دانشکده مهندسی مکانیک

پروژه دوم – درس رباتیک پیشرفته

مدل سازی ربات RoArm-M3 Pro در شبیه ساز و ایجاد مسیر حرکت و ارسال زوایای جوینت ها به ربات واقعی

رشته مهندسی مکانیک گرایش مکاترونیک

نام دانشجوها:

سیدمهدی سرفرازی

امین پارسافر

محمدحسین ابراهیمی

استاد:

دکتر سیدحسن ذبیحی فر

پاییز ۱۴۰۴


بسم الله الرحمن الرحيم

فهرست مطالب

فهرست اشکال

مقدمه

در سال‌های اخیر، ربات‌های بازویی رومیزی و آموزشی به عنوان ابزاری قدرتمند در آموزش مهندسی رباتیک، مکاترونیک و هوش مصنوعی جایگاه ویژه‌ای پیدا کرده‌اند. این ربات‌ها با هزینه نسبتاً پایین، ساختار ماژولار و پشتیبانی از اکوسیستم‌های نرم‌افزاری پیشرفته مانند ROS۲، امکان دسترسی دانشجویان، پژوهشگران و علاقه‌مندان به مفاهیم پیچیده‌ای نظیر سینماتیک مستقیم و معکوس، برنامه‌ریزی مسیر، کنترل هوشمند و انتقال دانش از شبیه‌سازی به واقعیت (Sim-to-Real) را فراهم آورده‌اند.

ربات RoArm-M۳ Pro محصول شرکت Waveshare، یکی از نمونه‌های برجسته در این دسته است. این بازوی رباتیک هوشمند با ۵ + ۱ درجه آزادی (۵ محور اصلی + گریپر مستقل)، ساختار سبک آلومینیومی، استفاده از سروموتورهای باس سریال تمام‌فلزی با گشتاور بالا (مدل ST۳۲۳۵)، کنترلر مرکزی مبتنی بر ESP۳۲ و پشتیبانی کامل از پروتکل‌های بی‌سیم (Wi-Fi، Bluetooth، ESP-NOW) و ROS۲، بستری مناسب برای پروژه‌های آموزشی، تحقیقاتی و توسعه الگوریتم‌های هوش مصنوعی فراهم می‌کند.

پروژه حاضر با عنوان «مدل‌سازی ربات RoArm-M۳ Pro در شبیه‌ساز و ایجاد مسیر حرکت و ارسال زوایای جوینت‌ها به ربات واقعی»، می باشد.

هدف اصلی، طی کردن چرخه کامل از مدل‌سازی ریاضی (Denavit-Hartenberg)، شبیه‌سازی بصری و کنترلی در محیط‌های RoboDK و Rviz، توسعه الگوریتم Pick-and-Place، ساخت نود جدید ROS۲ و نهایتاً انتقال دستورات به سخت‌افزار واقعی بوده است. این گزارش تلاش دارد تا چالش‌های واقعی کار با یک ربات غیرصنعتی را مستند کرده و راه‌حل‌های عملی ارائه دهد.

بیان مسئله

چگونه می‌توان مدل‌سازی، شبیه‌سازی و برنامه‌ریزی حرکت یک ربات بازویی آموزشی غیرصنعتی را به‌گونه‌ای انجام داد که الگوریتم‌های کنترلی (مانند Pick-and-Place) با حداقل اختلاف از محیط شبیه‌سازی به ربات واقعی منتقل شود و عملکرد قابل قبولی داشته باشد؟

به‌طور خلاصه، هدف حل چالش‌های زیر است:

- مدل‌سازی سینماتیکی دقیق با وجود ساختار غیراستاندارد
- انتخاب و پیاده‌سازی موفق شبیه‌ساز مناسب
- برنامه‌ریزی مسیر بدون برخورد و singularity

- ایجاد ارتباط دوطرفه پایدار Sim-to-Real
- کاهش منابع خطا و اختلاف مشاهده‌شده بین شبیه‌سازی و سخت‌افزار واقعی

ضرورت مدل‌سازی ربات‌های بازو

مدل‌سازی ربات‌های بازویی (چه در سطح سینماتیکی، چه دینامیکی و چه در محیط شبیه‌سازی) امروزه یکی از مراحل ضروری و غیرقابل چشم‌پوشی در طراحی، توسعه، آموزش و پژوهش رباتیک محسوب می‌شود. دلایل اصلی این ضرورت عبارتند از:

۱. کاهش هزینه و ریسک آزمایش روی سخت‌افزار واقعی
۲. توسعه و دیباگ سریع‌تر الگوریتم‌ها
۳. یادگیری و آموزش مفاهیم پیچیده بدون نیاز به سخت‌افزار گران
٤. پر کردن شکاف Sim-to-Real
٥. امکان آزمایش سناریوهای خطرناک یا غیرممکن
٦. بهینه‌سازی طراحی و پارامترها پیش از ساخت
۷. پشتیبانی از توسعه نرم‌افزار پیشرفته
۸. تکرارپذیری و مستندسازی نتایج

به‌طور خلاصه، مدل‌سازی ربات‌های بازو دیگر یک گزینه اختیاری نیست؛ بلکه پیش‌نیاز ضروری برای صرفه‌جویی در زمان و هزینه، افزایش ایمنی، تسریع توسعه، آموزش مؤثر و دستیابی به عملکرد قابل اعتماد در دنیای واقعی است به‌ویژه در مورد ربات‌های آموزشی و تحقیقاتی مانند RoArm-M۳ Pro که محدودیت‌های سخت‌افزاری قابل‌توجهی دارند.

اهداف اصلی پروژه

اهداف اصلی این پروژه به شرح زیر است:

۱. مدل‌سازی سینماتیکی دقیق ربات RoArm-M۳ Pro
۲. پیاده‌سازی موفق مدل ربات در محیط شبیه‌سازی
۳. توسعه الگوریتم برنامه‌ریزی حرکت Pick-and-Place
٤. ساخت و پیاده‌سازی نود جدید ROS۲ برای کنترل یکپارچه
٥. ایجاد معماری Sim-to-Real پایدار و کم‌تأخیر

٦. ارزیابی و مقایسه عملکرد شبیه‌سازی و واقعی

۷. مستندسازی چالش‌ها و راه‌حل‌ها برای کاربردهای آموزشی

بررسی فنی ربات RoArm-M۳ Pro

در این بخش، به بررسی مشخصات فنی و ساختاری ربات RoArm-M3 Pro می‌پردازیم. آشنایی دقیق با ویژگی‌های مکانیکی، الکترونیکی و نرم‌افزاری این ربات، پایه اصلی برای مدلسازی سینماتیکی، شبیه‌سازی و در نهایت کنترل عملی آن است. بدون شناخت کافی از سخت‌افزار و محدودیت‌های آن، انجام مدلسازی دقیق و انتقال موفق الگوریتم‌ها از شبیه‌ساز به دنیای واقعی امکان‌پذیر نخواهد بود.

اطلاعات ارائه‌شده در این بخش عمدتاً از مستندات رسمی شرکت سازنده (Waveshare) و به‌ویژه از صفحه ویکی این محصول [1] استخراج و تنظیم شده‌اند. این مرجع، جزئیات فنی معتبری را در مورد ابعاد، درجات آزادی، مشخصات موتورها، کنترلر و پروتکل‌های ارتباطی ارائه می‌دهد که به عنوان مبنا برای تحلیل‌های بعدی مورد استفاده قرار گرفته‌اند.

در ادامه، ابتدا ساختار مکانیکی و درجات آزادی ربات بررسی می‌شود. سپس عملگرها (سروموتورها) و سیستم انتقال قدرت معرفی خواهند شد. در بخش بعدی، کنترلر مرکزی و روش‌های ارتباطی سخت‌افزار مورد بحث قرار می‌گیرد. در پایان نیز فضای کاری (Workspace) ربات و محدودیت‌های فیزیکی آن تحلیل می‌شود. این بررسی، چارچوب لازم برای انجام محاسبات سینماتیکی و دینامیکی در مراحل بعدی پروژه را فراهم می‌کند.

ساختار مکانیکی

RoArm-M۳ Pro دارای ساختاری آلومینیومی سبک با طراحی ماژولار است که امکان مونتاژ، تعمیر و توسعه آسان را فراهم می‌کند. این ساختار برای کاربردهای رومیزی (Desktop Applications) بهینه شده است.

شکل ۱ تصویر ربات RoArm-M۳ Pro [۱]

درجات آزادی(Degrees of Freedom)

این بازوی رباتیک دارای **۱+۵ درجه آزادی** است که شامل موارد زیر می‌شود:

- چرخش پایه(Base Rotation)
- مفصل شانه(Shoulder Joint)
- مفصل آرنج(Elbow Joint)
- مچ دو درجه آزادی Pitch) و(Roll
- یک درجه آزادی مجزا برای عملگر نهایی (گریپر)

این تعداد درجات آزادی، ربات را قادر می‌سازد تا موقعیت و جهت‌گیری مناسبی از اندافکتور را در فضای سه‌بعدی فراهم کند و بسیاری از وظایف جابجایی ساده را انجام دهد.

دامنه حرکتی مفاصل

- چرخش پایه: ۰ تا ۳۶۰ درجه
- مفصل شانه: حدود ۰ تا ۱۸۰ درجه
- مفصل آرنج: حدود ۲۲۵ درجه
- مفاصل مچ: بین ۱۳۵ تا ۲۷۰ درجه بسته به محور

این دامنه‌ها انعطاف‌پذیری مناسبی در حرکت ایجاد می‌کنند، هرچند همچنان محدودتر از ربات‌های صنعتی هستند.

معرفی عملگرها

در نسخه Pro، از سروموتورهای باس سریال با گشتاور بالا استفاده شده است. این ربات به طور کلی شامل:

- ۶ سروموتور باس سریال
- سروهای تمام‌فلزی مدل ST۳۲۳۵ در مفاصل اصلی

هر سروموتور مجهز به **انکودر مغناطیسی ۱۲ بیتی** است که امکان اندازه‌گیری دقیق موقعیت زاویه‌ای مفاصل را فراهم می‌کند. این ویژگی نقش مهمی در بهبود دقت کنترل و تکرارپذیری حرکات دارد.

سیستم انتقال قدرت

سیستم انتقال قدرت عمدتاً به صورت **هدایت مستقیم** (**Direct Drive**) طراحی شده است. در این روش، گشتاور خروجی سروموتور مستقیماً به مفصل منتقل می‌شود. مزایای این ساختار عبارت‌اند از:

- کاهش لقی مکانیکی
- پاسخ دینامیکی سریع‌تر
- کاهش پیچیدگی مکانیکی و هزینه نگهداری

در مقابل، این طراحی منجر به محدودیت در گشتاور قابل تحمل و بار مجاز می‌شود.

بررسی کنترلر و پروتکل‌های ارتباطی سخت‌افزار

• کنترلر مرکزی

کنترلر اصلی این ربات **ESP۳۲-WROOM-۳۲** است که دارای پردازنده دوهسته‌ای با فرکانس ۲۴۰ مگاهرتز می‌باشد. این کنترلر به صورت همزمان قابلیت پردازش دستورات حرکتی و مدیریت ارتباطات بی‌سیم را فراهم می‌کند.

• ارتباطات سخت‌افزاری

روش‌های ارتباطی پشتیبانی‌شده شامل:

- USBو UART برای ارتباط سیمی
- Wi-Fiو Bluetooth داخلی
- پروتکل ESP-NOW برای ارتباط بی‌سیم کم‌تأخیر.

- **پشتیبانی نرم‌افزاری**

RoArm-M۳ Pro از ROS۲ پشتیبانی می‌کند و دارای فایل‌های URDF برای مدل‌سازی سینماتیکی و دینامیکی است. این قابلیت امکان استفاده از ابزارهای استانداردی نظیر MoveIt، شبیه‌سازی و توسعه الگوریتم‌های پیشرفته کنترل را فراهم می‌سازد.

تحلیل فضای کاری و محدودیت‌های فیزیکی

با توجه به چرخش کامل پایه و طول بازو، فضای کاری ربات تقریباً به صورت یک حجم استوانه‌ای با مشخصات زیر قابل تقریب است:

- شعاع افقی مؤثر: حدود ۰.۵ متر
- قطر دسترسی افقی: حدود ۱.۱۲ متر
- ارتفاع دسترسی عمودی: حدود ۰.۸ متر

این فضای کاری برای انجام وظایف آموزشی، آزمایشگاهی و جابجایی اشیاء کوچک مناسب است.

مهم‌ترین محدودیت‌های این ربات عبارت‌اند از:

- **ظرفیت بارگذاری پایین** حدود ۵۰۰ گرم در فاصله متوسط
- عدم مناسب بودن برای محیط‌های صنعتی و بارهای سنگین
- حساسیت نسبی سازه سبک در برابر نیروهای خارجی بزرگ

این محدودیت‌ها نشان می‌دهد که RoArm-M۳ Pro بیشتر برای کاربردهای تحقیقاتی و آموزشی طراحی شده است.

تحلیل ریاضی و مدل‌سازی سینماتیکی

تحلیل مفاصل و درجات آزادی در RoArm-M۳ Pro

ساختار ۵+۱ درجه آزادی این ربات به معنای وجود ۵ مفصل اصلی برای تعیین موقعیت و جهت‌گیری در فضا و یک مفصل اضافی برای عملکرد گیره است. این پیکربندی به ربات اجازه می‌دهد تا در یک فضای کاری دایره‌ای به قطر ۱.۱۲ متر به صورت ۳۶۰ درجه عمل کند.

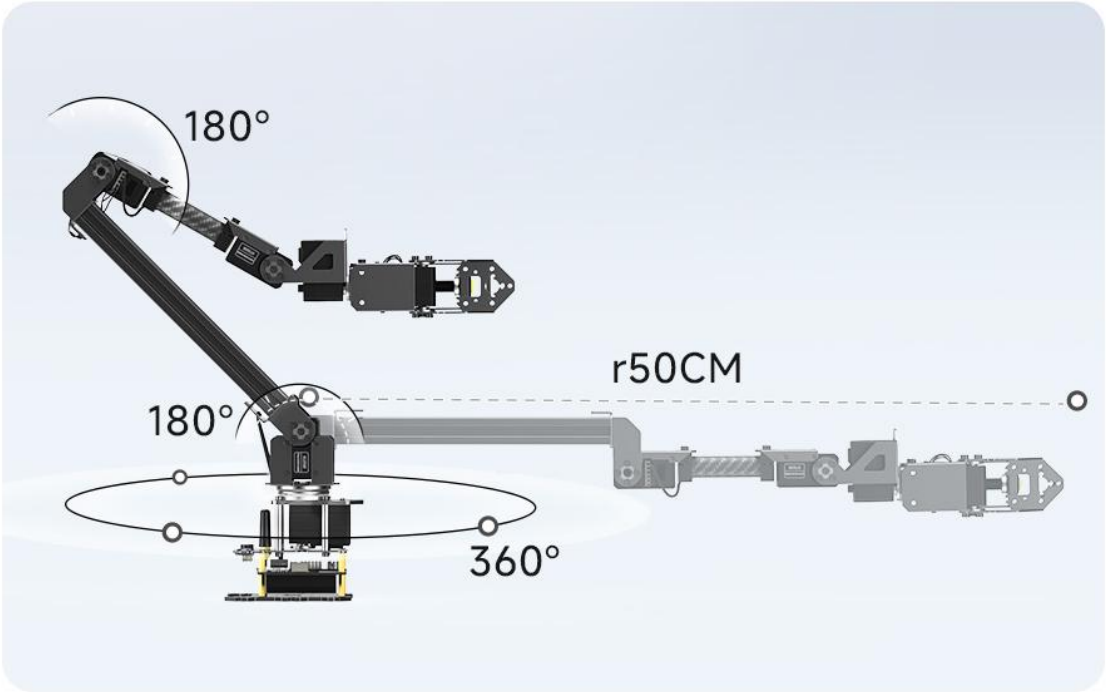
تفکیک وظایف مفاصل:

مفاصل این ربات به شرح زیر دسته‌بندی می‌شوند:

- مفصل پایه (Base Joint): مسئول چرخش افقی کل بدنه ربات در بازه ۳۶۰ درجه. این مفصل محور اصلی Z_0 را تعریف می‌کند.
- مفصل شانه (Shoulder Joint): این مفصل از تکنولوژی "Dual-Drive" بهره می‌برد که در آن دو سروو موتور به صورت هماهنگ عمل می‌کنند تا گشتاور را دوبرابر کرده و توانایی بلند کردن بازو را افزایش دهند.
- مفصل آرنج (Elbow Joint): تعیین‌کننده دسترسی عمودی و افقی بازو. بازه حرکتی این مفصل در نسخه‌های جدید تا ۲۲۵ درجه ارتقا یافته است.
- مفصل مچ ۱ (Wrist Pitch): وظیفه تغییر زاویه نوک بازو نسبت به افق (Pitch) را بر عهده دارد.
- مفصل مچ ۲ (Wrist Roll): امکان چرخش محوری گیره را فراهم می‌کند که برای عملیات‌هایی مانند پیچاندن یا تراز کردن اشیا حیاتی است.
- گیره (Gripper/EOAT): مفصل نهایی که برای گرفتن و رها کردن اشیا با دقت نیروی قابل کنترل طراحی شده است.

چارچوب‌بندی مفاصل و پارامترهای Denavit-Hartenberg

اگر به سایت سازنده ربات فوق مراجعه شود، دیتاهای زیادی مانند طول و ابعاد لینک‌ها بدست خواهد آمد. در شکل، میتوان نحوه حرکت ربات و میزان محدودیت‌های زوایای لینک‌های آن را به خوبی مشاهده کرد.



شکل ۲ نحوه حرکت ربات در مفاصل خود[۱]

همچنین در شکل زیر، ابعاد ربات به جهت مدلسازی برای بدست آوردن جدول DH نوشته شده است که میتوان از آن نیز بهره برد. برای محاسبات و نوشتن جدول DH نیاز به قرار دادن زاویه‌ها و محورها به صورت استاندارد و صحیح است. بدین ترتیب، محورها مانند شکل زیر بدست آمد.

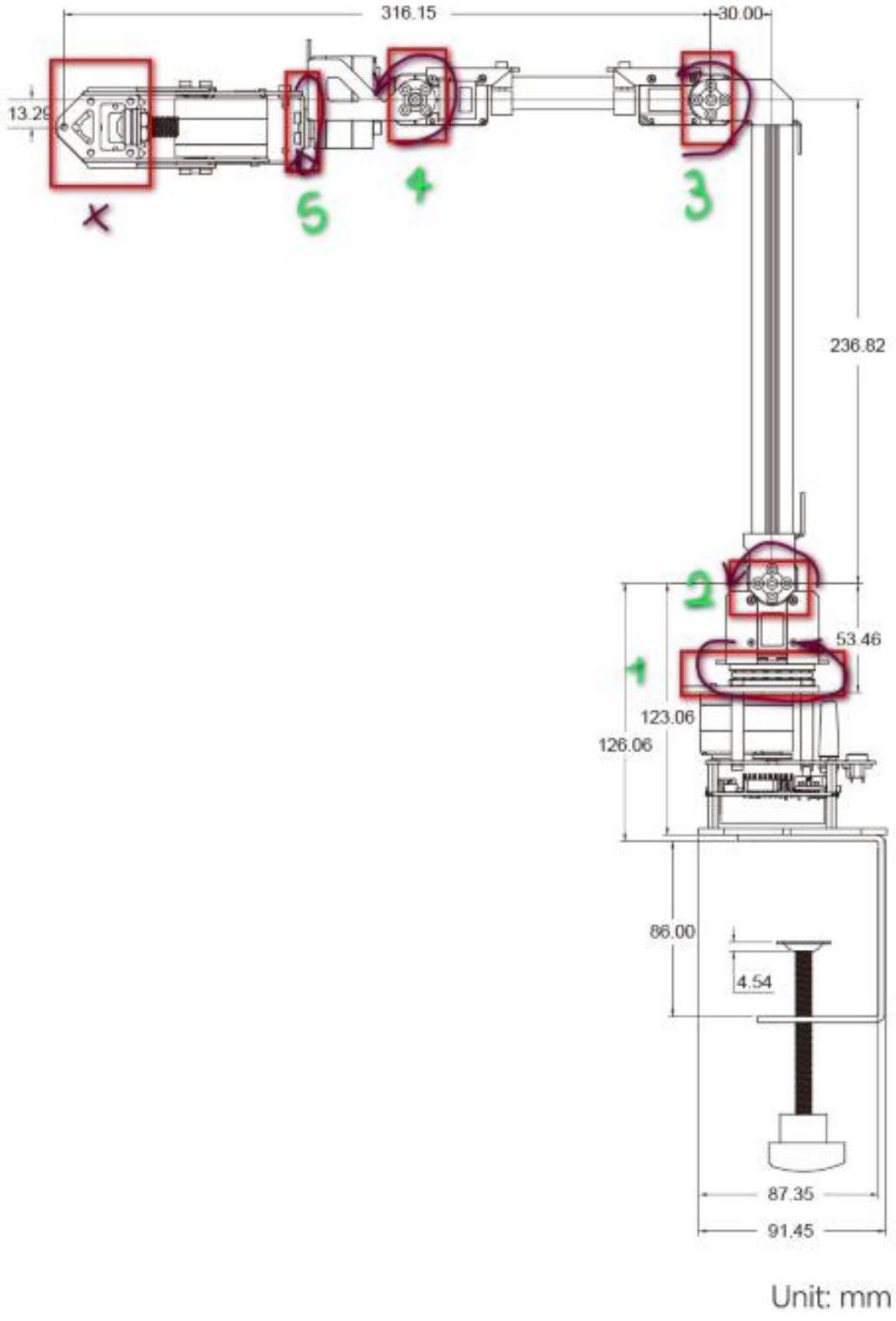
L_1 (ارتفاع پایه): از کف تا محور مفصل شانه برابر با ۱۲۶.۰۶ + ۵۳.۴۶ = ۱۷۹.۵۲ mm

L_2 (طول بازو - Shoulder Link): فاصله بین محور شانه و آرنج برابر با ۲۳۶.۸۲

L_{offset} (آفست آرنج): جابجایی جانبی بین شانه و آرنج که در تصویر با عدد ۳۰.۰۰ میلی متر مشخص شده است.

L_3 (طول ساعد - Forearm Link): فاصله محور آرنج تا مرکز مفصل مچ برابر با ۳۱۶.۱۵

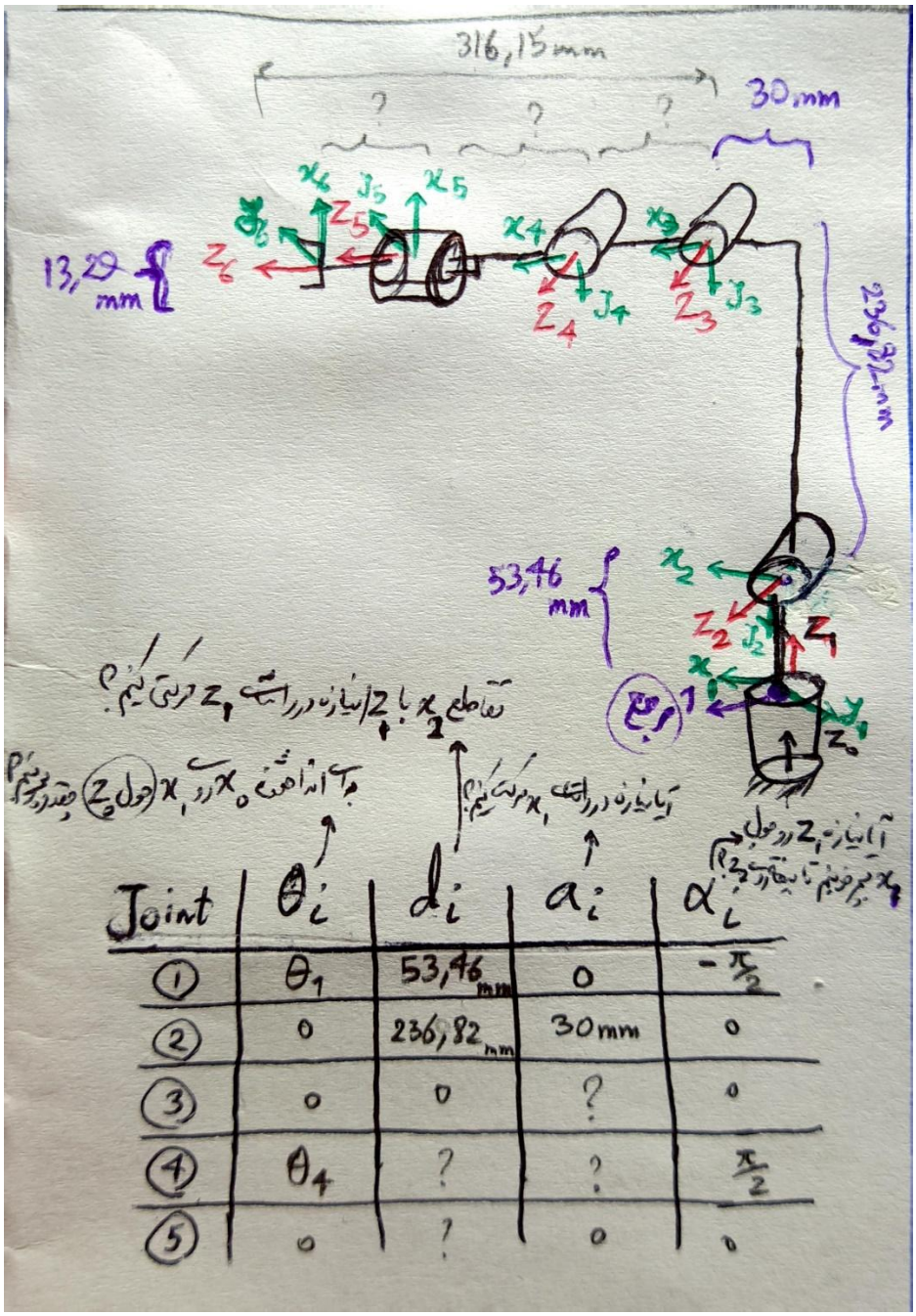
L_{end} (آفست ابزار): فاصله نهایی تا نوک گریپر که حدود ۱۳.۲۹ میلیمتر (آفست عمودی) در نظر گرفته می‌شود.



شکل ۳ ابعاد ربات برای بدست آوردن جدول DH و مختصات هر مفصل

جدول پارامترهای DH (Standard DH Table)

با اصلاح اشتباهات موجود در محاسبات دستی نوشته شده در تصویر زیر (مانند مقدار d_1 که کل ارتفاع پایه را شامل نمی‌شد)، جدول استاندارد دناویت هارتنبرگ سر انجام بدست می‌آید.



شکل ۴ محاسبات دستی برای بدست آوردن مختصات و نوشتن جدول استاندارد DH

محاسبات دستی کاملا از روی تصویر موجود در دیتاشیت انجام شد و به دلیل نبود اطلاعاتی نظیر علامت سوال‌های موجود در دیتاشیت، مجبور به اندازه گیری حدودی و بررسی آن از روی فایل‌های Cad شدیم. بدین ترتیب، جدول استاندارد دناویت هارتبرگ تشکیل شد و به صورت زیر درآمد:

جدول ۱ پارامترهای دناویت هارتبرگ استاندارد برای ربات RoArm M۳ Pro

مفصل (i)	θi	di (mm)	ai (mm)	αi (rad)
۱ (Base)	θ1∗	179.52	0	π/2
۲ (Shoulder)	θ2∗	30	236.82	0
۳ (Elbow)	θ3∗	0	316.15	0
۴ (Wrist P)	θ4∗	0	0	−π/2
۵ (Wrist R)	θ5∗	Ltool	0	0

نکته: ستون θi* متغیرهای کنترلی هستند که توسط انکودرهای ۱۲ بیتی ربات خوانده می‌شوند.

محاسبات سینماتیک مستقیم (Forward Kinematics)

برای هر لینک، ماتریس انتقال تکی (A_i) از فرمول زیر به دست می‌آید:

بر اساس جدول ۱پارامترهای مورد استفاده در کد پیاده سازی میشود. همچنین ماتریس تبدیل همگن برای هر لینک طبق رابطه زیر قابل محاسبه خواهد بود:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

برای یافتن موقعیت نهایی ابزار در فضای دکارتی، ماتریس نهایی از ضرب این ۵ ماتریس به دست می‌آید:

$$^0_5T = ^0_1T \cdot ^1_2T \cdot ^2_3T \cdot ^3_4T \cdot ^4_5T \cdot ^5_6T$$

^0_1T انتقال از پایه به شانه با چرخش حول محور عمودی.

^1_2T لحاظ کردن طول بازو ۲۳۶.۸۲و آفست جانبی ۳۰ میلی متر است.

و بقیه ماتریس های تبدیل نیز با توجه به شکل بدست می‌آیند. در این ماتریس نهایی ^0_5T، ستون چهارم نشان‌دهنده‌ی مختصات مرکز گریپر است.

برای انجام بخش تحلیل ریاضی و سینماتیکی پروژه، کد پایتون جامع و کامل بر اساس استانداردهای Denavit-Hartenberg (DH) نوشته شد. این کد پارامترهای اختصاصی ربات RoArm-M۳ Pro را به عنوان ورودی محاسبات استفاده می‌کند و محاسبات سینماتیک مستقیم، ماتریس تبدیل کل و شبیه‌سازی فضای کاری را انجام می‌دهد.

سینماتیک معکوس (Inverse Kinematics – Analytical Solution)

در سینماتیک معکوس، هدف یافتن زوایای مفاصل $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ بر اساس موقعیت (x, y, z) و جهت‌گیری مطلوب end-effector است. با توجه به ساختار ربات، از روش هندسی-تحلیلی استفاده می‌کنیم. در کاربردهای مهندسی، معمولاً موقعیت end-effector مشخص است و ما به دنبال زوایای مفاصل هستیم. مقاله‌ها برای افزایش تعداد معادلات و حل دقیق‌تر، از ضرب معکوس ماتریس‌های تبدیل در طرفین معادله استفاده کرده است:

$$ {}^{0}_{1}T^{-1} \cdot {}^{0}_{6}T = {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T $$

الف) محاسبه θ_1 (چرخش پایه):

با توجه به وجود آفست $d_2 = 30mm$ در مفصل شانه، تصویر بازو بر صفحه XY یک مثلث قائم‌الزاویه ایجاد می‌کند.

$$ \theta_1 = atan2(y, x) - atan2\left(d_2, \pm\sqrt{x^2 + y^2 - d_2^2}\right) $$

نکته: علامت $\pm$ نشان‌دهنده دو راه حل Left-arm و Right-arm است.

ب) محاسبه θ_2, θ_3 (صفحه بازو):

ابتدا مختصات هدف را به صفحه بازو (صفحه ۲ بعدی شامل لینک‌های L_۲ و L_۳) منتقل می‌کنیم. فرض کنید r فاصله افقی موثر باشد:

$$ r = \sqrt{x^2 + y^2 - d_2^2} $$

$$ s = z - d_1 $$

اکنون با یک مساله‌ی ۲ لینک صفحه‌ای روبرو هستیم. طبق قانون کسینوس‌ها در مثلث تشکیل شده توسط لینک‌ها و وتر داریم:

$$ \cos\theta_3 = \frac{r^2 + s^2 - L_2^2 - L_3^2}{2L_2L_3} $$

$$ \theta_3 = atan2\left(\pm\sqrt{1 - \cos^2\theta_3}, \cos\theta_3\right) $$

$$ \theta_2 = atan2(s, r) - atan2(L_3\sin\theta_3, L_2 + L_3\cos\theta_3) $$

ج) محاسبه θ_4, θ_5 (مچ):

این زوایا معمولاً بر اساس ماتریس دوران نهایی مطلوب (R_{target}) و با کسر دوران‌های به دست آمده از بازو (R_3^0) محاسبه می‌شوند:

$$R_5^3 = (R_3^0)^T \cdot R_{target}$$

زوایای مچ از درایه‌های ماتریس دوران باقی‌مانده استخراج می‌شوند.

حل تحلیلی ماتریس ژاکوبین (Jacobian Matrix)

ماتریس ژاکوبین (J) رابطه‌ی خطی بین سرعت مفاصل $\dot{q}$ و سرعت‌های دکارتی ابزار (v, ω) را برقرار می‌کند:

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix} = J(q)\dot{q}$$

برای ربات ۵ درجه آزادی ما، ژاکوبین یک ماتریس ۶*۵ است. هر ستون J_i مربوط به مفصل i بوده و به صورت زیر محاسبه می‌شود (چون تمام مفاصل دورانی هستند):

$$J_i = \begin{bmatrix} z_{i-1} \times (o_5 - o_{i-1}) \\ z_{i-1} \end{bmatrix}$$

که در آن:

z_{i-1}: محور دوران مفصل i (ستون سوم ماتریس انتقال $^0_{i-1}T$

o_{i-1}: مبدأ مختصات فریم $i-1$

o_5: موقعیت نهایی اندافکتور.

تحلیل سرعت و نقاط تکین (Singularity Analysis)

تحلیل سرعت به ما می‌گوید که در هر وضعیت، ربات با چه سرعتی می‌تواند در فضای دکارتی حرکت کند.

الف) رابطه‌ی سرعت:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$$

اگر $\dot{q}$ بردار سرعت موتورها باشد، بیشترین سرعت خطی ابزار زمانی رخ می‌دهد که بردار $\dot{q}$ در جهت بزرگترین مقدار ویژه ماتریس $J^T J$ باشد.

ب) نقاط تکین (Singularities):

نقاطی هستند که در آن‌ها ماتریس ژاکوبین دچار کاهش رتبه (Rank Deficiency) شده و ربات یک یا چند درجه آزادی خود را در فضای دکارتی از دست می‌دهد. در این نقاط دترمینان ژاکوبین (برای بخش مربع آن) صفر می‌شود:

۱. تکینگی بازو (Elbow Singularity): وقتی $\theta_3 = 0$ یا ۱۸۰ باشد (بازو کاملاً کشیده یا کاملاً جمع شده). در این حالت ربات نمی‌تواند در راستای طول بازو حرکت آنی داشته باشد.

۲. تکینگی پایه (Shoulder Singularity): وقتی مرکز مچ دقیقاً بالای محور گردش پایه (Z_0) قرار گیرد. در این حالت $x^2 + y^2 - d^2 = 0$ و تعیین θ_1 غیرممکن می‌شود.

به جهت جاگذاری در روابط، بایستی مقادیر زیر را جاگذاری نمود:

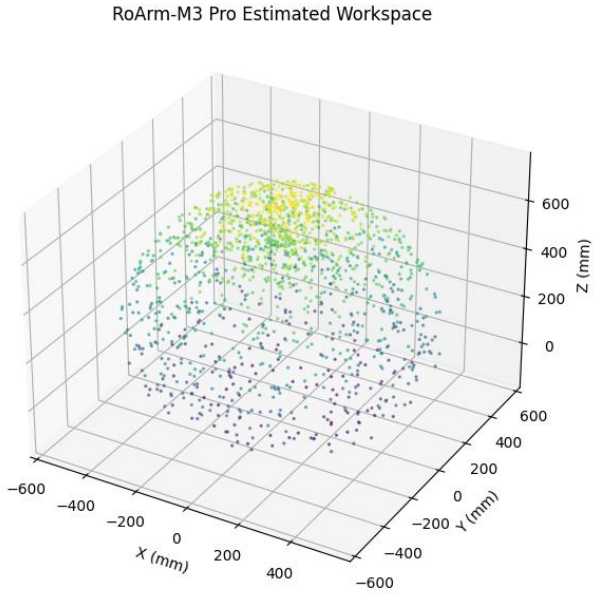
$$d_2 = 30.00, d_1 = 179.52$$
$$a_3 = 316.15, a_2 = 236.82$$
$$L_{tool} = 13.29$$

خروجی کد سینماتیک و فضای کاری ربات

پس از اجرای کد موجود در فلیل Project۲_Full_Parsafar_Sarfarazi_Ebrahimi.ipynb خواهیم دید که نتایج به صورت زیر قابل مشاهده خواهند بود:

```
فرمول مختصات دکارتی End-Effector:
X = ۳۰.۰*sin(theta۱) + ۲۳۶.۸۲*cos(theta۱)*cos(theta۲) + ۳۱۶.۱۵*cos(theta۱)*cos(theta۲ + theta۳)
Y = ۲۳۶.۸۲*sin(theta۱)*cos(theta۲) + ۳۱۶.۱۵*sin(theta۱)*cos(theta۲ + theta۳) - ۳۰.۰*cos(theta۱)
Z = ۲۳۶.۸۲*sin(theta۲) + ۳۱۶.۱۵*sin(theta۲ + theta۳) + ۱۷۹.۵۲
```

موقعیت نهایی در حالت Home (زوایای صفر): [۱۷۹.۵۲ -۳۰. -۵۵۲.۹۷]



شکل ۵ بدست آوردن فضای کاری ربات با کد پایتون

پیاده‌سازی مدل در محیط شبیه‌سازی

انتخاب و پیکربندی موتور شبیه‌ساز

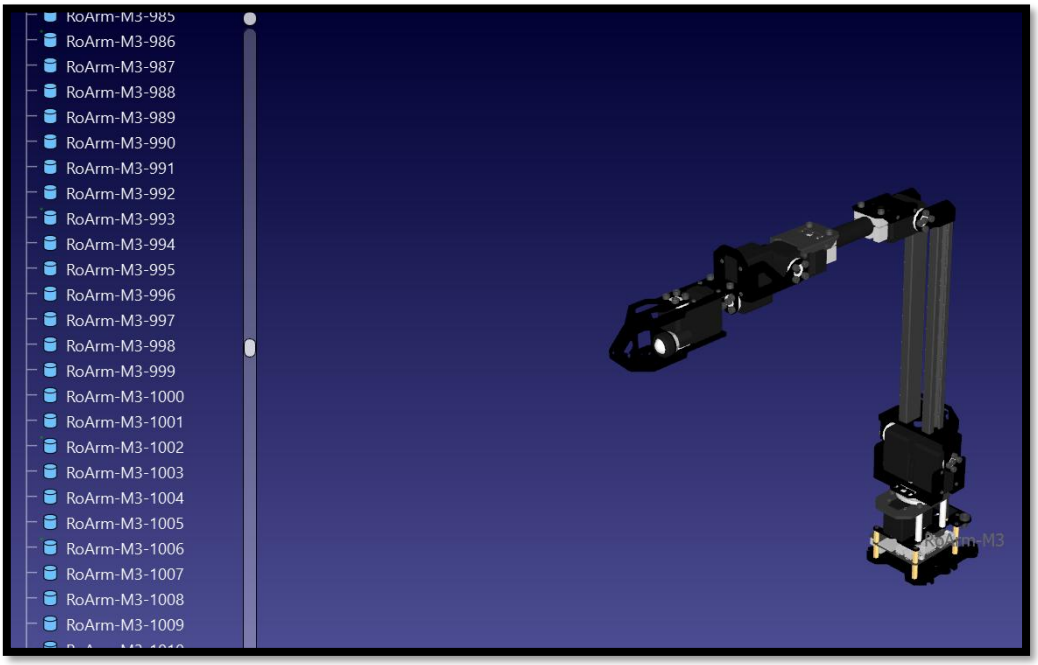
انتخاب محیط شبیه سازی مناسب نقش مهمی در کنترل درست و سهولت پیاده سازی کنترل ایفا میکند. با انتخاب درست محیط شبیه سازی، در ادامه کار، هنگام انتقال کنترل به ربات واقعی، عملکرد مناسب تری را شاهد خواهیم بود. با توجه به ماهیت پروژه ابتدا نرم افزار RoboDK برای شبیه سازی انتخاب شد. در نهایت پیاده سازی شبیه ساز با استفاده از Rviz صورت گرفت که در ادامه به صورت مفصل به بررسی هر کدام و دلیل تغییر شبیه ساز خواهیم پرداخت.

بررسی شبیه سازی در محیط RoboDK

در این بخش، فرآیند بارگذاری، مدل‌سازی و تلاش برای شبیه‌سازی بازوی رباتیک **RoArm-M۳ Pro** در محیط نرم‌افزاری **RoboDK** مورد بررسی قرار می‌گیرد. هدف اصلی، ایجاد یک مدل قابل حرکت از ربات و اجرای دستورات حرکتی در محیط شبیه‌سازی بوده است.

- **بارگذاری مدل ربات در RoboDK و مشکلات فایل STEP اولیه**

در مرحله نخست، فایل CAD ربات که به صورت STEPارائه شده بود، در محیط RoboDK بارگذاری شد. با این حال، اولین مشکل اساسی در این مرحله نمایان گردید. فایل STEP اصلی شامل تعداد بسیار زیادی قطعات مجزا بود که به دلیل طراحی آموزشی و جزئیات زیاد ظاهری ربات ایجاد شده بودند.



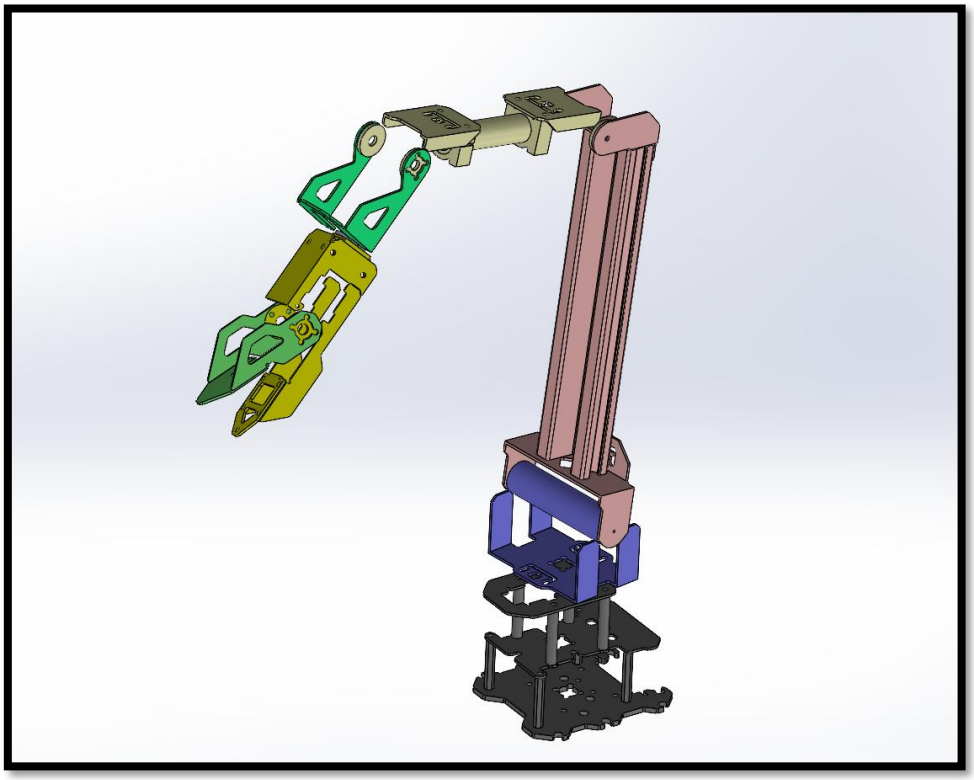
شکل ۶ قابلیت Split برای تعریف مکانیزم

در هنگام استفاده از قابلیت *Split* برای تعریف مکانیزم، مدل به چند صد قطعه مستقل تقسیم می‌شد. این موضوع عملاً امکان تعریف صحیح مفاصل (Joints) و لینک‌ها را از بین می‌برد و استفاده مستقیم از فایل STEP اصلی برای ایجاد مکانیزم را غیرممکن می‌ساخت.

- **بازطراحی مدل ربات با هدف ساده‌سازی ساختار مکانیکی**

به منظور رفع مشکل فوق، مدل ربات مجدداً در نرم‌افزار SolidWorks بازطراحی شد. در این بازطراحی، تمرکز اصلی بر:

- کاهش تعداد قطعات
- ادغام اجزای غیرضروری
- حفظ ساختار کلی و سینماتیک ربات



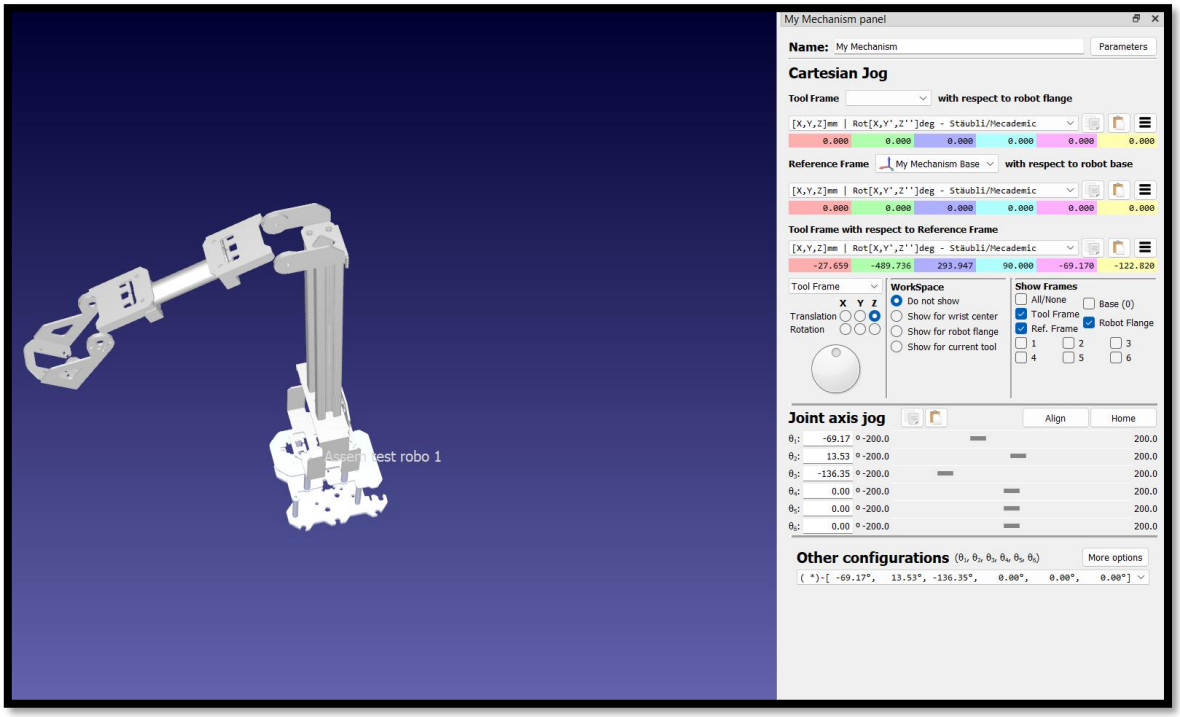
شکل ۷ مدل سازی انجام شده در سالیدوورک

نتیجه این فرآیند، ایجاد مدلی با تعداد قطعات محدود و منطقی برای هر لینک ربات بود. پس از اتمام بازطراحی، مدل جدید مجدداً به فرمت **STEP** تبدیل و وارد محیط RoboDK شد که این بار از نظر ساختاری برای تعریف مکانیزم مناسب‌تر بود.

- **تلاش برای تعریف مکانیزم و چالش‌های سینماتیکی**

پس از وارد کردن مدل ساده‌شده، مرحله تعریف مکانیزم ربات آغاز شد. در این مرحله مشخص گردید که به دلیل ماهیت آموزشی و غیرصنعتی طراحی ربات، ساختار آن با الگوی استاندارد ربات‌های صنعتی که RoboDK برای آن‌ها بهینه شده است، تطابق کامل ندارد.

مشکل اصلی در این بخش، عدم هم‌راستایی اوریجین‌ها و محورهای حرکتی مفاصل بود. هر لینک ربات از اوریجین متفاوتی حرکت می‌کرد و تعریف یک زنجیره سینماتیکی پیوسته و استاندارد (Serial Chain) امکان‌پذیر نبود.



شکل ۸ مکانیزم در روبو دی کی

۴–۷ .آزمون و خطا و نتیجه نهایی شبیه‌سازی

برای رفع این چالش، چندین روش به صورت آزمون و خطا مورد بررسی قرار گرفت، از جمله:

- تغییر اوریجین‌ها و محورهای مختصات در فایل CAD
- اصلاح دستی محورها در محیط RoboDK
- تلاش برای تعریف مفاصل به صورت سفارشی

با وجود این تلاش‌ها، به دلیل عدم تطابق کامل ساختار ربات با چارچوب سینماتیکی مورد انتظار نرم‌افزار RoboDK، امکان تعریف مکانیزم پایدار و اجرای حرکت کنترلی ربات فراهم نشد. در نتیجه، شبیه‌سازی دینامیکی و اجرای دستورات حرکتی در این محیط با محدودیت جدی مواجه گردید.

- **جمع‌بندی**

نتایج به‌دست‌آمده نشان می‌دهد که اگرچه RoboDK ابزار قدرتمندی برای شبیه‌سازی ربات‌های صنعتی استاندارد است، اما برای ربات‌های آموزشی با طراحی غیرمرسوم و جزئیات ظاهری زیاد، نیاز به بازطراحی عمیق سینماتیکی و حتی تعریف دستی مدل ریاضی ربات وجود دارد. این موضوع یکی از چالش‌های اصلی در شبیه‌سازی ربات‌های آموزشی در محیط‌های صنعتی‌محور محسوب می‌شود.

بررسی شبیه سازی در محیط Rviz

پس از بررسی‌های انجام‌شده در محیط RoboDK و مواجهه با چالش‌های ناشی از ساختار غیراستاندارد و آموزشی رباتRoArm-M۳ Pro ، نیاز به انتخاب محیط شبیه‌سازی انعطاف‌پذیرتر و سازگارتر با معماری متن‌باز احساس شد. در این مرحله، تمرکز پروژه به سمت **Rviz** به عنوان قلب بصری‌سازی و شبیه‌سازی در اکوسیستم ROS۲ معطوف گردید.

Rviz ابزاری قدرتمند و گسترش‌پذیر است که نه تنها امکان نمایش زنده و تعاملی مدل ربات را فراهم می‌آورد، بلکه به دلیل یکپارچگی عمیق با ROS۲، بستری ایده‌آل برای توسعه، آزمایش و اشکال‌زدایی الگوریتم‌های کنترلی قبل از اجرا روی سخت‌افزار واقعی محسوب می‌شود. برخلاف نرم‌افزارهای شبیه‌سازی بسته و صنعت‌محور، Rviz با پذیرش مدل‌های URDF و پشتیبانی از انتشار و اشتراک‌گذاری داده‌ها از طریق Topicهای استاندارد ROS۲، امکان شبیه‌سازی ربات‌های با طراحی غیرمتعارف - مانند RoArm-M۳ - را بدون نیاز به بازطراحی سنگین سینماتیکی فراهم کرد.

در این بخش، فرآیند کامل پیاده‌سازی مدل ربات در Rviz، تنظیم ارتباط‌های لازم، طراحی و اجرای الگوریتم برنامه‌ریزی حرکت Pick-and-Place و در نهایت آماده‌سازی بستر Sim-to-Real به تفصیل شرح داده می‌شود. این روال، گام‌های عملی از بارگذاری مدل تا خروجی‌های کنترلی را پوشش می‌دهد و چالش‌های مهمی از قبیل مدیریتSingularity ، همگام‌سازی زمانی و ایجاد نودهای سفارشی ROS۲ را مورد بحث قرار می‌دهد.

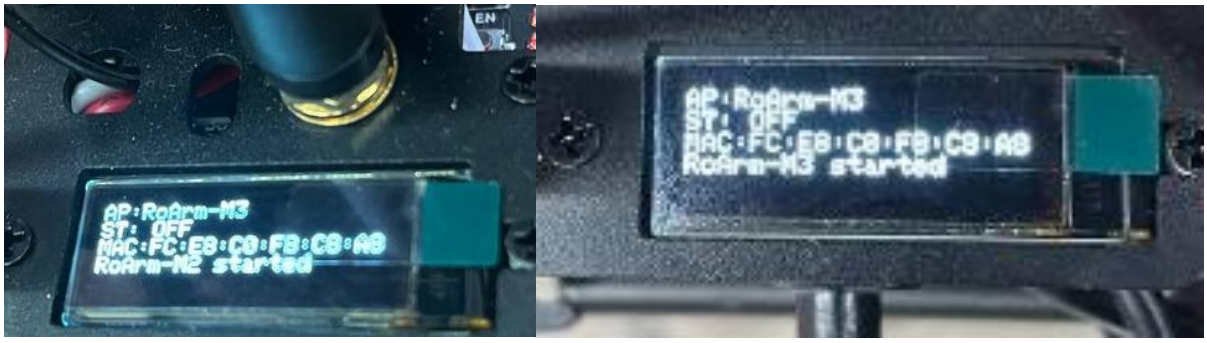
- **آماده سازی و شروع شبیه سازی در Rviz**

برای آماده سازی محیط شبیه سازی و مشاهده امکان اتصال ربات به شبیه ساز از ویدئوی ساخته شده توسط شرکت سازنده استفاده شد[۲]. در ویدئو توضیح میدهد که چگونه ماشین مجازی را ساخته و نرم افزار از پیش ساخته ربات را اجرا کنیم. با اجرای این مراحل میتوان ربات را به وسیله یک کنترل دستی کنترل کرد و زوایای مفاصل آن را تغییر داد.

شکل ۹ ستاپ اولیه ربات برای تست

- **مشکلات حین اتصال و بروز رسانی فریمور ربات**

با توجه به بروز نبودن ربات، در مرحله پیش به مشکل اتصال خوردیم و ربات قابل اتصال به سیستم نبود. با استفاده از نسخه فریمور اصلی ربات بر روی سایت[۱]، نرم افزار جدید با استفاده از کابل سریال بر روی ESP نصب گردید. بعد از بروزرسانی اتصال با موفقیت صورت گرفت و ربات کاملا با سیستم قابل کنترل بود.



شکل ۱۰ نرم افزار ربات قبل از بروزرسانی بر روی M2 تنظیم شده است که در خط آخر تصویر قابل مشاهده است. بعد از بروزرسانی به M3 که نسخه اصلی ربات است تغییر پیدا کرده است.

- **الگوریتم‌های تولید مسیر و برنامه‌ریزی حرکت**

برای برنامه ریزی مسیر یک الگوریتم pick and place در نظر گرفته شد که یک جسم را از نقطه ای برمیدارد و در نقطه ای دیگر قرار میدهد (برای مثال از روی یک نوار نقاله برمیدار و بر روی یک نوار نقاله دیگر قرار میدهد.).

سلسله حرکاتی که انجام میدهد به این ترتیب است:

۱. **ربات در موقعیت خانه (Home Position) قرار دارد.**

در ابتدای اجرای برنامه، ربات به‌طور کامل در موقعیت اســتارتی یا موقعیت اولیه خود قرار دارد. این موقعیت معمولاً یک نقطه ایمن و مشــخص در فضــای کاری ربات اســت که به‌عنوان مبدأ برای تمام حرکات بعدی استفاده می‌شود.

۲. از موقعیت خانه به سطح زمین حرکت کرده و انتهای ابزار (End Effector) باز میشود.

ربات از موقعیت خانگی به سمت نقطه‌ای بالای سطح زمین (مثلاً بالای یک نوار نقلیه یا سطحی که جسم قرار دارد) حرکت می‌کند. در این مرحله، انتهای ابزار (مانند گیره یا دســت) باز اســت تا بتولند جســم را به‌درستی بگیرد. این حرکت با دقت انجام می‌شود تا از برخورد ناخواسته با موانع جلوگیری شود.

۳. جسم گرفته می‌شود.

هنگامی که انتهای ابزار به مکان مورد نظر رسید، گیره یا دست ربات بسته می‌شود و جسم را به‌طور محکم گرفته می‌شود. این مرحله به‌صورت خودکار یا تحت کنترل برنامه انجام می‌شود و معمولاً با ارسال دستورات به موتورهای مربوطه از طریق سیستم کنترلی انجام می‌گیرد.

٤. به سمت بالا حرکت کرده و انتهای ابزار را مسیریابی می‌کند.

پس از گرفتن جسم، ربات به سمت بالا حرکت می‌کند تا جسم از سطح زمین جدا شود و از محدودیت‌های فیزیکی (مانند موانع یا سایر اشیاء) دور شود. در این مرحله، انتهای ابزار به‌صورت مسیریابی‌شده (مثلاً با یک مسیر منحنی یا خطی) به سمت نقطه مقصد حرکت می‌کند تا از برخورد ناخواسته جلوگیری شود.

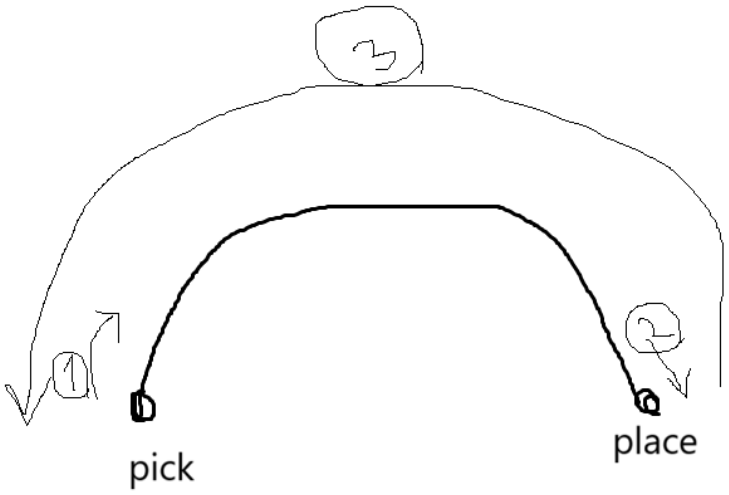
٥. به مکان دیگری می‌رود، به سطح زمین می‌آید، جسم را روی زمین قرار می‌دهد و انتهای ابزار را باز می‌کند.

ربات به نقطه‌ای که قرار اســت جســم در آن قرار گیرد (مثلاً یک نوار دیگر یا یک مکان مخصــوص) حرکت می‌کند. در این مرحله، ابتدا به سطح زمین می‌آید، سپس جسم را به‌طور دقیق روی سطح قرار می‌دهد و در نهایت گیره یا انتهای ابزار را باز می‌کند تا جسم آزاد شود. این عملیات با دقت بالا انجام می‌شود تا از افتادن یا از دست دادن جسم جلوگیری شود.

٦. به موقعیت خانه باز می‌گردد.

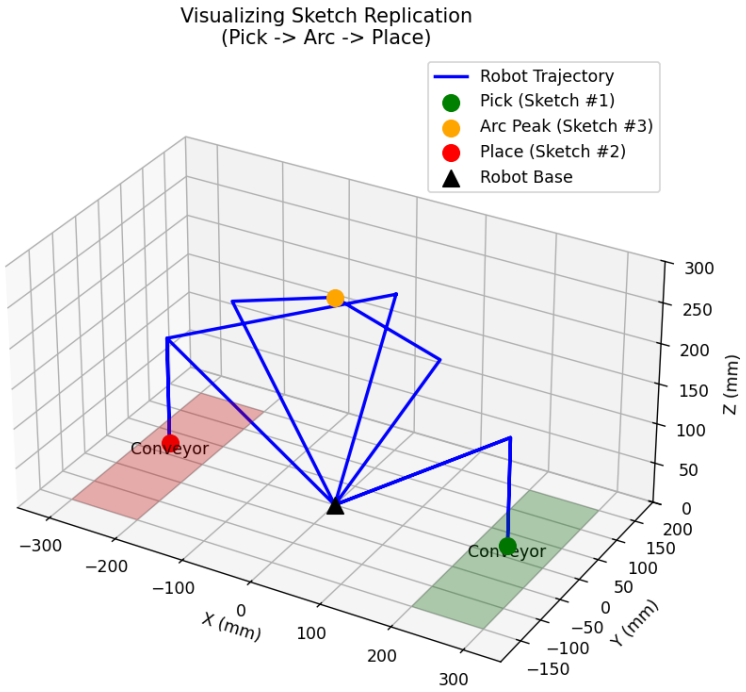
پس از اتمام عملیات گذاشــتن جســم، ربات به‌طور کامل به موقعیت اولیه یا موقعیت خانه خود بازمی‌گردد. این مرحله به عنوان بخشــی از چرخه کامل عملیات تکرارشــونده (Pick and Place) شــناخته می‌شــود و امکان شروع یک چرخه دیگر را فراهم می‌کند.

این سلسه حرکات در شکل ۱۱ به‌صورت شماتیک نمایش داده شده است و به‌عنوان یک سناریوی استاندارد در محیط شبیه‌سازی پیاده‌سازی شده است. این الگوریتم به‌صورت خودکار و با استفاده از کدهای پایتون و C++ در محیط ROS2 اجرا می‌شود و داده‌های زوایای مفاصل از شبیه‌ساز به ربات واقعی ارسال می‌گرد.



شکل۱۱ شماتیک مسیر مد نظر برای برداشتن و گذاشتن یک جسم بصورت متوالی و تکرار شونده

برای نمایش بهتر مسیر برنامه ریزی شده و بهبود عملکرد الگوریتم، یک برنامه توسعه داده شد که محتوای برنامه ریزی مسیر را به صورت json دریافت میکند و تصویر مسیر برنامه ریزی شده را بصورت خروجی نمایش میدهد. همانطور که در شکل ۱۲ قابل مشاهده است، مکان ربات، نقطه برداشت و نقطه گذاشتن کاملا مشخص شده است.



شکل ۱۲ خروجی کد پردازش json برنامه ریزی مسیر

- **پیاده سازی کنترل در محیط شبیه سازی Rviz**

Rviz یک ابزار گرافیکی قدرتمند در محیط ROS اســـت که به کاربران اجازه می‌دهد تا داده‌های ربات را به صورت زنده و بصری مشاهده کنند. این ابزار شبیه‌سازی ربات، موقعیت فضایی آن، مسیر حرکت، اطلاعات سـنسـورها (مثل لیزر یا دوربین)، و حتی داده‌های مربوط به مدل سـه‌بعدی ربات را به صـورت تصـویری و تعاملی نمایش می‌دهد. به عبارت ساده، Rviz مانند یک صفحه نمایش دیجیتال برای ربات عمل می‌کند که کاربران می‌توانند از طریق آن حرکات ربات را ببینند، مسیرهای برنامه‌ریزی شده را بررسی کنند و عملکرد سیستم را در محیط شبیه‌سازی ارزیابی کنند. این ابزار بسیار مفید است زیرا کار با داده‌های خام مثل زوایای مفاصل یا موقعیت ابزار را به شکل بصری و قابل فهم تبدیل می‌کند.

در محیط شـبیه‌سـازی Rviz، کنترل ربات به صـورت یک فرآیند چندمرحله‌ای انجام می‌شـود. ابتدا یک مدل سـه‌بعدی ربات (مثلاً RoArm-M۳ Pro) در Rviz بارگذاری می‌شـود. این مدل با اسـتفاده از فایل‌های URDF یا Xacro ساخته می‌شود و شامل تمام مفاصل، اتصالات و ابعاد فیزیکی ربات است. سپس یک نود (node) در ROS۲ ایجاد می‌شـود که دسـتورات کنترلی را دریافت کرده و به ربات در شـبیه‌سـازی اعمال می‌کند.

برای پیاده‌سازی کنترل، ابتدا یک کد (مثلاً به زبان C++ یا پایتون) نوشته می‌شود که مسیر حرکت ربات را برنامه‌ریزی می‌کند.

در این پروژه محیط ROS و Rviz از قبل توسـط سـازنده ربات ارائه شـده بود و چالش کار سـاخت یک نود جدید برای ROS2 مورد اسـتفاده بود که بتوان با آن ربات را به نحو احسـن کنترل کرد که در ادامه به این بخش می‌پردازیم.

- **دستورالعمل ساخت یک node جدید ROS2 برای کنترل ربات**

آماده سازی سیستم

برای پیاده سازی یک کنترل مجزا از کنترل های موجود برای ربات RoArm-M3 Pro، لازم است یک نود جدید در ROS2 تعریف شـود که همزمان با نود های دیگر فعال شـده و فرمان های کنترلی را ابتدا به ربات موجود در شبیه ساز و در نهایت به ربات واقعی ارسال نماید.

برای این هدف ابتدا باید فایل سـیسـتم عامل شـخصـی سـازی شـده برای ربات RoArm-M۳ Pro توسـط شرکت waveshare موجود در سایت مرجع[۱] دانلود شود و مطابق دستورالعمل[۲] بر روی یک ماشین مجازی نصب شده و وارد ماشین جدید می شویم.

ساخت node جدید برای ROS2

در ماشین مجازی ساخته شده، در ترمینال مراحل زیر را دنبال می کنیم:

1 . ساخت دایرکتوری های زیر:

أ. وارد دایرکتوری درست میشویم.

```
cd ~/roarm_ws/src/roarm_main
```

ب. پکیج ROS2 مناسب را می سازیم.

```
ros2 pkg create --build-type ament_cmake --dependencies rclcpp moveit_ros_planning_interface geometry_msgs moveit_msgs roarm_msgs tf۲_geometry_msgs -- roarm_p
```

ج. پوشه های مورد نیاز را درون پکیج میسازیم.

```
mkdir -p roarm_p/launch

mkdir -p roarm_p/src
```

د. فایل پایتون اجرای محیط شبیه سازی و اجرای کنترل را که در در مسیر زیر قرار میدهیم:

```
cd /home/ws/roarm_ws/src/roarm_main/roarm_p/launch/
```

خود فایل پایتون در پیوست ۲ قرار دارد.

ه. فایل CMakeLists.txt که برای تنظیمات node در ROS۲ استفاده میشود و در پیوست ۴ معرفی شده را در دایرکتوری زیر قرار میدهیم:

```
cd /home/ws/roarm_ws/src/roarm_main/roarm_p/
```

و. فایل کنترل اصلی که به زبان C++ نوشته شده و در پیوست ۳ توضیح داده شده است را در مکان زیر قرار میدهیم:

```
cd /home/ws/roarm_ws/src/roarm_main/roarm_p/src/
```

ز. در نهایت با استفاده از دستورات زیر پکیج ROS2 را ساخته و اجرا میکنیم.

```
cd /home/ws/roarm_ws

colcon build --packages-select roarm_p

source install/setup.bash

ros2 launch roarm_p auto_pick.launch.py
```

با اجرای این دستور محیط شبیه سازی باز شده و ربات تسک مورد نظر را انجام میدهد.

توسعه رابط نرم‌افزاری Sim-to-Real

معماری سیستم ارسال و دریافت داده

معماری ارتباطی این سیستم بر پایه ارتباط بین‌فرامینه (inter-process communication) در ROS2 مبتنی بر DDS (Data Distribution Service) است.

در ادامه Nodeهای اصلی لایه‌های ارتباطی در ROS2 معرفی میشوند.

- roarm_m3_driver (دریافت داده از سخت‌افزار ربات)
- joint_state_publisher (انتشار وضعیت جوینت‌ها)
- robot_state_publisher (محاسبه و انتشار ترانسفورم‌ها)
- rviz2 (نمایش داده‌ها)

Topicهای اصلی که نقش انتقال اطلاعات بین nodeها را دارند عبارتند از:

- joint_states: داده‌های وضعیت جوینت (زاویه، سرعت)
- tf: تبدیل‌های مختصات بین ارگان‌ها (base_link, link1, ...)
- /cmd_vel یا /joint_trajectory: دستورات کنترل (برای حرکت)
- /robot_description: مدل URDF ربات برای نمایش در Rviz

- **جریان داده از سخت‌افزار به Rviz**

کنترلر ربات (مثلاً با استفاده از MicroPython یا C++ روی ماژول ARM) از سنسورهای جوینت (Encoder یا IMU) داده می‌گیرد.

داده‌ها از طریق پروتکل Serial (UART) یا ROS2 Bridge (مثلاً با serial_bridge یا ros2_serial) به سیستم کنترل که در اینجا لپ تاپ است، ارسال می‌شود.

یک Node ROS2 (مثلاً roarm_m3_driver) این داده‌ها را دریافت کرده و آنها را به فرمت sensor_msgs/JointState تبدیل می‌کند.

این Node داده را در Topic /joint_states منتشر می‌کند.

robot_state_publisher از این داده و مدل URDF ربات، ترانسفورم‌های فضایی (TF) را محاسبه و منتشر می‌کند.

rviz2 از این داده‌ها (Topic /joint_states, /tf) استفاده کرده و مدل ربات را به‌صورت پویا و سه‌بعدی نمایش می‌دهد.

- **ارسال دستورات از Rviz به ربات**

کاربر در Rviz می‌تواند از طریق ابزارهایی مانند "Publish Point" یا "Interactive Markers" یا "Joint State Publisher"، دستورات حرکتی را ارسال کند.

این دستورات به فرمت trajectory_msgs/JointTrajectory یا std_msgs/Float64MultiArray ارسال می‌شوند.

یک Node کنترلر (مثلاً trajectory_controller) دستورات را دریافت و به سخت‌افزار ربات ارسال می‌کند (مثلاً از طریق UART یا CAN).

سخت‌افزار ربات این دستورات را پردازش کرده و جوینت‌ها را حرکت می‌دهد.

- **ویژگی‌های کلیدی معماری**

مبنای داده: استفاده از DDS (Cyclone DDS یا FastDDS) برای ارتباط سریع و قابل اطمینان.

پشتیبانی از Real-time: با استفاده از RT-Preemption و تنظیمات مناسب، زمان‌بندی دقیق برای کنترل جوینت فراهم می‌شود.

قابلیت شبیه‌سازی: مدل URDF و Gazebo یا Ignition می‌تواند برای تست بدون سخت‌افزار استفاده شود.

پشتیبانی از ROS2 Distroهای جدید: (مثلاً Humble, Iron) با استفاده از colcon و ament.

- **سناریوی همگام‌سازی و مدیریت تاخیر**

در سیستم‌های رباتیک زنده (real-time)، تاخیر (Latency) و عدم همگام‌سازی (Desynchronization) بین سنسورها، کنترلرها، و نمایشگرها (مانند Rviz) می‌تواند منجر به رفتارهای ناپایدار، نمایش نادرست وضعیت ربات، یا حتی خطا در کنترل شود. در مورد RoArm-M۳ Pro، که از پلتفرم‌های مبتنی بر ARM و ارتباطات سریال/UART استفاده می‌کند، مدیریت این چالش‌ها بسیار حیاتی است.

منابع تاخیر در این سیستم عبارتند از:

منبع تاخیر	توضیح
ارتباط سخت‌افزار(UART/SPI)	تأخیر در انتقال داده از جوینت‌ها به کنترلر) مثلاً ۵–۲۰ msبرای ارسال یک(state
پردازش درROS2 Node	زمان اجرای کد در (`roarm_m3_driver` مثلاً تبدیل داده، تولید (`JointState`
ارتباط بین فرآیندها(DDS)	تأخیر در انتقالTopic ها (مخصوصاً در شبکه‌های بی‌سیم یا بار بالا)
نمایش درRviz	تأخیر در دریافت و رندر کردن داده‌ها) مخصوصاً اگر نمایش چندین Topic با سرعت بالا باشد(

زمان‌بندی نامناسب(Non-Realtime)	عدم استفاده از سیستم عامل Real-Time مثلاً Raspbian بدون-RT Preemption)

برای همگام‌سازی و مدیریت تاخیر مراحل زیر به ترتیب صورت میگیرد تا سیستمی سریع و فاقد تاخیر داشته باشیم.

مرحله ۱: جمع‌آوری داده از سنسورها (سخت‌افزار)

سنسورهای جوینت (Encoder یا IMU) داده را به صورت دوره‌ای (مثلاً هر ۱۰ms) ارسال می‌کنند.

این داده‌ها از طریق UART به کنترلر (لپتاپ) می‌رسند.

تاخیر اولیه: ۵–۱۵ms (بسته به سرعت انتقال و بار سیستم).

مرحله ۲: پردازش در Node ROS۲ (roarm_m۳_driver)

Node مربوطه داده را دریافت و به فرمت sensor_msgs/JointState تبدیل می‌کند.

برای مدیریت تاخیر، از زمان‌بندی مبتنی بر rclcpp::Rate استفاده می‌شود (مثلاً ۱۰۰ms → ۱۰Hz).

تاخیر اضافی: ۲–۵ms (برای پردازش و تولید پیام).

مرحله ۳: انتشار داده در Topic /joint_states

داده از طریق DDS (FastDDS یا Cyclone DDS) منتشر می‌شود.

ROS۲ به‌طور پیش‌فرض زمان‌بندی زیر-میلی‌ثانیه‌ای (microsecond-level) را برای انتقال داده‌ها فراهم می‌کند.

تاخیر در انتقال: ۱–۳ms (در شبکه محلی، بدون تداخل).

مرحله ۴: دریافت توسط robot_state_publisher

این Node داده را دریافت کرده و از مدل URDF، ترانسفورم‌های فضایی (TF) را محاسبه می‌کند.

زمان پردازش: ۱–۲ms.

مرحله ۵: انتشار در Topic /tf و /robot_description

داده‌های TF به Rviz ارسال می‌شوند.

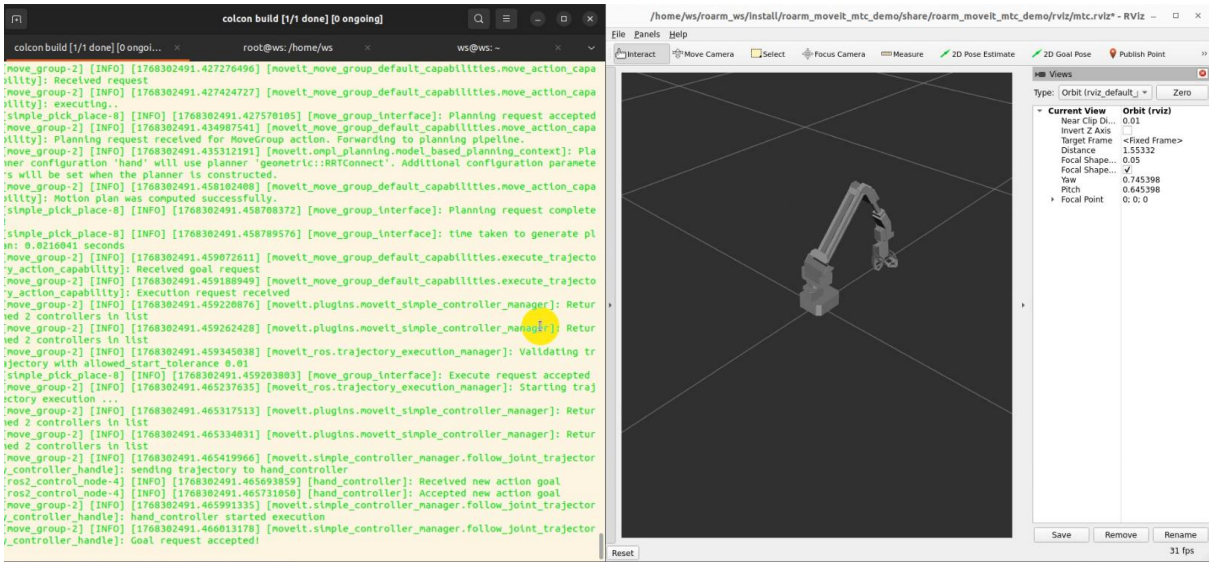
Rviz این داده‌ها را دریافت و مدل ربات را به‌صورت پویا و همگام با وضعیت واقعی نمایش می‌دهد.

ارزیابی عملکرد و تحلیل نتایج تجربی

شرح آزمایش‌های انجام شده در محیط شبیه‌سازی

در محیط شبیه سازی حرکت های مختلف از جمله pick and place تست شد. بارها درگیر مسئله singularity شدیم و برای حل مشکل، مکان بار و بازو جابجا شدند. برای مثال یک بار مسئله singularity هنگامی رخ میداد که مکان بار، خالی کردن بار و بیس ربات در یک خط بودند؛ در این صورت ربات تا نزدیک مکان گذاشتن بار می رفت ولی موفق نمی شد بار را به مقصد برساند و برمیگشت به نقطه ابتدایی تا از سمتی دیگر به مکان گذاشتن بار نزدیک شود. برای رفع این مشکل مکان بیس ربات را کمی جابجا کرده و به گونه ای تنظیم شد که این سه نقطه بر روی یک خط واقع نشوند.

در شکل ۱۳ ربات شبیه سازی شده همراه با لاگ های ثبت شده هنگام حرکت مشاهده میکنید. در این سناریو ربات یک جسم را از یک نقطه بر میدارد و در مکانی دیگر قرار میدهد.



شکل ۱۳ نمایشی از ربات در شبیه ساز هنگام خالی کردن بار و دستورات کنترلی به رنگ سبز در ترمینال نمایش داده میشود.

پیاده‌سازی سناریو روی ربات واقعی

پس از تکمیل مراحل شبیه‌سازی و اطمینان از عملکرد صحیح الگوریتم‌های کنترلی و برنامه‌ریزی مسیر در محیط Rviz، نوبت به آزمون نهایی و اجرای سناریوی Pick-and-Place بر روی ربات فیزیکی RoArm-M۳ Pro رسید. این مرحله، عیار واقعی پروژه و آزمونی برای اثبات کارایی چارچوب Sim-to-Real طراحی‌شده بود.

پیاده‌سازی با موفقیت کامل در محیط آزمایشگاه انجام پذیرفت. کل چرخه‌ی عملیاتی تعریف‌شده — شامل حرکت از موقعیت خانه، نزدیک شدن به جسم، درگیری گریپر، بلند کردن و انتقال جسم، رهاسازی آن در موقعیت هدف و بازگشت به نقطه شروع — به‌صورت روان و بدون وقفه توسط ربات اجرا شد. عملکرد هماهنگ تمامی مفاصل، دقت در موقعیت‌یابی انتهای ابزار و قابلیت اطمینان عملگر گریپر، همگی مورد تأیید قرار گرفت.

دستاورد کلیدی این مرحله، انتقال بی‌درنگ و یکپارچه‌ی دستورات از محیط شبیه‌سازی به سخت‌افزار بود. معماری ارتباطی مبتنی بر ROS۲ و نود سفارشی توسعه‌یافته، به گونه‌ای عمل کرد که ربات واقعی عیناً مسیر از پیش برنامه‌ریزی و آزمایش‌شده در Rviz را دنبال نمود. این موفقیت، صحت مدل‌سازی سینماتیکی، کدهای کنترل و نیز پایداری پل ارتباطی ایجادشده را به طور عملی اثبات کرد.

این فرآیند، اگرچه موفقیت‌آمیز بود، خالی از چالش نبود. مواردی مانند تأخیرهای جزئی در ارتباط سریال، نیاز به کالیبراسیون اولیه‌ی دقیق موقعیت مفاصل و توجه به محدودیت‌های گشتاوری ربات در دنیای واقعی، از جمله نکات تجربی ارزشمندی بودند که در حین اجرا شناسایی و مدیریت شدند.

اجرای موفقیت‌آمیز این سناریو بر روی ربات واقعی، نقطه‌ی عطف پروژه محسوب می‌شود. این موفقیت نه تنها صحت یافته‌های شبیه‌سازی را تأیید کرد، بلکه توانمندی ربات RoArm-M۳ Pro را به عنوان یک بستر آموزشی-تحقیقاتی کاربردی برای پیاده‌سازی الگوریتم‌های کنترلی پیشرفته به نمایش گذاشت. ویدئوی کامل این عملیات، به عنوان گواهی عینی بر این موفقیت، ضبط و به پروژه پیوست شده است.

نتیجه‌گیری و پیشنهادها

جمع‌بندی

در این پروژه، مدل‌سازی سینماتیکی، شبیه‌سازی و کنترل ربات بازویی آموزشی RoArm-M۳ Pro با موفقیت در محیط ROS۲ و ابزار Rviz پیاده‌سازی شد. پس از بررسی و کنار گذاشتن محیط RoboDK به دلیل عدم تطابق ساختار غیراستاندارد ربات با چارچوب‌های صنعتی این نرم‌افزار، تمرکز اصلی پروژه بر توسعه یک سیستم Sim-to-Real مبتنی بر ROS۲ قرار گرفت.

دستاوردهای پروژه

- توسعه و اجرای یک **نود جدید ROS۲** برای کنترل خودکار عملیاتPick-and-Place
- پیاده‌سازی الگوریتم برنامه‌ریزی مسیر کاربردی با مدیریت گریپر و جلوگیری از برخورد
- ایجاد ارتباط دوطرفه پایدار بین شبیه‌ساز (Rviz) و ربات واقعی

- شناسایی و مدیریت عملی چالش‌های رایج ربات‌های آموزشی سبک شامل **singularity**، backlashمحدود، ظرفیت بار پایین و تأخیرهای ارتباطی سریال
- دستیابی به تکرارپذیری قابل قبول حرکات بین شبیه‌سازی و ربات واقعی

این پروژه نشان داد که حتی با وجود محدودیت‌های سخت‌افزاری قابل‌توجه ربات‌های رومیزی ارزان‌قیمت، با استفاده از ابزارهای متن‌باز مدرن (ROS۲ + MoveIt + Rviz) می‌توان سطح قابل قبولی از کنترل هوشمند و انتقال دانش Sim-to-Real را به دست آورد.

محدودیت‌های پژوهش

عدم پیاده‌سازی کنترلر پیشرفته‌تر (مانند impedance control، force control یا vision-based servoing)

عدم جبران سیستماتیک backlash و انعطاف‌پذیری سازه در مدل دینامیکی

تست محدود روی سناریوهای متنوع (فقط Pick-and-Place ساده بدون موانع پویا)

نبود اندازه‌گیری کمی دقیق خطای موقعیت اندافکتور (فقط مقایسه بصری و زاویه‌ای)

پیشنهادها برای توسعه و تحقیقات آتی

۱. اضافه کردن بینایی ماشین

ادغام دوربین (RealSense / USB cam) + بسته‌های vision_opencv یا MoveIt Vision برای تشخیص محل دقیق اشیاء و جبران خطای موقعیت .

۲. بهبود برنامه‌ریزی حرکت

- استفاده کامل از MoveIt به جای مسیریابی دستی
- پیاده‌سازی Cartesian path planning با اجتناب ازsingularity
- اضافه کردن velocity/acceleration profiling نرم‌تر

۳. کنترل پیشرفته‌تر

- اجرای کنترلرهای مبتنی بر مدل دینامیکی Computed Torque Control یا Inverse Dynamics
- تست الگوریتم‌های adaptive control برای جبران تغییر بار و انعطاف‌پذیری سازه

٤. **کاهش تأخیر ارتباطی**

- اســـتفاده از ESP۳۲ با firmware بهینه‌تر یا مهاجرت به بردهای قوی‌تر / Jetson Nano Raspberry Pi ۵

٥. **کاربردهای جدید**

- سناریوهای چند جسمی متوالی
- همکاری انسان-ربات با تشخیص برخورد ایمن
- یادگیری تقویتی برای بهینه‌سازی مسیر در فضای کاری شلوغ

ضمائم و پیوست‌ها

۱. کد پایتون و فایل json نمایش مسیر برنامه‌ریزی‌شده:

"..\code\visualize_path.py"

"..\code\path_plan.json”

۲. کد پایتون اجرای برنامه و شبیه ساز:

"..\code\auto_pick.launch.py"

۳. کد برنامه C++ برنامه ریزی مسیر:

"..\code\pick_and_place_path_planning.cpp"

۴. کد تنظیمات node برای ROS2:

"..\code\CMakeLists.txt"

توضیح روش تدوین

در تدوین و نگارش این گزارش، از ابزارهای هوش مصنوعی از جمله مدل‌های زبانی پیشرفته برای تولید متن، بازنویسی بخش‌هایی از توضیحات تخصصی و بهینه‌سازی نگارشی بهره گرفته شده است. استفاده از این ابزارها با هدف افزایش دقت علمی، روانی بیان، و انسجام ساختاری گزارش انجام شده است و تمامی محتوای تولیدشده به‌دقت بازبینی و توسط نگارنده تأیید شده است.

منابع و مراجع

[۱] "RoArm-M۳ - Waveshare Wiki." https://www.waveshare.com/wiki/RoArm-M۳ (accessed Feb. ۰۶, ۲۰۲۶).

[۲] "How to Control Waveshare RoArm Easily【Tutorial】 - YouTube." https://www.youtube.com/watch?v=aGMcTiRKnYQ (accessed Feb. ۰۶, ۲۰۲۶).