ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΊΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗΝ ΡΟΜΠΟΤΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Κατασκευή και προγραμματισμός ρομποτικού βραχίονα»

Εκπονητής: **Τράπαλης Ξενοφών**

Επιβλέπων Καθηγητής: **Δημήτριος Σαγρής**

Σέρρες, 14-9-2023

Υπεύθυνη Δήλωση Φοιτητών:

Ο κάτωθι υπογεγραμμένος φοιτητής, έχοντας επίγνωση των συνεπειών του Νόμου περί λογοκλοπής, δηλώνει υπεύθυνα ότι είναι συγγραφέας αυτής της Μεταπτυχιακής Εργασίας, αναλαμβάνοντας την ευθύνη επί ολοκλήρου του κειμένου εξ ίσου, έχοντας δε αναφέρει στην Βιβλιογραφία όλες τις πηγές τις οποίες χρησιμοποίησε. Δηλώνει επίσης ότι, οποιοδήποτε στοιχείο ή κείμενο το οποίο έχει ενσωματώσει στην εργασία του προερχόμενο από βιβλία, άλλες εργασίες ή το διαδίκτυο, γραμμένο επακριβώς ή παραφρασμένο, το έχει πλήρως αναγνωρίσει ως πνευματικό έργο άλλου συγγραφέα και έχει αναφέρει ανελλιπώς το όνομά του και την πηγή προέλευσης.

Ο Φοιτητής:

Τράπαλης Ξενοφών

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην ολοκλήρωσή της.

Ευχαριστώ τον Δρ. Σαγρή Δημήτριο για την πολύτιμη βοήθεια του και τις συμβουλές που μου έδωσε.

Τέλος, ευχαριστώ θερμά την οικογένεια μου που χωρίς αυτή δεν θα τα είχα καταφέρει.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία διεξήχθη στο πλαίσιο των μεταπτυχιακών σπουδών του μεταπτυχιακού προγράμματος στη ρομποτική. Αντικείμενο της αποτελεί η κατασκευή και ο προγραμματισμός του ρομποτικού βραχίονα έξι αξόνων μέσω του CODESYS, το οποίο είναι ένα περιβάλλον ανάπτυξης για προγραμματισμό εφαρμογών ελεγκτή σύμφωνα με το διεθνές πρότυπο IEC 61131 – 3. Για τον προγραμματισμό του ρομποτικού βραχίονα χρησιμοποιείται το SoftMotion το οποίο είναι μια βιβλιοθήκη του CODESYS αποκλειστικά για την κίνηση των σερβοκινητήρων και την επίλυση κινηματικών προβλημάτων.

Επιπλέον, για την απεικόνιση της κίνησης σε πραγματικό χρόνο χρησιμοποιείται το CoppeliaSim το οποίο είναι ένα πρόγραμμα προσομοίωσης σε φυσικό περιβάλλον ωστόσο εδώ χρησιμοποιείται κυρίως ως πρόγραμμα απεικόνισης καθώς οι αρθρώσεις του ρομποτικού βραχίονα λαμβάνουν τις τιμές απευθείας από το CODESYS. Για την επικοινωνία του CODESYS PLC με το CoppeliaSim χρησιμοποιείται το πρωτόκολλο επικοινωνίας Modbus TCP το οποίο εκτελείται μέσω της Python.

Τέλος, μελετάται η επίλυση του κινηματικού προβλήματος ενός ρομποτικού βραχίονα, η κινηματική ανάλυση αλλά και οι παράμετροι Denavit – Hartenberg.

ENGLISH SUMMARY (ΑΓΓΛΙΚΗ ΠΕΡΙΛΗΨΗ)

This thesis was carried out as part of the graduate studies of the master's program in robotics. Its object is the construction and programming of a six-axis robotic arm through CODESYS, which is a development environment for programming controller applications according to the international standard IEC 61131 – 3. To program the robotic arm, SoftMotion is used which is a CODESYS library exclusively for moving servo motors and solving kinematic problems.

In addition, to visualize in real time the motion CoppeliaSim is used, which is a simulation program in an environment which inherit the law of physics, however in our case it is used as a visualization program as the joints of the robotic arm receive the values directly from CODESYS. For the communication of CODESYS PLC with CoppeliaSim, the Modbus TCP communication protocol is used which is executed through Python.

Finally, the solution of the kinematic problem of a robotic arm, the kinematic analysis and Denavit – Hartenberg parameters are studied.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ $1^{ο}$

1 Εισαγωγή

1.1 Κίνητρο

Τις τελευταίες δεκαετίες ο κλάδος της ρομποτικής έχει μεγάλη ανάπτυξη σε πολλούς επαγγελματικούς τομείς όπως η ιατρική, η αεροναυπηγική, η βιομηχανία. Ιδιαίτερα στον κλάδο της βιομηχανίας έχουμε μπει στην $4^{η}$ γενιά της βιομηχανίας (Industry 4.0) το οποίο σημαίνει ότι στόχος για τις γραμμές παραγωγής δεν είναι μόνο η αυτοματοποίηση τους με χρήση ρομπότ αλλά και η πλήρης αυτοματοποίηση ολόκληρων μονάδων παραγωγής χρησιμοποιώντας νέες τεχνολογίες όπως το Internet of Things(IoT).

Ένα βιομηχανικό ρομπότ σύμφωνα με το πρότυπο ISO είναι ένας αυτόματα ελεγχόμενος επαναπρογραμματιζόμενος πολλαπλός βραχίονας κατασκευασμένος με τρεις ή περισσότερους άξονες. Στον κλάδο της ρομποτικής ουσιαστικά μελετάμε και σχεδιάζουμε ρομποτικές μηχανές οι οποίες να μπορούν να βοηθήσουν ή να αντικαταστήσουν τον άνθρωπο, ιδιαίτερα σε εφαρμογές που είναι επίπονες για αυτόν, έτσι ώστε να συνδυάζουν την φυσική δραστηριότητα και την λήψη αποφάσεων.

Το κίνητρο για την ενασχόληση με την συγκεκριμένη εργασία δόθηκε από την πεποίθηση ότι η ρομποτική είναι ένας πολλά υποσχόμενος κλάδος, ο οποίος μπορεί να προσφέρει πολλά στην ανθρωπότητα και να βελτιώσει πολύ την ποιότητα ζωής των ανθρώπων. Η μελέτη κατασκευής και προγραμματισμού ενός ρομποτικού βραχίονα θεωρώ πως είναι η βάση της ρομποτικής επιστήμης η οποία έχει να δώσει πολλά ακόμα στην ανθρωπότητα και ως εκ τούτου αποφάσισα να ασχοληθώ με αυτό το αντικείμενο στην παρούσα εργασία.

1.2 Αντικείμενο εργασίας

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η κατασκευή ενός ρομποτικού βραχίονα βιομηχανικού τύπου του οποίου υπάρχει μόνο ο σκελετός (σασί). Θα σχεδιαστεί το ηλεκτρολογικό σχέδιο, θα κατασκευαστεί ο ηλεκτρολογικός πίνακας, θα γίνει η κατάλληλη επιλογή servo κινητήρων και servo drive, θα δημιουργηθεί ο κώδικας για την λειτουργία καθώς και το user interface για τον έλεγχο του από τον χρήστη.

Ο ρομποτικός βραχίονας είναι ένας βραχίονας της KUKA έξι αξόνων ύψους 2.5 μέτρων κατά προσέγγιση και τα κινητά μέρη μπορούν να επεκταθούν κατά προσέγγιση 3 μέτρα, ο οποίος είναι μη λειτουργικός. Ως εκ τούτου θα πρέπει να γίνει εκ νέου μελέτη για την τοποθέτηση των νέων servo κινητήρων καθώς και της καλωδίωσης τους. Εν συνεχεία, θα πρέπει να γίνει μελέτη για την κατασκευή του ηλεκτρολογικού πίνακα για την σωστή τοποθέτηση των servo drive και της επικοινωνίας τους με το PLC με ethercat πρωτόκολλο επικοινωνίας. Επιπλέον, θα πρέπει να γίνει όλη η καλωδίωση του πίνακα από το τριφασικό (400V) έως τα 24V που θα χρειαστούν για τα επιμέρους στοιχεία του πίνακα. Έπειτα θα πρέπει να επιτευχθεί η επικοινωνία όλων των drives με το PLC. Επιπλέον θα πρέπει να γίνει μελέτη για την ασφάλεια, τοποθέτηση E – stop καθώς και συστήματος ασφαλείας για την άμεση ακύρωση των κινήσεων του ρομποτικού βραχίονα σε περίπτωση ανάγκης.

Για τον προγραμματισμό του ρομποτικού βραχίονα θα χρησιμοποιηθεί το CODESYS το οποίο είναι περιβάλλον ανάπτυξης για προγραμματισμό εφαρμογών ελεγκτή σύμφωνα με το διεθνές πρότυπο IEC 61131 – 3. Πιο συγκεκριμένα, για τον έλεγχο κίνησης του ρομποτικού βραχίονα θα χρησιμοποιηθεί το SoftMotion το οποίο είναι μια βιβλιοθήκη για την βοήθεια της υλοποίησης της κίνησης, ενσωματωμένο στο περιβάλλον ανάπτυξης του CODESYS. Επιπλέον θα περιέχει φυσικούς διακόπτες ενεργοποίησης και στοπ καθώς και διακόπτες ενεργοποίησης και κίνησης των αξόνων στις τρεις διαστάσεις x, y, z στο User Interface.

Τελικός στόχος είναι η επίτευξη του συγχρονισμού των έξι servo κινητήρων έτσι ώστε να μπορούν να υλοποιήσουν κινήσεις τοποθέτησης στον χώρο με επιτυχία ακρίβειας δεκάτου χιλιοστού κάτι το οποίο είναι αρκετά απαιτητικό.

1.3 Απόκτηση Γνώσεων

Κατά την συγγραφή της εργασίας οι γνώσεις που αποκτήθηκαν αφορούσαν πολλά διαφορετικά πεδία. Αρχικά ερευνήθηκε το κινηματικό πρόβλημα και η επίλυση αυτού, μελετήθηκε η μεθοδολογία για την εύρεση των παραμέτρων του Denavit – Hartenberg αλλά και πρακτικά το πως κινείται ένας ρομποτικός βραχίονας και η σημαντικότητα της δυνατότητας να αλλάζουμε κατά βούληση το σύστημα συντεταγμένων του ρομποτικού βραχίονα. Επιπλέον μελετήθηκε σε βάθος το SoftMotion και πως μπορεί να μας διευκολύνει για να επιλύσουμε το κινηματικό πρόβλημα και να μπορέσουμε να δώσουμε κίνηση σε ένα σύστημα αξόνων όπως είναι αυτό του ρομποτικού βραχίονα έξι αξόνων. Επιπλέον μελετήθηκε το πρόγραμμα προσομοίωσης CoppeliaSim το οποίο μπορεί να μας δώσει σε πραγματικές συνθήκες πως θα κινηθεί το σύστημα μας στον χώρο. Εν συνεχεία μας δόθηκε η δυνατότητα να ασχοληθούμε με τα συστήματα επικοινωνίας όπως το Modbus TCP, που μπορούμε να χρησιμοποιήσουμε μεταξύ δύο μηχανών ή και προγραμμάτων για την ανταλλαγή δεδομένων μέσω πακέτων και την χρήση της Python για τον διαμοιρασμό δεδομένων.

1.4 Επιθυμητό Αποτέλεσμα

Το επιθυμητό αποτέλεσμα κατά τη διάρκεια της εργασίας είναι να καταφέρουμε να αναλύσουμε το κινηματικό πρόβλημα για έναν ρομποτικό βραχίονα έξι αξόνων να μελετήσουμε την ιστορία, τις ιδιότητες αλλά και την χρησιμότητα των ρομπότ στην ζωή μας, να αναλύσουμε τα εργαλεία αυτά που θα μας δώσουν την δυνατότητα να θέσουμε σε κίνηση έναν ρομποτικό βραχίονα έτσι ώστε να μπορεί να κινηθεί με ακρίβεια στον χώρο και τέλος να μπορεί να εκτελεί και αρχεία G – κώδικα ώστε να μπορούμε να το χρησιμοποιήσουμε μελλοντικά είτε σαν ρομπότ συγκόλλησης είτε σαν ρομπότ τρισδιάστατης εκτύπωσης ακόμα και σαν ρομπότ με λειτουργίες CNC μηχανής.

ΚΕΦΑΛΑΙΟ 2^{Ο}

2 Στάθμη γνώσεων

2.1 Τι είναι ρομποτική

Η ρομποτική είναι ένας σύγχρονος κλάδος της τεχνολογικής επιστήμης και συγκεκριμένα της Μηχανοηλεκτρονικής. Αντικείμενο μελέτης είναι ο καθορισμός και ο έλεγχος διαφόρων μηχανικών εξαρτημάτων (ρομπότ), τα οποία μπορούν να βρουν εφαρμογές σε διάφορους τομείς, όπως είναι ο τομέας της Ιατρικής. Η ρομποτική με το πέρασμα του χρόνου εισέρχεται δυναμικά στην καθημερινότητα των ανθρώπων με σκοπό την διευκόλυνση διαφόρων δραστηριοτήτων που ως τώρα ήταν επικίνδυνες, απαιτητικές ακόμα και χρονοβόρες ως προς την διεκπεραίωση τους (Φ.Ν. Κουμπούλης, Β.Γ. Μέρτζιος, 2002)

Το αμερικάνικο Ινστιτούτο Ρομπότ δίνει το δικό του ορισμό για το τι είναι ρομπότ, σύμφωνα με το οποίο λοιπόν *«ρομπότ είναι μια επαναπρογραμματιζόμενη πολυλειτουργική χειριστική διάταξη, σχεδιασμένη για τη μετακίνηση υλικών, εξαρτημάτων, εργαλείων και εξειδικευμένων διατάξεων, μέσω μεταβλητών, προγραμματισμένων κινήσεων για την εκτέλεση μιας σειράς εργασιών»* . Σε γενικές γραμμές το ρομπότ συγκροτείται από δύο συστήματα το μηχανικό στο οποίο περιλαμβάνεται το σύστημα κίνησης και το ηλεκτρονικό στο οποίο εμπεριέχεται και η επαναπρογραμματιζόμενη μνήμη του. Επιπλέον, για τα ρομπότ υπάρχουν διάφορα κριτήρια διάκρισης και αντίστοιχες κατηγοριοποιήσεις, μία από αυτές είναι η διάκριση τους σε τρεις «γενιές». Στην πρώτη, περιλαμβάνονται τα ρομπότ με περιορισμένη ευελιξία τα οποία χειρίζονται από ανθρώπους και συνήθως είναι απλά εργαλεία που επιτρέπουν την μετακίνηση επικίνδυνων αντικειμένων (π.χ. ραδιενεργών υλικών). Στην δεύτερη γενιά εντάσσονται τα ρομπότ «που είναι εφοδιασμένα με σταθερό πρόγραμμα δράσης, ενώ στην τρίτη γενιά κατατάσσονται τα ρομπότ που είναι εφοδιασμένα με:

- αισθητήριες «πληροφορίες» από το περιβάλλον
- διάταξη επεξεργασίας των πληροφοριών και
- κινητήριο σύστημα εκτέλεσης εργασιών.

Τα περισσότερα είδη ρομπότ ανήκουν στην κατηγορία ρομποτικών βραχιόνων και παρουσιάζουν διαφορετικό βαθμό αυτονομίας, δηλαδή ορισμένα ρομπότ προγραμματίζονται με σκοπό την ακριβή εκτέλεση συγκεκριμένων επαναλαμβανόμενων ενεργειών χωρίς να υπάρχουν μεταβολές

και υψηλός βαθμός ακριβείας. Οι δράσεις αυτές καθορίζονται από προγραμματισμένες ρουτίνες που ορίζουν την κατεύθυνση, την επιτάχυνση, την ταχύτητα, την επιβράδυνση και την απόσταση από μία μεριά συντονισμένων κινήσεων. Κάποια άλλα ρομπότ είναι περισσότερο ευέλικτα σχετικά με τον προσανατολισμό του αντικειμένου το οποίο λειτουργούν ή ακόμα και την εργασία που πρέπει να εκτελεστεί στο ίδιο αντικείμενο το οποίο μπορεί ακόμα να χρειαστεί να προσδιοριστεί από το ίδιο το ρομπότ. Προκειμένου να διδαχθούν την ακολουθία των κινήσεων τους απαιτείται η σύνδεση του ελεγκτή του με έναν υπολογιστή ή PLC. Το σύνολο, του ρομπότ και των περιφερειακών μηχανών του αναφέρεται ως κελί εργασίας ή κελί. Το κελί αποτελείται από την τροφοδοσία των μηχανημάτων τη μηχανή σχεδίασης και το ρομπότ. Οι διάφορες περιφερειακές μηχανές αποτελούν ένα ολοκληρωμένο σύστημα. Η αλληλεπίδραση του ρομπότ με τις υπόλοιπες μηχανές στο κελί θα πρέπει να προγραμματιστεί και ως προς τη θέση τους αλλά και τον συγχρονισμό τους (Ζωή Δουλγέρη, 2007).

2.2 Ιστορική αναδρομή

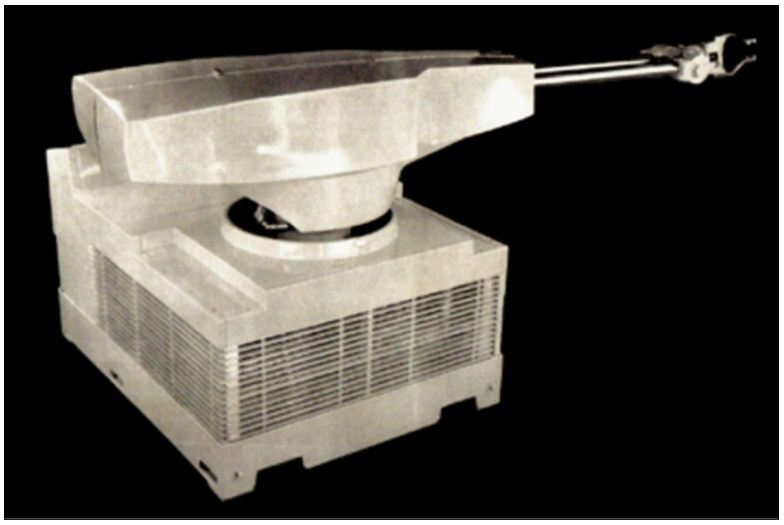
Η λέξη «ρομπότ» εμφανίζεται πρώτη φορά το 1921 στο έργο επιστημονικής φαντασίας R.U.R (Rossum's Universal Robot) του Τσέχου συγγραφέα Καρελ Τσαπεκ . Ο Κάρελ ήθελε σε αυτό το έργο να θίξει την αλλοίωση του ανθρώπου μέσα από τον αυτοματισμό της βιομηχανικής εποχής αλλά και των εργασιακών σχέσεων. Η λέξη ρομπότ έχει σλαβικές καταβολές και μεταφράζεται ως «καταναγκαστική εργασία». Έπειτα το 1942 η λέξη «ρομποτική» συναντάται σε μία σύντομη ιστορία του Ισαάκ Ασίμοφ, το «Runabout», ο οποίος έχει μια πιο οπτιμιστική άποψη για τα ρομπότ και τον ρόλο τους στην ανθρώπινη κοινωνία ως βοηθοί των ανθρώπων. Ο Ισσάκ Ασίμοφ πρότεινε τους τρεις "νόμους της ρομποτικής" οι οποίοι υιοθετήθηκαν από πολλές ακόμα ιστορίες. Οι τρεις νόμοι αυτοί είναι, ένα ρομπότ δεν θα τραυματίσει ποτέ έναν άνθρωπο ή μέσω τις απραξίας του να αφήσει έναν άνθρωπο να τραυματιστεί, ένα ρομπότ θα πρέπει να υπακούει στις εντολές των ανθρώπων εκτός και αν έρχονται σε σύγκρουση με τον πρώτο νόμο, ένα ρομπότ πρέπει να προστατεύει την ύπαρξη του εκτός και αν αυτό έρχεται σε σύγκρουση με τον πρώτο ή τον δεύτερο νόμο.

Βέβαια, την έννοια του ρομπότ τη συναντάμε ακόμα και στην Αρχαία Ελλάδα και συγκεκριμένα στη μυθολογία. Πρόκειται για τον Τάλω (Εικόνα 2.2.1.) ένα ανθρωπόμορφο κατασκεύασμα με τεράστια δύναμη που αρχικά το χρησιμοποιούσαν για να διώχνει τα εχθρικά πλοία από την Κρήτη. Αργότερα βέβαια εκμεταλλεύτηκε και για άλλες εργασίες (Παπακωνσταντίνου Πέτρος, 2020).

Εικόνα 2.2.1: Τάλως, το πρώτο ρομπότ που δημιουργήθηκε από την φαντασία

Ωστόσο ο πρώτος άνθρωπος που κατασκεύασε όντως ένα ανθρωπόμορφο προγραμματιζόμενο ρομπότ ήταν ο Al – Jazari το 1206, ο οποίος ήταν λόγιος, εφευρέτης και μηχανολόγος μηχανικός κατά την διάρκεια της χρυσής εποχής του Ισλαμ. Ο Al – Jazari νοείται ως "ο πατέρας της ρομποτικής" και όχι μόνο καθώς έχει καταγράψει άλλες 50 εφευρέσεις και νοείται και ως "ο πατέρας της σύγχρονης μηχανικής".

Ως πρωτοπόρος της σύγχρονης ρομποτικής όπως την γνωρίζουμε θεωρείται ο George C. Devol ο οποίος ήταν Αμερικάνος εφευρέτης και κατασκεύασε το πρώτο ψηφιακά προγραμματιζόμενο ρομποτ, το "Unimate" (Εικόνα 2.2.2.). Ωστόσο ήταν δέκα χρόνια μετά, το 1960 που κατάφερε το συγκεκριμένο ρομπότ να μπει στον χώρο της βιομηχανίας ύστερα από τις μερατροπές που υπέστει από τον Joseph Engleberger. Για την βιομηχανία ο Joseph Engleberger θεωρείται "ο πατέρας της ρομποτικής" (Αναστασοπούλου Ελένη, 2017).



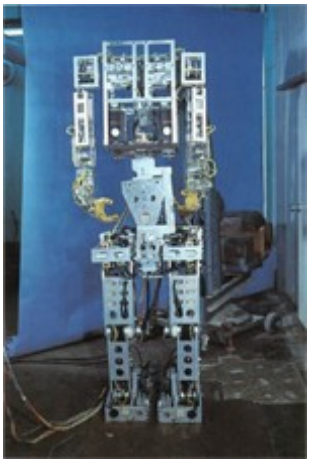
Εικόνα 2.2.2.: Unimate, το πρώτο προγραμματιζόμενο ρομπότ

Το 1960 στο πανεπιστήμιο του John Hopkins στη Βαλτιμόρη κατασκευάστηκε η πρώτη αυτοκινούμενη μηχανή που είχε τη δυνατότητα να επικοινωνεί με το εξωτερικό περιβάλλον σε μία στοιχειώδη βέβαια μορφή επικοινωνίας η επονομαζόμενη Beast (Εικόνα 2.2.3.). Αυτή η μηχανή μπορούσε να εντοπίζει εμπόδια, να αποφασίζει αν θα αποφύγει ένα εμπόδιο ή αν θα σταματήσει, καθώς και να αναζητά στους λευκούς τοίχους του εργαστηρίου τις μαύρες πρίζες και να επαναφορτίζεται για την αυτοσυντήρηση του. Μία δεκαετία αργότερα το ρομπότ αυτό μπορούσε να

επικοινωνήσει σε φυσική γλώσσα, δεχόταν εντολές τις οποίες μπορούσε να τις εκτελεί δημιουργώντας ένα δικό του σχεδιασμό.

Εικόνα 2.2.3.: Beast, το πρώτο αυτοκινούμενο με δυνατότητα επικοινωνίας με το εξωτερικό περιβάλλον

Με το πέρασμα των χρόνων σημειώνεται ραγδαία ανάπτυξη στον τομέα της ρομποτικής με πολλούς επιστήμονες να πειραματίζονται πάνω σε αυτό τον κλάδο. Χαρακτηριστικό παράδειγμα είναι μία ομάδα επιστημόνων από την Ιαπωνία όπου το 1971 δημιούργησαν το πρώτο ανθρωποειδές ρομπότ σε πραγματικές διαστάσεις το "WABOT – 1" (Εικόνα 2.2.4.), ενώ το 1986 κατασκευαστές της Honda αποφάσισαν να δημιουργήσουν ένα ρομπότ το οποίο θα ήταν σε θέση να στέκεται και να περπάτα στα δύο του πόδια.

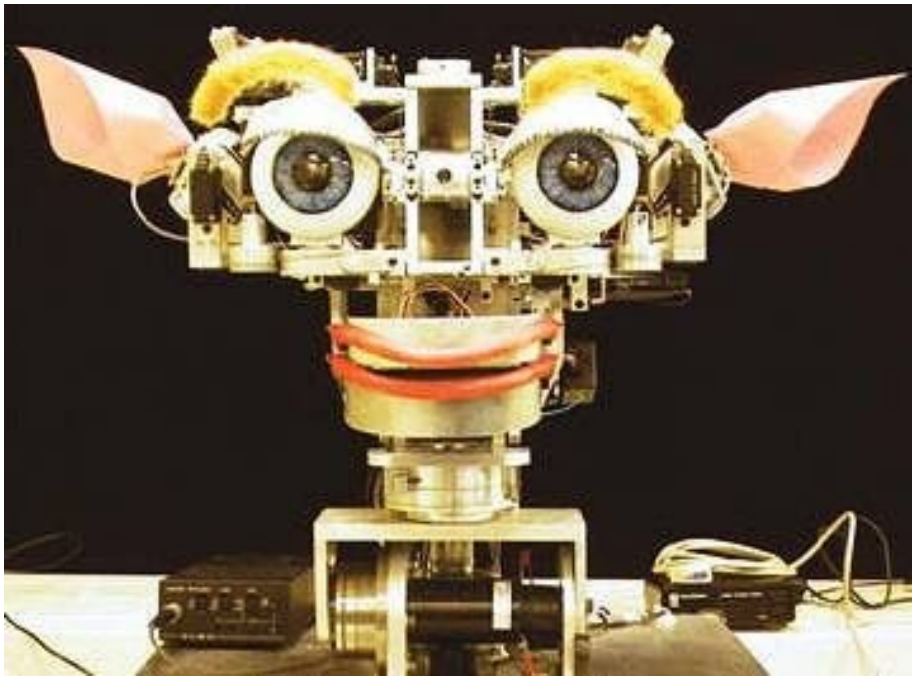


Εικόνα 2.2.4.: WABOT – 1: Το πρώτο ανθρωποειδές ρομπότ.

2.3 Ιδιότητες ρομπότ

Οι πέντε βασικές ιδιότητες που χαρακτηρίζουν ένα ρομπότ έτσι όπως τα ξέρουμε ως σήμερα είναι, η νοημοσύνη, η αντίληψη των αισθήσεων, η επιδεξιότητα, η ενέργεια και η ανεξαρτησία.

Η τεχνητή νοημοσύνη (Artificial Intelligence) των ρομπότ στηρίζεται στον τρόπο που λειτουργεί ο ανθρώπινος εγκέφαλος. Όπως ο ανθρώπινος εγκέφαλος αποτελείται από ένα σύνολο νευρώνων διασυνδεδεμένων κατάλληλα έτσι ώστε ο άνθρωπος να μπορεί να σκέφτεται και να αντιδρά κατάλληλα ανάλογα την περίσταση έτσι και ένα ρομπότ χρησιμοποιεί διαφορετικές βαθμίδες επεξεργασίας. Στο πανεπιστήμιο της Μασαχουσέτης (Massachusetts Institute of Technology) έχει αναπτυχθεί ένα ρομπότ με τεχνητή νοημοσύνη γνωστό ως Kismet (Εικόνα 2.3.1) το οποίο χρησιμοποιεί τα υψηλότερα επίπεδα επεξεργασίας για να αντιμετωπίζει δύσκολα προβλήματα ενώ τα χαμηλότερα επίπεδα επεξεργασίας για τα εύκολα προβλήματα και για διαδικασίες με επαναληψιμότητα. Τα ρομπότ βέβαια υστερούν στην συναισθηματική νοημοσύνη σε αντίθεση με τον άνθρωπο ωστόσο είναι ικανά να δείξουν ενσυναίσθηση (Ζωή Δουλγέρη, 2007).

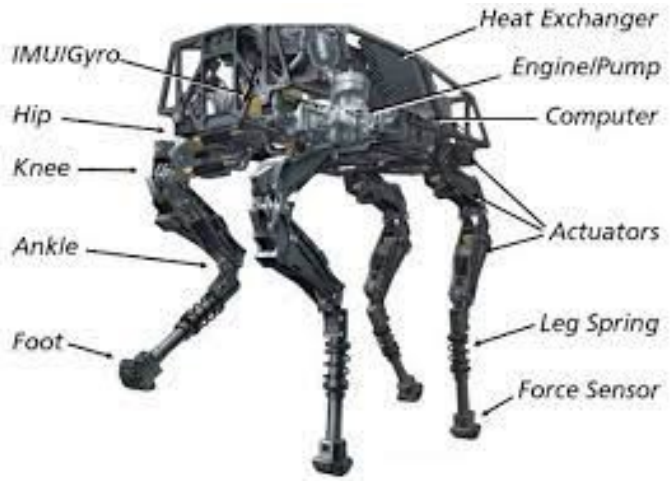


Εικόνα 2.3.1.: Kismet

Οι αισθήσεις των ανθρώπων χωρίζονται στις εξής, όραση, ακοή, αφή, όσφρηση, και γεύση οι οποίες όλες έχουν ήδη ενταχθεί ή γίνονται προσπάθειες να ενταχθούν στα ρομπότ. Ένα πολύ καλό παράδειγμα για να καταλάβουμε πως οι αισθήσεις των ρομπών δουλεύουν είναι τα αυτοκινούμενα

οχήματα, τα οποία μέσω αισθητηρίων LIDAR, ραντάρ, κάμερες, GPS και encoder συλλέγουν δεδομένα από το περιβάλλον σε πραγματικό χρόνο, τα οποία μέσω αναπτυγμένων αλγορίθμων λύνουν το πρόβλημα της αυτοκίνησης σε πραγματικό χρόνο, έτσι ώστε το όχημα να μπορεί να αντιλαμβάνεται και να προβλέπει μέσω της τεχνητής νοημοσύνης την ύπαρξη άλλων οχημάτων, ανθρώπων, εμποδίων και τις πιθανές κινήσεις αυτών.

Η επιδεξιότητα στα ρομπότ αναφέρεται στην λειτουργικότητα των άκρων ενός ρομπότ αλλά και του σώματος ενός ρομπότ. Το γνωστό BigDog ρομπότ (Εικόνα 2.3.2.) είναι ένα χαρακτηριστικό παράδειγμα επιδεξιότητας το οποίο όσο και να το σπρώχνουν βρίσκει τον τρόπο ώστε να σταθεί έτσι και να μην πέσει. Το συγκεκριμένο ρομπότ αναπτύσσεται από την Boston Dynamics. Ανθρωπόμορφα ρομποτικά άκρα που μπορούν να εκτελέσουν λεπτές εργασίες όπως να ανοίξουν ένα βάζο ή να γράψουν ένα κείμενο μπορούν να είναι πολύ βοηθητικά για τον άνθρωπο ή και ακόμα να σώσουν την ζωή του όταν αυτές πρέπει να εκτελεστούν σε ακραίες συνθήκες (Φ.Ν. Κουμπούλης, Β.Γ. Μέρτζιος,2002).



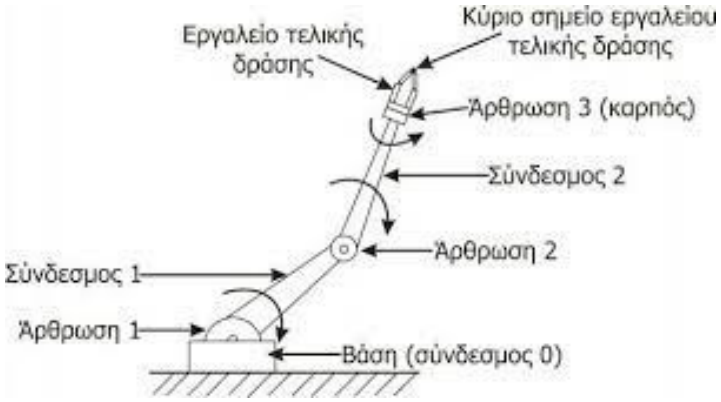
Εικόνα 2.3.2.: BigDog robot

Όλα τα ρομπότ απαιτούν ενέργεια ώστε να λειτουργήσουν. Υπάρχουν πολλοί διαφορετικοί τρόποι για την τροφοδοσία ενός ρομπότ, όπως μπαταρίες, κυψέλες καυσίμων, γεννήτριες. Όλες αυτές οι πηγές ενέργειας αποθηκεύονται τοπικά αλλά είναι προσωρινές και χρειάζονται συνεχώς ανανέωση. Ένα ακόμα μέσο τροφοδοσίας είναι οι φυσικές πηγές ενέργειας (πχ. ηλιακή ενέργεια, αιολική κλπ) ωστόσο οι πηγές ενέργειας αυτές περιορίζουν τις δυνατότητες και την ελευθέρια των ρομπότ. Όλα τα παραπάνω συντελούν στην ανεξαρτησία ενός ρομπότ.

2.4 Δομή ρομποτικού βραχίονα

Ένας ρομποτικός βραχίονας αποτελείται από τη βάση του ρομποτικού βραχίονα, τους συνδέσμους, τις αρθρώσεις, το εργαλείο τελικής δράσης και το κύριο σημείο του εργαλείου τελικής δράσης. Η αρίθμηση των αρθρώσεων και των συνδέσμων γίνεται πάντα ξεκινώντας από την βάση (Εικόνα 2.4.1.)

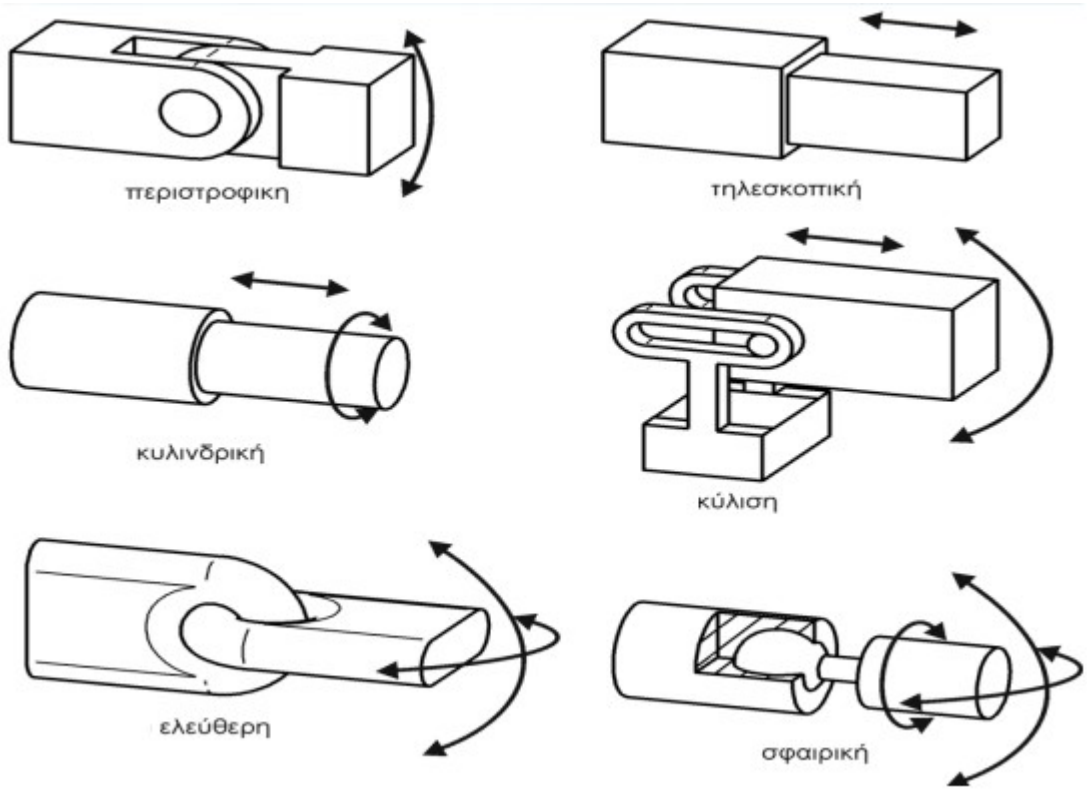
Ως βάση του ρομποτικού βραχίονα ορίζεται το τμήμα του ρομποτικού βραχίονα που είναι στερεωμένο στο έδαφος ή γενικά στο περιβάλλον του χώρου εργασίας του. Στη βάση του ρομποτικού βραχίονα συνδέονται μία σειρά διαδοχικών στερεών σωμάτων που ονομάζονται σύνδεσμοι και συνδέονται μεταξύ τους μέσω των αρθρώσεων σχηματίζοντας μια κινηματική αλυσίδα. Κινηματική αλυσίδα είναι το σύνολο των στερεών σωμάτων που συνδέονται μεταξύ τους μέσω των αρθρώσεων. Το εργαλείο τελικής δράσης είναι το εργαλείο με το οποίο ο βραχίονας εκτελεί εργασίες όπως ηλεκτροσυγκόλληση, αρπάγη κ.α. Τέλος το κύριο σημείο του εργαλείου τελικής δράσης είναι η θέση που είναι σημαντική για την εκτέλεση της εργασίας π.χ το σημείο ένωσης της αρπάγης (Ζωή Δουλγέρη 2007).



Εικόνα 2.4.1.: Αρίθμιση αρθρώσεων

Οι αρθρώσεις μπορεί να είναι πρισματικές, περιστροφικές, σφαιρικές, κυλινδρικές, ελεύθερες ή κύλισης (Εικόνα 2.4.2.). Οι περιστροφικές αρθρώσεις επιτρέπουν την σχετική στροφή μεταξύ δύο γειτονικών συνδέσμων και δίνει έναν βαθμό ελευθερίας αποκόπτοντας κάθε άλλη δυνατή κίνηση. Ως βαθμός ελευθερίας ορίζεται "*το σύνολο των ανεξαρτήτων μεταβλητών με βάση τις οποίες περιγράφεται πλήρως η θέση των υλικών σημείων του συστήματος"*. Οι πρισματικές ή τηλεσκοπικές αρθρώσεις είναι αυτές που επιτρέπουν την κίνηση σε ευθεία γραμμή, δίνουν έναν βαθμό ελευθερίας και επιτρέπουν την

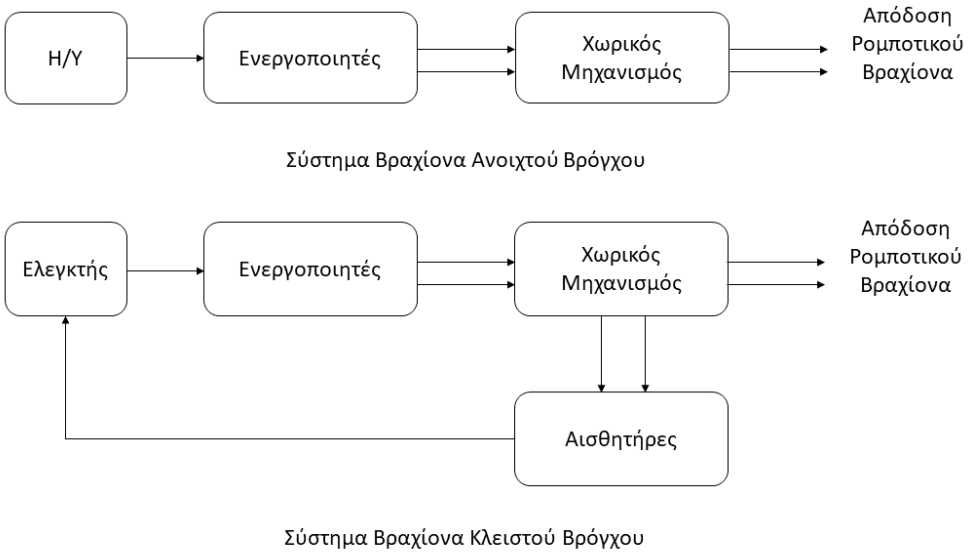
κίνηση προς την διεύθυνση ενός άξονα. Ωστόσο υπάρχουν και σύνθετες αρθρώσεις που επιτρέπουν την κίνηση σε πάνω από έναν άξονα δίνοντας πάνω από έναν βαθμό ελευθερίας. Η κυλινδρική άρθρωση επιτρέπει την μεταφορική κίνηση ως προς έναν άξονα και την περιστροφική ώς προς τον άξονα αυτό, κατ' επέκταση δίνει δύο βαθμούς ελευθερίας. Η ελεύθερη άρθρωση επιτρέπει την περιστροφική κίνηση ως προς δύο άξονες εμποδίζοντας οποιαδήποτε άλλη κίνηση. Ακόμα μία άρθρωση δύο βαθμών ελευθερίας είναι αυτή της κύλισης η οποία επιτρέπει την μεταφορική κίνηση ως προς έναν άξονα και την περιστροφική κίνηση ως προς τον κάθετο άξονα της μεταφορικής κίνησης. Τέλος η σφαιρική άρθρωση αποτελείται από τρεις βαθμούς ελευθερίας επιτρέποντας και τις τρεις περιστροφικές κινήσεις και εμποδίζοντας όλες τις μεταφορικές.



Εικόνα 2.4.2.: Τύποι αρθρώσεων

Οι ρομποτικοί βραχίονες χωρίζονται σε δύο κατηγορίες, στους βραχίονες ανοιχτού βρόγχου και στους βραχίονες κλειστού βρόγχου. Οι βραχίονες ανοιχτού βρόγχου εκτελούν την εργασία που είναι προγραμματισμένα να κάνουν χωρίς όμως κάποιον έλεγχο ανατροφοδότησης, εκτελούν τυφλά τις

εντολές που τους δίνουμε, ενώ οι βραχίονες κλειστού βρόγχου έχουν έλεγχο θέσης, ταχύτητας, επιτάχυνσης με χρήση αισθητηρίων και ελεγκτών συστημάτων αυτομάτου ελέγχου (Εικόνα 2.4.2.).



Εικόνα 2.4.2.: Κατηγορίες βραχιόνων

2.5 Κατηγορίες των ρομπότ στην βιομηχανία

Τα βιομηχανικά ρομπότ καθορίζονται από τα πρότυπα ISO ως ένας αυτόματα ελεγχόμενος επαναπρογραμματιζόμενος, πολλαπλός βραχίονας κατασκευασμένος με τρεις ή και περισσότερους άξονες. Τα βιομηχανικά ρομπότ ευρείας χρήσης είναι τα αρθρωτά, τα SCARA, τα σφαιρικά, τα κυλινδρικά, τα Cobot, τα ρομπότ που χρησιμοποιούν τις καρτεσιανές συντεταγμένες και τα ρομποτ Gantry.

Πιο αναλυτικά, τα αρθρωτά ρομπότ είναι από τους πιο γνωστούς τύπους ρομπότ καθώς είναι αυτά που στην όψη τους μοιάζουν με τον ανθρώπινο βραχίονα. Αυτά τα ρομπότ ταξινομούνται βάσει του αριθμού των σημείων περιστροφής ή των αξόνων που έχουν. Στην βιομηχανία αυτά που χρησιμοποιούνται περισσότερο είναι τα αρθρωτά ρομπότ 6 αξόνων ή αλλιώς ρομπότ με 6 βαθμούς ελευθερίας κι αυτό γιατί προσφέρει μεγαλύτερη ευελιξία, η οποία σε συνδυασμό με την εμβέλεια και την επιδεξιότητα τα καθιστούν ιδανικά για να εκτελούν κινήσεις ακριβείας. Κάποια από τα πλεονεκτήματα χρήσης των συγκεκριμένων ρομπότ είναι η συνεχόμενη λειτουργία τους χωρίς την παρουσία νεκρών χρόνων, η ικανότητα τους να διασφαλίζουν την ποιότητα της παραγωγής εξαιτίας της ακρίβειας τους, η ικανότητά τους να ανταπεξέρχονται σε οποιοδήποτε ρυθμό παραγωγής χωρίς επιπλέον κόστος σε ανθρώπινο δυναμικό, η ευελιξία που παρουσιάζουν σε πλήθος εργασιών, η διευκόλυνση των εργαζομένων ως προς τις συνθήκες εργασίας τους καθώς τα ρομπότ επωμίζονται το δυσκολότερο ή και επικίνδυνο κομμάτι εργασίας. Κάποιοι από τους τομείς που συναντάται τέτοιου είδους ρομπότ είναι η βιομηχανία τροφών και ποτών, μετάλλων, φαρμάκων και καλλυντικών, στον εγκιβωτισμό και την παλετοποίηση.

Τα ρομπότ SCARA είναι το ακρωνύμιο που αντιπροσωπεύει το selective compliance assembly robot arm και όπως υποδηλώνει το όνομά του, είναι κατάλληλο για εργασίες συναρμολόγησης. Τα ρομπότ αυτά έχουν βραχίονες οι οποίοι συμπεριφέρονται όμοια με τον ανθρώπινο βραχίονα, καθώς οι αρθρώσεις του του επιτρέπουν και κάθετες και οριζόντιες κινήσεις. Παρόλα αυτά ο «καρπός» του δεν παρουσιάζει την ίδια ευελιξία κινήσεων καθώς έχει περιορισμένη κίνηση, χαρακτηριστικό δείγμα που του δίνει το πλεονέκτημα για πολλούς τύπους εργασιών συναρμολόγησης και συσκευασίας.

Τα καρτεσιανά ρομπότ είναι ο τύπος ρομπότ που χρησιμοποιείται συχνότερα σε πολλές βιομηχανικές εφαρμογές. Τα συγκεκριμένα ρομπότ διαθέτουν τρεις γραμμικούς άξονες που βρίσκονται σε γωνία 90 μοιρών, δηλαδή δεν μπορούν να περιστραφούν αλλά μετακινούνται σε ευθεία γραμμή. Η

άκαμπτη δομή τους, τους επιτρέπει προηγμένη ακρίβεια και επαναληψιμότητα. Η χρήση τους συχνά συναντάται σε γραμμές συναρμολόγησης για την εκτέλεση απλών κινήσεων όπως είναι η παραλαβή και η μετακίνηση φορτίων. Τα καρτεσιανά ρομπότ δεν μπορούν να φτάσουν εύκολα ή κοντα σε εμπόδια ενώ οι εκτεθειμένοι μηχανισμοί ολίσθησης, τους καθιστούν λιγότερο κατάλληλους για περιβάλλοντα με σκόνη και είναι πιο δύσκολα για αυτά να συναρμολογήσουν, να ευθυγραμμίσουν και να συντονίσουν τους διαφορετικούς άξονες σε σχέση με τα άλλα ρομπότ.

Τέλος, τα συνεργατικά ρομπότ ή cobot αποτελούν νέας γενιάς ρομπότ υποβοηθούμενα από την τεχνολογία της τεχνητής νοημοσύνης, βοηθώντας τον κλάδο της βιομηχανίας να εξελιχθεί και να διευρυνθεί δημιουργώντας πρωτότυπες γραμμές παραγωγής. Ο λόγος που τα cobos είναι τόσο καινοτόμα για τον κλάδο της βιομηχανίας είναι επειδή έχουν τη δυνατότητα να δουλεύουν συνεργατικά με τους ανθρώπους καθιστώντας τα ιδανικά για μία ανθρωποκεντρική ανάπτυξη. Επιπλέον, προσφέρουν υψηλά επίπεδα ασφάλειας, ικανά να ανταπεξέλθουν στις απαιτήσεις της βιομηχανίας (άρση βαρών, ταχύτητα, επαναληψιμότητα) ιδανική λύση για οποιαδήποτε βιομηχανική παραγωγή (Χατζή Άννα, 2021).

2.6 Κινηματική ανάλυση

Η κινηματική (Kinematics) είναι ο κλάδος της φυσικής και κατ' επέκταση της κλασικής μηχανικής που ασχολείται με την γεωμετρική κίνηση των σωμάτων χωρίς να ενδιαφέρεται για τις δυνάμεις που επενεργούν πάνω σε αυτά. Η κινηματική δηλαδή ασχολείται με την κίνηση των σωμάτων και περιγράφει την ταχύτητα, την μετατόπιση, τον χρόνο και την επιτάχυνση.

Παραδείγματος χάριν στην μηχανική είναι σύνηθες να χρησιμοποιείται η κινηματική για να καθορίζει την ταχύτητα ή την θέση ενός αντικειμένου που συνδέεται με κάποιο άλλο αντικείμενο που η ταχύτητα του ή η θέση του είναι γνωστή.

Το πρόβλημα της κινηματικής χωρίζεται σε δύο υποκατηγορίες, στο πρόβλημα της ευθείας κινηματικής (Forward kinematics) και στο πρόβλημα της αντίστροφης κινηματικής (Inverse kinematics). Το πρόβλημα της ευθείας κινηματικής χρησιμοποιείται στα ρομπότ για να υπολογίσουμε την θέση και τον προσανατολισμό του τελικού σημείου δράσης του ρομπότ, χρησιμοποιώντας τις μεταβλητές των επιμέρους αρθρώσεων του ρομπότ, ενώ το πρόβλημα της αντίστροφης κινηματικής χρησιμοποιείται για την εύρεση της θέσης και του προσανατολισμού που πρέπει να έχουν οι επιμέρους αρθρώσεις του ρομπότ σύμφωνα με την επιθυμητή θέση του τελικού σημείου δράσης του ρομπότ.

2.6.1. Κινηματική Αλυσίδα

Κινηματική αλυσίδα ονομάζεται η διαδοχική σύνδεση των συνδέσμων και των αρθρώσεων του ρομπότ. Μία άρθρωση μπορεί να είναι μια απλή άρθρωση με έναν βαθμό ελευθερίας όπως μια πρισματική ή μια περιστροφική αλλά μπορεί να είναι και πιο πολύπλοκη όπως μία σφαιρική άρθρωση με τρεις βαθμούς ελευθερίας. Στην περίπτωση αυτή η σφαιρική άρθρωση μπορεί να θεωρηθεί ώς μία αλληλουχία αρθρώσεων ενός βαθμού ελευθερίας με συνδέσμους μηδενικού μήκους μεταξύ τους.

Ένας ρομποτικός βραχίονας που αποτελείται από n αρθρώσεις θα έχει n + 1 συνδέσμους. Για την επίλυση του κινηματικού προβλήματος αριθμούμε τις αρθρώσεις του βραχίονα από το 0 έως το n και τους συνδέσμους από το 1 έως το n + 1. Ως μηδενική άρθρωση θεωρείται η βάση του βραχίονα.

Για την επίλυση του κινηματικού προβλήματος επιλύσουμε τον ομογενή μετασχηματισμό για κάθε άρθρωση του ρομπότ (Πίνακας 2.6.1.1.):

Πίνακας 2.6.1.1.

$$H_n^{n-1}=\begin{bmatrix} R_n^{n-1} & d_n^{n-1} \\ 0\times 3 & 1 \end{bmatrix}$$

Όπως παρατηρούμε από τον παραπάνω ομογενή μετασχηματισμό θα πρέπει πρώτα να επιλύσουμε πρώτα τον πίνακα περιστροφής R_n^{n-1} και και το διάνυσμα μετατόπισης d_n^{n-1}. Ο πίνακας περιστροφής καθορίζεται από ποιον άξονα περιστρέφεται η άρθρωση του ρομπότ και ανάλογα τον άξονα που περιστρέφεται έχει την παρακάτω μορφή. Στην περίπτωση που η περιστροφή ενός άξονα είναι συνδυαστική τότε οι πίνακες περιστροφής συνδυάζονται αναλόγως.

Πίνακας 2.6.1.2.

$$Rx=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Πίνακας 2.6.1.3.

$$R_y=\begin{bmatrix} \cos\theta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\vartheta \end{bmatrix}$$

Πίνακας 2.6.1.4.

$$R_z=\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Στην συνέχεια θα πρέπει να υπολογιστεί το διάνυσμα μετατόπισης της άρθρωσης n ως προς την άρθρωση n – 1. Το διάνυσμα μετατόπισης αποτελείται από μόνο μία στήλη όπου η κάθε γραμμή αντιπροσωπεύει την μετατόπιση x, y, z αντίστοιχα της άρθρωσης n ως προς την άρθρωση n – 1.

Πίνακας 2.6.1.5.

$$d_n^{n-1} = \begin{bmatrix} x_n^{n-1} \\ y_n^{n-1} \\ z_n^{n-1} \end{bmatrix}$$

Σύμφωνα με τα παραπάνω ο τελικός ομογενής μετασχηματισμός για έναν ρομποτικό βραχίονα είναι ο εξής:

Πίνακας 2.6.1.6.

$$H_n^0 = \begin{bmatrix} R_n^0 & d_n^0 \\ 3 \times 0 & 1 \end{bmatrix}$$

2.6.2. Denavit – Hartenberg

Η μέθοδος Denavit – Hartenberg είναι η πιο διαδεδομένη μέθοδος επίλυσης του κινηματικού προβλήματος. Η μέθοδος αυτή είναι ένα σύνολο κανόνων η οποία ορίζει το πως πρέπει να οριστούν τα πλαίσια αναφοράς που σχετίζονται με κάθε σύνδεσμο. Ξεκινώντας από την βάση του ρομπότ κάθε πλαίσιο αναφοράς μπορεί να υπολογιστεί από 4 παραμέτρους γνωστές ως Denavit – Hartenberg παράμετροι. Είναι μία μέθοδος που μας επιτρέπει να χρησιμοποιήσουμε τις λιγότερες δυνατές παραμέτρους για την επίλυση του κινηματικού προβλήματος και χρησιμοποιείται ευρέος.

Ωστόσο, υπάρχουν κάποιοι περιορισμοί για την χρήση της μεθόδου αυτής. Ένας από τους πιο βασικούς περιορισμούς είναι ότι η σύνδεση των αρθρώσεων του ρομπότ θα πρέπει να είναι σειριακή ανεξαρτήτως από τους βαθμούς ελευθερίας του ρομπότ, θα πρέπει να είναι γνωστές οι διαστάσεις των συνδέσεων του ρομπότ, η κατεύθυνση της μετατόπισης των αρθρώσεων του ρομπότ καθώς και το πλαίσιο αναφοράς της βάσης του ρομπότ.

Επιπλέον κανόνες για την εύρεση των πλαισίων αναφοράς της κάθε άρθρωσης είναι οι παρακάτω:

I Ο z άξονας θα πρέπει να τοποθετηθεί κατά μήκος του άξονα περιστροφής.

II Ο x άξονας θα πρέπει να είναι συγγραμικός με την κοινή νοητή γραμμή που είναι κάθετη και με τις δύο αρθρώσεις.

III Ο y άξονας βρίσκεται με τον κανόνα του δεξιού χεριού.

Γενικά η μέθοδος Denavit – Hartenberg αποτελείται από 4 παραμέτρους, οι οποίοι είναι:

I d: είναι η απόσταση από την βάση του προηγούμενου z άξονα έως την νοητή κάθετη γραμμή των δύο αρθρώσεων που μελετούνται.

II θ: είναι η γωνία του προηγούμενου z άξονα σε σχέση με τον z άξονα που μελετάμε μετατοπίζοντας τον προηγούμενο x άξονα κατά τον καινούργιο.

III r: είναι η απόσταση κατά των x άξονα που μελετάμε μεταξύ του προηγούμενου z άξονα και του z αξονα του πλαισίου αναφοράς που μελετάμε.

IV α: είναι η γωνία που δημιουργείται μετατοπίζοντας των προηγούμενο z άξονα ως προς των z άξονα του πλαισίου που μελετάμε.

Ο πίνακας ομογενούς μετασχηματισμού της άρθρωσης που μελετάτε υπολογίζεται ως εξής:

Πίνακας 2.6.2.1.

$$ {}^{n-1}T_n = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n)\cdot\cos(\alpha_n) & \sin(\theta_n)\cdot\sin(\alpha_n) & r_n\cdot\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n)\cdot\cos(\alpha_n) & -\cos(\theta_n)\cdot\sin(\alpha_n) & r_n\cdot\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

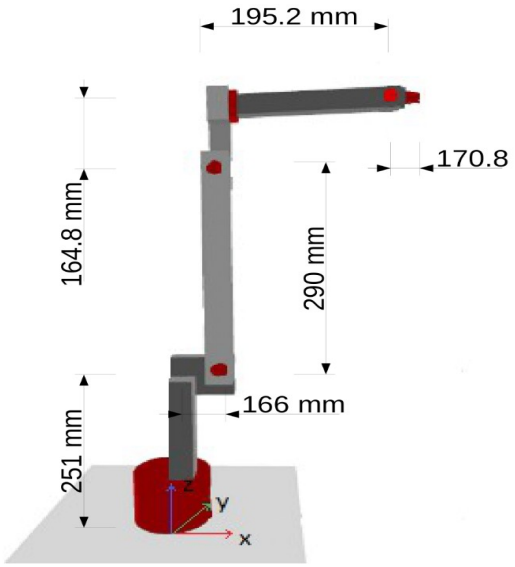
όπου το R είναι ένας υποπίνακας 3x3 ο οποίος δηλώνει την περιστροφή του πλαισίου αναφοράς στο σύστημα συντεταγμένων ενώ ο υποπίνακας T δηλώνει την μετατόπιση στο σύστημα συντεταγμένων του πλαισίου αναφοράς σε σχέση με το προηγούμενο, ενώ ο πίνακας ομογενούς μετασχηματισμού του τελικό σημείου δράσης του ρομπότ υπολογίζεται ως εξής:

Πίνακας 2.6.2.2.

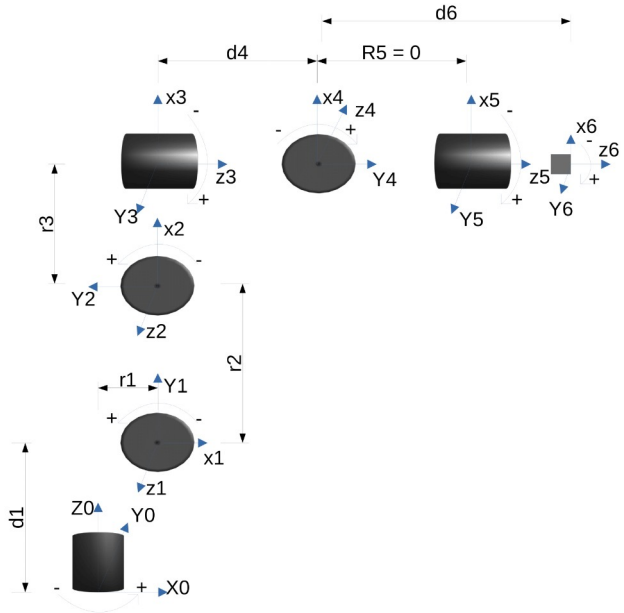
$$ {}^{0}T_6 = {}^{0}T_1 * {}^{1}T_2 * {}^{2}T_3 * {}^{3}T_4 * {}^{4}T_5 * {}^{5}T_6 $$

2.6.3. Παράμετροι Denavit – Hertenberg του ρομπότ

Ο ρομποτικός βραχίονας του οποίου θα μελετήσουμε την κινηματική είναι ένα ρομπότ έξι βαθμών ελευθερίας, το οποίο σημαίνει ότι θα έχουμε έξι πλαίσια αναφοράς μαζί με το σημείο αναφοράς του τελικού σημείου του ρομπότ να μελετήσουμε ώστε να καταλήξουμε στον ομογενή πίνακα του τελικού σημείου δράσης του ρομπότ. Στις επόμενες εικόνας φαίνεται το ρομπότ που θα μελετήσουμε (Εικόνα 2.6.3.1.) καθώς επίσης και η κινηματική του ανάλυση (Εικόνα 2.6.3.2.).



Εικόνα 2.6.3.1. : θέση και διαστάσεις ρομποτικού βραχίονα

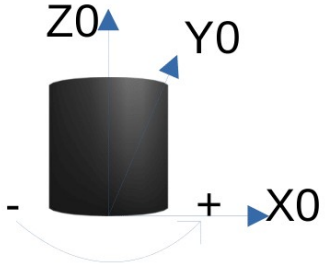


Εικόνα 2.6.3.2. Κινηματική ανάλυση ρομποτικού βραχίονα

Για την επίλυση του κινηματικού προβλήματος ξεκινάμε πάντα από την βάση του ρομπότ. Μελετάμε κάθε n άξονα σε σχέση με τον n – 1 άξονα και επιλύουμε έτσι ώστε να εντοπίσουμε της παραμέτρους του Denavit – Hartenberg για κάθε άξονα ωσότου έχουμε τον πλήρη πίνακα των παραμέτρων για όλους τους άξονες του ρομπότ. Παράλληλα δημιουργούμε τον πίνακα ομογενούς μετασχηματισμού για κάθε άξονα που μελετάμε.

2.6.3.1. Η βάση του ρομποτικού βραχίονα/ Σύνδεσμος 1 (link 1)

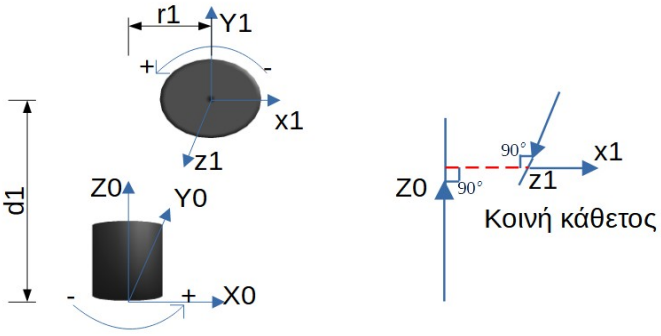
Ξεκινώντας από την βάση (Εικόνα 2.6.3.1.1.) του ρομπότ ορίζουμε το z_0 κατά μήκος του άξονα περιστροφής και με κατεύθυνση προς τα πάνω. Στην συνέχεια ορίζουμε το x_0 με κατεύθυνση προς τα δεξιά. Λόγω ότι είναι ο πρώτος άξονας το πως θα ορίσουμε το x_0 δεν έχει κάποιον περιορισμό αρκεί να είναι κάθετο ως προς το z_0 και να ικανοποιείται ο κανόνας του δεξιού χεριού.



Εικόνα 2.6.3.1.1

Έπειτα συνεχίζουμε μελετώντας τον πρώτο σύνδεσμο (link 1) και ορίζουμε το z_1 με κατεύθυνση προς τα έξω καθώς ο άξονας κινείται προς τα θετικά κινούμενος αριστερόστροφα. Στην συνέχεια βρίσκουμε την κοινή κάθετο των δύο αξόνων (z_0 και z_1) όπως φαίνεται στην παρακάτω εικόνα. Για την εύρεση της κατεύθυνσης του x_1 θα πρέπει να ικανοποιούνται οι εξής προϋποθέσεις, πρώτον το x_1 να είναι συγγραμικό προς την κοινή κάθετο και με κατεύθυνση που να απομακρύνεται από το z_0 και δεύτερον να τέμνει και το z_0 και το z_1. Τέλος το y_1 ακολουθεί τον κανόνα του δεξιού χεριού.

Από την θεωρία γνωρίζουμε ότι το d_n ορίζεται ως η απόσταση του z_{n-1} επεκτεινόμενο έως την κοινή κάθετη των δύο αξόνων, οπότε το d_1 ορίζεται από την βάση του z_0 έως το z1 (Εικόνα 2.6.3.1.2.) και το r_n είναι η απόσταση του z_{n-1} και του z_n κατά τον x. Το a_n ορίζεται ως η γωνία που δημιουργείται όταν περιστρέφουμε το z_{n-1} γύρω από τον x_n έως ότου επιτύχουμε την ίδια κατεύθυνση με το z_n. Το θ_n ορίζεται ως η γωνία που δημιουργείται όταν περιστρέφουμε το x_{n-1} γύρω από το z_{n-1} έως ότου επιτύχουμε την ίδια κατεύθυνση με το x_n.



Εικόνα 2.6.3.1.2.

Σύμφωνα με όλα τα παραπάνω έχουμε τις παρακάτω Denavit – Hartenberg παραμέτρους για τον πρώτο σύνδεσμο (link 1).

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$

Πίνακας 2.6.3.1.1.

Στην συνέχεια ορίζουμε τον πίνακα ομογενούς μετασχηματισμού ${}^0T_1$ όπως φαίνεται παρακάτω:

Πίνακας 2.6.3.1.2.

$$ {}^0T_1=\begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1)\cdot\cos(\alpha_1) & \sin(\theta_1)\cdot\sin(\alpha_1) & r_1\cdot\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1)\cdot\cos(\alpha_1) & -\cos(\theta_1)\cdot\sin(\alpha_1) & r_1\cdot\sin(\theta_1) \\ 0 & \sin(\alpha_1) & \cos(\alpha_1) & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

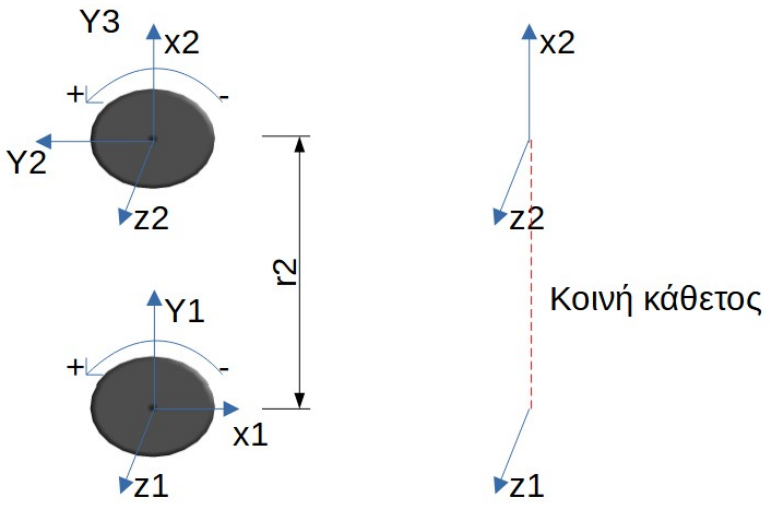
Πίνακας 2.6.3.1.3.

$$ {}^0T_1=\begin{bmatrix} 1 & 0 & 0 & 0.166 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.251 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.6.3.2. Σύνδεσμος 2 (link 2)

Μελετώντας τον επόμενο σύνδεσμο (link 2) ορίζουμε το z_2 με κατεύθυνση προς τα έξω καθώς ο άξονας έχει κατεύθυνση προς τα αριστερά. Εφόσον το z_1 και το z_2 έχουν κοινό προσανατολισμό η κοινή κάθετος τους μπορεί να ορισθεί οπουδήποτε κατ μήκος των αξόνων ωστόσο προτιμάμε να το ορίζουμε στην βάση τους για ευκολία. Ως εκ τούτου το x_2 ορίζεται με κατεύθυνση προς τα πάνω καθώς θα πρέπει να είναι συγγραμμικό με την κοινή κάθετο και κατεύθυνση απομακρυνόμενη από το z_1. Το y_2 ορίζεται προς τα αριστερά σύμφωνα με τον κανόνα του δεξιού χεριού.

Εφόσον η κοινή κάθετος έχει ορισθεί στην βάση των αξόνων και ως d_n ορίζεται η απόσταση του z_{n-1} επεκτεινόμενο έως την κοινή κάθετη των δύο αξόνων θεωρούμε το d_2 ίσο με το μηδέν. Το θ_2 το οποίο ορίζεται ως η γωνία του x_1 γύρω από το z_1 έως ότου συναντήσει το x_2 θα έχει τιμή ίση με 90^o. Το r_2 είναι η απόσταση του z_1 και του z_2 κατά τον x_2. Τέλος το α_2 επειδή το z_1 και το z_2 έχουν ίδια κατεύθυνση θα είναι ίσο με το μηδέν.



Εικόνα 2.6.3.2.1.

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$
2	$d_2=0\,m$	$\theta_2=90^o$	$r_2=0.290\,m$	$\alpha_2=0^o$

Πίνακας 2.6.3.2.1.

Ο πίνακας ομογενούς μετασχηματισμού ${}^{1}T_2$ ορίζεται ως εξής:

Πίνακας 2.6.3.2.2.

$$ {}^{1}T_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2)\cdot\cos(\alpha_2) & \sin(\theta_2)\cdot\sin(\alpha_2) & r_2\cdot\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2)\cdot\cos(\alpha_2) & -\cos(\theta_2)\cdot\sin(\alpha_2) & r_2\cdot\sin(\theta_2) \\ 0 & \sin(\alpha_2) & \cos(\alpha_2) & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

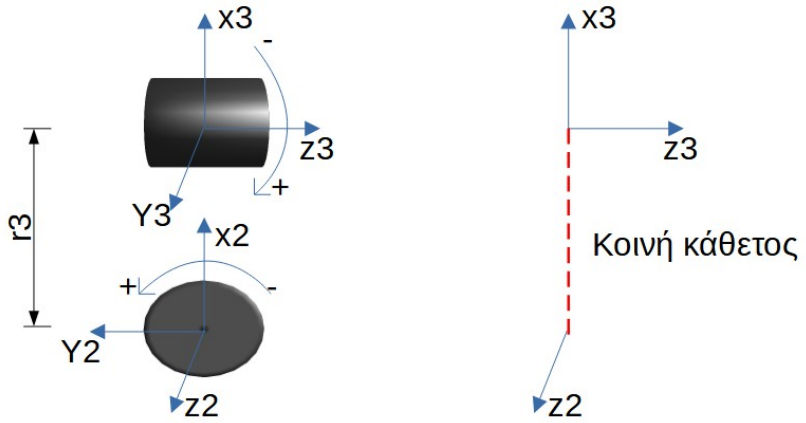
Πίνακας 2.6.3.2.3.

$$ {}^{1}T_2 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.290 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.6.3.3. Σύνδεσμος 3 (link 3)

Για τον τρίτο σύνδεσμο (link 3) ορίζουμε το z_3 με κατεύθυνση προς τα δεξιά καθώς ο άξονας έχει κατεύθυνση προς τα δεξιά κινούμενος θετικά. Η κοινή κάθετος των z_2 και z_3 ορίζεται στην βάση των αξόνων όπως φαίνεται στην εικόνα και ως εκ τούτου το x_3 θα έχει κατεύθυνση προς τα πάνω. Από τον κανόνα του δεξιού χεριού προκύπτει το y_3.

Εφόσον η κοινή κάθετος ορίζεται στην βάση των z_2 και z_3 το d_3 θα είναι ίσο με το μηδέν. Το θ_3 θα είναι επίσης ίσο με το μηδέν καθώς το x_2 και το x_3 έχουν κοινό προσανατολισμό. Το r_3 ορίζεται ως η απόσταση του z_2 και του z_3 κατά τον x_3. Τέλος, το α_3 το οποίο προκύπτει περιστρέφοντας το z_2 γύρω από x_3 έως ότου το z_2 και το z_3 έχουν την ίδια κατεύθυνση θα είναι ίσο με 90^o.



Εικόνα 2.6.3.3.1.

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$
2	$d_2=0\,m$	$\theta_2=90^o$	$r_2=0.290\,m$	$\alpha_2=0^o$
3	$d_3=0\,m$	$\theta_3=90^o$	$r_3=0.1648\,m$	$\alpha_3=90^o$

Πίνακας 2.6.3.3.1.

Ο πίνακας ομογενούς μετασχηματισμού ${}^{2}T_{3}$ ορίζεται ως εξής:

Πίνακας 2.6.3.3.2.

$$ {}^{2}T_{3}=\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3)\cdot\cos(\alpha_3) & \sin(\theta_3)\cdot\sin(\alpha_3) & r_3\cdot\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3)\cdot\cos(\alpha_3) & -\cos(\theta_3)\cdot\sin(\alpha_3) & r_3\cdot\sin(\theta_3) \\ 0 & \sin(\alpha_3) & \cos(\alpha_3) & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

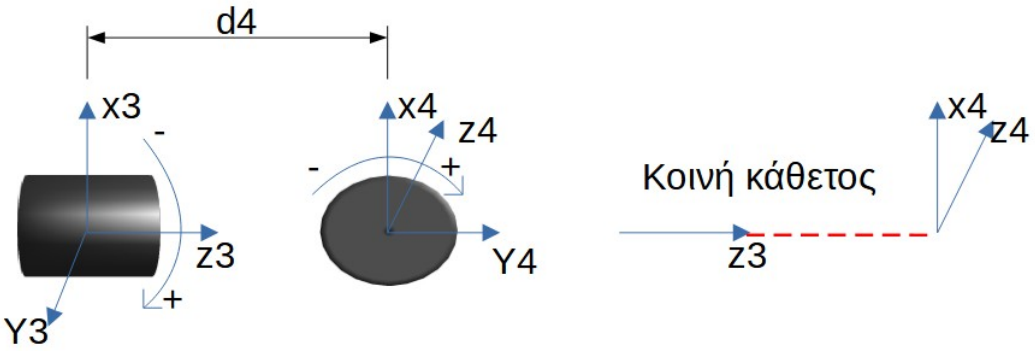
Πίνακας 2.6.3.3.3.

$$ {}^{2}T_{3}=\begin{bmatrix} 1 & 0 & 0 & 0.1648 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.6.3.4. Σύνδεσμος 4 (link 4)

Στον τέταρτο σύνδεσμο (link 4) ορίζουμε το z_4 με κατεύθυνση προς τα πίσω καθώς ο άξονας κινείται θετικά κινούμενος δεξιόστροφα. Το x_4 ορίζεται με κατεύθυνση προς τα πάνω όπως φαίνεται στην εικόνα και το y_4 προκύπτει από τον κανόνα του δεξιού χεριού.

Σύμφωνα με την θεωρία που έχουμε αναλύσει το d_4 είναι η απόσταση από την βάση του z_3 έως την βάση του z_4. Το θ_4 προκύπτει ίσο με το μηδέν καθώς το x_3 και το x_4 έχουν κοινό προσανατολισμό. Επίσης το r_4 ισούται με το μηδέν καθώς η απόσταση του z_3 από το z_4 κατά το x_4 είναι ίσο με το μηδέν. Το α_4 θα είναι ίσο με 90^o.



Εικόνα 2.6.3.4.1.

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$
2	$d_2=0\,m$	$\theta_2=90^o$	$r_2=0.290\,m$	$\alpha_2=0^o$
3	$d_3=0\,m$	$\theta_3=90^o$	$r_3=0.1648\,m$	$\alpha_3=90^o$
4	$d_4=0.1952\,m$	$\theta_4=0^o$	$r_4=0\,m$	$\alpha_4=90^o$

Πίνακας 2.6.3.4.1.

Ο πίνακας ομογενούς μετασχηματισμού ${}^{3}T_4$ ορίζεται ως εξής:

Πίνακας 2.6.3.4.2.

$$ {}^{3}T_4 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4)\cdot\cos(\alpha_4) & \sin(\theta_4)\cdot\sin(\alpha_4) & r_4\cdot\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4)\cdot\cos(\alpha_4) & -\cos(\theta_4)\cdot\sin(\alpha_4) & r_4\cdot\sin(\theta_4) \\ 0 & \sin(\alpha_4) & \cos(\alpha_4) & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

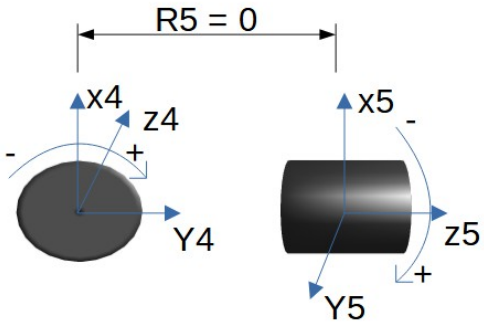
Πίνακας 2.6.3.4.3.

$$ {}^{3}T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.1952 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

2.6.3.5. Σύνδεσμος 5 (link 5)

Στον πέμπτο σύνδεσμο (link 5) το z_5 θα έχει κατεύθυνση προς τα δεξιά καθώς η άρθρωση κινείται στα θετικά δεξιόστροφα. Το x_5 θα έχει κατεύθυνση προς τα πάνω και από τον κανόνα του δεξιού χεριού το y_5 θα έχει κατεύθυνση προς τα έξω.

Από την εικόνα φαίνεται ότι το r_5 ισούται με το μηδέν και ως εκ τούτου η πραγματική απόσταση τον δύο αρθρώσεων είναι μηδενική απλά λόγω κατανόησης της κινηματικής έχουν σχεδιαστεί σε απόσταση. Οπότε και το d_5 ισούται με το μηδέν. Το θ_5 καθώς το x_4 και το x_5 έχουν κοινό προσανατολισμό θα είναι επίσης ίσο με το μηδέν. Τέλος το α_5 θα είναι ίσο με -90°.



Εικόνα 2.6.3.5.1.

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$
2	$d_2=0\,m$	$\theta_2=90^o$	$r_2=0.290\,m$	$\alpha_2=0^o$
3	$d_3=0\,m$	$\theta_3=90^o$	$r_3=0.1648\,m$	$\alpha_3=90^o$
4	$d_4=0.1952\,m$	$\theta_4=0^o$	$r_4=0\,m$	$\alpha_4=90^o$
5	$d_5=0\,m$	$\theta_5=0^o$	$r_5=0\,m$	$\alpha_5=-90^o$

Πίνακας 2.6.3.5.1.

Ο πίνακας ομογενούς μετασχηματισμού ${}^4T_5$ ορίζεται ως εξής:

Πίνακας 2.6.3.5.2.

$$ {}^4T_5=\begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5)\cdot\cos(\alpha_5) & \sin(\theta_5)\cdot\sin(\alpha_5) & r_5\cdot\cos(\theta_5) \\ \sin(\theta_5) & \cos(\theta_5)\cdot\cos(\alpha_5) & -\cos(\theta_5)\cdot\sin(\alpha_5) & r_5\cdot\sin(\theta_5) \\ 0 & \sin(\alpha_5) & \cos(\alpha_5) & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

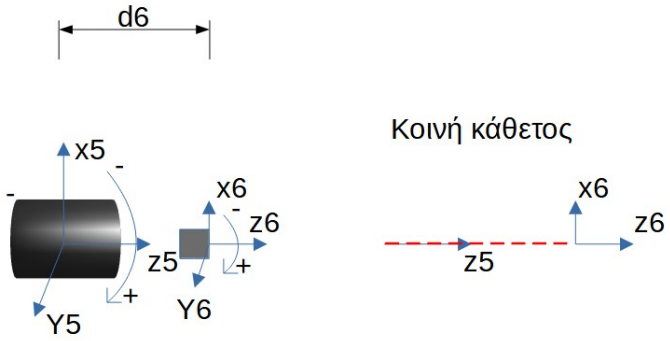
Πίνακας 2.6.3.5.3.

$$ {}^4T_5=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.6.3.6. Σύνδεσμος 6 (link 6)

Τέλος, ο έκτος σύνδεσμος (link 6) αφορά την μελέτη του κινηματικού στο τελικό σημείο του ρομπότ. Για τον λόγο αυτό επειδή η κίνηση είναι εξαρτώμενη από την τελευταία άρθρωση του ρομπότ το z_6, το x_6 και το y_6 θα έχουν ακριβώς την ίδια κατεύθυνση με την τελευταία άρθρωση.

Ως εκ τούτου, το θ_6, το r_6 και το α_6 θα ισούνται με μηδέν ενώ το d_6 θα ισούται με την απόσταση του z_5 από το z_6.



Εικόνα 2.6.3.6.1.

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^o$	$r_1=0.166\,m$	$\alpha_1=90^o$
2	$d_2=0\,m$	$\theta_2=90^o$	$r_2=0.290\,m$	$\alpha_2=0^o$
3	$d_3=0\,m$	$\theta_3=90^o$	$r_3=0.1648\,m$	$\alpha_3=90^o$
4	$d_4=0.1952\,m$	$\theta_4=0^o$	$r_4=0\,m$	$\alpha_4=90^o$
5	$d_5=0\,m$	$\theta_5=0^o$	$r_5=0\,m$	$\alpha_5=-90^o$
6	$d_6=0.1708\,m$	$\theta_6=0^o$	$r_6=0\,m$	$\alpha_6=0^o$

Πίνακας 2.6.3.6.1.

Ο πίνακας ομογενούς μετασχηματισμού ${}^{5}T_{6}$ ορίζεται ως εξής:

Πίνακας 2.6.3.6.2.

$$ {}^{5}T_{6}=\begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6)\cdot\cos(\alpha_6) & \sin(\theta_6)\cdot\sin(\alpha_6) & r_6\cdot\cos(\theta_6) \\ \sin(\theta_6) & \cos(\theta_6)\cdot\cos(\alpha_6) & -\cos(\theta_6)\cdot\sin(\alpha_6) & r_6\cdot\sin(\theta_6) \\ 0 & \sin(\alpha_6) & \cos(\alpha_6) & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Πίνακας 2.6.3.6.3.

$$ {}^{5}T_{6}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.1708 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} & R & & T \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.6.3.7. Ομογενής μετασχηματισμός από τη βάση έως το τελικό σημείο του ρομποτικού βραχίονα

Για να βρούμε τον πίνακα του ομογενούς μετασχηματισμού από την βάση του ρομπότ έως το τελικό σημείο του ρομπότ αρκεί να πολλαπλασιάσουμε τους πίνακες ομογενούς μετασχηματισμού των επιμέρους συνδέσμων:

Πίνακας 2.6.3.7.1.

$$^{0}T_{6} = {}^{0}T_{1} * {}^{1}T_{2} * {}^{2}T_{3} * {}^{3}T_{4} * {}^{4}T_{5} * {}^{5}T_{6}$$

Λύνοντας τον παραπάνω πολλαπλασιασμό των ομογενών πινάκων καταλήγουμε στο παρακάτω αποτέλεσμα:

Πίνακας 2.6.3.7.2.

$$^{0}T_{6} = \begin{bmatrix} 0 & 0 & 1 & 0.532 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.7058 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Όπου ο υποπίνακας R δηλώνει την περιστροφή του τελικού σημείου αναφοράς στο σύστημα συντεταγμένων ενώ ο υποπίνακας T δηλώνει την μετατόπιση. Με την μέθοδο αυτή μπορούμε να βρούμε την θέση του τελικού σημείου του ρομπότ στο σύστημα συντεταγμένων για όποιες γωνίες θ_n επιθυμούμε.

ΚΕΦΑΛΑΙΟ 3ο

Προγραμματιστικό μέρος

Για την ανάπτυξη του κώδικα θα χρησιμοποιήσουμε το CODESYS το οποίο είναι ένα περιβάλλον ανάπτυξης προγραμματισμού εφαρμογών ελεγκτή σύμφωνα με το διεθνές πρότυπο IEC 61131 – 3. Το CODESYS υποστηρίζει τις εξής γλώσσες προγραμματισμού, ST (Structured Text), FBD (Function Block Diagram), SFC (Sequential Function Chart), CFC (Continuous Function Chart) και LD (Ladder Logic Program). Κατά την ανάπτυξη του προγράμματος θα χρησημοποιηθούν οι, ST και CFC, γλώσσες προγραμματισμού. Για την ανάπτυξη του SoftMotion θα χρησιμοποιηθεί κυρίως η CFC γλώσσα προγραμματισμού ενώ για οτιδήποτε άλλο χρειαστεί θα χρησιμοποιηθεί η ST γλώσσα προγραμματισμού.

3.1. SoftMotion

Το SoftMotion είναι μια βιβλιοθήκη η οποία ενσωματώνεται στο CODESYS και χρησιμοποιείται για τον έλεγχο κίνησης οποιουδήποτε κινηματικού συστήματος. Πέρα από της απλές εφαρμογές που μπορεί να έχει το SoftMotion υποστηρίζει επίσης και πιο σύνθετα κινηματικά προβλήματα όπως η επίλυση ενός Tripod Robot ή ενός 6 Axis Robot και πολλές ακόμη εφαρμογές. Το SoftMotion μπορεί να έχει την πιο απλή εφαρμογή όπως η κίνηση ενός Servo κινητήρα έως τα πιο πολύπλοκα κινηματικά προβλήματα καθώς σου δίνει την δυνατότητα το CODESYS να δημιουργήσεις την δικιά σου βιβλιοθήκη κινηματικής.

Αποτελείται από πολλά μπλοκ λειτουργιών τα οποία καλύπτουν όλες τις απαιτήσεις για την κίνηση ενός κινητήρα, ενδεικτικά, μερικά από τα πιο βασικά για την λειτουργία ενός Servo κινητήρα, είναι τα παρακάτω:

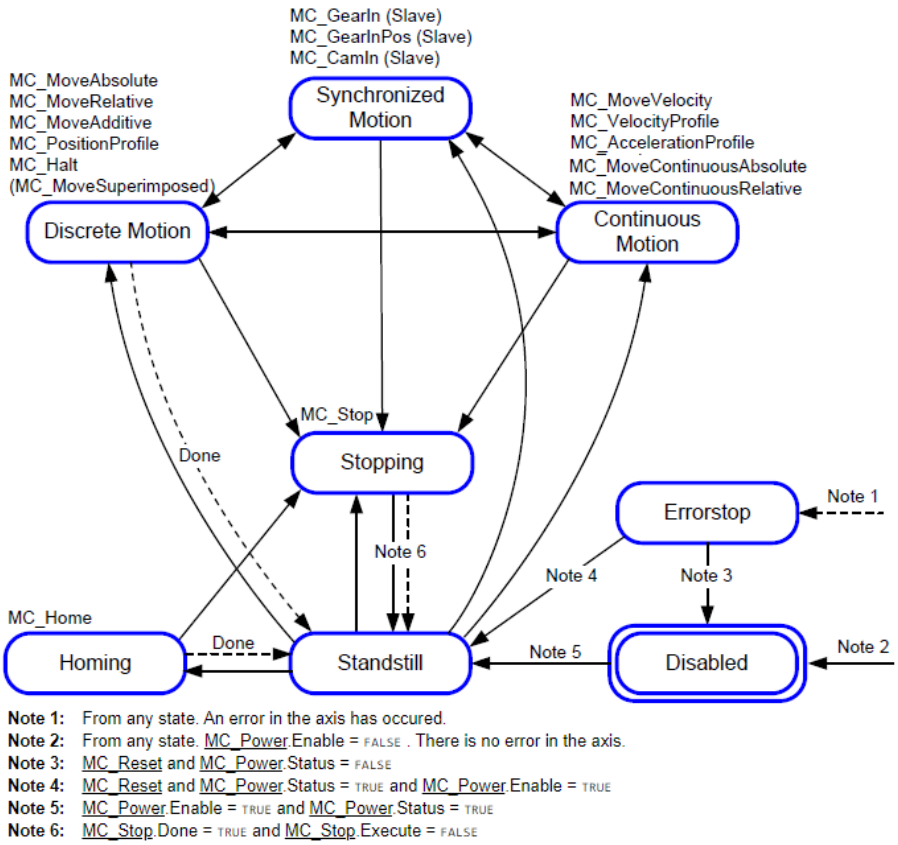
- MC_Power: Θέτει σε λειτουργία On ή Off το μοτέρ.
- MC_Reset: Επαναφέρει το μοτέρ σε σωστή λειτουργία μετά από κατάσταση σφάλματος
- MC_Stop: Θέτει το μοτέρ σε κατάσταση στοπ ακυρώνοντας οποιαδήποτε κίνηση εκτελούσε το μοτέρ ή είχε σκοπό να εκτελέσει. Καμία κίνηση δεν μπορεί να εκτελεστεί από το μοτέρ αν δεν βγει από την κατάσταση στοπ.
- MC_Jog: Επιτρέπει την χειροκίνητη κίνηση του μοτέρ.
- MC_Home: Θέτει το μοτέρ είτε στην αρχική του θέση είτε στην αρχική θέση που εμείς έχουμε ορίσει.
- MC_MoveAbsolute: Κινεί το μοτέρ σε απόλυτη θέση. Το μοτέρ όταν φτάσει στην θέση την οποία του έχουμε ορίσει θα σταματήσει και θα μπει σε κατάσταση αναμονής.
- MC_MoveRelative: Κινεί το μοτέρ σε σχετική θέση. Το μοτέρ ανεξαρτήτως από την θέση που έχει την στιγμή που του δώσουμε την εντολή θα κινηθεί την απόσταση που του έχουμε ορίσει.
- SMC_SetMovementType: Ορίζει αν ένα μοτέρ κατά την κίνηση του θα κάνει γραμμική κίνηση (linear movement) ή κυκλική κίνηση (modulo movement).
- SMC_SetSoftwareLimits: Οριοθετεί το ελάχιστο και το μέγιστο στην κίνηση ενός μοτέρ είτε αυτή είναι γραμμική είτε κυκλική.
- και πολλά ακόμα ώστε να μπορεί να ικανοποιηθεί οποιαδήποτε παράμετρος

Επιπλέον, εκτός από τα παραπάνω μπλοκ τα οποία αφορούν ξεχωριστά κάθε έναν κινητήρα υπάρχουν και μπλοκ τα οποία αφορούν ολόκληρα γκρουπ αφού έχουμε ορίσει την κινηματική του γκρουπ.

Για την επίλυση του κινηματικού προβλήματος του ρομπότ έξι αξόνων το ορίζουμε στο γκρουπ των αξόνων. Στην συνέχεια θα χρησιμοποιήσουμε τα παρακάτω μπλοκ για να επιτύχουμε την κίνηση του στον χώρο.

- MC_GroupPower: Θέτει σε λειτουργία On ή Off ολόκληρο το γκρουπ των Servo κινητήρων
- MC_GroupEnable: Θέτει ολόκληρο το γκρουπ σε κατάσταση ακινησίας (Standstill) αφού το έχουμε θέσει πρώτα σε κατάσταση λειτουργίας (On).
- MC_GroupDisable: Απενεργοποιεί το γκρουπ των κινητήρων του κινηματικού (Off).
- MC_GroupReset: Θέτει σε σωστή λειτουργία το γκρουπ των αξόνων αφού έχει εμφανιστεί σφάλμα.
- MC_GroupStop: Σταματάει την κίνηση του γκρουπ των αξόνων.
- MC_Jog: Επιτρέπει την χειροκίνητη κίνηση των αξόνων.
- MC_MoveDirectAbsolute: Ορίζει την τελική θέση που θέλουμε να φτάσει το τελικό σημείο του ρομπότ. Η κίνηση που εκτελείται με την χρήση του μπλοκ αυτού είναι απόλυτη και επιλέγει την βέλτιστη διαδρομή.
- MC_MoveLinearAbsolute: Ορίζει την τελική θέση που θέλουμε να φτάσει το τελικό σημείο του ρομπότ. Ωστόσο η κίνηση αυτή ενώ είναι απόλυτη είναι και γραμμική. Αυτό σημαίνει ότι το ρομπότ θα πάει από το ένα σημείο στο άλλο γραμμικά και όχι επιλέγοντας την βέλτιστη διαδρομή.
- MC_GroupReadActualPosition: Διαβάζει την θέση του τελικού σημείου του ρομπότ.
- SMC_COORD_SYSTEM: Ορίζει το σύστημα συντεταγμένων που κινείται το ρομπότ.
- MC_SetCoordinateTransform: Ορίζει νέο σύστημα συντεταγμένων σχετικό με τις συντεταγμένες του κόσμου που βρίσκεται το ρομπότ.
- SMC_Interpolator: Μεταφράζει μία συνεχή διαδρομή σε διακριτά σημεία. Συνήθως συνδιάζεται με το SMC_NCInterpreter το οποίο μετατρέπει τον G κώδικα σε συνεχή διαδρομή.
- SMC_ReadNCFile2: Διαβάζει τα αρχεία που είναι αποθηκεύμενα στο PLC. Στην προκειμένη διαβάζουμε ορισμένα αρχεία G κώδικα ώστε να μπορούμε να τα φορτώσουμε στον ρομποτικό βραχίονα.

Σύμφωνα με το SoftMotion οι πιθανές καταστάσεις ενός άξονα είναι οι παρακάτω (Εικόνα 3.1.1.) και υπάρχουν συγκεκριμένες συνθήκες για να μεταπηδήσει από την μία κατάσταση στην άλλη. Πάντα ξεκινάει από την κατάσταση απενεργοποιημένος (Disabled/ power_Off) και καταλήγει στην κατάσταση ακινητοποιημένος (Standstill) αφού πρώτα έχει ενεργοποιηθεί (MC_Power). Από την κατάσταση της ακινητοποίησης μπορεί να μεταπηδήσει σε πολλές διαφορετικές καταστάσεις ανάλογα το μπλοκ κινήσεων που θα καλέσουμε. Για να καταλήξουμε στην κατάσταση της αρχικοποίησης (Homing) αρκεί να καλέσουμε το μπλοκ MC_Home το οποίο αρχικοποιεί τον άξονα. Για να καταλήξουμε σε διακριτή κίνηση (Discrete Motion) αρκεί να καλέσουμε τα μπλοκ MC_MoveAbsolute, MC_MoveRelative κ.α., μετά το τέλος της κίνησης η κατάσταση του άξονα επιστρέφει στην ακινητοποίηση (Standstill) ενώ μπορεί να καταλήξει στην κατάσταση στοπ (Stopping) αν καλέσουμε το μπλοκ MC_Stop όπου μετά το τέλος του καταλήγει πάλι στην κατάσταση της ακινητοποίησης. Το ίδιο ισχύει και για τις καταστάσεις συγχρονισμένη κίνηση (Sychronized Motion) και συνεχής κίνηση (Continuous Motion). Για κάθε κίνηση θα πρέπει να εκτελεστούν τα κατάλληλα μπλοκ. Τέλος από οποιαδήποτε κατάσταση μπορεί ο άξονας να καταλήξει στην κατάσταση του σφάλματος (Errorstop).



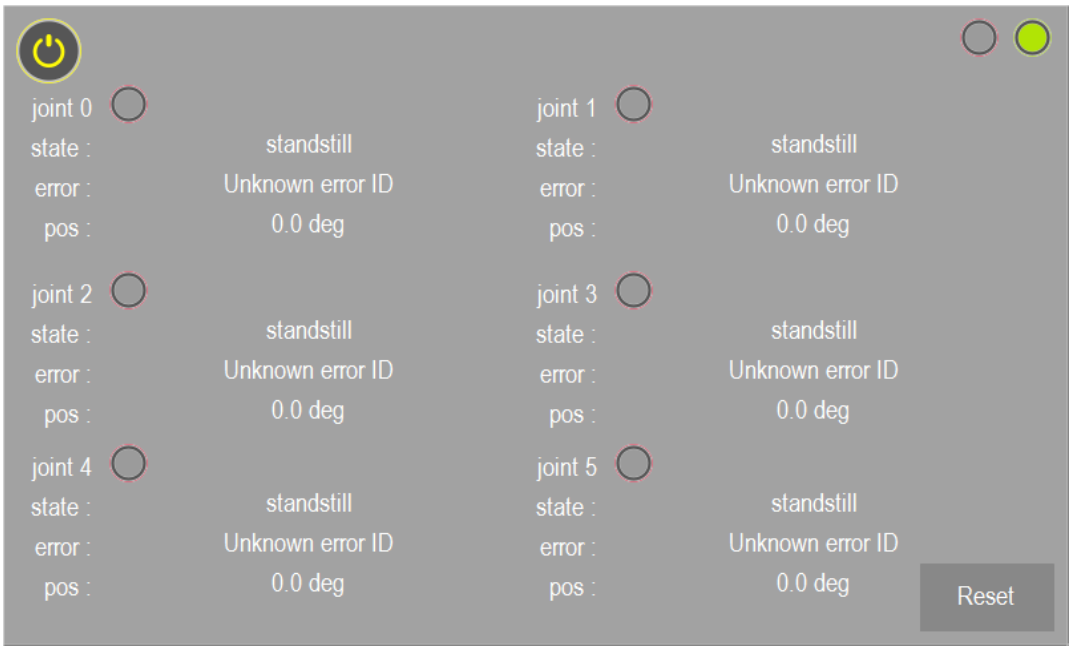
Εικόνα 3.1.1. Καταστάσεις άξονα

3.2. Human Machine Interface

Το Human Machine Interface ή αλλιώς HMI είναι η οθόνη που έχει ο χειριστής έτσι ώστε να μπορεί να χειρίζεται το εκάστοτε σύστημα. Είναι ο τρόπος για να μπορεί ο χειριστής να επικοινωνεί με την μηχανή. Στο CODESYS υπάρχει η επιλογή να δημιουργήσεις το HMI είτε κατευθείαν στο PLC αν σε αυτό υπάρχει και οθόνη είτε στο δίκτυο σε σελίδα .htm με διεύθυνση http://localhost:8080/webvisu.htm, όπου localhost θέτουμε την IP του PLC.

Στην συνέχεια παρουσιάζεται το HMI που δημιουργήθηκε για τον έλεγχο του ρομπότ έξι αξόνων που έχουμε αναλύσει στο 2° κεφάλαιο. Συνολικά έχουν δημιουργηθεί πέντε καρτέλες για τον συνολικό έλεγχο του ρομπότ. Στην πρώτη καρτέλα (Εικόνα 3.2.1.) υπάρχει ένας διακόπτης για την ενεργοποίηση του συστήματος (Εικόνα 3.2.2.) καθώς και οπτική απεικόνιση που μας ενημερώνει για το αν το σύστημα μας είναι ενεργοποιημένο σωστά ή όχι, αν το σύστημα δεν έχει ενεργοποιηθεί τότε τίποτα δεν μπορεί να δουλέψει. Επίσης υπάρχουν όλα τα βασικά στοιχεία για οπτικό έλεγχο των αξόνων του ρομπότ, όπως η κατάσταση που βρίσκονται, τα τυχόντα σφάλματα που μπορεί να παρουσιάσουν και η θέση τους.

Εικόνα 3.2.1. Πρώτη καρτέλα HMI (απενεργοποιημένο σύστημα)



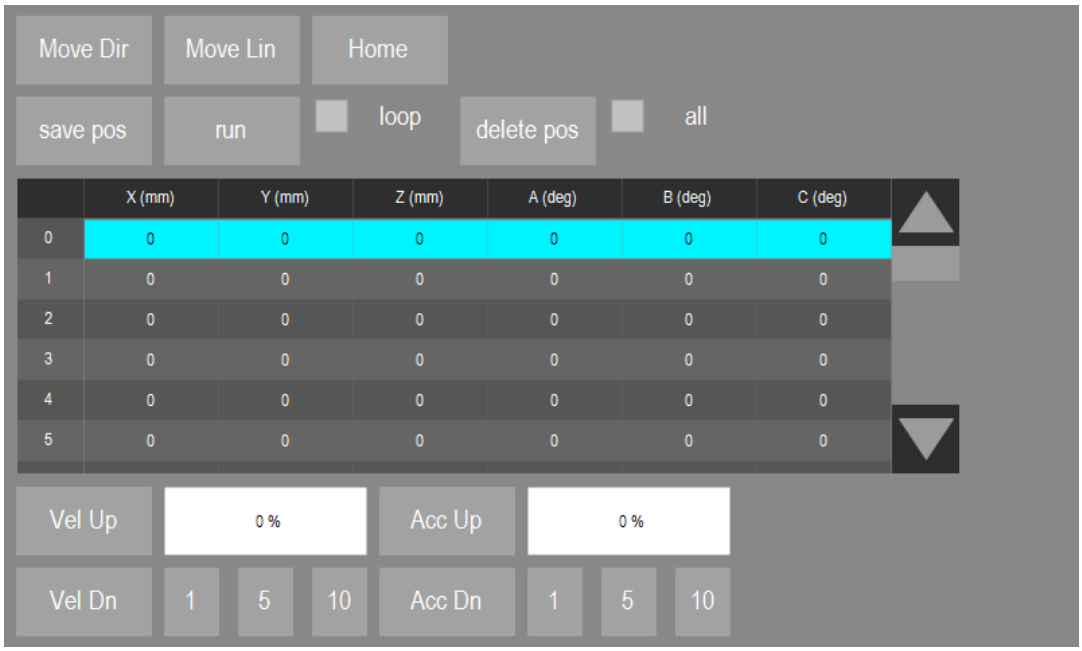
Εικόνα 3.2.2.Πρώτη καρτέλα HMI (ενεργοποιημένο σύστημα)

Στην δεύτερη καρτέλα (Εικόνα 3.2.3.) υπάρχουν κουμπιά κίνησης για κάθε άξονα ξεχωριστά έτσι ώστε να μπορούμε να κινήσουμε το ρομπότ στον χώρο χειροκίνητα. Πέρα από τις κίνησης υπάρχουν υπάρχουν τρία κουμπιά που μας επιτρέπουν να αλλάζουμε το σύστημα συντεταγμένων μεταξύ των WCS (World Coordinate System), ACS (Axis Coordinate System), PCS_1 (Product Coordinate System). Όταν είναι επιλεγμένο το WCS τότε στα κελιά όπου εμφανίζεται η θέση του ρομπότ βλέπουμε την πραγματική θέση του ρομπότ στον κόσμο με την αρχή των αξόνων να βρίσκεται στην βάση του ρομπότ. Έχοντας επιλεγμένο το ACS τότε εμφανίζονται οι γωνίες των αξόνων κατά την κίνηση, ενώ όταν είναι επιλεγμένο το PCS_1 τότε βλέπουμε την θέση του ρομπότ σε σχέση με ένα σημείο που έχουμε εμείς ορίσει. Το σημείο αυτό μπορούμε να το ορίσουμε έχοντας μετακινήσει το ρομπότ στο επιθυμητό σημείο και αφού έχουμε πατήσει το κουμπί set Transf. Με την διαδικασία αυτή μεταφέρουμε το μηδέν του ρομπότ σε όποιο σημείο στον κόσμο επιθυμούμε. Τέλος στα λευκά κελιά της οθόνης μπορούμε να ορίσουμε την επιθυμητή θέση που θέλουμε να πάει το ρομπότ.

Εικόνα 3.2.3.

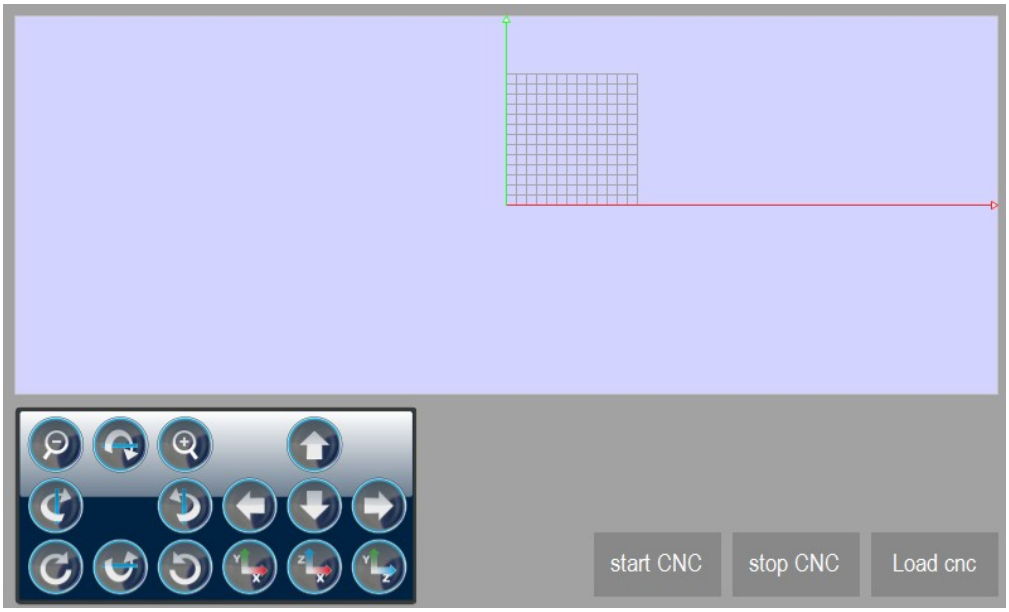
Η τρίτη καρτέλα (Εικόνα 3.2.4.) περιέχει την κύρια κίνηση του ρομπότ και δουλεύει σε συνδυασμό με την δεύτερη καρτέλα που είδαμε παραπάνω. Στην παρακάτω καρτέλα υπάρχουν οι εντολές για να κινηθεί το ρομπότ είτε απευθείας στο σημείο που έχουμε ορίσει είτε γραμμικά. Επίσης υπάρχει εντολή έτσι ώστε το ρομπότ να μετακινηθεί στην αρχική του θέση, η κίνηση αυτή επιλέγει πάντα την βέλτιστη διαδρομή. Επιπλέον μας δίνει την επιλογή να σώσουμε

κάποιες επιθυμητές θέσεις που θέλουμε να κινηθεί το ρομπότ και στην συνέχεια να επιλέξουμε να κινηθεί σε αυτές είτε μόνο μια φορά είτε σε επανάληψη δημιουργώντας έτσι ένα είδος προγράμματος. Επεκτείνοντας την παραπάνω λειτουργία, μας δίνεται επίσης η δυνατότητα να διαγράψουμε της θέσεις αυτές σε περίπτωση λάθους ή και όλο το πρόγραμμα. Επιπλέον, εμφανίζονται οπτικά όλες οι θέσεις που έχουμε ορίσει στο ρομπότ. Τέλος υπάρχουν εντολές για την αύξηση και την μείωση της ταχύτητας και της επιτάχυνσης του ρομπότ με βήμα ένα, πέντε και δέκα.

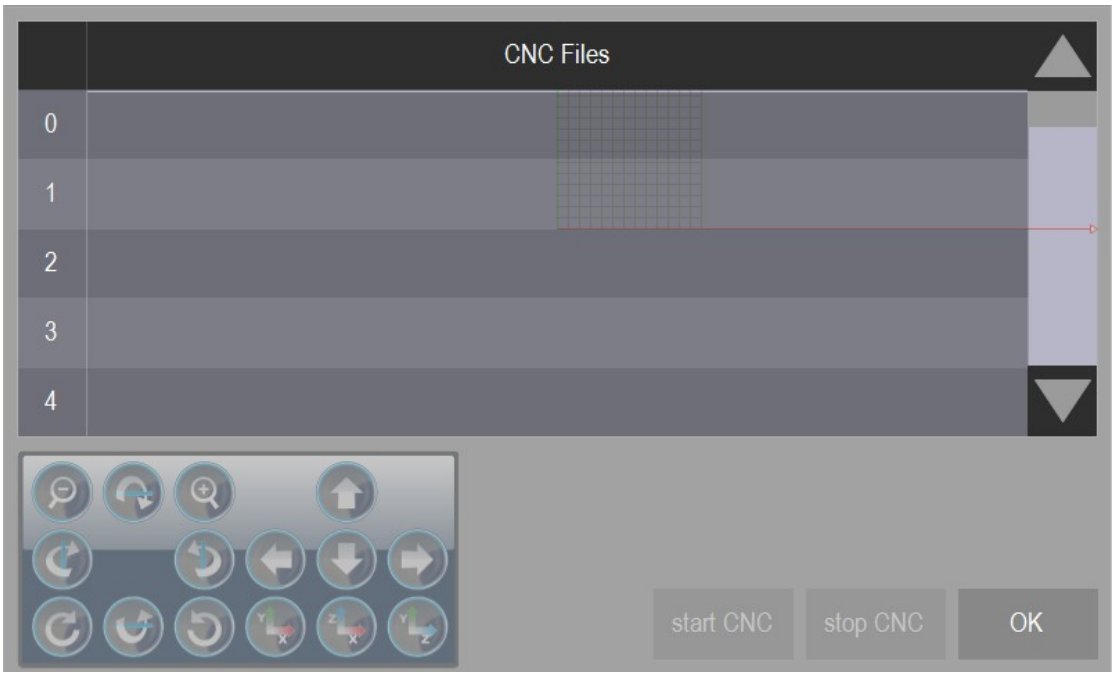


Εικόνα 3.2.4.

Στην τέταρτη καρτέλα (Εικόνα 3.2.5.) έχουμε την οπτική απεικόνιση της κίνησης του ρομπότ στο καρτεσιανό σύστημα. Η τελευταία λειτουργία που έχει το ρομπότ είναι η δυνατότητα εκτέλεσης G – κώδικα. Από την εντολή Load CNC μπορούμε να φορτώσουμε το επιθυμητό αρχείο CNC που έχουμε ήδη αποθηκεύσει στον αποθηκευτικό χώρο του PLC (Εικόνα 3.2.6.) και στην συνέχεια να εκτελώντας το πρόγραμμα ενεργοποιώντας το Start CNC να δούμε το ρομπότ να κινείται στον χώρο σύμφωνα με τον G – κώδικα.



Εικόνα 3.2.5.



Εικόνα 3.2.6.

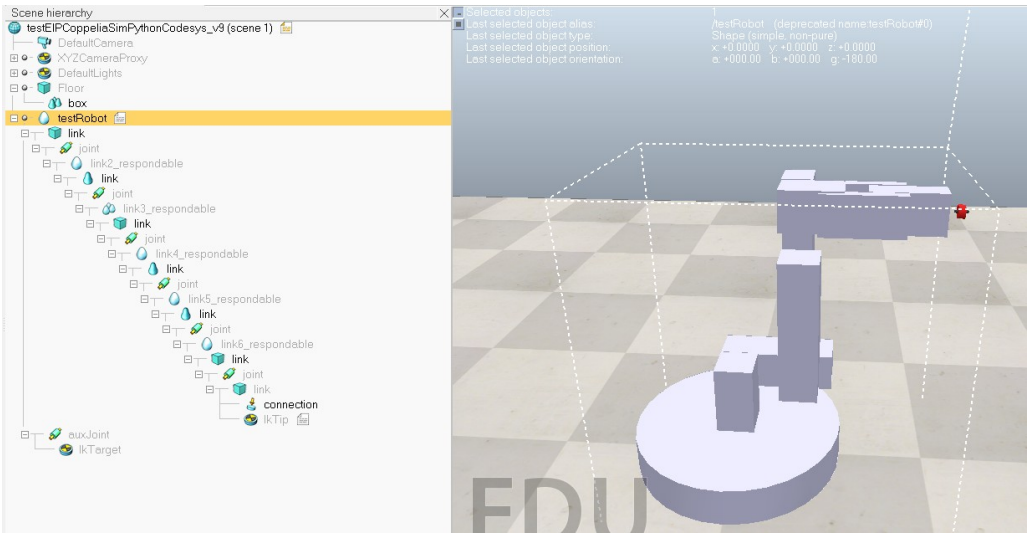
3.3. CoppeliaSim (V – Rep)

Για την οπτική απεικόνιση του ρομπότ χρησιμοποιήθηκε το CoppeliaSim. Το CoppeliaSim γνωστό ως V – Rep είναι ένα πρόγραμμα το οποίο χρησιμοποιείται στην βιομηχανία, στην εκπαίδευση αλλά και στην έρευνα. Αρχικά αναπτύχθηκε από την Toshiba R&D και πλέον αναπτύσσεται από την Robotics AG. Είναι ένα πρόγραμμα προσομοίωσης το οποίο υποστηρίζει Python και Lua κώδικα ακόμα και C/C++. Επιπλέον μπορεί ασύγχρονα να συνεργάζεται με εξωτερικά προγράμματα όπως το ROS, Remote API, ZeroMQ με γλώσσες όπως python, Java, C/C++ και Matlab. Το CoppeliaSim μπορεί να επιλύσει κινηματικά προβλήματα καθώς επίσης υποστηρίζει διάφορες βιβλιοθήκες φυσικής όπως MuJoCo, Bullet, ODE, Vortex, Newton Game Dynamics.

Στην εκτέλεση της εργασίας χρησιμοποιήθηκε κυρίως για την οπτική απεικόνιση της κίνησης του ρομπότ σε τρισδιάστατα γραφικά καθώς το CODESYS δεν υποστηρίζει αυτή την λειτουργία. Ως εκ τούτου αρχικά σχεδιάστηκε το μοντέλο του ρομπότ στο CoppeliaSim, έπειτα δημιουργήθηκε ο κώδικας για την κίνηση του ρομπότ σε Lua κώδικα και ενεργοποιήθηκε το Remote API για να επιτευχθεί η επικοινωνία του CoppeliaSim με το CODESYS PLC. Η επικοινώνια επιτυγχάνεται μέσω κώδικα Python που λειτουργεί ως μεσάζοντας καθώς επικοινωνεί μέσω ModbusTCP με το PLC και με το CoppeliaSim χρησιμοποιώντας τις βιβλιοθήκες RemoteAPI.

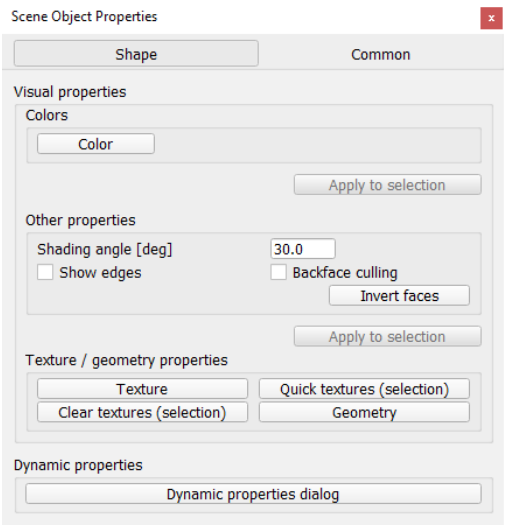
3.3.1. Σχεδιασμός Ρομποτ στο CoppeliaSim

Για τον σχεδιασμό του ρομπότ ξεκινάμε δημιουργώντας το δέντρο ιεραρχίας των επιμέρους μερών του ρομπότ. Τα μέρη του ρομπότ αποτελούνται από τους συνδέσμους και τις αρθρώσεις (Εικόνα 3.3.1.1.).

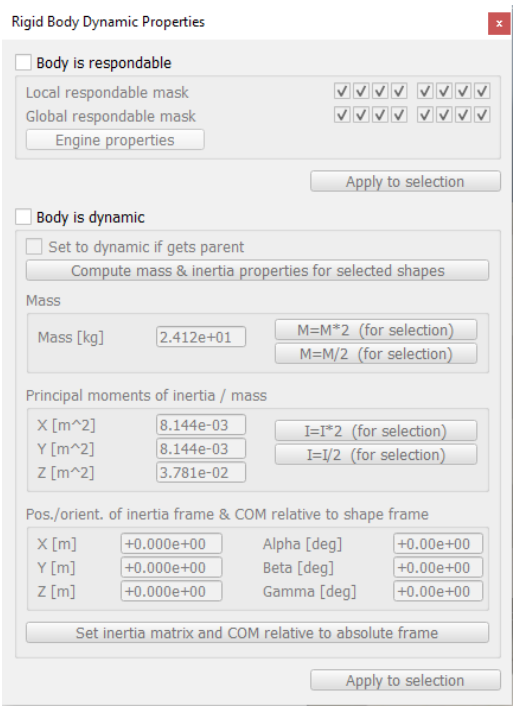


Εικόνα 3.3.1.1. Δέντρο ιεραρχίας και απεικόνιση του ρομποτικού βραχίονα

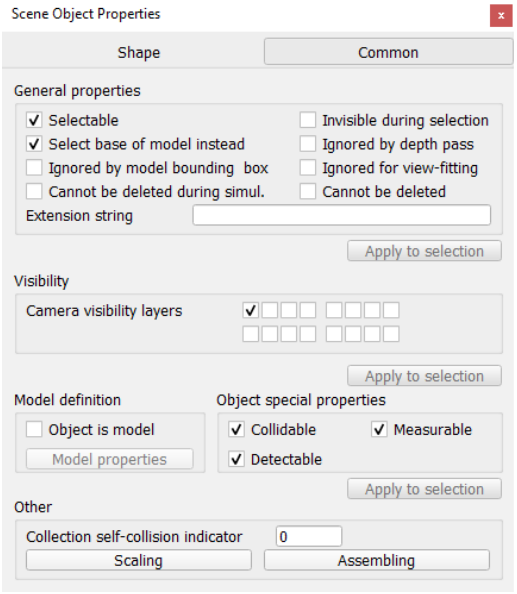
Κάθε αντικείμενο που προσθέτουμε στο σχέδιο μας επιτρέπει να του ορίσουμε τις ιδιότητές του. Στους συνδέσμους μας δίνεται η δυνατότητα να ορίσουμε την γεωμετρία του αντικειμένου, τα texture του αντικειμένου καθώς επίσης και τις δυναμικές ιδιότητές του (Εικόνα 3.3.1.2.), όπως για παράδειγμα αν το αντικείμενο είναι δυναμικό, την μάζα την αδράνεια κ.α (Εικόνα 3.3.1.3.). Επίσης κάποιες πιο γενικές ιδιότητες είναι αν το αντικείμενο θα είναι ορατό, επιλέξιμο, ανιχνεύσιμο, αν θα μπορεί να συγκρουστεί, αν θα είναι μετρήσιμο κ.α (Εικόνα 3.3.1.4.).



Εικόνα 3.3.1.2.

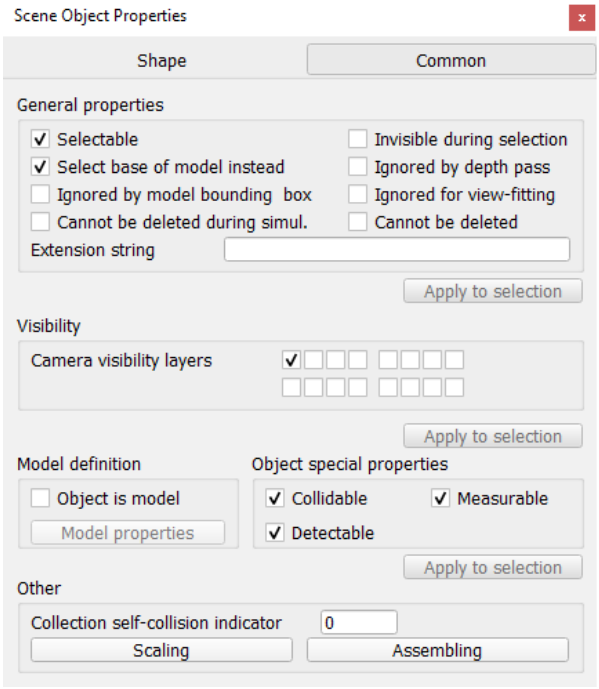


Εικόνα 3.3.1.3.

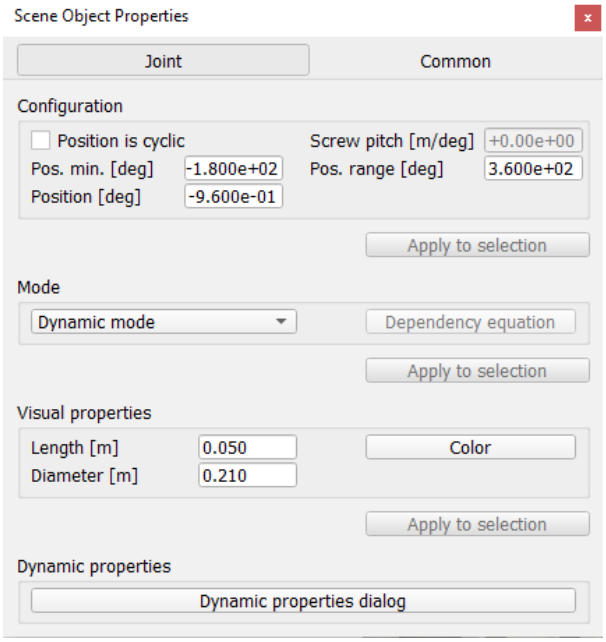


Εικόνα 3.3.1.4.

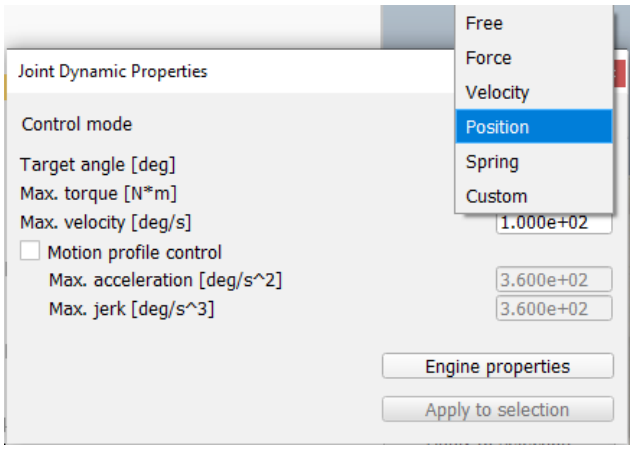
Επιπλέον, στις αρθρώσεις μπορούμε να ορίσουμε τον τύπο της άρθρωσης που έχουμε, γραμμική ή κυκλική, την θέση της άρθρωσης, το ελάχιστο που θα μπορεί να φτάσει αλλά και το εύρος της κίνησης, το μήκος της άρθρωσης και τη διάμετρο όπως επίσης και τις δυναμικές ιδιότητες της άρθρωσης (Εικόνα 3.3.1.5.). Στις δυναμικές ιδιότητες μπορούμε να ορίσουμε τον τύπο της κίνησης που θα κάνει η άρθρωση, αν θα έχει λειτουργία θέσης, λειτουργία ταχύτητας, αν θα κάνει ελεύθερη κίνηση κ.α (Εικόνα 3.3.1.6.).



Εικόνα 3.3.1.4.



Εικόνα 3.3.1.5.



Εικόνα 3.3.1.6.

Για την σωστή λειτουργία της προσομοίωσης το ρομπότ έχει σχεδιαστεί σύμφωνα με τις παραμέτρους του Denavit – Hartenberg που έχουμε επιλύσει κατά την λύση του κινηματικού προβλήματος στο 2° Κεφάλαιο.

3.3.2. Προγραμματισμός του ρομπότ σε LUA κώδικα στο CoppeliaSim

Για την κίνηση του ρομπότ στο CoppeliaSim αλλά και για την επικοινωνία με το Codesys πρέπει να δημιουργήσουμε κώδικα σε LUA στο CoppeliaSim. Αρχικά, πρέπει να δημιουργήσουμε δύο διαδικασίες για την επικοινωνία μέσω του Remote API (Εικόνα 3.3.2.1.). Η μια διαδικασία με ονομασία legacyRemoteApi_movementDataFunction αφορά την λήψη των δεδομένων (θέση, ταχύτητα, επιτάχυνση) του ρομπότ και η δεύτερη με την ονομασία legacyRemoteApi_executeMovement την εκτέλεση των δεδομένων που έχουν ληφθεί. Μέσω των δύο αυτών διαδικασιών υπάρχει επικοινωνία μεταξύ CoppeliaSim και Python.

```
-- Functions called by the legacy remote API client:
----------------------------------------------------
function legacyRemoteApi_movementDataFunction(intData,floatData,stringData,buffer)
    if not messagePack then
        messagePack=require('messagePack')
        messagePack.set_string('string')
    end
    local movData=messagePack.unpack(buffer)
    allMovementData[movData.id]=movData
    return {},{},{},''
end

function legacyRemoteApi_executeMovement(intData,floatData,stringData,buffer)
    movementToExecute[#movementToExecute+1]=buffer
    return {},{},{},''
end
```

Εικόνα 3.3.2.1.

Στην συνέχεια ακολουθούν ορισμένες ακόμα διαδικασίες που αφορούν την αρχικοποίηση του ρομποτικού βραχίονα στο CoppeliaSim και την κίνηση του. Αρχικά, ακολουθεί η αρχικοποίηση του βραχίονα (Εικόνα 3.3.2.2.) για όταν καλείται η σκηνή του CoppeliaSim για πρώτη φορά, όπου μηδενίζει την ταχύτητα και την επιτάχυνση για κάθε άξονα του βραχίονα και θέτει το ρομποτικό βραχίονα στην αρχική του θέση.

```
function sysCall_init()
    stringSignalName='/testRobot_executedMovId'
    movementToExecute={}
    allMovementData={}
    currentVel={0,0,0,0,0,0,0}
    currentAccel={0,0,0,0,0,0,0}
    maxJerk={100,100,100,100,100,100}
    sim.setStringSignal(stringSignalName,'ready')
    jointHandles={-1,-1,-1,-1,-1,-1}
    currentPosVelAccel={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
    for i=1,6,1 do
        jointHandles[i]=sim.getObject('./joint',{index=i-1})
    end
    corout = coroutine.create(coroutineStart)
end
```

Εικόνα 3.3.2.2.

Έπειτα ακολουθεί η εκτέλεση των κινήσεων (Εικόνα 3.3.2.3.) που έχουν ληφθεί από το Remote API. Για την εκτέλεση των κινήσεων υπάρχουν δύο επιλογές είτε από σημείο σε σημείο ορίζοντας τον τύπο της κίνησης ως mov, είτε σαν μια ολοκληρωμένη κίνηση (path) ορίζοντας την κίνηση ως pts. Το όρισμα του τύπου κίνησης που θα κάνει το ρομπότ γίνεται από τον κώδικα της Python μέσω του Remote API.

```
function coroutineStart()
    while true do
        if #movementToExecute>0 then
            local id=table.remove(movementToExecute,1)
            local movementData=allMovementData[id]
            allMovementData[id]=nil
            if movementData.type=='mov' then
                local currentConfig={}
                for i=1,#jointHandles,1 do
                    currentConfig[i]=sim.getJointPosition(jointHandles[i])
                end
                newPos,currentVel,currentAccel=sim.moveToConfig(-1,currentConfig,currentVel
                print("maxVel", movementData.maxVel)
            end
            if movementData.type=='pts' then
                executePtpMovement(jointHandles,movementData)
            end
            print(movementData.type)
            sim.setStringSignal(stringSignalName,id)
        else
            sim.switchThread()
        end
    end
end
```

Εικόνα 3.3.2.3.

Αν έχει ορισθεί ο τύπος κίνησης ως pts τότε καλείται και η τελευταία διαδικασία του προγράμματος που αφορά την δημιουργία διαδρομής που θα εκτελέσει ο ρομποτικός βραχίονας (path planning) (Εικόνα 3.3.2.4.).

```
function executePtpMovement(handles,data)
    -- Apply joint configs in an interpolated manner:
    local lb=sim.setThreadAutomaticSwitch(false)
    local path={}
    for i=1,#data.times,1 do
        path[(i-1)*6+1]=data.j1[i]
        path[(i-1)*6+2]=data.j2[i]
        path[(i-1)*6+3]=data.j3[i]
        path[(i-1)*6+4]=data.j4[i]
        path[(i-1)*6+5]=data.j5[i]
        path[(i-1)*6+6]=data.j6[i]
    end
    local startTime=sim.getSimulationTime()
    local t=0
    while t<data.times[#data.times] do
        local conf=sim.getPathInterpolatedConfig(path,data.times,t)
        applyJointTargetPositions(handles,conf)
        sim.switchThread()
        t=sim.getSimulationTime()-startTime
    end
    local conf=sim.getPathInterpolatedConfig(path,data.times,data.times[#data.times])
    applyJointTargetPositions(handles,conf)
    sim.setThreadAutomaticSwitch(lb)
end
```

Εικόνα 3.3.2.4.

3.4. Python

Η Python χρησιμοποιείται, όπως έχουμε ήδη αναφέρει, για την επικοινωνία του CODESYS PLC με το CoppeliaSim. Στον κώδικα της Python εκτελείται το Modbus TCP για την επικοινωνία με το PLC και την λήψη των θέσεων των αξόνων του ρομποτικού βραχίονα κατά την εκτέλεση της κίνησης σε πραγματικό χρόνο. Ταυτόχρονα εκτελείται το Remote API για την επικοινωνία με το CoppeliaSim και να έχουμε την τρισδιάστατη απεικόνιση της κίνησης. Οι βιβλιοθήκες που χρησιμοποιούνται είναι οι:

- Sim: είναι η βιβλιοθήκη του CoppeliaSim και χρησιμοποιείται για την επίτευξη της επικοινωνίας με το λογισμικό και την αποστολή των δεδομένων.
- Msgpack: επιτρέπει την αποστολή πακέτων δεδομένων.
- PyModbusTCP: χρησιμοποιείται για την επικοινωνία με το CODESYS PLC.

Αρχικά στον κώδικα εκτελείται η επικοινωνία των προγραμμάτων. Δημιουργούνται δύο clients ένας για την Modbus TCP επικοινωνία και ένας για το Remote API. Στο Mosbus TCP σαν host ορίζεται η IP του PLC ενώ για το Remote API η localhost του υπολογιστή καθώς και τα CoppeliaSim εκτελείται στον υπολογιστή όπως επίσης και ο κώδικας της Python.

```
ModClient = ModbusClient(host='192.168.1.49', debug=False, auto_open=True)

class Client:
    def __enter__(self):
        self.executedMovId1 = 'notReady'
        sim.simxFinish(-1)  # just in case, close all opened connections
        self.id = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)  # Connect to CoppeliaSim
        return self

    def __exit__(self, *err):
        sim.simxFinish(-1)
        print('Program ended')


with Client() as client:
    print("running")

    if client.id != -1:
        print('Connected to remote API server')

        targetArm = '/testRobot'

        client.stringSignalName1 = targetArm + '_executedMovId'
```

Εικόνα 3.4.1.

Αφού έχει επιτευχθεί η επικοινωνία τότε εκτελείται το κύριο μέρος του κώδικα όπου λαμβάνει δεδομένα από το CODESYS PLC και στην συνέχεια τα προωθεί στο CoppeliaSim. Αν δεν καταφέρει να διαβάσει δεδομένα από το PLC τότε εμφανίζει μήνυμα ότι δεν ήταν δυνατό να λάβει δεδομένα.

```
while True:
    # read 10 registers at address 0, store result in regs list
    ModbusTCPDataRead = ModClient.read_input_registers(0, 14)

    if ModbusTCPDataRead:
        # positions over Modbus TCP
        joint1 = (ModbusTCPDataRead[0] - c) / m
        joint2 = (ModbusTCPDataRead[1] - c) / m
        joint3 = (ModbusTCPDataRead[2] - c) / m
        joint4 = (ModbusTCPDataRead[3] - c) / m
        joint5 = (ModbusTCPDataRead[4] - c) / m
        joint6 = (ModbusTCPDataRead[5] - c) / m
        maxAcc = ModbusTCPDataRead[6]
        vel = ModbusTCPDataRead[7]
        maxVel = ModbusTCPDataRead[8]
        acc = ModbusTCPDataRead[9]
    else:
        print('unable to read registers')
```

Εικόνα 3.4.2.

Έπειτα τα προωθεί στο CoppeliaSim αφού πρώτα έχει δημιουργηθεί το πακέτο των δεδομένων που θα σταλθεί. Τέλος καλεί μια διαδικασία η οποία περιμένει την επιβεβαίωση από το CoppeliaSim ότι έλαβε τα νέα δεδομένα και αφού λάβει την επιβεβαίωση συνεχίζει να εκτελείται ο κώδικας.

```
if joint1 != prJoint1 or joint2 != prJoint2 or joint3 != prJoint3 or joint4 != prJoint4 or joint5 != prJoint5 or joint6 != prJoint6:

    # Send first movement sequence:
    targetConfig = [joint1, joint2, joint3, joint4, joint5,
                    joint6]
    movSeq = "movSeq" + str(posCount)
    movementData = {"id": movSeq, "type": "mov", "targetConfig": targetConfig,
                    "maxVel": targetVel, "maxAccel": currentAccel}
    packedMovementData = msgpack.packb(movementData)
    sim.simxCallScriptFunction(client.id, targetArm, sim.sim_scripttype_childscript,
                               'legacyRemoteApi_movementDataFunction', [], [], [],
                               packedMovementData,
                               sim.simx_opmode_oneshot)

    # Execute first movement sequence:
    sim.simxCallScriptFunction(client.id, targetArm, sim.sim_scripttype_childscript,
                               'legacyRemoteApi_executeMovement', [], [], [], movSeq,
                               sim.simx_opmode_oneshot)

    # Wait until above movement sequence finished executing:
    waitForMovementExecuted1(movSeq)
```

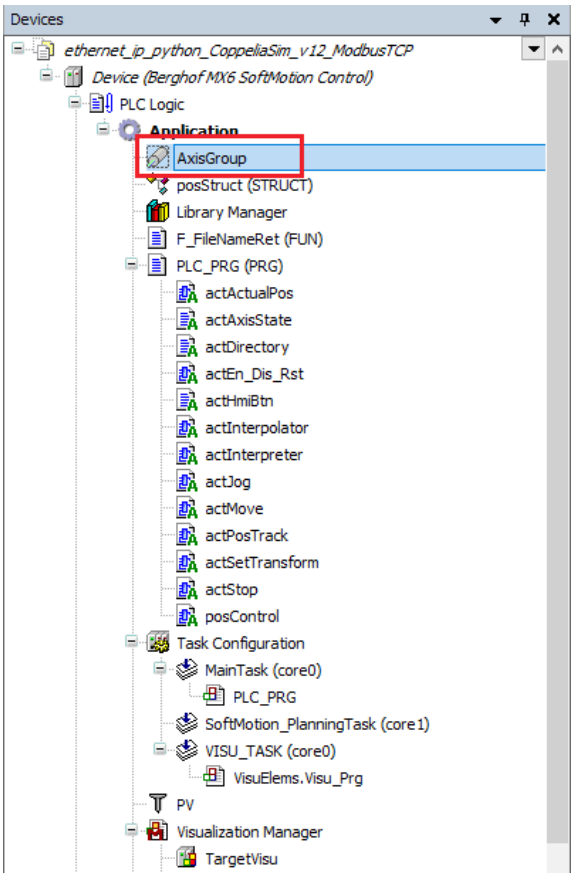
Εικόνα 3.4.3.

```
def waitForMovementExecuted1(id):
    while client.executedMovId1 != id:
        retCode, s = sim.simxGetStringSignal(client.id, client.stringSignalName1, sim.simx_opmode_buffer)
        if retCode == sim.simx_return_ok:
            if type(s) == bytearray:
                s = s.decode('ascii')  # python2/python3 differences
            client.executedMovId1 = s
            return True
```

Εικόνα 3.4.4.

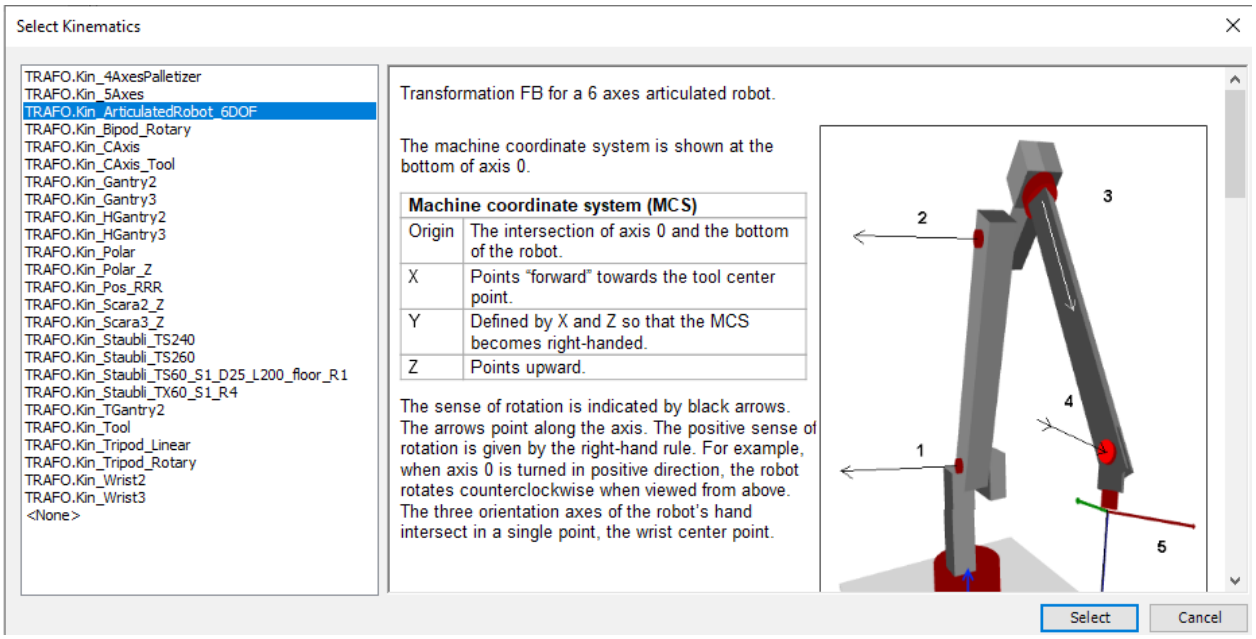
3.5. Ανάλυση κώδικα CODESYS

Όπως έχουμε ήδη αναφέρει στο CODESYS για να επιλύσουμε κινηματικά προβλήματα θα πρέπει αρχικά να έχουμε εγκαταστήσει την βιβλιοθήκη SoftMotion η οποία πέρα ότι μας δίνει την επιλογή να κινήσουμε έναν άξονα εύκολα με έτοιμα μπλοκ μας δίνει και την δυνατότητα να επιλύσουμε σύνθετα κινηματικά προβλήματα ορίζοντας γκρουπ αξόνων. Θα πρέπει ωστόσο πρώτα να προσθέσουμε ένα γκρουπ αξόνων στην εργασία (Εικόνα 3.5.1.).



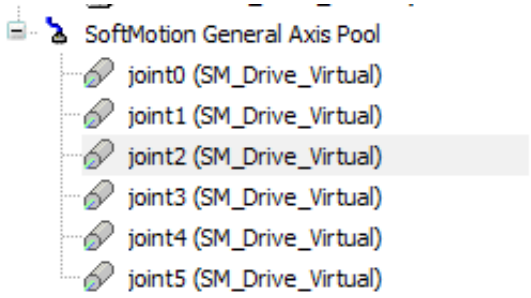
Εικόνα 3.5.1. Προσθέτουμε ένα γκρουπ αξόνων στην εργασία

Όταν ορίζουμε γκρουπ αξόνων μας δίνει την ευχέρεια να επιλέξουμε από κάποια έτοιμα κινηματικά προβλήματα (Εικόνα 3.5.2.) όπως είναι και αυτό του ρομποτικού βραχίονα έξι αξόνων.

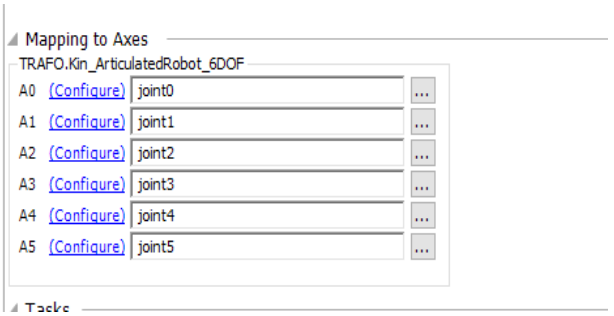


Εικόνα 3.5.2.

Έπειτα και αφού έχουμε πρώτα προσθέσει τους άξονες μας στην εργασία μας (Εικόνα 3.5.3.) θα πρέπει να τους ορίσουμε στο γκρουπ των αξόνων (Εικόνα 3.5.4.) για την επίλυση του κινηματικού.



Εικόνα 3.5.3.



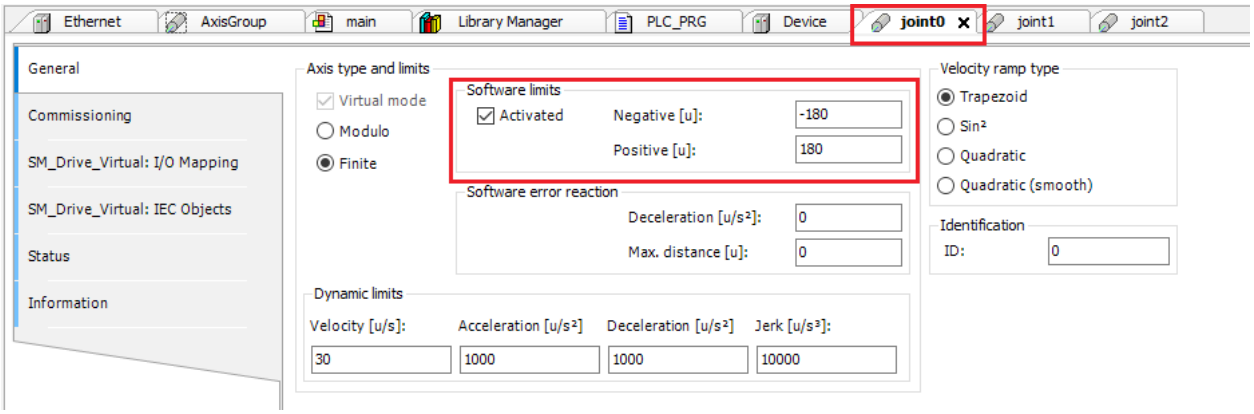
Εικόνα 3.5.4.

Επιλέγοντας το κινηματικό του ρομποτικού βραχίονα εκτός από κάποια στοιχεία για την θέση των αξόνων και την κατεύθυνση τους στον χώρο, μας δίνονται επίσης τα όρια του κάθε άξονα (άρθρωση) (Εικόνα 3.5.5.) του ρομπότ.

Axis	Configurable	Default	Min/Max
a0	YES	[-180°, 180°]	Unlimited
a1	YES	[-180°, 180°]	Unlimited
a2	YES	[-90°, 180°]	Unlimited
a3	YES	[-180°, 180°]	Unlimited
a4	YES	[-180°, 180°]	[-180°, 180°]
a5	YES	[0°, 360°]	Unlimited

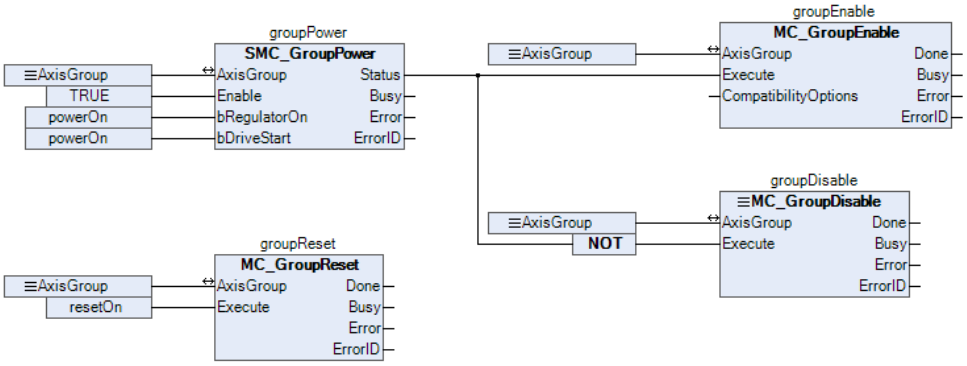
Εικόνα 3.5.5.

Τα όρια αυτά θα πρέπει να τα ορίσουμε σε κάθε άξονα που έχουμε ορίσει στην εργασία μας αναλόγως τα στοιχεία που μας δίνει το κινηματικό (Εικόνα 3.5.6.).



Εικόνα 3.5.6.

Αφού έχουμε ορίσει της βασικές ρυθμίσεις για τους άξονες του ρομπότ μπορούμε να συνεχίσουμε με τον προγραμματισμό. Αρχικά ορίζουμε τα μπλοκ για να μπορούμε να ενεργοποιούμε και να απενεργοποιούμε το γκρουπ των αξόνων όπως έχουμε ήδη αναφέρει (Power_Off → Standstill) καθώς επίσης και το μπλοκ για να μπορούμε να επαναφέρουμε τους άξονες σε κατάσταση ακινητοποίησης εάν εμφανίσουν κάποιο σφάλμα (Εικόνα 3.5.7.).



Εικόνα 3.5.7.

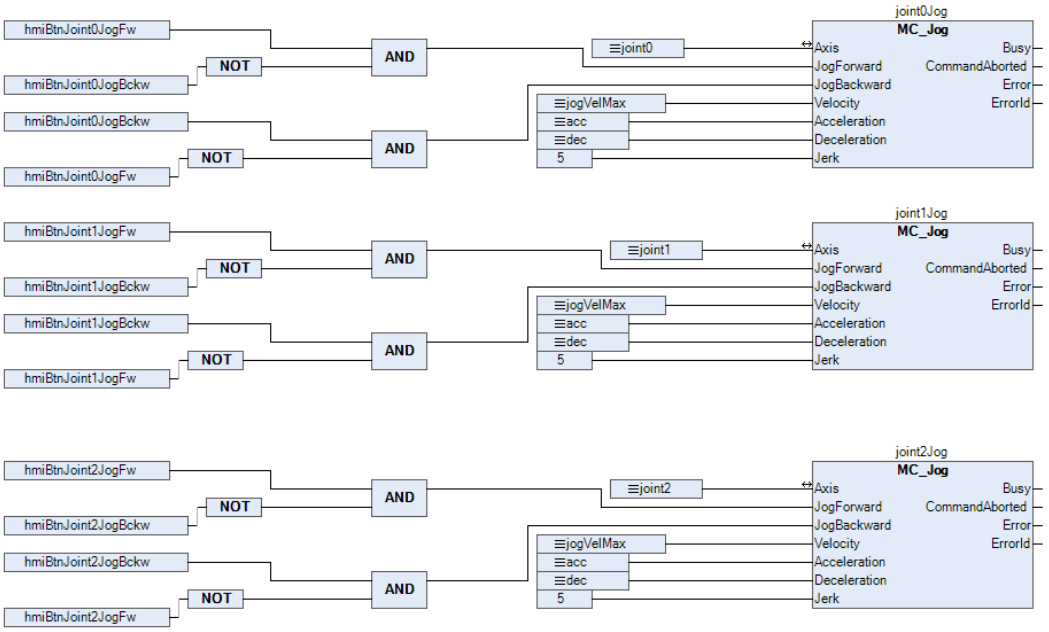
Έπειτα δημιουργούμε ένα κομμάτι κώδικα έτσι ώστε να μπορούμε να βλέπουμε κάθε στιγμή σε τι κατάσταση βρίσκεται ο κάθε άξονας η οποία εμφανίζεται και στην πρώτη καρτέλα του HMI.

```
IF joint0.nAxisState = 0 THEN
    joint0State     :=  'power_off';
ELSIF joint0.nAxisState = 1 THEN
    joint0State     :=  'errorstop';
    joint0Error     :=  TRUE;
ELSIF joint0.nAxisState = 2 THEN
    joint0State     :=  'stopping';
ELSIF joint0.nAxisState = 3 THEN
    joint0State     :=  'standstill';
ELSIF joint0.nAxisState = 4 THEN
    joint0State     :=  'discrete_motion';
ELSIF joint0.nAxisState = 5 THEN
    joint0State     :=  'continuous_motion';
ELSIF joint0.nAxisState = 6 THEN
    joint0State     :=  'synchronized_motion';
ELSIF joint0.nAxisState = 7 THEN
    joint0State     :=  'homing';
END_IF
```

Εικόνα 3.5.8.

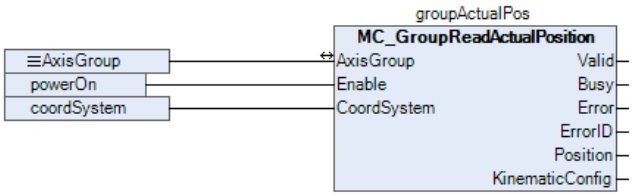
Ο παραπάνω κώδικας επαναλαμβάνεται για κάθε άξονα του ρομποτικού βραχίονα.

Στην συνέχεια καλούμε τα μπλοκ (MC_Jog) για να μπορούμε να κινούμε χειροκίνητα τον κάθε άξονα του ρομπότ ξεχωριστά στον χώρο έτσι ώστε να μπορούμε να τοποθετήσουμε το τελικό σημείο του ρομποτικού βραχίονα όπου θέλουμε στον κόσμο.



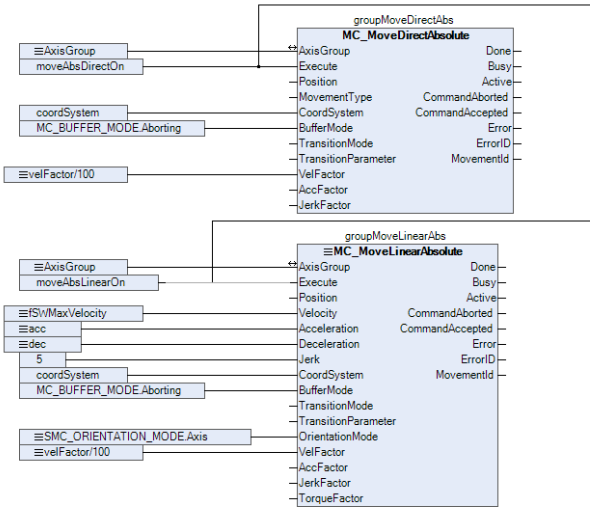
Εικόνα 3.5.9.

Ταυτόχρονα καλούμε το μπλοκ (MC_GroupReadActualPos) έτσι ώστε κατά την κίνηση του ρομπότ να διαβάζουμε που βρίσκεται στον χώρο. Ενώ ταυτόχρονα ορίζουμε και μια μεταβλητή coordSystem από την οποία ορίζουμε το σύστημα συντεταγμένων.

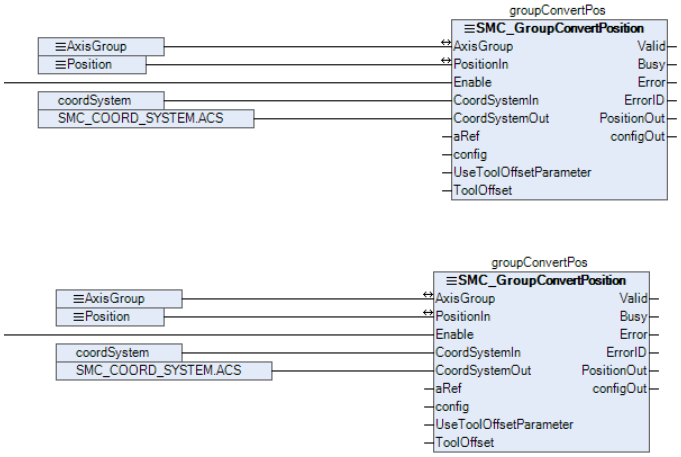


Εικόνα 3.5.10.

Στην συνέχεια ορίζουμε τα μπλοκ για την απόλυτη άμεση κίνηση του ρομπότ και την απόλυτη ευθύγραμμη κίνηση του ρομπότ (Εικόνα 3.5.11.) ενώ ταυτόχρονα μετατρέπουμε την κίνηση του ρομπότ από οποιοδήποτε σύστημα συντεταγμένων και αν βρίσκεται στο σύστημα συντεταγμένων των αξόνων (Εικόνα 3.5.12.) έτσι ώστε να μπορούμε να στείλουμε τα δεδομένα αυτά στο CoppeliaSim καθώς τις θέσεις του ρομποτικού βραχίονα πρέπει να τις μετατρέπουμε σε μοίρες.

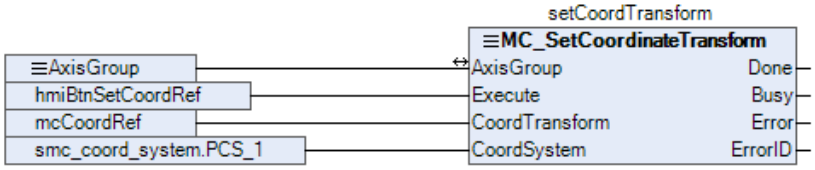


Εικόνα 3.5.11.



Εικόνα 3.5.12.

Για να μπορούμε να μεταφέρουμε το μηδέν του συστήματος συντεταγμένων του ρομποτικού βραχίονα οπουδήποτε στον χώρο πρέπει να καλέσουμε το μπλοκ MC_SetCoordinateTransform (Εικόνα 3.5.13.). Η μετατροπή αυτή μας διευκολύνει καθώς μπορούμε ενώ έχουμε δημιουργήσει ένα σύνολο κινήσεων πάνω σε ένα αντικείμενο, αν του αλλάξουμε θέση να μην χρειαστεί να ορίσουμε της κινήσεις από την αρχή αλλά απλά να αλλάξουμε το μηδέν του συστήματος συντεταγμένων και να έχουμε ακριβώς τις ίδιες κινήσεις.

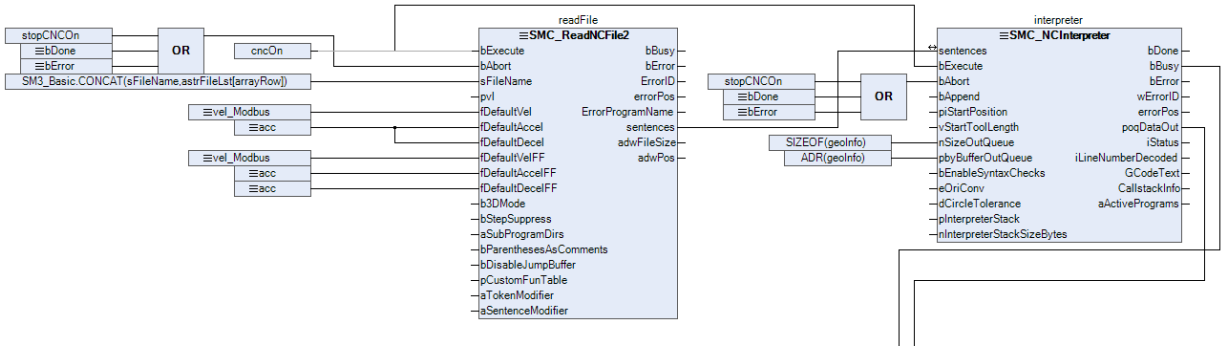


Εικόνα 3.5.13.

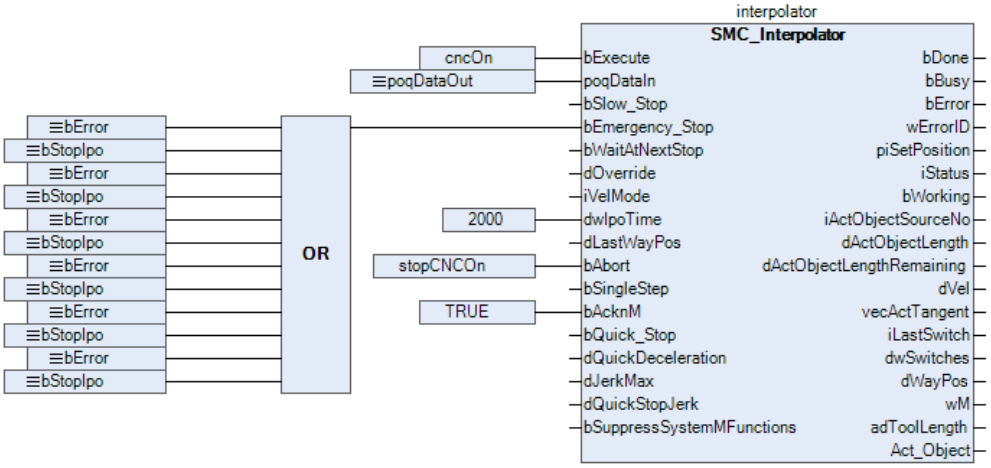
Με όλα τα παραπάνω μπλοκ μπορούμε να κινήσουμε τον ρομποτικό βραχίονα οπουδήποτε θέλουμε στον χώρο και να κάνουμε κάποιες βασικές κινήσεις. Ωστόσο για να μπορέσουμε να εκτελέσουμε πιο ολοκληρωμένες κινήσεις όπως για παράδειγμα G – κώδικα δεν αρκούν μόνο αυτά. Θα

πρέπει να χρησιμοποιήσουμε τον Interpolator και αν ο G – κώδικας προέρχεται από αρχείο στο PLC τότε θα πρέπει να χρησιμοποιηθεί και ο Interpreter.

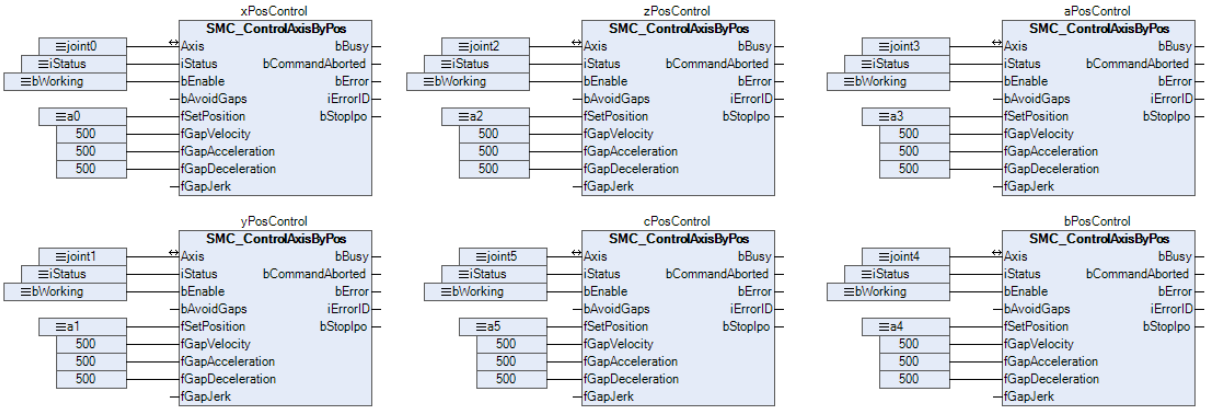
Ο Interpreter δέχεται σαν είσοδο το αρχείο του G – κώδικα και τον μετατρέπει σε μία συνεχεί διαδρομή (Εικόνα 3.5.14.) ενώ στην συνέχεια ο Interpolator την συνεχή διαδρομή την μετατρέπει σε διακριτά σημεία (Εικόνα 3.5.15.) τα οποία στην συνέχεια διοχετεύονται στους άξονες (Εικόνα 3.5.16.).



Εικόνα 3.5.14.



Εικόνα 3.5.15.



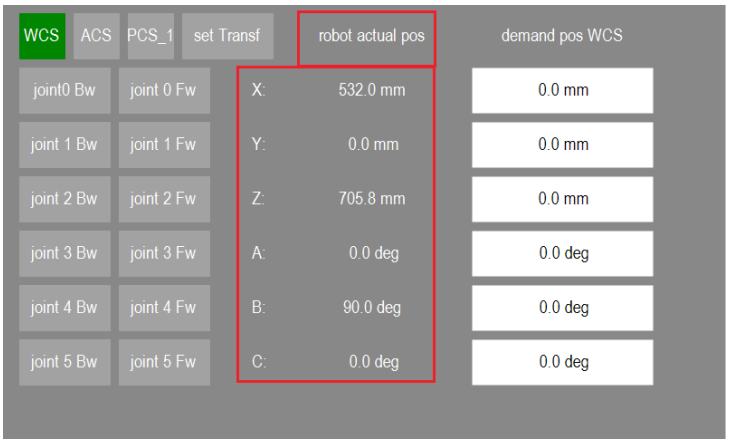
Εικόνα 3.5.16.

Ταυτόχρονα πολλές από τις παραπάνω τιμές όπως η θέση του ρομπότ ή το σύστημα συντεταγμένων ή τα σημεία που θέλουμε να κινηθεί ο ρομποτικός βραχίονας καθώς επίσης και τα σφάλματα του κάθε άξονα φαίνονται στο HMI ώστε να μπορεί ο χειριστής να ξέρει ανά πάσα στιγμή τι συμβαίνει στο σύστημα του ρομπότ και να μπορεί να αντιμετωπίσει κάποιες αναπάντεχες καταστάσεις.

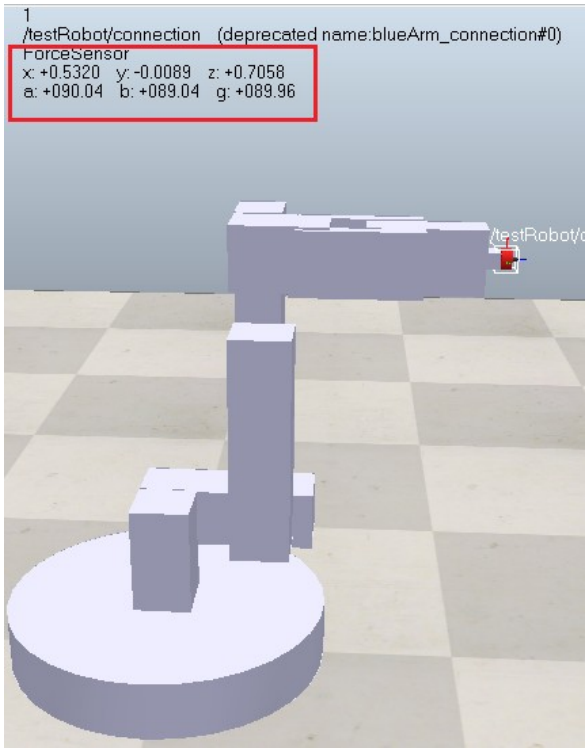
ΚΕΦΑΛΑΙΟ 4^{ο}

4. Συμπεράσματα

Αφού έχουν ολοκληρωθεί όλα τα παραπάνω βήματα εκτελούμε συνολικά το πρότζεκτ. Αρχικά, γίνεται η σύνδεση του CODESYS με το PLC και εκτελούμε τον κώδικα στο PLC. Με την εκτέλεση του κώδικα παρατηρούμε στο HMI την αρχική θέση (parking) του ρομπότ (Εικόνα 4.1.). Έπειτα ανοίγουμε την σκηνή στο CoppeliaSim και παρατηρούμε την αρχική θέση (parking) του ρομπότ. Από την σύγκριση των δύο θέσεων βλέπουμε ότι το ρομπότ και στα δύο προγράμματα είναι σωστά ορισμένα (Εικόνα 4.2.). Στο CODESYS η θέση εμφανίζεται σε χιλιοστά (mm) ενώ στο CoppeliaSim σε μέτρα (m).



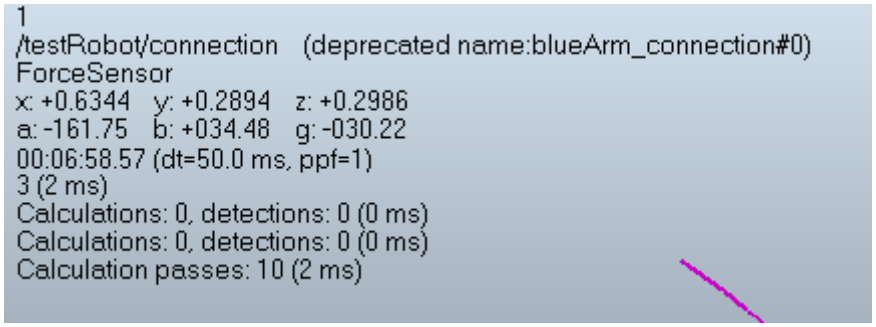
Εικόνα 4.1. Η αρχική θέση του ρομπότικού βραχίονα στο CODESYS.



Εικόνα 4.2. Η αρχική θέση του ρομποτικού βραχίονα στο CoppeliaSim.

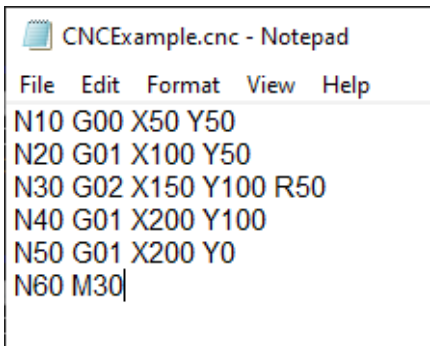
Έπειτα εκτελούμε την επικοινωνία ώστε τα δύο προγράμματα να συγχρονιστούν και να έχουμε οπτική απεικόνιση των κινήσεων του ρομπότ του CODESYS στο CoppeliaSim. Κουνώντας έναν έναν τους άξονες κατά μήκος των αξόνων (x, y, z) ελέγχουμε αν υπάρχει συσχετισμός των θέσεων των δύο προγραμμάτων (Εικόνα 4.3., Εικόνα 4.4.)

Εικόνα 4.3.Τυχαία θέση του ρομποτικού βραχίονα στον χώρο.

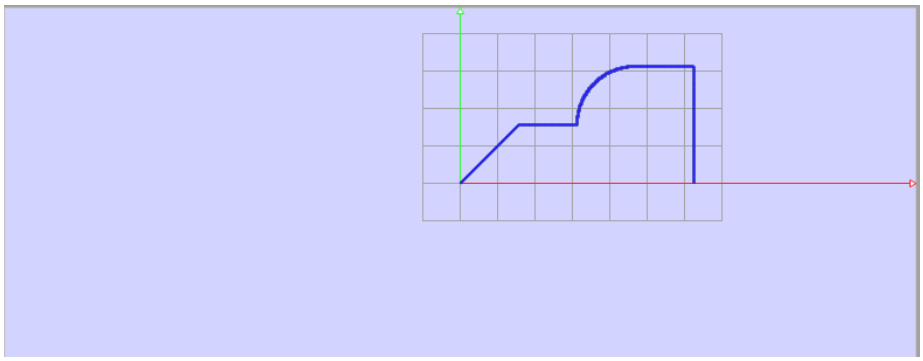


Εικόνα 4.4.Συγχρονισμένη θέση του ρομποτικού βραχιόνα στο CoppeliaSim από το CODESYS.

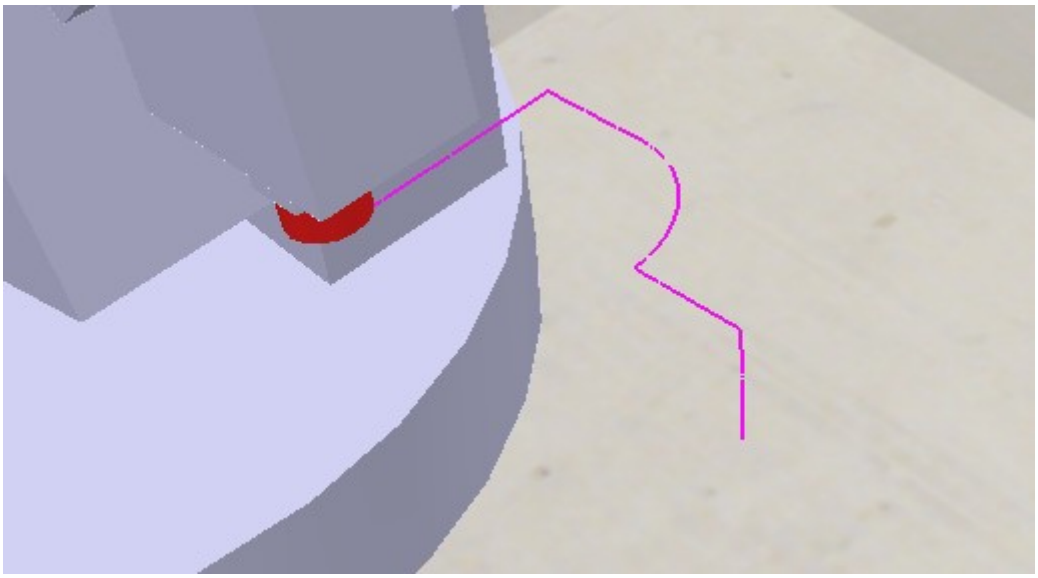
Έχοντας επιβεβαιώσει ότι ο προσανατολισμός του ρομποτικού βραχίονα είναι σωστός και η θέση του ρομπότ στο CoppeliaSim λαμβάνει σε πραγματικό χρόνο τις θέσεις των αξόνων από το CODESYS εκτελούμε G – κώδικα στο CODESYS (Εικόνα 4.5.) ώστε να έχουμε και την τελική επιβεβαίωση ότι ο ρομποτικός βραχίονας λειτουργεί σωστά (Εικόνα 4.6.).



Εικόνα 4.5.G – κώδικας.



Εικόνα 4.6. Γραφική απεικόνιση του τελικού σημείου του ρομποτικού βραχίονα από την εκτέλεση του G – κώδικα.



Εικόνα 4.7.3D απεικόνιση στο CopelliaSim από τις θέσεις που έλαβε από το CODESYS.

Βιβλιογραφία

1. Αναστασοπούλου Ελένη, "*Η τεχνητή νοημοσύνη και οι εφαρμογές της*", 2017.
2. Δουλγέρη Ζωή, "*Ρομποτική: Κινηματική, δυναμική και έλεγχος αρθρωτών βραχιόνων*", 2007.
3. Παπακωνσταντίνου Πέτρος, "*Άνθρωποι και ρομπότ, οι προκλήσεις της τεχνιτής νοημοσύνης*", 2020.
4. Φ.Ν. Κουμπούλης, Β.Γ. Μέρτζιος, "*Εισαγωγή στη Ρομποτική*", Αθήνα, 2002.
5. Χατζή Άννα, "*Η ανάδυση των ρομπότ στη βιομηχανική παραγωγή*", 2021
   Διαθέσιμο στον ιστότοπο: https://industry-news.gr/i-anadysi-ton-rompot-sti-viomichaniki-paragogi/ .
6. ΤΕΙ ΙΩΑΝΝΙΝΩΝ ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΙΑΣ, "*ΕΥΘΕΙΑ ΚΑΙ ΑΝΤΙΣΤΡΟΦΗ ΚΙΝΗΜΑΤΙΚΗ*"
   Διαθέσιμο στον ιστότοπο: https://eclass.teiwm.gr/modules/document/file.php/BSMM121/%CE%9A%CE%95%CE%A6%CE%91%CE%9B%CE%91%CE%99%CE%9F%203.pdf .
7. S.K. Saha, Tata McGraw-Hill, "I*ntroduction to Robotics*", New Delhi, 2008.
8. Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, "*Robot Dynamics and Control, Second Edition*", 2004.
9. Alfredo Valverde , Panagiotis Tsiotras, "*Spacecraft Robot Kinematics Using Dual Quaternions*", 2018.

Ιστοσελίδες

1. https://ifr.org/industrial-robots
2. https://el.wikipedia.org/wiki/%CE%A1%CE%BF%CE%BC%CF%80%CE%BF%CF%84%CE%B9%CE%BA%CE%AE
3. https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html
4. https://www.humanoid.waseda.ac.jp/booklet/kato_2.html
5. https://help.codesys.com/webapp/_sm_f_softmotion;product=codesys_softmotion;version=4.11.0.0
6. https://www.coppeliarobotics.com/helpFiles/