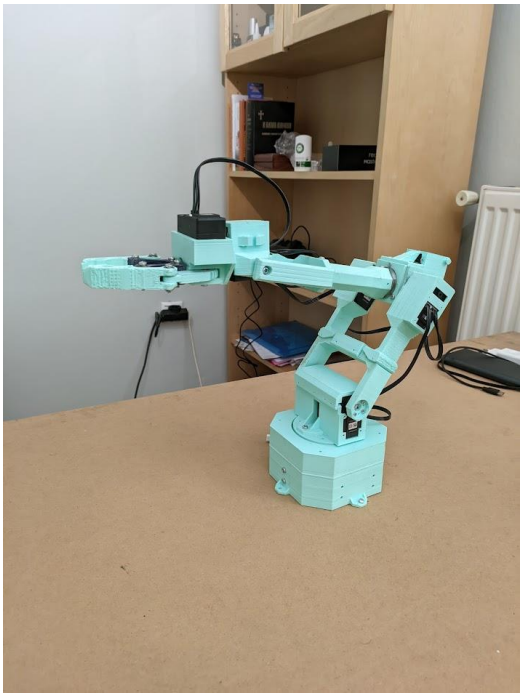
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Σχεδίαση και Ανάπτυξη Συστήματος Ρομποτικού Βραχίονα για Εφαρμογές Συναρμολόγησης



Φοιτητής: Ορφέας Καπερώνης
ΑΜ: 43575

Επιβλέπων:
Διονύσης Κανδρής
Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ, 2022

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Design and Development of a Robotic Arm System for Pick and Place Applications

Student: Orfeas Kaperonis
Registration Number: 43575

Supervisor

Dionisis Kandris,
Professor

ATHENS-EGALEO, ATHENS-EGALEO, OCTOBER 202

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Διονύσης Κανδρής, Καθηγητής	Γιώργος Πάτσης, Καθηγητής	Ηλίας Ζώης, Αναπληρωτής Καθηγητής
Dionisis Kandris Digitally signed by Dionisis Kandris Date: 2022.10.13 12:07:22 +03'00' (Υπογραφή)	Digitally signed by GEORGIOS PATSIS Date: 2022.10.12 18:43:44 +03'00' (Υπογραφή)	Ilias Zois Digitally signed by Ilias Zois Date: 2022.10.13 11:58:18 +03'00' (Υπογραφή)

Copyright © Με επιφύλαξη παντός δικαιώματος. All rights reserved.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ και Ορφέας Καπερώνης, Οκτώβριος, 2022

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον/την συγγραφέα του και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις θέσεις του επιβλέποντος, της επιτροπής εξέτασης ή τις επίσημες θέσεις του Τμήματος και του Ιδρύματος.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Ορφέας Καπερώνης του Αριστείδη, με αριθμό μητρώου 43575 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο Δηλών
Ορφέας Καπερώνης

Ευχαριστίες

Ένα μεγάλο και ιδιαίτερο ευχαριστώ οφείλω στον επιβλέποντα καθηγητή κ. Διονύση Κανδρή για την πολύτιμη βοήθεια και υποστήριξη κατά την διάρκεια εκπόνησης της εργασίας. Θα ήθελα να ευχαριστήσω θερμά την αδερφή μου για τη βοήθεια στην σύνταξη του κειμένου. Ευχαριστίες οφείλω, στον συνάδελφο μου Παναγιώτη Αγγελάκη για την σημαντική καθοδήγηση κατά τη διάρκεια υλοποίησης του παρόντος project. Τέλος, ένα ευχαριστώ οφείλω στο εργαστήριο ΠΟΙΩ του δήμου Αθηναίων για την παροχή των υπηρεσιών τρισδιάστατης εκτύπωσης.

Περίληψη

Τα αυτόνομα ρομποτικά συστήματα διαδραματίζουν όλο και πιο σημαντικό ρόλο, ιδίως στη βιομηχανία. Ωστόσο, ορισμένες εργασίες είναι πολύ περίπλοκες για να αυτοματοποιηθούν και απαιτούν ανθρώπινη παρουσία για τον έλεγχο του ρομπότ, κάτι που ονομάζεται τηλεχειρισμός. Στην περίπτωση αυτή, τα ρομπότ μπορούν να ελέγχονται απευθείας ή με την βοήθεια του χειριστή έχοντας παράλληλα ένα ορισμένο επίπεδο αυτονομίας. Ο τηλεχειρισμός μπορεί επίσης να χρησιμοποιηθεί για να διδάξει τα ρομπότ πιο φυσικές κινήσεις με μάθηση μέσω της επίδειξης.

Σε αυτή τη διπλωματική εργασία παρουσιάζεται το σύνολο των διαδικασιών που έλαβαν χώρα για τη σχεδίαση και κατασκευή ενός ρομποτικού βραχίονα έξι βαθμών ελευθερίας, και την ανάπτυξη μιας διεπαφής τηλεχειρισμού του βραχίονα η οποία βασίζεται στο λειτουργικό σύστημα ρομπότ ROS. Η διεπαφή δεν είναι συγκεκριμένη όσον αφορά τη συσκευή εισόδου και τον βραχίονα, αρκεί το λογισμικό να υποστηρίζει το ROS. Ο χρήστης μπορεί να ελέγχει απευθείας το τελικό στοιχείο δράσης του ρομπότ στον καρτεσιανό χώρο.

Λέξεις – κλειδιά

Λειτουργικό Σύστημα Ρομπότ ROS, έξυπνοι σερβοκινητήρες, ανθρωπομορφικός βραχίονας, αρπάγη.

Abstract

Autonomous robotic systems play an increasingly important role, especially in industry. However, some tasks are too complex to be automated and require human presence to control the robot, something called remote control. In this case, robots can be controlled directly or with the help of the operator while having a certain level of autonomy. Remote control can also be used to teach robots more natural movements by learning through demonstration.

This thesis presents the set of procedures that took place to design and build a six-degree-of-freedom robotic arm, and develop a remote-control interface of the arm based on the ROS robot operating system. The interface is not input device and arm specific, as long as the hardware supports ROS. The user can directly control the end effector of the robot in cartesian space.

Keywords

Robot Operation System, ROS, Smart servos, Anthropomorphic Manipulator, End-Effector, Gripper.

Περιεχόμενα

Κατάλογος Εικόνων

Αλφαβητικό Ευρετήριο

2D: Two-dimensional space

3D: Three-dimensional space

CNC: Computer numerical control

D-H: Denavit–Hartenberg

DOF: Degrees of freedom

FK: Forward Kinematic

GUI: Graphical User Interface

IK: Inverse Kinematic

ROS: Robot Operating System

URDF: Unified Robot Description Format

IEEE: The Institute for Electrical and Electronics Engineers

ΕΙΣΑΓΩΓΗ

Μία σημαντική παραδοχή που επηρέασε το αντικείμενο μελέτης της παρούσας εργασίας είναι το γεγονός ότι ο σημερινός κόσμος βιώνει μια ραγδαία επανάσταση στον τομέα της αυτοματοποίησης. Ένα χαρακτηριστικό παράδειγμα αυτής της ανάπτυξης είναι η αυξανόμενη τάση ως προς τη χρήση ρομπότ ή ρομποτικών βραχιόνων, η οποία αποτελεί μια διαρκώς διευρυνόμενη ανάγκη. Σε αυτό το πλαίσιο, αξίζει να παρατηρηθεί η ολοένα αυξανόμενη επιθυμία των ανθρώπων για την εφαρμογή της ρομποτικής στην καθημερινή τους ζωή. Ωστόσο, έχει υποστηριχθεί ότι ενώ τα ρομπότ στη βιομηχανία δημιουργούνται και χρησιμοποιούνται για πολύ συγκεκριμένους σκοπούς, αυτό έχει οδηγήσει σε έλλειψη ποικιλομορφίας όσον αφορά το λογισμικό και τη χρήση τους.
Λόγω των αβεβαιοτήτων στη μοντελοποίηση του ρομποτικού συστήματος και την ποιότητα των αισθητήρων, ο έλεγχος των κινήσεων του ρομπότ και ο σχεδιασμός της τροχιάς παρουσιάζει σημαντικές προκλήσεις. Ωστόσο, δεν μπορεί να αμφισβητηθεί το γεγονός ότι ένα ρομπότ διευκολύνει την ανθρώπινη εργασία μέσω των δραστηριοτήτων που του ζητείται κάθε φορά να εκτελέσει. Με την ύπαρξη ενός απομονωμένου περιβάλλοντος μπορεί να αποτρέψει οποιαδήποτε βλάβη, ταλαιπωρία και δυσχέρεια συναντάται σε έναν εργασιακό χώρο.
Ως εκ τούτου, με την εξέλιξη της τεχνολογίας οι περισσότερες εργασίες που εκτελούνται από μηχανές γίνονται όλο και πιο πολύπλοκες. Οι μηχανές γίνονται όλο και πιο ικανές να εκτελούν εργασίες που παλαιότερα εκτελούνταν αποκλειστικά από ανθρώπους. Ιδιαίτερα σημαντική πρόκληση για τα ρομπότ του μέλλοντος είναι η απόκτηση ανθρώπινης υπεροχής ως προς τη δομή, την ευφυΐα, τη λειτουργικότητα, αλλά και την ικανότητα να αντιδρούν έγκαιρα σε περίπλοκες καταστάσεις. Παράλληλα, σημαντικός στόχος που επιδιώκεται να αποκτηθεί από τα ρομπότ είναι η εκτέλεση εργασιών καθ' ομοίωσίν του ανθρώπου και η ύπαρξη αποτελεσματικής και ασφαλούς συνεργασίας μεταξύ ανθρώπου και ρομπότ.
Οι προαναφερθέντες λόγοι καθιστούν τα ρομπότ, τα οποία δεν παύουν να αποτελούν συστήματα ικανά για υψηλό επίπεδο αυτονομίας, εξαιρετικά πολύπλοκα ηλεκτρομηχανικά συστήματα, η αναλυτική περιγραφή των οποίων απαιτεί προηγμένες μαθηματικές μεθόδους.

Αντικείμενο της διπλωματικής εργασίας

Σκοπός και στόχοι

Ο στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μιας επικοινωνίας μεταξύ ενός custom 3D printed ρομποτικού βραχίονα και του ROS, η παρουσίαση τεχνικών υπολογισμού των forward και inverse kinematics και η επίτευξη μιας λειτουργίας Pick and Place του βραχίονα που κατασκευάστηκε.

Μεθοδολογία

Στοχεύοντας στην επίτευξη του στόχου της εργασίας, η όλη διαδικασία ξεκίνησε με τον σχεδιασμό του URDF, το οποίο αντικατοπτρίζει τον ρομποτικό βραχίονα σε μια προσομοίωση. Πάνω στο λειτουργικό σύστημα των Linux -και με τη βοήθεια του ROS και των εργαλείων του (Rviz και Moveit)- δημιουργήθηκε ένας πλήρως λειτουργικός τρόπος επικοινωνίας με τον ρομποτικό βραχίονα. Η επικοινωνία μεταξύ του ρομπότ και του ROS χτίστηκε πάνω σε 9 σερβοκινητήρες συνδεδεμένους σε σειρά, έτσι ώστε οι κινήσεις του ρομπότ να ελέγχονται κατά την κρίση του χρήστη, εντός των ορίων ασφαλείας.

Κατά τη διάρκεια του χειρισμού, ο βραχίονας κινείται κατά μήκος μίας προκαθορισμένης τροχιάς, η οποία καταλήγει στην επιθυμητή θέση και τον προσανατολισμό του end-effector. Ένα ρομπότ που έχει σχεδιαστεί για βιομηχανικές εφαρμογές δύναται να θεωρηθεί ως ένας μηχανισμός ανοικτής αλυσίδας (open-chain) που αποτελείται από ένα άκαμπτο σώμα και από έναν αριθμό αρθρώσεων που συνδέει τα άκαμπτα σώματα μεταξύ τους. Οι αρθρώσεις επιτρέπουν στα συνδεδεμένα σώματα να ακολουθούν συγκεκριμένες κινήσεις το ένα σε σχέση με το άλλο. Η περιστροφική άρθρωση λειτουργεί ως άρθρωση μεταξύ των συνδεδεμένων σωμάτων και η περιστροφική κίνηση μεταξύ τους περιορίζεται σε μία μικρή περιστροφή γύρω από τον άξονα της άρθρωσης. Είναι γνωστό ότι μία κινηματική αλυσίδα αποτελείται από μία ομάδα άκαμπτων σωμάτων που συνδέονται μεταξύ τους με αρθρώσεις. Αποτελείται από συνδέσμους (links), οι οποίοι είναι μεμονωμένα άκαμπτα σώματα που απαρτίζουν την αλυσίδα. Μια τέτοια κινηματική αλυσίδα μπορεί να είναι σειριακή, παράλληλη ή συνδυασμός και των δύο. Η ομαλή λειτουργία, για να είναι επιτυχής, απαιτεί τον υπολογισμό όλων των σημείων σε καρτεσιανές συντεταγμένες.

Καινοτομία

Η διεπαφή που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής εργασίας δεν είναι συγκεκριμένη όσον αφορά τη συσκευή εισόδου και τον βραχίονα, αρκεί το υλισμικό να υποστηρίζει το ROS. Ο χρήστης μπορεί να ελέγχει απευθείας τον τελικό ενεργοποιητή του ρομπότ στον καρτεσιανό χώρο.

Δομή

Στο πρώτο κεφάλαιο πραγματοποιείται η περιγραφή του θεωρητικού υποβάθρου που απαιτείται για την εκπόνηση της διπλωματικής εργασίας. Πέρα από την επεξήγηση βασικών εννοιών και ορισμών της ρομποτικής, συμπεριλαμβάνεται η κινηματική ανάλυση μαζί με κάποια παραδείγματα επίλυσης της ορθής και αντίστροφης κινηματικής. Επιπλέον, γίνεται αναφορά στα είδη βραχιόνων και αρπαγών που χρησιμοποιούνται τόσο στον τομέα της έρευνας όσο και της βιομηχανίας.

Το δεύτερο κεφάλαιο αναφέρεται στο λειτουργικό ROS με τις μεθόδους και τα εργαλεία που προσφέρει, αλλά και στις μεθόδους που χρησιμοποιήθηκαν.

Στο τρίτο κεφάλαιο αναλύεται η διαδικασία συναρμολόγησης του custom 6-DOF Manipulator. Ειδικότερα, παρουσιάζονται ο εξοπλισμός, τα εξαρτήματα και η κατασκευή του με τη χρήση εκτυπωτή 3D αντικειμένων.

Το τέταρτο κεφάλαιο αποτελεί τη σύνοψη της παρούσας εργασίας. Ειδικότερα, παρουσιάζονται τόσο τα προβλήματα που ανέκυψαν κατά τη διάρκεια της έρευνας όσο και τρόποι αντιμετώπισής τους. Ακόμη, καταγράφονται τα κυριότερα συμπεράσματα της έρευνας και σχολιάζονται κριτικά τα αποτελέσματα που προέκυψαν μέσα από αυτήν. Το κεφάλαιο ολοκληρώνεται με προτάσεις που παρατίθενται και αποσκοπούν σε περαιτέρω επέκταση και μελλοντική ανάπτυξη του project που απασχόλησε τον ερευνητή.

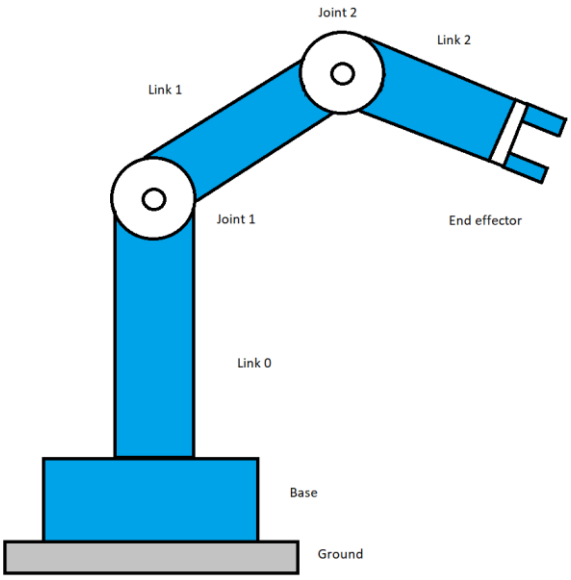
1 Θεωρητικό υπόβαθρο

1.1 Εισαγωγή κεφαλαίου

Σε αυτό το κεφάλαιο καταστρώνεται το θεωρητικό υπόβαθρο που αφορά την ανατομία, τα λειτουργικά χαρακτηριστικά και τον προγραμματισμό κίνησης των ρομποτικών συστημάτων.

1.2 Ορισμός του ρομπότ

Σύμφωνα με το Robot Institute of America (1979) ένα ρομπότ είναι:
"Ένας επαναπρογραμματιζόμενος, πολυλειτουργικός χειριστής που έχει σχεδιαστεί για να μετακινεί υλικά, εξαρτήματα, εργαλεία ή εξειδικευμένες συσκευές μέσω διαφόρων προγραμματισμένων κινήσεων για την εκτέλεση ποικίλων εργασιών". Η δυνατότητα επαναπρογραμματισμού σε συνδυασμό με αρκετά χαρακτηριστικά τους τα καθιστούν κατάλληλα για εφαρμογές σε βιομηχανικό περιβάλλον. Χονδρικά τα ρομπότ μπορούν να ταξινομηθούν με βάση τον **τύπο μετάδοσης κίνησης** (ηλεκτρική, πνευματική, υδραυλική), τη **μηχανική δομή** (ανθρωπομορφικά, καρτεσιανά, σφαιρικά κ.α.) και τον **τρόπο ελέγχου και κίνησης** (servro και non-servo, συνεχής κίνηση ή σημείο προς σημείο).



Εικόνα 1.1 - Ρομποτικός Βραχίονας

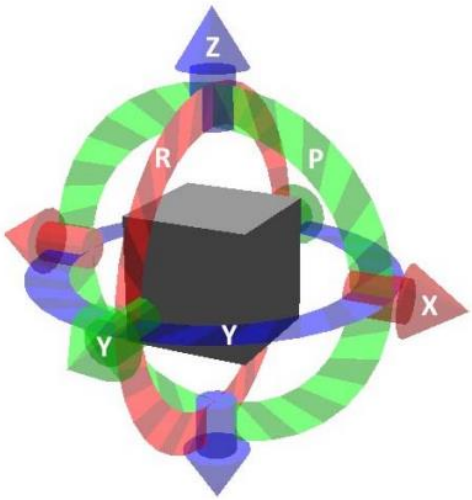
1.3 Ρομποτικός βραχίονας

Στο κείμενο αυτής της πτυχιακής ο όρος ρομπότ αφορά έναν μηχανικό χειριστή, ο οποίος ελέγχεται με τη χρήση υπολογιστή. Τα τρία δομικά του στοιχεία είναι: ο βραχίονας (χειριστής), δηλαδή το

μηχανικό τμήμα που κινείται, ο υπολογιστής που εκτελεί τα προγράμματα ελέγχου και οι κινητήρες (στοιχεία δράσης). Πέρα από τα παραπάνω, άλλα σημαντικά μέρη μπορούν να είναι αισθητήρες (sensors) που μεταφέρουν σήματα από το περιβάλλον ή καταγράφουν κινήσεις μερών του ρομπότ. Τέλος, είναι εξοπλισμένο με ένα ηλεκτρονικό σύστημα κίνησης μεταξύ της κωδικοποίησης και της μεταφοράς του σήματος κάθε αισθητήρα και αποκωδικοποίησης και εκτέλεσης κάθε εντολής που στέλνεται στους κινητήρες. Ο βραχίονας χωρίζεται στο κυρίως σώμα και τον καρπό που στην άκρη του συνδέεται με το τελικό στοιχείο δράσης (end-effector). Στα κεφάλαια 1.9 και 1.10 αναφέρονται τα είδη και οι εφαρμογές τους.

Όπως φαίνεται στην
1-1, το κυρίως σώμα απαρτίζεται από μία αλυσίδα συνδέσμων (links), οι οποίοι συνδέονται με αρθρώσεις (joints). Οι κινήσεις των συνδέσμων ελέγχονται από τις αρθρώσεις. Η καθεμία από τις αρθρώσεις ενεργοποιείται αυτόνομα μέσω ενός ενεργού στοιχείου (actuator). Η κίνηση του actuator είναι περιστροφική ή γραμμική. Ο αρχικός σύνδεσμος είναι σταθερός και ονομάζεται βάση (base link), ενώ ο τελευταίος αποτελεί το τελικό στοιχείο δράσης (end-effector).

1.4 Βαθμοί ελευθερίας



Εικόνα 1.2 – Άξονες περιστροφής(www.researchgate.net)

Ένα άκαμπτο σώμα στον τρισδιάστατο χώρο έχει έξι βαθμούς ελευθερίας (DOF). Τρεις από αυτούς είναι για τη θέση: κίνηση κατά μήκος του x, κίνηση κατά μήκος του y και κίνηση κατά μήκος του z, και τρεις είναι για τον προσανατολισμό: περιστροφή γύρω από το x ή roll, περιστροφή γύρω από το y ή pitch και περιστροφή γύρω από το z ή yaw. Στην περίπτωση ενός ρομποτικού βραχίονα, καθώς έχει την μορφή μίας κινηματικής αλυσίδας και η κάθε άρθρωση καθορίζεται από μία μεταβλητή, το σύνολο των αρθρώσεων ισούται με αυτό των βαθμών ελευθερίας (6-dof).

1.5 Κινηματική ανάλυση

Κινηματική ονομάζεται ο τομέας της Μηχανικής που ερευνά την κίνηση των στερεών σωμάτων, χωρίς να περιλαμβάνει τη μάζα τους και τις δυνάμεις που μπορεί να προκαλέσει η κίνηση τους. Η κινηματική ανάλυση στη ρομποτική αφορά τη γεωμετρική θέση και την κίνησή της ως προς κάποιο σταθερό σύστημα συντεταγμένων. Δεν αναφέρεται στις ροπές ή τις δυνάμεις που προκαλούν την κίνησή της. Διακρίνεται σε δύο τομείς: την ορθή κινηματική (forward kinematics) η οποία αναφέρεται στο ευθύ κινηματικό πρόβλημα και την αντίστροφη κινηματική (inverse kinematics) η οποία αναφέρεται στις κινηματικές εξισώσεις.

1.6 Ευθύ κινηματικό πρόβλημα

Παίρνοντας ως δεδομένα τις μεταβλητές των συνδέσμων (joints) να προσδιοριστεί ο προσανατολισμός και η θέση του end-effector ενός ρομπότ. Στις περιστροφικές αρθρώσεις μεταβλητές θεωρούνται οι γωνίες μεταξύ ενώσεων (links), ενώ στις πρισματικές αρθρώσεις μεταβλητή θεωρείται η επιμήκυνσή τους. Επιζητείται ο υπολογισμός της γραμμικής και της γωνιακής ταχύτητας του end-effector [16] .

1.7 Αντίστροφο κινηματικό πρόβλημα

Παίρνοντας ως δεδομένο τον επιθυμητό προσανατολισμό και τη θέση του end effector να προσδιοριστεί ένα σύνολο μεταβλητών των συνδέσμων(joints) που επιτυγχάνουν τον επιθυμητό προσανατολισμό και θέση. Επιζητείται ο υπολογισμός των ταχυτήτων των joints που σαν αποτέλεσμα θα έχουν συγκεκριμένη γραμμικής και γωνιακής ταχύτητας για τον end – effector[16].

1.8 Χώρος εργασίας

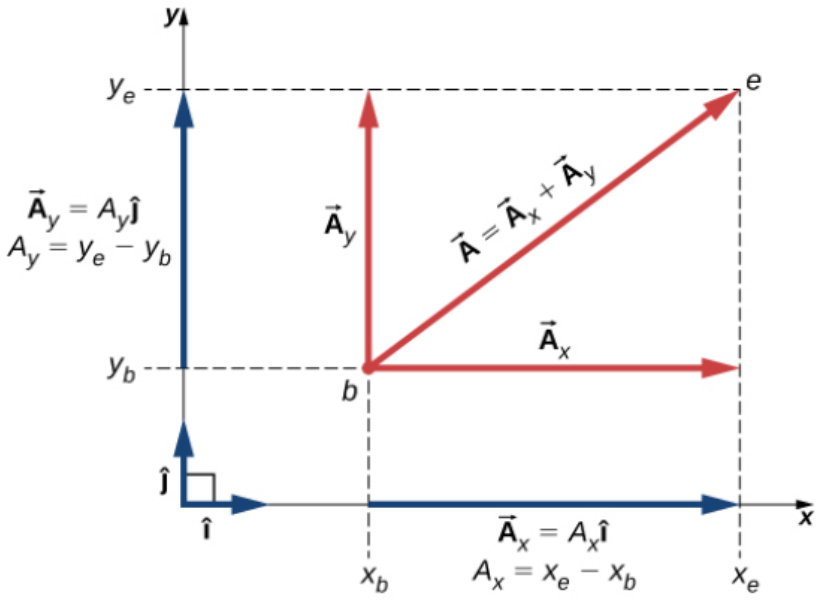
Ο χώρος εργασίας είναι το σύνολο όλων των σημείων που είναι προσβάσιμα από τον end-effector και αποτελεί πρωταρχικό κριτήριο κατά την επιλογή ενός βραχίονα σταθερής βάσης. Ο χώρος εργασίας μπορεί να χωριστεί σε δύο περιοχές:

το **reachable (προσπελάσιμο) workspace**, δηλαδή αυτό που υπονοείται από τον όρο **workspace**, και το **dexterous workspace**. Dexterous workspace είναι το σύνολο όλων των θέσεων που μπορούν να επιτευχθούν με όλους τους δυνατούς προσανατολισμούς. Το dexterous workspace είναι ένα υποσύνολο του reachable workspace.

1.9 Διανύσματα

Τα διανύσματα στα μαθηματικά είναι μία γεωμετρική οντότητα που έχει τόσο μέγεθος όσο και κατεύθυνση. Τα διανύσματα έχουν ένα αρχικό σημείο στο σημείο όπου ξεκινούν και ένα τελικό σημείο που δηλώνει την τελική θέση του σημείου. Σε ένα ορθογώνιο (καρτεσιανό) σύστημα συντεταγμένων xy, ένα σημείο σε ένα επίπεδο περιγράφεται από ένα ζεύγος συντεταγμένων (x, y). Έτσι και το διάνυσμα $\vec{A}$ σε ένα επίπεδο περιγράφεται από ένα ζεύγος διανυσματικών συντεταγμένων. Η συντεταγμένη x του διανύσματος $\vec{A}$ ονομάζεται x-συνιστώσα($\overrightarrow{Ax}$) και η συντεταγμένη y του διανύσματος ονομάζεται y-συνιστώσα($\overrightarrow{Ay}$). Στο καρτεσιανό σύστημα, οι συνιστώσες x και y ενός διανύσματος είναι οι ορθογώνιες προβολές αυτού του διανύσματος στους άξονες x και y αντίστοιχα. Κάθε διάνυσμα στο καρτεσιανό επίπεδο μπορεί να εκφραστεί ως το διανυσματικό άθροισμα των διανυσματικών συνιστωσών(1.1) του:

$$\vec{A} = \overrightarrow{Ax} + \overrightarrow{Ay} (1.1)$$



Εικόνα 1.3 - Διάνυσμα στο καρτεσιανό επίπεδο

Συνηθίζεται να συμβολίζεται η θετική κατεύθυνση στον άξονα x με το μοναδιαίο διάνυσμα $\hat{\imath}$ και η θετική κατεύθυνση στον άξονα y με το μοναδιαίο διάνυσμα$\hat{\jmath}$. Τα μοναδιαία διανύσματα των αξόνων, $\hat{\imath}$ και $\hat{\jmath}$, ορίζουν δύο ορθογώνιες κατευθύνσεις στο επίπεδο. Όπως φαίνεται στο (Σχήμα), οι x- και y-συνιστώσες ενός διανύσματος μπορούν τώρα να γραφούν ως προς τα μοναδιαία διανύσματα των αξόνων:

$$\overrightarrow{Ax} = Ax\hat{\imath} \text{ (1.2)}$$
$$\overrightarrow{Ay} = Ay\hat{\jmath} \text{ (1.3)}$$

Τα διανύσματα $\overrightarrow{Ax}$,$\overrightarrow{Ay}$ (1.2,1.3)που ορίζονται στην Εικόνα 1.3, είναι οι διανυσματικές συνιστώσες του διανύσματος $\vec{A}$. Οι αριθμοί Ax και Ay είναι οι κλιμακωτές συνιστώσες του διανύσματος $\vec{A}$. Συνδυάζοντας το σχήμα και τις παραπάνω εξισώσεις καταλήγουμε(1.4):

$$\vec{A} = Ax\hat{\imath} + Ay\hat{\jmath} \text{ (1.4)}$$

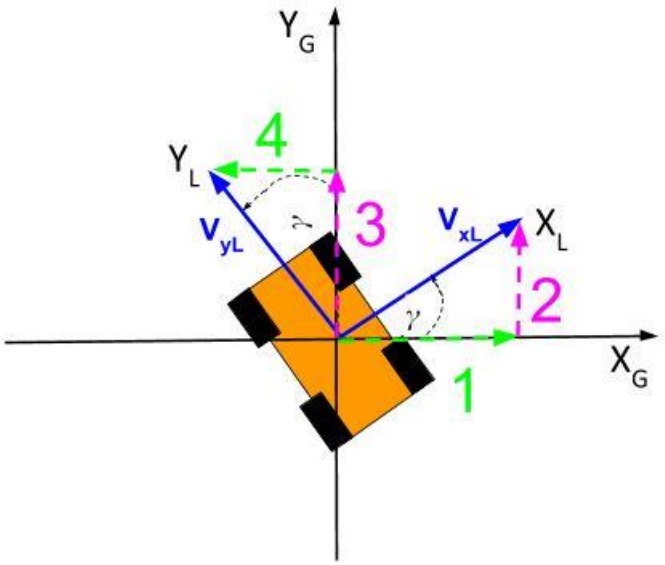
Αν γνωρίζουμε τις συντεταγμένες $b(x_b, y_b)$ του σημείου αρχής ενός διανύσματος (όπου b σημαίνει "αρχή") και των συντεταγμένων $e(x_e, y_e)$ του τελικού σημείου ενός διανύσματος (όπου e σημαίνει "τέλος"), μπορούμε να λάβουμε τις συνιστώσες ενός διανύσματος(1.5,1.6) απλά αφαιρώντας τις συντεταγμένες του σημείου αρχής από τις συντεταγμένες του τελικού σημείου:

$$Ax = x_e - x_b \text{ (1.5)}$$
$$Ay = y_e - y_b \text{ (1.6)}$$

1.10 Πίνακας περιστροφής

Σε αυτή την ενότητα γίνεται η περιγραφή των πινάκων περιστροφής οι οποίοι μπορούν να περιγράψουν την κίνηση ενός ρομπότ σε δύο και τρεις διαστάσεις.

1.10.1 Περιστροφή ενός ρομπότ σε 2D



Εικόνα 1.4 – Ρομπότ σε 2D

Το παρακάτω ρομπότ κινείται σε ένα πλαίσιο συντεταγμένων 2D. Η θέση του σε αυτό το πλαίσιο συντεταγμένων μπορεί να περιγραφεί ανά πάσα στιγμή από τη συντεταγμένη x και τη συντεταγμένη y.

Αυτό το πλαίσιο συντεταγμένων χαρακτηρίζεται ως "παγκόσμιο" πλαίσιο αναφοράς, επειδή ορίζει τον κόσμο στον οποίο κινείται το ρομπότ.

Επίσης, υπάρχει ένα άλλο πλαίσιο συντεταγμένων που χαρακτηρίζεται ως "τοπικό" πλαίσιο αναφοράς. Το τοπικό πλαίσιο αναφοράς είναι το πλαίσιο συντεταγμένων από την οπτική γωνία του ρομπότ (σε αντίθεση με τον κόσμο).

γ = Γωνιακή διαφορά μεταξύ του παγκόσμιου άξονα x και του τοπικού άξονα x

Η ταχύτητα είναι ένα μέτρο του πόσο γρήγορα κινείται κάτι προς μία συγκεκριμένη κατεύθυνση. Είναι η ταχύτητα (π.χ. μίλια ανά ώρα, μέτρα ανά δευτερόλεπτο κ.λπ.) + κατεύθυνση.

Η ταχύτητα του 2D ρομπότ μπορεί να αναλυθεί σε δύο συνιστώσες: ταχύτητα στην κατεύθυνση x και ταχύτητα στην κατεύθυνση y. Το άθροισμα των δύο αυτών διανυσμάτων ορίζει την ταχύτητα του ρομπότ.

Παρατηρείται ότι η γωνία μεταξύ Vx και Vy είναι 90 μοίρες, οπότε με βάση το Πυθαγόρειο θεώρημα(1.7) μπορεί να υπολογιστεί η ταχύτητα (δηλαδή το μέγεθος της ταχύτητας) του ρομπότ.

$$|V| = \sqrt{Vx^2 + Vy^2} \quad (1.7)$$

Αυτή είναι η ταχύτητα του ρομπότ στο τοπικό σύστημα αναφοράς. Με άλλα λόγια, τόσο η Vx όσο και η Vy είναι παράλληλες με τον άξονα x και τον άξονα y του ρομπότ αντίστοιχα.

Πώς μετατρέπουμε αυτές τις ταχύτητες στο τοπικό σύστημα αναφοράς σε ταχύτητες στο παγκόσμιο σύστημα αναφοράς;

Ο λόγος αυτής της μετατροπής είναι η γνώση της θέσης του ρομπότ στον κόσμο σε μία συγκεκριμένη χρονική στιγμή στο μέλλον. Η απλή γνώση των διανυσμάτων ταχύτητας του ρομπότ δεν είναι βοηθητική, καθώς στο παγκόσμιο σύστημα αναφοράς είναι απαραίτητη η γνώση του τρόπου μεταβολής της θέσης του ρομπότ σε σχέση με τον χρόνο.

Η ταχύτητα του ρομπότ στο παγκόσμιο σύστημα ως προς τους άξονες x και y ορίζεται αντίστοιχα ως VxG και VyG. Από το παραπάνω σχήμα των ροζ και πράσινων διανυσμάτων (1,4 - Εικόνα 1.5) γίνεται εμφανές ότι:

VxG = (διάνυσμα 1) - (διάνυσμα 4) Αρνητικό πρόσημο επειδή το διάνυσμα 4 δείχνει προς την αρνητική κατεύθυνση του παγκόσμιου άξονα x.

VyG = (διάνυσμα 2) + (διάνυσμα 3) Θετικό πρόσημο επειδή και τα δύο διανύσματα δείχνουν προς τη θετική παγκόσμια κατεύθυνση του άξονα y.

διάνυσμα 1 = VxL * cos(γ)

διάνυσμα 2 = VxL * sin(γ)

διάνυσμα 3 = VyL * cos(γ)

διάνυσμα 4 = VyL * sin(γ)

Συνδέοντας αυτές τις πληροφορίες, ακολουθεί η χρήση των δύο εξισώσεων(1.8,1.9) που επιτρέπουν τη μετατροπή των τοπικών ταχυτήτων σε παγκόσμιες ταχύτητες.

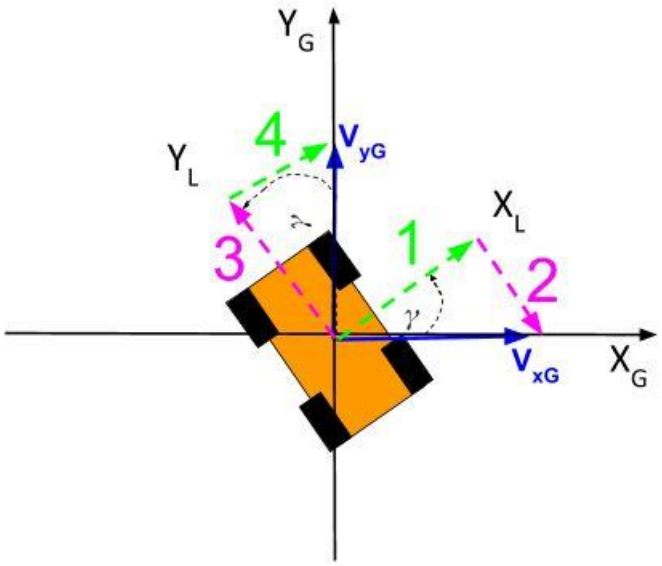
$$VxG = \big(VxL * cos(\gamma)\big) - \big(VyL * sin(\gamma)\big) \quad (1.8)$$

$$VyG = \big(VxL * sin(\gamma)\big) + \big(VyL * cos(\gamma)\big) \quad (1.9)$$

Έπειτα, ακολουθεί η μετατροπή τους στους παρακάτω πίνακες(1.10):

$$\begin{bmatrix} Vx_G \\ Vy_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{bmatrix} Vx_L \\ Vy_L \end{bmatrix} \quad (1.10)$$

Στο ακόλουθο σχήμα φαίνεται η μετατροπή της ταχύτητας από το παγκόσμιο σύστημα αναφοράς σε ταχύτητα στο τοπικό σύστημα αναφοράς.



Εικόνα 1.5 – Ρομπότ σε 2D με μετατροπή ταχύτητας στο τοπικό σύστημα αναφοράς

Σε αυτό το σημείο καταγράφονται οι σχέσεις που αφορούν τις τοπικές ταχύτητες στις διευθύνσεις x και y:

VxL = (διάνυσμα 1) + (διάνυσμα 4) ... Πρόσημο συν επειδή και τα δύο διανύσματα δείχνουν προς τη θετική τοπική κατεύθυνση του άξονα x.

VyL = -(διάνυσμα 2) + (διάνυσμα 3) ... Πρόσημο μείον επειδή το διάνυσμα 2 δείχνει προς την αρνητική τοπική κατεύθυνση του άξονα y.

διάνυσμα 1 = VxG * cos(γ)

διάνυσμα 2 = VXG * sin(γ)

διάνυσμα 3 = VyG * cos(γ)

διάνυσμα 4 = VyG * sin(γ)

Συνδέοντας τις παραπάνω πληροφορίες, γίνεται αξιοποίηση των δύο εξισώσεων(1.11,1.12) που επιτρέπουν τη μετατροπή των παγκόσμιων ταχυτήτων σε τοπικές ταχύτητες.

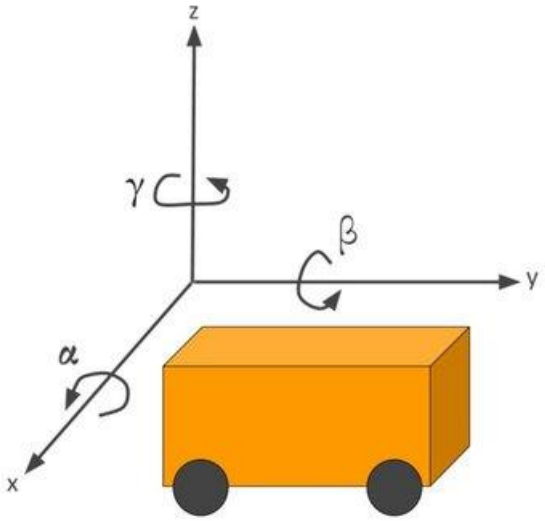
$$VxL = \big(VxG * cos(\gamma)\big) + \big(VyG * sin(\gamma)\big) (1.11)$$
$$VyL = -\big(VxG * sin(\gamma)\big) + \big(VyG * cos(\gamma)\big) (1.12)$$

Σε μορφή πίνακα(1.13), παρατίθενται:

$$\begin{bmatrix} Vx_L \\ Vy_L \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{bmatrix} Vx_G \\ Vy_G \end{bmatrix} (1.13)$$

1.10.2 Περιστροφή ενός ρομπότ σε 3D

Για να περιγραφεί η περιστροφή ενός ρομπότ σε τρεις διαστάσεις, είναι αναγκαίο να χρησιμοποιηθεί ο τρισδιάστατος πίνακας περιστροφής.



Εικόνα 1.6 – Ρομπότ σε 3D

Στις δύο διαστάσεις, η μόνη γωνία που απασχολεί τον ερευνητή είναι η γ. Η γ αντιπροσωπεύει το ποσό της περιστροφής (σε μοίρες ή ακτίνια) του άξονα x του ρομπότ από τον παγκόσμιο άξονα x, με φορά αντίθετη προς τη φορά των δεικτών του ρολογιού.

Η γ είναι η γωνία περιστροφής γύρω από τον άξονα z.

γωνία περιστροφής γύρω από τον άξονα x = roll angle = α

γωνία περιστροφής γύρω από τον άξονα y = pitch angle = β

Πώς προκύπτει, λοιπόν, ο τρισδιάστατος πίνακας περιστροφής; Για να απαντηθεί αυτό το ερώτημα, πρέπει να υπολογιστεί ο πίνακας περιστροφής για όλες τις περιστροφές που μπορεί να κάνει ένα ρομπότ σε έναν τρισδιάστατο χώρο. Δεδομένου ότι υπάρχουν τρεις άξονες, υπάρχουν τρεις

διαφορετικές περιστροφές που πρέπει να υπολογιστούν: περιστροφή γύρω από τον άξονα x, περιστροφή γύρω από τον άξονα y και περιστροφή γύρω από τον άξονα z.

Yaw (Περιστροφή γύρω από τον άξονα z)

Οι υπολογισμοί(1.14) ξεκινούν με την περιστροφή γύρω από τον άξονα z.

$$\begin{bmatrix} X_G \\ Y_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \end{bmatrix} \quad (1.14)$$

Εδώ πρόκειται για τη δισδιάστατη περίπτωση. Οι συντεταγμένες x και y ενός σημείου στο πλαίσιο αναφοράς του ρομπότ θα αλλάξουν όταν μετατραπούν σε συντεταγμένες στο παγκόσμιο πλαίσιο αναφοράς. Ωστόσο, η συντεταγμένη z θα παραμείνει σταθερή.

Εδώ παρατίθεται ο πίνακας (1.15) περιστροφής που φροντίζει για την περιστροφή ενός ρομπότ σε 3D γύρω από τον παγκόσμιο άξονα z:

YAW

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad (1.15)$$

Pitch (Περιστροφή γύρω από τον άξονα y)

Εν συνεχεία, η περιστροφή του σημείου γίνεται γύρω από τον άξονα y, με αποτέλεσμα οι συντεταγμένες x και z του σημείου να αλλάξουν, αλλά η συντεταγμένη y να παραμείνει η ίδια.

Ο ακόλουθος πίνακας(1.16) περιστροφής επιτρέπει τη μετατροπή ενός σημείου (ή διανύσματος) στο τοπικό σύστημα αναφοράς σε ένα σημείο (ή διάνυσμα) στο παγκόσμιο σύστημα αναφοράς, όταν το μόνο που συμβαίνει είναι η περιστροφή του ρομπότ γύρω από τον παγκόσμιο άξονα y.

PITCH

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad (1.16)$$

Roll (Περιστροφή γύρω από τον άξονα x)

Τέλος, όταν γίνει η περιστροφή ενός σημείου γύρω από τον άξονα x, οι συντεταγμένες y και z του σημείου θα αλλάξουν, αλλά η συντεταγμένη x θα παραμείνει η ίδια.

Εδώ παρατίθεται ο πίνακας(1.17) περιστροφής που επιτρέπει τη μετατροπή ενός σημείου (ή διανύσματος) στο τοπικό σύστημα αναφοράς σε ένα σημείο (ή διάνυσμα) στο παγκόσμιο σύστημα αναφοράς, όταν το μόνο που συμβαίνει είναι η περιστροφή του ρομπότ γύρω από τον παγκόσμιο άξονα x.

ROLL

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) \\ 0 & \sin(a) & \cos(a) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad (1.17)$$

Όλα μαζί

Συνδυάζοντας τους τρεις πίνακες που παρατέθηκαν παραπάνω, μπορεί να γίνει ο υπολογισμός του πλήρους τρισδιάστατου πίνακα περιστροφής(1.18).

YAW **PITCH** **ROLL**

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) \\ 0 & \sin(a) & \cos(a) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix}$$

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & \sin(\beta)\sin(\alpha) & \sin(\beta)\cos(\alpha) \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ -\sin(\beta) & \cos(\beta)\sin(\alpha) & \cos(\beta)\cos(\alpha) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad (1.18)$$

$$\begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\gamma)\cos(\beta) & \cos(\gamma)\sin(\beta)\sin(\alpha)-\sin(\gamma)\cos(\alpha) & \cos(\gamma)\sin(\beta)\cos(\alpha)+\sin(\gamma)\sin(\alpha) \\ \sin(\gamma)\cos(\beta) & \sin(\gamma)\sin(\beta)\sin(\alpha)+\cos(\gamma)\cos(\alpha) & \sin(\gamma)\sin(\beta)\cos(\alpha)-\cos(\gamma)\sin(\alpha) \\ -\sin(\beta) & \cos(\beta)\sin(\alpha) & \cos(\beta)\cos(\alpha) \end{bmatrix} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix}$$

1.11 Πίνακες περιστροφής

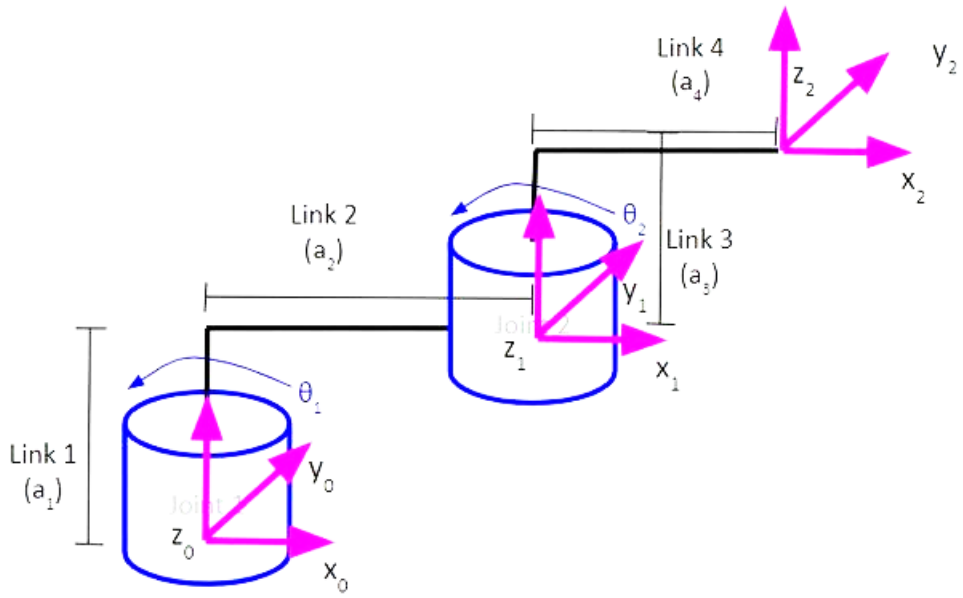
Στη ρομποτική, ο προσανατολισμός ενός ρομποτικού συστήματος μπορεί να αναπαρασταθεί με μαθηματικούς όρους χρησιμοποιώντας πίνακες περιστροφής. Οι πίνακες περιστροφής μετασχηματίζουν τους άξονες συντεταγμένων (π.χ. x, y και z) που αντιπροσωπεύουν τον προσανατολισμό ενός τρισδιάστατου αντικειμένου σε ένα πλαίσιο στους άξονες συντεταγμένων ενός άλλου πλαισίου.

Στην περίπτωση ενός ρομποτικού βραχίονα δύο βαθμών ελευθερίας, υπάρχουν τρία πλαίσια συντεταγμένων που αποτελούν τα πλαίσια αναφοράς.

Το βασικό πλαίσιο συντεταγμένων (άξονες x0, y0 και z0 στην Εικόνα 1.7) θα μπορούσε να είναι το παγκόσμιο πλαίσιο αναφοράς που ακολουθείται στην παρούσα εργασία.

Στη συνέχεια, υπάρχει ένα τοπικό πλαίσιο αναφοράς (x1, y1, z1), το οποίο περιστρέφεται κατά γωνία από το παγκόσμιο πλαίσιο αναφοράς.

Μετά από αυτό, ακολουθεί ένα άλλο τοπικό πλαίσιο αναφοράς (x2, y2, z2), το οποίο περιστρέφεται σε σχέση με το τοπικό πλαίσιο αναφοράς πριν από αυτό (x1, y1, z1).



Εικόνα 1.7 – Περιγραφή βραχίονα 2 βαθμών ελευθερίας

Καθώς αλλάζει το θ1, το πλαίσιο 1 θα περιστρέφεται γύρω από τον άξονα z0. Επομένως, αυτή η περιστροφή αναπαρίσταται χάρη στη χρήση του τυπικού πίνακα περιστροφής z. Επιπλέον, αυτός ο πίνακας πρέπει να πολλαπλασιαστεί με τον πίνακα που αναπαριστά την προβολή του πλαισίου 1 στο πλαίσιο 0 όταν η κοινή μεταβλητή είναι θ1 = 0 μοίρες(1.19).

$$rot\_mat\_0\_1 = \begin{bmatrix} \cos\Theta_1 & \sin\Theta_1 & 0 \\ \sin\Theta_1 & \cos\Theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta_1 & \sin\Theta_1 & 0 \\ \sin\Theta_1 & \cos\Theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1.19)$$

Ακολουθώντας την ίδια λογική, ο πίνακας περιστροφής(1.20) μεταξύ του frame 1 και του frame 2 είναι:

$$rot\_mat\_0\_2 = \begin{bmatrix} \cos\Theta_2 & \sin\Theta_2 & 0 \\ \sin\Theta_2 & \cos\Theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} (1.20)$$

Πίνακας περιστροφής από το πλαίσιο 0 στο πλαίσιο 2 Όταν θ1 = 90 μοίρες

	x_2	y_2	z_2
x_0	0	-1	0
y_0	1	0	0
z_0	0	0	1

Ο παραπάνω πίνακας είναι ένας άλλος τρόπος αναπαράστασης ενός πίνακα περιστροφής, καθώς δείχνει την προβολή των αξόνων x2, y2 και z2 στους άξονες x0, y0 και z0 αντίστοιχα. Η λέξη «*προβολή*» χρησιμοποιείται εδώ, για να γίνει εμφανές ότι ένας πίνακας περιστροφής προβάλλει τους άξονες συντεταγμένων ενός πλαισίου αναφοράς στους άξονες συντεταγμένων ενός άλλου πλαισίου αναφοράς.

Εάν ένας άξονας σε ένα πλαίσιο δείχνει προς την ίδια κατεύθυνση με έναν άξονα σε ένα άλλο πλαίσιο, τοποθετείται το σύμβολο 1 στο αντίστοιχο κελί του πίνακα.

Εάν ένας άξονας σε ένα πλαίσιο δείχνει προς την αντίθετη κατεύθυνση από έναν άξονα σε ένα άλλο πλαίσιο, τοποθετείται το σύμβολο -1 στο αντίστοιχο κελί του πίνακα.

Εάν ένας άξονας σε ένα πλαίσιο είναι κάθετος σε έναν άξονα σε ένα άλλο πλαίσιο, τοποθετείται το σύμβολο 0 στο αντίστοιχο κελί του πίνακα.

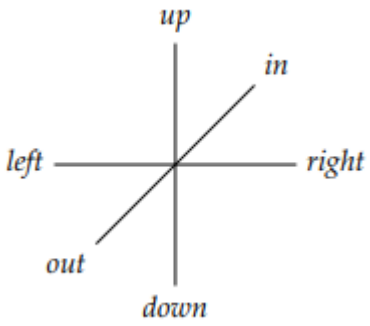
1.12 Περιστροφές σε τρεις διαστάσεις

Η έννοια των μετασχηματισμών συντεταγμένων στις τρεις διαστάσεις είναι η ίδια όπως και στις δύο διαστάσεις. Ωστόσο, η μαθηματική τους απεικόνιση είναι πιο περίπλοκη. Επιπλέον, πολλοί άνθρωποι δυσκολεύονται να οπτικοποιήσουν εξηγήσεις της τρισδιάστατης κίνησης. Ο μεγάλος μαθηματικός του 18^{ου} αιώνα Leonhard Euler απέδειξε ότι μία αυθαίρετη τρισδιάστατη περιστροφή μπορεί να προκύψει από τρεις μεμονωμένες περιστροφές γύρω από τους άξονες. Προς τιμήν του, οι γωνίες των περιστροφών ονομάζονται γωνίες Euler.

Περιστροφές γύρω από τους τρεις άξονες

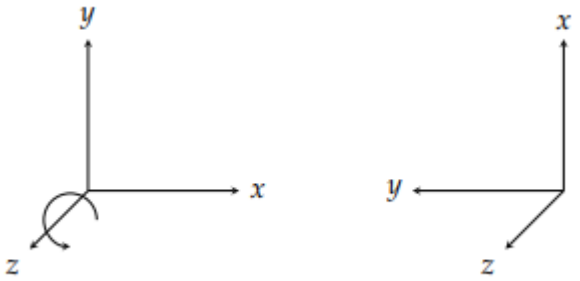
Ένα δισδιάστατο σύστημα συντεταγμένων x-y μπορεί να θεωρηθεί μέρος του τρισδιάστατου συστήματος συντεταγμένων με την προσθήκη ενός άξονα z κάθετου στους άξονες x και y.

Ακολουθεί μία δισδιάστατη αναπαράσταση του τρισδιάστατου πλαισίου συντεταγμένων:



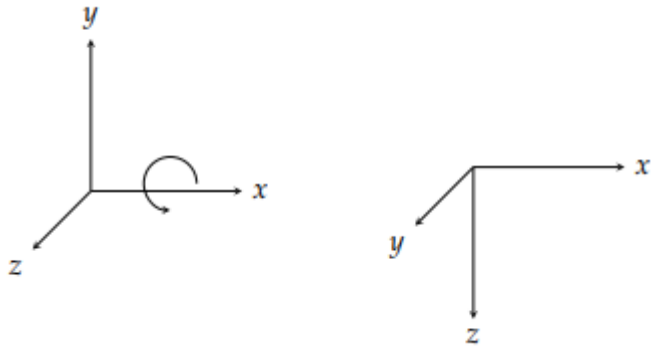
Εικόνα 1.8 – Άξονες τρισδιάστατου συστήματος συντεταγμένων

Ο άξονας x σχεδιάζεται αριστερά και δεξιά στο χαρτί και ο άξονας y σχεδιάζεται πάνω και κάτω στο χαρτί. Η διαγώνια γραμμή αντιπροσωπεύει τον άξονα z, ο οποίος είναι κάθετος στις άλλες δύο γραμμές - άξονες και επομένως η θετική του κατεύθυνση είναι έξω από το χαρτί και η αρνητική του κατεύθυνση είναι μέσα στο χαρτί. Το τυπικό σύστημα συντεταγμένων x-y-z έχει τις θετικές κατευθύνσεις των αξόνων του να δείχνουν προς τις κατευθύνσεις «δεξιά, πάνω, έξω», όπως φαίνεται στο αριστερό διάγραμμα(Εικόνα 1.9 –(α)):



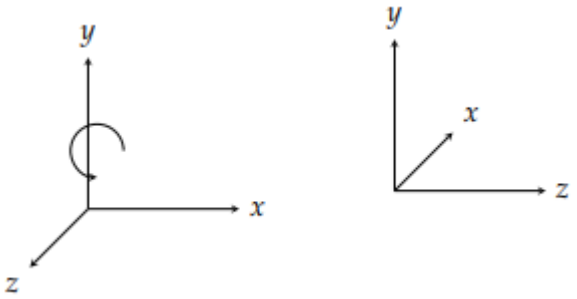
Εικόνα 1.9 –(α) Πλαίσιο συντεταγμένων x-y-z (β) Περιστροφή του πλαισίου

Η περιστροφή του πλαισίου συντεταγμένων γίνεται αριστερόστροφα γύρω από τον άξονα z, έτσι ώστε ο άξονας z να παραμένει αμετάβλητος (Εικόνα 1.9 –(β)). Ο νέος προσανατολισμός του πλαισίου είναι «πάνω, αριστερά, έξω». Στο παρακάτω διάγραμμα(Εικόνα 1.10) εξετάζεται μία περιστροφής 90◦ γύρω από τον άξονα x:



Εικόνα 1.10 – Πλαίσιο συντεταγμένων x-y-z περιστροφή 90◦ γύρω από τον άξονα x

Τέλος, στο διάγραμμα (Εικόνα 1.11) μία περιστροφή 90◦ γύρω από τον άξονα y:

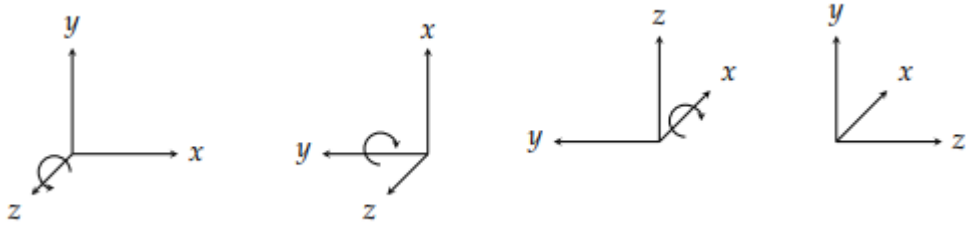


Εικόνα 1.11 – Πλαίσιο συντεταγμένων x-y-z περιστροφή 90◦ γύρω από τον άξονα y

Ο άξονας x "πέφτει κάτω" από το χαρτί και ο άξονας z "πέφτει ακριβώς" πάνω στο χαρτί. Η νέα θέση του πλαισίου είναι «μέσα, πάνω, δεξιά».

Πολλαπλές περιστροφές

Ακολουθεί ένα παράδειγμα(Εικόνα 1.12) τριών διαδοχικών περιστροφών ενός συνδιαμορφωμένου πλαισίου: πρώτα, 90◦ γύρω από τον άξονα z, στη συνέχεια 90◦ γύρω από τον άξονα y και τέλος 90◦ γύρω από τον άξονα x:



Εικόνα 1.12 – Πολλαπλές περιστροφές

Αυτό ονομάζεται περιστροφή γωνίας Euler zyx. Ο τελικός προσανατολισμός είναι «μέσα, πάνω, δεξιά».

Υπάρχουν τρεις άξονες, οπότε θα πρέπει να υπάρχουν 3^3 = 27 ακολουθίες γωνιών Euler. Ωστόσο, δεν έχει νόημα να περιστραφούμε γύρω από τον ίδιο άξονα δύο φορές διαδοχικά, επειδή το ίδιο αποτέλεσμα μπορεί να προκύψει περιστρέφοντας μία φορά κατά το άθροισμα των γωνιών, οπότε υπάρχουν μόνο 3 - 2 - 2 - 2 = 12 διαφορετικές ακολουθίες γωνιών Euler:

xyx xyz xzx xzy

yxy yxz yzx yzy

zxy zxz zyx zyz

1.13 Πίνακας παραμέτρων Denavit-Hartenberg

Σε αυτήν την ενότητα, φαίνεται πως βρίσκεται ο πίνακας παραμέτρων Denavit-Hartenberg για ρομποτικούς βραχίονες. Αυτή η μέθοδος είναι μία συντόμευση για την εύρεση ομογενών πινάκων μετασχηματισμού. Οι πίνακες ομογενών μετασχηματισμών επιτρέπουν να εκφραστεί η θέση και ο προσανατολισμός του πλαισίου του end-effector σε σχέση με τη βάση του ρομπότ.

Για να υπολογιστεί ο ομογενής πίνακας μετασχηματισμού από το πλαίσιο βάσης στο πλαίσιο του end-effector, οι μόνες τιμές που χρειάζονται είναι το μήκος κάθε συνδέσμου και η γωνία κάθε σερβοκινητήρα.

Για την εύρεση του πίνακα παραμέτρων Denavit-Hartenbeg και των ομογενών πινάκων μετασχηματισμού για έναν βραχίονα, ακολουθούνται τα εξής τρία βήματα:

1. Σχεδιασμός του κινηματικού διαγράμματος σύμφωνα με τους τέσσερις κανόνες Denavit-Hartenberg.

2. Δημιουργία του πίνακα παραμέτρων Denavit-Hartenberg.

Αριθμός σειρών = Αριθμός πλαισίων - 1

Αριθμός στηλών = 4: Δύο στήλες για την περιστροφή και δύο στήλες για τη μετατόπιση

3. Εύρεση των πινάκων ομογενών μετασχηματισμών.

Ορισμός των παραμέτρων

Οι πίνακες παραμέτρων Denavit-Hartenberg αποτελούνται από τέσσερις μεταβλητές(Εικόνα 1.13):

Οι δύο μεταβλητές που χρησιμοποιούνται για την περιστροφή είναι οι θ και α.

Οι δύο μεταβλητές που χρησιμοποιούνται για τη μετατόπιση είναι οι r και d.

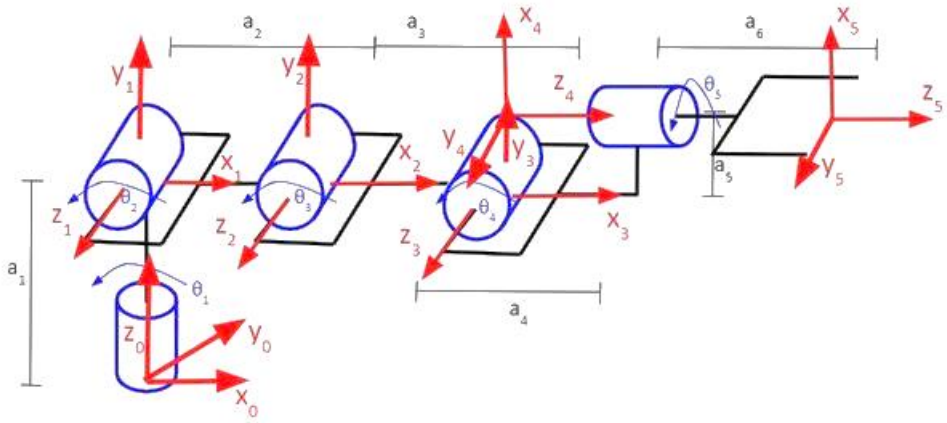
Ακολουθεί το πρότυπο πίνακα παραμέτρων D-H για έναν ρομποτικό βραχίονα με τέσσερα πλαίσια αναφοράς:

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1				
2				
3				

Εικόνα 1.13 – Πίνακας παραμέτρων D-H

Για έναν ρομποτικό βραχίονα 6 βαθμών ελευθερίας, χρειάζονται πέντε σειρές στον πίνακα παραμέτρων Denavit-Hartenberg.

- θ είναι η γωνία από το x_{n-1} στο x_n γύρω από το z_{n-1}.
- α είναι η γωνία από το z_{n-1} στο z_n γύρω από το x_n.
- r (μερικές φορές απεικονίζεται με το γράμμα "α" αντί του "r") είναι η απόσταση μεταξύ της αρχής του πλαισίου $n-1$ και της αρχής του πλαισίου n κατά μήκος της διεύθυνσης x_n.
- d είναι η απόσταση από το x_{n-1} στο x_n κατά μήκος της διεύθυνσης z_{n-1}.



Εικόνα 1.14 - Περιγραφή βραχίονα 6 βαθμών ελευθερίας

Ο πίνακας που προκύπτει είναι ο ακόλουθος(Εικόνα 1.15).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	90	0	a_1
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	$\theta_4 + 90$	90	a_5	0
5	θ_5	0	0	$a_4 + a_6$

Εικόνα 1.15 – Πίνακας παραμέτρων D-H για βραχίονα 6DOF

1.14 Κινηματικό διάγραμμα σύμφωνα με τους κανόνες Denavit-Hartenberg για SCARA Robot

Εικόνα 1.16 – Epson LS6-B SCARA Robot(epson.com)

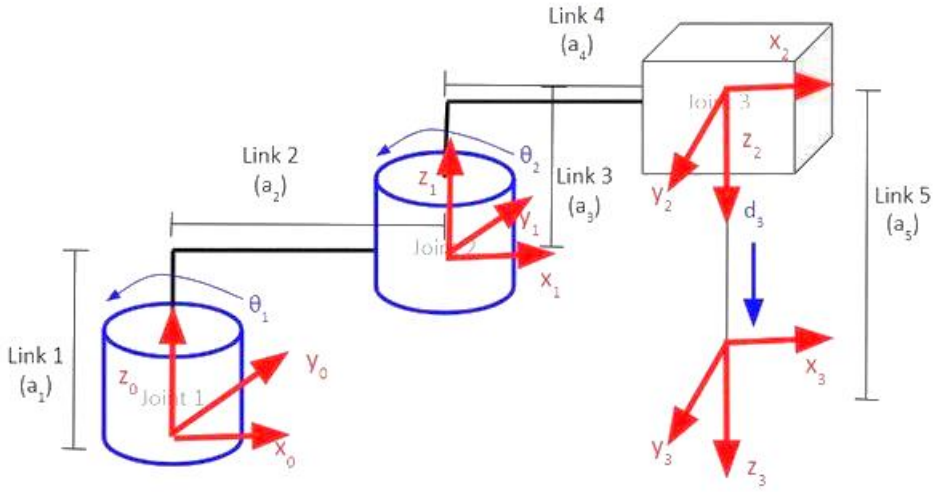
Δημιουργία του πίνακα παραμέτρων Denavit-Hartenberg

Αρχικά, βρίσκονται οι παράμετροι Denavit-Hartenberg. Εδώ υπάρχουν τέσσερα πλαίσια συντεταγμένων και τρεις σειρές που αφορούν τις αρθρώσεις(Εικόνα 1.17).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1				
2				
3				

Εικόνα 1.17 – Πίνακας παραμέτρων D-H Scara Robot

Εύρεση θ



Εικόνα 1.18 - Περιγραφή βραχίονα 6 Scara

Για το Servo 0, δίνεται έμφαση στη σχέση μεταξύ του πλαισίου 0 και του πλαισίου 1, ενώ θ είναι η γωνία από x0 σε x1 γύρω από το z0.

Στο διάγραμμα(Εικόνα 1.18), το x0 και το x1 δείχνουν και τα δύο προς την ίδια κατεύθυνση. Επομένως, οι άξονες είναι ευθυγραμμισμένοι. Όταν το ρομπότ βρίσκεται σε κίνηση, το θ1 θα αλλάξει (γεγονός που θα προκαλέσει την κίνηση του πλαισίου 1). Η γωνία από το x0 στο x1 γύρω από το z0 θα είναι θ1, οπότε καταγράφεται στον πίνακα (Εικόνα 1.19).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1			
2				
3				

Εικόνα 1.19 – Πίνακας παραμέτρων D-H Scara Robot εύρεση θ1

Στη συνέχεια, παρουσιάζεται το Servo 1. Παρατηρώντας την περιστροφή μεταξύ του frame 1 και του frame 2, θ είναι η γωνία από το x1 στο x2 γύρω από το z1.

Στο διάγραμμα, το x1 και το x2 δείχνουν και τα δύο προς την ίδια κατεύθυνση, με αποτέλεσμα οι άξονες να είναι ευθυγραμμισμένοι. Όταν το ρομπότ βρίσκεται σε κίνηση, το θ2 αλλάζει, γεγονός που προκαλεί την κίνηση του πλαισίου 2. Η γωνία από το x1 στο x2 γύρω από το z1 θα είναι θ2, οπότε τοποθετείται στη δεύτερη γραμμή του πίνακα(Εικόνα 1.20).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1			
2	θ_2			
3				

Εικόνα 1.20 – Πίνακας παραμέτρων D-H Scara Robot εύρεση θ2

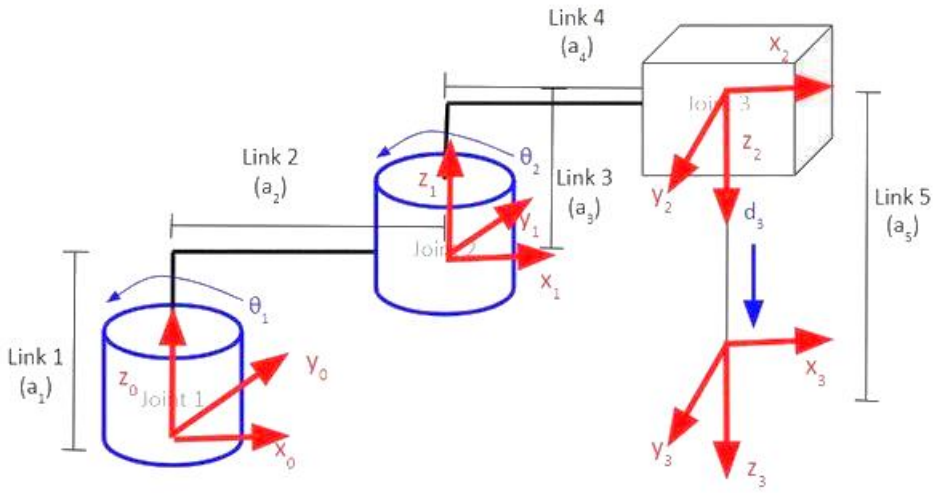
Συνεχίζοντας με το Servo 2, στην περιστροφή μεταξύ του καρέ 2 και του καρέ 3, θ είναι η γωνία από το x2 στο x3 γύρω από το z2.

Στο διάγραμμα, το x2 και το x3 δείχνουν και τα δύο προς την ίδια κατεύθυνση. Επομένως, οι άξονες είναι ευθυγραμμισμένοι. Όταν το ρομπότ κινείται, υπάρχει μόνο γραμμική κίνηση κατά μήκος του z2. Η γωνία από το x2 στο x3 γύρω από το z2 παραμένει 0, οπότε τοποθετείται στην τρίτη γραμμή του πίνακα(Εικόνα 1.21).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1			
2	θ_2			
3	0			

Εικόνα 1.21 – Πίνακας παραμέτρων D-H Scara Robot εύρεση θ3

Εύρεση του α



Εικόνα 1.22 - Περιγραφή βραχίονα Scara

Ξεκινώντας με το Servo 0 του πίνακα, **α είναι η γωνία από z0 σε z1 γύρω από το x1.**

Στο διάγραμμα, η γωνία είναι 0 μοίρες, οπότε τοποθετείται ο αριθμός 0 στον πίνακα(Εικόνα 1.23).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0		
2	θ_2			
3	0			

Εικόνα 1.23 – Πίνακας παραμέτρων D-H Scara Robot εύρεση α1

Στη σειρά 1 που αφορά το Servo 1 του πίνακα, το α είναι η γωνία από το z1 στο z2 γύρω από το x2. Η γωνία αυτή είναι 180 μοίρες, οπότε τοποθετείται ο αριθμός 180 στον πίνακα(Εικόνα 1.24).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0		
2	θ_2	180		
3	0			

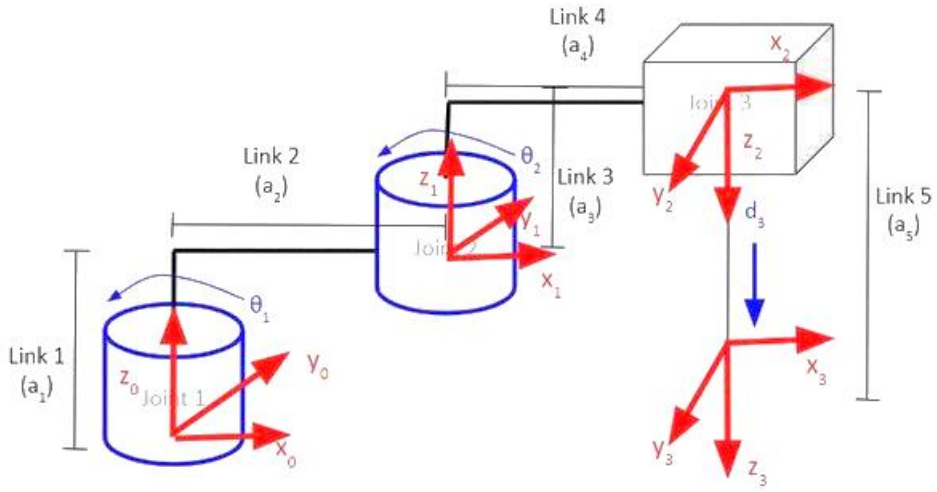
Εικόνα 1.24 – Πίνακας παραμέτρων D-H Scara Robot εύρεση α2

Στη σειρά 2 που αφορά το Servo 2 του πίνακα, το α είναι η γωνία από το z2 στο z3 γύρω από το x3. Η γωνία αυτή είναι 0 μοίρες, οπότε τοποθετείται ο αριθμός 0 στον πίνακα(Εικόνα 1.25).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0		
2	θ_2	180		
3	0	0		

Εικόνα 1.25 – Πίνακας παραμέτρων D-H Scara Robot εύρεση α3

Εύρεση του r



Εικόνα 1.26 – Περιγραφή βραχίονα Scara

Ξεκινώντας με τη γραμμή του πίνακα που αφορά το Servo 0, r είναι η απόσταση μεταξύ της αρχής του πλαισίου 0 και της αρχής του πλαισίου 1 κατά μήκος της διεύθυνσης x1. Στο διάγραμμα(Εικόνα 1.27) η απόσταση αυτή είναι a2.

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	
2	θ_2	180		
3	0	0		

Εικόνα 1.27 – Πίνακας παραμέτρων D-H Scara Robot εύρεση r1

Για τη σειρά που αφορά το Servo 1, το r είναι η απόσταση μεταξύ της αρχής του πλαισίου 1 και της αρχής του πλαισίου 2 κατά μήκος της διεύθυνσης x2. Στο διάγραμμα(Εικόνα 1.28) η απόσταση αυτή είναι a4.

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	
2	θ_2	180	a_4	
3	0	0		

Εικόνα 1.28 – Πίνακας παραμέτρων D-H Scara Robot εύρεση r2

Για τη σειρά που αφορά το Servo 2, το r είναι η απόσταση μεταξύ της αρχής του πλαισίου 2 και της αρχής του πλαισίου 3 κατά μήκος της διεύθυνσης x3. Στο διάγραμμα(Εικόνα 1.29) η απόσταση αυτή είναι 0.

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	
2	θ_2	180	a_4	
3	0	0	0	

Εικόνα 1.29 – Πίνακας παραμέτρων D-H Scara Robot εύρεση r3

Εύρεση του d

Το d είναι η απόσταση από το x0 στο x1 κατά μήκος της διεύθυνσης z0. Η απόσταση αυτή είναι a1(Εικόνα 1.30).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	a_1
2	θ_2	180	a_4	
3	0	0	0	

Εικόνα 1.30 – Πίνακας παραμέτρων D-H Scara Robot εύρεση d1

Από το πλαίσιο 1 στο πλαίσιο 2, το d είναι η απόσταση από το x1 στο x2 κατά μήκος της διεύθυνσης z1. Η απόσταση αυτή είναι a3(Εικόνα 1.31).

Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	a_1
2	θ_2	180	a_4	a_3
3	0	0	0	

Εικόνα 1.31 – Πίνακας παραμέτρων D-H Scara Robot εύρεση d2

Από το πλαίσιο 2 στο πλαίσιο 3, το d είναι η απόσταση από το x2 στο x3 κατά μήκος της διεύθυνσης z2. Η απόσταση αυτή είναι a5 + d3(Εικόνα 1.32).

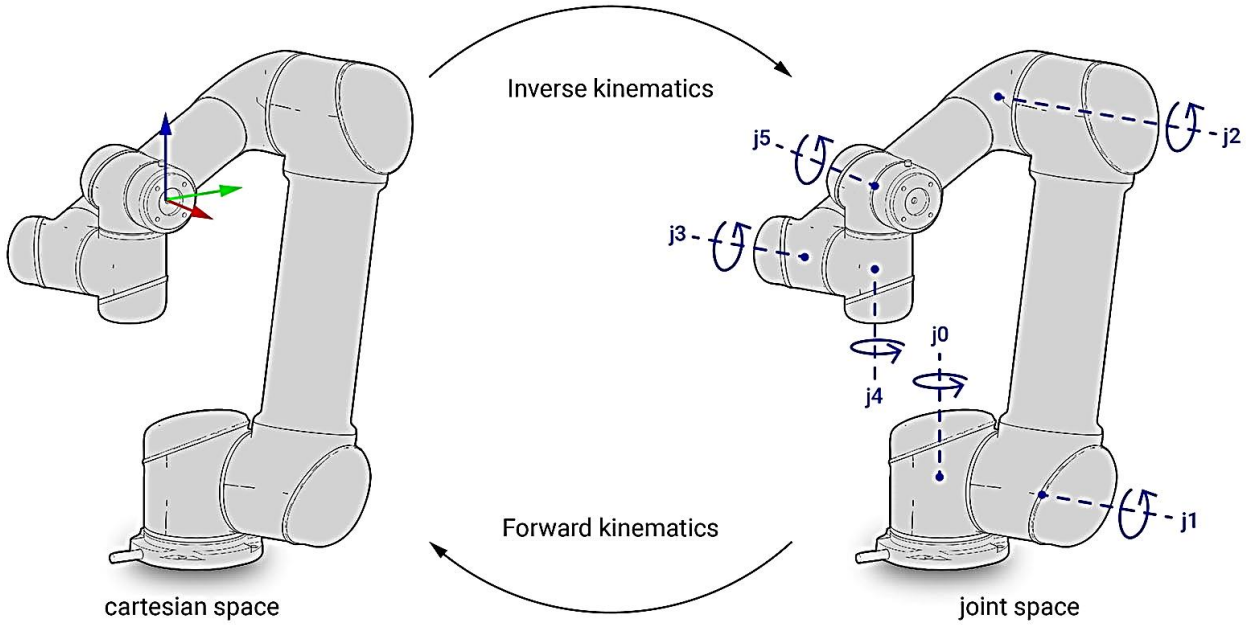
Joint i	θ_i (deg)	α_i (deg)	r_i (cm)	d_i (cm)
1	θ_1	0	a_2	a_1
2	θ_2	180	a_4	a_3
3	0	0	0	$a_5 + d_3$

Εικόνα 1.32 – Πίνακας παραμέτρων D-H Scara Robot εύρεση d3

1.15 Ορθή κινηματική (forward kinematics) και αντίστροφη κινηματική (inverse kinematics)

Σε αυτό το υποκεφάλαιο προσεγγίζεται το θεωρητικό υπόβαθρο που αφορά το ορθό και το αντίστροφο κινηματικό πρόβλημα. Γίνεται ο ορισμός του κάθε προβλήματος ξεχωριστά και παρουσιάζεται ο τρόπος επίλυσης τους. Για την ορθή κινηματική χρησιμοποιείται η μέθοδος Denavit-Hartenberg ενώ για την αντίστροφη κινηματική χρησιμοποιείται η αλγεβρική λύση της.

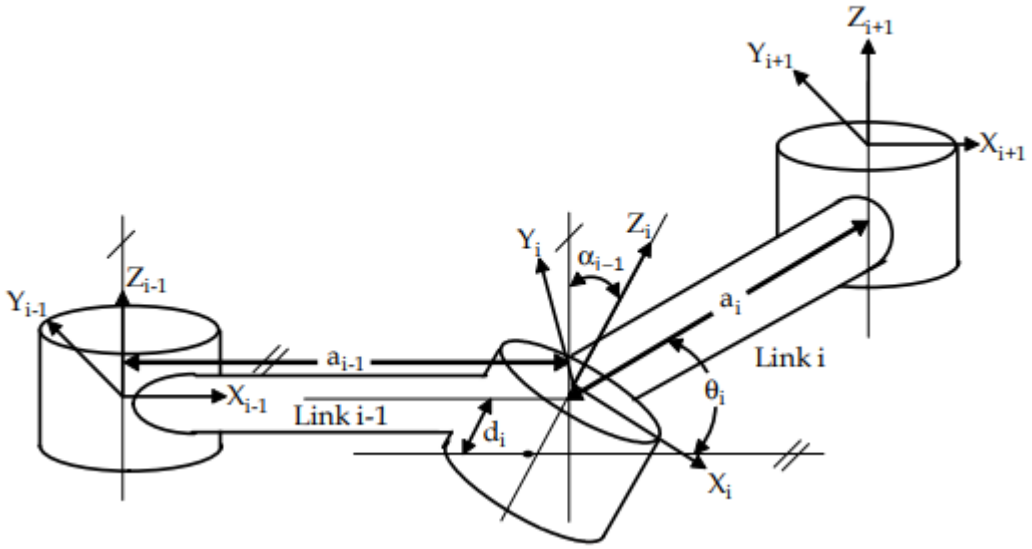
1.15.1 Ορθή κινηματική (forward kinematics)



Εικόνα 1.32 – Inverse kinematics και Forward kinematics (www.making.unsw.edu.au/)

Ένας βραχίονας αποτελείται από συνδέσμους, οι οποίοι συνδέονται μεταξύ τους με περιστροφικές ή πρισματικές αρθρώσεις από το πλαίσιο βάσης έως τον end-effector. Ο υπολογισμός της θέσης και του προσανατολισμού του end-effector ως προς τις μεταβλητές των αρθρώσεων ονομάζεται «*ορθή κινηματική*». Προκειμένου να βρεθεί η ορθή κινηματική για ένα ρομπότ, πρέπει να χρησιμοποιηθεί το κατάλληλο κινηματικό μοντέλο. Η μέθοδος Denavit-Hartenberg, η οποία χρησιμοποιεί τέσσερις

παραμέτρους, είναι η πιο συνήθης μέθοδος για την περιγραφή της κινηματικής ενός ρομπότ. Αυτές οι παράμετροι ai-1,α -1i, di και θi είναι το μήκος του συνδέσμου, η περιστροφή του συνδέσμου, η μετατόπιση του συνδέσμου και η γωνία της άρθρωσης αντίστοιχα (Εικόνα 1.33). Σε κάθε άρθρωση προσαρτάται ένα πλαίσιο συντεταγμένων για τον προσδιορισμό των παραμέτρων DH. Ο άξονας Zi του πλαισίου συντεταγμένων δείχνει κατά μήκος της κατεύθυνσης περιστροφής ή της ολίσθησης των αρθρώσεων.



Εικόνα 1.33 – Inverse kinematics και Forward kinematics (www.semanticscholar.org/)

Όπως φαίνεται στην Εικόνα 1.33, η απόσταση από το Zi-1 στο Zi που μετράται κατά μήκος του Xi-1 αποδίδεται ως ai-1, η γωνία μεταξύ Zi-1 και Zi που μετράται κατά μήκος του Xi αποδίδεται ως αi-1, η απόσταση από το Xi-1 στο Xi μετρούμενη κατά μήκος του Zi αποδίδεται ως di και η γωνία μεταξύ Xi-1 και Xi μετρούμενη γύρω από το Zi αποδίδεται ως θi. Ο γενικός πίνακας μετασχηματισμού ${}^{i-1}_{i}T$ για έναν μεμονωμένο σύνδεσμο μπορεί να προκύψει ως εξής(1.21).

$$
\begin{aligned}
{}^{i-1}_{i}\mathrm{T} &= R_x(a_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & ca_{i-1} & -sa_{i-1} & 0 \\ 0 & sa_{i-1} & ca_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.21) \\
&= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i ca_{i-1} & c\theta_i ca_{i-1} & -sa_{i-1} & -sa_{i-1}d_i \\ s\theta_i sa_{i-1} & c\theta_i sa_{i-1} & ca_{i-1} & ca_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

όπου τα Rx και Rz δηλώνουν την περιστροφή, τα Dx και Qi τη μετατόπιση και τα cθi και sθi είναι οι συντομεύσεις των cosθi και sinθi αντίστοιχα. Η ορθή κινηματική του end-effector σε σχέση με το πλαίσιο βάσης προσδιορίζεται πολλαπλασιάζοντας όλα τα ${}^{i-1}_{i}T$ των πινάκων(1.22).

$$
{}^{base}_{end\_effector}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{n-1}_{n}T \quad (1.22)
$$

1.16 Αντίστροφη κινηματική (inverse kinematics)

Το πρόβλημα της αντίστροφης κινηματικής των σειριακών ρομπότ έχει μελετηθεί εδώ και πολλές δεκαετίες, καθώς είναι απαραίτητο για τον έλεγχο των ρομπότ. Η επίλυση της αντίστροφης κινηματικής είναι υπολογιστικά δαπανηρή και γενικά διαρκεί πολύ στον έλεγχο των ρομπότ σε πραγματικό χρόνο. Οι εργασίες που πρέπει να εκτελεστούν από ένα ρομπότ βρίσκονται στον καρτεσιανό χώρο, ενώ οι ενεργοποιητές λειτουργούν στον χώρο των αρθρώσεων. Ο καρτεσιανός χώρος περιλαμβάνει πίνακα προσανατολισμού και διάνυσμα θέσης. Ωστόσο, ο χώρος των αρθρώσεων αντιπροσωπεύεται από τις γωνίες των αρθρώσεων. Η μετατροπή της θέσης και του προσανατολισμού ενός end-effector από τον καρτεσιανό χώρο στον χώρο των αρθρώσεων ονομάζεται «*πρόβλημα αντίστροφης κινηματικής*». Υπάρχουν δύο προσεγγίσεις, η γεωμετρική και η αλγεβρική που χρησιμοποιούνται για την εύρεση της λύσης της αντίστροφης κινηματικής.

1.16.1 Αλγεβρική λύση

Για τους χειριστές με περισσότερους από 2 συνδέσμους και των οποίων ο βραχίονας εκτείνεται σε 3 διαστάσεις, η γεωμετρία γίνεται πολύ πιο δύσκολη. Ως εκ τούτου, επιλέγεται η αλγεβρική προσέγγιση για τη λύση της αντίστροφης κινηματικής. Για την εύρεση της λύσης αντίστροφης κινηματικής για έναν χειριστή έξι αξόνων, ανακαλείται η παρακάτω εξίσωση(1.23).

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6) \quad (1.23) $$

Για να βρεθεί η λύση της αντίστροφης κινηματικής για την πρώτη άρθρωση (q1) ως συνάρτηση των γνωστών στοιχείων της ${}^{base}_{end-effector}T$, ο αντίστροφος μετασχηματισμός συνδέσμου πολλαπλασιάζεται ως εξής(1.24).

$$ \left[{}^{0}_{1}T(q_1) \right]^{-1} {}^{0}_{6}T = \left[{}^{0}_{1}T(q_1) \right]^{-1} {}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6) \quad (1.24) $$

όπου το $[{}^{0}_{1}T(q_1)]^{-1}\,{}^{0}_{1}T(q_1) = I$ και το I είναι πίνακας ταυτότητας. Στην περίπτωση αυτή, η παραπάνω εξίσωση δίνεται από τη σχέση(1.25):

$$ \left[{}^{0}_{1}T(q_1) \right]^{-1} {}^{0}_{6}T = {}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6) \quad (1.25) $$

Για να βρεθούν οι υπόλοιπες μεταβλητές, προκύπτουν οι ακόλουθες εξισώσεις με παρόμοιο τρόπο(1.26).

$$\left[{}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\right]^{-1}\,{}^{0}_{6}T = {}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6)$$
$$\left[{}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\right]^{-1}\,{}^{0}_{6}T = {}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6)$$
$$\left[{}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\right]^{-1}\,{}^{0}_{6}T = {}^{4}_{5}T(q_5)\,{}^{5}_{6}T(q_6) \quad (1.26)$$
$$\left[{}^{0}_{1}T(q_1)\,{}^{1}_{2}T(q_2)\,{}^{2}_{3}T(q_3)\,{}^{3}_{4}T(q_4)\,{}^{4}_{5}T(q_5)\right]^{-1}\,{}^{0}_{6}T = {}^{5}_{6}T(q_6)$$

Υπάρχουν 12 μη γραμμικές εξισώσεις που πρέπει να επιλυθούν. Η μόνη άγνωστος στην αριστερή πλευρά των εξισώσεων είναι το q1. Τα 12 μη γραμμικά στοιχεία του πίνακα της δεξιάς πλευράς είναι είτε μηδενικά είτε σταθερά είτε συναρτήσεις των q2-q6. Εάν τα στοιχεία της αριστερής πλευράς που είναι συνάρτηση του q1 εξισώνονται με τα στοιχεία της δεξιάς πλευράς, τότε η κοινή μεταβλητή q1 μπορεί να επιλυθεί συναρτήσει των r11,r12, ... r33, px, py, pz και των σταθερών παραμέτρων του συνδέσμου. Αφού βρεθεί η q1, τότε οι άλλες κοινές μεταβλητές επιλύονται με τον ίδιο τρόπο. Δεν είναι απαραίτητο η πρώτη εξίσωση να παράγει q1 και η δεύτερη q2 κ.λπ. Για να βρεθεί η κατάλληλη εξίσωση για την επίλυση του προβλήματος αντίστροφης κινηματικής, μπορεί να χρησιμοποιηθεί οποιαδήποτε εξίσωση που ορίζεται παραπάνω. Ορισμένες τριγωνομετρικές εξισώσεις που χρησιμοποιούνται για την επίλυση του προβλήματος της αντίστροφης κινηματικής δίνονται παρακάτω.

$a\sin\theta + b\cos\theta = c \qquad \theta = \mathrm{A}\tan 2(a,b) \mp \mathrm{A}\tan 2(\sqrt{a^2+b^2-c^2}, c)$ (1.27)

$a\sin\theta + b\cos\theta = 0 \qquad \theta = \mathrm{A}\tan 2(-b,a) \quad \acute{\eta} \quad \theta = \mathrm{A}\tan 2(b,-a)$ (1.28)

$\cos\theta = \alpha, \sin\theta = b \qquad \theta = \mathrm{A}\tan 2(b,a)$ (1.29)

$\cos\theta = a \qquad \theta = \mathrm{A}\tan 2(\mp\sqrt{1-a^2}, a)$ (1.30)

$\sin\theta = a \qquad \theta = \mathrm{A}\tan 2(a, \mp\sqrt{1-a^2})$ (1.31)

1.17 Είδη βραχίονα σταθερής βάσης

Οι περισσότεροι σειριακοί βραχίονες που χρησιμοποιούνται στη βιομηχανία έχουν μεταξύ 4 και 6 βαθμούς ελευθερίας. Θεωρώντας τη σταθερή βάση ως σύνδεσμο 0 και τον πρώτο κινητό σύνδεσμο ως σύνδεσμο 1, τότε οι σύνδεσμοι με αριθμό 2, 3 και 4 αναφέρονται συνήθως στον βραχίονα. Οποιαδήποτε άλλα αρθρωτά τμήματα μετά τον βραχίονα αποτελούν τον καρπό. Πολύ συχνά, ο καρπός αποτελείται από τρεις περιστρεφόμενους άξονες των οποίων οι άξονες περιστροφής τέμνονται σε ένα σημείο. Αυτός ο τύπος καρπού είναι γνωστός ως σφαιρικός καρπός, επειδή παρέχει τρεις περιστροφικούς βαθμού ελευθερίας, όπως μία σφαιρική άρθρωση. Ο βραχίονας που παρουσιάζεται στην παρούσα εργασία έχει σφαιρικό καρπό. Όπως καταδεικνύεται, ένας σφαιρικός καρπός προσφέρει πολλές δυνατότητες όσον αφορά την Jacobian, το motion planning και τα inverse kinematics.

Αυτό οφείλεται στο γεγονός ότι οι 3 πρώτες αρθρώσεις ελέγχουν τη θέση του κέντρου του καρπού, ενώ οι τρεις τελευταίες ελέγχουν μόνο τον προσανατολισμό του end effector.

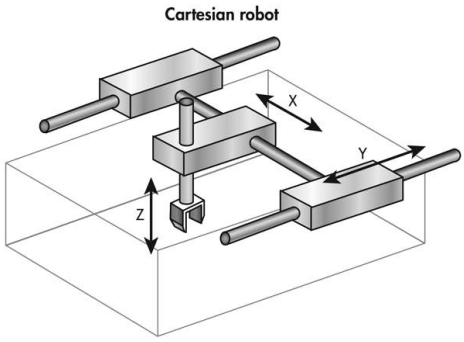
Ένας από τους τρόπους με τους οποίους ταξινομούνται οι βραχίονες είναι ως προς την κινηματική διαμόρφωση του βραχίονά τους, δηλαδή την ακολουθία των τύπων αρθρώσεων για τους τρεις πρώτους κινητούς συνδέσμους.

Με τη χρήση μόνο περιστροφικών και πρισματικών αρθρώσεων, υπάρχουν 8 πιθανοί τύποι βραχιόνων. Ωστόσο, μόνο 4 τύποι χρησιμοποιούνται συνήθως στη βιομηχανία.

1. Καρτεσιανής διαμόρφωσης (Prismatic-Prismatic-Prismatic)
2. Κυλινδρικής διαμόρφωσης (Revolute-Prismatic-Prismatic)
3. Σφαιρικής διαμόρφωσης (Revolute-Revolute-Prismatic)
4. Διαμόρφωσης SCARA (Revolute-Revolute-Prismatic)
5. Ανθρωπομορφικής διαμόρφωσης (Revolute-Revolute-Revolute)

Οι τύποι 3 και 4 έχουν δύο περιστροφικές και μία πρισματική άρθρωση, αλλά οι άξονες των αρθρώσεων είναι ευθυγραμμισμένοι διαφορετικά. Στην περίπτωση του ρομπότ SCARA και οι τρεις άξονες των αρθρώσεων είναι παράλληλοι.

1.17.1 Καρτεσιανής Διαμόρφωσης



Εικόνα 1.34 – Cartesian Robot(www.newequipment.com)

Κινείται πάνω στου άξονες x, z, y και χαρακτηρίζεται από υψηλή ακρίβεια και επαναληψιμότητα λόγω της στιβαρής του κατασκευής. Τα ρομπότ τύπου Ganty είναι ένα πολύ γνωστό και διαδεδομένο ρομπότ καρτεσιανής(Εικόνα 1.34) διαμόρφωσης που χρησιμοποιείται σε pick and place εφαρμογές.

Πλεονεκτήματα:

- Μπορεί να έχει πολύ υψηλή ακρίβεια θέσης
- Μεγάλα ωφέλιμα φορτία (Ganty)
- Απλή στρατηγική ελέγχου αφού δεν υπάρχουν περιστροφικές κινήσεις
-Άκαμπτη δομή

Μειονεκτήματα:

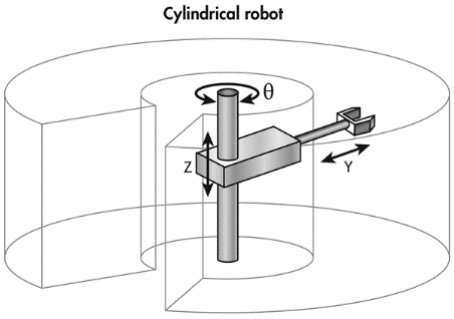
- Όλα τα εξαρτήματα και ο σχετικός εξοπλισμός πρέπει να βρίσκονται εντός του χώρου εργασίας του
- Απαιτεί μεγάλο χώρο λειτουργίας

Τυπικές εφαρμογές:

- Παλετοποίηση
- Βαριές εργασίες συναρμολόγησης

1.17.2 Κυλινδρικής διαμόρφωσης

Ο κύριος βραχίονας των κυλινδρικών ρομπότ(Εικόνα 1.35) ανεβοκατεβαίνει. Ένας κύλινδρος ενσωματωμένος στον ρομποτικό βραχίονα παράγει αυτήν την κίνηση, δηλαδή τεντώνεται και ανασύρεται.



Εικόνα 1.35 – Cylindrical Robot (www.newequipment.com)

Πλεονεκτήματα:

- Μεγάλο, εύκολο στην απεικόνιση του χώρου εργασίας
- Σχετικά φθηνό για το μέγεθος και το ωφέλιμο φορτίο του

Μειονεκτήματα:

- Χαμηλή μέση ταχύτητα
- Μικρότερη επαναληψιμότητα σε σχέση με το SCARA

Τυπικές εφαρμογές:

- Οι μικρές εκδόσεις χρησιμοποιούνται για συναρμολόγηση ακριβείας, οι μεγαλύτερες για φόρτωση/εκφόρτωση μηχανών

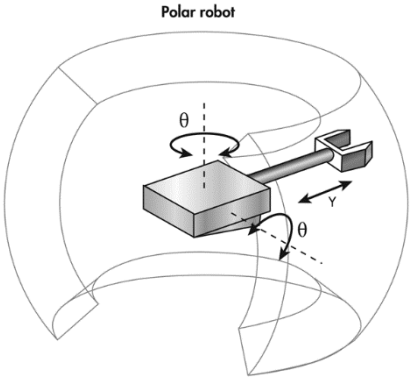
1.17.3 Σφαιρικής διαμόρφωσης

Ένας βραχίονας σφαιρικής διαμόρφωσης(Εικόνα 1.36) αποτελείται από τρεις περιστροφικές αρθρώσεις, των οποίων οι άξονες περιστροφής τέμνονται όλοι σε ένα κοινό σημείο.

Τα σφαιρικά ρομπότ έχουν έναν βραχίονα με δύο περιστροφικά joints και έναν γραμμικό joint που συνδέεται με τη βάση με έναν joint περιστροφής. Το ρομπότ έχει έναν σφαιρικό χώρο στον οποίο μπορεί να εργάζεται.

Πιθανώς, η πιο γνωστή εκδοχή αυτού του κινηματικού τύπου είναι το σύστημα του Stanford Scheinman, που εφευρέθηκε από τον Victor Scheinman το 1969. Προσαρμόστηκε από τους

κατασκευαστές για να γίνει το κορυφαίο ρομπότ στη συναρμολόγηση και τη συγκόλληση προϊόντων (από αντλίες καυσίμων μέχρι υαλοκαθαριστήρες για αυτοκίνητα).



Εικόνα 1.36 – Polar Robot
(www.newequipment.com)

Πλεονεκτήματα:

- Μεγάλος χώρος εργασίας

Μειονεκτήματα:

- Πολύπλοκες συντεταγμένες που είναι πιο δύσκολο να απεικονισθούν, να ελεγχθούν και να προγραμματιστούν
- Χαμηλή ακρίβεια
- Σχετικά αργό

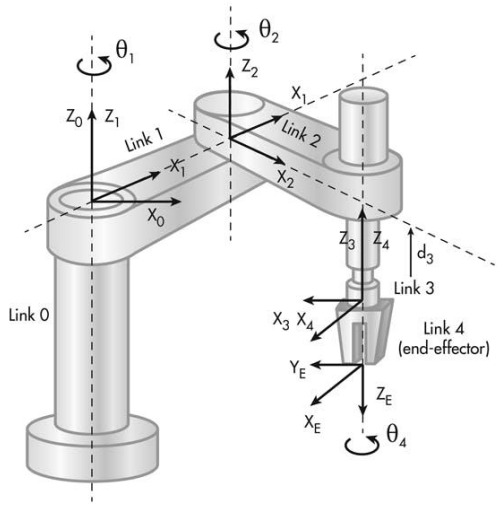
Τυπικές εφαρμογές:

- Μεταφορά υλικών
- Συγκόλληση

1.17.4 Διαμόρφωσης SCARA

Τα ρομπότ SCARA(Εικόνα 1.37) χρησιμοποιούνται κυρίως για εφαρμογές συναρμολόγησης. Ο βραχίονας, ο οποίος έχει κυλινδρικό σχεδιασμό, αποτελείται από δύο παράλληλα joint. Αυτά τα

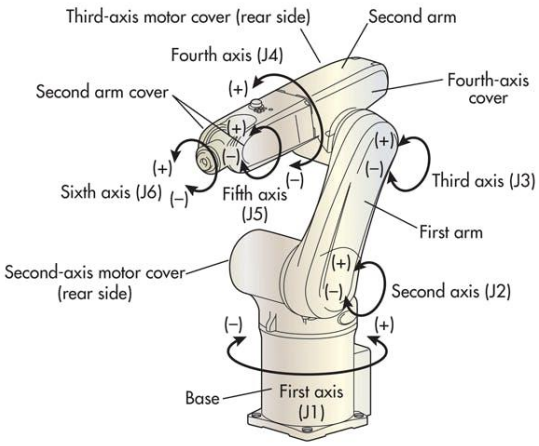
ρομπότ χρησιμοποιούνται για εφαρμογές pick-and-place, εφαρμογές σφραγίσματος, εφαρμογές συναρμολόγησης και χειρισμό διάφορων εργαλείων.



Εικόνα 1.37 – Scara Robot (www.newequipment.com)

1.17.5 Ανθρωπομορφικής διαμόρφωσης

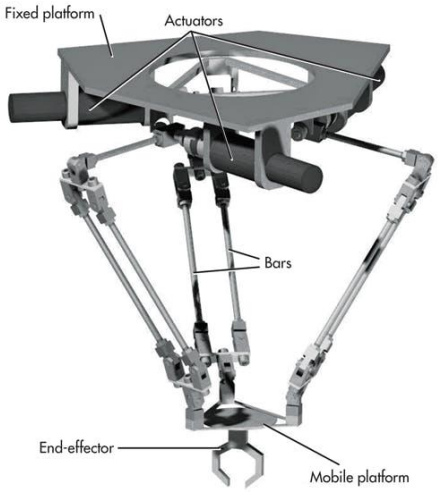
Το ανθρωπομορφικό ρομπότ(Εικόνα 1.38) μοιάζει πολύ με ένα ανθρώπινο χέρι. Ο βραχίονας είναι στερεωμένος σε μία βάση με ένα περιστρεφόμενο joint. Ο ίδιος ο βραχίονας μπορεί να διαθέτει από δύο περιστροφικά joint έως δέκα που λειτουργούν ως άξονες, ενώ με κάθε πρόσθετο link (σύνδεσμο) επιτρέπεται μεγαλύτερος βαθμός κίνησης. Τα περισσότερα αρθρωτά ρομπότ χρησιμοποιούν τέσσερα ή έξι joints. Τυπικές εφαρμογές για αρθρωτά ρομπότ είναι η συναρμολόγηση και η συσκευασία.



Εικόνα 1.38 – Anthropomorphic Manipulator(www.newequipment.com)

1.17.6 Διαμόρφωσης Delta

Ακόμη υπάρχουν τα ρομπότ δέλτα (Εικόνα 1.39) που χρησιμοποιούνται συνήθως για εφαρμογές pick-and-place (πχ. διαλογής φαρμάκων και τροφίμων) και ξεχωρίζουν λόγω της ταχύτητας και ακρίβειας που μπορούν να επιτύχουν.



Εικόνα 1.39 – Delta Robot(www.newequipment.com)

1.18 End-Effector

Οι ρομποτικοί βραχίονες χρειάζονται grippers ή end-effectors για να εκτελούν τα καθήκοντά τους. Λειτουργούν με την προσάρτηση στους βραχίονες χειρισμού ή στον καρπό ενός ρομπότ, γεγονός που επιτρέπει στον εξοπλισμό αυτόν να αλληλοεπιδρά με ένα αντικείμενο.

Οι χειριστές συχνά βασίζονται στους grippers για να σηκώνουν και να χειρίζονται ορισμένα αντικείμενα. Τα εξαρτήματα αυτά είναι ιδανικά για εφαρμογές συναρμολόγησης, pick-and-place και χειρισμού μηχανημάτων. Οι μηχανικοί χρησιμοποιούν τους end-effectors για να βελτιώσουν τα μέτρα ασφαλείας και να εξαλείψουν τον κίνδυνο γύρω από τη χρήση αυτών των μηχανημάτων. Οι τύποι end-effector που ακολουθούν στα υποκεφάλαια είναι και οι πιο διαδεδομένοι.

1.18.1 Ηλεκτρικές αρπάγες (Electric Grippers)

Οι ηλεκτρικές αρπάγες χρησιμοποιούν δάχτυλα με κινητήρα, τα οποία επιτρέπουν τον εύκολο έλεγχο της θέσης και της ταχύτητας. Πολλές εφαρμογές χρησιμοποιούν προσαρμοστικούς ηλεκτρικούς end-effectors, μεταξύ άλλων, όπως η εξυπηρέτηση μηχανημάτων, ο χειρισμός και η συλλογή κάδων.

1.18.2 Πνευματικές αρπάγες (Pneumatic Grippers)

Αυτές οι αρπάγες χρησιμοποιούν αέρα για να λειτουργήσουν, συνήθως με την πίεση πεπιεσμένου αέρα μέσω ενός εμβόλου. Οι πνευματικές αρπάγες επιτρέπουν γωνιακή ή παράλληλη κίνηση.

1.18.3 Βεντούζες (Suction Cups)

Οι βεντούζες χρησιμοποιούν κενό για τη λήψη εξαρτημάτων. Πέρα από οικονομική αποδοτικότητα, ο απλός σχεδιασμός τους προσφέρει άφθονη ευελιξία για τον χειρισμό υλικών. Ωστόσο, δεν μπορούν να χειριστούν διάτρητα υλικά.

1.18.4 Μαγνητικές αρπάγες (Magnetic Grippers)

Αυτές οι αρπάγες διαθέτουν σχεδιασμό παρόμοιο με τις βεντούζες, αλλά χρησιμοποιούνται ειδικά για τον χειρισμό σιδηρούχων υλικών. Επιπλέον, εξαλείφουν τον κίνδυνο πτώσης υλικών κατά τη διάρκεια διακοπής ρεύματος ή απώλειας αέρα και δε συνοδεύονται από κόστος αέρα. Επίσης, ο απλός σχεδιασμός τους απαιτεί ελάχιστη συντήρηση.

1.18.5 Μηχανικές αρπάγες (Mechanical Grippers)

Οι μη ηλεκτροκίνητες μηχανικές αρπάγες διαθέτουν συνήθως σχέδια για συγκεκριμένους τύπους εξαρτημάτων. Αυτές οι αρπάγες μπορεί να περιλαμβάνουν πιρούνια, άγκιστρα ή σύνθετα δάχτυλα που επιτρέπουν στους χειριστές να τα σφίγγουν και να τα περιστρέφουν.

1.19 Εφαρμογές End-Effector

1.19.1 Pick & Place

Συχνά βρίσκονται μεταξύ κάθε εργασίας σε μία γραμμή παραγωγής και για αυτό οι εφαρμογές pick-and-place αποτελούν ένα αναγκαίο κακό. Η χρήση τους ξεκινά από την παραλαβή πρώτων υλών, τεμαχίων, τελικών προϊόντων ή συσκευασιών και καταλήγει στην τοποθέτησή τους σε άλλο επίπεδο εργασίας για την ομαλή συνέχισή της. Η λειτουργία αυτή εκτελείται από ανθρώπους. Αυτοματοποιώντας αυτές τις εργασίες και απελευθερώνοντας ανθρώπινα χέρια, μεγιστοποιείται η απόδοση στην παραγωγή. Με πολλούς διαθέσιμους end-effectors, υπάρχει λύση για σχεδόν κάθε κατάσταση.

1.19.2 Χειρισμός Μηχανημάτων (Machine Tending)

Ο χειρισμός μηχανημάτων είναι μία από τις πιο διαδεδομένες χρήσεις των συνεργατικών ρομπότ στην αγορά. Μία τέτοια διαδικασία μπορεί να επαναλαμβάνεται ασταμάτητα, εφόσον το ρομπότ λαμβάνει συνεχώς ακατέργαστα υλικά.

Ορισμένες βιομηχανίες χρησιμοποιούν ρομπότ για ένα μόνο βήμα της παραγωγής, όπως το άδειασμα μηχανών χύτευσης ή μηχανών CNC. Όταν η παραγωγή λειτουργεί όλο το εικοσιτετράωρο, τα ρομπότ επιτρέπουν την ελαχιστοποίηση του χρόνου λειτουργίας και τη συνεχή λειτουργία αφαιρώντας εξαρτήματα από την περιοχή εργασίας της μηχανής.

Ο βασικός ορισμός του tending είναι η παροχή φροντίδας για κάποιον ή κάτι. Στην προκειμένη περίπτωση, machine tending σημαίνει φόρτωση ή/και εκφόρτωση μίας συγκεκριμένης μηχανής με εξαρτήματα ή υλικά. Επί του παρόντος, οι περισσότερες εφαρμογές φροντίδας μηχανών γίνονται από ανθρώπους. Τα σύγχρονα μηχανουργεία χρησιμοποιούν συχνά μηχανές CNC (όπως τόρνους και φρέζες). Αυτές οι μηχανές πρέπει να φροντίζονται από εργάτες, οι οποίοι τοποθετούν την πρώτη ύλη στη μηχανή και την αφαιρούν μόλις η μηχανή ολοκληρώσει την ανατιθέμενη εργασία. Ωστόσο,

δεδομένου ότι είναι όλο και πιο δύσκολο να βρεθούν εξειδικευμένοι εργαζόμενοι, οι εταιρείες εισάγουν ρομπότ στα εργαστήριά τους για να αναπληρώσουν την έλλειψη εργαζομένων.

1.19.3 Παλετοποίηση (Palletizing)

Παλετοποίηση ονομάζεται η ενέργεια της τοποθέτησης προϊόντων σε παλέτα για αποστολή ή αποθήκευση στις αλυσίδες εφοδιασμού των logistics. Ιδανικά, τα προϊόντα στοιβάζονται σε ένα μοτίβο που μεγιστοποιεί την ποσότητα του προϊόντος στο φορτίο βάσει βάρους και όγκου, ενώ είναι αρκετά σταθερό ώστε να αποτρέπεται η μετατόπιση, η ανατροπή ή η συντριβή των προϊόντων μεταξύ τους. Επί του παρόντος, οι περισσότερες εφαρμογές παλετοποίησης πραγματοποιούνται με χειρωνακτική εργασία. Μία αυτοματοποιημένη διαδικασία παλετοποίησης μπορεί να επαναλαμβάνεται ατελείωτα, εφόσον η ροή των κιβωτίων και των παλετών είναι σταθερή.

1.19.4 Συναρμολόγηση (Assembly)

Η συναρμολόγηση είναι μία κατασκευαστική διαδικασία κατά την οποία τα ρομπότ συναρμολογούν προϊόντα προσθέτοντας ένα προς ένα τα επιμέρους εξαρτήματα σε μία συγκεκριμένη σειρά. Κατά την εκτέλεση αυτών των εργασιών, ο end-effector του ρομπότ χειρίζεται αντικείμενα διαφόρων σχημάτων και μεγεθών.

1.19.5 Φινίρισμα (Finishing)

Είναι πλέον δυνατή η αυτοματοποίηση μερικών από τις πιο δύσκολες και βρώμικες εργασίες στην παραγωγή. Τα ρομπότ επιτρέπουν να προγραμματιστεί γρήγορα μία εφαρμογή φινιρίσματος κατά την οποία φέρνουν ένα εξάρτημα σε ένα περιστροφικό εργαλείο (τροχό). Ακόμη, υπάρχει η δυνατότητα αντιστάθμισης της φθοράς του εργαλείου χρησιμοποιώντας τον έλεγχο δύναμης του ρομπότ.

1.19.6 Έλεγχος Ποιότητας (Quality Testing)

Ο έλεγχος ποιότητας αποτελείται από μία σειρά ελέγχων που μπορεί να διαπιστώσει εάν υπάρχουν ελαττωματικά εξαρτήματα ή προϊόντα. Σε μία τέτοια περίπτωση η διαδικασία ελέγχου ποιότητας δημιουργεί ένα σήμα σφάλματος ενημερώνοντας τους εργαζομένους. Αυτού του είδους δοκιμές μπορεί να απαιτούν πολύ απλές λειτουργίες, όπως το πάτημα μίας ακολουθίας κουμπιών.

2 Εισαγωγή στο ROS

2.1 Εισαγωγή κεφαλαίου

Στο παρόν κεφάλαιο γίνεται μια περιγραφή του λογισμικού και της δομής του κώδικα που χρησιμοποιείται για την επίτευξη του στόχου της εργασίας.

2.2 Λειτουργικό Ubuntu

Το λειτουργικό Ubuntu είναι μια έκδοση των Linux που αναπτύχθηκε από την Canonical και είναι μια από τις πιο δημοφιλείς εκδόσεις, χάρη στην ευκολία χρήσης του. Το πιο σημαντικό είναι ότι προσφέρει υποστήριξη για το λειτουργικό σύστημα ROS που χρειάζεται linux για να εγκατασταθεί.

2.3 Τι είναι το ROS;

Το αρκτικόλεξο ROS σημαίνει *«Robot Operating System»*. Παρά την έννοια που αποδίδεται μέσα από τις συγκεκριμένες λέξεις, το ROS δεν είναι ένα πραγματικό λειτουργικό σύστημα, αφού λειτουργεί πάνω από το Linux Ubuntu – και αντίστοιχα πάνω από το Mac, ενώ πρόσφατα λειτούργησε και πάνω από τα Windows. Το ROS είναι ένα framework πάνω από το O.S. που του επιτρέπει να αφαιρέσει το hardware από το software. Παρέχει τις υπηρεσίες που είναι απαραίτητες για ένα τέτοιο λειτουργικό σύστημα και συμπεριλαμβάνει hardware abstraction του ελέγχου συσκευών χαμηλού επιπέδου (low-level), της διαχείρισης μηνυμάτων μεταξύ διεργασιών και της διαχείρισης πακέτων. Επίσης, παρέχει εργαλεία και βιβλιοθήκες για την προσομοίωση, την οπτικοποίηση και τη συγγραφή κώδικα. Τέλος, δίνει τη δυνατότητα πειραματισμού χωρίς να είναι απαραίτητη η φυσική κατασκευή του ρομπότ[6].

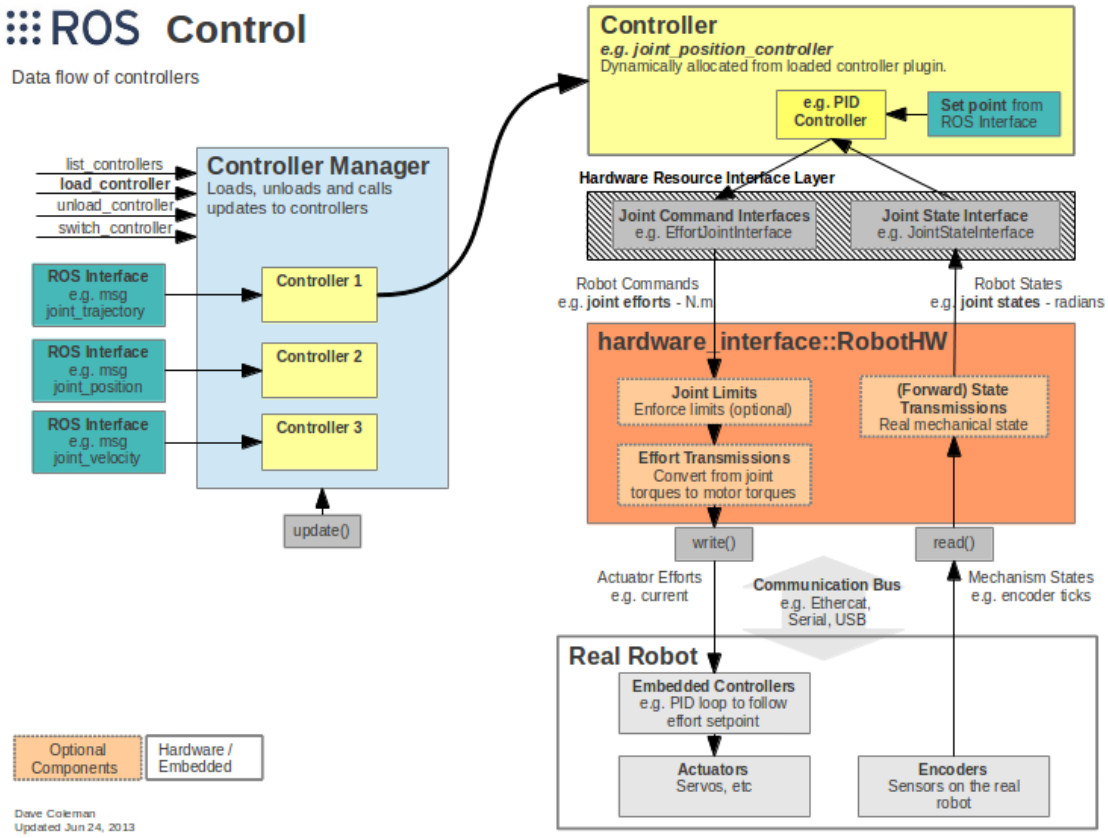
2.3.1 Σύντομη Ιστορική Αναδρομή

Το ROS υπάρχει από το 2006, αφού δημιουργήθηκε από το ερευνητικό εργαστήριο ρομποτικής Willow Garage (Keenan Wyrobek. Eric Berger). Αυτή η ομάδα ήταν συνδεδεμένη με τις πρώτες προσπάθειες του Stanford για την προώθηση της ρομποτικής και ωρίμασε μέσω της συμμετοχής στο Grand Challenge.

Το Willow Garage είναι γνωστό για την ανάπτυξη του ρομπότ PR2, το οποίο έχει γίνει πρότυπο για την έρευνα στη ρομποτική. Το PR2 είναι η ναυαρχίδα του ρομπότ που χρησιμοποιείται για το ROS. Παραδείγματος χάριν, 11 πανεπιστήμια διαθέτουν αυτό ακριβώς το ρομπότ προκειμένου να βοηθήσουν στην εκμάθηση και να αναπτύξουν την τυποποίηση του υλικού. Το Willow Garage διατηρεί, επίσης, τη βιβλιοθήκη Point Cloud Library (PCL) και το OpenCV, το οποίο έχει κρίσιμη σημασία για την ανίχνευση 2D και 3D (3D Perception).

Το Willow Garage εξελίχθηκε τελικά στο ίδρυμα Open Source Robotics Foundation (2014 - σήμερα). Αξίζει να σημειωθεί ότι το εν λόγω ίδρυμα συντηρεί το ROS και τον προσομοιωτή Gazebo, ο οποίος χρησιμοποιείται από εταιρείες και ερευνητικά εργαστήρια σε όλο τον κόσμο ως προσομοιωτής για τη ρομποτική[6].

2.4 Ros Control



Εικόνα 2.1 – ROS Control (ROS.org)

Τα πακέτα ros_control[18] είναι μια αναδιατύπωση των πακέτων pr2_mechanism για να κάνουν τους controllers λειτουργικούς για όλα τα ρομπότ πέρα από το PR2. Η επικοινωνία γίνεται με ROS messages μέσω topic ή service. Κάθε controller έχει μια καθορισμένη κατάσταση (loaded,unloaded και running) και διαχειρίζεται από τον Controller Manager. Οι ROS controllers δεν περιορίζονται στον έλεγχο ενός μόνο actuator. Αντίθετα, μπορούν να υλοποιήσουν σύνθετους μηχανισούς που ελέγχουν μια ολόκληρη ομάδα αρθρώσεων. Τα πακέτα ros_control λαμβάνουν ως δεδομένα εισόδου τα δεδομένα κατάστασης της άρθρωσης από τους encoders του actuator του ρομπότ. Χρησιμοποιεί έναν μηχανισμό ανατροφοδότησης βρόχου, συνήθως έναν ελεγκτή PID, για να ελέγξει την έξοδο, που αποστέλλεται στους actuators.

2.5 Nodes

Ένα node είναι μία διεργασία που εκτελεί υπολογισμούς. Τα nodes συνδυάζονται μεταξύ τους σε ένα γράφημα και επικοινωνούν μεταξύ τους χρησιμοποιώντας streaming topics, RPC services και τον Parameter Server. Ένα σύστημα ελέγχου ρομπότ συνήθως αποτελείται από πολλά nodes. Για παράδειγμα, ένα node ελέγχει ένα αποστασιόμετρο (laser), ένα άλλο node ελέγχει τους κινητήρες των τροχών ενός ρομπότ, ένα node εκτελεί path planning κ.ο.κ.

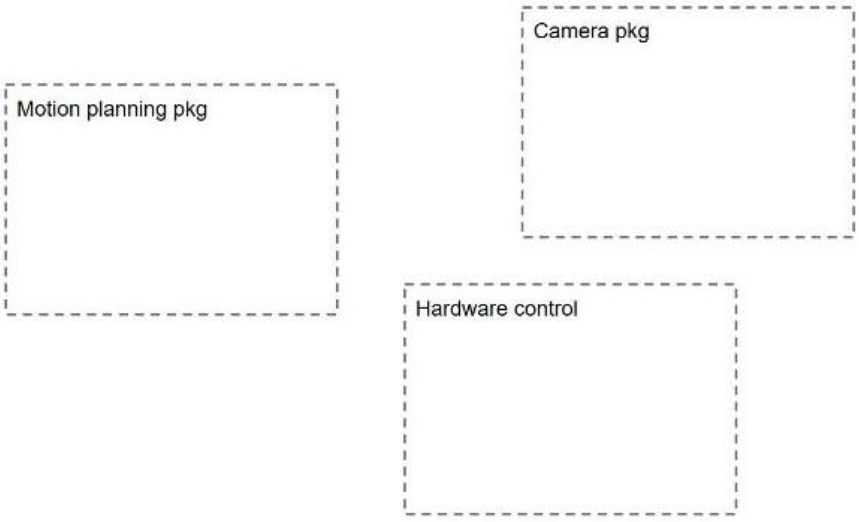
Η χρήση των nodes στο ROS παρέχει διάφορα πλεονεκτήματα στο σύστημα. Υπάρχει μεγάλη ανοχή σε σφάλματα, καθώς το ενδεχόμενο σφάλμα απομονώνεται σε μεμονωμένα nodes. Ως αποτέλεσμα, η πολυπλοκότητα του κώδικα μειώνεται, ενώ παράλληλα παρέχεται η ελευθερία χρήσης διαφορετικών γλωσσών προγραμματισμού. Έτσι, ένα node μπορεί να είναι γραμμένο σε Python, ενώ

ένα άλλο σε C++, χωρίς ωστόσο να εμποδίζεται η μεταξύ τους επικοινωνία. Ακόμη, εάν χρειαστεί να γραφτεί μία εφαρμογή σε Python, αλλά χρειάζεται γρηγορότερη ταχύτητα εκτέλεσης, είναι δυνατόν να χρησιμοποιηθεί η C++.

2.5.1 Παράδειγμα εφαρμογής Mobile Robot που ελέγχεται μέσω κάμερας

Τα τρία κύρια μέρη της εφαρμογής Mobile Robot(Εικόνα 2.2) είναι τα εξής πακέτα[9] :

1. Hardware control package: ελέγχει απευθείας το hardware (wheels, actuators)
2. Motion planning package: παρακολουθεί και ελέγχει την τροχιά του ρομπότ
3. Camera package: επεξεργάζεται εικόνες και δίνει χρήσιμες πληροφορίες και εντολές στο ρομπότ



Εικόνα 2.2 – Τρία κύρια μέρη της εφαρμογής Mobile Robot

Nodes για το camera package:

Αρχικά, ένας driver για την κάμερα είναι αναγκαίος, ώστε να λαμβάνονται frames από αυτήν έπειτα από τον κατάλληλο προγραμματισμό. Στη συνέχεια, απαιτείται και ένα πρόγραμμα που θα λαμβάνει αυτά τα frames και θα προχωρά στην επεξεργασία των εικόνων.

Nodes για το motion planning package:

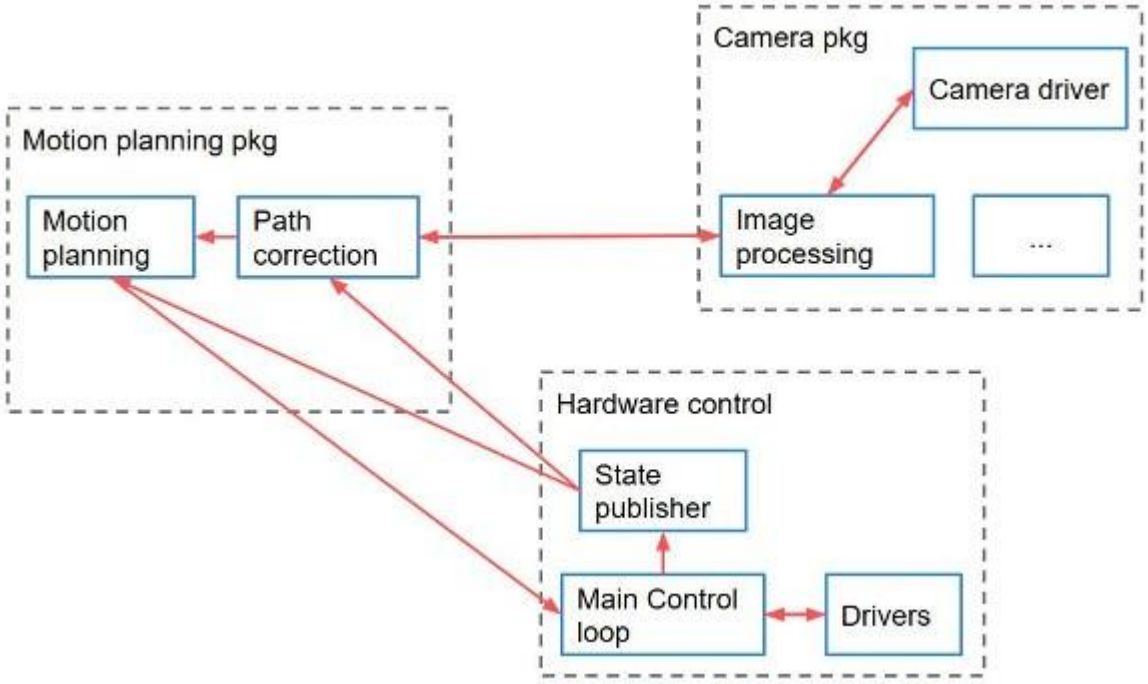
Σε αυτό το πακέτο υπάρχει ένα motion planning node, το οποίο σχεδιάζει το motion planning για το ρομπότ.

Ακόμη, γίνεται να προστεθεί ένας node διόρθωσης διαδρομής, ο ρόλος του οποίου είναι να τροποποιεί τον σχεδιασμό κίνησης λόγω εξωτερικών παραγόντων.

Nodes για το hardware control package:

Αυτό το πακέτο είναι χρήσιμο για να ελέγχει το hardware του ρομπότ και μπορεί να αποτελείται από ρόδες, αρθρώσεις ρομποτικού βραχίονα ή οτιδήποτε άλλο.

Σε αυτό το πακέτο υπάρχουν κάποιοι drivers για τον έλεγχο των κινητήρων. Οι drivers με τη σειρά τους ελέγχονται από τον κύριο control loop node. Αυτά τα δεδομένα δημοσιεύονται επίσης από το state publisher node (Εικόνα 2.3)[9].



Εικόνα 2.3 – Εφαρμογή Mobile Robot

Τα nodes ενώνονται σε ένα γράφημα (rqt_graph) και επικοινωνούν μεταξύ τους χρησιμοποιώντας ROS topics, services, actions κ.λπ.

2.5.2 rqt_graph tool

Το rqt_graph είναι ένα εργαλείο debugging στο ROS.

Σε αυτό το σημείο θα παρατεθεί ένα παράδειγμα, ώστε να φανεί ο τρόπος με τον οποίο λειτουργεί.

Σε ένα terminal τρέχει η εντολή:

rostopic pub /news std_msgs/String "data: 'news_pub_test'" -r1 __name:=news_pub

Αυτή η εντολή θα δημιουργήσει έναν publisher node με το όνομα new_pub, ο οποίος θα δημοσιεύει το string type message που ονομάζεται news_pub_test στο topic με όνομα /news.

Ακολουθεί η δημιουργία ενός subscriber node με το όνομα news_sub που κάνει subscribe στο topic news.

rostopic echo /news __name:=news_sub

Ο subscriber εκτυπώνει το μήνυμα(Εικόνα 2.4) που δημοσιεύτηκε στο published από το news_pub node.

```
^Corfeas@orfeas-B85M-D2V:~/catkin_ws$ rostopic echo /news __name:=news_sub
data: "news_pub_test"
---
data: "news_pub_test"
---
data: "news_pub_test"
---
data: "news_pub_test"
---
```

Εικόνα 2.4 – Μήνυμα subscriber

Με την τελευταία εντολή φαίνεται το rqt_graph:

rqt_graph

Απεικονίζεται η σχέση(Εικόνα 2.5) μεταξύ κάθε node και των topic με τα οποία επικοινωνούν.

Εικόνα 2.5 – Σχέση μεταξύ κάθε node και των topic

Ένα ρομπότ θα έχει πολύ περισσότερα nodes topics και services.

Topic: Είναι ένα bus μεταξύ των nodes, μέσω των οποίων μπορούν να περνούν μηνύματα. Για την αποστολή ενός message σε ένα topic, ένα node πρέπει πρώτα να κάνει publish σε ένα topic. Ομοίως, για να λάβει ένα message σε ένα topic, ένα node πρέπει να κάνει subscribe σε αυτό. Στον rqt graph τα βέλη αναπαριστούν τη ροή μηνυμάτων από τους publishers στους subscribers. Κάθε node μπορεί ταυτόχρονα να κάνει publish και subscribe σε διαφορετικά topic (Publish Subscribe architecture)[17].

Μεταβίβαση μηνυμάτων: Κάθε έκδοση του ROS συνοδεύεται από μία μεγάλη ποικιλία predefined messages. Αυτά μπορούν να περιλαμβάνουν messages για φυσικές ποσότητες όπως (Position, Velocity, Rotations) ή για αισθητήρες (Laser scans, Images, Point Clouds, Inertial Measurements). Ωστόσο, κάποια στιγμή θα χρειαστεί να οριστεί ένας τύπος messages που θα ανταποκρίνεται στις ανάγκες του μηχανικού. Αν και τα μηνύματα έχουν text based περιεχόμενο μπορούν στην πραγματικότητα να περιέχουν οποιοδήποτε είδος δεδομένων.

Services: Η μεταβίβαση μηνυμάτων τύπου Publisher/Subscriber είναι χρήσιμη, αλλά δεν είναι μία λύση επικοινωνίας που ταιριάζει σε όλες τις περιπτώσεις. Υπάρχουν φορές που το μοτίβο request-response είναι χρήσιμο. Για αυτούς τους τύπους αλληλεπιδράσεων το ROS παρέχει τα services.

Επιτρέπουν -όπως και τα topics- τη μεταβίβαση μηνυμάτων μεταξύ των nodes, αλλά σε αντίθεση με τα topics δεν είναι bus και δεν υπάρχουν Publisher και Subscriber. Αντίθετα, τα nodes αλληλοεπιδρούν μέσω services. Αυτό γίνεται σε βάσεις 1 προς 1 χρησιμοποιώντας ένα request response pattern.

Actions: Τα actions ROS έχουν μία σχέση επικοινωνίας client-to-server με ένα συγκεκριμένο πρωτόκολλο. Τα actions χρησιμοποιούν ROS topics για την αποστολή goal messages από έναν client στον server. Υπάρχει δυνατότητα ακύρωσης των στόχων (goal) χρησιμοποιώντας τον action client. Μετά τη λήψη ενός goal, ο server τον επεξεργάζεται και μπορεί να δώσει πληροφορίες πίσω στον client, οι οποίες είναι οι ακόλουθες:

- status of the server
- state of the current goal
- feedback on that goal during operation
- result message

[10]

2.5.3 Launch files και configuration files

Σε αυτό το σημείο προκύπτει η ανάγκη εκκίνησης πολλών διαφορετικών node σε πολλά διαφορετικά nodes σε πολλά διαφορετικά terminal. Το ROS παρέχει έναν μηχανισμό για την ταυτόχρονη εκκίνηση του master και πολλών nodes, χρησιμοποιώντας ένα αρχείο που ονομάζεται launch file. Η χρήση των launch files είναι ευρέως διαδεδομένη σε πολλά ROS packages. Κάθε εφαρμογή που χρησιμοποιεί περισσότερα από ένα ή δύο nodes έχει ανάγκη τα launch files, για να προσδιορίσει και να ρυθμίσει αυτά τα nodes.

Ένα launch file:

- ξεκινάει αυτόματα το ROS Master και πολλά nodes με μία εντολή
- μπορεί να ορίσει default paramaters στον paramater server
- κάνει αυτόματη επανεκκίνηση διεργασιών που έχουν σταματήσει

Στα launch files μπορούμε να συμπεριλάβουμε αρχεία YAML που περιέχουν configuration values.

Τα launch files έχουν τη μορφή .launch και χρησιμοποιούν μία μορφή τύπου XML. Είναι δυνατόν να τοποθετηθούν οπουδήποτε μέσα σε ένα package directory, αλλά είναι σύνηθες ο μηχανικός να δημιουργεί έναν κατάλογο με όνομα "Launch" μέσα στον directory του workspace για να οργανώσει όλα τα αρχεία εκκίνησης. Τα περιεχόμενα ενός launch file πρέπει να περιέχονται μεταξύ ενός ζεύγους launch tags

<launch> … </launch>

Για την εκκίνηση ενός node, χρησιμοποιούνται τα tags <node>, ενώ απαιτούνται τα arguments pkg, type και name.

<node pkg="..." type="..." name="..." respawn=true ns="..."/>

pkg/type/name: Το argument "pkg" παραπέμπει στο πακέτο που σχετίζεται με τον node που θα ξεκινήσει, ενώ το "type" αναφέρεται στο όνομα του εκτελέσιμου αρχείου του node. Το "name" αναφέρεται στο όνομα του node.

Respawn/Required: Είναι σύνηθες να υπάρχει είτε ένα επιχείρημα respawn είτε ένα required, αλλά όχι και τα δύο. Αν respawn=true, τότε το συγκεκριμένο node θα επανεκκινηθεί εάν για κάποιον λόγο κλείσει. Το required=true θα κάνει το αντίθετο, δηλαδή θα κλείσει όλα τα nodes που σχετίζονται με ένα launch file, εάν αυτό το συγκεκριμένο node κλείσει.

ns: Χρησιμοποιείται συχνά στα launch files, ώστε να τρέξει έναν node εντός ενός namespace Αυτό είναι χρήσιμο όταν χρησιμοποιούνται multiple instances στον ίδιο node.

arg: Μερικές φορές είναι απαραίτητο να χρησιμοποιείται μία τοπική μεταβλητή (local variable) στα launch files. Αυτό μπορεί να γίνει με τη χρήση του

<arg name="…" value="…">
[11]

Ένα παράδειγμα launch file είναι το εξής[11]:

```
<launch>

<group ns="turtlesim1">
        <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
</group>

<group ns="turtlesim2">
        <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
</group>

<node pkg="turtlesim" name="mimic" type="mimic">
        <remap from="input" to="turtlesim1/turtle1"/>
        <remap from="output" to="turtlesim2/turtle1"/>
</node>

</launch>
```

Αυτό το launch file εκκινεί 2 turtlesim_node κάτω από διαφορετικό namespace και ένα mimic node. Στο mimic node φαίνεται το remapping των ονομάτων των topic. Αυτό είναι χρήσιμο εάν τα topic δεν έχουν τα default names που απαιτούνται από ένα node.

2.5.4 Simple Publisher (C++)

Ο κώδικας:

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>

int main(int argc, char **argv)
{

	ros::init(argc, argv, "talker");
	ros::NodeHandle n;
	ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
	ros::Rate loop_rate(10);

	int count = 0;
	while (ros::ok())
	{
	  std_msgs::String msg;

	  std::stringstream ss;
	  ss << "hello world " << count;
	  msg.data = ss.str();

	  ROS_INFO("%s", msg.data.c_str());
	  chatter_pub.publish(msg);

	  ros::spinOnce();

	  loop_rate.sleep();
	  ++count;
}


	return 0;
}
```

Ο κώδικας αναλυτικά:

```
#include "ros/ros.h"
```

Το ros/ros.h είναι ένα include που περιλαμβάνει όλα τα headers που είναι απαραίτητα για τη χρήση των πιο διαδεδομένων στοιχείων του συστήματος ROS.

```
#include "std_msgs/String.h"
```

Αυτό περιλαμβάνει το std_msgs/String message, το οποίο βρίσκεται στο std_msgs package. Πρόκειται για ένα header που παράγεται από το αρχείο String.msg του εν λόγω package.

ros::init(argc, argv, "talker");

Πριν την κλήση οποιαδήποτε άλλης roscpp function σε ένα node, καλείται η:

ros::init() function.

Αυτό επιτρέπει στο ROS να κάνει name remapping μέσω της γραμμής εντολών. Εδώ γίνεται και ο καθορισμός του ονόματος του node.

ros::NodeHandle n;

Ακολουθεί η δημιουργία ενός handle στον node. Το πρώτο NodeHandle που θα δημιουργηθεί θα κάνει στην πραγματικότητα το initialization του node και το τελευταίο που θα καταστραφεί θα κάνει cleanup τυχόν resources που χρησιμοποιούσε ο node.

ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

Το node ενημερώνει τον master ότι πρόκειται να δημοσιευθεί ένα μήνυμα τύπου std_msgs/String στο topic chatter. Αυτό επιτρέπει στον master να δώσει εντολή σε όλα τα node που ακούνε στο chatter ότι πρόκειται να κάνει publish δεδομένα σε αυτό το topic. Το δεύτερο argument είναι το μέγεθος του publishing queue. Σε αυτήν την περίπτωση, εάν δημοσιευθεί πολύ γρήγορα, θα αποθηκεύσει το πολύ 1000 μηνύματα πριν αρχίσει να απομακρύνει τα παλιά.
Η function NodeHandle::advertise() επιστρέφει ένα ros::Publisher object, το οποίο εξυπηρετεί δύο σκοπούς: 1) περιέχει μία μέθοδο publish() που επιτρέπει να κάνει publish messages στο topic με το οποίο δημιουργήθηκε και 2) όταν βγαίνει εκτός παραμέτρων, καταργείται αυτόματα.

.

ros::Rate loop_rate(10);

Ένα ros::Rate object επιτρέπει να καθοριστεί μία συχνότητα με την οποία είναι επιθυμητό να γίνει ένα loop. Αυτό παρακολουθεί πόσος χρόνος έχει περάσει από την τελευταία κλήση της Rate::sleep() και κοιμάται για το κατάλληλο χρονικό διάστημα.

```
int count = 0;
while (ros::ok())
{
```

Η ros::ok() θα επιστρέψει false εάν:

- λάβει SIGINT (Ctrl-C) από το command line
- έαν έλα άλλο node με το ίδιο όνομα έχει διώξει από τον χρήστη από το δίκτυο
- η ros::shutdown() έχει κληθεί από άλλο τμήμα της εφαρμογής

std_msgs::String msg;

std::stringstream ss;
ss << "hello world " << count;
msg.data = ss.str();

Γίνεται εκπομπή ενός μηνύματος στο ROS χρησιμοποιώντας ένα message-adapted class, η οποία γενικά παράγεται από ένα αρχείο msg. Έπειτα, χρησιμοποιείται το standard String message, το οποίο έχει ένα στοιχείο: "data".

ROS_INFO("%s", msg.data.c_str());
η αντικατάσταση της printf/cout στο ROS

chatter_pub.publish(msg);
Σε αυτό το σημείο μεταδίδεται το μήνυμα σε όποιον είναι συνδεδεμένος.

ros::spinOnce();
Η κλήση του για αυτό το απλό πρόγραμμα δεν είναι απαραίτητη, επειδή δε γίνεται η λήψη κανενός callback. Δεδομένου ότι δεν υπάρχει callback, δε θα γίνει καμία εκτέλεση.

loop_rate.sleep();
++count;
}

2.5.5 Simple Subscriber (C++)

Ο κώδικας[13]:

```
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
```

```
{
ros::init(argc, argv, "listener");
ros::NodeHandle n;
ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
ros::spin();

return 0;
}
```

Ο κώδικας αναλυτικά:

```
void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
ROS_INFO("I heard: [%s]", msg->data.c_str());
}
```

Αυτή είναι η callback function που καλείται όταν ένα νέο μήνυμα έχει φτάσει στο chatter topic. Το μήνυμα περνάει σε ένα boost shared_ptr, που σημαίνει ότι μπορεί να αποθηκευτεί.

```
ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
```

Κάνει subscribe στο counter topic με τον master. Το ROS καλεί τη counter Callback() function κάθε φορά που λαμβάνει ένα νέο μήνυμα. Το δεύτερο argument είναι το μέγεθος της ουράς (queue size).

```
ros::spin();
```

Εισέρχεται σε ένα loop, καλώντας message callbacks όσο το δυνατόν γρηγορότερα. Εάν δεν υπάρχει message, δε θα χρησιμοποιήσει πολύ CPU. Θα τερματίσει μόλις η ros::ok() επιστρέψει false, πράγμα που σημαίνει ότι η ros::shutdown() έχει κληθεί.

Εφόσον έχει γραφτεί ο κώδικας, μένει να κληθεί η εντολή catkin_make. Το catkin δεν έχει ιδέα πώς να βρει ή να χτίσει τον κώδικα.

Αρχικά, γίνεται μεταφορά στον φάκελο src του workspace και δημιουργείται ένα package με την εντολή:

catkin_create_pkg pkg_name dependencies

$ catkin_create_pkg pub_sub roscpp
Δημιουργεί ένα pub_sub package με το roscpp ως depedency

Τώρα στο δημιουργημένο αρχείο CmakeLists.txt πρέπει να προστεθούν οι ακόλουθες γραμμές:

```
add_executable(talker talker.cpp)
target_link_libraries(talker ${roscpp_LIBRARIES} ${std_msgs_LIBRARIES})

add_executable(listener listener.cpp)
target_link_libraries(listener ${roscpp_LIBRARIES} ${std_msgs_LIBRARIES})
```

Και με το catkin_make θα γίνει το build. Φτιάχνοντας ένα launch file, θα ξεκινήσουν τα nodes:

$ rosrun pub_sub talker.cpp
$ rosrun pub_sub listener.cpp

2.5.6 Δημιουργία Custom Message

Το πρώτο πράγμα που πρέπει να γίνει είναι η δημιουργία ενός φακέλου msg στο πακέτο. Έπειτα, ακολουθεί η δημιουργία ενός αρχείου τύπου .msg:

NodeExample.msg

που περιέχει:

string message

int32 a

int32 b

Εδώ το costume message θα έχει τρία πεδία.

Αρχικά, στο package.xml θα πρέπει να βεβαιωθεί ότι αυτές οι γραμμές έχουν αφαιρεθεί από το σχόλιο

<build_depend>message_generation**</build_depend>**

<build_export_depend>message_runtime**</build_export_depend>**

<exec_depend>message_runtime**</exec_depend>**

Στη συνέχεια, στο αρχείο **CmakeLists.txt** πρέπει να προστεθεί το **message_generation**

depedency find_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std_msgs

message_generation

)

Επίσης, να εισαχθεί το **message runtime dependency.**

catkin_package(

CATKIN_DEPENDS message_runtime roscpp std_msgs...

)

Επίσης, εισάγεται το **NodeExample.msg** κάνοντας uncomment τα παρακάτω:

add_message_files(

FILES

NodeExample.msg

)

και

generate_messages(

Depedencies

std_msgs

)

Μετά την κατασκευή, μπορεί να γίνει ο έλεγχος για τη σωστή λειτουργία όλων των παραπάνω εκτελώντας:

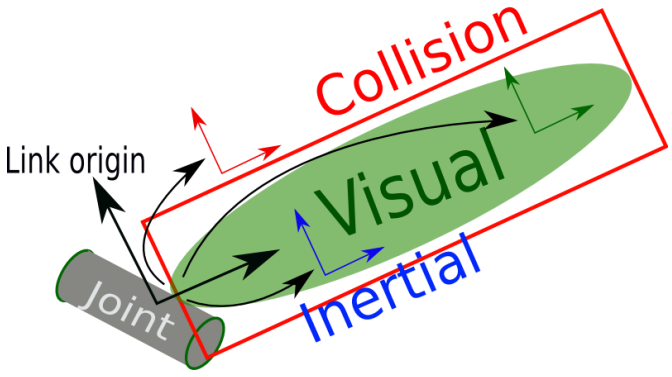
rosmsg show package_name/NodeExample

2.5.7 URDF

Το μοντέλο URDF είναι μία συλλογή αρχείων που περιγράφουν τη φυσική περιγραφή ενός ρομπότ σε ROS. Τα αρχεία URDF είναι απαραίτητα για την κατανόηση του ROS και την προσομοίωση καταστάσεων με το ρομπότ, προτού κάποιος αποκτήσει το πραγματικό ρομπότ.

Για να περιγραφεί ο βραχίονας 6dof, θα χρειαστούν Links(Εικόνα 2.6) και Joints (Εικόνα 2.7), καθένα από τα οποία θα περιλαμβάνει **inertia, visual, collision και geometry.**

Στο αρχείο **configuration.urdf.xacro** ορίζονται οι διαστάσεις που χρησιμοποιούνται εντός του **urdf.**



Εικόνα 2.6 – Link(www.Ros.org)

Τα **links**[20] που θα περιγραφούν έχουν τη μορφή:

```
<link name="my_link">
 <inertial>
  <origin xyz="0 0 0.5" rpy="0 0 0"/>
  <mass value="1"/>
  <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
 </inertial>

 <visual>
  <origin xyz="0 0 0" rpy="0 0 0" />
 <geometry>
  <box size="1 1 1" />
 </geometry>
 <material name="Cyan">
  <color rgba="0 1.0 1.0 1.0"/>
 </material>
</visual>

<collision>
 <origin xyz="0 0 0" rpy="0 0 0"/>
 <geometry>
  <cylinder radius="1" length="0.5"/>
 </geometry>
</collision>
</link>
```

<inertial>: Οι αδρανειακές ιδιότητες του συνδέσμου.

<origin>:

xyz: Αντιπροσωπεύει τους άξονες xzy.

rpy: Αντιπροσωπεύει τις γωνίες roll, pitch και yaw του άξονα.

<mass>: Αντιπροσωπεύει τη μάζα του link.

<inertia>: Ο πίνακας ροπής αδράνειας μάζας καθορίζεται από τα 6 στοιχεία ixx, ixy, ixz, iyy, iyz, izz, επειδή είναι συμμετρικός.

<visual> : Οι οπτικές ιδιότητες του link που καθορίζουν το σχήμα του αντικειμένου (πλαίσιο, κύλινδρος, κουτί κλπ.).

<geometry> : Το σχήμα του οπτικού αντικειμένου.

Αυτό μπορεί να είναι ένα από τα ακόλουθα: **<box>,<cylinder>,<sphere>,<mesh>.**

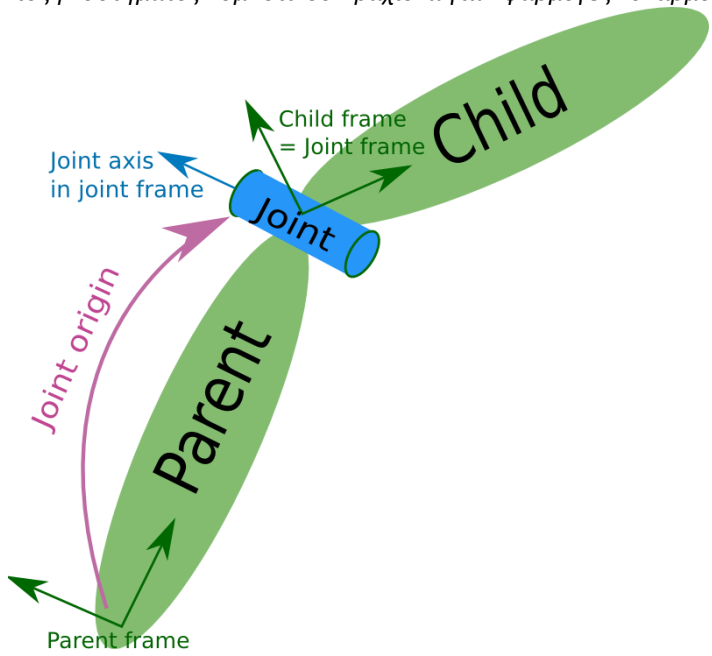
<material> : Καθορίζει το υλικό του αντικειμένου και βρίσκεται η επιφάνειά του ονοματικά (π.χ. name of the material).

<color> (optional)

rgba: Το χρώμα του υλικού που ορίζεται από 4 αριθμούς που αντιπροσωπεύουν τα χρώματα red/green/blue/alpha με [0,1].

<collision> : Οι ιδιότητες της σύγκρουσης για ένα link. Το collision μπορεί να είναι διαφορετικό από τις οπτικές (visual) ιδιότητες ενός link. Συνήθως χρησιμοποιούνται απλούστερα μοντέλα σύγκρουσης για τη μείωση του χρόνου υπολογισμού.

Η δομή των αρθρώσεων (joints) απεικονίζεται στο παρακάτω σχήμα.



Εικόνα 2.7 – Joint(www.Ros.org)

Παρακάτω παρατίθεται η περιγραφή ενός joint[21] όπως χρησιμοποιείται σε ένα URDF αρχείο:

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>

    <calibration rising="0.0"/>
    <dynamics damping="0.0" friction="0.0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
    <safety_controller k_velocity="10" k_position="15" soft_lower_limit="-2.0"
soft_upper_limit="0.5" />
</joint>
```

Type: Καθορίζει τον τύπο της άρθρωσης(joint)

Τύποι άρθρωσης

• **Revolute** - περιστρέφεται κατά μήκος του άξονα και έχει περιορισμένο εύρος.

• **Continuous** - μία συνεχής άρθρωση που περιστρέφεται γύρω από τον άξονα χωρίς περιορισμούς.

• **Prismatic** - συρόμενη άρθρωση που ολισθαίνει κατά μήκος του άξονα και έχει περιορισμένο εύρος.

• **Fixed** - αυτό δεν είναι πραγματική άρθρωση επειδή δεν μπορεί να κινηθεί. Όλοι οι βαθμοί ελευθερίας είναι κλειδωμένοι.

• **Floating** - αυτή η άρθρωση επιτρέπει κίνηση και για τους 6 βαθμούς ελευθερίας.

• **Planar** - αυτή η άρθρωση επιτρέπει κίνηση σε επίπεδο κάθετο στον άξονα.

<parent> : Το όνομα του συνδέσμου που είναι parent προηγείται αυτού του joint στη δομή του ρομπότ.

<child> : Το όνομα του συνδέσμου που είναι child έπεται αυτού του joint στη δομή του ρομπότ.

<axis> : Αφορά τα περιστρεφόμενα joints (revolute, prismatic, planar) και περιλαμβάνει τα xzy και rpy.

xyz: Αντιπροσωπεύει τους άξονες xzy.

rpy: Αντιπροσωπεύει τις γωνίες roll, pitch και yaw του άξονα.

<limit> (απαιτείται μόνο για revolute και prismatic joints)

Το limit θα πρέπει να συμβαδίζει με τα όρια των **servo** που θα χρησιμοποιηθούν για την κατασκευή του βραχίονα. Τα **Lower** και τα **upper limits** υπολογίζονται σε μοίρες για **revolute** και μέτρα για **prismatic**.

Effort : Μέγιστη δύναμη που μπορεί να ασκήσει το joint.

velocity : Μέγιστη ταχύτητα που μπορεί να έχει το joint.

Ένα παράδειγμα στα όρια που δόθηκαν στον καρπό.

<limit lower="${wrist_2_lower_limit}" upper="${wrist_2_upper_limit}" effort="330.0" velocity="2.16"/>

Ένα πολύ σημαντικό κομμάτι για την προσομοίωση είναι το <transmission>, το οποίο είναι επέκταση του URDF και χρησιμοποιείται για να περιγράψει τη σχέση μεταξύ actuator και joint. Επιτρέπει να μοντελοποιηθούν οι σχέσεις μετάδοσης (gear ratios) και οι παράλληλες συνδέσεις (parallel linkages). Αποτελείται από:

<type>: Προσδιορίζει τον τύπο του transmission.

<joint>: Προσδιορίζει ένα joint ονομαστικά που συνδέεται με το συγκεκριμένο transmission.

<hardwareInterface>: Το Hardware Interface χρησιμοποιείται από το ROS σε συνδυασμό με ROS controller για την αποστολή και λήψη εντολών στο υλικό.

<actuator>: Ένας actuator με τον οποίο συνδέεται το transmission.Προσδιορίζεται από το όνομά του.

<mechanicalReduction> : Καθορίζει μία μηχανική μείωση στη μετάδοση joint/actuator.

Ένα κομμάτι από το αρχείο generic_arm.transmissions.xacro

```
<transmission name="${prefix}shoulder_pan_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="${prefix}shoulder_pan_joint">
      <hardwareInterface>${hw_interface}</hardwareInterface>
    </joint>
    <actuator name="${prefix}shoulder_pan_motor">
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
```

2.5.8 XACRO

Το XACRO είναι μία μακροεντολή Xml (macro) που επιτρέπει τη χρήση μακροεντολών σε ένα αρχείο urdf. Δίνει τη δυνατότητα χρήσης μεταβλητών, μαθηματικών και macro. Επιτρέπει να χωρίσουμε το μοντέλο του ρομπότ (URDF) σε διαφορετικά αρχεία[14].

Property

Αντί να οριστούν σταθερές τιμές, μπορούν να χρησιμοποιηθούν μεταβλητές (variables), οι οποίες ονομάζονται property στο xacro.

Constants

```
<xacro:property name="width" value="0.2" />
<xacro:property name="bodylen" value="0.6" />

<link name="base_link">
  <visual>
    <geometry>
      <cylinder radius="${width}" length="${bodylen}"/>
    </geometry>
  </visual>
</link>
```

Στο τελευταίο απόσπασμα γίνεται η δημιουργία δύο properties με default values:

```
<xacro:property name="width" value="0.2" />
<xacro:property name="bodylen" value="0.6" />
```

Math

```
<cylinder radius="${wheeldiam/2}" length="0.1"/>
<origin xyz="${reflect*(width+.02)} 0 0.25" />
```

Simple Macro

Marco είναι κομμάτια κώδικα που μπορεί να χρησιμοποιηθεί ως function.

```
<xacro:macro name="default_origin">
  <origin xyz="0 0 0" rpy="0 0 0"/>
</xacro:macro>
```

Parameterized Macro

```
<xacro:macro name="default_inertial" params="mass">
```

```
    <inertial>
        <mass value="${mass}" />
        <inertia ixx="1.0" ixy="0.0" ixz="0.0"
            iyy="1.0" iyz="0.0"
            izz="1.0" />
    </inertial>
</xacro:macro><
```

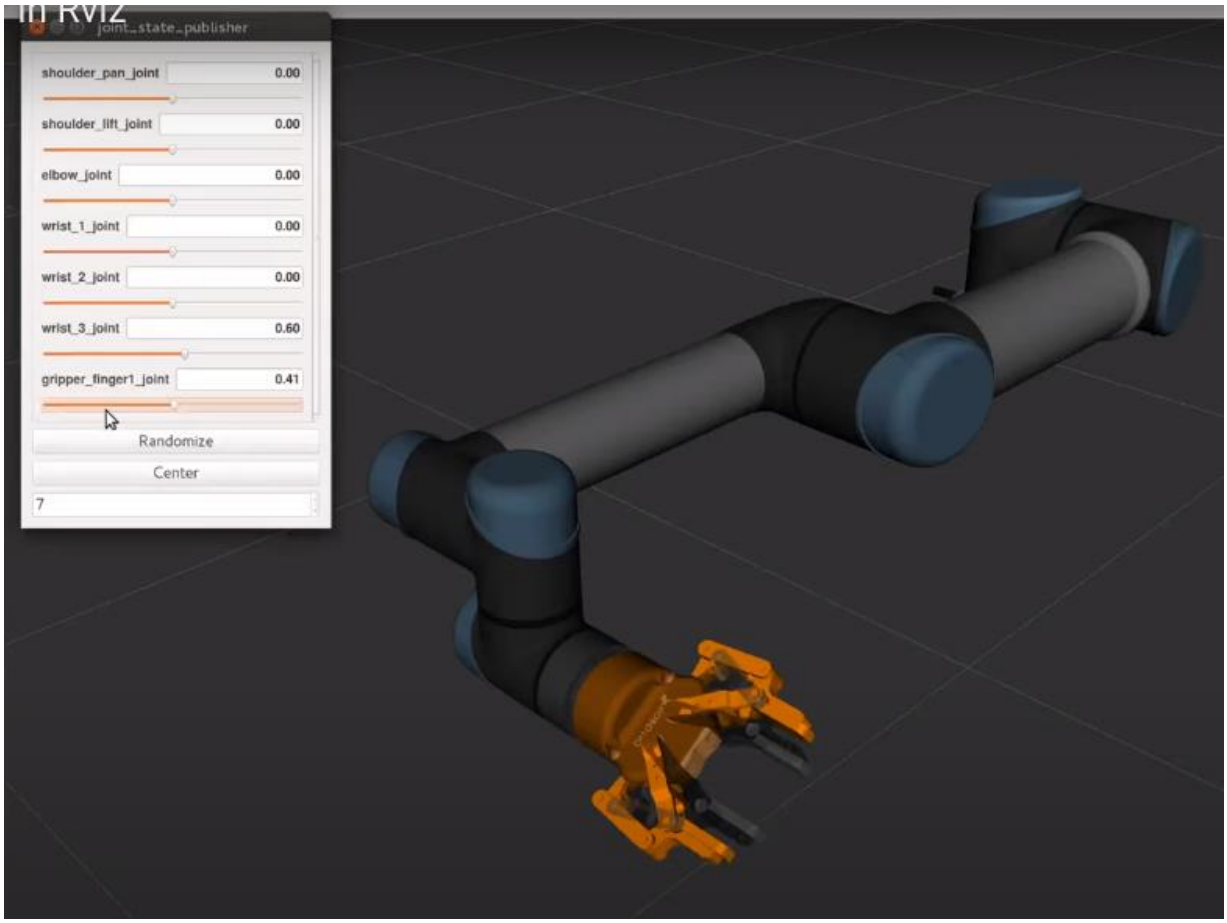
Αυτό μπορεί να χρησιμοποιηθεί με τον κώδικα:

```
<xacro:default_inertial mass="10"/>
```

Including other xacro files

```
<xacro:include filename="$(find package_name)/urdf/wheel.xacro" />
```

2.5.9 Joint state publisher



Εικόνα 2.8 - Joint state publisher

Αυτό το πακέτο διαβάζει την παράμετρο robot_description, βρίσκει όλες τις μη σταθερές αρθρώσεις και δημοσιεύει ένα JointState μήνυμα με όλες αυτές τις αρθρώσεις. Το Joint State Publisher package παρακολουθεί τη θέση (δηλαδή τη γωνία σε ακτίνια για έναν σερβοκινητήρα ή τη μετατόπιση σε μέτρα για έναν linear actuator) και την ταχύτητα κάθε άρθρωσης ενός ρομπότ και δημοσιεύει αυτές τις τιμές στο ROS ως μηνύματα sensor_msgs/JointState.

Υπάρχουν τέσσερις πιθανές πηγές για την τιμή που παίρνει κάθε JointState:

1.Τιμές που εισάγονται απευθείας μέσω του GUI (Graphical User Interface)

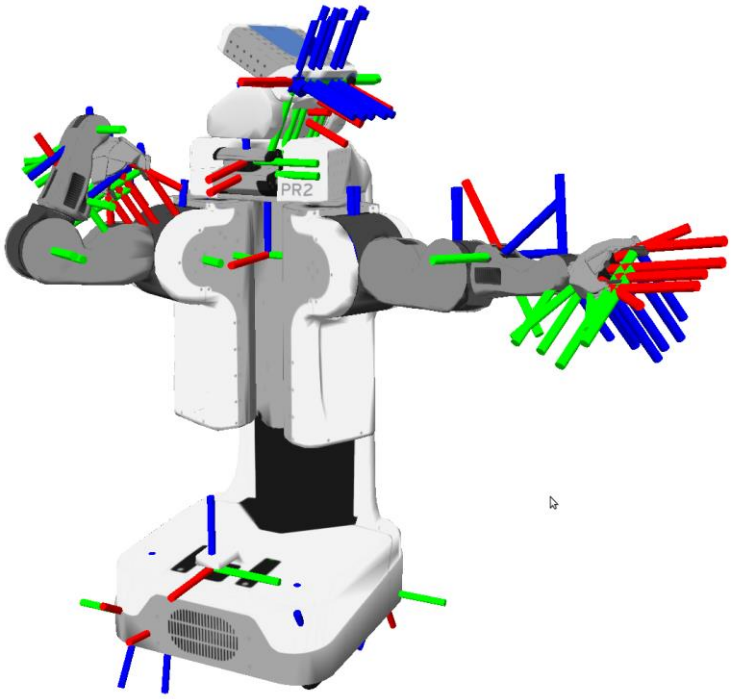
2. Μηνύματα JointState στα οποία κάνει subscribe το node

3. Η τιμή μίας άλλης άρθρωσης

4. Η προεπιλεγμένη τιμή

Έτσι, στην ουσία αυτό το node διαβάζει το urdf, βρίσκει όλες τις κινούμενες αρθρώσεις και δημοσιεύει το joint state κάθε μίας από αυτές τις αρθρώσεις.

2.5.10 Robot state publisher



Εικόνα 2.9 - Robot state publisher(www.Ros.org)

Ο Robot State Publisher λαμβάνει στη συνέχεια δύο input:

1.Τα μηνύματα sensor_msgs/JointState από τον Joint State Publisher.

2.Ένα μοντέλο του ρομπότ σε μορφή αρχείου URDF.

Ο Robot State Publisher λαμβάνει αυτές τις πληροφορίες, εξάγει τη θέση, τον προσανατολισμό και τις συντεταγμένες του ρομπότ και δημοσιεύει αυτά τα δεδομένα στο tf2 package.

Το πακέτο tf2 είναι υπεύθυνο για την παρακολούθηση της θέσης και του προσανατολισμού όλων των συντεταγμένων ενός ρομπότ με την πάροδο του χρόνου. Ανά πάσα στιγμή, μπορεί να ζητηθεί από το πακέτο tf2 να γίνει γνωστή η θέση και ο προσανατολισμός οποιουδήποτε πλαισίου συντεταγμένων (δηλ. "child frame") σε σχέση με ένα άλλο πλαίσιο συντεταγμένων (δηλ. "parent frame").

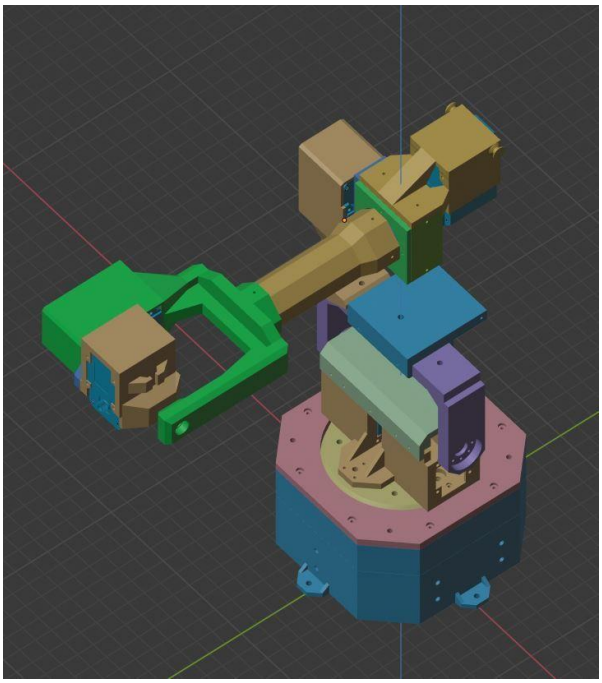
3 Κατασκευή και χρήση ρομποτικού βραχίονα 6 βαθμών ελευθερίας

3.1 Εισαγωγή κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία της κατασκευής του ρομποτικού βραχίονα 6 βαθμών ελευθερίας. Θα γίνει περιγραφή τόσο των εξαρτημάτων, του κώδικα και του εξοπλισμού που χρησιμοποιήθηκε, όσο και του κόστους τους.

Περιγράφεται η κατασκευή ενός ρομποτικού βραχίονα 6 βαθμών ελευθερίας και ενός gripper μέσω 3D εκτυπωτή. Ακόμα, γίνεται περιγραφή του είδους των έξυπνων σερβοκινητήρων, του τρόπου τροφοδοσίας και του σένσορας kinect που χρησιμοποιήθηκε για την επίτευξη του στόχου.

3.2 Βραχίονας – 3D μοντέλο



Εικόνα 3.1 - 3D μοντέλο 6 DOF manipulator(www.cults3d.com)

Το μοντέλο του βραχίονα που επιλεχθηκε και τυπώθηκε για την παρούσα εργασία είναι ο 6DOF ROBOT ARM V2 :

https://cults3d.com/en/3d-model/various/6dof-robot-arm-v2

και η έκδοση του gripper είναι ο IMPROVED GRIPPER:

https://cults3d.com/en/3d-model/various/improved-gripper

Τα STL διαθέτονται στο https://cults3d.com/ από τον χρήστη και δημιουργό

https://cults3d.com/en/users/robolab19

Για την εκτύπωση του βραχίονα χρειάστηκαν:

- Νήμα PLA 1.75mm Devil (22 ευρώ)

Η εκτύπωση πραγματοποιήθηκε στο εργαστήριο ΠΟΙΩ του Δήμου Αθηναίων με τη χρήση εκτυπωτή Creality3D Ender-3.

3.3 Βραχίονας – Κατασκευή

Για τη συναρμολόγησή του χρειάστηκαν:

- Διάφορες βίδες και παξιμάδια 3 mm(10 ευρώ)

- Ρουλεμάν 693ZZ (3x8x4mm) και 6705ZZ (25x32x4mm, 2 τεμάχια)(10 ευρώ)

- Smart servos LX16A (9 τεμάχια)(160 ευρώ)

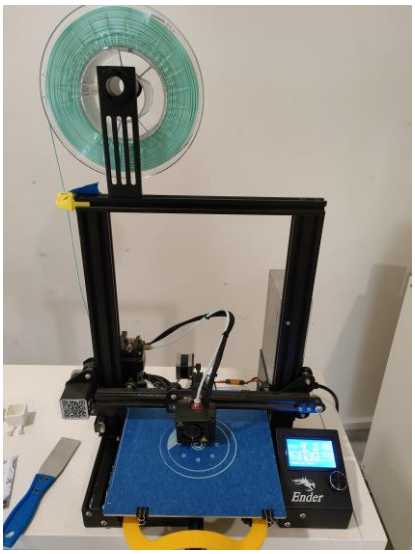
- Μπάλες airsoft BB (πλαστικές) 6mm για το ρουλεμάν της βάσης του βραχίονα(5 ευρώ)

- Hiwonder TTL USB Debugging Board(13 ευρώ)

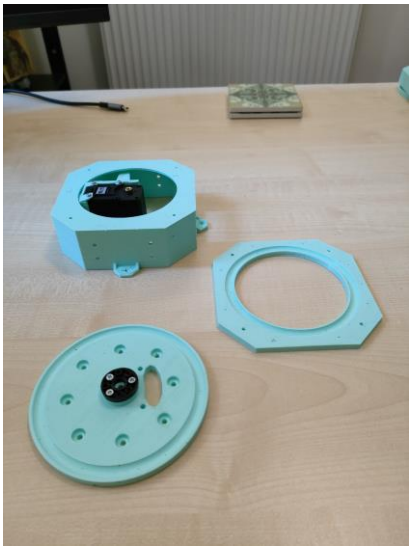
Συνολικό κόστος εκτύπωσης και συναρμολόγησης βραχίονα:220 ευρώ

3.3.1 Εκτύπωση βραχίονα

Σε αυτό το υποκεφάλαιο παρουσιάζεται η διαδικασία εκτύπωσης(Εικόνα 3.2) του βραχίονα με την χρήση του εκτυπωτή Creality3D Ender-3.

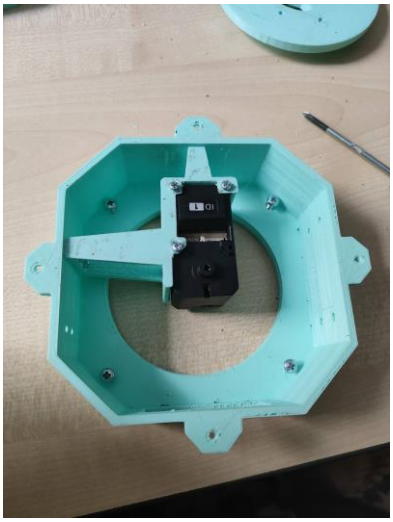


Εικόνα 3.2 – Εκτύπωση βάσης βραχίονας



Εικόνα 3.3 – Κομμάτια βάσης βραχίονα

Στην Εικόνα 3.4 φαίνεται η δομή της βάσης του βραχίονα και ο σερβοκινητήρας(base joint) που ευθύνεται για την περιστροφή της βάσης. Οι μπάλες(Εικόνα 3.5) airsoft BB χρησιμοποιήθηκαν ώστε η περιστροφή να γίνεται ομαλά και ουσιαστικά λειτουργούν ως ρουλεμάν.



Εικόνα 3.4 – Σερβοκινητήρας βάσης βραχίονας

Εικόνα 3.5 –Ρουλεμάν βάσης βραχίονας

Στις παρακάτω εικόνες(Εικόνα 3.6, Εικόνα 3.7) παρουσιάζονται κάποια στιγμιότυπα της συναρμολόγησης και η τελική μορφή του manipulator αφού συνδέθηκε ο end-effector.

Εικόνα 3.6 – Σερβοκινητήρες 1

Εικόνα 3.7 – Σερβοκινητήρες 2

Εικόνα 3.8 – Τελικό στάδιο κατασκευής βραχίονα

3.3.2 Smart servos LX16A

Εικόνα 3.9 – Hiwonder LX-16A(www.hiwonder.hk)

Ο Hiwonder LX-16A(Εικόνα 3.9) είναι ένας serial bus servo με πλήρες μεταλλικό γρανάζι και υψηλή ροπή 17kg, ο οποίος μπορεί να παρέχει temperature και position feedback. Το LX-16A Bus Servo χρησιμοποιεί private serial protocol. Έτσι, είναι απαραίτητο να χρησιμοποιηθεί η πλακέτα TTL/USB debug board (συμβατή για Arduino) για τον έλεγχο του bus servo.

Name:	LX-16A serial intelligent bus servo	**Control method:**	UART serial command
Brand:	Hiwonder	**Communication baud rate:**	115200
Weight:	54g	**Storage:**	save data when the power off
Size:	45.22 x 24.72 x 36.3mm	**Servo ID:**	0~253 for user setting, ID 1 is default
Working voltage:	6-8.4V	**Angel read back function:**	support
Speed:	0.19sec/60° 7.4V	**Protection:**	avoid stalling and overheat
Torque:	17kg.cm 6V; 19.5kg.cm 7.4V	**Data feedback:**	temperature, voltage, position
Rotation:	0-240°	**Indicator:**	LED
No-load current:	100mA	**Working mode:**	servo mode and deceleration motor mode
Stalled current:	2.4~3A	**Gear:**	metal
Servo accuracy:	0.3°	**Wire:**	20cm
Control angle range:	0-1000, correspond to 0-240°	**Connector:**	5264-3P

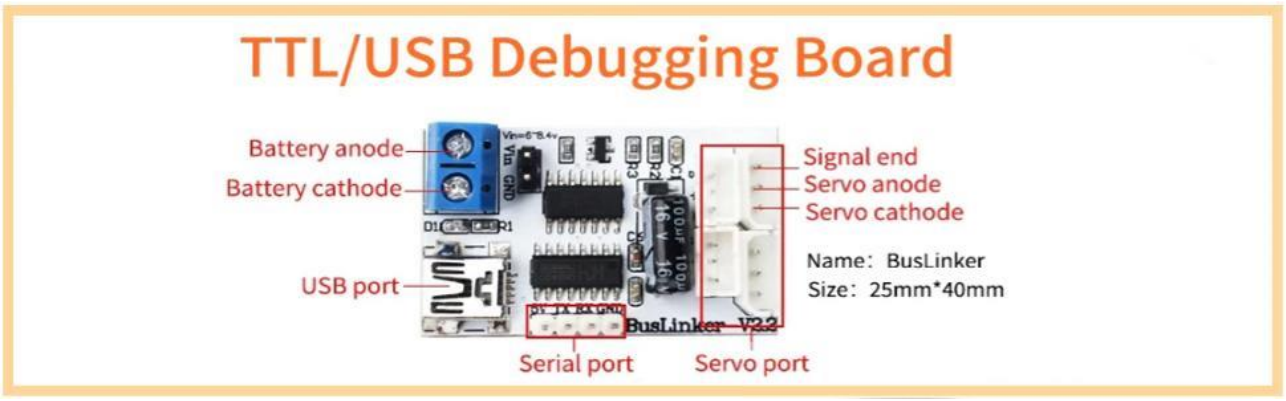
3.3.3 Hiwonder TTL USB Debugging Board

Εικόνα 3.10 - Hiwonder TTL USB Debugging Board(www.hiwonder.hk)

Ο Hiwonder TTL / USB Debugging Board(Εικόνα 3.10) είναι ένας servo debugger, ο οποίος είναι σε θέση να συνδέσει τους smart servo με το λογισμικό του υπολογιστή ή άλλες συσκευές και να τις χρησιμοποιήσει για να ρυθμίσει τις παραμέτρους των servo.

Τα χαρακτηριστικά του είναι τα εξής:

- Δυνατότητα σύνδεσης λογισμικού PC και προσαρμογής των παραμέτρων του servo
- Ανάγνωση της γωνία του servo
- Δυνατότητα σύνδεσης τα TXD και RDX του single-chip για τον έλεγχο του servo



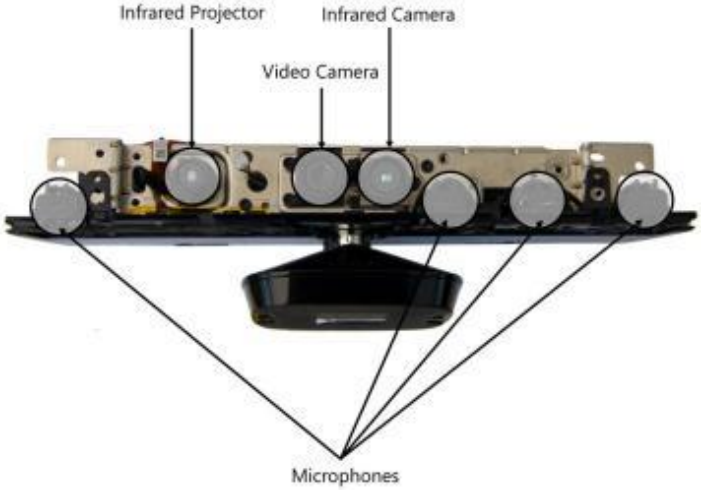
Εικόνα 3.11- Hiwonder TTL USB Debugging Board αναλυτικά(www.hiwonder.hk)

3.4 Kinect

Εικόνα 3.12 – Kinect(www.microsoftpressstore.com)

Μέχρι πρόσφατα οι υπολογιστές είχαν μια πολύ περιορισμένη εικόνα του κόσμου γύρω τους και οι χρήστες είχαν πολύ περιορισμένους τρόπους επικοινωνίας με τους υπολογιστές. Με την πάροδο των ετών, οι υπολογιστές απέκτησαν κάμερες και εισόδους ήχου, αλλά αυτές χρησιμοποιήθηκαν κυρίως για μη αναγνωρίσιμες. Οι υπολογιστές μπορούν να αποθηκεύσουν και να αναπαράγουν το περιεχόμενο των εισόδων, αλλά ήταν πολύ δύσκολο να κατανοήσουν τα δεδομένα των εισόδων σε αυτές τις μορφές.

Για παράδειγμα, όταν οι άνθρωποι ακούνε έναν ήχο, μπορούν να κρίνουν την απόσταση και την κατεύθυνση της ηχητικής πηγής σε σχέση με τη δική τους θέση. Μέχρι πρόσφατα, οι υπολογιστές δυσκολεύονταν περισσότερο να κάνουν τέτοιες εκτιμήσεις. Οι πληροφορίες ήχου από έναν αριθμό μικροφώνων παρέχουν σημαντικές πληροφορίες σχετικά με την απόσταση και την κατεύθυνση της πηγής ήχου, αλλά ο προσδιορισμός αυτών των πληροφοριών είναι δύσκολος για τα προγράμματα. Ομοίως, μια εικόνα βίντεο παρέχει μια εικόνα του περιβάλλοντος για να την αναλύσει ο υπολογιστής, αλλά ο υπολογιστής πρέπει να εργαστεί πολύ σκληρά για να εξάγει πληροφορίες σχετικά με τα αντικείμενα στις εικόνες ή το βίντεο, επειδή μια εικόνα δείχνει μια επίπεδη, δισδιάστατη αναπαράσταση ενός τρισδιάστατου κόσμου. Το Kinect(Εικόνα 3.12) τα άλλαξε όλα αυτά. Ο αισθητήρας Kinect περιέχει δύο κάμερες, μια ειδική πηγή υπέρυθρου φωτός και τέσσερα μικρόφωνα. Διαθέτει επίσης επεξεργασία σήματος που είναι σε θέση να δώσει νόημα σε όλα τα δεδομένα που μπορούν να παράγουν οι κάμερες, το υπέρυθρο φως και τα μικρόφωνα. Συνδυάζοντας την έξοδο από αυτούς τους αισθητήρες, ένα πρόγραμμα μπορεί να παρακολουθεί και να αναγνωρίζει αντικείμενα μπροστά του, να καθορίζει την κατεύθυνση των ηχητικών σημάτων και να τα απομονώνει από το θόρυβο του περιβάλλοντος.



Εικόνα 3.13 – Kinect χωρίς κάλυμμα(www.microsoftpressstore.com)

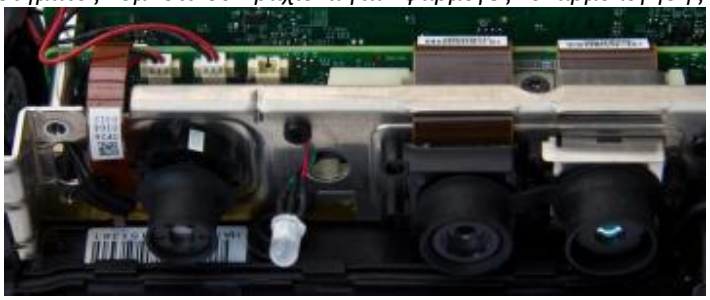
Στην Εικόνα 3.13 απεικονίζεται ένα Kinect με αφαιρεμένο το κάλυμμα. Φαίνονται οι δύο κάμερες στη μέση και η ειδική πηγή φωτός στα αριστερά. Τα τέσσερα μικρόφωνα είναι τοποθετημένα κατά μήκος του κάτω μέρους της μπάρας αισθητήρων. Μαζί, αυτές οι συσκευές παρέχουν την "εικόνα" που έχει το Kinect για τον κόσμο.

Εικόνα 3.14 – Kinect χωρίς κάλυμμα, πίσω πλευρά(www.microsoftpressstore.com)

Για να χωρέσουν όλα στο λεπτό σχήμα της ράβδου(Εικόνα 3.14), οι σχεδιαστές έπρεπε να στοιβάξουν τις πλακέτες κυκλωμάτων η μία πάνω στην άλλη. Ορισμένα από αυτά τα εξαρτήματα παράγουν αρκετή θερμότητα, οπότε ένας μικροσκοπικός ανεμιστήρας που φαίνεται στην άκρη δεξιά της εικόνας αναρροφά αέρα κατά μήκος των κυκλωμάτων για να τα διατηρεί δροσερά. Η βάση περιέχει ένα ηλεκτρικό μοτέρ και μια διάταξη γραναζιών που επιτρέπει στο Kinect να ρυθμίζει κάθετα τη γωνία θέασης.

3.4.1 Ο αισθητήρας βάθους

Το Kinect έχει τη μοναδική ικανότητα να "βλέπει" τρισδιάστατα. Σε αντίθεση με τα περισσότερα άλλα συστήματα υπολογιστικής όρασης, το σύστημα Kinect είναι σε θέση να δημιουργήσει ένα "depth map" της περιοχής μπροστά του. Αυτός ο χάρτης παράγεται εξ ολοκλήρου μέσα στη μπάρα του αισθητήρα(Εικόνα 3.15) και στη συνέχεια μεταδίδεται μέσω του καλωδίου USB στον κεντρικό υπολογιστή με τον ίδιο τρόπο που θα μεταφερόταν μια τυπική εικόνα κάμερας, με τη διαφορά ότι αντί για πληροφορίες χρώματος για κάθε εικονοστοιχείο σε μια εικόνα, ο αισθητήρας μεταδίδει τιμές απόστασης. Ο αισθητήρας χρησιμοποιεί μια έξυπνη τεχνική που αποτελείται από έναν προβολέα υπερύθρων και μια κάμερα που μπορεί να δει τις μικροσκοπικές κουκκίδες που παράγει ο προβολέας.



Εικόνα 3.15 – Αισθητήρας βάθους(www.microsoftpressstore.com)

Ο προβολέας είναι το αριστερό στοιχείο στην εικόνα. Μοιάζει κάπως με κάμερα, αλλά στην πραγματικότητα είναι ένας μικροσκοπικός προβολέας υπερύθρων. Η κάμερα υπερύθρων βρίσκεται στη δεξιά πλευρά της εικόνας. Μεταξύ του προβολέα και της κάμερας υπάρχει μια λυχνία LED που εμφανίζει την κατάσταση της συσκευής Kinect και μια κάμερα που καταγράφει μια τυπική δισδιάστατη προβολή της σκηνής.

3.5 Κώδικας

Στο παρόν κεφάλαιο παρουσιάζεται η ο κώδικας που απαιτείται για την σύνδεση και την λειτουργία του ρομποτικού βραχίονα με το ROS.

3.5.1 URDF του ρομπότ

```
<?xml version='1.0'?>

<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <!-- Include Inertia Macros -->
  <xacro:include filename="$(find arm_description)/urdf/macros/inertia_macros.urdf.xacro" />

  <!-- Include Transmissions -->
  <xacro:include filename="$(find
arm_description)/urdf/arms/generic_arm/generic_arm.transmissions.xacro" />

  <!-- Instantiate Transmissions -->
  <xacro:generic_arm_transmission prefix="${prefix}"
hw_interface="${transmission_hw_interface}" />

  <!-- If we use simulation include Gazebo Stuff -->
  <xacro:if value="$(arg sim)">

    <!-- Gazebo link properties -->
    <xacro:include filename="$(find
arm_description)/urdf/arms/generic_arm/generic_arm.gazebo.xacro" />
    <!-- Instantiate it -->
    <xacro:generic_arm_gazebo prefix="${prefix}" />
```

```
  </xacro:if>
```

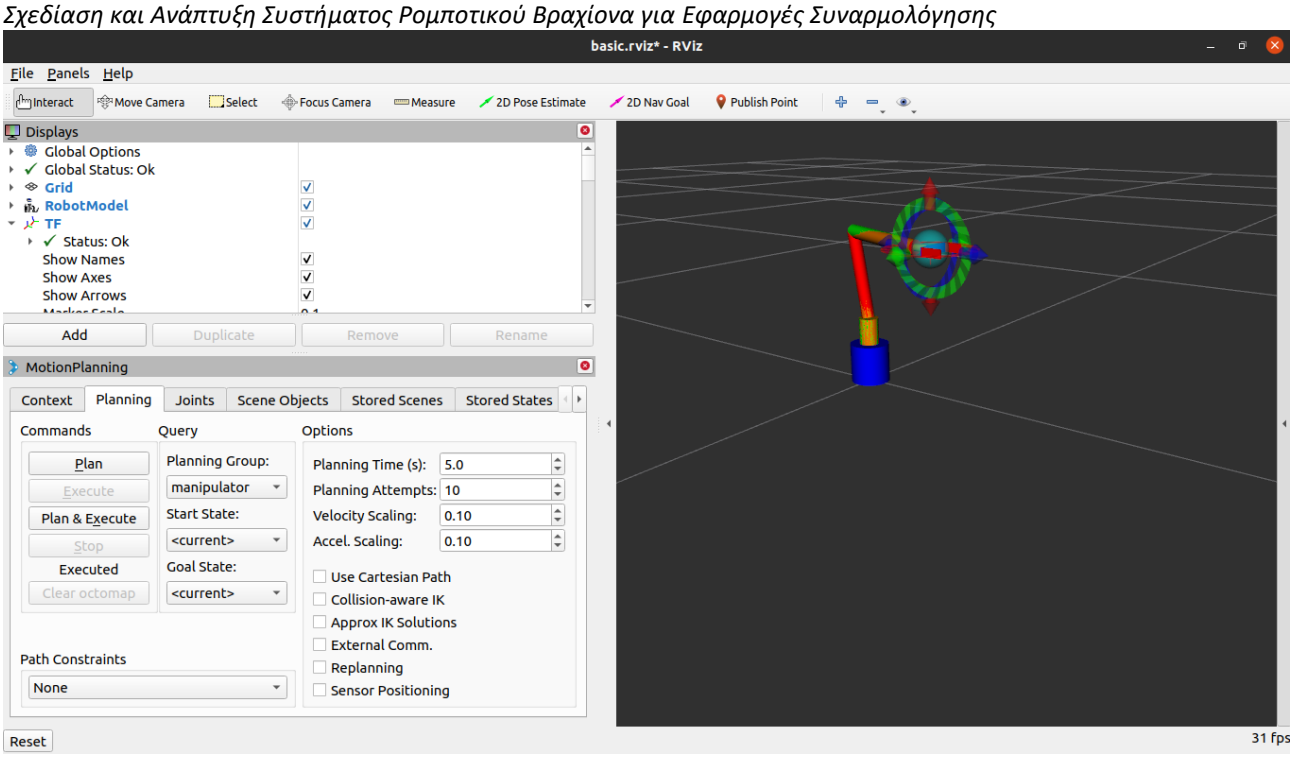
3.5.2 Εμφάνιση του ρομπότ στην προσομοίωση

Τρέχοντας το ακόλουθο κύριο αρχείο εκκίνησης εμφανίζεται ο βραχίονας στο Rviz(Εικόνα 3.16) και δίνεται η δυνατότητα αποστολής στόχων, ο σχεδιασμός της κίνησης του βραχίονα και η επίτευξη της εκτέλεσης των στόχων.

main_real.launch

```
<?xml version="1.0" encoding="UTF-8"?>

<launch>

  <!-- sim bringup-->
  <include file="$(find driver_bringup)/launch/real_bringup.launch">

    <arg name="robot" value="generic_arm"/>
    <arg name="sim" value="false"/>

  </include>

  <!-- start move group-->
  <include file="$(find arm_moveit_config)/launch/move_group.launch">
    <arg name="publish_monitored_planning_scene" value="true" />
    <arg name="allow_trajectory_execution" value="true"/>
    <arg name="info" value="true"/>
    <arg name="load_robot_description" value="false" />
  </include>

  <!-- MoveToPose Action Server -MoveIt -->
  <include file="$(find moveit_servers)/launch/include/move_topose_arm_action_server.launch"/>
  <!-- Joint Arm Action Server - MoveIt -->
  <include file="$(find moveit_servers)/launch/include/move_joints_arm_action_server.launch"/>
  <!--MoveIt collision object interface-->
  <include file="$(find
moveit_collision_object_interface)/launch/moveit_collision_object_interface.launch"/>
```

```
  <!--MoveIt Servo-->
  <!-- <include file="$(find moveit_interface)/launch/moveit_servo/moveit_servo.launch"/> -->
  <!--Pose Tracking-->
  <!-- <include file="$(find pose_tracking)/launch/pose_tracking.launch"/> -->
  <!--Interactive marker-->
  <!-- <node name="pose_tracking_marker_control" pkg="robot_markers"
type="interactive_marker" /> -->
  <!-- set start position if you want-->
  <!-- <node name="Set_Start_Arm_Pos" pkg="moveit_interface" type="set_start_pos.py"
respawn="false" output="screen"/> -->

  <!-- rviz -->
  <arg name="rviz_config" default="basic"/>
  <include file="$(find arm_viz)/launch/rviz.launch">
    <arg name="config" value="$(arg rviz_config)" />
  </include>

  <!-- <include file="$(find freenect_launch)/launch/freenect.launch"/> -->


   <!-- <param name="kinect_description"
         command="$(find xacro)/xacro '$(find
arm_description)/urdf/robots/generic_arm/kinect_robot.urdf.xacro'"
  />
  <node name="kinect_state_publisher"
        pkg="robot_state_publisher"
        type="robot_state_publisher"
  >
      <param name="robot_description" value="kinect_description"/>
  </node> -->
  <!-- <node name="kinect_tf" pkg="driver_bringup" type="kinect_tf.py" /> -->

  <!-- <node name="joystick_vel_control" pkg="driver_bringup" type="joystick_vel_control.py" />
-->

</launch>
```



Εικόνα 3.16 –Ο βραχίονας στο Rviz

3.5.3 Εκτέλεση του driver του φυσικού ρομπότ

real_bringup.launch

```
<?xml version="1.0" encoding="UTF-8"?>

<launch>

  <!-- sim bringup-->
  <include file="$(find driver_bringup)/launch/real_bringup.launch">

    <arg name="robot" value="generic_arm"/>
    <arg name="sim" value="false"/>

  </include>

  <!-- start move group-->
  <include file="$(find arm_moveit_config)/launch/move_group.launch">
    <arg name="publish_monitored_planning_scene" value="true" />
    <arg name="allow_trajectory_execution" value="true"/>
    <arg name="info" value="true"/>
    <arg name="load_robot_description" value="false" />
  </include>
```

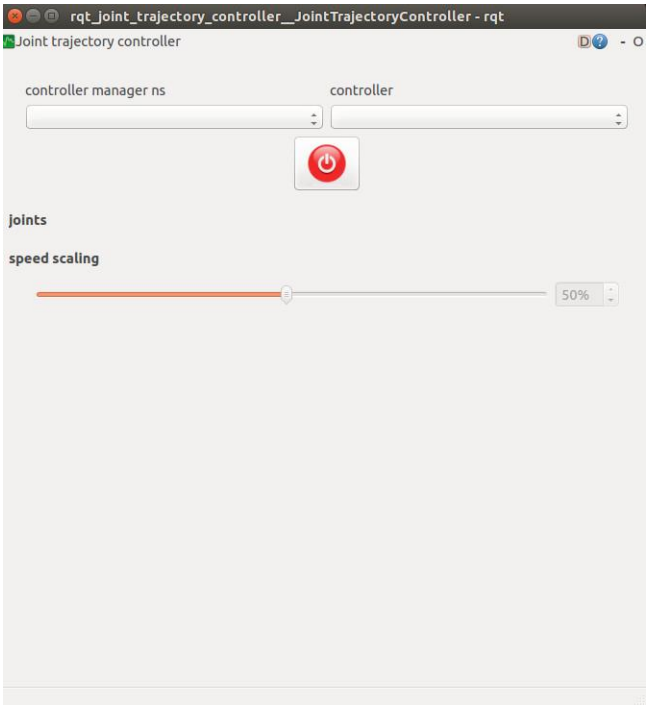
```
  <!-- MoveToPose Action Server -MoveIt -->
  <include file="$(find moveit_servers)/launch/include/move_topose_arm_action_server.launch"/>
  <!-- Joint Arm Action Server - MoveIt -->
  <include file="$(find moveit_servers)/launch/include/move_joints_arm_action_server.launch"/>
  <!--MoveIt collision object interface-->
  <include file="$(find
moveit_collision_object_interface)/launch/moveit_collision_object_interface.launch"/>

 <!-- rviz -->
  <arg name="rviz_config" default="basic"/>
  <include file="$(find arm_viz)/launch/rviz.launch">
    <arg name="config" value="$(arg rviz_config)" />
  </include>

</launch>
```

3.5.4 Κίνηση αρθρώσεων με το rqt GUI

Το εργαλείο joint trajectory controller προσφέρει την δυνατότητα ελέγχου των joint που έχουν οριστεί για το μοντέλο του βραχίονα. Καλείται με την εντολή :

rosrun rqt_joint_trajectory_controller rqt_joint_trajectory_controller

Το περιβάλλον του rqt GUI



Εικόνα 3.17 – Ο joint trajectory controller

Αρχικά στην πτυσσόμενη λίστα στα αριστερά επιλέγεται ο controller_manager και στη συνέχεια οι επιλέγονται οι ελεγκτές των διαφόρων ομάδων αρθρώσεων που παρατίθενται στην πτυσσόμενη λίστα στα δεξιά.

Εικόνα 3.18 – Ο controller_manager

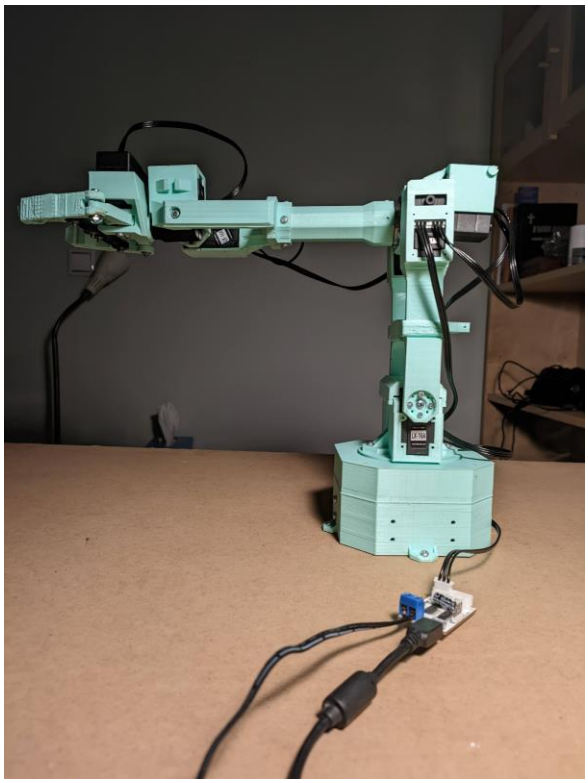
Το συγκεκριμένο εργαλείο σε συνδυασμό με την εντολή rostopic echo arm_command δίνει την δυνατότητα της εύρεσης των επιθυμητών γωνιών των smartservo ώστε να αντιστοιχούν στις γωνίες του βραχίονα. Αυτό επιτυγχάνεται θέτοντας τα κατάλληλα OFFSET στο αρχείο **real_controllers.yaml** το οποίο περιλαμβάνει τις ρυθμίσεις των servo όπως φαίνεται παρακάτω.

```
# Settings for generic robot driver

generic_robot_driver:

  get_start_state: false

  threshold: 0.01 #0.05 #0.022 0.00418 per step

  has_feedback: false

  # servos

  servo_pins: [1, 2, 3, 4, 5, 6, 7, 8, 9]

  joint_lower: [-2.094, -1.5707, -1.5707, -2.094, -2.094, -2.094, 0.0]

  joint_upper: [2.094, 1.5707, 1.17286, 2.094, 2.094, 2.094, 0.03]
```

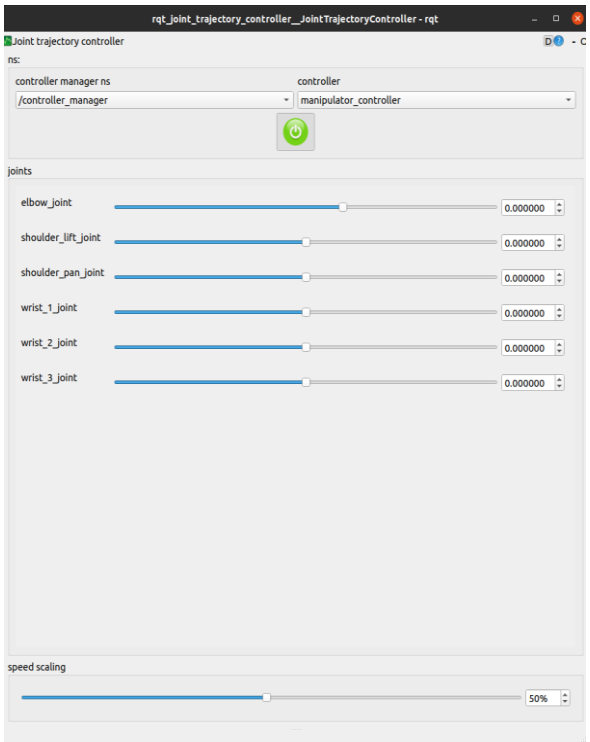
```
  servo_lower: [0, 0, 0, 0, 0, 0, 300]
  servo_upper: [1000, 1000, 1000, 1000, 1000, 1000, 600]
  directions_reverse: [false, false, true, false, false, false, true]
  offsets: [0, -30, 90, 0, 0, 0, 0]

servo_driver:
  serial_port: "/dev/ttyUSB0"
  servo_ids: [1,2,3,4,5,6,7,8,9]
  # offsets: [0,0,0,0,0,0,0,0]
  min_positions: [0,0,0,0,0,0,0,0,0]
  max_positions: [1000,1000,1000,1000,1000,1000,1000,1000,1000]
  min_voltages: [4500,4500,4500,4500,4500,4500,4500,4500,4500]
  max_voltages: [12000,12000,12000,12000,12000,12000,12000,12000,12000]
  min_temperatures: [50,50,50,50,50,50,50,50,50]
  max_temperatures: [100,100,100,100,100,100,100,100,100]
  state_publish_frequency: 20
```

3.6 Αρχική θέση βραχίονα

Ο ρομποτικός βραχίονας έχει την αρχική θέση. Η θέση που φαίνεται στην Εικόνα 3.19. Στην αρχική θέση κάθε γωνία άρθρωσης έχει μηδέν μοίρες ή ακτίνια. Στο rqt_joint_trajectory_controller αυτό αντιστοιχεί στο κέντρο της μπάρας κάθε joint Εικόνα 3.20.



Εικόνα 3.19 – Βραχίονας έχει την αρχική θέση

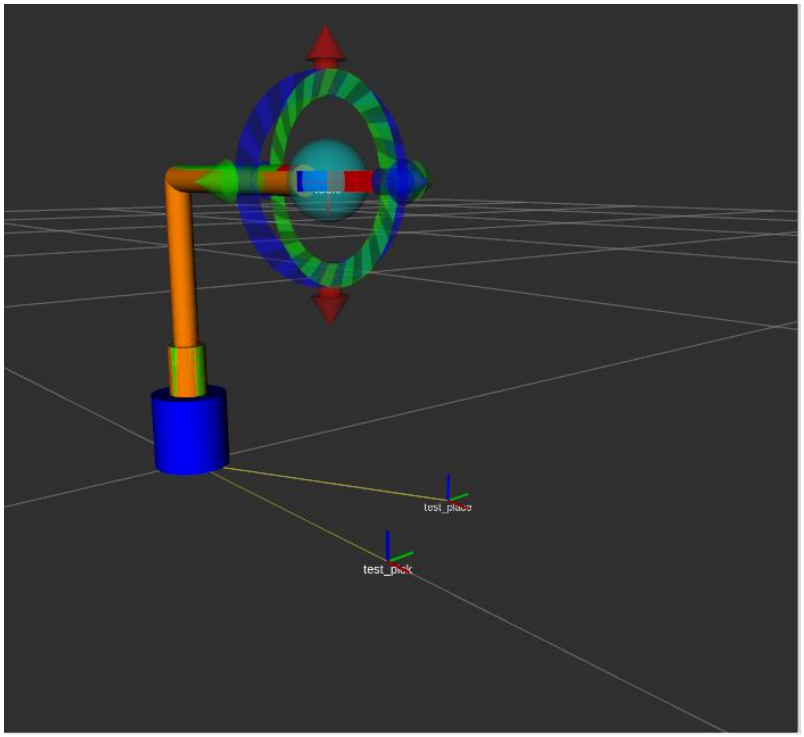


Εικόνα 3.20 – Ο controller_manager στην αρχική θέση

3.7 Λειτουργία Pick and Place του ρομποτικού βραχίονα

Για την επίτευξη μια απλής κίνησης pick and place σε ένα pyhon script αρχικά κάνοντας publish δύο σημεία tf με την χρήση της **tf::TransformBroadcaster Class**. Το πρώτο αντιστοιχεί στην θέση(Εικόνα 3.21- test_pick) στην οποία βρίσκεται το αντικείμενο, το οποίο θέλουμε να μετακινήσουμε, και το δεύτερο η επιθυμητή θέση(Εικόνα 3.21- test_place) που θέλουμε να το μεταφέρουμε.

```
def publish_transforms(self, e):
    self.br.sendTransform(self.constructTransform("world","test_pick",0.25,0,0,0,0,0))
    self.br.sendTransform(self.constructTransform("world","test_place",0.2,0.1,0,0,0,0))
```



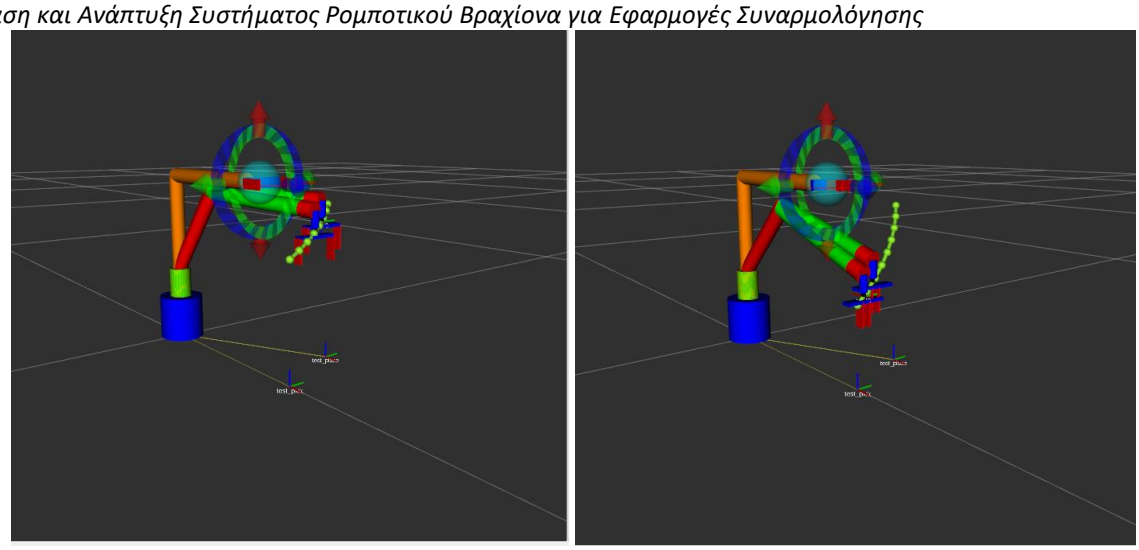
Εικόνα 3.21 – Test position και Pick position

Η κίνηση του βραχίονα στην επιθυμητή θέση σε σχέση με τον end-effector επιτυγχάνεται με την χρήση της **moveit_python.move_group_interface.MoveGroupInterface Class**:

```
def demo(self):
    time.sleep(0.5)
```

```
        # aproach
        self.send_movetopose("test_pick",0,0,0.1,0,180,0)
        # grasp pose
        self.send_movetopose("test_pick",0,0,0.05,0,180,0)
        # close gripper
        self.send_gripper_command(False)
        time.sleep(0.5)
        # lift
        self.send_movetopose("tool0",0,0,-0.1,0,0,0)
        # lift place pose1
        self.send_movetopose("test_place",0,0,0.2,0,180,0)
        #approach place pose
        self.send_movetopose("test_place",0,0,0.1,0,180,0)
        #place pose
        self.send_movetopose("test_place",0,0,0.08,0,180,0)
        # open gripper
        self.send_gripper_command(True)
        time.sleep(0.5)
        # lift place pose2
        self.send_movetopose("test_place",0,0,0.2,0,180,0)
```

Η κίνηση καταγράφεται στο Rviz όπως φαίνεται στην Εικόνα 3.22.

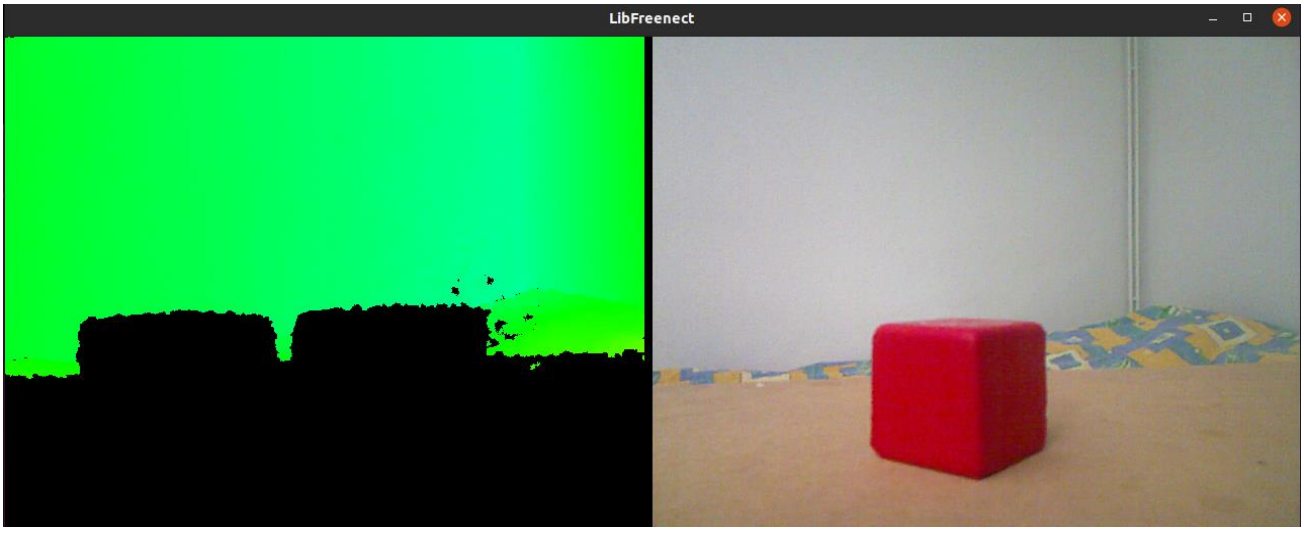


Εικόνα 3.22 – Pick and Place στο RViz

Στον σύνδεσμος του βίντεο στο YouTube όπου παρέχεται φαίνεται η λειτουργία pick and place: https://youtube.com/shorts/-CrQX3gdOlQ?feature=share

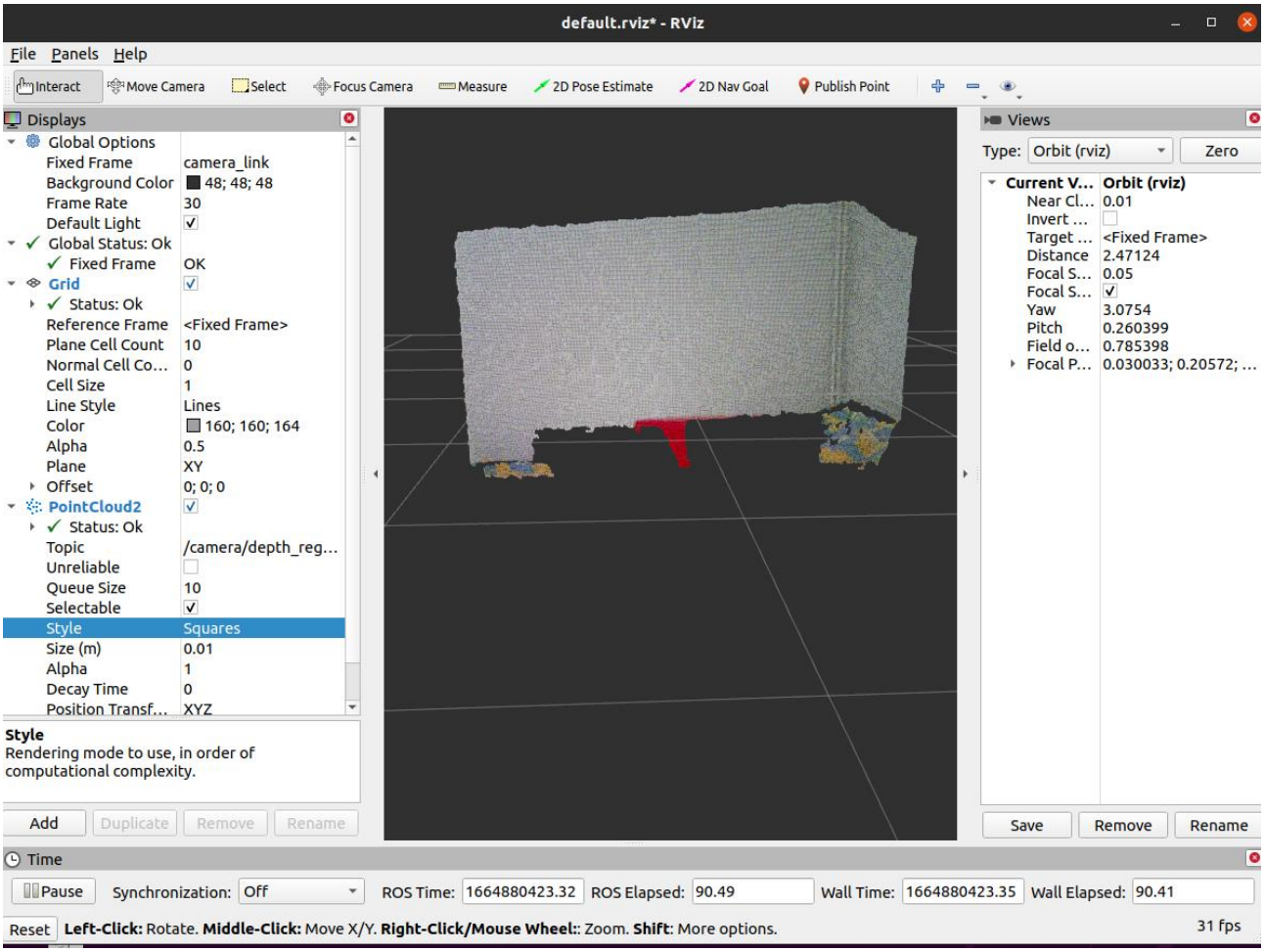
3.8 Λειτουργία Kinect

Η σύνδεση του Kinect επιτεύχθηκε με την χρήση της βιβλιοθήκης Freenect η οποία παρέχεται δωρεάν. Μετά την εγκατάσταση του Freenect δίνεται η δυνατότητα λήψης εικόνας από τις κάμερες του Kinect(Εικόνα 3.23). Από την μία κάμερα μπορεί να δοθεί η εικόνα βάθους ενώ την άλλη η πραγματική εικόνα της VGA κάμερας. Με την χρήση της εντολής : freenect-glview



Εικόνα 3.23 – Freenect-glview

Στην συνέχεια ανοίγοντας το Rviz και κάνοντας subscribe στο κατάλληλο topic (/camera/depth_registered/points) εμφανίζεται η εικόνα του Kinect.



Εικόνα 3.24 – Kinect στο RViz

Αυτό είναι το πρώτο βήμα για την μελλοντική ανάπτυξη της εφαρμογής αναγνώρισης αντικειμένων.

4 Επίλογος

4.1 Εισαγωγή κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάζεται η σύνοψη της διπλωματικής εργασίας με θέμα την Σχεδίαση και Ανάπτυξη Συστήματος Ρομποτικού Βραχίονα για Εφαρμογές Συναρμολόγησης. Θα γίνει περιγραφή τόσο των προβλημάτων που προέκυψαν αλλά και του τρόπου αντιμετώπισης τους. Περιγράφονται παρατηρήσεις και συμπεράσματα ,καθώς και μελλοντικές προτάσεις για την ανάπτυξη του project.

4.2 Σύνοψη

Ο στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μιας διεπαφής ελέγχου για το λειτουργικό ROS, η οποία θα λειτουργεί με διαφορετικούς βραχίονες έξι βαθμών ελευθερίας με αρπάγη. Στο πρώτο κεφάλαιο έγινε μια αναφορά στις βασικές έννοιες της ρομποτικής και του θεωρητικού μέρους της κινηματικής ανάλυσης του ρομποτικού βραχίονα. Στο δεύτερο κεφάλαιο έγινε ανάλυση και επεξήγηση των βασικών μερών του λογισμικού, τα οποία χρησιμοποιήθηκαν για τον έλεγχο του βραχίονα. Η κατασκευή του βραχίονα μέσω 3D εκτυπωτή μαζί με τα εξαρτήματα που χρησιμοποιήθηκαν παρουσιάστηκαν στο τρίτο κεφάλαιο. Η προσέγγιση και τα εργαλεία που πραγματοποιούν τον έλεγχο για το ρομπότ στο ROS, αλλά και αναπτύσσουν την εφαρμογή pick and place αναλύθηκαν στη συνέχεια του κεφαλαίου. Στο τελευταίο μέρος του τέταρτου κεφαλαίου παρουσιάζεται ένα τμήμα του κώδικα που ήταν απαραίτητος για την ανάπτυξη της εφαρμογής, αλλά και η λειτουργία του kinect, η οποία θα χρησιμεύσει μελλοντικά τόσο για την αποφυγή συγκρούσεων όσο και για την αναγνώριση αντικειμένων.

4.3 Προβλήματα – Αντιμετώπιση

Η εκτύπωση 3D αντικειμένων αποδείχθηκε ιδιαίτερα απαιτητική διαδικασία, καθώς κατά τη συναρμολόγηση και τη δοκιμή των σερβοκινητήρων δημιουργήθηκαν ρωγμές στα εκτυπωμένα κομμάτια. Οι εκτυπώσεις που έγιναν μέχρι την επιτυχή συναρμολόγηση του βραχίονα ήταν πολλές. Το κομμάτι που αποδείχθηκε πιο ευαίσθητο ήταν η αρπάγη, καθώς οι δαγκάνες και το κομμάτι τις βάσης της έσπασαν κατά τη δοκιμή του εύρους λειτουργίας του σερβοκινητήρα.

4.4 Παρατηρήσεις - Συμπεράσματα

Η απόκλιση μεταξύ του μοντέλου που προέκυψε από το URDF αρχείο και του φυσικού ρομπότ είναι αδιαμφισβήτητη. Αυτό οφείλεται στην έλλειψη ακρίβειας κατά τη μέτρηση των διαστάσεων του φυσικού ρομπότ, αλλά και στην αδράνεια που έχει το ρομπότ λόγω του βάρους των σερβοκινητήρων. . Με τη χρήση τόσο του Moveit και του Rviz όσο και κάποιων ros messages επιτεύχθηκε η βαθμονόμηση του βραχίονα και ο ορισμός μίας αρχικής θέσης του. Έτσι αντιμετωπίστηκαν οι εν λόγω περιορισμοί και ολοκληρώθηκε η επίτευξη του στόχου της εργασίας. Η τελική κίνηση για το pick and place επιτυγχάνεται απλά ορίζοντας δύο σημεία -ένα για το σημείο που βρίσκεται το αντικείμενο και άλλο ένα για το επιθυμητό σημείο μεταφοράς του. Κατά τη διάρκεια της μεταφορά του κύβου προστέθηκαν κάποια ενδιάμεσα σημεία -σε σχέση πάντα με τον end-effector-, ώστε η κίνηση να είναι πιο ομαλή αποφεύγοντας τη σύγκρουση με το δάπεδο.

4.5 Προτάσεις μελλοντικής εξέλιξης

Μία πρόταση για τη μελλοντική ανάπτυξη του συγκεκριμένου project είναι η αναγνώριση αντικειμένων με τη χρήση της κάμερας kinect. Κάτι τέτοιο θα προσφέρει στο σύστημα του βραχίονα την αυτονομία υπολογισμού της κίνησης σύμφωνα με την τοποθεσία του εκάστοτε αντικειμένου.

Επίσης, η αναγνώριση αντικειμένων θα δώσει τη δυνατότητα συνδυασμού του βραχίονα με έναν ταινιόδρομο, ο οποίος θα μπορεί να μεταφέρει αντικείμενα διαφόρων σχημάτων.

Επιπρόσθετα, ο σχεδιασμός μίας διαφορετικού τύπου αρπάγης που ταιριάζει στον συγκεκριμένο custom βραχίονα θα επεκτείνει το εύρος των εργασιών που θα εκτελεί.

Επίσης, η δημιουργία ενός driver για έλεγχο του βραχίονα με τηλεχειριστήριο θα έδινε την δυνατότητα σε έναν μη εξειδικευμένο χρήστη της εφαρμογής να χειριστεί με ευκολία τον βραχίονα.

Τέλος, η χρησιμοποίηση διάταξης και μοντέλων ελέγχου μέσω ηλεκτροεγκεφαλικών σημάτων του χειριστή είναι μια εξαιρετική μέθοδος για τη χρήση του ρομποτικού συστήματος από άτομα με περιορισμούς ή απώλεια της δυνατότητας κίνησης των άνω άκρων τους [22].

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] #HowToRobot - Grippers and End Effectors: Uses, Benefits, and Cost Analysis https://www.howtorobot.com/expert-insight/robot-end-effectors

[2] ROBOTIQ - Robot End Effector: Definition and Examples: https://blog.robotiq.com/bid/53266/Robot-End-Effector-Definition-and-Examples

[3] UNIDEX - Different Types of End Effectors: https://www.unidex-inc.com/blog/end-effector-what-it-is-how-it-works-types-applications/

[4] ROBOTIQ - Applications: https://robotiq.com/applications

[5] Lambda Geeks - Robot End Effector: 7 Σημαντικά Χαρακτηριστικά: https://el.lambdageeks.com/robot-end-effector/

[6] ROS.org – ROS:http://wiki.ros.org/ROS/

[7] Ainstein.ai – Understand ROS: https://ainstein.ai/an-introduction-to-the-robot-operating-system-ros/

[8] ROS.org – Nodes: http://wiki.ros.org/Nodes

[9] The Robotics Back–End -What is a ROS Node?:https://roboticsbackend.com/what-is-a-ros-node/#A_mobile_robot_controlled_by_a_camera

[10] ROS.org – actionlib :http://wiki.ros.org/actionlib

[11] Clearpath Robotics- Launch Files http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Launch%20Files.html#writing-a-launch-file

[12] ROS.org – Using rqt_console and roslaunch: http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch

[13] ROS.org – Writing a Simple Publisher and Subscriber (C++): http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

[14] ROS Notes – Xacro: https://abedgnu.github.io/Notes-ROS/chapters/ROS/10_robot_modeling/xarco.html

[15] ROS.org – rqt_joint: http://wiki.ros.org/Robots/TIAGo/Tutorials/motions/rqt_joint

[16] Μπουτάλης, Γιάννης Σ., Ρομποτική Ανάλυση, Έλεγχος και Προγραμματισμός Ρομποτικών Χειριστών Σταθερής Βάσης, Εκδόσεις ΚΡΙΚΟΣ, 2017.

[17] ROS Notes – Topics: http://wiki.ros.org/Topics

[18] MoveIt Motion Planning Framework - MoveIt Setup Assistant: http://docs.ros.org/en/kinetic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html

[19] ROS.org – ros control: http://wiki.ros.org/ros_control

[20] ROS.org – link: http://wiki.ros.org/urdf/XML/link

[21] ROS.org – joint: http://wiki.ros.org/urdf/XML/joint

[22] Korovesis, N., Kandris, D., Koulouras, G., & Alexandridis, A. (2019). Robot motion control via an EEG-based brain–computer interface by using neural networks and alpha brainwaves. *Electronics*, *8*(12), 1387.