

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

## Ρομποτική 1

### Εργαστηριακή Άσκηση 4

Cobot

**Ομάδα A3-4**

Νικόλαος Αδαμόπουλος	03122074
Λεωνίδας Λαγομιτζής	03121061
Ηλίας-Στυλιανός Ηλιάδης	03122140
Ηλίας Πουλόπουλος	03122160
Ανδρέας Αγγελόπουλος	03122049

## Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>2</b>
1.1	Setup της εργαστηριακής άσκησης . . . . .	2
1.2	Βασικές διαφορές μεταξύ Robot και Cobot . . . . .	3
1.3	Χαρακτηριστικά του MyCobot 280 Jetson Nano . . . . .	3
1.4	Πεδία εφαρμογών . . . . .	3
1.5	Συνοπτική επιστημονική περιγραφή . . . . .	3
<b>2</b>	<b>1ο Μέρος - Pick and Place</b>	<b>4</b>
2.1	Υλοποίηση Διαδικασίας Pick-and-Place με Καρτεσιανές Συντεταγμένες . . . . .	4
2.1.1	Διαδικασία Εκτέλεσης . . . . .	4
2.1.2	Παρατηρήσεις & Μέτρα Ασφαλείας . . . . .	4
<b>3</b>	<b>2ο Μέρος - Ευθεία Κινηματική</b>	<b>6</b>
<b>4</b>	<b>3ο Μέρος - Kinesthetic Teaching</b>	<b>14</b>
4.1	Στόχος και Περιγραφή Διαδικασίας . . . . .	14
4.2	Βήματα Εκτέλεσης . . . . .	14
4.3	Σύγκριση με το 1ο Μέρος (Pick-and-Place με Συντεταγμένες) . . . . .	14
4.4	Πλεονεκτήματα της μεθόδου Kinesthetic Teaching . . . . .	15
4.5	Σχολιασμός Κώδικα και Μηχανισμός Επαλήθευσης Θέσης . . . . .	15

## 1 Εισαγωγή

### 1.1 Setup της εργαστηριακής άσκησης

Το setup που χρησιμοποιήθηκε στην άσκηση αποτελείται από τα εξής υποσυστήματα.

#### Ο ρομποτικός βραχίονας – MyCobot 280 Jetson Nano

Πρόκειται για τον συνεργατικό βραχίονα myCobot 280 Jetson Nano, έναν μικρορομποτικό βραχίονα 6 βαθμών ελευθερίας με ενσωματωμένο NVIDIA Jetson Nano και μικροελεγκτή ATOM για διπλή επεξεργασία. Σύμφωνα με το συνοδευτικό PDF (σελ. 3) το ρομπότ ζυγίζει περίπου 1030 g, σηκώνει μέχρι 250 g, έχει ακτίνα χεριού 280 mm και επαναληψιμότητα της τάξης των  $\pm 5$  mm. Είναι τοποθετημένο πάνω σε σταθερή βάση στο εργαστήριο.

#### Υπολογιστής χειρισμού

Το setup περιλαμβάνει έναν desktop PC συνδεδεμένο μέσω USB με το ρομπότ και εκτελεί ειδικό πρόγραμμα σε Python για την άσκηση. Ο υπολογιστής στέλνει εντολές σειριακά προς το myCobot, διαβάζει αρχεία .txt με θέσεις και εντολές gripper και εκτελεί τα scripts του εργαστηρίου (move, pick-place, kinesthetic teaching).

#### Λογισμικό και βιβλιοθήκες Python

Στον κώδικα χρησιμοποιείται η βιβλιοθήκη pymycobot για την επικοινωνία με το ρομπότ· οι βασικές λειτουργίες υλοποιούνται μέσω συναρτήσεων όπως `send_angles/sync_send_angles`, `send_coords` και `set_gripper_value`.

#### Χώρος εργασίας – αντικείμενα

Στο πρώτο μέρος (pick & place) χρησιμοποιήσαμε έναν μικρό πλαστικό κύβο και καθορισμένες θέσεις λήψης και εναπόθεσης στην επιφάνεια εργασίας, με ιδιαίτερη προσοχή στους περιορισμούς του workspace ώστε να αποφευχθούν συγκρούσεις.

#### Διεπαφή Kinesthetic Teaching (Μέρος 3)

Το GUI είναι υλοποιημένο σε tkinter και επιτρέπει στον χρήστη να καθοδηγεί χειροκίνητα το ρομπότ, να αποθηκεύει σημεία αναφοράς, να ελέγχει τον gripper και να εκτελεί αυτόνομα την αποθηκευμένη διαδικασία.

## 1.2 Βασικές διαφορές μεταξύ Robot και Cobot

Τα κλασικά βιομηχανικά ρομπότ χαρακτηρίζονται από υψηλές ταχύτητες και ακρίβεια, απαιτούν ειδικούς χώρους ασφαλείας και εξειδικευμένο προγραμματισμό, και σχεδιάζονται για βαριά, επαναλαμβανόμενα καθήκοντα. Αντίθετα, τα συνεργατικά ρομπότ διαθέτουν αισθητήρες ροπής και μηχανισμούς ασφάλειας που επιτρέπουν άμεση συνεργασία με τον άνθρωπο· σταματούν σε περίπτωση αντίστασης, έχουν μικρότερη ισχύ αλλά μεγαλύτερη προσαρμοστικότητα και διευκολύνουν τον προγραμματισμό (π.χ. drag & teach / kinesthetic teaching). Ο myCobot 280 ανήκει στη δεύτερη κατηγορία και προορίζεται για καθοδήγηση με το χέρι και χρήση σε εκπαιδευτικό περιβάλλον.

## 1.3 Χαρακτηριστικά του MyCobot 280 Jetson Nano

Μηχανικά, ο myCobot έχει 6 βαθμούς ελευθερίας, ακτίνα εργασίας 280 mm, επανάληψη περίπου  $\pm 5$  mm, μέγιστο φορτίο 250 g και βάρος περίπου 1030 g· τα μήκη συνδέσμων και ο πλήρης πίνακας Denavit–Hartenberg περιλαμβάνονται στο διάγραμμα (PDF, σελ. 4–5). Η αρχιτεκτονική περιλαμβάνει onboard Jetson Nano για επεξεργασία εικόνας/AI και μικροελεγκτή ATOM για real-time low-level control. Λογισμικά υποστηριζόμενα περιλαμβάνουν Python API (pymycobot), ROS και συμβατότητα με Linux/Windows, ενώ υπάρχει δυνατότητα χρήσης διάφορων end-effectors όπως gripper, suction cup ή camera module.

## 1.4 Πεδία εφαρμογών

Ο myCobot 280 προορίζεται κυρίως για εκπαιδευτικά εργαστήρια και πανεπιστημιακά μαθήματα ρομποτικής, όπου χρησιμοποιείται για διδασκαλία κινηματικής και πειράματα με vision/AI. Επιπλέον, μπορεί να εξυπηρετήσει ελαφριές βιομηχανικές εργασίες χαμηλού βάρους, απλές αυτοματοποιήσεις και ερευνητικές εφαρμογές σε HRI, έλεγχο αρθρώσεων και collaborative robotics.

## 1.5 Συνοπτική επιστημονική περιγραφή

Η άσκηση στηρίζεται στη χρήση του συνεργατικού myCobot 280, συνδεδεμένου με υπολογιστή που εκτελεί Python κώδικα. Ο χρήστης δίνει εντολές είτε μέσω αρχείων .txt είτε μέσω GUI, και το ρομπότ εκτελεί καρτεσιανές κινήσεις, κινήσεις άρθρωσης και εντολές gripper· η χαμηλή αδράνεια και οι αισθητήρες θέσης καθιστούν το σύστημα ασφαλές για άμεση ανθρώπινη αλληλεπίδραση.

## 2 1ο Μέρος - Pick and Place

### 2.1 Υλοποίηση Διαδικασίας Pick-and-Place με Καρτεσιανές Συντεταγμένες

Στην παρούσα ενότητα υλοποιήθηκε η διαδικασία λήψης-και-εναπόθεσης (pick-and-place) κύβων, προγραμματίζοντας τις συντεταγμένες των ενδιαμέσων θέσεων (waypoints) του ρομποτικού βραχίονα.

Για τον έλεγχο της κίνησης και της λαβής χρησιμοποιήθηκαν οι ακόλουθες εντολές:

- `set_coords ( $X, Y, Z$ )`: Ορισμός ενδιαμέσων θέσεων κίνησης, όπου  $X, Y, Z$  οι καρτεσιανές συντεταγμένες σε χιλιοστά (mm).
- `set_gripper_state ( $S$ )`: Καθορισμός κατάστασης της αρπάγης/λαβής. Για  $S = 0$  η αρπάγη ανοίγει, ενώ για  $S = 1$  η αρπάγη κλείνει.
- `set_coords_pick_pos ( $X, Y, Z$ )`: Ορισμός της θέσης λήψης (pick position) σε καρτεσιανές συντεταγμένες (mm).
- `set_coords_place_pos ( $X, Y, Z$ )`: Ορισμός της θέσης εναπόθεσης (place position) σε καρτεσιανές συντεταγμένες (mm).

#### 2.1.1 Διαδικασία Εκτέλεσης

Η ακολουθία των κινήσεων που σχεδιάστηκε και εκτελέστηκε περιλαμβάνει τα εξής βήματα:

1. **Σημείο Προσέγγισης (Approach Point)**: Μετακίνηση σε θέση ασφαλείας, κατακόρυφα πάνω από τον κύβο, με ανοιχτή λαβή.
2. **Θέση Λήψης**: Κάθοδος στη θέση του κύβου και εντολή κλεισίματος της λαβής ( $S = 1$ ).
3. **Ανύψωση**: Επιστροφή στο σημείο προσέγγισης (Θέση 1) για ασφαλή μεταφορά.
4. **Σημείο Εναπόθεσης (Approach Place Point)**: Μετακίνηση σε θέση ασφαλείας, κατακόρυφα πάνω από τον στόχο εναπόθεσης.
5. **Θέση Εναπόθεσης**: Κάθοδος στο επιθυμητό σημείο και εντολή ανοίγματος της λαβής ( $S = 0$ ).
6. **Απομάκρυνση**: Επιστροφή στο σημείο εναπόθεσης (Θέση 4) πριν την επαναφορά στην αρχική θέση.

Η παραπάνω διαδικασία φαίνεται στο επισυναπτόμενο βίντεο `part_1_pnp.mp4`.

#### 2.1.2 Παρατηρήσεις & Μέτρα Ασφαλείας

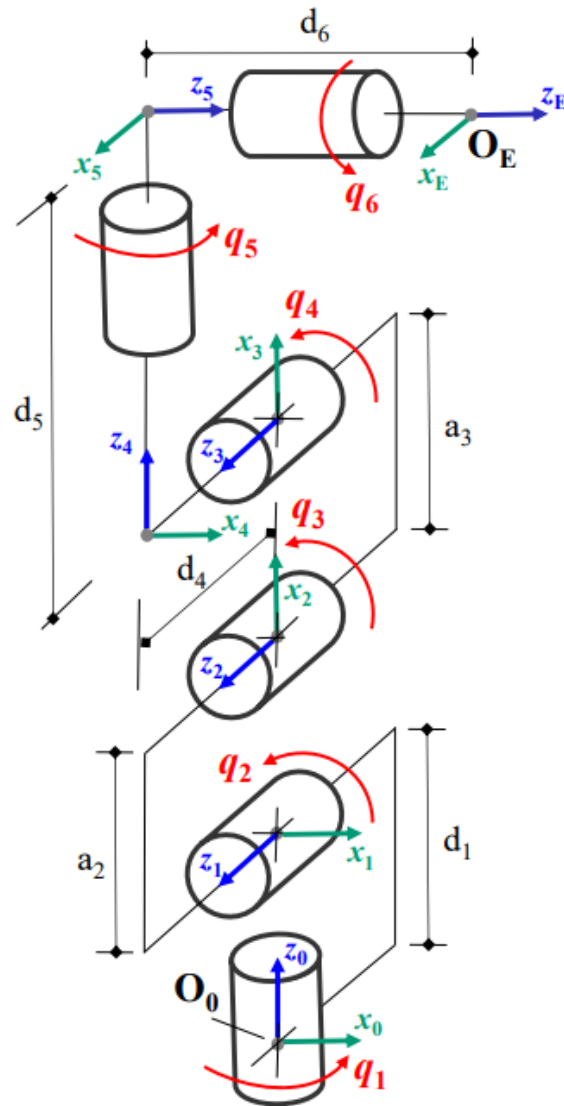
Κρίσιμη παράμετρος για την επιτυχία του πειράματος ήταν η χρήση των ενδιαμέσων σημείων προσέγγισης και απομάκρυνσης (Βήματα 1 και 4/6).

- Η παρεμβολή της **Θέσης 1** μεταξύ της αρχικής θέσης και της θέσης λήψης εξασφαλίζει κάθετη προσέγγιση. Χωρίς αυτή, το ρομπότ θα εκτελούσε διαγώνια κίνηση, αυξάνοντας τον κίνδυνο ανατροπής του κύβου πριν τη λήψη.
- Αντίστοιχα, η χρήση της **Θέσης 6** μετά την εναπόθεση είναι απαραίτητη ώστε ο βραχίονας να απομακρυνθεί κατακόρυφα. Σε αντίθετη περίπτωση, μια διαγώνια κίνηση επιστροφής θα μπορούσε να χτυπήσει και να ρίξει τον τοποθετημένο κύβο.

Τέλος, κατά τον προγραμματισμό των συντεταγμένων  $(X, Y, Z)$  δόθηκε ιδιαίτερη προσοχή στους περιορισμούς του χώρου εργασίας (workspace constraints), ώστε να αποφευχθούν συγκρούσεις (collisions) του ρομπότ με τον εαυτό του (self-collision), τη βάση στήριξης ή την επιφάνεια εργασίας.

### 3 2ο Μέρος - Ευθεία Κινηματική

Το κινηματικό διάγραμμα με τις κινηματικές παραμέτρους του βραχίονα καθώς και τα ενδιάμεσα πλαίσια αναφοράς (coordinate frames) τοποθετημένα με βάση τη μέθοδο Denavit–Hartenberg (D–H) παρατίθεται ακολούθως.



Σχήμα 1: Κινηματικό διάγραμμα του myCobot 280 Jetson Nano όπου τα ενδιάμεσα πλαίσια αναφοράς είναι τοποθετημένα σύμφωνα με τη σύμβαση Denavit–Hartenberg (D–H)

$i$	$\theta_i$ (rad)	$d_i$ (mm)	$a_i$ (mm)	$\alpha_i$
1	$q_1$	131.22	0	$\frac{\pi}{2}$
2	$q_2 + \frac{\pi}{2}$	0	110.4	0
3	$q_3$	0	96	0
4	$q_4 - \frac{\pi}{2}$	63.4	0	$-\frac{\pi}{2}$
5	$q_5 - \frac{\pi}{2}$	75.05	0	$-\frac{\pi}{2}$
6	$q_6$	45.6	0	0

Πίνακας 1: Πίνακας παραμέτρων Denavit–Hartenberg

Σύμφωνα με τη μέτρα Denavit–Hartenberg:

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

Συνεπώς:

$$A_1^0 = \begin{bmatrix} \cos q_1 & 0 & \sin q_1 & 0 \\ \sin q_1 & 0 & -\cos q_1 & 0 \\ 0 & 1 & 0 & 131.22 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^1 = \begin{bmatrix} -\sin q_2 & -\cos q_2 & 0 & -110.4 \sin q_2 \\ \cos q_2 & -\sin q_2 & 0 & 110.4 \cos q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} \cos q_3 & -\sin q_3 & 0 & 96 \cos q_3 \\ \sin q_3 & \cos q_3 & 0 & 96 \sin q_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$A_4^3 = \begin{bmatrix} \sin q_4 & 0 & \cos q_4 & 0 \\ -\cos q_4 & 0 & \sin q_4 & 0 \\ 0 & -1 & 0 & 63.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5^4 = \begin{bmatrix} \sin q_5 & 0 & \cos q_5 & 0 \\ -\cos q_5 & 0 & \sin q_5 & 0 \\ 0 & -1 & 0 & 75.05 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6^5 = \begin{bmatrix} \cos q_6 & -\sin q_6 & 0 & 0 \\ \sin q_6 & \cos q_6 & 0 & 0 \\ 0 & 0 & 1 & 45.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Οπότε:

$$A_2^0 = A_1^0 A_2^1 = \begin{bmatrix} \cos q_1 & 0 & \sin q_1 & 0 \\ \sin q_1 & 0 & -\cos q_1 & 0 \\ 0 & 1 & 0 & 131.22 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\sin q_2 & -\cos q_2 & 0 & -110.4 \sin q_2 \\ \cos q_2 & -\sin q_2 & 0 & 110.4 \cos q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} -c_1 s_2 & -c_1 c_2 & s_1 & -110.4 c_1 s_2 \\ -s_1 s_2 & -s_1 c_2 & -c_1 & -110.4 s_1 s_2 \\ c_2 & -s_2 & 0 & 131.22 + 110.4 c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^0 = A_2^0 A_3^2 = \begin{bmatrix} -c_1 s_2 & -c_1 c_2 & s_1 & -110.4 c_1 s_2 \\ -s_1 s_2 & -s_1 c_2 & -c_1 & -110.4 s_1 s_2 \\ c_2 & -s_2 & 0 & 131.22 + 110.4 c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & -s_3 & 0 & 96 c_3 \\ s_3 & c_3 & 0 & 96 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} -c_1 s_{23} & -c_1 c_{23} & s_1 & -110.4 c_1 s_2 - 96 c_1 s_{23} \\ -s_1 s_{23} & -s_1 c_{23} & -c_1 & -110.4 s_1 s_2 - 96 s_1 s_{23} \\ c_{23} & -s_{23} & 0 & 131.22 + 110.4 c_2 + 96 c_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4^0 = A_3^0 A_4^3 = \begin{bmatrix} -c_1 s_{23} & -c_1 c_{23} & s_1 & -110.4 c_1 s_2 - 96 c_1 s_{23} \\ -s_1 s_{23} & -s_1 c_{23} & -c_1 & -110.4 s_1 s_2 - 96 s_1 s_{23} \\ c_{23} & -s_{23} & 0 & 131.22 + 110.4 c_2 + 96 c_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_4 & 0 & c_4 & 0 \\ -c_4 & 0 & s_4 & 0 \\ 0 & -1 & 0 & 63.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} c_1 c_{234} & -s_1 & -c_1 s_{234} & -110.4 c_1 s_2 - 96 c_1 s_{23} + 63.4 s_1 \\ s_1 c_{234} & c_1 & -s_1 s_{234} & -110.4 s_1 s_2 - 96 s_1 s_{23} - 63.4 c_1 \\ s_{234} & 0 & c_{234} & 131.22 + 110.4 c_2 + 96 c_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5^0 = A_4^0 A_5^4$$

$$= \begin{bmatrix} c_1 c_{234} & -s_1 & -c_1 s_{234} & -110.4 c_1 s_2 - 96 c_1 s_{23} + 63.4 s_1 \\ s_1 c_{234} & c_1 & -s_1 s_{234} & -110.4 s_1 s_2 - 96 s_1 s_{23} - 63.4 c_1 \\ s_{234} & 0 & c_{234} & 131.22 + 110.4 c_2 + 96 c_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_5 & 0 & c_5 & 0 \\ -c_5 & 0 & s_5 & 0 \\ 0 & -1 & 0 & 75.05 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_5 s_1 + c_{234} c_1 s_5 & c_1 s_{234} & c_{234} c_1 c_5 - s_1 s_5 & 63.4 s_1 - 75.05 c_1 s_{234} - 96 c_1 s_{23} - 110.4 c_1 s_2 \\ c_{234} s_1 s_5 - c_1 c_5 & s_1 s_{234} & c_1 s_5 + c_{234} c_5 s_1 & -63.4 c_1 - 75.05 s_1 s_{234} - 96 s_1 s_{23} - 110.4 s_1 s_2 \\ s_{234} s_5 & -c_{234} & s_{234} c_5 & 131.22 + 96 c_{23} + 110.4 c_2 + 75.05 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6^0 = A_5^0 A_6^5$$

$$= \begin{bmatrix} c_5 s_1 + c_{234} c_1 s_5 & c_1 s_{234} & c_{234} c_1 c_5 - s_1 s_5 & 63.4 s_1 - 75.05 c_1 s_{234} - 96 c_1 s_{23} - 110.4 c_1 s_2 \\ c_{234} s_1 s_5 - c_1 c_5 & s_1 s_{234} & c_1 s_5 + c_{234} c_5 s_1 & -63.4 c_1 - 75.05 s_1 s_{234} - 96 s_1 s_{23} - 110.4 s_1 s_2 \\ s_{234} s_5 & -c_{234} & s_{234} c_5 & 131.22 + 96 c_{23} + 110.4 c_2 + 75.05 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 45.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} R & p_E \\ 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} c_6(c_5 s_1 + c_{234} c_1 s_5) + s_{234} c_1 s_6 & s_{234} c_1 c_6 - s_6(c_5 s_1 + c_{234} c_1 s_5) & c_{234} c_1 c_5 - s_1 s_5 \\ s_{234} s_1 s_6 - c_6(c_1 c_5 - c_{234} s_1 s_5) & s_6(c_1 c_5 - c_{234} s_1 s_5) + s_{234} c_6 s_1 & c_1 s_5 + c_{234} c_5 s_1 \\ s_{234} c_6 s_5 - c_{234} s_6 & -c_{234} c_6 - s_{234} s_5 s_6 & s_{234} c_5 \end{bmatrix}$$

$$p_E = \begin{bmatrix} s_1(63.4 - 45.6 s_5) - c_1(75.05 s_{234} + 96 s_{23} + 110.4 s_2 - 45.6 c_{234} c_5) \\ c_1(45.6 s_5 - 63.4) - s_1(75.05 s_{234} + 96 s_{23} + 110.4 s_2 - 45.6 c_{234} c_5) \\ 75.05 c_{234} + 96 c_{23} + 110.4 c_2 + 45.6 s_{234} c_5 + 131.2 \end{bmatrix}$$

Στην παρακάτω εικόνα απεικονίζονται οι γωνίες των αρθρώσεων που έδωσε ο χρήστης και αυτές που υπολογίστηκαν από τους encoders. Επιπλέον, για κάθε συνδυασμό γωνιών δίνονται οι τελικές θέσεις που υπολογίστηκαν στον κώδικα της Python.

```

elephant@elephant-desktop:~/robotics1$ python3 second_task.py
User input --> J1:-30.0, J2:30.0, J3:-30.0, J4:0.0, J5:0.0, J6:45.0
DH with user input --> X:-40.01488906090521, Y:-50.106413272417946, Z:397.87860143493873
Joints angles from encoders --> J1:-29.35, J2:29.88, J3:-29.79, J4:-1.31, J5:-0.7, J6:44.29
DH with encoder input --> X:-38.29290333854706, Y:-51.843486531059966, Z:397.00605543805335
Cartesian coordinates from encoders --> X:-38.2, Y:-51.8, Z:396.9

User input --> J1:40.0, J2:-90.0, J3:90.0, J4:0.0, J5:0.0, J6:45.0
DH with user input --> X:160.2557317822926, Y:51.70702431144788, Z:302.2700051411918
Joints angles from encoders --> J1:39.63, J2:-90.52, J3:89.29, J4:-1.31, J5:-0.08, J6:44.29
DH with encoder input --> X:164.73758623779744, Y:54.02632094737639, Z:299.15136699999715
Cartesian coordinates from encoders --> X:164.7, Y:54.0, Z:299.1

User input --> J1:120.0, J2:40.0, J3:-10.0, J4:0.0, J5:0.0, J6:45.0
DH with user input --> X:113.40599268906186, Y:-69.62328053173522, Z:386.72413691787216
Joints angles from encoders --> J1:119.35, J2:40.51, J3:-9.14, J4:0.08, J5:0.79, J6:44.29
DH with encoder input --> X:114.48700423889696, Y:-75.52803724336692, Z:384.9371109879319
Cartesian coordinates from encoders --> X:114.5, Y:-75.5, Z:384.9

```

Σχήμα 2: Γωνίες χρήστη, γωνίες encoders και οι αντίστοιχες καρτεσιανές συντεταγμένες του τελικού εργαλείου δράσης

Αρκεί να υπολογίσουμε τις συντεταγμένες του end-effector χρησιμοποιώντας ως γωνίες των αρθρώσεων αυτές των encoders.

- $q_1 = -29.35^\circ$ ,  $q_2 = 29.88^\circ$ ,  $q_3 = -29.79^\circ$ ,  $q_4 = -1.31^\circ$ ,  $q_5 = -0.7^\circ$ ,  $q_6 = 44.29^\circ$

$$p_E = \begin{bmatrix} -38.291859143472900 \\ -51.843077768995656 \\ 397.0066583177501 \end{bmatrix}$$

- $q_1 = 39.63^\circ$ ,  $q_2 = -90.52^\circ$ ,  $q_3 = 89.29^\circ$ ,  $q_4 = -1.31^\circ$ ,  $q_5 = -0.08^\circ$ ,  $q_6 = 44.29^\circ$

$$p_E = \begin{bmatrix} 164.7375363728268 \\ 54.026940109758270 \\ 299.1513541312829 \end{bmatrix}$$

- $q_1 = 119.35^\circ$ ,  $q_2 = 40.51^\circ$ ,  $q_3 = -9.14^\circ$ ,  $q_4 = 0.08^\circ$ ,  $q_5 = 0.79^\circ$ ,  $q_6 = 44.29^\circ$

$$p_E = \begin{bmatrix} 114.4860319110242 \\ 75.527988772924320 \\ 384.9379322035499 \end{bmatrix}$$

Παρατηρούμε ότι τα παραπάνω αποτελέσματα διαφέρουν σημαντικά από τα αντίστοιχα που υπολόγισε ο κώδικας Python (DH with encoder input). Μάλιστα, σε μερικές περιπτώσεις διαφέρουν από το τρίτο δεκαδικό ψηφίο, απόκλιση που δεν μπορεί να θεωρηθεί αμελητέα.

Η συμπεριφορά αυτή οφείλεται στο ότι στον κώδικα Python χρησιμοποιήθηκε η προσέγγιση  $\frac{\pi}{2} = 1.5708$  για τον υπολογισμό των παραμέτρων DH, όπως φαίνεται στην ακόλουθη εικόνα.

```

1 import time
2 import numpy as np
3 from pymycobot.mycobot import MyCobot
4
5 # Mycobot280 robot DH parameters
6 # (theta, d, a, alpha offset # mm and radians)
7 # pi/2 = 1.5708
8 mycobot_dh_parameters = [
9     [0.0, 131.22, 0.0, 1.5708, 0.0],
10    [0.0, 0.0, 110.4, 0.0, 1.5708],
11    [0.0, 0.0, 96.0, 0.0, 0.0],
12    [0.0, 63.4, 0.0, -1.5708, -1.5708],
13    [0.0, 75.05, 0.0, -1.5708, -1.5708],
14    [0.0, 45.6, 0.0, 0.0, 0.0]
15 ]
16
17 # Define transformation (theta, d, a, alpha)
18 def transform(theta, d, a, alpha):
19     return np.array([
20         [np.cos(theta), -np.sin(theta)*np.cos(alpha), np.sin(theta)*np.sin(alpha), a*np.cos(theta)],
21         [np.sin(theta), np.cos(theta)*np.cos(alpha), -np.cos(theta)*np.sin(alpha), a*np.sin(theta)],
22         [0, np.sin(alpha), np.cos(alpha), d],
23         [0, 0, 0, 1]])
24
25 # Define forward kinematics (dh_params, joint_angles)
26 def forward_kinematics(dh_params, joint_angles):
27     # Function to compute overall transformation from the base to the end-effector
28     T = np.eye(4)
29     for i in range(len(joint_angles)):
30         theta, d, a, alpha, offset = dh_params[i] # Unpack DH parameters
31         theta += joint_angles[i]
32         T = T @ transform(theta + offset, d, a, alpha) # Compute transformation
33     return T # Return the end-effector position (x, y, z)

```

Σχήμα 3: Προσέγγιση  $\frac{\pi}{2} = 1.5708$  στον κώδικα Python

Προκειμένου να δείξουμε ότι πράγματι η απόκλιση που παρατηρείται οφείλεται αποκλειστικά σε αυτή την προσέγγιση, δημιουργήσαμε το ακόλουθο πρόγραμμα Matlab.

```

1 % =====
2 % FORWARD KINEMATICS WITH  $\pi/2 = 1.5708$ 
3 % =====
4 clear all
5 clc
6 format long g
7 pi_2 = 1.5708; % Approximation for  $\pi/2$ 
8 % DH parameters table [ $\theta$ _offset, d, a,  $\alpha$ ]
9 dh_params = [
10     [0, 131.22, 0, pi_2];
11     [pi_2, 0, 110.4, 0];
12     [0, 0, 96, 0];
13     [-pi_2, 63.4, 0, -pi_2];
14     [-pi_2, 75.05, 0, -pi_2];
15     [0, 45.6, 0, 0]
16 ];
17 syms q1 q2 q3 q4 q5 q6 real
18 dh_T = @(theta, d, a, alpha) [
19     cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha), a*cos(theta);
20     sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta);
21     0, sin(alpha), cos(alpha), d;
22     0, 0, 0, 1
23 ];
24 A = cell(6,1);
25 A{1} = dh_T(q1 + dh_params(1,1), dh_params(1,2), dh_params(1,3), dh_params(1,4));
26 A{2} = dh_T(q2 + dh_params(2,1), dh_params(2,2), dh_params(2,3), dh_params(2,4));
27 A{3} = dh_T(q3 + dh_params(3,1), dh_params(3,2), dh_params(3,3), dh_params(3,4));
28 A{4} = dh_T(q4 + dh_params(4,1), dh_params(4,2), dh_params(4,3), dh_params(4,4));
29 A{5} = dh_T(q5 + dh_params(5,1), dh_params(5,2), dh_params(5,3), dh_params(5,4));
30 A{6} = dh_T(q6 + dh_params(6,1), dh_params(6,2), dh_params(6,3), dh_params(6,4));
31 A06 = A{1} * A{2} * A{3} * A{4} * A{5} * A{6};
32 A06_simplified = simplify(A06);
33 p_E = A06_simplified(1:3, 4);

34 angle_sets = {
35     [-29.35, 29.88, -29.79, -1.31, -0.7, 44.29];
36     [ 39.63, -90.52, 89.29, -1.31, -0.08, 44.29];
37     [119.35, 40.51, -9.14, 0.08, 0.79, 44.29]
38 };
39 results_python = {
40     [-38.29290333854706, -51.843486531059966, 397.00605543805335];
41     [164.73758623779744, 54.02632094737639, 299.15136699999715];
42     [114.48700423889696, -75.52803724336692, 384.9371109879319];
43 };
44 for i = 1:length(angle_sets)
45     angles_deg = angle_sets{i};
46     angles_rad = angles_deg * (pi / 180);
47     python_result = results_python{i};
48     p_E_val = double(subs(p_E, ...
49         {q1, q2, q3, q4, q5, q6}, ...
50         num2cell(angles_rad)));
51     fprintf('q1 = %.2f°, q2 = %.2f°, q3 = %.2f°, q4 = %.2f°, q5 = %.2f°, q6 = %.2f°\n', ...
52         angles_deg(1), angles_deg(2), angles_deg(3), angles_deg(4), angles_deg(5), angles_deg(6));
53     fprintf('p_E=\n');
54     fprintf('    [%.16f;\n', p_E_val(1));
55     fprintf('    [%.16f;\n', p_E_val(2));
56     fprintf('    [%.16f;\n', p_E_val(3));
57     diff = p_E_val - python_result;
58     fprintf('p_E-p_{Epython}=\n');
59     fprintf('    [%.16f;\n', diff(1));
60     fprintf('    [%.16f;\n', diff(2));
61     fprintf('    [%.16f;\n', diff(3));
62     nn=sqrt(sum(diff.^2));
63     fprintf('norm: ||p_E - p_{Epython}|| = %.16f\n', nn);
64     fprintf('-----');
65     fprintf('\n');
66 end

```

Σχήμα 4: Κώδικας Matlab με την προσέγγιση  $\frac{\pi}{2} = 1.5708$

Η έξοδος του προγράμματος δίνεται ακολούθως.

```

q1 = -29.35°, q2 = 29.88°, q3 = -29.79°, q4 = -1.31°, q5 = -0.70°, q6 = 44.29°
p_E=
    [-38.2929033385471129;
     -51.8434865310599307;
     397.0060554380533517]

p_E-p_{Epython}=
    [-0.0000000000000497;
     0.0000000000000355;
     0.0000000000000000]

norm: ||p_E - p_{Epython}|| = 0.0000000000000611
-----
q1 = 39.63°, q2 = -90.52°, q3 = 89.29°, q4 = -1.31°, q5 = -0.08°, q6 = 44.29°
p_E=
    [164.7375862377975011;
     54.0263209473764121;
     299.151366999970944]

p_E-p_{Epython}=
    [0.0000000000000568;
     0.0000000000000213;
     -0.0000000000000568]

norm: ||p_E - p_{Epython}|| = 0.0000000000000832
-----
q1 = 119.35°, q2 = 40.51°, q3 = -9.14°, q4 = 0.08°, q5 = 0.79°, q6 = 44.29°
p_E=
    [114.4870042388970006;
     -75.5280372433670095;
     384.9371109879318169]

p_E-p_{Epython}=
    [0.0000000000000426;
     -0.0000000000000853;
     -0.0000000000000568]

norm: ||p_E - p_{Epython}|| = 0.0000000000001110
-----

```

Σχήμα 5: Έξοδος κώδικα Matlab

Παρατηρούμε πλέον ότι οι αποκλίσεις από τις τιμές που υπολόγισε ο κώδικας Python είναι αμελητέες. Δείξαμε συνεπώς ότι οι συντεταγμένες που προσδιορίσαμε με το κινηματικό μοντέλο διαφέρουν από εκείνες που υπολογίζει ο κώδικας Python αποκλειστικά λόγω της προσέγγισης  $\frac{\pi}{2} = 1.5708$ .

## 4 3ο Μέρος - Kinesthetic Teaching

### 4.1 Στόχος και Περιγραφή Διαδικασίας

Στο τρίτο μέρος της εργαστηριακής άσκησης, στόχος ήταν η υλοποίηση του σενάριου λήψης και εναπόθεσης (pick-and-place) που εκτελέστηκε στο πρώτο μέρος, χωρίς όμως την εκ των προτέρων γνώση των καρτεσιανών συντεταγμένων των σημείων. Για τον σκοπό αυτό χρησιμοποιήθηκε η μέθοδος της Κιναισθητικής Διδασκαλίας (Kinesthetic Teaching).

Η μέθοδος αυτή βασίζεται στη φυσική αλληλεπίδραση του χρήστη με το ρομπότ. Ο χειριστής καθοδηγεί χειροκίνητα τον βραχίονα στις επιθυμητές θέσεις και το σύστημα καταγράφει τις γωνίες των αρθρώσεων, ώστε να μπορεί να επαναλάβει την κίνηση αυτόνομα αργότερα.

Για την υλοποίηση χρησιμοποιήθηκε το γραφικό περιβάλλον (GUI) που αναπτύχθηκε σε Python (βιβλιοθήκη tkinter), το οποίο παρείχε τα απαραίτητα εργαλεία για την αποθήκευση σημείων (poses) και τον έλεγχο της αρπάγης.

### 4.2 Βήματα Εκτέλεσης

Η διαδικασία που ακολουθήθηκε για την εκπαίδευση του ρομπότ ήταν η εξής:

1. **Έναρξη Καταγραφής:** Μέσω της επιλογής “Teaching points to robot” στο GUI, ενεργοποιήθηκε η λειτουργία εκμάθησης, η οποία επιτρέπει την ελεύθερη κίνηση των αρθρώσεων του ρομπότ από τον χρήστη.
2. **Καθορισμός Διαδρομής:** Ο βραχίονας οδηγήθηκε χειροκίνητα στα ίδια σημεία ενδιαφέροντος με το 1ο μέρος (Προσέγγιση → Λήψη → Ανύψωση → Εναπόθεση → Απομάκρυνση).
3. **Αποθήκευση Σημείων και Εντολών:**
  - Σε κάθε επιθυμητή θέση, γινόταν αποθήκευση της στάσης του ρομπότ μέσω της εντολής “Save pose”.
  - Στα σημεία λήψης και εναπόθεσης, προστέθηκαν οι κατάλληλες εντολές για τον έλεγχο της αρπάγης πατώντας τα κουμπιά “Catch object” (κλείσιμο) και “Release object” (άνοιγμα) αντίστοιχα.
4. **Εκτέλεση:** Μετά την ολοκλήρωση της καταγραφής (“Stop recording”), το ρομπότ εκτέλεσε αυτόνομα το αποθηκευμένο σενάριο πατώντας “Run robot program”, μεταφέροντας επιτυχώς τον κύβο.

Η εκτέλεση της διαδικασίας φαίνεται στο επισυναπτόμενο βίντεο `part_3_kinesthetic_pnp.mp4`.

### 4.3 Σύγκριση με το 1ο Μέρος (Pick-and-Place με Συντεταγμένες)

Η βασική διαφορά μεταξύ των δύο μερών έγκειται στον τρόπο προγραμματισμού της κίνησης:

Στο **1ο Μέρος** απαιτούνταν η ακριβής γνώση των καρτεσιανών συντεταγμένων  $(x, y, z)$  για κάθε σημείο της διαδρομής. Ο προγραμματισμός γινόταν μέσω στατικού αρχείου κειμένου (.txt) και η ακρίβεια εξαρτιόταν από μετρήσεις στον χώρο ή υπολογισμούς. Αντιθέτως, στο **2ο Μέρος**, οι άγνωστες αρχικά συντεταγμένες προέκυψαν δυναμικά από τους encoders του ρομπότ καθώς ο χρήστης το μετακινούσε στον χώρο. Το ρομπότ “έμαθε” τη διαδρομή μέσω μίμησης και όχι μέσω μαθηματικού ορισμού σημείων.

#### 4.4 Πλεονεκτήματα της μεθόδου Kinesthetic Teaching

Τα κύρια πλεονεκτήματα της μεθόδου που παρατηρήθηκαν είναι τα εξής:

1. **Ευκολία και Ταχύτητα Προγραμματισμού:** Δεν απαιτείται συγγραφή κώδικα ή πολύπλοκων μετρήσεων στον χώρο εργασίας. Ένας χρήστης χωρίς εξειδικευμένες γνώσεις προγραμματισμού μπορεί να “διδάξει” στο ρομπότ μια εργασία πολύ γρήγορα.
2. **Προσαρμοστικότητα:** Η μέθοδος είναι ιδανική για αδόμητα περιβάλλοντα όπου οι θέσεις των αντικειμένων δεν είναι γνωστές εκ των προτέρων ή αλλάζουν συχνά.
3. **Ασφάλεια και Ακρίβεια:** Καθώς ο χρήστης κινεί το ρομπότ, έχει άμεση αντίληψη των ορίων του χώρου εργασίας και των πιθανών εμποδίων, αποφεύγοντας συγκρούσεις που θα μπορούσαν να προκύψουν από λανθασμένο υπολογισμό συντεταγμένων στο 1ο μέρος.

#### 4.5 Σχολιασμός Κώδικα και Μηχανισμός Επαλήθευσης Θέσης

Ένα σημαντικό στοιχείο της υλοποίησης του λογισμικού ελέγχου (αρχείο `utils.py`) είναι η διαχείριση της επιβεβαίωσης της θέσης του ρομπότ. Όπως επισημαίνεται στις οδηγίες της άσκησης, οι ενσωματωμένες συναρτήσεις σύγχρονης κίνησης (`sync_send_angles` και `sync_send_coords`) παρουσίασαν πειραματικά αστάθειες και εσφαλμένες εκτελέσεις.

Για την αντιμετώπιση αυτού του προβλήματος, αναπτύχθηκε μια προσαρμοσμένη (custom) μέθοδος ελέγχου (`send_coordinates_synchronous` και `send_angles_synchronous`), η οποία λειτουργεί ως εξής:

1. **Έλεγχος Κίνησης:** Ελέγχει αρχικά αν το ρομπότ κινείται μέσω της εντολής `is_moving()`.
2. **Ανατροφοδότηση από Encoders:** Δεν βασίζεται μόνο στην εντολή αποστολής, αλλά διαβάζει τις πραγματικές τιμές των γωνιών/συντεταγμένων από τους encoders (`get_angles`, `get_coords`).
3. **Όριο Ανοχής (Threshold):** Συγκρίνει την επιθυμητή θέση με την πραγματική θέση. Η κίνηση θεωρείται ολοκληρωμένη μόνο αν η διαφορά τους είναι μικρότερη από ένα καθορισμένο κατώφλι ανοχής (`threshold`), όπως ορίζεται στη συνάρτηση `is_angle_close`.
4. **Μηχανισμός Timeout:** Για την αποφυγή ατέρμονων βρόχων σε περίπτωση μηχανικής εμπλοκής, έχει υλοποιηθεί μηχανισμός χρονικού ορίου (`timeout`).



Η προσέγγιση αυτή διασφαλίζει ότι το ρομπότ έχει φυσικά προσεγγίσει τον στόχο πριν εκτελεστεί η επόμενη εντολή, προσφέροντας μεγαλύτερη αξιοπιστία σε σχέση με την open-loop αποστολή εντολών.