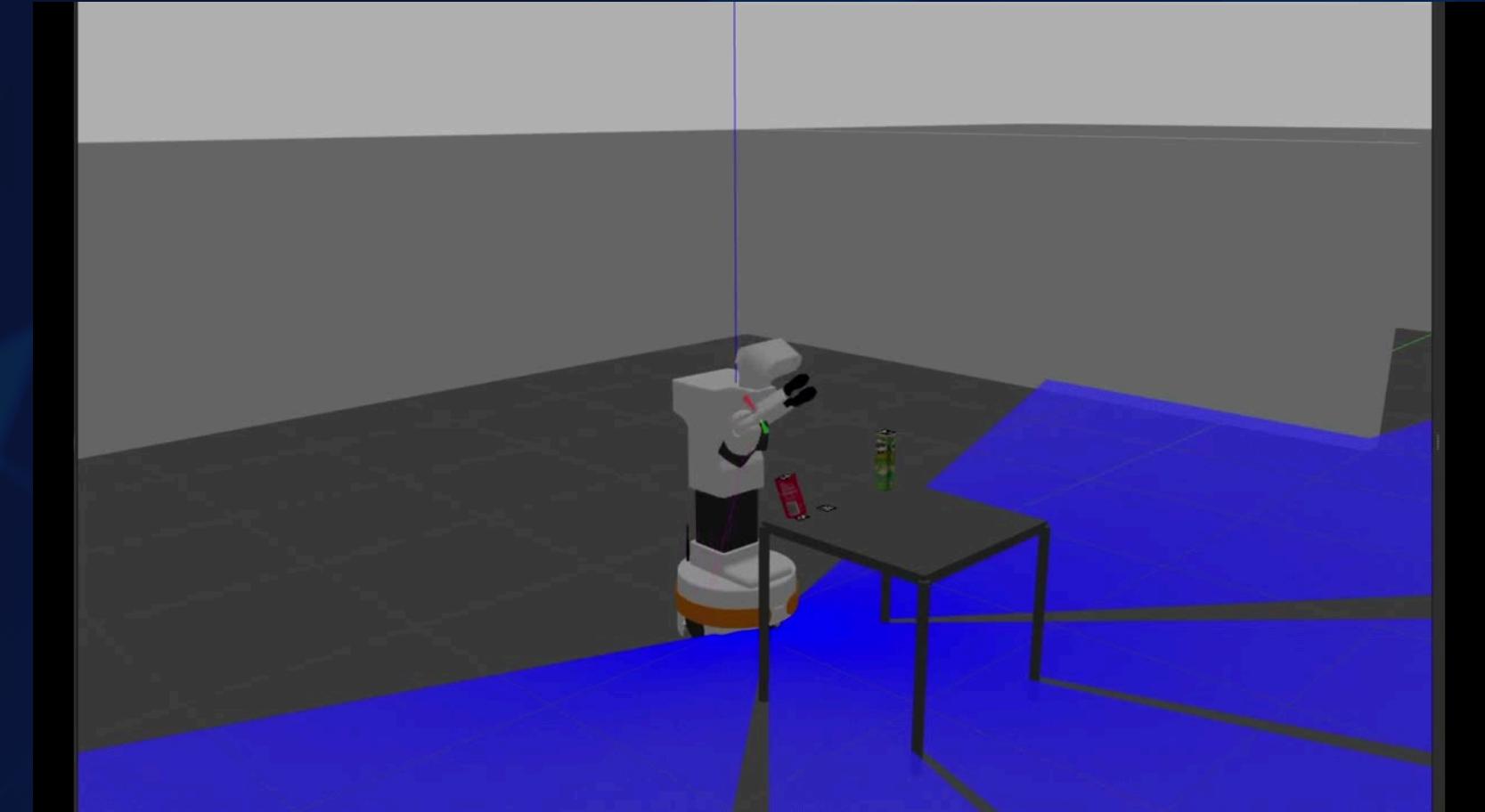


# TASK 3: ESECUZIONE

Pick and place della  
Coca-cola



Pick and place delle  
Pringles



# **GRAZIE PER L'ATTENZIONE**

