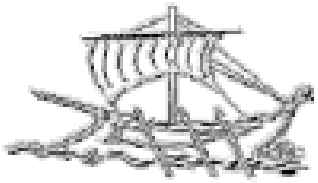


ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. "ΕΦΑΡΜΟΣΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Control and modeling of an industrial robotic arm

Γεώργιος Μανής

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Control and modeling of an industrial robotic arm

Γεώργιος Μανής
A.M. AIS0078

Εισηγητής:

Δρ Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Αναστασία Βελώνη, Καθηγήτρια
Πάρις Μαστοροκώστας, Καθηγητής

Ημερομηνία εξέτασης: 17/10/2017

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γεώργιος Μανής, του Δημητρίου, με αριθμό μητρώου AIS 0078 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία έγινε μετά απο αρκετή προσπάθεια στο πανεπιστήμιο του Cranfield στην Αγγλία. Θα ήθελα να ευχαριστήσω τον καθήγητη μου Al.Savvaris που ήταν πάντα παρών όταν τον χρειαζόμουν και τον διδακτορικό φοιτητή Κωνσταντίνο Μαλανδράκη για τις συμβουλές του.

Ακόμα θα ήθελα να ευχαριστήσω τον καθηγητή μου Ιωάννη Έλληνα που με παρότρυνε να κάνω τη διπλωματική μου εργασία στο συγκεκριμένο πανεπιστήμιο και με βοήθησε με την εργασία όσο μπορούσε. Τέλος σημαντικό ρόλο είχε η οικογένεια μου που με υποστήριζαν ψυχολογικά όλους αυτούς του μήνες.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία αφορά την μοντελοποίση και τον απομακρυσμένο έλεγχο μέσω διακομιστή ενός ρομποτικού βραχίονα, ονόματι UR5, η οποία έλαβε χώρα στο Πανεπιστήμιο του Cranfield σε συνεργασία με το πανεπιστήμιο του Πεκίνου. Για τον έλεγχο του δημιουργήθηκαν τρία προγράμματα, εκ των οποίων το ένα με τη βοήθεια του Matlab, είναι υπεύθυνο για την αποστολή επιθυμητών εντολών, το δεύτερο είναι σε κώδικα Java και χρησιμοποιείται στον server για την μετάδοση δεδομένων και το τρίτο είναι γραμμένο σε URScript στο controller του UR5 και είναι υπεύθυνο για την κίνηση του βραχίονα. Η μοντελοποίηση του UR5 έγινε στο πρόγραμμα Matlab και αποτελείται από τη δημιουργία οδεύσεων, την αντίστροφη κινηματική, τη δυναμική και τέλος το μοντέλο του.

ABSTRACT

The following thesis contains the modeling and remote control via server of a robotic arm, named UR5, at Cranfield University in cooperation with the University of Beijing. At first, for its control, three programs were created, one of them by using Matlab, which is responsible for sending covetable commands. The second one, which is written in Java, is used for transmitting data through the server. The controller of the UR5 operates with the third and final program, which is written in URScript and is responsible for its motion. The modeling of UR5 was made in Matlab and consists of inverse kinematic, trajectory creation, dynamics and its model.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: ‘Ελεγχος και μοντελοποίηση ρομποτικού βραχίονα
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: matlab, inverse kinematic, dynamics, server, controller

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ 16
1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας 16
1.2 Ιστορική αναδρομή 17
1.3 Βασικές έννοιες 20
2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΒΡΑΧΙΟΝΑ 24
2.1 Κινηματική βραχίονα 25
2.1.1 Αναπαραστάσεις σωμάτων 25
2.1.2 Ευθύγραμμη κινηματική 31
2.1.3 Αντίστροφη κινηματική 32
2.1.4 Κινηματική UR5 33
2.2 Δημιουργία οδεύσεων 38
2.3 Δυναμική και έλεγχος του βραχίονα 43
2.4 Μοντέλο του UR5 47
3. ΕΛΕΓΧΟΣ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ 54
3.1 Δημιουργία προγράμματος στο Matlab 54
3.1.1 Έλεγχος του UR5 μέσω του Matlab 54
3.1.2 Πρόγραμμα ανατροφοδότησης 61
3.2 Δημιουργία προγράμματος για το Controller του UR5. 65
3.3 Πρόγραμμα διακομιστή 68
4. ΣΥΝΔΕΣΗ ΤΟΥ UR5 ΜΕ ΤΟ MATLAB 72
4.1 Αποτελέσματα πειράματος 74
4.2 Συμπεράσματα πειράματος για μελλοντική χρήση 76

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1.3.1: Σταθερός τροχός, πηγή [7]
Σχήμα 1.3.2: Καθοδηγίσιμος τροχός, πηγή [7]
Σχήμα 1.3.3: Προσανατολισμένος τροχός, πηγή [7]
Σχήμα 2.1: Μοντελοποίηση UR5 στο Matlab.
Σχήμα 2.1.1: Αναπαράσταση Διανύσματος θέσης, πηγή [6]
Σχήμα 2.1.1.1: Γραφική αναπαράσταση πλαισίου, πηγή [7]
Σχήμα 2.1.2.1: Πλαίσια και μέλη του βραχίονα, πηγή [1]
Σχήμα 2.1.4.1: Διαστάστεις UR5. Πηγη [8]
Σχήμα 2.1.4.2: Simulink για το το πρόβήμα της αντίστροφης κινηματικής
Σχήμα 2.2.1: Simulink για τη δημιουργία τροχιάς
Σχήμα 2.3.1: Σύστημα ελέγχου βραχίονα, πηγή [1]
Σχήμα 2.4.1: Μοντέλο του UR5 στο Matlab.
Σχήμα 2.4.2: Μοντέλο του UR5 στο Simulink.
Εικόνα 2.4.1 Ρυθμίσεις βαρύτητας.
Σχήμα 2.4.3 Συνδέσεις σωμάτων
Εικόνα 2.4.2: Ρυθμίσεις σωμάτων
Εικόνα 2.4.3: Ρυθμίσεις συντεταγμένων σώματος.
Σχήμα 2.4.4: Άρθρωση βραχίονα
Εικόνα 2.4.4: Ρυθμίζεις άρθρωσης.
Σχήμα 3.1.1.1: Δημιουργία πίνακα επιθυμητής θέσης και προσανατολισμού.
Εικόνα 3.1.1.1: To instrument
Εικόνα 3.1.1.2: Αποστολή πρώτης εντολής
Εικόνα 3.1.1.3: Δήλωση στοιχείων που θα αποσταλούν.
Εικόνα 3.1.1.4: Επιλογή τρόπου αποστολής
Εικόνα 3.1.1.5: Δημιουργία αντικειμένου.
Σχήμα 3.1.1.2: Αποστολή δεδομένων στο Simulink.
Εικόνα 3.1.2.1: Query Instrument
Εικόνα 3.1.2.1: Λήψη δεδομένων.
Εικονα 3.1.2.2: Display.
Εικόνα 3.1.2.3: Scope
Εικόνα 3.1.2.4: Το workspace

Εικόνα 3.1.2.5: Το file.
Σχήμα 3.1.2.1: Σύστημα ανατροφοδότησης στο Simulink.
Εικονα 4.1: Σύνδεση Server με τους Clients, πηγή [9]
Σχήμα 4.1: Τελικό σχέδιο στο Simulink.
Εικόνα 4.1.1: Αρχική θέση.
Εικόνα 4.1.2: Τελική θέση.
Εικόνα 4.1.3: Οι εντολές που διαβάζονται στο server.

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 2.1.2.1: Παράμετροι μελών του βραχίονα, πηγή [1]
Πίνακας 2.1.4.1: Πίνακας παραμέτρων, πηγή [1]

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα αναλυθεί το αντικείμενο της διπλωματικής εργασίας που έχει να κάνει με τον απομακρυσμένο έλεγχο ενός ρομπτικού βραχίωνα ανάμεσα σε δύο χώρες και τη μοντελοποίηση του. Στη συνέχεια θα γίνει μία σύντομη ιστορική αναδρομή σχετικά με τους ρομποτικούς βραχίονες απο τη μυθολογία μέχρι σήμερα και τέλος θα δοθούν κάποιες βασικές έννοιες σχετικά με τη ρομποτική.

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι ο απομακρυσμένος έλεγχος και η μοντελοποίηση ενός βιομηχανικού ρομποτικού βραχίονα. Ο συγκεκριμένος βραχίονας έχει 6 βαθμούς ελευθερίας (degrees of freedom), ονομάζεται UR5 και βρίσκεται στο Πανεπιστημίο του Πεκίνου. Ο έλεγχος του γίνεται από έναν υπολογιστή στο πανεπιστήμιο του Cranfield στο Ηνωμένο Βασίλειο που έχει εγκατεστημένο το πρόγραμμα του Matlab.

Αρχικά δημιουργούνται οι επιθυμητές εντολές απο το Matlab, μεταφέρονται μέσω internet με τη μέθοδο TCP/IP στον ελεγκτή (controller) του UR5 και η μεταφορά αυτή περιλαμβάνει έναν διακομιστή (server) που βρίσκεται στο πανεπιστήμιο του Πεκίνου που είναι προγραμματισμένος σε κώδικα Java. Η γλώσσα προγραμματισμού του ελεγκτή (controller) ονομάζεται URscript και μετατρέπει τα δεδομένα που λαμβάνει σε κίνηση του ρομποτικού βραχίονα.

Το πρόγραμμα του Matlab εκτός απο την αποστολή των εντολών στο UR5 περιλαμβάνει επίσης την μοντελοποίηση του. Δηλαδή:

[1] Την απεικόνιση του UR5.
[2] Το δυναμικό σύστημα για τη σωστή κίνηση του.
[3] Την επιθυμητή τροχιά που θα ακολουθήσει.
[4] Η αντίστροφη κινηματική για την εύρεση των γωνιών του κάθε βραχίονα.

Τέλος φτιάχτηκε ένα σύστημα ανατροποδότησης προκειμένου να μπορεί ο χρήστης να συγκρίνει τις αρχικές και τελικές τιμές των γωνιών της κάθε άρθρωσης του βραχίονα ή το χρόνο καθυστέρησης (delay time) των δεδομένων.

1.2 Ιστορική αναδρομή

Η έννοια της λέξης ρομπότ έχει ξεκινήσει απο αρχαιοτάτων χρόνων με την επιθυμία του ανθρώπου να κατασκευάσει ένα υποκατάστατο του, το οποίο τον μιμείται στην εκτέλεση εργασιών. Τέτοια παραδείγματα παρατηρούνται στην ελληνική μυθολογία, με παραδείγματα όπως του **Προμηθέα** και του **Τάλου**. Ο Προμηθέας ήταν τιτάνας, γιός του Ιαπετου και θεωρείται ότι κατασκεύασε τους πρώτους ανθρώπους απο πυλό και φωτιά. Ο Ταλώς με τη σειρά του ήταν ένας χάλκινος γίγαντας, ο οποίος θεωρείται το πρώτο ρομπότ που αναφέρεται στη μυθολογία και δόθηκε απο τον Δία στον μινωικό πολιτισμό για να προστατεύει την Κρήτη απο τους εισβολείς.

Έπειτα παρατηρείται η επιθυμία αυτή στο βιβλίο της Αγγλίδας συγγραφέα Μαίρη Σέλεϊ με τίτλο «**Φρανκενσταϊν ή ο Σύγχρονος Προμηθέας**» που εκδόθηκε το 1816. Πρόκειται για ένα γοτθικό μυθιστόρημα που περιγράφει έναν νεαρό επιστήμονα, ο οποίος έκανε διάφορα πειράματα ώστε να δώσει ζωή σε έναν νεκρό άντρα.

Στη συνέχεια σειρά πήρε το θέατρο, με έργα μεγάλων σκηνοθετών και ποιο γνωστό «**Τα Παγκόσμια Ρομπότ του Rossum**» που γράφτηκε το 1920 απο τον Τσέχο συγγραφέα Karel Capek.

Οι έννοιες ρομπότ όμως σαν μεταλλική κατασκευή και ρομποτική δόθηκε πρώτη φορά απο τον συγγραφέα Isaac Asimov το 1940. Σύμφωνα με αυτόν η επιστήμη της ρομποτικής βασίζεται σε τρείς νόμους:[1]

1. Ένα ρομπότ δεν μπορεί να τραυματίσει ή μέσω αδράνειας να βλάψει ένα ανθρώπινο ον.

[1] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villiani, Giuseppe Oriolo,"Ρομποτική", εκδόσεις Fountas, 2009

2. Ένα ρομπότ πρέπει να υπακούει τις εντολές που δίνονται απο ανθρώπους,εκτός και αν τέτοιες εντολές έρχονται σε αντίθεση με το πρώτο νόμο.
3. Ένα ρομπότ πρέπει να προστατεύει την ίδια του την ύπαρξη, εφόσον μία τέτοια προστασία δεν έρχεται σε αντίθεση με τον πρώτο ή τον δεύτερο νόμο.

Όμως η ρομποτική δεν άργησε να μεταβεί και στη μεγάλη οθόνη. Το **Star Wars** είναι μία σειρά ταινιών επιστημονικής φαντασίας που δημιουργήθηκε το 1977 απο τον Τζόρτζ Λούκας, απέκτησε πάρα πολλούς θαυμαστές και η οποία συνεχίζεται μέχρι και τώρα. Στις ταινίες αυτές σημαντικό ρόλο έχουν δύο ρομπότ ονόματι R2-D2 και C-3PO, τα οποία μπορούν να επικοινωνούν μεταξύ τους.

Μία ακόμα ταινία που άρεσε υπερβολικά στο κοινό είναι το «**The Terminator**». Πρόκειται για ένα ρομπότ απο το μέλλον που έχει τη μορφή ανθρώπου και προγραμματίζεται ώστε να γυρίσει στο παρελθόν με σκοπό να σκοτώσει έναν συγκεκριμένο άνθρωπο. Η ταινία αυτή γυρίστηκε το 1984 και διαδραματίζεται στο Los Angeles.

Μία διαφορετική άποψη σχετικά με τη ρομποτική παρουσιάζεται στη ταινία «**A Space Odyssey**». Η ταινία γυρίστηκε απο τον Κιούμπρικ το 2001 και παρουσιάζει τα αρνητικά της ρομποτικής. Το ρομποτ ονόματι HAL 9000 παρακούει τις εντολές ενός αστρονάυτη και δρά μόνο του.

Εκτός απο το θέατρο και το σινεμά, επιστήμονες απο όλο το κόσμο προσπαθούν καθημερινά να φτίαξουν όλο και καλύτερα ρομπότ. Το πρώτο παραδείματα είναι τα «**Grey Walter's tortoises**». Πρόκειται για δύο αυτόνομες μηχανές που κατασκευάστηκαν το 1948 απο τον βρετανό Γουίλιαμ Γκρέι Γουόλτερ χρησιμοποώντας ξυπνητήρια. Προκειμένου να κινηθούν χρησιμοποιήθηκαν αισθητήρες φωτός και αφής.

Στη συνέχεια η ρομποτική πήρε διάσταση και στη βιομηχανία με το «**Unimate**». Είναι ο πρώτος ρομποτικός βραχίονας που χρησιμοποιήθηκε το 1961 σε βιομηχανία. Ζύγιζε 2 τόνους και αξιοποιήθηκε σε γραμμή συναρμολόγησης στη βιομηχανία General Motos.

Λίγα χρόνια αργότερα οι ρομποτικοί βραχίονες βρήκαν θέση και στην ιατρική με το **Rancho Arm**. Δημιουργήθηκε το 1963, πρόκειται για ένα

ρομπότ με έξι αρθρώσεις και χρησιμοποιήθηκε στο νοσοκομείο Λος Αμίγκος της Καλιφόρνια ωστε να δημιουργηθούν τεχντητά μέλη. Οι κινήσεις παρομοίαζαν αυτές του ανθρώπινου χεριού και οι εντολές δίνονταν απο ηλεκτρονικό υπολογιστή.

Μεγάλο πρόβλημα μέχρι εκείνη τη στιγμή ήταν οτι η ρομποτική ήταν αρκετα στατική. Για το λόγο αυτό, οι επιστήμονες προσπάθησαν να δημιουργήσουν ένα ρομπότ που να κινείται και το κατάφεραν μέσω του «**Shakey**». Το περιοδικό Life το αναφέρει ως «Το πρώτο ηλεκτρονικό άτομο» και πρόκειται για ένα κινούμενο ρομπότ που κατασκευάστηκε στο Stanford Research Institute to 1970. Περιέχει σύστημα ανατροφοδότησης με έναν υπολογιστή και αισθητήρες χτυπήματος ή άλλους τρόπου συλλογής δεδομένων. Η χρήση του εστίαζε κυρίως σε ξεκλείδωμα πορτών και διακοπτών.

Τα επόμενα χρόνια δόθηκε ιδιαίτερο ενδιαφέρον στην ακρίβεια κίνησης στους ρομποτικούς βραχίονες. Κατασκευάστηκε ο **βραχίονας άμεσης προώθησης** απο τον Τακέο Κανάντε το 1981 και ήταν ο πρώτος με διαφορετικούς κινητήρες σε κάθε άρθρωση. Έτσι οι βραχίονες κατάφεραν να γίνουν πιο γρήγοροι και με μεγαλύτερη ακρίβεια.

Ένα μεγάλο βήμα στη συνέχεια ήταν η χρήση τους υπο αντίξοες συνθήκες. Πρόκειται για το «**Dante 1**» και είναι το πρώτο ρομπότ που χρησιμοποιήθηκε για μελέτη περιβαλλοντικού φαινομένου στο κρατήρα του ηφαιστείου Σπουρ της Αλάσκα. Επειδή οι θερμοκρασίες ήταν υπερβολικά υψηλές ,οι άνθρωποι δεν μπορούσαν να προσεγγίσουν το σημείο για να συλλεξουν πληροφορίες και χρησιμοποίησαν το συγκεκριμένο ρομπότ. Κατασκευάστηκε το 1992 στο πανεπιστήμιο Κάρνεγκι Μελον και είχε τη μορφή αράχνης.

Ένα επιπλέον παράδειγμα ιατρικής χρήσης είναι το σύστημα «**Da Vinci**». Είναι το πρώτο σύστημα χειρουργικής ρομποτικής που πήρε έγκριση απο τον Αμερικανικό Οργανισμό Φαρμάκων και Υλικών FDA το 2000. Περιέχει ειδικούς βραχίονες, εναν ενδοσκοπικό πύργο και τη χειρουργική κονσόλα. Με τη χρήση αυτού του συστήματος ο ιατρός έχει μεγαλύτερη ακρίβεια έχοντας μπροστά του μόνο μία μεγενθυμένη εικόνα απο το χειρουργικό πεδίο (καλύτερη εστίαση) και τους βραχίονες.

Η ρομποτική απο το 2000 και μετά εκτός απο την βιομηχανία, την ιατρική ή για την εξαγωγή πληροφοριών σε περιβαλλοντικα φαινόμενα, έχει αρχίσει και

χρησιμοποιείται και στην καθημερινότητα των ανθρώπων. Ένα παράδειγμα είναι ο καθαριστής **Roomba** για οικιακή χρήση που κατασκευάστηκε απο την iRobot to 2002.

1.3 Βασικές έννοιες

Στη συγκεκριμένη υποενότητα θα αναλυθούν οι δομές των ρομπότ, σημαντικές έννοιες όπως οι αρθρώσεις, οι βαθμοί ελευθερίας και ο χώρος εργασίας ενός ρομποτικού βραχίονα. Επίσης ιδιάιτερη έμφαση θα δωθεί στην απεικόνιση αντικειμένων σε τρισδιάστατο χώρο.

Σημαντικό στοιχείο στους ρομποτικούς βραχίονες είναι η δομή τους και η ταξινόμηση τους γίνεται ανάλογα τη βάση τους. Στη περίπτωση που η βάση είναι σταθερή τότε ονομάζονται ρομποτικοί χειριστές, ενώ στη περίπτωση κινούμενης βάσης κινούμενα ρομπότ.

- Ρομποτικοί Χειριστές

Οι ρομποτικοί χειριστές αποτελούνται απο τον **βραχίονα** (arm) για την κινητικότητα του, τον **καρπό** που χρησιμεύει στην επιδεξιότητα και το **τελικό στοιχείο δράσης** (end effector) για την εκτέλεση των εργασιών.Τα μέλη αυτά συνδέονται μέσω **αρθρώσεων** (joints) και ανάλογα τη επιθυμητή συνδεσμολογία οι πιο γνωστές αρθρώσεις είναι:

- Πρισματική άρθρωση (prismatic)
- Στροφική άρθρωση (revolute)

Ανάλογα το εργασία που πρέπει να υλοποιηθεί, το πλήθος των αρθρώσεων αλλάζει. Το πλήθος αυτό προσδιορίζει τους **βαθμούς ελευθερίας** του βραχίονα (DOF-degrees of freedom) για τη συγκεκριμένη εργασία. Για παράδειγμα, η μετακίνηση αντικειμένου σε τρισδιάστατο χώρο απαιτείται απαιτεί βραχίονα έξι αρθρώσεων για τη σωστή θέση προσανατολισμό του αντικειμένου. Η προσδιορισμός θέσης και προσανατολισμού θα αναλυθεί παρακάτω.

Ένα ακόμα σημαντικο στοιχείο είναι ο **χώρος εργασίας** στον οποίο εργάζεται ο βραχίονας. Ο χώρος εργασίας αποτελεί το τμήμα του περιβάλλοντος όπου έχει προσβαση το τελικό στοιχείο δράσης του χειριστή. Το σχήμακαι ο όγκος του εξαρτάται τόσο απο τη δομή του χειριστή όσο και απο την παρουσία ορίων των μηχανικών αρθρώσεων.[2]

Τέλος οι ρομποτικοί χειριστές ανάλογα το είδος των βαθμών ελευθερίας και την ακολουθία τους διαχωρίζονται σε:

I. Καρτεσιανούς (Cartesian)
II. Κυλινδρικούς (Cylindrical)
III. Σφαιρικούς (Spherical)
IV. Scara
V. Ανθρωπομορφοτικούς (Anthropomorphic)

- Κινούμενα Ρομπότ

Ένα κινούμενο ρομπότ έχει τη δυνατότητα να κινείται στο χώρο με τη χρήση στερεών σωμάτων και τα ποία χρησιμοποιούν σύστημα κίνησης (locomotion system). Υπάρχουν δύο κατηγοριές κινούμενων ρομπότ:

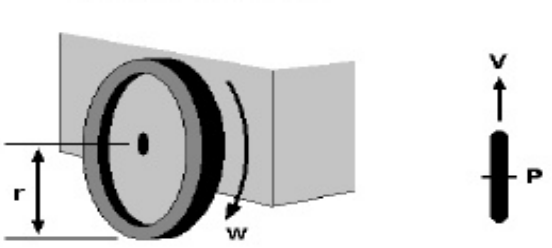
1. Τα τροχοφόρα (wheeled).
   Αποτελούνται απο δύο τμήματα, το άκαμπτο σώμα και το τμήμα κίνησης που εφάπτεται στο έδαφος.
2. Τα βραδίζοντα (legged).
   Αποτελείται απο πολλές αρθρώσεις, κάποιες απο τις οποίες συνδόνται σε στερεά σώματα για την παροδική επαφή τους με το έδαφος.

Κοινό στοιχείο των παραπάνω κατηγοριών είναι οι τροχοί. Με τη σειρά τους χωρίζονται σε :

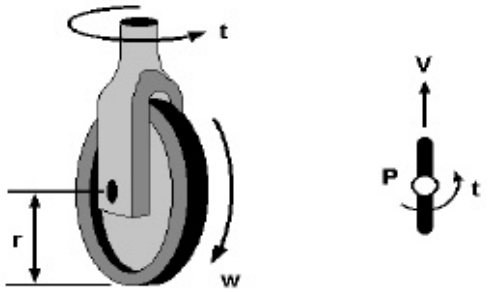
[2] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villiani, Giuseppe Oriolo,"Ρομποτική", εκδόσεις Fountas, 2009

1. Σταθερούς τροχούς (fixed wheel) που έχουν ένα βαθμό περιστροφής.
2. Καθοδηγίσιμος τροχός (steerable wheel) με δύο άξονες περιστροφής.
3. Προσανατολισμένος τροχός (caster wheel) με δύο άξονες πριστροφής και έχει διαφορά με τον καθοδηγίσιμο τροχό στο ‘οτι ο σταθερός άξονας δεν περνάει απο το κέντρο του τροχού.

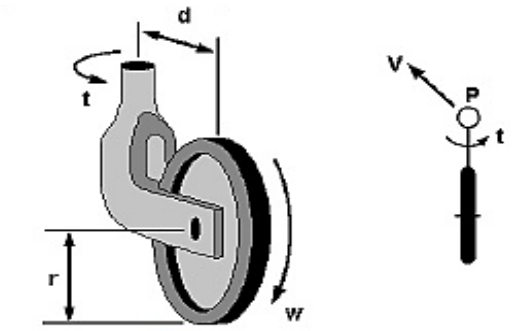
Τα σχήματα αυτών φαίνονται παρακάτω:



Σχήμα 1.3.1: Σταθερός τροχός



Σχήμα 1.3.2: Καθοδηγίσιμος τροχός



Σχήμα 1.3.3: Προσανατολισμένος τροχός

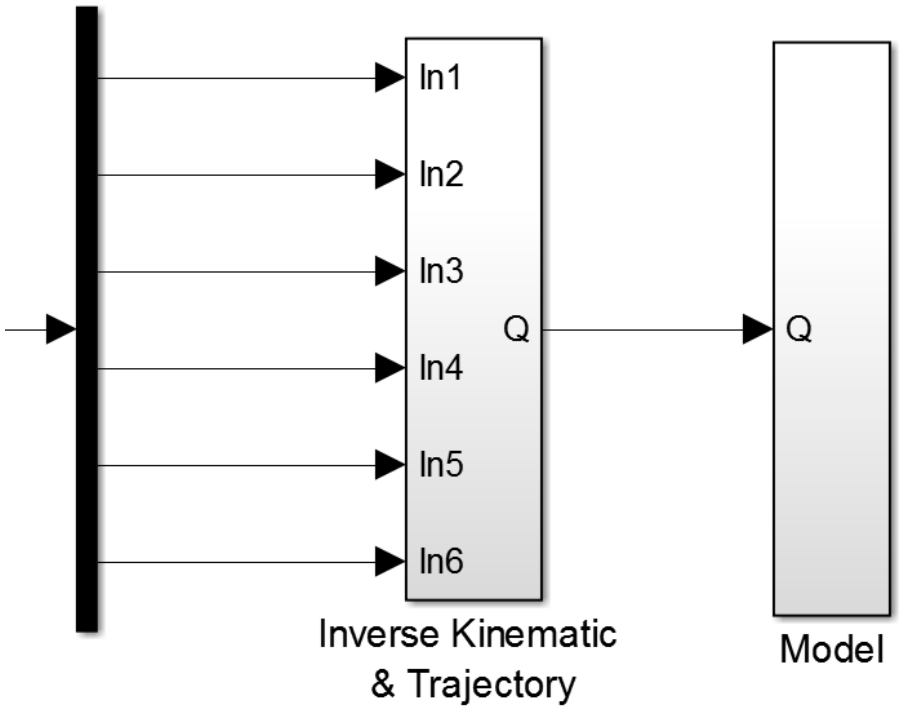
ΚΕΦΑΛΑΙΟ 2

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΒΡΑΧΙΟΝΑ

Για την συγκεκριμένη πτυχιακή εργασία έγινε μοντελοποίηση του ρομποτικού βραχίονα UR5, η οποία έγινε σε ειδικό πρόγραμμα που ονομάζεται Matlab (version 16a) και η διαμόρφωση του αποτελείται απο τέσσετα διαφορετικά τμήματα. Τα τμήματα αυτά είναι:

i) Η επίλυση της αντίστροφης κινηματικής του συγκεκριμένου βραχίονα.
ii) Δημιουργία της επιθυμητής τροχιάς.
iii) Έλεγχος και δυναμική του ρομποτικού βραχίονα.
iv) Μοντέλο του UR5.

Στο παρακάτω σχήμα φαίνεται το ολοκληρωμένο κομμάτι της μοντελοποίησης του UR5 στο Matlab. Στη συνέχεια θα αναλυθεί το κάθε τμήμα ξεχωριστά και πώς αλληλεπιδρούν μεταξύ τους.



Σχήμα 2.1: Μοντελοποίηση UR5 στο Matlab.

2.1 Κινηματική βραχίονα

Η κινηματική του βραχίονα έχει σχέση με τη θέση και τον προσανατολισμό του βραχίονα στο χώρο, όπως επίσης και με την ταχύτητα ή την επιτάχυνση του. Στη συνέχεια θα δωθούν βασικές έννοιες για την κατανόηση της θέσης, του προσανατολισμού και ο σωστός τρόπος εύρεσης της ευθύγραμμής και αντίστροφης κινηματικής του ρομποτικού βραχίονα. Τέλος θα βρεθούν οι απαραίτητες εξισώσεις της αντίστροφης κινηματικής του UR5.

2.1.1 Αναπαραστάσεις σωμάτων

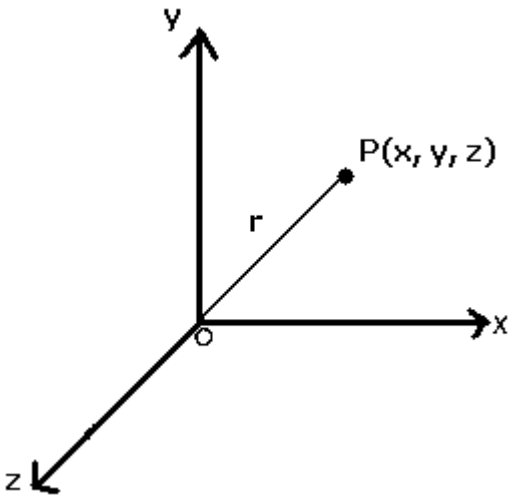
Η αναπαράσταση των σημάτων έχει ιδιαίτερο ρόλο στη ρομποτική και μας δίνει μια εικόνα για τη θέση και τον προσπανατολισμό ενός αντικειμένου ή του τελικού στοιχείου δράσης του βραχίονα στον τρισδιάστατο χώρο. Ιδιαίτερο βάρος δίνεται στον ορισμό του πλαισίου, που δίνει μία πιο ολοκληρωμένη εικόνα του αντικειμένου εφόσων περιλαμβάνει και τα δύο μαζί (θέση – προσανατολισμό). Στη συνέχεια θα δωθεί ο σωστός τρόπος εύρεσης προσανατολισμού του πλαισίου , το οποίο περιστρέφεται γύρω απο έναν ή περισσότερους άξονες.

<u>Θέση αντικειμένου:</u>

Ο προσδιορισμός της θέσης γίνεται μέσω του διανύσματος θέσης (position vector) , αναπαριστάται απο έναν πίνακα 3x1και συμβολίζεται ως ${}^{A}P$:

$$ {}^{A}P = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix} \quad (2.1.1.1) $$

,όπου το A είναι το σύστημα συντεταγμένων και τα Px,Py,Pz δίνουν τις συντεταγμένες του σημείου στον τρισδιάστατο χώρο.



Σχήμα 2.1.1: Αναπαράσταση Διανύσματος θέσης

Προσανατολισμός αντικειμένου:

Ο προσδιορισμός ενός αντικειμένου γίνεται αρχικά ορίζοντας ένα σύστημα συντεταγμένων, το προσάρτουμε στο σώμα και στη συνέχεια αναπαριστούμε αυτό το σύστημα συντεταγμένων ως προς το σημείο αναφοράς. Για να γίνει αυτό θα πρέπει ένα σώμα με σύστημα συντεταγμένων {B} να εκφραστεί με τρία μοναδιαία διανύσματα ως προς ένα σύστημα {A}. Δηλώνουμε τα μοναδιαία διανύσματα των τριών αξόνων του {B} μετα σύμβολα $\hat{X}_B$, $\hat{Y}_B$, $\hat{Z}_B$. Όταν τα εκφράζουμε ως προς το σύστημα συντεταγμένων {A}, συμβολίζονται με ${}^A\hat{X}_B$, ${}^A\hat{Y}_B$, ${}^A\hat{Z}_B$. Αποδεικνύεται βολικό να θεωρήσουμε ότι αυτά τα τρία μοναδιαία διανύσματα, αποτελούν στήλες ενός πίνακα 3x3 με διάτακη ${}^A\hat{X}_B$, ${}^A\hat{Y}_B$, ${}^A\hat{Z}_B$. Ονομάζουμε τον πίνακα αυτόν πίνακα στροφής και του δίνουμε το σύμβολο ${}^A_B R$ αφού δηλώνει την αναπαράσταση του {B} ως προς {A}.[3]

$$ {}^A_B R = [{}^A\hat{X}_B \quad {}^A\hat{Y}_B \quad {}^A\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2.1.1.2) $$

[3] John J. Craig, "Εισαγωγή στη ρομποτική", 3η έκδοση, εκδόσεις Τζίολα 2009

Οι παρακάτω εξισώσεις χρησιμοποιούνται για την εύρεση του προσανατολισμού του πλαισίου ανάλογα τον άξονα περιστροφής ως προς έναν μονο άξονα:

- Περιστροφή γύρω απο τον άξονα του X με γωνία περιστροφής Θ_X:

$$ {}^{A}_{B}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_\chi) & -\sin(\theta_\chi) \\ 0 & \sin(\theta_\chi) & \cos(\theta_\chi) \end{bmatrix} \quad (2.1.1.3) $$

- Περιστροφή γύρω απο τον άξονα του Y με γωνία περιστροφής Θ_y:

$$ {}^{A}_{B}R = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} (2.1.1.4) $$

- Περιστροφή γύρω απο τον άξονα του Z με γωνία περιστροφής Θ_Z:

$$ {}^{A}_{B}R = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} (2.1.1.5) $$

Πλαίσιο:

Είναι απαραίτητο για τον ολοκληρομένο προσδιορισμό του ρομποτικού βραχίονα και ουσιαστικά είναι ένα σύστημα αναφοράς που περιέχει τέσσερα διανύσματα. Το ένα απο αυτά είναι το διάνυσμα θέσης και τα υπόλοιπα τρία προσδιορίζουν τον προσανατολισμό του σώματος. Οπότε το σύστημα {B} περιγράφεται χρησιμοποιώντας δύο μεταβλητές:

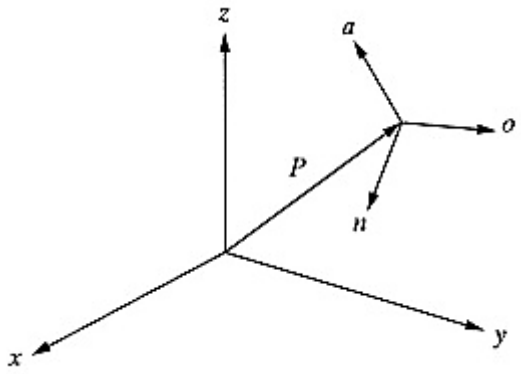
$$\{B\} = \left\{ {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R}, {}^{A}P_{BORG} \right\}$$

${}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R}$ = Προσανατολισμός πλαισίου

${}^{\mathrm{A}}\mathrm{P}_{\mathrm{BORG}}$ = διάνυσμα της αρχής των αξόνων του {B}

Οπότε έχουμε:

$${}^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R}\,{}^{\mathrm{B}}\mathrm{P}_{\mathrm{BORG}} \text{ (2.1.1.6)}$$



Σχήμα 2.1.1.1: Γραφική αναπαράσταση πλαισίου.

Πολλές φορές όμως γνωρίζουμε τις συντεταγμένες ενός διανύσματος σε σχέση με ένα άλλο πλαίσιο και όχι με την αρχή των αξόνων. Οπότε η απεικόνιση του νέου πλαισίου θα είναι:

$${}^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R}\,{}^{\mathrm{B}}\mathrm{P} \text{ (2.1.1.7)}$$

Στη περίπτωση που θέλουμε να ορίσουμε ένα πλαίσιο σε σχέση με ένα άλλο που έχουν ίδιο προσανατολισμό αλλα απέχουν απόσταση ${}^{\mathrm{A}}\mathrm{P}_{\mathrm{BORG}}$ τότε θα έχουμε:

$${}^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}\mathrm{P} + {}^{\mathrm{A}}\mathrm{P}_{\mathrm{BORG}} \text{ (2.1.1.8)}$$

Έστω οτι έχουμε ένα πλαίσιο B έχει υποστεί αλλαγή στον προσανατολισμό του ώς προς το πλαίσιο A και έχει μετακινηθεί κατα ${}^{A}P_{BORG}$. Τότε συνδιάζοντας τις δύο παραπάνω περιπτώσεις θα έχουμε τη σχέση:

$$^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R}\,{}^{\mathrm{B}}\mathrm{P} + {}^{\mathrm{A}}\mathrm{P}_{\mathrm{BORG}} \quad (2.1.1.9)$$

Η παραπάνω εξίσωση δεν είναι αρκετα εύχρηστη και για τον λόγο αυτό χρησιμοποιείται η παρακάτω εξίσωση που θεωρείται θεμελειώδης:

$$^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{T}\,{}^{\mathrm{B}}\mathrm{P} \quad (2.1.1.10)$$

Το ${}^{A}_{B}T$ λέγεται τελεστής μετασχηματισμού ή πινακας μετασχηματισμού και απεικονίζει την αλλαγή της θέσης και του προσανατολισμού τουB ώς προς A.

Η παραπάνω σχέση χρησιμοποιώντας πίνακες θα πάρει τη μορφή: [4]

$$\begin{bmatrix} {}^{\mathrm{A}}\mathrm{P} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{R} & {}^{\mathrm{A}}\mathrm{P}_{\mathrm{BORG}} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{B}}\mathrm{P} \\ 1 \end{bmatrix} \quad (2.1.1.11)$$

Με άλλα λόγια,

1) Προστίθεται «1», ως τελευταίο στοιχείο των διανυσμάτων 4x1.

2) Προστίθεται μια γραμμή «[0 0 0 1]» ως τελευταία γραμμή του πίνακα 4x4.

Έτσι όταν επιθυμούμε στη ρομποτική να ορίσουμε ένα πλαίσο σε σχέση με ένα άλλο τότε χρησιμοποιούμε την παραπάνω εξίσωση. Αν υπάρχουν περισσότερα απο ένα πλαίσια και αναζητούμε το τελικό πίνακα μετασχηματισμού, τότε αρκεί να πολλαπλασιάσουμε όλους τους πίνακες μετασχηματισμού. Για παράδειγμα αν έχουμε τρία πλαίσια A,B,C τότε:

$$^{\mathrm{A}}\mathrm{P} = {}^{\mathrm{A}}_{\mathrm{B}}\mathrm{T}\,{}^{\mathrm{B}}_{C}\mathrm{T}\,{}^{\mathrm{C}}\mathrm{P} \quad (2.1.1.12)$$

[4] John J. Craig, "Εισαγωγή στη ρομποτική", 3η έκδοση, εκδόσεις Τζίολα 2009

Παραμετροποίηση Στροφής:

Τις περισσότερες φορές η στροφή ενός πλαισίου δεν γίνεται μόνο σε έναν άξονα. Για αυτό το λόγο αξιοποιούμε τις εξισώσεις και τον ορισμό του Euler. Κατα τον Euler η μέθοδος αναπαράστασης πλαισίου {B} είναι η ακόλουθει: [5]

Θεωρούμε ένα πλαίσιο {B} που ταυτίζεται αρχικά με ένα γνωστό πλαίσιο {A}. Το στρέφουμε πρώτα κατα γωνία α ως προς $\hat{Z}_B$. Στη συνέχεια κατα β ως προς $\hat{Y}_B$ και τέλος κατα γωνία γ ως προς τον $\hat{X}_B$.

Οι στροφές που έχει υποστεί το σύστημα ονομάζονται **γωνίες στροφής Euler ως προς Z-Y-X (Euler angles Z-Y-X)** και ο τελικός προσανατολισμός δίνεται απο την παρακάτω εξίσωση:

$$ {}^{A}_{B}R = R_z(\Theta_z)R_y(\Theta_y)R_X(\Theta_\chi) = $$

$$ = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_\chi) & -\sin(\theta_\chi) \\ 0 & \sin(\theta_\chi) & \cos(\theta_\chi) \end{bmatrix} = $$

$$ \begin{bmatrix} \cos(\theta_z)\cos(\theta_y) & \cos(\theta_z)\sin(\theta_y)\sin(\theta_x) - \sin(\theta_z)\cos(\theta_x) & \cos(\theta_z)\sin(\theta_y)\sin(\theta_x) + \sin(\theta_z)\sin(\theta_x) \\ \sin(\theta_z)\cos(\theta_y) & \sin(\theta_z)\sin(\theta_y)\sin(\theta_x) + \cos(\theta_z)\cos(\theta_x) & \sin(\theta_z)\sin(\theta_y)\cos(\theta_x) - \cos(\theta_z)\sin(\theta_x) \\ -\sin(\theta_y) & \cos(\theta_y)\sin(\theta_x) & \cos(\theta_y)\cos(\theta_x) \end{bmatrix} $$

(2.1.1.13)

[5] John J. Craig, "Εισαγωγή στη ρομποτική", 3η έκδοση, εκδόσεις Τζίολα 2009

2.1.2 Ευθύγραμμη κινηματική

Μετά την κατανόηση των βασικών εννοιών, μπορούμε άνετα να επιλύσουμε την ευθύγραμμη κινηματική του βραχίονα. Υπολογίζονται οι θέσεις του κάθε μέλους του βραχίονα , ο προσανατολισμός τους και η κίνηση του τελικού στοιχείου δράσης. Στη κινηματική ενδιαφερόμαστε επίσης για την ταχύτητα και την επιτάχυνση του τελικού στοιχείου δράσης αλλα αυτό το κομμάτι θα αναλυθεί στο κεφάλαιο σχετικά με τη τροχιά του βραχίονα.

Αρχικά φτιάχνουμε ένα πίνακα με τα δεδομένα του εκάστοτε βραχίονα. Ο πίνακας αυτός έχει την παρακάτω μορφή:[6]

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
⋮				
n				

Πίνακας 2.1.2.1: Παράμετροι μελών του βραχίονα

Όπου:

a_i = η απόσταση μεταξύ των $\hat{Z}_i$ και $\hat{Z}_{i+1}$ μετρούμενη κατα μήκος του $\hat{X}_\iota$.

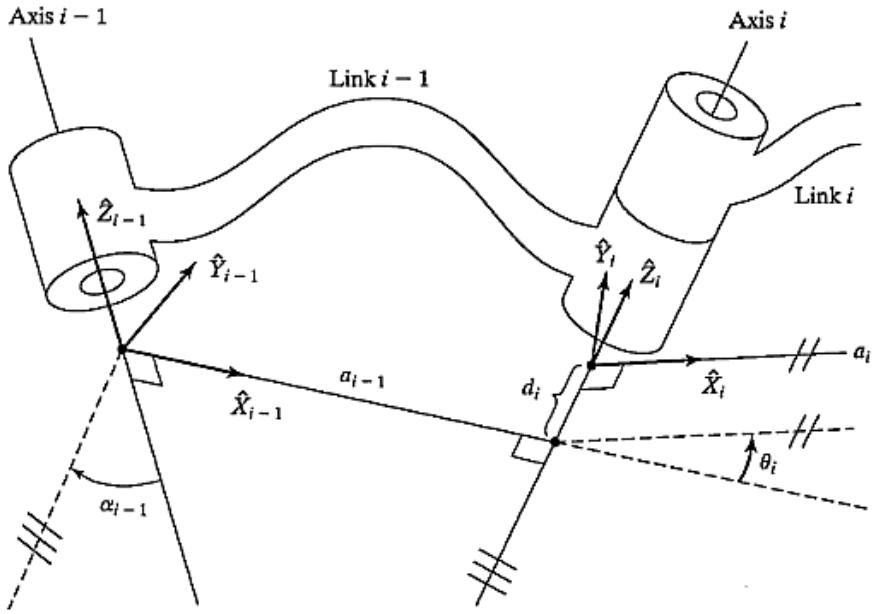
α_i = η γωνία μεταξύ των $\hat{Z}_i$ και $\hat{Z}_{i+1}$ μετρούμενη ως προς $\hat{X}_\iota$.

d_i = η γωνία μεταξύ των $\hat{X}_{i-1}$ και $\hat{X}_i$ μετρούμενη κατα μήκος του $\hat{Z}_\iota$.

θ_i = η γωνία μεταξύ των $\hat{X}_{i-1}$ και $\hat{X}_i$ μετρούμενη ως προς $\hat{Z}_\iota$.

[6] John J. Craig, "Εισαγωγή στη ρομποτική", 3η έκδοση, εκδόσεις Τζίολα 2009

Για την καλύτερη κατανόηση των παραμέντρων δίνεται η παρακάτω σχηματική απεικόνιση των πλαισίων στα μέλη του βραχίονα:



Σχήμα 2.1.2.1: Πλαίσια και μέλη του βραχίονα.

Στη συνέχεια αξιοποιώντας τον πίνακα φτιάχνεται ο πίνακας μετασχηματισμού του κάθε μέλους και στο τέλος πολλαπλασιάζοντας τους, δίνεται ο τελικός πίνακας μετασχηματισμού. Ο πίνακας αυτός περιλαμβάνει τα τελικό προσανατολισμό και θέση του τελικού στοιχείου δράσης σε σχέση με τις γωνίες κλήσεις του κάθε μέλους. Οπότε ξέροντας τις γωνίες κλήσης μπορούμε ανα πάσα στιγμή να βρούμε που βρίσκεται το τελικό στοιχείο δράσης στον τρισδιάστατο χώρο.

2.1.3 Αντίστροφη κινηματική

Στην ρομποτική όμως το βάρος πέφτει στην αντίστροφη κινηματική και όχι στην ευθύγραμμη. Αυτό συμβαίνει για το λόγο ότι μας ενδιαφέρει να δίνουμε τη θέση και το προσανατολισμό του σημείου στο οποίο θέλουμε να πάει το

τελικό στοιχείο δράσης και να υπολογίζονται οι κλίσεις του κάθε μέλους του βραχίονα. Το αντίθετο δηλαδή απο την ευθύγραμμη κινηματική.

Αρχικά πρέπει να βρεθεί ο χώρος εργασίας, δηλαδή να οριστούν τα όρια κλίσης του κάθε μέλους του βραχίονα για τη σωστή λειτουργία του και στη συνέχεια να βρεθούν όλοι οι συνδιασμοί λύσεων. Ένας βραχίονας έχει τη δυνατότητα να προσεγγίσει το τελικό σημείο με διαφορετικούς συνδιασμούς και συνήθως όσο περισσότερα μέλη υπάρχουν τοσο περισσότεροι είναι οι συνδιασμοί. Τέλος η σωστή επιλογή συνδιασμού ορίζεται απο τον μηχανικό και συνήθως επιλέγεται ο συνδιασμός που υλοποιείται με τη λιγότερη δυνατή κίνηση του βραχίονα.

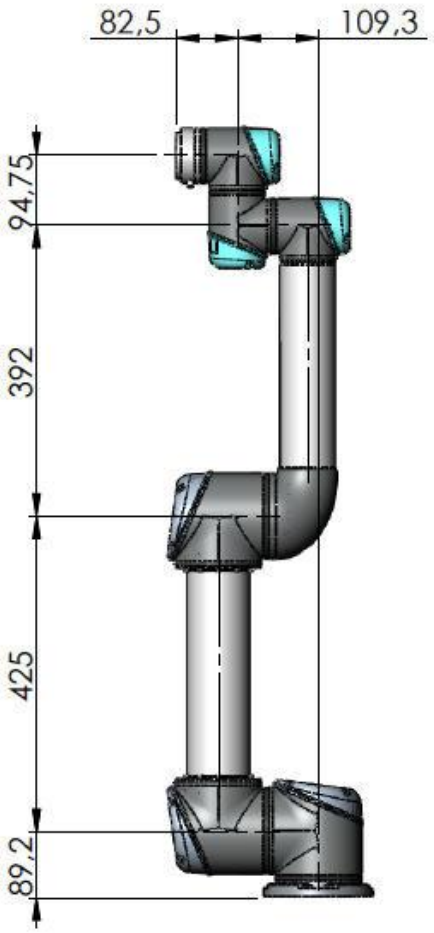
Για την επίλυση της αντίστροφης κινηματικής υπάρχουν δύο μέθοδοι:

- Αλγευρική Προσέγγιση
  Μετασχηματίζει τις εξισώσεις της ευθύγραμμης κινηματικής με τέτοιο τρόπο που να έχουμε σαν αγνώστους τις γωνίες των μελών του βραχίονα.

- Γεωμετρική Προσέγγιση
  Χωρίζει το βραχίονα σε μικρά γεωμετρικά σχήματα και προσπαθεί να βρεί εξισώσεις σχετικά με τις γωνίες των μελών του με τη βοήθεια τριγωνομετρίας και επιπεδομετρίας.

2.1.4 Κινηματική UR5

Για την εύρεση των εξισώσεων της αντίστροφης κινηματικής του βιομηχανικού βραχίονα UR5 αρχικά πρέπει να προσδιοριστεί ο πίνακας παραμέτρων.



Σχήμα 2.1.4.1: Διαστάστεις UR5.

Απο το παραπάνω σχήμα προκύπτει ο πίνακας παραμέτρων:

i	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
0	0	0	-	-
1	0	$\frac{\pi}{2}$	0.0892	θ_1
2	0.425	0	0	θ_2
3	0.392	0	0	θ_3
4	0	$\frac{\pi}{2}$	0.1093	θ_4
5	0	$-\frac{\pi}{2}$	0.09476	θ_5
6	-	-	0.0825	θ_6

Πίνακας 2.1.4.1: Πίνακας παραμέτρων

Στη συνέχεια βρέθηκαν με τη χρήση του παράνω πίνακα οι εξισώσεις ευθύγραμμης κινηματικής : [7]

$$p_x = d_5 c_1 s_{234} + d_4 s_1 - d_6 c_1 c_{234} + a_2 c_1 c_2 + d_6 c_5 s_1 + a_3 c_1 c_2 c_3 - a_3 c_1 s_2 s_3$$

(2.1.4.1)

$$\mathrm{p_y} = \mathrm{d_5 s_1 s_{234}} - \mathrm{d_4 c_1} - \mathrm{d_6 s_1 c_{234}} - \mathrm{d_6 c_1 c_5} + \mathrm{a_2 c_2 s_1} + \mathrm{a_3 c_2 c_3 s_1} - \mathrm{a_3 s_1 s_2 s_3}$$ (2.1.4.2)

$$\mathrm{p_z} = \mathrm{d_1} - \mathrm{d_6 s_{234} s_5} + \mathrm{a_3 s_{23}} + \mathrm{a_2 s_2} - \mathrm{d_5 c_{234}}$$ (2.1.4.3)

Όπου,

$$\mathrm{s_i} = \sin(\theta_\iota)$$
$$\mathrm{c_i} = \cos(\theta_\iota)$$

και

$$\mathrm{s_{234}} = \sin(\theta_2 + \theta_3 + \theta_4)$$
$$\mathrm{c_{234}} = \cos(\theta_2 + \theta_3 + \theta_4)$$

Όσον αφορά τις εξισώσεις της αντίστροφή κινηματική, χρησιμοποποιήθηκε η γεωμετρική προσέγγιση. Έτσι οι εξισώσεις των έξι γωνιών για το UR5 θα είναι:

Γωνία θ_1:

$$\theta_1 = \psi + \varphi + \frac{\Pi}{2} \qquad (2.1.4.5)$$

Με

$$\psi = \mathrm{atan2}((\mathrm{P_5^0})_\mathrm{y}, (\mathrm{P_5^0})_\mathrm{x})$$

[7] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, “Kinematic and Dynamic Modelling of UR5 Manipulator”, IEEE, 2016

$$\varphi = \pm\arccos\left(\frac{d_4}{\sqrt{(P_5^0)_x{}^2 + (P_5^0)_y{}^2}}\right)$$

και

$$P_5^0 = T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Όπου ο πίνακας μετασχηματισμού T_6^0 βρίσκεται υλοποιόντας τις εξισώσεις Euler.Τέλος υπάρχουν δύο πιθανές λύσεις για αριστερά - δεξιά.

Γωνία θ_5:

$$\theta_5 = \pm\arccos\left(\frac{(P_6^1)_z - d_4}{d_6}\right) \quad (2.1.4.6)$$

Με

$$(P_6^1)_z = (P_6^0)_x \sin(\theta_1) - (P_6^0)_y \cos(\theta_1)$$

Για να είναι σωστή η λυση της εξίσωσης θα πρέπει να ισχύει $\frac{(P_6^1)_z - d_4}{d_6} \leq 1$. Οι λύσεις τις εξίσωσης είναι ξανά δύο και αυτη τη φορά είναι για το πάνω - κάτω.

Γωνία θ_6:

$$\theta_6 = atan2\left(\frac{-Z_y}{sin(\theta_5)}, \frac{Z_x}{sin(\theta_5)}\right) (2.1.4.7)$$

Γωνία θ_3:

$$\theta_3 = \pm\arccos\left(\frac{\left\|\overrightarrow{P_3^1}\right\|^2 - a_2^2 - a_3^2}{2a_2a_3}\right) (2.1.4.8)$$

Όπου $\left\|\overrightarrow{P_3^1}\right\|$ το μέτρο του διανύσματος $\overrightarrow{P_3^1}$:

$$\overrightarrow{P_3^1} = T_4^1 \begin{bmatrix} 0 \\ -d_4 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Με

$$T_4^1 = T_6^1 (T_5^4 T_6^5)^{-1}$$

Γωνία θ_2:

$$\theta_2 = -atan2\left((P_3^1)_y, (P_3^1)_x\right) + \arcsin\left(\frac{a_3 \sin(\theta_3)}{\left\|\overrightarrow{P_3^1}\right\|}\right) \quad (2.1.4.9)$$

Γωνία θ_4:

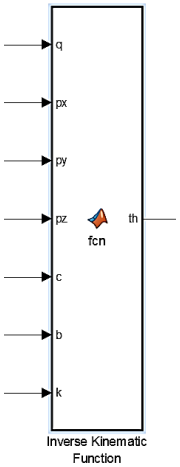
$$\theta_4 = \text{atan2}\left(x_x, x_y\right) \quad (2.1.4.10)$$

x_x και x_y είναι στοιχεία απο τον πίνακα T_4^3 :

$$T_4^3 = (T_2^1 T_4^3)^{-1} T_4^1$$

Πρόγραμμα Matlab:

Αξιοποιώντας τις παραπάνω εξισώσεις φτιάχτηκε στο Simulink του Matlab ένα πρόγραμμα προκειμένου δίνοντας τις επιθυμές συντεταγμένες θέσης και προσανατολισμού να τις μετατρέπει σε γωνίες κλίσης του κάθε μέλους ξεχωριστά.



Σχήμα 2.1.4.2 : Simulink για το το πρόβήμα της αντίστροφης κινηματικής

Στο Simulink το εικονίδιο function χρησιμοποιείται για να μπορεί ο χρήστης να γράψει κώδικα μεσα στο πρόγραμμα και όπως βλεπούμε στην παραπάνω εικόνα αριστερά του έχουμε τις εισόδους και δεξιά τις εξόδους του συστήματος. Σαν είσοδο έχουμε:

- τα px,py,px που είναι οι επιθυμητές συντεταγμένες θέσης που θέλουμε να πάει το τελικό στοιχείο δράσης.
- τα c,b,k που προσδιορίζουν τον επιθυμητό προσανατολισμό.
- το q που είναι ένας πίνακας 6x1 και περιέχει τις τωρινές συντεταγμένες θέσης και προσανατολισμού του τελικού στοιχείου δράσης.

Το πρόγραμμα βρίσκει τις σωστούς συνδιασμούς και συγκρίνοντας τις τωρινές με τις επιθυμητές θέσεις, επιλέγει τον κατάλληλο συνδιασμό προκειμένου να έχουμε την μικρότερη κίνηση στον βραχίονα. Τέλος έχουμε μία μόνο έξοδο που είναι ένας πίνακας 6x1 με τις σωστές γωνίες κλίσης του κάθε μέλους.

2.2 Δημιουργία οδεύσεων

Με την λέξη τροχιά ή όδευση στη ρομποτική εννοούμε την επιθυμητή κίνηση του βραχίονα στον τρισδιάστατο χώρο. Η τροχιά αυτή μπορεί να περιλαμβάνει αρχικό και τελικό σημείο ή ακόμα και ενδιάμεσα σημεία απο τα οποία πρέπει να περάσει το τελικό στοιχείο δράσης του βραχίονα. Αρχικά υπολογίζονται οι γωνίες κλίσης του κάθε μέλους και στη συνέχεια η μετάβαση τους απο την αρχική μέχρι την τελική τους θέση πρέπει να γίνει στο ίδιο χρονικό διάστημα.

Στην πρώτη περίπτωση που θα μελετήσουμε, θεωρούνται γνωστά η αρχική και τελική θέση του βραχίονα (θ_0 και θ_τ αντίστοιχα), η αρχική ταχύτητα του (u_0) και ο χρόνος ολοκλήρωσης (T_τ). Παρακάτω φαίνονται οι εξισώσεις για την εύρεση θέσης θ(t), της ταχύτητας u(t) και της επιτάχυνσης a(t) :

$$\theta(t) = \theta_0 + u_0 t + a_1 t^2 + a_2 t^3 \quad (2.2.1)$$

$$u(t) = u_0 + 2a_1 t + 3a_2 t^2 \quad (2.2.2)$$

$$a(t) = 2a_1 + 6a_2 t \text{ (2.2.3)}$$

Όπου,

$$a_1 = \frac{3}{{T_T}^2}(\theta_T - \Theta_0)$$

$$a_2 = -\frac{2}{{T_T}^3}(\theta_T - \Theta_0)$$

Πολλές φορές όμως επιθυμούμε το τελικό στοιχείο δράσης να περνά απο ενδιάμεσα σημεία χωρίς να σταματάει ή με κάποια συγκεκριμένη ταχύτητα. Στη περίπτωση αυτή η τροχιά χωρίζεται σε επιμέρους κομμάτια και εφαρμόζονται οι εξισώσεις που δίνονται παρακάτω σε κάθε κομμάτι ξεχωριστά.

$$\theta(t) = \theta_0 + u_0 t + a_1 t^2 + a_2 t^3 \text{ (2.2.4)}$$

$$a_1 = \frac{3}{{T_T}^2}(\theta_T - \Theta_0) - \frac{2}{T_T}u_0 - \frac{1}{T_T}u_\tau \text{ (2.2.5)}$$

$$a_2 = -\frac{2}{{T_T}^3}(\theta_T - \Theta_0) + \frac{1}{{T_T}^2}(u_0 + u_\tau) \text{ (2.2.6)}$$

Με u_τ η ταχύτητα που θέλουμε να έχει το μέλος του βραχίονα όταν περνάει απο το συγκεκριμένο σημείο.

Ένας άλλος τρόπος δημιουργίας της τροχιάς των μελών του βραχίονα είναι να θεωρήσουμε μία γραμμική συνάρτηση με εξομαλύνσεις στα άκρα της χρησιμοποιώντας παραβολικές μίξεις. Αυτό μας βοηθάει να μην έχουμε μεγάλες ταχύτητες που ξεπερνούν τα ανώτερα όρια των κινητήρων του βραχίονα. Υποθέτουμε οτι οι παραβολικές μίξεις έχουν την ίδια χρονική διάρκεια και οτι η ταχύτητα στο τελος τους ισούται με την ταχύτητα γραμμικής περιοχής. Οπότε οδηγούμαστε στη παρακάτω σχέση:

$$at_\Pi = \frac{\theta_M - \Theta_\Pi}{T_M - T_\Pi} \text{ (2.2.7)}$$

Όπου,

Θ_Π = η τιμή που έχει η Θ στο τέλος της περιοχής μίξης.

a = η επιτάχυνση στη περιοχή μίξης.

Ισχύει:

$$\Theta_{\Pi} = \Theta_0 + \frac{1}{2}\mathrm{a}t_{\Pi}{}^2 \text{ (2.2.8)}$$

Συνδιάζοντας τις παραπάνω εξισώσεις θα έχουμε:

$$\mathrm{a}t_{\Pi}{}^2 - \mathrm{a}t_{\Pi}\mathrm{t} + (\theta_{\mathrm{T}} - \Theta_0) = 0 \text{ (2.2.9)}$$

Με t ο συνολικός χρόνος κίνησης. Λύνοντας ως προς t_{Π} έχουμε:

$$t_{\Pi} = \frac{\mathrm{t}}{2} - \frac{\sqrt{\mathrm{a}^2\mathrm{t}^2 - 4\mathrm{a}(\theta_{\mathrm{T}} - \Theta_0)}}{2\mathrm{a}} \text{ (2.2.10)}$$

Οπότε γνωρίζοντας την αρχική και τελική τιμή κλίσης , την επιθυμητή επιτάχυνση στο τέλος της περιοχής μίξης και τον συνολικό χρόνο της κίνησης μπορείς να βρείς το χρόνο της περιοχής μίξης. Για να ισχύει η παραπάνω εξίσωση θα πρέπει να ισχύει:

$$a = \frac{4(\theta_{\mathrm{T}} - \Theta_0)}{t^2} \text{ (2.2.11)}$$

Στην περίπτωση που έχουμε ενδιάμεσα σημεία τότε ο τρόπος εύρεσης της τροχιάς με παρεμβολές αλλάζει και υπάρχουν διαφορετικές εξισώσεις για τα ενδιάμεσα σημεία και διαφορετικές για το πρώτο και το τελευταίο κομμάτι. Αυτή η μέθοδος έχει σαν προϋποθεση ο βραχίονας να μη σταματά στα ενδίαμεσα σημεία. Παρακάτω φαίνονται οι αντίστοιχες εξισώσεις:[8]
Εξισώσεις ενδιάμεσων σημείων:

$$\dot{\theta}_{\mathrm{jk}} = \frac{\theta_{\mathrm{k}} - \theta_{\mathrm{j}}}{t_{\mathrm{djk}}} \text{ (2.2.12)}$$

$$\ddot{\theta}_{\mathrm{k}} = \mathrm{SGN}\left(\dot{\theta}_{\mathrm{kl}} - \dot{\theta}_{\mathrm{jk}}\right)\left|\ddot{\theta}_{\mathrm{k}}\right| \text{ (2.2.13)}$$

[8] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, “Kinematic and Dynamic Modelling of UR5 Manipulator”, IEEE, 2016

$$t_k = \frac{\dot{\theta}_{kl} - \dot{\theta}_{jk}}{\ddot{\theta}_k} \quad (2.2.14)$$

$$t_{jk} = t_{djk} - \frac{1}{2}\, t_j - \frac{1}{2} t_k \quad (2.2.15)$$

Όπου:

t_{jk} = Η διάρκεια που αντιστοιχεί στο ευθύγραμμο τμήμα της διαδρομής μεταξύ δύο συνεχόμενων σημείων j και k.

t_{djk} = Η συνολική διάρκεια που αντιστοιχεί στη περιοχή μίξης μεταξύ δύο συνεχόμενων σημείων j και k.

t_j = Η διάρκεια της περιοχής μίξης γύρω απο το j.

t_k = Η διάρκεια της περιοχής μίξης γύρω απο το k.

$\dot{\theta}_{jk}$ = Η ταχύτητα στη γραμμική περιοχή μεταξύ των δύο συνεχόμενων σημείων j και k.

$\dot{\theta}_{kl}$ = Η ταχύτητα στη γραμμική περιοχή μεταξύ των επόμενων δύο συνεχόμενων σημείων k και l.

$\ddot{\theta}_k$ = Η επιτάχυνση κατα την μίξη κοντά στο j.

Εξισώσεις πρώτου τμήματος:

$$\frac{\theta_2 - \theta_1}{t_{12} - \frac{1}{2} t_1} = \ddot{\theta}_1 t_1 \quad (2.2.16)$$

$$\ddot{\theta}_1 = \mathrm{SGN}(\theta_2 - \theta_1)\left|\ddot{\theta}_1\right| \quad (2.2.17)$$

$$t_1 = t_{d12} - \sqrt{{t_{d12}}^2 - \frac{2(\theta_2 - \theta_1)}{\ddot{\theta}_1}} \quad (2.2.18)$$

$$\dot{\theta}_{12} = \frac{\theta_2 - \theta_1}{t_{d12} - \frac{1}{2} t_1} \quad (2.2.19)$$

$$t_{12} = t_{d12} - t_1 - \frac{1}{2} t_2 \quad (2.2.20)$$

Εξισώσεις τελευταίου τμήματος:

$$\frac{\theta_{n-1}-\theta_n}{t_{d(n-1)n}-\frac{1}{2}t_n} = \ddot{\theta}_n t_n \text{ (2.2.21)}$$

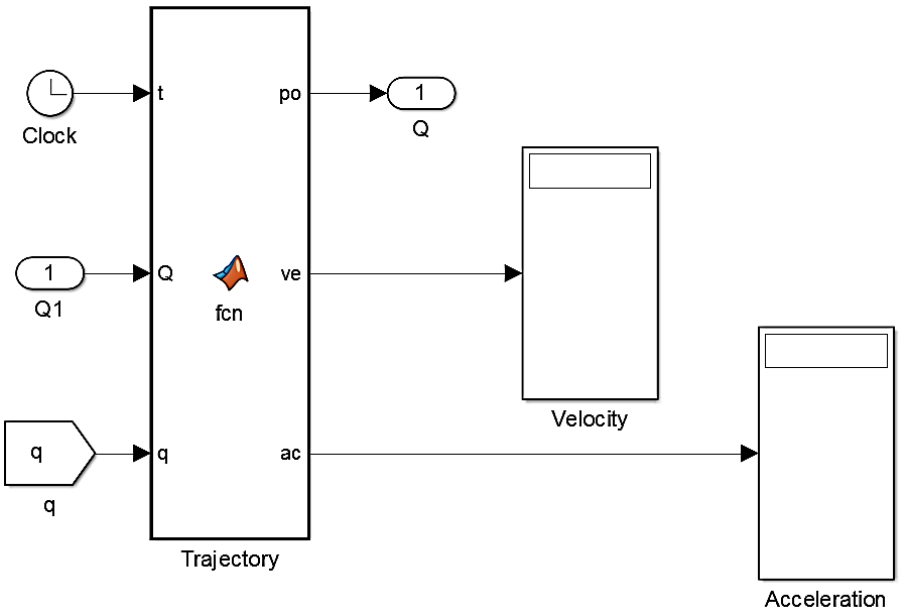
$$\ddot{\theta}_n = \mathrm{SGN}(\theta_{n-1}-\theta_n)\left|\ddot{\theta}_n\right| \text{ (2.2.22)}$$

$$t_n = t_{d(n-1)n} - \sqrt{{t_{d(n-1)n}}^2 - \frac{2(\theta_{n-1}-\theta_n)}{\ddot{\theta}_n}} \text{ (2.2.23)}$$

$$\dot{\theta}_{(n-1)n} = \frac{\theta_{n-1}-\theta_n}{t_{d(n-1)n}-\frac{1}{2}t_n} \text{ (2.2.24)}$$

$$t_{(n-1)n} = t_{d(n-1)n} - t_n - \frac{1}{2}t_{n-1} \text{ (2.2.25)}$$

Για την δημιουργία της διαδρομής του βραχίονα UR5 σε πραγματικό χρόνο, πρέπει αρχικά να μελετηθούν τα δεδομένα και ο λόγος υλοποίησης της τροχιάς αυτής. Στη συνέχεια επιλέγονται οι σωστές εξισώσεις και δημιουργείται ο κατάλληλος κώδικας. Για το λόγο αυτό στη συνέχεια θα εστιάσουμε στις εισόδους και εξόδους του συστήματος και όχι στο κώδικα της τροχιάς του προγράμματος. Παρακάτω φαίνεται το κομμάτι του Simulink στο οποίο υπολογίζονται οι κατάλληλες θέσεις των μελών του βραχίονα σε πραγματικό χρόνο.



Σχήμα 2.2.1: Simulink για τη δημιουργία τροχιάς

Αριστερά του συστήματος βλέπουμε στη κορυφή το εικονίδο του ρολογιού, το οποίο δίνει στο σύστημα τον πραγματικό χρόνο. Απο κάτω του έχουμε σαν είσοδο του συστήματος έναν πίνακα 6x1 με ονομασία “Q1” που κάθε στοιχείο του παρουσιάζει την επιθυμητή γωνία του κάθε μέλους. Ο πίνακας αυτός είναι αποτέλεσμα της αντίστροφης κινηματικής που αναλύθηκε στο προηγούμενο κεφάλαιο. Κάτω αριστερά βλέπουμε έναν ακόμα πίνακα 6x1 με ονομασία “q “ που τα στοιχεία του αποτελούν τις τωρινές κλίσεις των μελων του βραχίονα. Δεξιά φαίνονται οι εξόδοι του συστήματος και συγκεκριμένα δύο πίνακας για την ταχύτητα και την επιτάχυνση του κάθε μέλους για πειραματικούς λόγους και ένας πίνακας 6x1 ονόματι Q που περιέχει την γωνία κλίσης του κάθε μέλους την αντίστοιχη χρονική στιγμή.

2.3 Δυναμική και έλεγχος του βραχίονα

Όταν αναφερόμαστε στη δυναμική του βραχίονα εννοούμε τον υπολογισμό των δυνάμεων και των ροπών που ασκούνται σε κάθε μέλους του βραχίονα προκειμένου αυτο να κινηθεί με τον κατάλληλο τρόπο, να διαγράψει την

επιθυμητή τροχιά και να φτάσει στο επιθυμητό σημείο. Στη περίπτωση που θέλουμε να βρούμε τις δυνάμεις αυτές στη μοντελοποίση του βραχίονα στο προγραμμα του Matlab, υπάρχουν δύο πιθανές λύσεις. Η μία λύση περιλαμβάνει τον αυτόματο υπολογισμό των ροπών μέσω του προγράμματος, δίνοντας σαν είσοδο της κάθε άρθρωσης την επιθυμητή γωνία κλίσης και η δεύτερη λύση είναι πιο περίπλοκή και χρειάζεται να δημιουργηθεί σύστημα ελέγχου για τον υπολογισμό των ροπών. Στη συνέχεια θα εξηγήσουμε πως λειτουργεί το σύστημα αυτό.

Η δύναμη ενός στερεού σώματος υπακούει στην παρακάτω σχέση:[9]

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) \quad (2.3.1)$$

όπου,

$M(\theta)$ = πίνακας αδράνειας του βραχίονα (nxn)

$G(\theta)$ = διάνυσμα βαρυτικής επίδρασης (nx1)

$V(\theta, \dot{\theta}) =$ διάνυσμα που περιλαμβάνει τις φυγόκεντρες και coriolis δυνάμεις (nx1)

Οι δυνάμεις αυτές όπως βλέπουμε είναι συναρτήση της ταχύτητας και της θέσης των αρθρώσεων του βραχίονα. Για να μπορέσουμε να διαχειριστούμε την παραπάνω εξίσωση τότε θα πρέπει να χρησιμοποιήσουμε ελεγκή και η εξίσωση θα είναι της μορφής:

$$\tau = \alpha\tau' + \beta \quad (2.3.2)$$

Με:

τ = το διάνυσμα ροπών στις αρθρώσεις του βραχίονα (nx1)

$\alpha = M(\theta)$

$\beta = V(\theta, \dot{\theta}) + G(\theta) + F(\theta, \dot{\theta})$

[9] John J. Craig, "Εισαγωγή στη ρομποτική", 3η έκδοση, εκδόσεις Τζίολα 2009

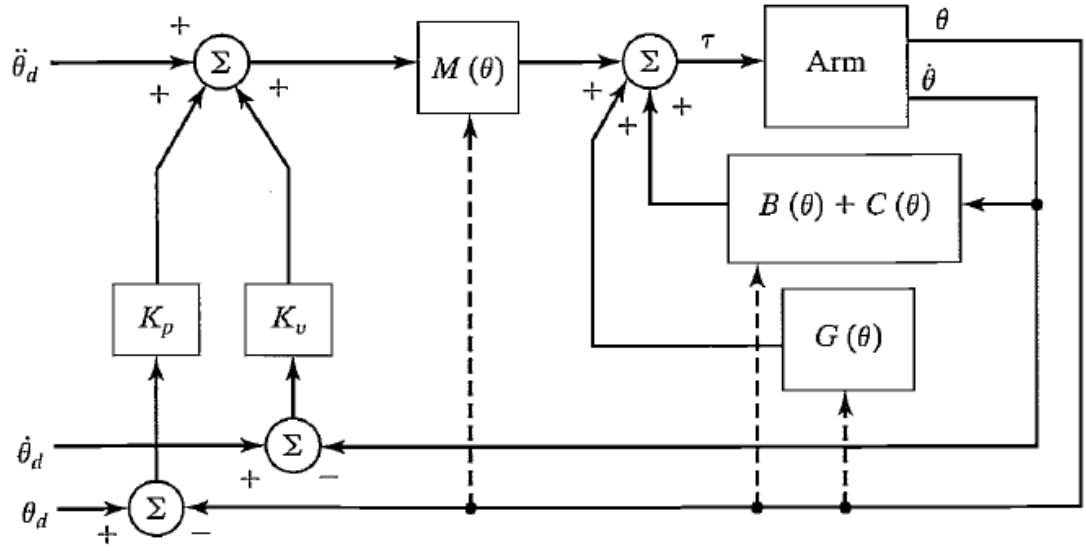
Επίσης στη περίπτωση του ελεγκτή χρειαζόμαστε ένα τμήμα ανατροφοδότησης όπου:

$$\tau' = \ddot{\theta}_d + K_u\dot{E} + K_pE \text{ (2.3.3)}$$

Και

$$E = \theta_d - \theta \text{ (2.3.4)}$$

Για να καταλάβουμε καλύτερα το σύστημα ελέγχου για έναν οποιονδήποτε ρομποτικό βραχίονα, δίνεται παρακάτω η σχηματική απεικόνιση:



Σχήμα 2.3.1: Σύστημα ελέγχου βραχίονα.

Με εξίσωση κλειστού βρόχου:

$$\ddot{E} + K_u\dot{E} + K_pE = 0 \text{ (2.3.5)}$$

Οπως βλέπουμε για να επιτύχουμε τον έλεγχο των μελών του βραχίονα θα πρέπει να χρησιμοποιήσουμε PD ελεγκτή και αναλυτικότερα δύο διαγώνιους πίνακες nxn. Στη δική μας περίπτωση οι πίνακες αυτοι θα ήταν 6x6 και θα πρέπει να ικανοποιούν την παραπάνω εξίσωση.

Για να μπορέσουμε να βρούμε τους πίνακες αδράνειας «M», βαρύτητας «G» και φυγόκεντρου-coriolis «V», θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα και να υπολογίσουμε:

1. Τον πίνακα προσανατολισμού για κάθε μέλος ξεχωριστά.
2. Ο πίνακας θέσεως για κάθε μέλος ξεχωριστά.
3. Τον πίνακα μετασχηματισμού για το κάθε μέλος ξεχωριστά και τον τελικό πίνακα μετασχηματισμού χρησιμοποιώντας τους παραπάνω πίνακες.
4. Ο πίνακας θέσεως και προσανατολισμού για κάθε μέλος απο τη βάση του βραχίονα έως το τελικό στοιχείο δράσης.
5. Τα γραμμικά και γωνιακά κομμάτια των Ιακοβιανών πινάκων.
6. Τον πίνακα αδράνειας του βραχίονα «M».

$$M(q) = [\sum_{i=1}^{n}(m_i u_{ui}^T J_{ui} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i})]^{10} \text{ (2.3.6)}$$

Όπου, J_{ui} και $J_{\omega i}$ είναι το γραμμικό και καρτεσιανό μέρος του Ιακοβιανού πίνακα J_i αντίστοιχα.

7. Τους πίνακες για τις φυγόκεντρες και Coriolis δυνάμεις «V».

$$V_{ij} = \sum_{k=1}^{n} \frac{1}{2}\left(\frac{\theta m_{ij}}{\theta q_k} + \frac{\theta m_{ik}}{\theta q_j} + \frac{\theta m_{kj}}{\theta q_i}\right)\dot{q}_k \text{ [11] (2.3.7)}$$

8. Τους επιμέρους και τον τελικό πίνακα για τις δυνάμεις που οφείλονται στην βαρύτητα των μελών «G».

$$g_i\left(\underline{q}\right) = \frac{\Theta P}{\theta q_i} \text{ [12] (2.3.8)}$$

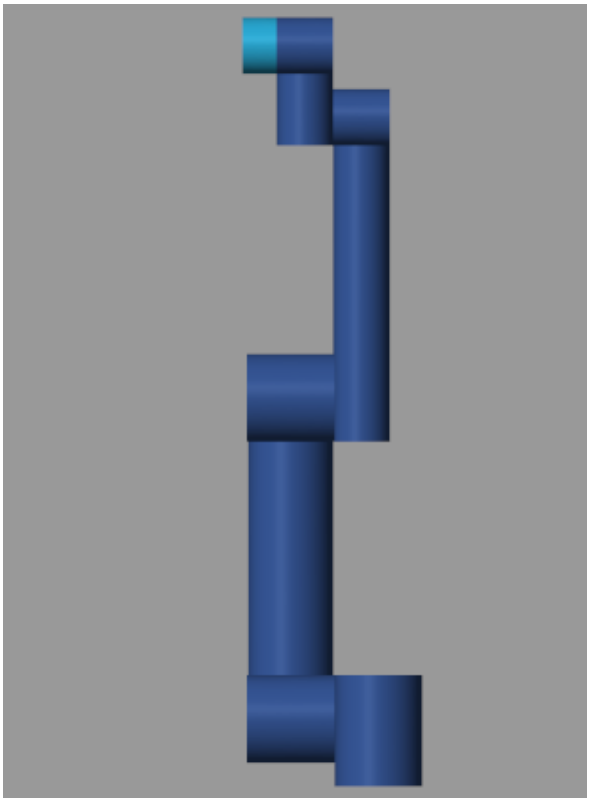
Για τον υπολογισμό τον πρώτον τεσσάρων βημάτων πρέπει να γνωρίζουμε τον πίνακα παραμέτρων απο την ενώτητα της κινηματικής του βραχίονα και

[10] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, "Kinematic and Dynamic Modelling of UR5 Manipulator", IEEE, 2016
[11] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, "Kinematic and Dynamic Modelling of UR5 Manipulator", IEEE, 2016
[12] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, "Kinematic and Dynamic Modelling of UR5 Manipulator", IEEE, 2016

να έχουμε γνώσεις σχετικά με τη θέση και τον προσανατολισμό πλαισίων , αλλά και σχετικά με τους πίνακες μετασχηματισμού.

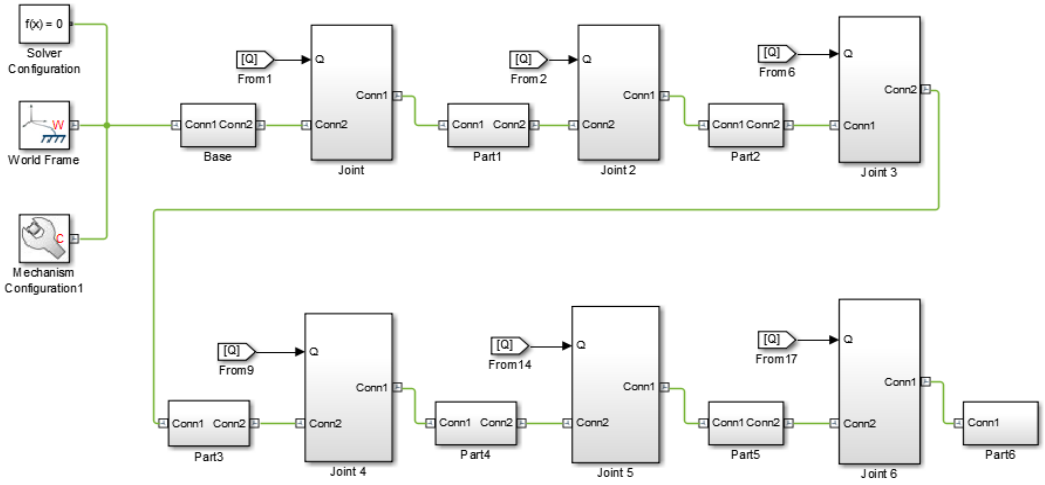
2.4 Μοντέλο του UR5

Τελευταίο κομμάτι για τη μοντεοποίηση του βραχίονα αποτελεί ο σχεδιασμός του μοντέλου του UR5. Πάνω σε αυτό θα ασκηθούν οι δυνάμεις που υπολογίσταν σύμφωνα με τη δυναμική του βραχίονα και θα δούμε την κίνηση του γραφικά στο Matlab. Για το μοντέλο του συγκεκριμένου βραχίονα χρησιμοποιήθηκαν τα δεδομένα του πίνακα απο την ευθύγραμμη κινηματική του και η απεικόνιση του στο πρόγραμμα του matlab φαίνεται παρακάτω:



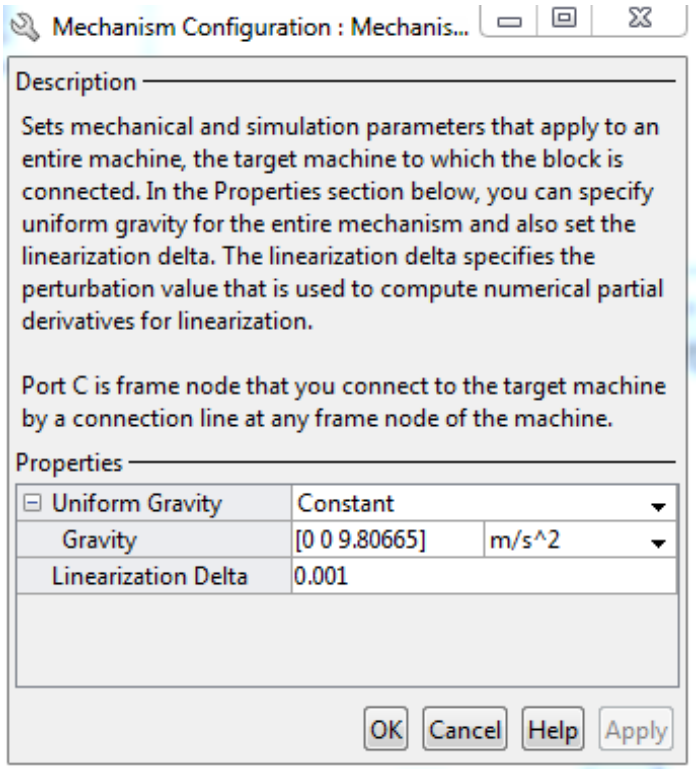
Σχήμα 2.4.1 : Μοντέλο του UR5 στο Matlab.

Δυστυχώς για την επίτευξη του χρειάστηκε να φτιάξουμε το κάθε κομμάτι του ξεχωριστά στο Simulink και εν συνεχεία να ενωθούν. Το πρόγραμμα του Simulink για την συγκεκριμένη απεικόνιση φαίνεται στη συνέχεια:



Σχήμα 2.4.2: Μοντέλο του UR5 στο Simulink.

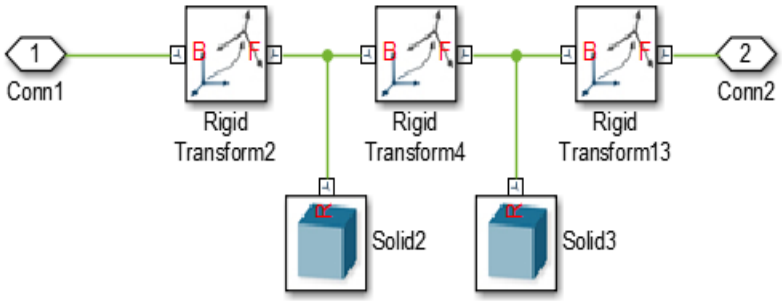
Συγκεκριμένα για το μοντέλο του UR5 στο Simulink χρησιμοποιήθηκαν αρκετα διαφορετικά εικονίδια. Αριστερά βλέπουμε τρία εικονίδια ονόματι Mechanism Configuration, Word Frame και Solver Configuration, τα οποία αποτελούν τη βάση κάθε ρομποτικού βραχίονα. Το πρώτο περιλαμβάνει τα δεδομένα για τη ασκούμενη βαρύτητα και ο πίνακας ρυθμίσεων του φαίνεται παρακάτω:



Εικόνα 2.4.1 Ρυθμίσεις βαρύτητας.

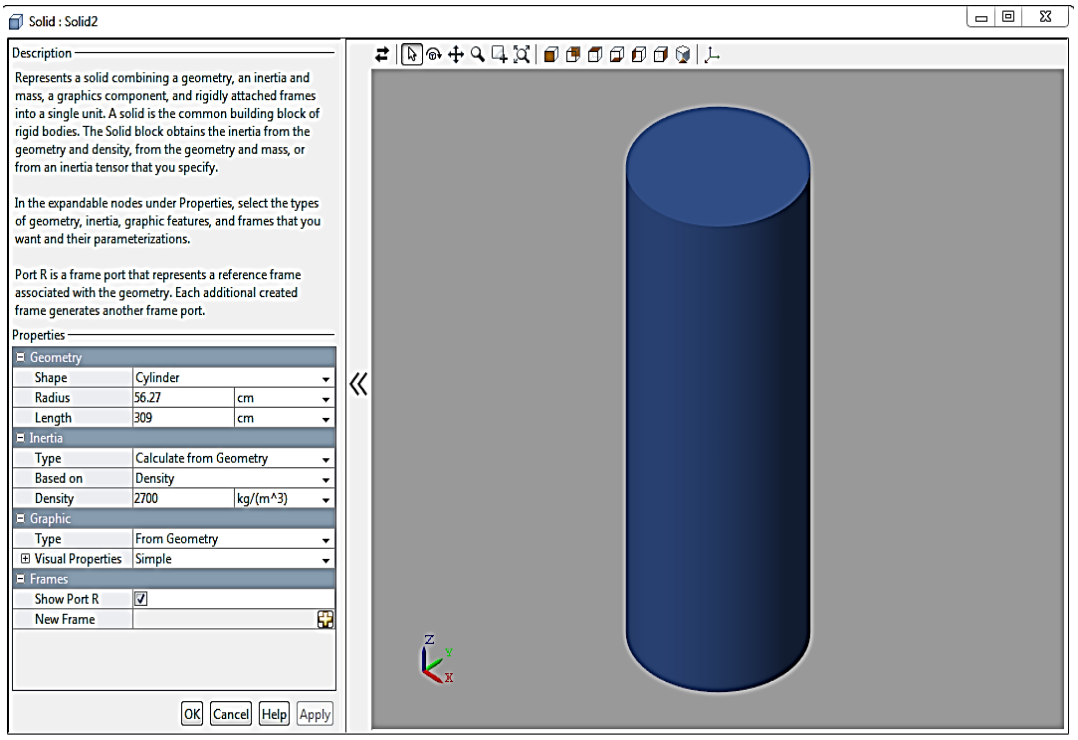
Μπορείς να ορίσεις αν η βαρύτητα θα είναι σταθερή, μεταβαλλόμενη , απο ποιά πλευρά ασκείται , η δύναμη της και διάφορα άλλα. Το δεύτερο δηλώνει τη βάση του βραχίονα, δηλαδή την αρχή των αξόνων μας και το τρίτο περιλαμβάνει τις ρυθμίσεις των παραμέτρων της αριθμητικής επίλυσης του βραχίονα.

Στη συνέχεια όπως βλέπουμε επαναλλαμβάνονται τα εικονίδια με ονομασία part και joint. Part είναι ένας συνδιασμός εικονιδίων που περιλαμβάνει κάποια απο τα μέλη του βραχίονα και αναλύει πως συνδέονται μεταξύ τους. Παρακάτω βλέπουμε την συνδεσμολογία του part2:



Σχήμα 2.4.3 Συνδέσεις σωμάτων

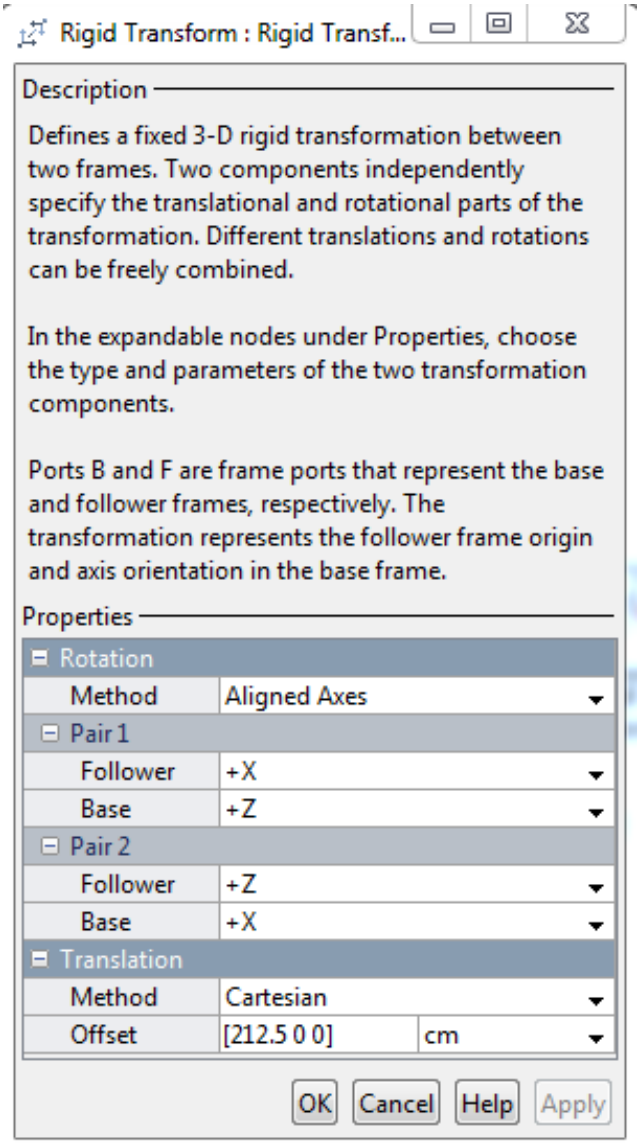
Τα solid 2 και 3 κοιτώντας το σχήμα του UR5 απο κάτω προς τα πάνω αντιστοιχούν στου κυλίνδρους 3 και 4. Για να δημιουργήσουμε του συγκεκριμένους κυλίνδρους, πηγαίνουμε στις ρυθμίσεις τους, επιλέγουμε να έχουν κυλινδρική μορφή και τις διαστάσεις τους. Στη συνέχεια βλεπουμε τις ρυθμίσεις του Solid 2:



Εικόνα 2.4.2: Ρυθμίσεις σωμάτων

Οπως βλέπουμε δεξία είναι η σχηματική απεικόνιση του αντικειμένου μας και δεξιά τοι ρυθμίσεις του. Μέσω των ρυθμίσεων αυτών μπορούμε να ορίσουμε τη μορφή του σχήματος μας και τις αντίστοιχες παραμέτρους του όπως μήκος, πλάτος και ακτίνα.

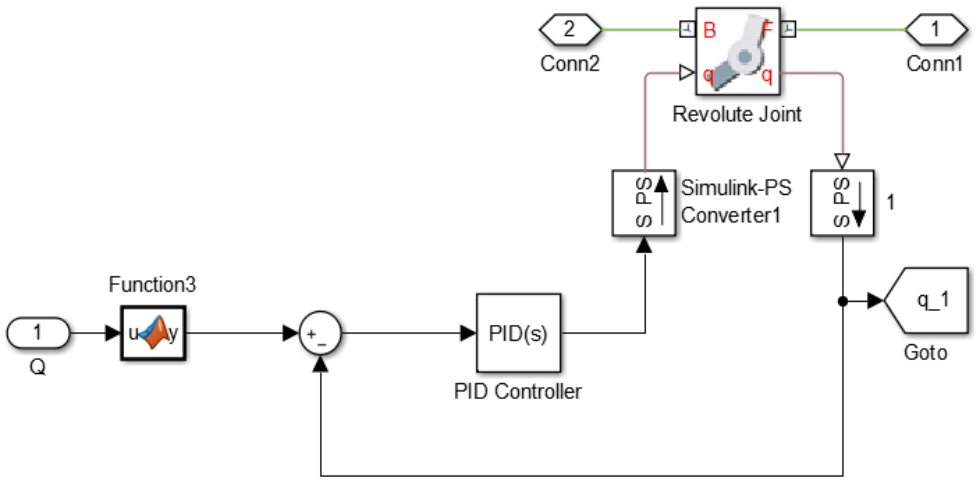
Μόλις δημιουργήσουμε τα αντικείμενα που μας χρειάζονται, το επόμενο βήμα είναι η σύνδεση τους. Σε αυτό μας βοηθάει η χρήση του Rigid transform, του οποίου οι ρυθμίσεις φαίνονται στη συνέχεια.



Εικόνα 2.4.3: Ρυθμίσεις συντεταγμένων σώματος.

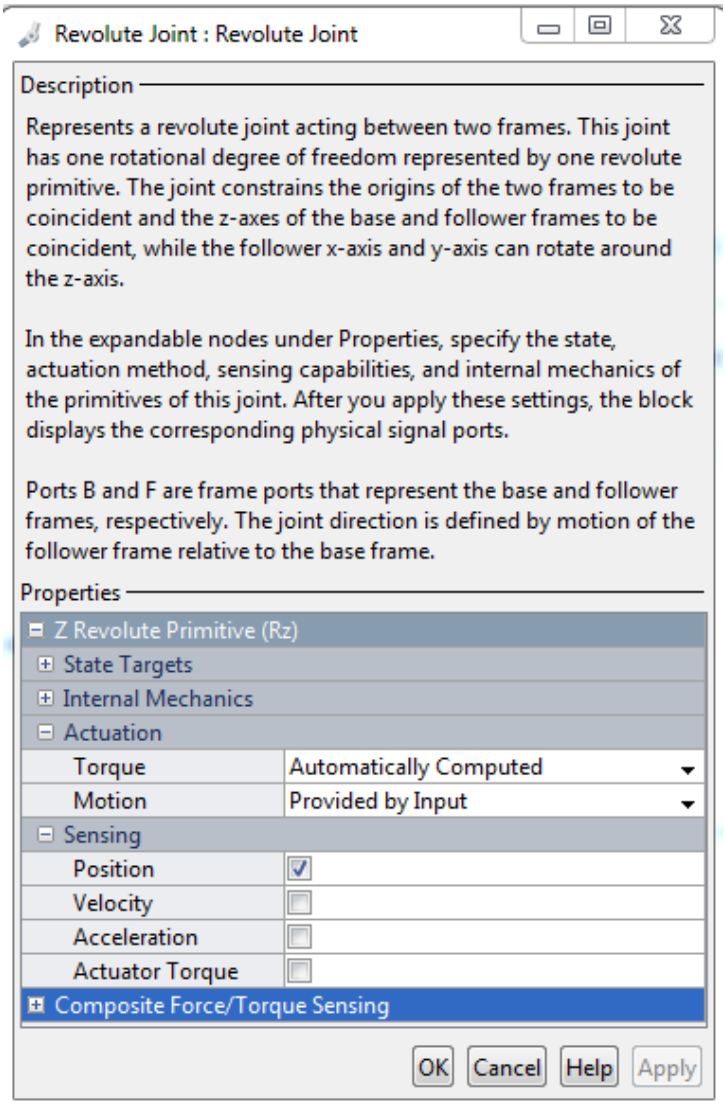
Στο σημείο αυτό προσδιορίζουμε τη μετατόπιση και την αλλαγή του προσανατολισμού ενός σχήματος σε σχέση με κάποιο άλλο. Στο παραπάνω σχήμα φαίνεται πώς γίνεται η ένωση του solid 2 με το προηγούμενο κομμάτι (part1).

Το τελευταίο που πρέπει να αναλυθεί είναι το εικονίδιο joint και πρόκειται για τις αρθρώσεις του βραχίονα. Στη συνέχεια βλέπουμε την πρώτη άρθρωση του:



Σχήμα 2.4.4: Άρθρωση βραχίονα

Ξεκινώντας να το αναλύουμε απο αριστερά προς τα δεξιά φαίνεται αρχικά η είσδος Q που είναι ο πίνακας 6x1 που δημιουργήθηκε σε προηγούμενο βήμα για τη δημιουργία τροχιάς. Το function3 επιλέγει το σωστό στοιχείο του πίνακα και το αντιστοιχεί στην άρθρωση που είμαστε. Στη συνέχεια υπάρχει ένας αθροιστής που βρίσκει τη διαφορά επιθυμητής και παρούσας θέσης της συγκεκριμένης άρθρωσης, ώστε να αξιοποιηθεί απο τον ελεγκτή μας. Η σωστή τιμή δίνεται στο εικονίδιο joint για τη κίνηση του βραχίονα και το οποιό θα αναλυθεί αργότερα. Σαν έξοδο έχουμε την παρούσα θέση του της άρθρωσης και ώς αναφορά το εικονίδιο SPS πρόκειται για μετατροπέα της τιμής του ελεγκτή σε αντίστοιχη τιμή ικανή να διαβαστεί απο το joint.



Εικόνα 2.4.4: Ρυθμίζεις άρθρωσης.

Παραπάνω είναι ο πίνακας ρυθμίσεων για το εικονίδιο joint. Όπως βλέπουμε έχουμε βάλει να έχουμε σαν είσοδο την επιθυμητή θέση που θέλουμε να έχει η άρθρωση, να υπολογίζει μόνο του τις ροπές που ασκούνται σε αυτόν και σαν έξοδο να έχουμε τη θέση του. Στη περίπτωση που υπολογίζαμε τη δυναμική του βραχίονα μόνοι μας τότε θα είχαμε τη ροπή σαν είσοδο και σαν έξοδο τη θέση, την ταχύτητα και την επιτάχυνση.

Τέλος το εικονίδιο go to μας εξηπηρετεί στο να χρησιμοποιήσουμε τη παρούσα τιμή της θέσης της άρθρωσης σε άλλά τμήματα του προγράμματος. Για παράδειγμα μπορεί να χρησιμοποιηθεί στο κομμάτι της ανατροφοδότησης για τη σύγκριση των θεωρητικών και πραγματικών τιμών των αρθρώσεων του βραχίονα.

ΚΕΦΑΛΑΙΟ 3

ΕΛΕΓΧΟΣ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

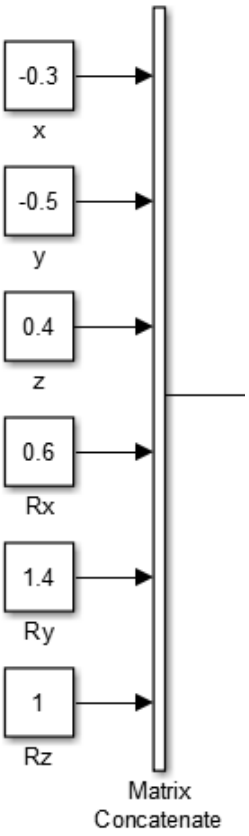
Για τον έλεγχο του ρομποτικού βραχίονα αρχικά έπρεπε να βρούμε έναν τρόπο δημιουργίας των κατάλληλων εντολών που θα έπρεπε να σταλθούν στο Controller του ρομποτικού βραχίονα UR5. Οπότε χρησιμοποιήθηκε το πρόγραμμα του Matlab και συγκεκριμένα του Simulink μέσω του οποίου μπορέσαμε να στείλουμε εύκολα τις επιθυμητές εντολές. Στη συνέχεια αποφασίστηκε η μετάδοση της εντολής αυτής να γίνει μέσω internet και συγκεκριμένα με τη μέθοδο TCP/IP. Όμως και ο υπολογιστής στην αγγλία και το Controller του UR5 είχαν private address, με αποτέλεσμα να χρησιμοποιήσουμε έναν υπολογιστή στο πανεπιστήμιο του Πεκίνου που τρέχει Linux ώς server. Τέλος για να μπορέσουμε να αξιοποιήσουμε τη σύνδεση αυτή για μελλοντική χρήση δημιουργήθηκε σύστημα ανατροφοδότησης μέσω του οποίου στέλνονται δεδομένα απο το UR5 και εμφανίζονται στο πρόγραμμα του Matlab.

3.1 Δημιουργία προγράμματος στο Matlab

Το πρόγραμμα του Matlab είναι αρκετα διαδεδομένο, ειδικότερα το κομμάτι του Simulink και χρησιμοποιείται σε πολλούς διαφορετικούς τομείς για την επίλυση μαθηματικών προβλημάτων. Επίσης έχει τη δυνατότητα δημιουργίας προγραμμάτων, χρησιμοποιώντας εντόλες απο τη γλώσσα C++ και απεικόνιση διαγραμμάτων. Στο συγκεκριμένο κεφάλαιο θα αναλυθούν τα βασικά στοιχεία που χρησιποποιήθηκαν ώστε να πραγματοποιηθεί ο έλεγχος του ρομποτικού βραχίονα UR5 και η ανατροφοδότηση του.

3.1.1 Έλεγχος του UR5 μέσω του Matlab

Αρχικά φτιάχτηκε ένας πίνακας 6x1, όπου τα 3 πρώτα στοιχεία δηλώνουν την επιθυμητή θέση και τα άλλα 3 τον προσανατολισμό στον οποίο θέλουμε να καταλήξει το τελικό στοιχείου δράσης.



Σχήμα 3.1.1.1: Δημιουργία πίνακα επιθυμητής θέσης και προσανατολισμού.

Το εικονίδιο του matrix δημιουργεί τον πίνακα χρησιμοποιώντας τα στοιχεία αριστερα του σαν είσοδο και δεξιά του είναι η έξοδος.

Στη συνέχεια μετατρέπουμε τον πίνακα αυτό σε εντολή αναγνωρίσιμη απο το Controller του UR5. Η εντολή αυτή έχει τρείς μορφές:

I. MoveJ

   Ο βραχίονας κινείται ελεύθερα και φτάνει στο τελικό του προορισμό. Τα στοιχεία που δηλώνονται είναι οι κλίσεις της κάθε άρθρωσης, η μέγιστη ταχύτητα, η επιτάχυνση και ο χρόνος διεκπεραίωσης. Η εντολή διαμορφώνεται ώς εξής:

   movej(q, ai= a, ui=u, ti=t)

   Με a, u , t η επιθυμητή επιτάχυνση, ταχύτητα και χρόνος διεκπαιρέωσης αντίστοιχα.

Για δηλώσουμε τις γωνίες κλίσης των μελών του βραχίονα μπορούμε είτε να τις πάρουμε απο την αντίστροφη κινηματική είτε χρησιμοποιώντας την παρακάτω εντολή στη θέση του q :

(get_inverse_kin(p))

Όπου p τα στοιχεία του προηγούμενου πίνακα διαχωρίζοντας τα με « , » .

II. MoveL

Είναι η ίδια περίπτωση με την παραπάνω με τη διαφορά οτι δίνεται ο τελικός προορισμός (τα έξι στοιχεία απο τον πίνακα του προηγούμενου βήματος) διαχωρίζοντας τα με « , » . Η αντίστοιχη εντολή είναι:

movel(p , ai= a, ui=u, ti=t)

III. MoveP

Δίνοντας αυτή την εντολή ο ρομποτικός βραχίονας κινείται προς την επιθυμητή θέση του, χρησιμοποιώντας παραβολικές μίξεις. Οπότε η εντολή σε αυτή τη περίπτωση είναι:

movel(p , ai= a, ui=u, ri=r)

Με r την απόσταση της περιοχής μίξης της παραβολής.

Για να γίνει η μετατροπή αυτή χρησιμοποιείται ένα εικονίδιο ονόματι function, στο οποίο έχουμε τη δυνατότητα να γράψουμε κώδικα. Έτσι γράφοντας εντολές σε γλώσσα C++ μπορούμε να μετατρέψουμε τα στοιχεία που είναι γράμματα όπως το « , » σε αριθμούς και να δημιουργήσουμε έναν μεγαλύτερο πίνακα αριθμών σαν έξοδο. Αυτό επιτυγχάνεται με την εντολή:

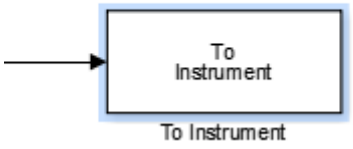
K(n,1)=double (‘g’);

Όπου,

K(n,1) = ο πίνακας μας και «n» το nστο στοιχείο του.

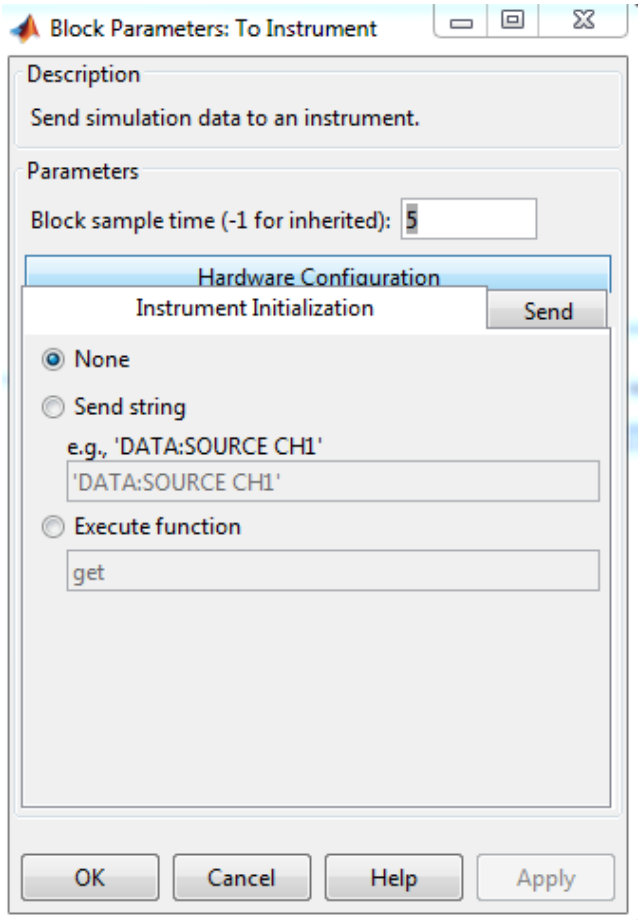
g = το επιθυμητό γράμμα που θέλουμε να μετατρέψουμε σε αριθμό.

Ο τελικός αυτός πίνακας καταλήγει σε ένα εικονίδιο που λέγεται «to instrument» , το οποίο χρησιμοποιείται για να γίνει η σύνδεση μεταξύ του υπολογιστή μας με τον server και να μετατρέψει τον πίνακα αυτόν σε κατάλληλη μορφή String ώστε να σταλθεί.



Εικόνα 3.1.1.1: To instrument

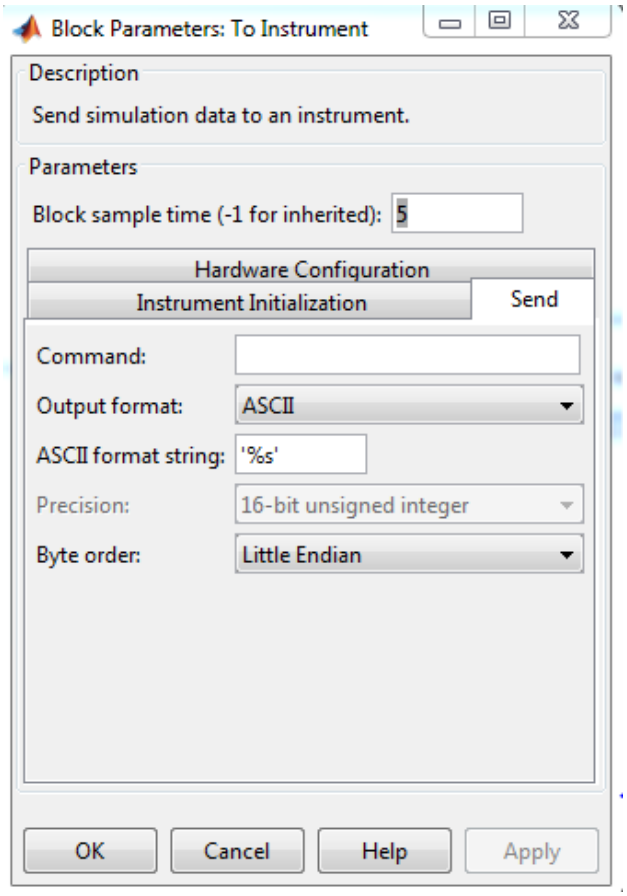
Οι ρυθμίσεις του φαίνονται στις επόμενες εικόνες:



Εικόνα 3.1.1.2: Αποστολή πρώτης εντολής

Όπως βλέπουμε στο παράθυρο αυτό, μπορούμε να δηλώσουμε στο sample time τη συχνότητα που μεταδίδονται οι εντολες μας και τι θέλουμε να

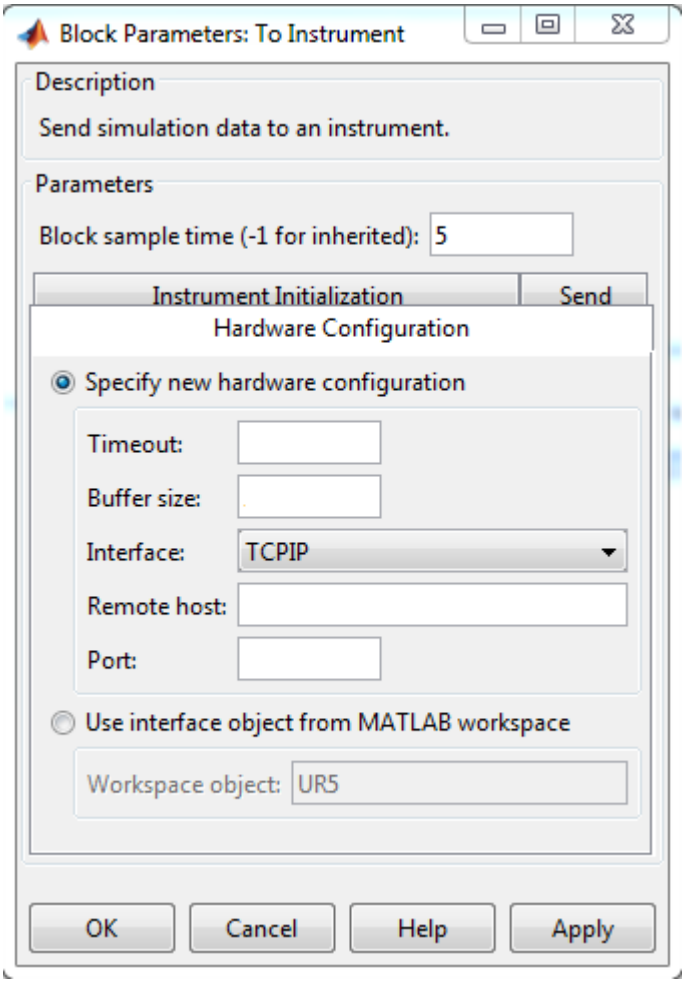
στείλουμε με το που γίνει εφικτή η σύνδεση μεταξύ του υπολογιστή μας και του server.



Εικόνα 3.1.1.3: Δήλωση στοιχείων που θα αποσταλούν.

Στο σημείο αυτό δηλώνουμε τη μορφή των στοιχείων που θέλουμε να στείλουμε. Στη δική μας περίπτωση θέλουμε να έχουν τη μορφή ASCII και συγκεκριμένα να είναι String «%S».

Παρακάτω φαίνεται ο τρόπος σύνδεσης με τον Server. Υπάρχουν δύο δυνατές περιπτώσεις. Στη πρώτη επιλέγεις έναν απο τους κλασσικούς τρόπους μετάδοσης (serial , GPIB , usb , TCP/IP , UDP) και συμπληρώνεις τα δεδομένα όπως φαίνεται στη εικόνα:

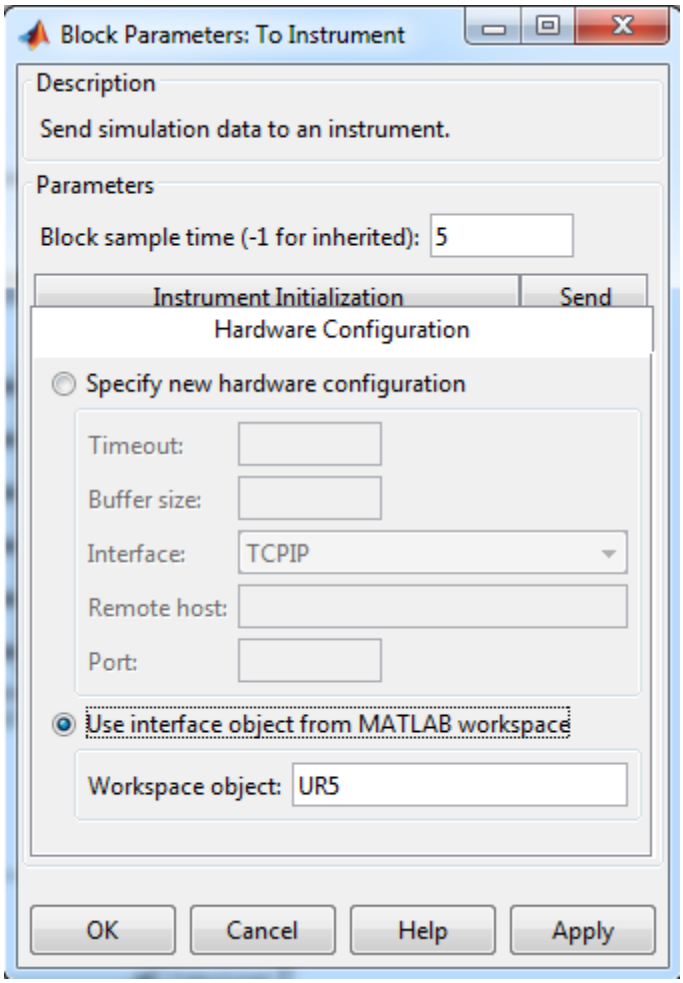


Εικόνα 3.1.1.4: Επιλογή τρόπου αποστολής

Για την περίπτωση μας (TCP/IP) τα δεδομένα αυτά είναι:

- Timeout
  Ο μέγιστος χρόνος για τη μετάδοση των δεδομένων
- Buffer Size
  Ο μέγιστος όγκος δεδομένων
- Interface
  Εδώ επιλέγεται ο τρόπος μετάδοσης
- Remote Host
  Η διεύθυνση (IP) του Server
- Port
  Η επιθυμητή θύρα σύνδεσης

Ο δεύτερος τρόπος, ο οποίος χρησιμοποιήθηκε για τον έλεγχο του UR5 είναι η δημιουργία ενός αντικειμένου ονόματι UR5 όπως φαίνεται παρακάτω:



Εικόνα 3.1.1.5: Δημιουργία αντικειμένου.

Η δήλωση των παραμέτρων του συγκεκριμένου αντικειμένου γίνεται πηγαίνοντας στις παραμέτρους του μοντέλου (δεξί κλικ -> model parameters), επιλέγοντας τη καρτέλα Callbacks και στη συνέχεια:

- InitFcn:

  Στο κομμάτι αυτό γίνεται η σύνδεση του υπολογιστή με τον server γράφοντας των παρακάτω κώδικα:

```
UR5 = tcpip('ip',port,'NetworkRole','Client');
set(UR5,'OutputBufferSize',1900);
```

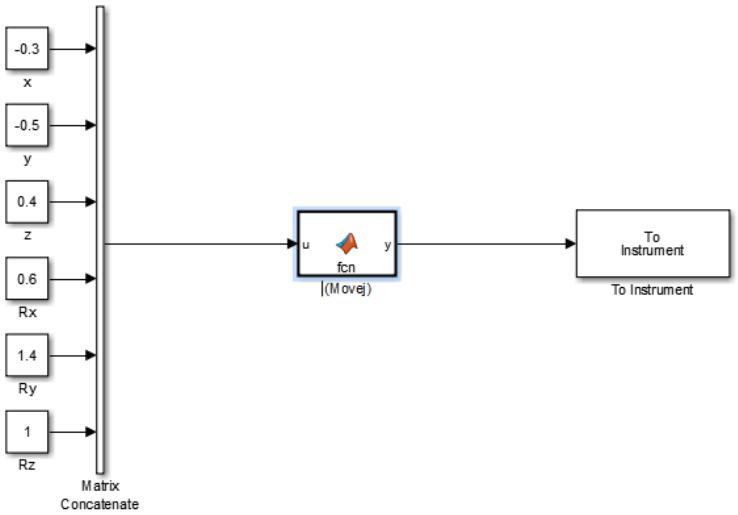
Δηλαδή δημιουργείται σύνδεση TCP/IP με «ip» την διεύθυνση IP του Server, στη θύρα «port» με τη μέθοδο NetworkRole και ο υπολογιστής μάς χρησιμοποιείται σαν Client.

- StopFcn

  Γράφουμε εντολές που μας βοηθάνε στο κλείσιμο του προγράμματος. Μία τέτοια εντολή είναι η παρακάτω που κλείνει μία υπάρχουσα σύνδεση.

  fclose(UR5);

Στην συνέχεια φαίνεται ολοκληρωμένο το Simulink για τον έλεγχο του UR5 με τυχαίες τιμές για την επιθυμητή θέση του τελικού στοιχείου δράσης:



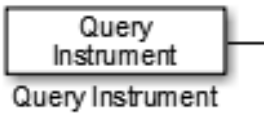
Σχήμα 3.1.1.2: Αποστολή δεδομένων στο simulink

3.1.2 Πρόγραμμα ανατροφοδότησης

Για τον σωστό έλεγχο του UR5 δεν αρκεί μόνο να στέλνουμε εντολές για την κίνηση του αλλα και να δεχόμαστε δεδομένα απο αυτό. Για το λόγο αυτό, στο υπάρχον πρόγραμμα προστέθηκε ένα κομμάτι όπου λαμβάνονται τα δεδομένα απο το UR5 και στη συνέχεια εμφανίζονται σε μορφή πινάκων ή διαγραμμάτων. Επίσης υπάρχει η επιλογή η προσωρινή ή μόνιμη αποθήκευσης τους για περεταίρω χρήση.

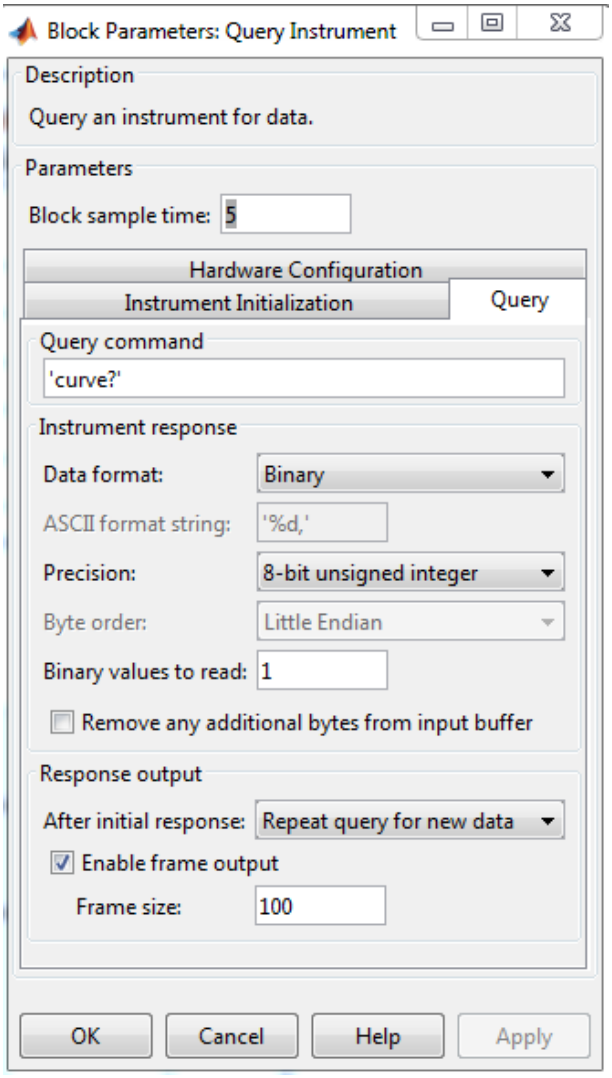
Αρχικά για την συλλογή των δεδομένων χρησιμοποιήθηκε το εικονίδο «Query Instrument», το οποίο χρησιμεύει επίσης για την σύνδεση του

υπολογιστή με το server. Οι ρυθμίσεις αυτού είναι περίπου ίδιες με το «το instrument».



Εικόνα 3.1.2.1: Query Instrument.

Όσον αφορά το παράθυρο για τον προσδιορισμό της πρώτης εντολής που θα σταλθεί όταν γίνει η σύνδεση, είναι ακριβώς το ίδο. Μεγάλη διαφορά παρατηρούμε στο παρακάτω παράθυρο query:



Εικόνα 3.1.2.1: Λήψη δεδομένων.

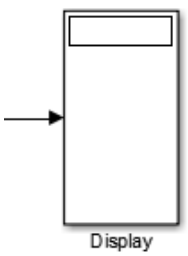
Στο αρχή της καρτέλας δηλώνουμε την επιθυμητή μορφή (Data format) που θέλουμε για να διαβάσουμε τα δεδομένα και παρακάτω έχουμε τη δυνατότητα να προσθέσουμε περισσότερες λεπτομέρειες για τη μορφη αυτή όπως το αν ειναι integer, double και άλλα. Στο κάτω μέρος γίνεται η επιλογή του μεγέθους των εισερχόμενων δεδομένων (frame size) και τι να κάνει το πρόγραμμα μόλις έρθουν νέα δεδομένα (After initial responce).Το τελευταίο παράθυρο περιλαμβάνει τη σύνδεση του υπολογιστή με τον server και οι επιλογές που έχουμε είναι ίδιες με αυτές που είχαμε και στο «to instrument». Απλά πρέπει να προσέξουμε και να δημιουργήσουμε αντικείμενο με διαφορετικό όνομα με το προηγούμενο και διαφορετική θύρα.

Στη συνέχεια τοποθετήσαμε ένα function στο οποίο γίνεται η μετατροπή των δεδομένων που έρχονται σε πίνακα. Συνήθώς πρόκεται για ένα string το οποίο ξεκινάει με «(», τελειώνει με «)» και τα στοιχεία που χρειαζόμαστε διαχωρίζονται με «,». Στο σημείο αυτό δημιουργήθηκε μία επανάληψη μέσα στην οποία:

1. Το πρόγραμμα καταλαβαίνει την αρχή και το τέλος της εντολής.
2. Αναγνωρίζει τους αριθμούς απο τα γράμματα.
3. Τοποθετεί τους αριθμούς σε έναν πίνακα, τον οποίο αποθηκεύει σε ένα buffer.
4. Βγάζει σαν έξοδο τον τελικό πίνακα απο το buffer.

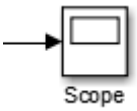
Τέλος ο πίνακας αυτός μπορεί να αξιοποιηθεί με διαφόρους τρόπους ανάλογα τα περιεχόμενα του. Μερικά παραδείγματα είναι τα παρακάτω:

- Εμφάνιση του πίνακα μέσω του display.



Εικονα 3.1.2.2: Display.

- Παρουσίαση των δεδομένων σε γράφημα μέσω του scope.

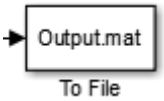


Εικόνα 3.1.2.3: Scope

- Αποθήκευση των δεδομέων στο πρόγραμμα για προσωρινή χρήση

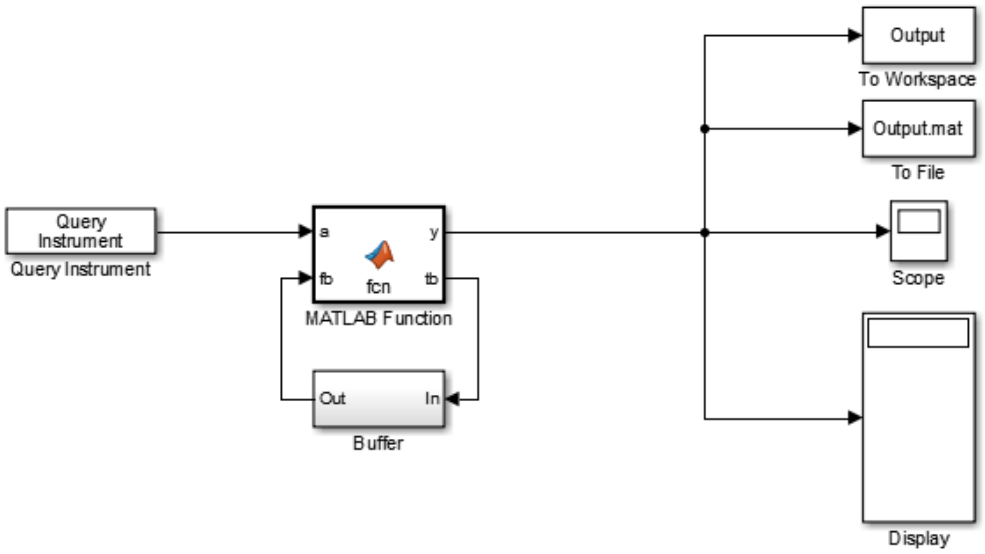
Εικόνα 3.1.2.4: To workspace.

ή σε αρχείο



Εικόνα 3.1.2.5: To file.

Το τελικό πρόγραμμα της ανατροφοδότησης είναι το παρακάτω:



Σχήμα 3.1.2.1: Σύστημα ανατροφοδότησης στο Simulink.

3.2 Δημιουργία προγράμματος για το Controller του UR5.

Κάθε ρομποτικός βραχίονας έχει έναν controller, ο οποίος είναι υπεύθυνος για τον έλεγχο του. Αυτό γίνεται γράφοντας κάποιο πρόγραμμα στον controller σε γλώσσα που αναγνωρίζεται απο αυτον. Στην προκειμένη περίπτωση η γλώσσα που χρησιμοποιείται στο UR5 ονομάζεται URScript και στη συνέχεια αναφέρουμε χρήσιμες πληροφορίες για την συγγραφή της:

- Οι βασικοί τύποι μεταβλητών της είναι:
  - Boolean (αληθής ή ψευδής)
  - Αριθμοί (integer or float).
  - Pose (Ο συνδιασμός θέσης και προσανατολισμού).
  - Χαρακτήρες.

- Για τον έλεγχο ροής χρησιμοποιούμε τις δηλώσεις if και while. Παραδείγματα αυτων φαίνονται παρακάτω.

```
if b>3 :
    a=a+1
elif c<7:
    c=c*a
else:
    a=a+b
end
```

```
k=[1,2,3,4,5]
l=0
while l<5
    k[l] = k[l]*2
end
```

- Υπάρχει η δυνατότητα δήλωσης μίας συνάρτησης και ορίζεται ώς:

```
def sum(b,c)
    return b+c
end
```

- Αρκετά συμαντική είναι η δημιουργία Threads. Η σύνταξη των οποίων φαίνεται στη συνέχεια:

```
thread thr():
    #κάνε κάτι
    return
end
```

Και για να την καλέσουμε γίνεται με την εντολή :

```
Th=run thr()
```

Στο τέλος για να κλείσουμε το Thread γράφουμε:

```
kill Th
```

Οι εντολές για την κίνηση του βραχίονα αναλύθηκαν στο κεφάλαιο για την αποστολή της επιθυμητής εντολής σε κατάλληλη μορφή απο το περιβάλλον του προγράμματος Matlab. Τέλος υπάρχουν αρκετές εντολές για την άντληση δεδομένων απο το UR5 σχετικά με την κίνηση που εκτελεί και την αποστολή αυτών στο πρόγραμμα του Matlab . Για περισσότερες πληροφορίες πάνω στις εντολές αυτές υπάρχει ειδικό manual συγγραφής της URScript. Εμείς στη συνέχεια θα αναλύσουμε μόνο αυτές που χρησιμοποιήσαμε στο πείραμα μας.

Ένα απλό πρόγραμμα για την λήψη δεδομένων απο έναν υπολογιστή γνωρίζοντας τη διέυθυνση IP και η κίνηση αυτού σύμφωνα με τα δεδομένα αυτά, είναι η παρακάτω:

```
BeforeStart
  open:=socket_open("127.0.0.1",21)
  Loop open= False
    open:=socket_open("127.0.0.1",21)
  targetPos:=p[0,0,0,0,0,0]
  counter:=0
Robot Program
  receiveFromServ:=socket_read_ascii_float(6)
  Loop receiveFromServ[0]≠6
    Wait: 0.3
    receiveFromServ:=socket_read_ascii_float(6)
  Loop counter<6
    targetPos[counter]=receiveFromServ[counter+1]
    counter:=counter+1
  MoveJ
    targetPos
  counter:=0
```

Στο παραπάνω πρόγραμμα αρχικά πρίν την εκκίνηση του προγράμματος δημιουργείται σύνδεση μεταξύ του Controller και του υπολογιστή με IP=127.0.0.1 στη θύρα 21. Στη συνέχεια οι εντολές του προγράμματος εκτελούνται ξανα και ξανά μέχρι να κλείσει η σύνδεση ή να το σταματήσουμε απο το Controller χειροκίνητα. Οι εντολές αυτές διαβάζουν τα δεδομένα που λαμβάνονται σε μορφή ASCII, αποθηκεύουν τα πρώτα έξι στοιχεία σε έναν πίνακα και εν συνεχεία αυτός χρησιμοποιείται για την κίνηση του βραχίονα μέσω της εντολής MoveJ.

Στη περίπτωση που θέλουμε να έχουμε και κάποια δεδομένα απο τη κίνηση του UR5 και να τα αποστείλουμε στο πρόγραμμα του Matlab (μέσω του server), τότε πρέπει να δημιούργήσουμε ένα Thread που να τρέχει παράλληλα και να καλείται απο το πρόγραμμα μέσω κατάλληλης εντολής. Στο συγκεκριμένο Thread εμείς επιχειρήσαμε να βρούμε τις γωνίες κλήσης της κάθε άρθρωσης αξιοποιώντας την εντολή:

```
Get_joint_positions()
```

Η εντολή αυτή, επιστρέφει ένα διάνυσμα έξι δεδομένων, το οποίο αποθηκεύται σε πίνακα 6x1 και στη συνέχεια μπορούμε να το αποστείλουμε στο server γράφοντας:

Socket_set_var(name,value)

Με «name» το όνομα της μεταβλητής και «value» τον πίνακα.

3.3 Πρόγραμμα διακομιστή

Για τη δημιουργία σύνδεσης ανάμεσα σε δύο υπολογιστές και τη μεταφορά δεδομένων μεταξύ τους, χρειάζεται να λειτουργεί τουλάχιστον ο ένας απο τους δύο υπολογιστές σαν Server και να είναι γνωστή η διευθυνση.

Στη δική μας περίπτωση, οι δύο αυτοί υπολογιστές είναι το Controller του UR5, ο υπολογιστής με το πρόγραμμα του Matlab και δυστυχώς οι διευθύνσεις και των δύο ειναι private (δηλαδή δεν μας είναι γνωστές). Έτσι χρησιμοποιήθηκε ένας υπολογιστής με public ip διεύθυνση (σε περιβάλλον linux) που είναι τοποθετημένος στο πανεπιστήμιο του Πεκίνου. Σε αυτόν τον υπολογιστή συνδέεται το controller του UR5 μαζέ με τοω υπολογιστή που περιέχει το Matlab και χρησιμοποιείται σαν μεσολαβητής για τη μεταφορά των δεδομένων μας.

Για τη δημιουργία του κατάλληλου προγράμματος αξιοποιήθηκε η εφαρμογή NetBeans και η συγγραφή του έγινε σε γλώσσα Java. Στη συνέχεια φαίνονται οι σημαντικότερες εντολές:

- Επιλογή της επιθυμητής θύρας:

  int port = Integer.parseInt(po);

- Δημιουργεία των συνδέσεων του server με το Matlab και το UR5 :

```
ServerSocket socket = new ServerSocket(port);

JOptionPane.showMessageDialog(null, "Connect Matlab Program and then press OK!");
   Socket sock = socket.accept();

JOptionPane.showMessageDialog(null, "Connect UR5 and then press OK!");
   Socket sock2 = socket.accept();
```

- Έλεγχος των συνδέσεων:

```
if ((sock != null) && (sock2 != null))
{
    JOptionPane.showConfirmDialog(null, "Do you want to start?");
  }
  else
  {
    JOptionPane.showMessageDialog(null, "Connection Error. Exit program!");
    System.exit(0);
  }
```

Αν είναι επιτυχείς οι συνδέσεις τοτε σε ρωτάει αν επιθυμείς να ξεκινήσει η μεταφορά των δεδομένων. Διαφορετικά σου λέει "Connection Error. Exit program!" και το πρόγραμμα σταματάει.

- Μεταφορά δεδομένων:

```
while ((sock != null) && (sock2 != null))
   {
    Scanner scan1 = new Scanner(sock.getInputStream());
    String receive = scan1.next();
     send.println(receive);
   }
```

Για να έχουμε τη δυνατότητα να κλείσουμε ανα πάσα στιγμή τη σύνδεση μεταξύ του server με τους client τοποθετήθηκε το παρακάτω thread :

```
class MyThread
  extends Thread
{
  public void run()
  {
    JOptionPane.showMessageDialog(null, "If you want to close the program, press ok!");
    System.exit(0);
  }
}
```

Στο τέλος το συγκεκριμένο πρόγραμμα στάλθηκε στο πανεπιστήμιο του πεκίνου, όπου για να μπορέσει να τρέξει έγιναν τα παρακάτω βήματα:

1) Ανοίξαμε το Terminal

2)Ελέγξαμε αν ο υπολογιστής περιέχει κάποια έκδοση που να διαβάζει προγράμματα java:

java –version

3)Δυστυχώς δεν είχε και εγκαταστήσαμε εμείς γράφοντας :

sudo apt-get install openjdk-9-jdk

4) Αλλάξαμε τις άδεις χρήσης:

Δεξί κλικ στο αρχείο jar --> Properties --> Permissions --> Execute: Allow executing file as program

5) Έγινε η επιλογή του προγράμματος, με το οποίο θα ανοίγει τα jar αρχεία:

Δεξί κλικ στο αρχείο --> Open with --> Other Application --> OpenJDK java Runtime

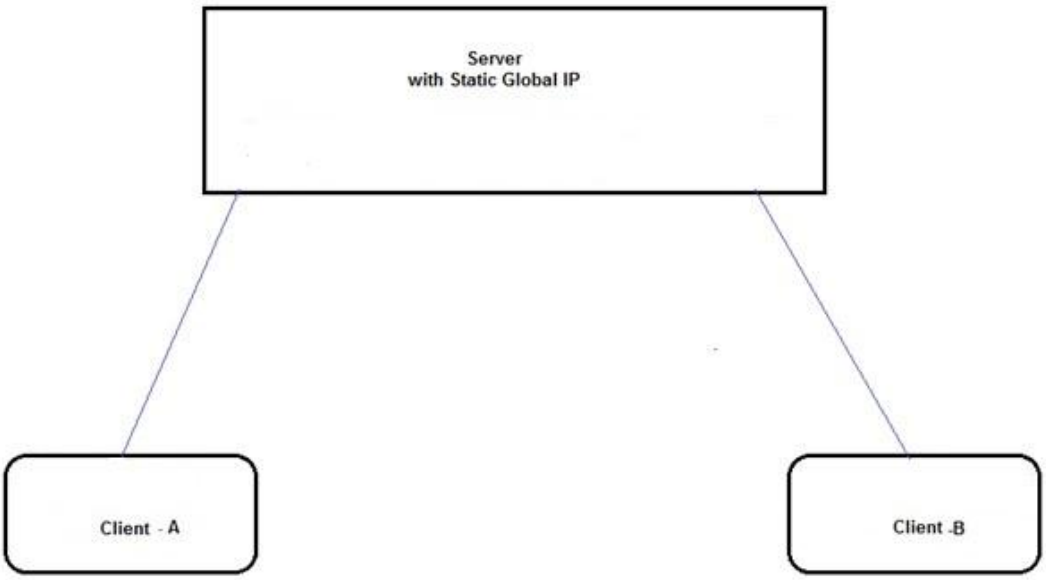
6)Διπλό κλικ στο αρχείο για να τρέξει.

Για να είναι σωστή η σύνδεση μας είναι απαραίτητο να τρέξει πρώτα το πρόγραμμα του server και στη συνέχεια ακολουθώντας τις οδηγίες του προγράμματος να τρέξουμε το controller του UR5 και το Matlab.

ΚΕΦΑΛΑΙΟ 4

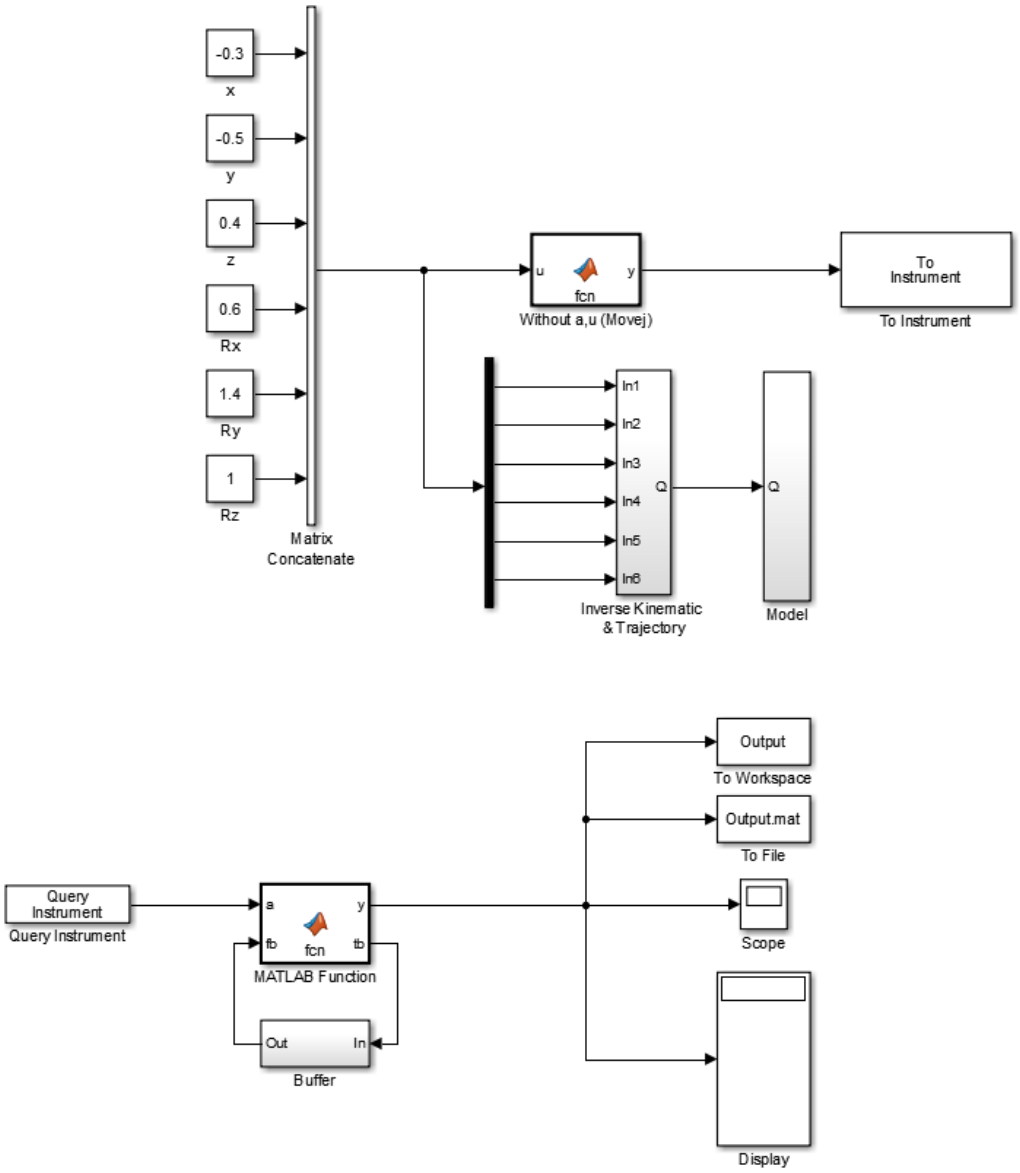
ΣΥΝΔΕΣΗ ΤΟΥ UR5 ΜΕ ΤΟ MATLAB

Στα προηγούμενα κεφάλαια δόθηκαν οδηγίες για το πως γίνεται ο προγραμματισμός του controller για το UR5 και η δημιουργία σύνδεσης του με τον πρόγραμμα του Matlab. Αποτέλεσμα της σύνδεσης αυτής είναι το παρακάτω σχήμα:



Εικονα 4.1: Σύνδεση Server με τους Clients.

Στα προηγούμενα κεφάλαια αναπτύχθηκαν έννοες του Matlab που περιλαμβάνουν τον έλεγχο, την ανατροφοδότηση και τη μοντελοποίηση του ρομποτικού βραχίονα. Παρακάτω φαίνεται το αποτέλεσμα απο την ένωση των κομματιών αυτών:



Σχήμα 4.1: Τελικό σχέδιο στο Simulink.

4.1 Αποτελέσματα πειράματος

Για να δοκιμάσουμε το πείραμα, χρειάστηκε η βοήθεια ενός φοιτητή απο το πανεπιστήμιο του Πεκίνου. Αρχικά έγραψε το πρόγραμμα στο controller, μετά έκανες τις επιθυμητές ενέργειες που περιγράψαμε προηγουμένος για να μπορέσει να τρέχει το πρόγραμμα στον server και τέλος εφόσον έγινε η σύνδεση τράβηξε εικόνες απο τον Server και το UR5. ‘Ετσι τρέχοντας όλα αυτά τα προγράμματα τελικά επιτεύχθηκε η κίνηση του ρομποτικού βραχίονα στο επιθυμητό σήμειο όπως φαίνεται στις φωτογραφίες παρακάτω:

Εικόνα 4.1.1: Αρχική θέση.

Εικόνα 4.1.2: Τελική θέση.

Στη συνέχεια βλέπουμε φωτογραφία απο το server:

```
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
transferred:(-3.000000e-01,-5.000000e-01,4.000000e-01,6.000000e-01,-1.400000e+00
,1.200000e+00)
```

Εικόνα 4.1.3: Οι εντολές που διαβάζονται στο server.

4.2 Συμπεράσματα πειράματος για μελλοντική χρήση

Μετά απο διάφορα πειράματα που έγιναν, τα συμπεράσματα που προέκυψαν είναι ότι η σύνδεση αυτή είναι τελικά εφικτή με δύο σημαντικά προβλήματα. Το πρώτο μειωνέκτημα ήταν στο delay της μεταφοράς των δεδομένων και το δεύτερο ήταν ότι με τη γεωμετρική επίλυση της αντίστροφης κινηματικής είχαμε ελάχιστη απόκλιση των τιμών.

Στα περισσότερα πειράματα μας το delay αντιστοιχούσε κατα μέσω όρο στα τριάντα δευτερόλεπτα. Αρχικά τα προγράμματα μας ήταν πιο σύνθετα και προσπαθήσαμε να τα κάνουμε πιο λιτά αλλά το delay μειώθηκε ελάχιστα. Μεγάλη διαφορά στο delay παρατηρήσαμε όταν αλλάξαμε το sample time, δηλαδή τον χρόνο στον οποίο στέλνονται οι εντολες. Μεγαλώνοντας το χρόνο το σύστημα μας είχε καλύτερη ανταπόκριση και μειώσαμε το delay περίπου κατα δέκα δευτερόλεπτα.

Οπότε θα ήταν καλό στο μέλλον να βρεθούν τρόποι μείωσης αυτής της καθυστέρησης, ώστε ο ρομποτικός βραχίονας να ανταποκρίνεται άμεσα. Ένας τρόπος επίλυσης αυτού του προβλήματος είναι η μέθοδος NPD (network predictive control) κατα την οποία προβλέπονται τα σωστά δεδομένα που έρχονται στη περίπτωση αλλοίωσης τους.

Όσον αφορά το δεύτερο μειωνέκτημα που αντιμετωπίσαμε σχετικά με τις γωνίες κλίσης των συνδέσμων του βραχίονα, η μεγαλύτερη απόκλιση που βρήκαμε ήταν της τάξεως του 5%. Δυστυχώς η απόκλιση αυτή είναι αρκετα μεγάλη όταν χρειαζόμαστε ακρίβεια στις κινήσεις του βραχίονα.

Ένας σωστός τρόπος για την εύρεση των σωστών τιμών των αρθρώσεων είναι αν χρησιμοποιήσουμε τη μέθοδο machine learning, η οποία υπάρχει στο Matlab και μπορεί να τις βρίσκει μόνο του. Για να επιτευχθεί αυτό όμως θα χρειαστεί να παρθούν αρκετα πειραματικά δεδομένα και να τα εισάγουμε στο σύστημα.

Τέλος για μελλοντική χρήση θα ήταν χρήσιμο να υπάρχει ένα πρόγραμμα που να εισάγει στο πρόγραμμα του Matlab τις επιθυμητές θέσεις του βραχίονα και να μην τις τοποθετούμε εμείς χειροκίνητα. Αυτό θα μπορούσε να γίνει μέσω ενός raspberry το οποίο αξιοποιούσε τα δεδομένα που λάμβανε απο μία κάμερα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] John J. Craig, “Εισαγωγή στη ρομποτική”, 3η έκδοση, εκδόσεις Τζίολα 2009

[2] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villiani, Giuseppe Oriolo,“Ρομποτική”, εκδόσεις Fountas, 2009

[3] Katharina Kufieta, “Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments”, NTNU, 2014

[4] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, “Robot Dynamics and Control”, Second Edition, 2014

[5] Parham M. Kebria, Saba Al-wais, Hamid Abdi, and Saeid Nahavandi, “Kinematic and Dynamic Modelling of UR5 Manipulator”, IEEE, 2016

ΑΛΛΕΣ ΠΗΓΕΣ

[6] http://math.tutorcircle.com

[7] https://www.slideshare.net

[8] http://www.zacobria.com

[9] http://socketprogramming.blogspot.gr