ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ

Προγραμματισμός βιομηχανικού ρομποτικού βραχίονα για αποφυγή σύγκρουσης με συνεργαζόμενο άνθρωπο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΑΡΑΓΚΟΣ ΧΡΗΣΤΟΣ

Επιβλέπων : Γ.Χ.ΒΟΣΝΙΑΚΟΣ
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα Καθηγητή Γ.Χ. Βοσνιάκο για την άπταιστη συνεργασία που είχαμε κατά την εκπόνηση της διπλωματικής μου εργασίας. Η καθοδήγηση του ήταν πολύ κρίσιμη σε ερευνητικά, τεχνικά και προσωπικά ζητήματα. Θα ήθελα επίσης να τον ευχαριστήσω που μου εμπιστεύτηκε ένα τόσο ενδιαφέρον και καίριο θέμα και που πίστεψε στις δυνατότητες μου. Με την μοναδική αυτή εμπειρία που μου προσέφερε έιχα την δυνατότητα να ασχοληθώ με καινούριους για μένα τομείς (π.χ Ρομποτική), να διευρύνω σημαντικά τις γνώσεις μου ως μηχανικός και να βελτιώσω τον τρόπο σκέψης μου και εργασίας μου.

Ευχαριστώ επίσης τους ανθρώπους που δουλέψαμε μαζί αυτούς τους μήνες στο εργαστήριο, για το φιλόξενο κλίμα που έχουν δημιουργήσει και ιδιαιτέρως τους Ηλία Μάτσα, Γιώργο Παπαζέτη, Ευάγγελο Τζίμα και Νίκο Κούκη για τις πολύτιμες οδηγίες που μου παρείχαν. Επιπλέον, θα ήθελα να ευχαριστήσω την Μαρία Μπούφαλη, τον Lucas Blein και την Ευγενία Μάνου που συμμετείχαν στις πειραματικές δοκιμές.

Ευχαριστώ τους επιστήθιους φίλους μου Μάριο Σταθακόπουλο, Βαγγέλη Μάρκου, Τηλέμαχο Παπαευαγγέλου και Άρη Σαρακηνό που μου συμπαραστέκονται και με συμβουλεύουν την τελευταία δεκαετία.

Τέλος, δεν θα μπορούσα να παραλείψω τους γονείς μου Γλυκερία Γιαννάκη, Φραγκίσκο Μαραγκό, τα αδέρφια μου Κωσταντίνο και Ελένη τους οποίους ευχαριστώ πολύ για την υποστήριξη και την αγάπη που μου έχουν δείξει όλα αυτά τα χρόνια.

Μαραγκός Χρήστος

Αθήνα , Οκτώβριος 2017

Περίληψη

Στο πλαίσιο της παρούσας διπλωματικής εργασίας διερευνάται ο προγραμματισμός βιομηχανικού ρομπότ Staubli RX-90 σε γλώσσα προγραμματισμού V+ ώστε το πρόγραμμα που έχει αρχικά φτιαχτεί να μπορεί να αλλάζει κατά τη διάρκεια της εκτέλεσης του για λόγους ασφαλείας του ανθρώπου που δουλεύει μαζί με το ρομπότ. Συγκεκριμένα θα πρέπει είτε το ρομπότ να σταματά, είτε να αλλάζει πορεία όταν στην πορεία του βρίσκεται ο άνθρωπος.

Για τον προσδιορισμό της θέσης του ανθρώπου χρησιμοποιήθηκε τεχνολογία επαυξημένης πραγματικότητας. Η ανάπτυξη του εικονικού μοντέλου πραγματοποιήθηκε στο περιβάλλον του λογισμικού Unity3d. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για τον καθορισμό της συμπεριφοράς των τρισδιάστατων αντικειμένων είναι η C#. Ως συσκευή απεικόνισης χρησιμοποιήθηκε οθόνη ηλεκτρονικού υπολογιστή, ενώ η εικόνα του πραγματικού χώρου λαμβάνεται από τον αισθητήρα Kinect σε πραγματικό χρόνο.

Για την επικοινωνία του λογισμικού Unity3d με τον ελεγκτή του ρομπότ χρησιμοποιήθηκε η πλακέτα ArduinoMega. Η πλακέτα Arduino λαμβάνει σήματα από το εικονικό περιβάλλον του Unity3d όταν ο άνθρωπος βρίσκεται σε κάποιες προκαθορισμένες θέσεις στο πραγματικό χώρο κοντά στο ρομπότ και στη συνέχεια τα στέλνει στις εισόδους του ελεγκτή του ρομπότ. Ανάλογα με τις τιμές αυτών των σημάτων, το πρόγραμμα του ρομπότ λαμβάνει αντίστοιχα μέτρα μεταβολής της κίνησης του ρομποτικού βραχίονα, χρησιμοποιώντας αντίστοιχες υπορουτίνες. Η λογική του προγράμματος επιδεικνύεται για συγκεκριμένο παράδειγμα, αλλά είναι γενική και εφαρμόσιμη σε ποικιλία περιπτώσεων.

Λέξεις Κλειδιά

Ρομπότ Staubli RX90, V+ language, Unity3d, C#, Arduino, C/C++, Microsoft Kinect Xbox One, HRI, Ασφάλεια .

Περιεχόμενα

Περιεχόμενα Εικόνων

Περιεχόμενα Πινάκων

1

Εισαγωγή

1.1 Γενικά

Η ρομποτική είναι εκείνος ο κλάδος της επιστήμης του μηχανικού, που ασχολείται με τη σύλληψη, τη σχεδίαση, την κατασκευή και τις εφαρμογές των ρομπότ. Τα ρομπότ είναι μηχανές, που δεν έχουν τη μορφή ή τη συμπεριφορά του ανθρώπου, αλλά μπορούν να εκτελούν εργασίες, που κάνει ο άνθρωπος.

Η εξέλιξη των ρομπότ έχει περάσει από πολλά στάδια. Τα ρομπότ πρώτης γενιάς δεν είχαν την ικανότητα υπολογισμού και αίσθησης, σε αντίθεση με τα ρομπότ δεύτερης γενιάς, τα οποία διαθέτουν περιορισμένη υπολογιστική ισχύ, γλώσσες προγραμματισμού υψηλού επιπέδου και αισθητήρες ανατροφοδότησης. Τα ρομπότ της τρίτης γενιάς διαθέτουν νοημοσύνη με την έννοια ότι είναι ικανά να παίρνουν αποφάσεις κατά τη διάρκεια εκτέλεσης της εργασίας τους. Τις ικανότητες αυτές, τις αποκτούν μέσω τεχνικών της τεχνητής νοημοσύνης σε συνδιασμό με εξελιγμένες μορφές αισθητήρων αφής, δύναμης, απόστασης, όρασης, κ.ο.κ..

Τα βιομηχανικά ρομπότ είναι εξελιγμένα συστήματα αυτοματισμού, που χρησιμοποιούν ηλεκτρονικό υπολογιστή σαν μια βασική συνιστώσα του ελέγχου τους. Σήμερα, οι υπολογιστές αποτελούν ένα βασικό μέρος του βιομηχανικού αυτοματισμού. Κατευθύνουν γραμμές παραγωγής και ελέγχουν συστήματα κατασκευής (π.χ. εργαλειομηχανές, συγκολλητές, κοπτικές διατάξεις Laser κ.α.). Τα νέα ρομπότ εκτελούν ποικίλες εργασίες στα βιομηχανικά συστήματα και γενικά συμμετέχουν στον πλήρη αυτοματισμό των εργοστασίων.

Σύμφωνα με το Robot Institute της Αμερικής [42], ως ρομπότ μπορούμε να ορίσουμε έναν μηχανισμό σχεδιασμένο, ώστε μέσω προγραμματιζόμενων κινήσεων να μεταφέρει υλικά, τεμάχια, εργαλεία ή

ειδικευμένες συσκευές με σκοπό την εκτέλεση ποικίλων εργασιών. Ένας τέτοιος μηχανισμός περιλαμβάνει, συνήθως τις ακόλουθες συνιστώσες : ένα μηχανολογικό υποσύστημα, ένα υποσύστημα αίσθησης και ένα σύστημα ελέγχου.

Το μηχανολογικό υποσύστημα αποτελείται από μηχανισμούς, που επιτρέπουν στο ρομπότ να κινείται, όπως για παράδειγμα οι αρθρώσεις, το σύστημα μετάδοσης κίνησης, οι επενεργητές-κινητήρες , οι οδηγοί, κ.λ.π..

Το υποσύστημα αίσθησης βοηθάει το ρομπότ να συλλέγει πληροφορίες για την κατάσταση, στην οποία βρίσκονται τόσο το ίδιο, όσο και το περιβάλλον του. Εκτός των άλλων, δέχεται εξωτερικές εντολές, τις επεξεργάζεται, τις μεταφράζει σε ηλεκτρικά σήματα, που θα δοθούν στους κινητήρες τους ρομπότ, καθώς, επίσης, παράγει σήματα εξόδου, που θα πληροφορούν την κατάσταση του συστήματος. Στο υποσύστημα αίσθησης περιλαμβάνονται όργανα μέτρησης, αισθητήρες, ηλεκτρονικά στοιχεία, κ.λ.π..

Το σύστημα ελέγχου συνδυάζει με κατάλληλο τρόπο την αίσθηση με τη δράση έτσι, ώστε το ρομπότ να λειτουργεί αποτελεσματικά και με τον επιθυμητό τρόπο. Ο ελεγκτής του ρομπότ επιβλέπει και συντονίζει ολόκληρο το σύστημα, ενώ για τη σχεδίαση και την υλοποίηση του απαιτείται ο συνδιασμός γνώσεων από πολλές γνωστικές περιοχές, όπως είναι ο αυτόματος έλεγχος, η τεχνητή νοημοσύνη, η επιστήμη των υπολογιστών, κ.λ.π..

1.2 Ιστορική αναδρομή

Τα βιομηχανικά ρομπότ αναπτύχθηκαν ταυτόχρονα με τον υπολογιστικό αριθμητικό έλεγχο (CNC). Βεβαίως, το πρώτο ρομπότ κατασκευάστηκε το 1961, αλλά τα ρομπότ άρχισαν να παίζουν πρωτεύοντα ρόλο στη βιομηχανική παραγωγή κατά τα τέλη της δεκαετίας του 1970.

Τα ρομπότ υπάρχουν στη σκέψη του ανθρώπου χιλιάδες χρόνια ήδη. Χαρακτηριστικότερο παράδειγμα είναι ο θεός Ήφαιστος της Ελληνικής Μυθολογίας, ο οποίος στο εργαστήριο του στα έγκατα της γης είχε κατασκευάσει ανθρωποειδή ρομπότ-υπηρέτριες με ικανότητα κίνησης, ομιλίας και σκέψης, 'αυτόματα τρίποδα' που εξυπηρετούσαν, όπως αναφέρεται στην Ιλιάδα, αλλά και τον γνωστό Τάλω. Ο Τάλως ήταν ένα γιγάντιο ανθρωπόμορφο άρμα που ο Ήφαιστος έφτιαξε κατά παραγγελία του Μίνωα,

βασιλιά της Κρήτης και ετεροθαλή αδερφού του. Προφανής σκοπός του ήταν να προστατεύει το βασίλειο του Μίνωα από πιθανούς εισβολείς, έχοντας μάλιστα την ικανότητα να γυρνάει όλο το νησί της Κρήτης τρεις φορές την ημέρα. [5]

Η λέξη «ρομπότ» πρωτοεμφανίστηκε πριν από περίπου έναν αιώνα. Ο Τσέχος δραματογράφος Karel Capek, χρησιμοποίησε για πρώτη φορά το 1921 αυτόν τον όρο στο έργο του R.U.R. ("Rossum's Universal Robots")(**Εικόνα 1.1**). Αυτός ο όρος επινοήθηκε από την τσέχικη λέξη **«robota»**, η οποία έχει τη σημασία της καταναγκαστικής εργασίας. Στο έργο του περιγράφεται η κατασκευή έξυπνων συσκευών, οι οποίες χρησιμοποιούνται ως υπηρέτες του ανθρώπινου δημιουργού τους. [3]

Εικόνα 1.1: Φωτογραφία από το έργο R.U.R (ανθρωποειδή πίσω)

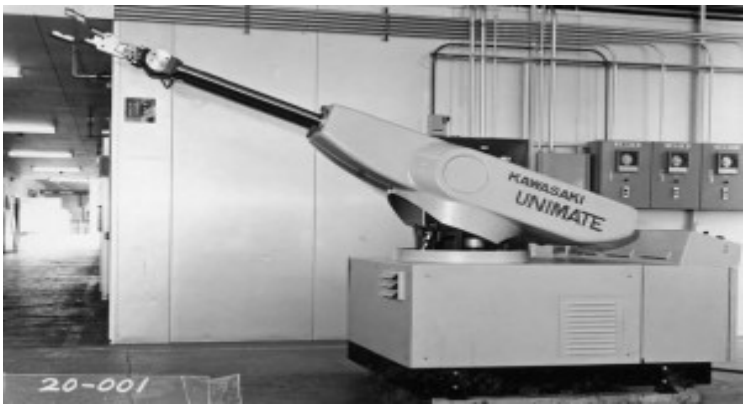
Κατά την διάρκεια του 1940, ο Ρώσος συγγραφέας επιστημονικής φαντασίας, **Isaac Asimov** περιέγραψε το ρομπότ ως μια μηχανή με εμφάνιση ανθρώπου απαλλαγμένο, όμως, από συναισθήματα [4]. Η συμπεριφορά του υπαγορευόταν από έναν εγκέφαλο προγραμματισμένο από ανθρώπους με ιδιαίτερο τρόπο, ώστε να ακολουθεί ηθικές αρχές. Για πρώτη φορά, επίσης, χρησιμοποιήθηκε από τον Asimov ο όρος «ρομποτική» για να περιγράψει τον

τομέα της επιστήμης που ασχολείται με τα ρομπότ. Το 1942 ο Asimov όρισε τους **τρεις νόμους που διέπουν την λειτουργία ενός ρομπότ** [4] :

- **Ένα ρομπότ δεν μπορεί να τραυματίσει ή μέσω της αδράνειας του να βλάψει ένα ανθρώπινο πλάσμα.**
- **Ένα ρομπότ πρέπει να υπακούει στις εντολές που δίνονται από τους ανθρώπους, εκτός και αν αυτό έρχεται σε αντίθεση με τον πρώτο νόμο.**
- **Ένα ρομπότ πρέπει να προστατεύει την ίδια τουν την ύπαρξη, εκτός και αν αυτό έρχεται σε αντίθεση με τον πρώτο ή τον δεύτερο νόμο.**

Ο όρος βιομηχανικό ρομπότ (industrial robot) καθιερώθηκε το 1954 από τον G.C.Devol (ΗΠΑ). Ο Devol περιέγραψε πως μπορεί να κατασκευαστεί ένα ελεγχόμενο μηχανικό χέρι, το οποίο μπορεί να εκτελεί διάφορες εργασίες στη βιομηχανία. Το πρώτο βιομηχανικό ρομπότ κατασκευάστηκε και τέθηκε σε λειτουργία το **1961** από την εταιρία **Unimation**. Έκτοτε, τέθηκαν σε λειτουργία χιλιάδες ρομπότ στην Αμερική, στην Ιαπωνία και στην Ευρώπη.

Η πρώτη γενιά των σύγχρονων ρομπότ απείχε κατά πολύ από τα ανθρωπόμορφα μηχανήματα, καθώς οι περισσότεροι κατασκευαστές δεν είχαν σκοπό να μιμηθούν το ανθρώπινο γένος. Το δημοφιλές ρομπότ Unimate(**Εικόνα 1.2**) της δεκαετίας του 1960 ήταν ικανό να κινήσει μονάχα το χέρι του προς διάφορες κατευθύνσεις και να ανοιγοκλείνει την παλάμη του.



Εικόνα 1.2 : Το Unimate, πρώτο βιομηχανικό ρομπότ

Το **1969** ο **Victor Scheinman** επινοεί στο Πανεπιστήμιο του Stanford τον πρώτο ρομποτικό βραχίονα(**Εικόνα 1.3**), ο οποίος προσομοιώνει την κίνηση ενός ανθρώπινου χεριού. Αυτή η ανακάλυψη δίνει νέα δυναμική στη ρομποτική. Τα ρομπότ πλέον μπορούν να εκτελέσουν πιο απαιτητικές και

ακριβείς διαδικασίες όπως το να συγκεντρώνουν αντικείμενα. Ακολουθεί ένας δεύτερος βραχίονας στο MIT, για αυτό και θα ονομαστεί 'MIT arm' , η μετεξέλιξη του οποίου θα βγει στην παραγωγή με την ονομασία PUMA (Programmable Universal Macshine for Assembly) (**Εικόνα 1.4**).

Εικόνα 1.3 : Stanford arm

Εικόνα 1.4 : Puma robot

1.3 Βασικές έννοιες ενός ρομπότ

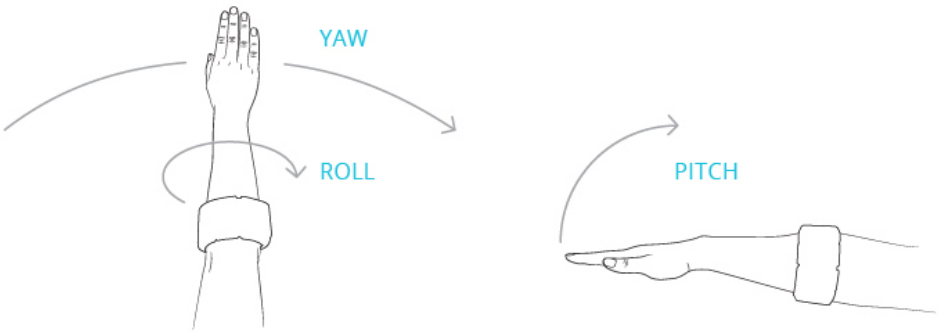
Κατ' αρχάς θα αναφερθούμε σε κάποιες βασικές έννοιες που χαρακτηρίζουν ένα ρομπότ, που θα διευκολύνουν στην καλύτερη κατανόηση όλων όσα ακολουθούν. Βασικές παράμετροι ενός ρομπότ είναι :

- **Ο αριθμός των αξόνων με τους οποίους κινείται** : Ένα ρομπότ που κινείται με δύο άξονες μπορεί να επιτελέσει εργασίες πάνω σε ένα επίπεδο, ενώ κάποιο άλλο που κινείται σε τρεις άξονες μπορεί να επιτελέσει εργασίες στον τρισδιάστατο χώρο.
- **Οι βαθμοί ελευθερίας** : Είναι το σύνολο των ανεξάρτητων μεταβλητών θέσης. Συνήθως ταυτίζονται με τον αριθμό των αρθρώσεων του ρομπότ. Ο αριθμός αυτός πρέπει να είναι γνωστός και μονοσήμαντα ορισμένος για κάθε διάταξη έτσι, ώστε να είναι εφικτός ο προσδιορισμός της θέσης των τμημάτων που τη συνιστούν.

- **Περιοχή δράσης (working space)** : Πρακτικά αποτελείται από ότι βρίσκεται μέσα στην εμβέλεια δράσης του ρομπότ και μπορεί να το 'φτάσει'.
- **Η κινηματική του** : Αυτή εξαρτάται από τους βραχίονες, τους συνδέσμους και γενικότερα όλα τα μέλη του ρομπότ και καθορίζει τις κινήσεις του, άρα και τη λειτουργία του.
- **Ικανότητα εξυπηρέτησης βάρους** : Τι βάρος μπορεί να εξυπηρετήσει το ρομπότ υπό ονομαστική και υπό μέγιστη ταχύτητα,
- **Ταχύτητα του ρομπότ** : Πόσο γρήγορα μπορεί το ρομπότ να φέρει το τελικό σημείο δράσης του στο ζητούμενο σημείο στην περιοχή δράσης, και να επιτελέσει τη ζητούμενη εργασία.
- **Επιτάχυνση** : Πόσο γρήγορα οι κινητήρες του ρομπότ μπορούν να επιταχύνουν και να επιβραδύνουν.
- **Ακρίβεια** : Πόσο κοντά μπορεί να φτάσει το ρομπότ στο ζητούμενο σημείο κατά μέσο όρο.
- **Επαναληψιμότητα** : Πόσο κοντά μεταξύ τους είναι οι θέσεις του ρομπότ κατά την επαναληπτική εκτέλεση μιας εντολής κίνησης.

1.4 Δομή βιομηχανικών ρομπότ

Ένα βιομηχανικό ρομπότ μιμείται το ανθρώπινο χέρι. Επειδή τα βιομηχανικά ρομπότ εκτελούν διάφορους χειρισμούς, ονομάζονται και μηχανικοί χειριστές. Το κύριο σώμα του ρομπότ αποτελείται από μια διαδοχή συνδέσμων ή μελών (links), που συνδέονται με αρθρώσεις (joints). Οι αρθρώσεις ελέγχουν τις κινήσεις των συνδέσμων. Ένα ρομπότ έχει αρθρώσεις ανάλογες αυτών του ανθρώπινου χεριού, δηλαδή τον «ώμο»(shoulder), τον «αγκώνα»(elbow), τον «καρπό»(arm). Ο καρπός μπορεί να περιστρέφεται γύρω από τρεις άξονες, τον διαμήκη, που έχει την κατεύθυνση των δαχτύλων(roll), τον εγκάρσιο, που βρίσκεται στο επίπεδο της παλάμης(pitch), και είναι κάθετος ως προς τα δάκτυλα, και τον κάθετο άξονα, που είναι κάθετος ως προς τους άλλους δύο(yaw), δηλαδή ως προς το επίπεδο της παλάμης.



Εικόνα 1.5 : Yaw, Pitch, Roll στον καρπό

Κάθε μια από τις αρθρώσεις του βραχίονα και του καρπού παρέχει έναν βαθμό ελευθερίας στην κίνηση του τελικού στοιχείου δράσης. Έτσι, ένα ρομπότ με k βαθμούς ελευθερίας περιέχει k αρθρώσεις ή k άξονες κίνησης γενικά. Οι αρθρώσεις οδηγούνται από κινητήρες, έμβολα ή γενικότερα από στοιχεία δράσης, που μπορεί να είναι πνευματικά, υδραυλικά, ηλεκτρικά.

Το βιομηχανικό ρομπότ είναι εφοδιασμένο στο άκρο του με μια διάταξη εργασίας, που ονομάζεται τελικό σημείο δράσης ή εργαλείο (end effector / tool), και είναι ικανό να εκτελεί διάφορες βιομηχανικές εργασίες. Το εργαλείο μπορεί να είναι μια κεφαλή συγκόλλησης, ένα πιστόλι βαφής, ένα μηχανικό εργαλείο ή μια αρπάγη, που ανοιγοκλείνει ανάλογα με την εφαρμογή, στην οποία πρόκειται να χρησιμοποιηθεί το ρομπότ.

Η κίνηση του τελικού στοιχείου δράσης ρυθμίζεται ελέγχοντας τη θέση και την ταχύτητα των αξόνων κίνησης του ρομπότ. Στη ρομπότική ένας άξονας κίνησης ισοδυναμεί με έναν βαθμό ελευθερίας, ως προς τον οποίο μπορεί να κινηθεί το ρομπότ. Για να μπορέσει ένα ρομπότ να «φτάσει» ένα αυθαίρετο σημείο μέσα στην περιοχή δράσης του με έναν επιθυμητό προσανατολισμό του εργαλείου, χρειάζεται να έχει έξι άξονες (βαθμούς ελευθερίας) κίνησης. Ακόμη και ένας διαφορετικός προσανατολισμός του εργαλείου μπορεί να αλλάξει εντελώς τη θέση του βραχίονα του ρομπότ.

Οι άξονες κίνησης ενός ρομποτικού βραχίονα μπορεί να είναι άξονες στροφικής κίνησης (περιστροφικές αρθρώσεις), ή γραμμικής μεταφορικής κίνησης (πρισματικές αρθρώσεις). Ένας στροφικός άξονας οδηγείται άμεσα σε έναν ηλεκτρικό κινητήρα ή έμμεσα από ένα σύστημα αλυσίδας ή οδοντωτών

τροχών. Η κίνηση κατά μήκος ενός γραμμικού άξονα πραγματοποιείται τυπικά από ένα πρισματικό ζεύγος ή μέσω μιας κοχλιωτής σύνδεσης. Ένα πρισματικό ζεύγος διαθέτει ένα υδραυλικό ή πνευματικό έμβολο, ενώ ένας κοχλίας μετατρέπει την περιστροφική κίνηση ενός ηλεκτρικού κινητήρα σε γραμμική κίνηση κατά μήκος του αντίστοιχου άξονα του βραχίονα.

Ο υπολογιστής ενός σύγχρονου ρομπότ περιέχει ένα πρόγραμμα ελέγχου και ένα πρόγραμμα εργασίας. Το πρόγραμμα ελέγχου δίνεται από τον κατασκευαστή και ρυθμίζει την κίνηση κάθε άρθρωσης του ρομπότ. Το πρόγραμμα εργασίας δίνεται από τον χρήστη και καθορίζει τις κινήσεις, που χρειάζονται για να εκτελεστεί κάθε φορά η επιθυμητή εργασία. Ένα πρόγραμμα εργασίας μπορεί να παραχθεί είτε περνώντας το ρομπότ από τις επιθυμητές θέσεις(μέσω διδασκαλίας - teaching), είτε χρησιμοποιώντας τις κατάλληλες γλώσσες προγραμματισμού. Όταν χρησιμοποιείται μια γλώσσα προγραμματισμού, το ρομπότ περιέχει τον κατάλληλο επεξεργαστή, που μεταφράζει το πρόγραμμα εργασίας και παρέχει τα δεδομένα, που χρειάζεται το πρόγραμμα ελέγχου για να καθοδηγήσει το ρομπότ στις επιθυμητές κινήσεις.

1.5 Βιομηχανικά ρομπότ και Ασφάλεια

Στα σύγχρονα συστήματα παραγωγής, τα τελευταία χρόνια έχει εμφανισετεί η ανάγκη για συνεργασία Ανθρώπου-Βιομηχανικού Ρομπότ(A-P) σε κοινό χώρο εργασίας, για την από κοινού εκτέλεση βιομηχανικών καθηκόντων ή κατεργασιών. Ο σκοπός της συνεργασίας και της συνύπαρξης A-P κατά την εκτέλεση καθηκόντων, είναι η βελτίωση της ποιότητας και η αύξηση της παραγωγικότητας, εμπλουτίζοντας τη γνώση, τις δεξιότητες και τις κιναισθητικές δυνατότητες του εργαζομένου με την ακρίβεια, την δύναμη και την επαναληψιμότητα των ρομπότ. [21]

Καθώς εμφανίζονται νέες ανάγκες για υβριδικά ρομποτικά συστήματα παραγωγής με συνεργασία A-P και η τεχνολογία και οι κανονισμοί εξελίσσονται, η φυσική περίφραξη και η χωρική απομόνωση που είθισται σήμερα μεταξύ A-P, έρχεται να δώσει τη θέση της σε διαφορετικά συστήματα και τεχνικές ασφαλείας, τα οποία θα επιτρέψουν στους εργαζόμενους να αλληλεπιδρούν και να συνεργάζονται με τα βιομηχανικά ρομπότ. Εν τούτοις,

παρά το ότι η συνεργασία A-P φαίνεται να είναι μια ισορροπημένη λύση μεταξύ παραγωγικότητας, ποιότητας, ευελιξίας και κόστους, τα ζητήματα ασφαλείας που προκύπτουν, κρατούν ακόμα την εφαρμογή της συνεργασίας A-P ουσιαστικά ανεκμετάλλευτη στη βιομηχανία. [21]

Εικόνα 1.6 : Παράδειγμα συνεργασίας Ανθρώπου Ρομπότ στη συναρμολόγηση (Audi)

Η εγγύτητα και ο διαμοιρασμός του κοινού χώρου εργασίας μεταξύ του ανθρώπου και του κινούμενου ρομπότ είναι η γενεσιουργός αιτία των περισσότερων ζητημάτων ασφάλειας. Τα κρισιμότερα ζητήματα από πλευράς κινδύνου, είναι, αφενός, ο κινούμενος ρομποτικός βραχίονας και, αφετέρου, η αντίληψη και η αίσθηση των κινήσεων του βραχίονα από τον άνθρωπο. Οι τρέχουσες τεχνικές ασφαλούς συνεργασίας A-P συνδυάζουν φράκτες υπέρυθρων ακτίνων, αισθητήρες, σαρωτές λέιζερ, ρομποτική όραση με κάμερες, κουμπιά πανικού(emergency stop) και συναγερμούς, με σκοπό είτε την επιβράδυνση του ρομπότ σε ασφαλή ταχύτητα είτε το σταμάτημα της κίνησης του ρομπότ μέχρις ότου είναι ασφαλής ο άνθρωπος. Παρά ταύτα, αυτές οι τεχνικές μπορούν να σταματήσουν ολόκληρη την παραγωγή, όταν πρόκειται για μια γραμμή παραγωγής, ή να ακυρώσουν την ουσιαστική συνεργασία και να καθυστερήσει την παραγωγή, όταν πρόκειται για ένα μικρότερο κύτταρο παραγωγής. [21]

Ανατρέχοντας κανείς στην εφαρμογή των ρομπότ στη βιομηχανία, μπορεί να βρει πολλά παραδείγματα όπου το ρομπότ εργάζεται ως ανεξάρτητο στο δικό του προστατευμένο χώρο, συχνά αποκαλούμενο «κλουβί» (cage). Η λειτουργία του διακόπτεται ακαριαία στην περίπτωση που κάποιος άνθρωπος ανοίξει την πόρτα για να εισέλθει στο χώρο εργασίας του ρομπότ. Η

προσέγγιση αυτή αποτελεί επαρκή λύση στην περίπτωση που το ρομπότ εκτελεί αυτοτελείς εργασίες, οι οποίες συνήθως είναι διαρκώς επαναλαμβανόμενες αποκαλούμενες και εργασίες χαμηλού επιπέδου (low-level task), όπως π.χ. pick & place. Τέτοια ελεγχόμενα περιβάλλοντα, εξασφαλίζουν ότι το ρομπότ «γνωρίζει» την σχετική κατάσταση του περιβάλλοντος του με σχετική βεβαιότητα. Οι εργασίες που εκτελεί το ρομπότ είναι προσεκτικά ρυθμισμένες και αυστηρά καθορισμένες, ενώ το περιβάλλον επιβάλλεται να είναι δομημένο, δηλαδή ελεγχόμενο (π.χ. ως προς τις συνθήκες φωτισμού) και στατικό κατά κύριο λόγο, προκειμένου να ενδείκνυται για αναγνώριση των χαρακτηριστικών του περιβάλλοντος με χρήση αισθητήρων (sensors) από το ρομπότ κατά τις εργασίες που αυτό απαιτείται. [24]

Κατά τη διαρκή εξέλιξη της επιστήμης της ρομποτικής, έχει εμφανιστεί η ανάγκη για συνεργασία μεταξύ ανθρώπου και ρομπότ. Αναπτύσσονται έτσι τρόποι συνεργασίας που μπορούν να συνδυάσουν τα χαρακτηριστικά στα οποία τα ρομπότ υπερτερούν, όπως είναι για παράδειγμα η μεγάλη δύναμη, η υψηλή ακρίβεια, και η ικανότητα να επαναλαμβάνουν αέναα πανομοιότυπες εργασίες, χωρίς κούραση και υποβάθμιση της ποιότητας, με τα ανθρώπινα χαρακτηριστικά που υπερτερούν, όπως είναι η ευκολία στην αντίληψη του περιβάλλοντος, και η αμεσότητα στην λήψη αποφάσεων. Το αποτέλεσμα στις περιπτώσεις αυτές εισάγει νέες ικανότητες στη συνεργασία ανθρώπου-ρομπότ, οι οποίες ενσωματώνονται με ταχείς ρυθμούς τόσο σε βιομηχανικά ρομπότ (industrial robots), όσο και σε ρομπότ «υπηρεσιών» (service robots). Η ανθρώπινη όμως παρουσία στο χώρο εργασίας του ρομπότ, εισάγει επίσης νέες προκλήσεις καθώς καθιστά το περιβάλλον πλέον δυναμικό και μη δομημένο, διότι χαρακτηρίζεται από διαρκή μετακίνηση, τροποποίηση δηλαδή του περιβάλλοντος, η οποία δεν έχει προκληθεί από το ρομπότ. Η αλληλεπίδραση επίσης του ρομπότ με τον άνθρωπο, προϋποθέτει το ρομπότ να λειτουργεί υπό συνθήκες πραγματικού χρόνου, θέτοντας όρια στη διάρκεια των αριθμητικών υπολογισμών (real-time constraints), ενώ ενισχύεται σημαντικά η ανάγκη για ικανότητα του να αντιλαμβάνεται το περιβάλλον του μέσω αισθητήρων. [24]

Η αντιμετώπιση εργασιών που απαιτούν χερισμό (manipulation), σαν τις συνεργατικές διεργασίες και το αποτέλεσμα που παράγεται από τη συνεργασία αυτή, θα ήταν αδύνατο να παραχθεί αποκλειστικά από τον άνθρωπο ή αποκλειστικά από το ρομπότ. Παρ' όλα αυτά, τουλάχιστον για το άμεσο μέλλον, τα ρομπότ που δρουν σε ανθρώπινο περιβάλλον, θα εξαρτώνται διαρκώς από τον άνθρωπο. Όσο όμως η αναγκαιότητα χρήσης των ρομπότ σε κάποιες εργασίες παραμένει συγκριτικά μεγαλύτερη από το «κόστος» που

απαιτεί η υποστήριξη της λειτουργίας τους από τον άνθρωπο, η πλήρης αυτονομία των ρομπότ, σε πολλές περιπτώσεις καθίσταται αχρείαστη. [24]

Γίνεται λοιπόν αντιληπτό ότι η αναγκαία συνθήκη για την εξασφάλιση της ασφαλούς συνεργασίας του ανθρώπου με το ρομπότ στον ίδιο χώρο, και κατ' επέκταση της ασφαλούς συνεργασίας τους, είναι η ενσωμάτωση δυνατοτήτων αναγνώρισης συγκρούσεων μεταξύ τους και η αντίστοιχη έγκαιρη αντίδραση από τη μεριά του ρομπότ, ώστε να διαφυλαχθεί η ασφάλεια του ρομπότ. Τακτικές εκδόσεις από τον Διεθνή Οργανισμό Προτύπων (ISO) θέτουν πολύ αυστηρούς περιορισμούς σε αυτή τη συνύπαρξη, και η ανάπτυξη νέων μεθόδων και τεχνολογιών είναι απαραίτητη για την ικανοποίηση τους. Οι συνεργατικές διεργασίες ανθρώπου-ρομπότ που εκτελούνται στην βιομηχανία, ενδέχεται από τη φύση τους να ενέχουν υψηλό κίνδυνο από τον χειριστή καθώς συχνά περιλαμβάνουν ανύψωση βαρέων φορτίων, μετακίνηση σε υψηλές ταχύτητες ή μηχανουργικές κατεργασίες με χρήση επικίνδυνων εργαλείων. Με δεδομένο ότι το αποτέλεσμα αυτής της συνεργασίας αναβαθμίζει ιδιαίτερα την παραγωγή, το ζήτημα της ασφάλειας του «χειριστή-συνεργάτη» του ρομπότ κατά τη διάρκεια αυτών των εργασιών θα πρέπει να λαμβάνεται σοβαρά υπόψη. [24]

Επιπλέον στόχος, ωστόσο, είναι η διασφάλιση της ομαλής και απρόσκοπης λειτουργίας του ρομπότ κατά την συνεργασία. Οι μέθοδοι που χρησιμοποιούνται, λοιπόν, θα πρέπει να διακρίνουν τις συγκρούσεις από τις επιθυμητές επαφές που προκύπτουν κατά τη συνεργασία ανθρώπου-ρομπότ. Η σημαντικότητα αυτής της μελέτης , μπορεί να περιγραφεί από το γεγονός ότι ο αριθμός των μεθόδων που εξετάζουν τα παραπάνω χαρακτηριστικά είναι πολύ μικρός, με αποτέλεσμα να περιορίζεται ιδιαίτερα η ανάπτυξη του συγκεκριμένου αντικειμένου. [24]

1.6 Βιβλιογραφική ανασκόπηση

Στη βιβλιογραφία υπάρχει αρκετό υλικό το οποίο αναλύει συγκεκριμένους τομείς που σχετίζονται με τη συνεργασία ανθρώπου-ρομπότ

Οι M. Vasic και A. Billard [28] στο άρθρο τους παρουσιάζουν αρχικά τις 3 κύριες κατηγορίες όσον αφορά τα ατυχήματα που σχετίζονται με την συνεργασία ανθρώπου-ρομπότ όπως είναι τα σχεδιαστικά λάθη, λάθη του

ανθρώπου-χειριστή και οι ελλιπείς συνθήκες περιβάλοντος. Δίνεται βάση στα ατυχήματα που συμβαίνουν στην βιομηχανία και αφορούν τα βιομηχανικά ρομπότ(industrial robots) και τα κινούμενα ρομπότ(mobile robots). Υπάρχουν δύο ειδών ατυχήματα, τα ατυχήματα που εγκλωβίζουν τον άνθρωπο και δεν μπορεί να κινηθεί(pinch injuries) και τα ατυχήματα σύγκρουσης (impact injuries). Επίσης, αναλύθηκαν μέτρα ασφάλειας σύμφωνα με συγκεκριμένα πρότυπα όπως είναι για παράδειγμα το ISO 10218 και το ANSI/RIA R15.06-2012.

Οι T. Kerezovic, G. Sziebig, B. Solvang και T. Latinovic [29] στο άρθρο τους παρουσιάζουν τρόπους ασφαλούς συνεργασίας ανθρώπου-ρομποτ από παλιά ως σήμερα, αναλύουν τις πιο κοινές συσκευές ασφάλειας που χρησιμοποιούνται σε ρομποτικά κύτταρα, όπως είναι ανιχνευτές λέιζερ, συστήματα με κάμερες, ακτίνα φωτός (Laser Beam) και κουρτίνα φωτός (Light Curtain), κάποια πρότυπα που χρησιμοποιούνται στην βιομηχανία που αφορούν τα robotic cells και ένα project που πραγματοποιήθηκε στο Narvik University College, στο οποίο χρησιμοποιήθηκαν τρία ρομπότ, Kuka K30-3, ABB FlexPicker IRB340 και ABB IRB 1500.

Οι K. Eder, C. Harper και U. Leonards [30] στο άρθρο τους εστιάζουν στην διασφάλιση της ασφάλειας μέσω προτύπων όπως είναι το ISO 13482 και το TS 15066. Επίσης, αναλύεται ο προγραμματισμός ρομπότ μέσω επίδειξης και τέλος λαμβάνονται υπόψη οι ψυχολογικές δοκιμασίες που αφορούν την συνεργασία ανθρώπου-ρομπότ σε διάφορα ευέλικτα σενάρια παραγωγής.

Οι S. Haddadin, A. Albu-Schäffer και G. Hirzinger [31] στο άρθρο τους εστίασαν στην απρόσμενη σύγκρουση του ρομπότ με τον άνθρωπο, όταν αυτός, τη στιγμή της σύγκρουσης, παραμένει σταθερός. Μέσω πειραμάτων που διεξήχθησαν σύμφωνα με το πρότυπο EuroNCAP, τα αποτελέσματα έδειξαν ότι δεν προκαλείται σοβαρός τραυματισμός στον άνθρωπο σύμφωνα με συγκεκριμένα πρότυπα που χρησιμοποιήθηκαν.

Οι J. A. Marvel, J. Falco και I. Marstio [32] στο άρθρο τους κάνουν μια περιγραφή της αλληλεπίδρασης ανθρώπου-ρομπότ και γνωστοποιούν μια βάση όσον αφορά την ασφαλή λειτουργία των ρομπότ για συγκεκριμένα καθήκοντα. Επιπλέον, παρουσιάζεται μια οντολογία για καθήκοντα συνεργασίας και περιγράφεται πως η συγκεκριμένη οντολογία χρησιμοποιείται ως μέρος αξιολόγησης και περιορισμού των κινδύνων. Τέλος, αναλύεται μελέτη περίπτωσης (case study) για να διευκρινιστεί η εφαρμογή της οντολογίας που

αφορά την διαδικασία συνεργασίας από την επιλογή των καθηκόντων μέσω περιορισμού των κινδύνων.

Οι Y. H. Weng, C. H. Chen και C. T. Sun [33] στο άρθρο τους τονίζουν ότι η επόμενη γενιά ρομπότ θα έχει σχετική αυτονομία, κάτι το οποίο αυξάνει τον αριθμό των θεμάτων ασφαλείας. Περιγράφουν ένα πλαίσιο για ένα νομικό σύστημα εστιασμένο στα θέματα ασφαλείας σχετικά με την επόμενη γενιά ρομπότ, περιλαμβανομένης της ιδέας για ευφυή ασφάλεια που απευθύνεται στους κινδύνους των ρομπότ (Open-Texture Risk). Τέλος, εκφράζουν τις αμφιβολίες τους σχετικά με το κατά πόσο ένα μοντέλο βασισμένο στους τρεις νόμους της ρομποτικής του Isaac Asimov μπορεί να γίνει ποτέ εφικτό για την δημιουργία ενός τεχνητού ηθικού οργανισμού που θα διασφαλίζει την ασφάλεια μεταξύ ανθρώπου-ρομπότ.

Οι M. A. Goodrich και A. C. Schultz [34] στο άρθρο τους πραγματοποιούν μια έρευνα όσον αφορά τη σύγχρονη αλληλεπίδραση μεταξύ ανθρώπου-ρομπότ. Υποστηρίζουν ότι η αλληλεπίδραση μεταξύ ανθρώπου-ρομπότ είναι ένα νέο πεδίο επιστημονικής έρευνας. Επιπλέον, καθορίζουν το πρόβλημα της αλληλεπίδρασης αυτής με μια ιδιαίτερη έμφαση στους παράγοντες που μπορεί να διαμορφώσει ένας σχεδιαστής. Τέλος, κατηγοριοποιούν τις περιοχές εφαρμογής σε δύο γενικές κατηγορίες, απομακρυσμένη και κοντινότερη αλληλεπίδραση.

Οι A. Cherubini, R. Passama, A. Crosnier, A. Lasnier και P. Fraisse [35] στο άρθρο τους συνοψίζουν την τελευταία λέξη της τεχνολογίας όσον αφορά την συνεργασία ανθρώπου-ρομπότ και τονίζουν την συνεισφορά τους στο αντικείμενο της συγκεκριμένης έρευνας. Επίσης, παρουσιάζουν την εφαρμογή της συναρμολόγησης μιας άρθρωσης με σταθερή ταχύτητα (homokinetic joint).

Οι A. De Santis, B. Siciliano, A. De Luca και A. Bicchi [36] στο άρθρο τους παρουσιάζονται θέματα μηχανικά και ελέγχου με έμφαση στις τεχνικές εκείνες που παρέχουν ασφάλεια με ένα ενδογενή τρόπο ή με μέσα ελέγχου των εξαρτημάτων. Ιδιαίτερη προσοχή δίνεται στην ανεξαρτησία, κυρίως λόγω αισθητήρων, αρχιτεκτονικών ελέγχου, και χειρισμό λαθών και ανοχής. Τέλος, παρέχονται προτάσεις για αξιολόγηση της ασφάλειας και ανεξαρτησίας στο θέμα της συνεργασίας ανθρώπου-ρομπότ.

Οι D. Kulic και E. Croft [37], στο άρθρο τους, παρουσιάζουν μέθοδο επίτευξης ασφάλειας στην αλληλεπίδραση ανθρώπου-ρομπότ μέσω

προγραμματισμού και ελέγχου. Η μέθοδος είναι βασισμένη σε μια κατηγορηματική ποσοτικοποίηση με βάση το βαθμό του κινδύνου στην αλληλεπίδραση. Προτείνονται διάφοροι αλγόριθμοι προγραμματισμού και ελέγχου για ελαχιστοποίηση του εκτιμώμενου κινδύνου. Επίσης, προτάθηκε ένα σύστημα παρακολούθησης του ανθρώπου για την βελτίωση της ασφάλειας της αλληλεπίδρασης μέσω της χρήσης εικονικής και πραγματικής φυσιολογικής πληροφορίας. Τέλος, αναπτύχθηκε μια μεθοδολογία για ομαλή ενσωμάτωση ασφαλών στρατηγικών για διαφορετικά εύρη τροχιάς.

Οι W. Shackleford, G. Cheok, T. Hong, K. Saidi και M. Shneier [38] στο άρθρο τους παρουσιάζουν την δουλειά τους πάνω στην παρακολούθηση και στον εντοπισμό ανθρώπου και αντικειμένου. Επίσης, γίνεται μια αναφορά στο πως μεταφέρεται η πληροφορία σε ένα κοινό σύστημα συντεταγμένων για σύγκριση αποτελεσμάτων. Τέλος, περιγράφονται οι μετρήσεις που ελήφθησαν καθώς και τα αποτελέσματά τους.

Οι P. Zhang, P. Jin, G. Du, X. Liu [39] στο άρθρο τους τονίζουν την δυνατότητα της κάμερας Kinect να εντοπίσει τον άνθρωπο σε πραγματικό χρόνο. Χρησιμοποιείται το φίλτρο Interval Kalman για την ομαλή μεταφορά των δεδομένων από την Kinect. Στη συνέχεια, προβλέπουν τις μελλοντικές κινήσεις του ανθρώπου μέσω του Gaussian Mixture Model (GMM), και διευρύνουν τον όγκο των ορίων (bounding volume) του ανθρώπου ως ασφαλείς περιοχές δευτέρου επιπέδου σε συγκεκριμένο χρόνο με σκοπό την πιθανή διόρθωση της πρόβλεψης.

Οι A. Bicchi, M. A. Peshkin και J. E. Colgate [40] παρουσιάζουν διαφορετικές προσεγγίσεις για να πετύχουν την καλύτερη απόδοση, δεδομένης της ασφάλειας, που παρέχεται όσο γίνεται εκτέλεση καθηκόντων από το ρομπότ. Επίσης, αναφέρουν μια λίστα από έξυπνες βοηθητικές συσκευές (Intelligent Assist Devices), οι οποίες ξεπερνούν τις συμβατικές αντιλήψεις της ασφάλειας από τα ρομπότ με σκοπό την προστασία του ανθρώπου. Στη συνέχεια, τονίζουν την αναγκαιότητα βελτίωσης των προτύπων ασφάλειας και αξιοπιστίας όσον αφορά την χρήση των ρομπότ στην βιομηχανία. Τέλος, αναλύονται οι προκλήσεις για την ανάπτυξη νέων ρομποτικών συστημάτων για ασφαλή και αποτελεσματική συνεργασία ανθρώπου-ρομπότ.

Οι C. A. Cordero, G. Carbone, M. Ceccarelli, J. Echavarri και J. L. Munoz [41] στο άρθρο τους περιγράφουν μια πειραματική διαδικασία, που αποτελείται από πειράματα σύγκρουσης του ανθρώπινου κεφαλιού με ένα

βιομηχανικό ρομπότ με σκοπό την επαλήθευση του δείκτη ασφάλειας New Index for Robots (NIR), καθώς και τα αποτελέσματα των πειραμάτων.

1.9 Στόχος και οργάνωση της διπλωματικής εργασίας

Η εργασία αυτή έχει ως στόχο τον προγραμματισμό του ρομποτικού βραχίονα Staubli RX90L με σκοπό καταρχάς την αποφυγή του ανθρώπου όσο εκείνος βρίσκεται σε κοντινή απόσταση από τον ρομποτικό βραχίονα και δεύτερον την ασφαλή συνεργασία ρομποτικού βραχίονα–ανθρώπου με διακοπή της λειτουργίας του ρομποτικού βραχίονα, μέχρις ότου ο άνθρωπος να εξέλθει από τον χώρο εργασίας του ρομποτικού βραχίονα.

Το κείμενο της διπλωματικής εργασίας χωρίζεται σε τρία κύρια μέρη-θεματικές ενότητες :

- Στο **πρώτο μέρος**, επιχειρήθηκε μια ουσιαστική εισαγωγή στις βασικές έννοιες που χαρακτηρίζουν ένα ρομπότ, στα χαρακτηριστικά και τη δομή ενός ρομπότ, και στο ρόλο που παίζει η ασφάλεια στα σύγχρονα συστήματα παραγωγής. Επίσης έγινε μια σύντομη ιστορική αναδρομή στην Ρομποτική και σχετική βιβλιογραφική ανασκόπηση.
- Στο **δεύτερο μέρος**, αναλύεται η υποδομή της παρούσας διπλωματικής εργασίας. Στο $2^{ο}$ κεφάλαιο γίνεται εμβάθυνση στο συγκεκριμένο ρομποτικό βραχίονα Staubli RX90L και στη γλώσσα προγραμματισμού που χρησιμοποιεί. Στο $3^{ο}$ κεφάλαιο γίνεται μια μικρή εισαγωγή στην Εικονική Πραγματικότητα και στο λογισμικό Unity3d. Στο ίδιο κεφάλαιο γίνεται μια σύντομη αναφορά στην Microsoft Kinect και γενικότερα στην ανάγκη για χρησιμοποίηση της όρασης μέσω κάμερας. Τέλος, στο $4^{ο}$ κεφάλαιο γίνεται εισαγωγή στην πλατφόρμα μικροεπεξεργαστή-ελεγκτή ArduinoMega που χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία.
- Το **τρίτο μέρος** αναφέρεται στο πρακτικό τμήμα της διπλωματικής εργασίας, δηλαδή στην επικοινωνία του Unity3d με την πλατφόρμα ArduinoMega και στη συνέχεια την επικοινωνία αυτής με τον ελεγκτή του ρομπότ, ούτως ώστε μέσω προγραμματισμού του ρομπότ να υπάρχει ασφαλής κίνηση του ρομποτικού βραχίονα όταν βρίσκεται στο

χώρο εργασίας του ρομπότ ο άνθρωπος. Στο 5^{ο}, 6^{ο},και 7^{ο} κεφάλαιο αναλύονται οι τρεις κυρίως εφαρμογές: στην πρώτη γίνεται απλή αποφυγή του ανθρώπου από το ρομπότ, στην δεύτερη γίνεται η σύνθετη αποφυγή όταν ο άνθρωπος βρίσκεται στη νέα περιοχή όπου κινείται το ρομπότ λόγω αρχικής αποφυγής του ανθρώπου, και στην τρίτη ακινητοποιείται ο ρομποτικός βραχίονας όσο ο άνθρωπος παραμένει μέσα στον χώρο εργασίας του ρομπότ. Τέλος, στο 8^{ο} κεφάλαιο αναλύονται κάποιοι κώδικες για την παραλαβή συντεταγμένων του ανθρώπου και του ρομποτικού βραχίονα από την πλακέτα ArduinoMega.

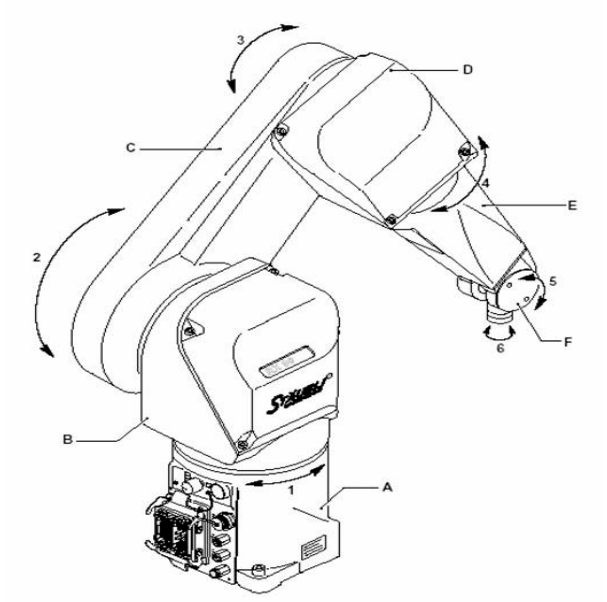
2

Ρομπότ Staubli RX90L

2.1 Περιγραφή

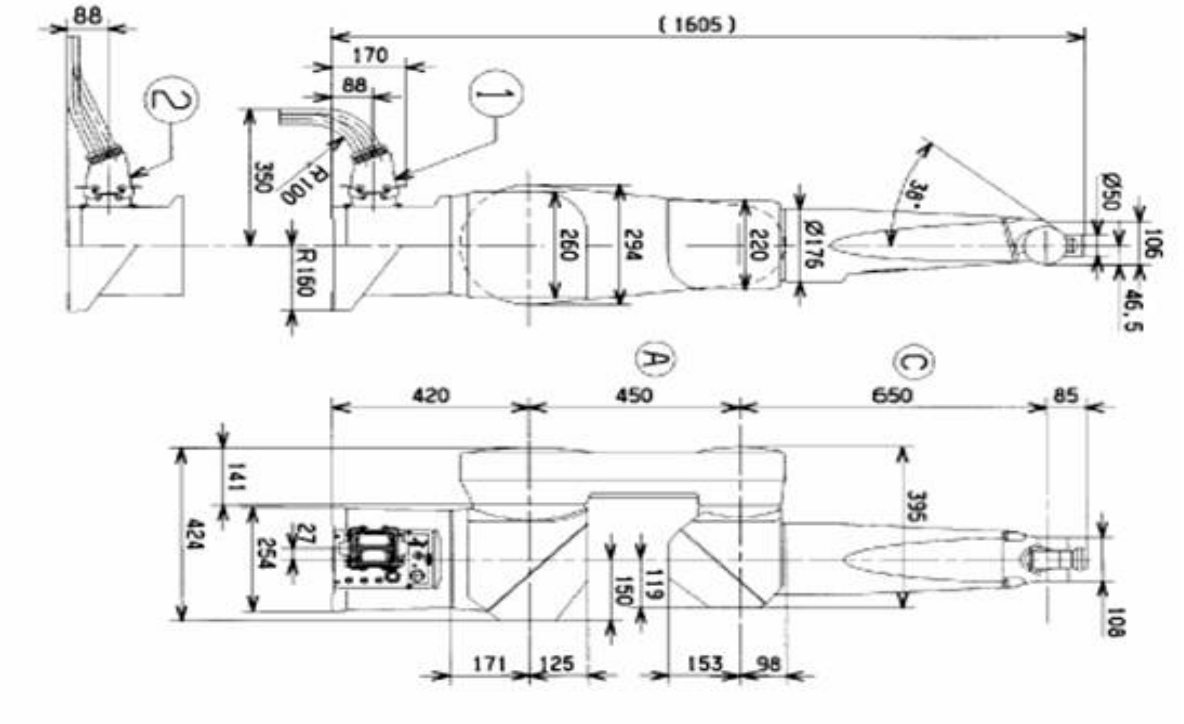
Ο βραχίονας αποτελείται από τμήματα ή μέλη διασυνδεμένα με αρθρώσεις(**Εικόνα 2.1**). Κάθε άρθρωση περιλαμβάνει έναν άξονα γύρω από τον οποίο περιστρέφονται δύο μέλη. Η κίνηση των αρθρώσεων του ρομπότ παράγεται από κινητήρες χωρίς ψήκτρες συζευγμένους με γωνιοαναλυτές. Ο ρομποτικός βραχίονας έχει επαρκή ευελιξία και είναι δυνατό να εκτελέσει ποικιλία εφαρμογών. [22]

Τα κύρια στοιχεία του ρομποτικού βραχίονα είναι : **βάση- base (A), ώμος-shoulder (B), βραχίονας-arm (C), αγκώνας-elbow (D), πήχυς-forearm (E) και καρπός-wrist (F)** και είναι εμφανή στην **Εικόνα 2.1**. [22]



Εικόνα 2.1 : Staubli RX90L

Η συναρμολόγηση του ρομπότ εμπεριέχει τα μοτέρ, τα φρένα, μηχανισμούς μετάδοσης κίνησης, δεσμίδες καλωδίων, πνευματικά και ηλεκτρικά κυκλώματα για τον χρήστη, καθώς επίσης και το σύστημα αντιστάθμισης. Στην **Εικόνα 2.2** παρατηρούμε τις διαστάσεις του ρομποτικού βραχίονα Staubli RX90L.

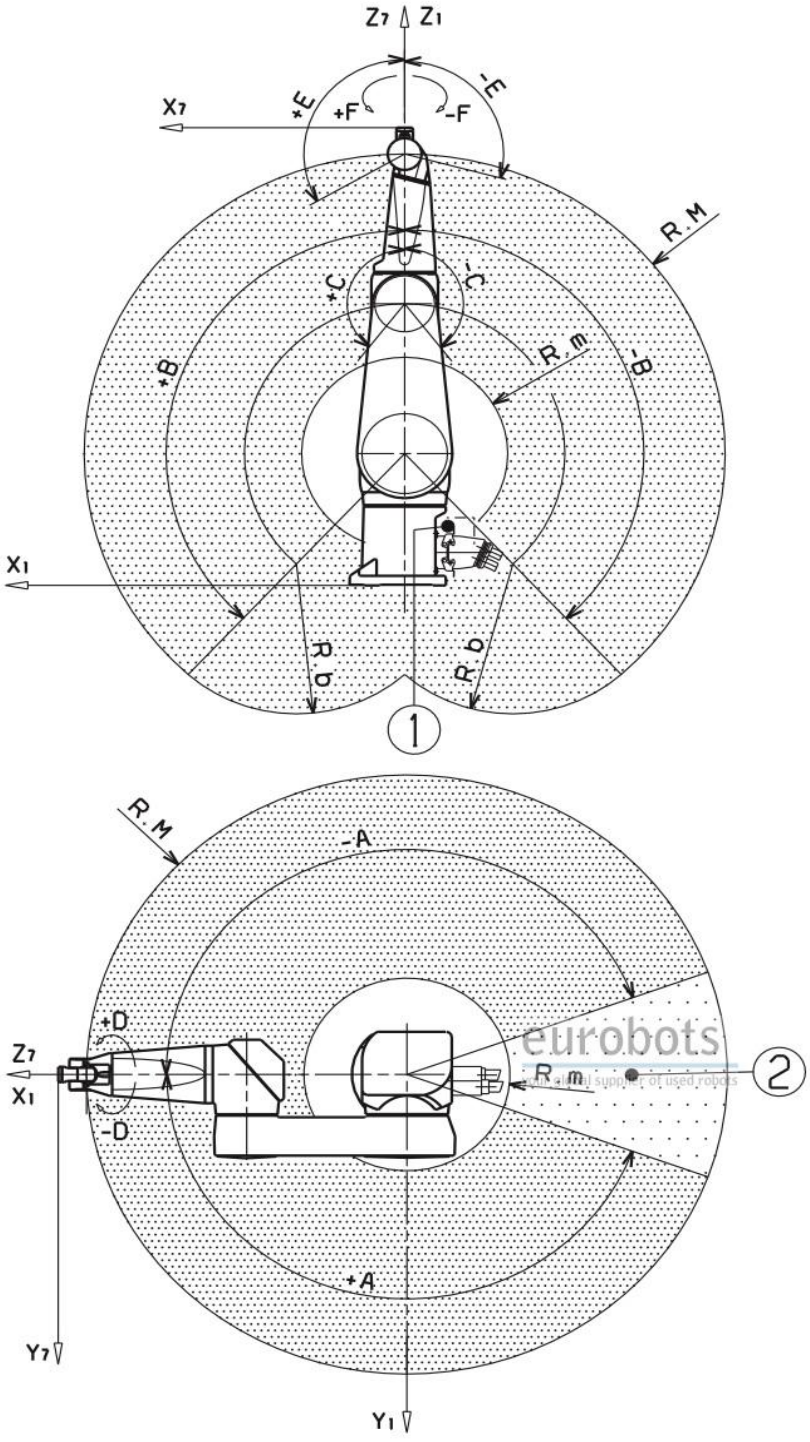


Εικόνα 2.2 : Διαστάσεις Staubli RX90L

2.2 Γενικά Χαρακτηριστικά

2.2.1 Συνθήκες Εργασίας

- Θερμοκρασία Περιβάλλοντος: +5° C έως +40° C
- Υγρασία: 30% έως 95%
- Υψόμετρο: έως 2000 μέτρα



Εικόνα 2.3 : Χώρος Εργασίας Staubli RX90L

2.2.2 Βάρος

Το βάρος του ρομποτικού βραχίονα Staubli RX90L είναι 113 κιλά.

2.3 Απόδοση

- R.M μέγιστη απόσταση μεταξύ άρθρωσης 2 και 5 : 1100 mm
- R.m ελάχιστη απόσταση μεταξύ άρθρωσης 2 και 5 : 401 mm
- R.b απόσταση μεταξύ αρθρώσεων 3 και 5 : 650 mm
- Μέγιστη ταχύτητα εργασίας : 12.6 m/s, 12.2[(1)] m/s
- Επαναληψιμότητα σε σταθερή θερμοκρασία : ± 0.025 mm

R.M, R.m και R.b φαίνονται στην **Εικόνα 2.3**.

2.3.1 Όρια Ροπής, Εύρος Γωνίας, Ταχύτητα και Γωνιακή Ανάλυση

Στον παρακάτω Πίνακα(**Πίνακας 2.1**) αναλύονται τα χαρακτηριστικά του ρομποτικού βραχίονα Staubli RX90L.

Χαρακτηριστικά						
Άρθρωση	1	2	3	4	5	6
Εύρος Γωνίας(^{0})	320	275	285	540	225	540
Εύρος Εργασίας(^{0})	±160	±137.5	±142.5	±270	+120 − 105	±270
Ονομαστική Ταχύτητα(^{0}/s)	236 177[(1)]	200 150[(1)]	286 143[(1)]	401 201[(1)]	320 160[(1)]	580 390[(1)]
Μέγιστη Ταχύτητα(^{0}/s)	356	356	296	409	480 288[(1)]	1125[(2)]
Γωνιακή Ανάλυση(^{0}.10^{-3})	0.87	0.87	0.72	1	1.95	2.75
Στατική Ροπή(Nm)	–	–	-	-	26[(3)] 16[(4)]	10
Μέγιστη Ροπή(Nm)	–	–	-	–	100[(3)] 57[(4)]	43

Πίνακας 2.1: Χαρακτηριστικά Staubli RX90L[1]

Ελάχιστη Ταχύτητα για χειροκίνητο έλεγχο του χειριστηρίου (Εικόνα 2.4):

- Καρτεσιανή Λειτουργία: 250 mm/s
- Περιστροφική Λειτουργία: 10% της ονομαστικής ταχύτητας
- Μέγιστη Καρτεσιανή Ταχύτητα: 1.5 m/s

[1] Μέγιστο φορτίο [2] Χωρίς αλληλεπίδραση της 5ης άρθρωσης [3] Εάν η ροπή της 6ης άρθρωσης είναι μηδενική [4] Για μέγιστη ροπή στην 6η άρθρωση

Εικόνα 2.4: Χειριστήριο Staubli RX90L

2.3.2 Ωφέλιμο Φορτίο

Παρακάτω αναλύεται στους (**Πίνακας 2.2, Πίνακας 2.3**) το ωφέλιμο φορτίο για τον ρομποτικό βραχίονα Staubli RX90L για δύο περιπτώσεις.

Περίπτωση (1)

Χαρακτηριστικά Φορτίου:

Load center of gravity position M : z = 150 mm from centerline of joint 5 and x= 75 mm from centerline of joint 6 (**Εικόνα 2.2**).

Ωφέλιμο Φορτίο (1)	
Ονομαστική Ταχύτητα	$3.5kg$
Μειωμένη Ταχύτητα	6kg

Πίνακας 2.2: Ωφέλιμο Φορτίο Περίπτωση (1)

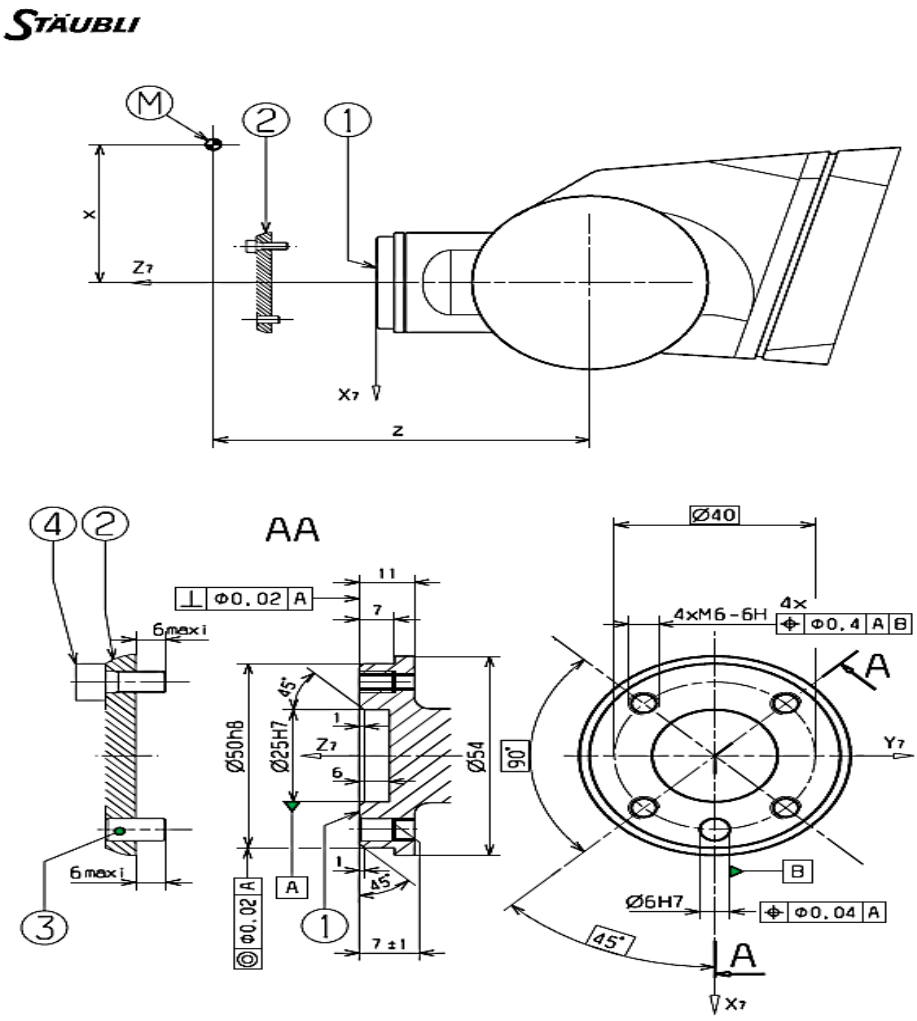
Περίπτωση (2)

Χαρακτηριστικά Φορτίου:

Load center of gravity position M : z = 150 mm from centerline of joint 5 and y = 50 mm from centerline of joint 6 (**Εικόνα 2.5**).

Ωφέλιμο Φορτίο (2)	
Ονομαστική Ταχύτητα	$8kg$

Πίνακας 2.3: Ωφέλιμο Φορτίο Περίπτωση (2)



Εικόνα 2.5: Κατασκευαστικό Σχέδιο 5ης, 6ης άρθρωσης

2.4 Ελεγκτής CS7

2.4.1 Γενική Περιγραφή

Η καμπίνα του ελεγκτή αποτελείται από έναν ελεγκτή, το έξυπνο μέρος της εγκατάστασης, ο οποίος ελέγχει το ρομπότ μέσω ενισχυτών ισχύος που επικοινωνούν με κάθε άρθρωση του βραχίονα. Η παρεμβολή που χρησιμοποείται είναι γραμμική. Η ηλεκτρική ενέργεια μετατρέπεται από ένα μπλοκ τροφοδοσίας, το οποίο τροφοδοτεί σε καθένα από τα παραπάνω στοιχεία τις κατάλληλες τάσεις που απαιτούνται για τη σωστή λειτουργία ξεκινώντας από τριφασική τάση που παρέχεται από το ηλεκτρικό δίκτυο. Τα καλώδια απαραίτητα για την ασφάλεια που αφορά τον ηλεκτρισμό τοποθετούνται στην «πλακέτα ασφαλείας». Όλα αυτά τα στοιχεία λειτουργούν σε κλειστό χώρο, και ο έλεγχος της θερμοκρασίας εξασφαλίζεται από έναν εναλλάκτη θερμότητας αέρα προς αέρα. Δύο μονάδες ανεμιστήρα ψύχουν τα κυκλώματα του ελεγκτή.

Εικόνα 2.6: Το εσωτερικό του ελεγκτή CS7

Ο ελεγκτής έχει κατασκευαστεί με αρχιτεκτονική VME (Versa Module Europe bus) και περιλαμβάνει:

- Μονάδα επεξεργασίας τους συστήματος (CPU 030) για έλεγχο
- Περιφερειακή μονάδα διαχείρησης (SIO), η οποία ελέγχει τον σκληρό δίσκο, τα σήματα εισόδου/εξόδου, τις σειριακές συνδέσεις γενικού σκοπού καθώς και την επικοινωνία με το χειριστήριο του ελεγκτή και τον πίνακα ελέγχου
- Μονάδα κοινής διεπαφής (MI6), με ενισχυτές ελέγχου των κινητήρων
- Μονάδα απόλυτης βαθμονόμησης (ACB), όταν ενεργοποείται το ρομπότ

  Η πλακέτα ασφαλείας περιλαμβάνει όλα τα σήματα και τα εξαρτήματα που διασφαλίζουν το τρέξιμο της εγκατάστασης κάτω από κατάλληλες συνθήκες ασφαλείας.

Τα χαρακτηριστικά είναι τα ακόλουθα:

- Ελεγκτής τύπου CS7 ελεγχόμενος από έναν 68030 μικροεπεξεργαστή (40 MHZ) και έναν 68882 συνεπεξεργαστή (33 MHZ)
- VMEbus (Versa Module Europe bus)
- 4 Mbyte RAM
- Σκληρός δίσκος των 85 Mbytes το ελάχιστο
- 3 1/2¨ σκληροί δίσκοι συμβατοί με υπολογιστή
- 12 είσοδοι/6 έξοδοι από τις οποίες 3 υψηλής ταχύτητας είσοδοι (1ms)
- 4 RS232C/1 RS422/485 σειριακά καλώδια
- 1 πρότυπο τερματικό οθόνης
- Χειριστήριο του ρομποτικού βραχίονα (Teach pendant)
- Καλώδιο διασύνδεσης του ρομποτικού βραχίονα με τον ελεγκτή (5/10 μέτρα)

2.4.2 Παρουσίαση του μπροστινού πάνελ

Ο κύριος σκοπός του μπροστινού πάνελ είναι η ενεργοποίηση ποικίλων λειτουργιών και η ένδειξη της κατάστασης του συστήματος. Το μπροστινό πάνελ φαίνεται στην **Εικόνα 2.7**.

Ενδείξεις :

Power: Υποδεικνύει ότι είναι ενεργοποιημένος ο ελεγκτής.
Arm Power: Υποδεικνύει την αποτελεσματική επιβεβαίωση ότι ο ρομποτικός βραχίονας είναι έτοιμος για λειτουργία.

Arm Power ON/OFF: Υποδεικνύει το σήμα που ενεργοποιεί τον ρομποτικό βραχίονα (για ενεργοποίηση του ρομποτικού βραχίονα, είναι αναγκαίο το πάτημα του κουμπιού όταν η ένδειξη αναβοσβήνει).

Fault: Υποδεικνύει την ενεργοποίηση του κυκλώματος έκτακτης ανάγκης, το οποίο μπορεί να προκληθεί από τα ακόλουθα:

- Κουμπί έκτακτης ανάγκης (μπροστινό πάνελ),
- Κουμπί έκτακτης ανάγκης(χειριστήριο ρομποτικού βραχίονα),
- Εξωτερική ένδειξη έκτακτης ανάγκης από την πλακέτα ασφαλείας,
- Όταν μια άρθρωση δεν μπορεί να φτάσει σε συγκεκριμένο σημείο στο χώρο,
- Επιλογή φρένων.

Εικόνα 2.7: Μπροστινό πάνελ του ελεγκτή

Prog Run: Υποδεικνύει ότι ο ελεγκτής τρέχει ένα πρόγραμμα.

Prog Start: Υποδεικνύει το σήμα για να τρέξει το πρόγραμμα όταν χρησιμοποιείται το χειριστήριο του ρομποτικού βραχίονα.

Κουμπιά :

Arm Power ON/OFF: Ενεργοποιεί τον ρομποτικό βραχίονα όταν η ένδειξη αναβοσβήνει.

Emergency Stop: Σταματά τον ρομποτικό βραχίονα εάν βρίσκεται σε κίνηση και ενεργοποιεί τα φρένα. **Η χρήση του συγκεκριμένου κουμπιού περιορίζεται σε συγκεκριμένες περιπτώσεις έκτακτης ανάγκης.**

2.5 Γλώσσα Προγραμματισμού V+

2.5.1 Περιγραφή

Η V+ είναι ένα σύστημα ελέγχου βασισμένο σε υπολογιστή και μια γλώσσα προγραμματισμού ειδικά σχεδιασμένη για να χρησιμοποιείται στα βιομηχανικά ρομπότ της Adept Technology, σε συστήματα όρασης, καθώς και σε συστήματα ελέγχου κίνησης.

Ως σύστημα πραγματικού χρόνου, ο συνεχής υπολογισμός της τροχιάς από τη V+, επιτρέπει σύνθετες κινήσεις να εκτελούνται άμεσα με αποδοτική χρήση της μνήμης του συστήματος, καθώς και με μείωση της πολυπλοκότητας του συστήματος συνολικά. Το σύστημα V+, δημιουργεί συνέχεια εντολές ελέγχου του ρομπότ και μπορεί ταυτόχρονα να αλληλεπιδρά με ένα χειριστή, επιτρέποντας έτσι την δημιουργία και τροποποίηση προγραμμάτων.

Η V+ παρέχει όλη τη λειτουργικότητα των μοντέρνων υψηλού επιπέδου γλωσσών προγραμματισμού, όπως :

- Η κλήση υπορουτίνων
- Οι δομές ελέγχου
- Περιβάλλον πολλαπλών εργασιών
- Εκτέλεση προγραμμάτων αναδρομικά και με επανείσοδο

2.5.2 Εκτέλεση και Έλεγχος ενός προγράμματος

2.5.2.1 Εκτέλεση ενός προγράματος

Για να ξεκινήσει ένα πρόγραμμα χρησιμοποιείται η ακόλουθη εντολή:

EXECUTE program name, αριθμός επαναλήψεων, γραμμή που θα ξεκινήσει το πρόγραμμα στην 1η επανάληψη,

Π.χ. EX STAUBLI, 3, 4

Θα εκτελεστεί το πρόγραμμα STAUBLI, 3 φορές επιτυχημένα, (ένας αρνητικός αριθμός θα οδηγούσε σε άπειρο αριθμό επαναλήψεων) από την 4η γραμμή του προγράμματος μόνο για την πρώτη επανάληψη.

Το όνομα του προγράμματος πρέπει πάντα να :

- **Ξεκινά από γράμμα και ποτέ από αριθμό**
- **Αποτελείται από τον μέγιστο αριθμό των 15 χαρακτήρων, αλφαβητικά(a-z), ψηφία(0-9), τους χαρακτήρες(-,_) και πιθανόν τελείες(.)**

2.5.2.2 Εμφάνιση στην οθόνη των βημάτων προγράμματος

Σε command mode (.), με την ακόλουθη εντολή, τα βήματα του προγράμματος που εκτελούνται εμφανίζονται στην οθόνη του υπολογιστή :

ENABLE TRACE

Για την επιστροφή σε normal mode :

DISABLE TRACE

2.5.2.3 Διακοπή εκτέλεσης προγράμματος

Υπάρχουν τρεις τρόποι για διακοπή της εκτέλεσης ενός προγράμματος:

- Με το πάτημα του κουμπιού **RUN/HOLD** μέσω του χειριστηρίου του ρομποτικού βραχίονα για την άμεση διακοπή της λειτουργίας του ρομποτικού βραχίονα και του προγράμματος,
- Με την εκτέλεση της εντολής **ABORT** μέσω του πληκρολογίου εφόσον το πρόγραμμα πρόκειται να διακοπεί στο τέλος της εντολής που βρίσκεται σε εκτέλεση,
- Με την εκτέλεση της εντολής **PANIC** μέσω του πληκρολογίου για την άμεση διακοπή του ρομποτικού βραχίονα.

2.5.3 Editor

2.5.3.1 Επεξεργασία προγράμματος

Με τον editor είναι δυνατόν να γίνει:

- Τροποποίηση ενός προγράμματος,
- Εισαγωγή νέων εντολών στο πρόγραμμα,
- Διαγραφή εντολών από το πρόγραμμα,
- Αναζήτηση για συγκεκριμένες εντολές, κ.λ.π.

Υπάρχουν δύο ειδών editor :

- Ένας line editor: **EDIT**
- Ένας full-page editor : **SEE**

SEE prog.name: Δημιουργεί ένα νέο πρόγραμμα στην μνήμη του συστήματος

Υπάρχουν 3 λειτουργίες όταν χρησιμοποιείται ο editor :

- **COMMAND MODE:** Σε αυτή τη λειτουργία βρίσκεται ο χρήστης αμέσως μετά την εκτέλεση της εντολής **SEE prog.name**,
- **INSERT MODE :** Με το πάτημα του κουμπιού **I** στο πληκτρολόγιο ο χρήστης μπαίνει σε αυτήν την λειτουργία, ενώ για να βγει από αυτή ο χρήστης πρέπει να πατήσει το κουμπί **ESC**,

- **REPLACE MODE:** Με το πάτημα του κουμπιού **R** στο πληκτρολόγιο ο χρήστης μπαίνει σε αυτήν την λειτουργία, ενώ για να βγει από αυτή ο χρήστης πρέπει να πατήσει το κουμπί **ESC**.

Στην πρώτη λειτουργία ο χρήστης μπορεί απλά να δει μια λευκή σελίδα εάν το πρόγραμμα δεν υπάρχει στην μνήμη ή να δει το πρόγραμμα εάν αυτό υπάρχει ήδη στην μνήμη, στην δεύτερη λειτουργία ο χρήστης μπορεί να εισαγάγει ότι επιθυμεί μέσα στο πρόγραμμα, ενώ στην τρίτη λειτουργία ο χρήστης μπορεί να αντικαταστήσει τον χαρακτήρα που βρισκεται μετά από τον κέρσορα.

2.5.4 Μεταβλητές χώρου και αποθήκευση στη μνήμη

2.5.4.1 Καρτεσιανιανά σημεία/σημεία ακρίβειας

Ένα **σημείο ακρίβειας** (precision point) είναι ένα σημείο του οποίου οι συντεταγμένες δεν είναι εκφρασμένες σε καρτεσιανές συντεταγμένες (σε χιλιοστά με βάση τους άξονες X, Y, Z, κ.τ.λ.π) αλλά σε μοίρες με βάση την απόλυτη θέση της κάθε άρθρωσης.

Ένα **σημείο ακρίβειας** (precision point) αναγνωρίζεται ως τέτοιο όταν το σύμβολο **#** υπάρχει μπροστά από το όνομα της θέσης.

Π.χ. **#a , a** είναι δύο διαφορετικά σημεία.

Το πρώτο περιγράφει ένα σημείο με συντεταγμένες που αφορούν τις αρθρώσεις (**#a**), ενώ το δεύτερο περιγράφει ένα σημείο με καρτεσιανές συντεταγμένες (**a**).

	Άρθρωση 1	18^o		X	120mm
	Άρθρωση 2	-123^o		Y	12mm
#a	Άρθρωση 3	14^o	**a**	Z	5mm
	Άρθρωση 4	90^o		y	90^o
	Άρθρωση 5	-90^o		p	-90^o
	Άρθρωση 6	0^o		r	0

Εντολή SET

Ανάθεση συντεταγμένων σημείων

Για σημείο ακρίβειας :

SET #a = #ppoint(j1_value, j2_value, j3_value, j4_value, j5_value, j6_value)

Για καρτεσιανό σημείο :

SET a = trans(X_value, Y_value, Z_value, y_value, p_value, r_value)

2.5.4.2 Αποθήκευση στην μνήμη

Παρόλο που μπορεί να έχει γραφτεί ένα πρόγραμμα στον υπολογιστή, για να αποθηκευτεί στην μνήμη RAM του ρομπότ χρειάζεται να εκτελεστεί συγκεκριμένη εντολή. Ο σκληρός δίσκος του ελεγκτή είναι ο '**C**', οπότε όταν ενεργοποιείται ο ελεγκτής του ρομπότ ο χρήστης βρίσκεται στον φάκελο(directory)/σκληρό δίσκο '**C**'.

Παρακάτω αναλύονται οι απαραίτητες εντολές για την εμφάνιση στην οθόνη όλων των φακέλων ή προγραμμάτων που υπάρχουν στον σκληρό δίσκο του ρομπότ, την είσοδο του χρήστη σε έναν φάκελο, την αποθήκευση ενός προγράμματος στη μνήμη RAM του ρομπότ, τη φόρτωση ενός προγράμματος στην μνήμη του συστήματος και τέλος την αντιγραφή, διαγραφή και μετανομασία ενός προγράμματος.

Εντολή FDIRECTORY

Εμφανίζει όλα τα αρχεία που περιέχονται στον σκληρό δίσκο ('**C**'), καθώς και τον διαθέσιμο χώρο που υπάρχει στην μνήμη, όταν η εντολή εκτελείται μόνη της. Για να εμφανιστούν τα αρχεία που υπάρχουν μέσα σε έναν φάκελο, πληκτρολογούμε απλά την εντολή **FDIRECTORY ONOMA_FAKELOU**.

Εντολή DEF D = … OR CD

Αλλάζει τον φάκελο ή σκληρό δίσκο στον οποίο βρίσκεται ο χρήστης.

Π.χ. **DEF D= C:\FAKELOS1\FAKELOS2\etc…**

ή **CD FAKELOS1**

Οπότε ο χρήστης πρέπει να χρησιμοποιήσει ένα path με την εντολή **DEF D=…** είτε την εντολή **CD** για να μετακινηθεί μέσα σε έναν φάκελο. Εδώ φαίνεται και η λειτουργικότητα της πρώτης εντολής έναντι της δεύτερης. Με την εντολή **DEF D=..** ο χρήστης μπορεί να μετακινηθεί στον προηγούμενο φάκελο από εκείνον που βρίσκεται.

Εντολή STORE

Αποθηκεύει όλα τα προγράμματα (ακόμη και αυτά που έχουν φορτωθεί στην μνήμη) και τις σχετικές θέσεις στην μνήμη του ελεγκτή του ρομπότ.

Π.χ. **STORE ARXEIO = PROGRAMMA1,…,PROGRAMMA n**

Εάν πληκτρολογήσουμε μόνο **STORE ARXEIO** τότε όλα τα προγράμματα, οι μεταβλητές και τα υποπρογράμματα θα αποθηκευτούν στον φάκελο **ARXEIO.V2** στην μνήμη RAM του ρομπότ.

Εντολή LOAD

Φορτώνει το συγκεκριμένο αρχείο στην μνήμη RAM του ρομπότ.

Π.χ. **LOAD ARXEIO.V2** (εάν πρόκειται για φάκελο) ή **LOAD ARXEIO.PG** (εάν πρόκειται για πρόγραμμα)

Εντολή FLIST

Εμφανίζει τα περιεχόμενα του συγκεκριμένου φακέλου στην οθόνη του υπολογιστή.

Π.χ. **FLIST c:\ARXEIO.V2**

Εντολή FCOPY

Αντιγράφει ένα υπάρχον αρχείο σε ένα νέο αρχείο.

Π.χ. **FCOPY c:NEOARXEIO.V2 = c:PALAIOARXEIO.V2**

Εντολή FRENAME

Μετανομάζει το όνομα ενός αρχείου.

Π.χ. **FRENAME c:NEO.V2 = PALAIO.V2**

Εντολή FDELETE

Διαγράφει το συγκεκριμένο αρχείο.

Π.χ. **FDELETE ONOMA.ARXEIOY**

2.5.5 Εντολές κίνησης

Εντολή MOVE

Αυτή η εντολή αναγκάζει το ρομπότ να κινηθεί σε συγκεκριμένη θέση μέσω μιας τροχιάς που δεν είναι καθορισμένη από τον χρήστη.

Π.χ. **MOVE ONOMA.THESIS**

Το λογισμικό μπορεί να εκτελέσει, παράλληλα, ένα πρόγραμμα που θα περιλεμβάνει κινήσεις του ρομποτικού βραχίονα καθώς και ένα πρόγραμμα που σχετίζεται με την διαχείρηση εισόδων/εξόδων.

Ακόμη, καθ' όλη την διάρκεια της εκτέλεσης των ποικίλων εντολών κίνησης, το ρομπότ δεν θα περιμένει να εκτελέσει πρώτα τις εντολές κίνησης αλλά μπορεί να αναλύσει και εκτελέσει μαθηματικές πράξεις, να παραλάβει σήματα από τις εισόδους ή να στείλει σήματα στις εξόδους.

Αυτό μπορεί να οδηγήσει, εάν δεν ληφθούν τα κατάλληλα μέτρα, σε ανεξήγητες τροχιές του ρομποτικού βραχίονα.

Π.χ. **MOVE a1**

MOVE a2

x = h + 30

SIG 1

t = x + 60

MOVE b

Οι εντολές x, SIG 1, t θα εκτελεστούν κατά την διάρκεια της εκτέλεσης της κίνησης στο σημείο a2.

Εντολή MOVES

Αυτή η εντολή αναγκάζει το ρομπότ να κινηθεί σε μια τροχιά ευθείας γραμμής. Η τροχιά ευθείας γραμμής δεν είναι πάντα η γρηγορότερη διαδρομή παρόλο

που η πορεία μπορεί να είναι η κοντινότερη (έχει να κάνει με τις διαφορετικές ταχύτητες που μπορεί να αποκτήσει η κάθε άρθρωση).

Εντολή DELAY

Η συγκεκριμένη εντολή σταματά το ρομπότ για ένα συγκεκριμένο χρόνο που δηλώνεται μαζί με την εντολή. Ο χρόνος αυτός μπορεί να δοθεί σε δευτερόλεπτα(second) και δεν μπορεί να είναι μικρότερος από 16ms(millisecond).

Π.χ. **MOVE a**

BREAK

DELAY 180

BREAK

MOVE b

Η εντολή **BREAK** θα αναλυθεί σε παρακάτω υποενότητα(υποενότητα **2.5.8)**.

Εντολη SPEED

Με αυτήν την εντολή μπορεί να μεταβληθεί η ταχύτητα του ρομπότ.

Π.χ. **SPEED τιμή** μονάδες μέτρησης ALWAYS

Υπάρχουν τρεις περιπτώσεις :

1. Εάν δεν έχουν οριστεί οι μονάδες μέτρησης στην εντολή **SPEED**, τότε η τιμή θα θεωρηθεί ως ένα ποσοστό της ονομαστικής ταχύτητας[2] (100%), κάτι το οποίο σημαίνει ότι εάν η ονομαστική ταχύτητα είναι 40% και η ταχύτητα μέσα στο πρόγραμμα έχει οριστεί ως 60%, τότε η τελική ταχύτητα του ρομπότ θα είναι 24% της ονομαστικής.
2. Εάν οι μονάδες που έχουν οριστεί είναι MMPS, τότε η κίνηση εκφράζεται σε χιλιοστά ανά δευτερόλεπτο δεδομένου ότι η ονομαστική ταχύτητα είναι 100%.
3. Εάν οι μονάδες που έχουν οριστεί είναι IPS, τότε η κίνηση εκφράζεται σε ίντσες ανά δευτερόλεπτο δεδομένου ότι η ονομαστική ταχύτητα είναι 100%.

[2] Ονομαστική ταχύτητα ονομάζουμε την ταχύτητα που ελέγχει ο χρήστης από την οθόνη του υπολογιστή όταν δεν εκτελείται κάποιο πρόγραμμα, επομένως υπάρουν δύο ταχύτητες, η ονομαστική και η ταχύτητα που έχει οριστεί μέσα στο πρόγραμμα.

Εάν στο τέλος της εντολής **SPEED** περιλαμβάνεται η λέξη **ALWAYS**, τότε η ταχύτητα θα είναι σταθερή για όλες τις εντολές που ακολουθούν μέχρις ότου δοθεί νέα ταχύτητα. Εάν στο τέλος της εντολής **SPEED** δεν περιλαμβάνεται η λέξη **ALWAYS**, τότε η ταχύτητα θα αφορά μόνο την επόμενη κίνηση του ρομπότ.

Παρατήρηση: Καλό είναι να αποφεχθεί ο ορισμός ονομαστικής ταχύτητας μεγαλύτερης από 100, καθώς έχουν οριστεί τεχνικές προδιαγραφές πάνω σε αυτήν την τιμή. Εάν η τιμή είναι πολύ υψηλή, τότε η πραγματική ταχύτητα του ρομπότ θα μειωθεί από τις φυσικές προδιαγραφές του ρομπότ και της κίνησης που πρόκειται να εκτελεστεί.

Π.χ. **SPEED 100MMPS ALWAYS**

MOVES a

MOVES b

BREAK

SPEED 60MMPS ALWAYS

MOVE d

Η ταχύτητα θα είναι 100 χιλιοστά το δευτερόλεπτο για την κίνηση στα σημεία a και b και 60 χιλιοστά το δευτερόλεπτο για την κίνηση στο σημείο d εάν η ονομαστική ταχύτητα είχε οριστεί εκτός προγράμματος στην τιμή 100.

2.5.6 Ψηφιακές Είσοδοι και Έξοδοι

2.5.6.1 Ψηφιακές Έξοδοι

Υπάρχουν 8 ψηφιακές έξοδοι στον ελεγκτή του ρομποτικού βραχίονα Staubli RX90L. Οι έξοδοι 7 και 8 χρησιμοποιούνται από τις ηλεκτρομαγνητικές βαλβίδες του ρομποτικού βραχίονα.Οι έξοδοι 9-16 έχουν εγκατασταθεί ως οι 8 ψηφιακές έξοδοι του ρομποτικού βραχίονα. Τέλος, η τάση της κάθε ψηφιακής εξόδου είναι 0-24volts.

Εντολή SIGNAL

Με αυτή την εντολή ενεργοποιείται μια εξωτερική ψηφιακή έξοδος. Οι ψηφιακές έξοδοι παραμένουν ως όπως είναι μέχρις ότου δοθεί μια νέα εντολή **SIGNAL** που θα ακυρώσει την αντίστοιχη έξοδο.

Π.χ. **SIG 2, -1, 4**

Οι έξοδοι 2 και 4 θα πάρουν την λογική τιμή **1**, ενώ η έξοδος 1 θα πάρει την λογική τιμή **0**.

Εντολή RESET

Επαναφέρει όλες τις εξόδους στην αρχική τους κατάσταση.

Π.χ. **SIG 1, 5**

RESET

Οι έξοδοι 1, 5(και όλες οι έξοδοι που έχουν πάρει την λογική τιμή **1**) θα πάρουν ταην λογική τιμή **0**.

2.5.6.2 Ψηφιακές Είσοδοι

Ο ελεγκτής του ρομποτικού βραχίονα Staubli RX90L έχει 12 ψηφιακές εισόδους, και αυτές είναι οι 1001-1012. Οι 4 πρώτες ψηφιακές είσοδοι (1001-1004) θεωρούνται είσοδοι υψηλής ταχύτητας (δηλαδή έχουν πιο άμεση απόκριση).

Εντολή FOR…TO…STEP…END

Επαναλαμβάνει ένα σετ εντολών για συγκεκριμένο αριθμό φορών ορισμένο από τον χρήστη.

Π.χ. **FOR t = 10 TO 1 STEP -2**
MOVE a
MOVE b
END

Σε αυτήν την περίπτωση, η κίνηση του ρομπότ στα σημεία a, b θα εκτελεστεί 5 φορές.

Εντολή WHILE…DO

Εκτελεί ένα σετ εντολών όσο η λογική κατάσταση είναι θετική.

Π.χ. **MOVE a**
WHILE (number<10) DO
MOVE b
MOVE c
END
MOVE d

Το ρομπότ θα κινηθεί στο σημείο a, στη συνέχεια θα ελέγξει εάν η μεταβλητή number είναι μικρότερη από τον αριθμό 10, εάν είναι θα κινηθεί στα σημεία b, c επαναληπτικά με έλεγχο κάθε φορά του number<10, ενώ εάν είναι μεγαλύτερη από τον αριθμό 10 τότε το ρομπότ θα κινηθεί στο σημείο d και το πρόγραμμα θα σταματήσει.

Εντολή WAIT…

Αυτή η εντολή θα διακόψει το πρόγραμμα μέχρις ότου η σχετική παράσταση είναι αληθής.

Π.χ. **WAIT (SIG(1001, -1002) OR SIG(1008))**

Σε αυτή την περίπτωση το ρομπότ θα περιμένει είτε το σήμα 1001 να γίνει **θετικό(1)** και το σήμα 1002 να γίνει **αρνητικό(0),** είτε το σήμα 1008 να γίνει **θετικό(1)** ώστε να συνεχίσει στην επόμενη εντολή.

Εντολή CALL( )

Αυτή η εντολή καλεί ένα υποπρόγραμμα. Το υποπρόγραμμα είναι ένα σετ ανεξάρτητων εντολών το οποίο μπορεί να καλέσει ο χρήστης μέσω του κυρίου προγράμματος. Επίσης, επιτρέπει σε ένα πολύπλοκο πρόγραμμα να διαχωριστεί σε αρκετά στοιχειώδη μέρη. Τέλος, κάθε υποπρόγραμμα πρέπει υποχρεωτικά να τερματίζεται με την εντολή **RETURN**.

Π.χ.

ΚΥΡΙΟ ΠΡΟΓΡΑΜΜΑ	ΥΠΟΠΡΟΓΡΑΜΜΑ SP1
MOVE a	**t =(cos(60)*sin(t))/cos(456)**
r =(cos(x)*cos(t))/500	**u=(sin(566)*cos(r))/680**
MOVE b	**y =sin(68)/90**
CALL SP1( )	**RETURN**
MOVE c	
CALL SP1( )	
MOVE d	

2.5.7 Εντολές REACT, REACTI και IGNORE

Οι εντολές **REACT, REACTI** και **IGNORE** έπαιξαν καθοριστικό ρόλο στην ολοκλήρωση της παρούσας διπλωματικής εργασίας, καθώς είναι οι εντολές που έχουν άμεση σχέση με το έγκαιρο σταμάτημα της εκτέλεσης ενός προγράμματος όταν ο ελεγκτής του ρομποτικού βραχίονα παραλάβει ένα σήμα στην είσοδο του, που έχει ως αποτέλεσμα την εκτέλεση ενός υποπρογράμματος.

Εντολή REACT

Η εντολή αυτή αναγκάζει το ρομπότ να παρεκλίνει από τις εντολές του κυρίως προγράμματος και να εκτελέσει το υποπρογράμμα όταν το σήμα στην αντίστοιχη είσοδο του ελεγκτή του ρομποτικού βραχίονα είναι θετικό(**1**), μέχρις ότου ολοκληρωθούν όλες οι εντολές του υποπρογράμματος.

Π.χ. **REACT σήμα, προγράμματα,** επίπεδο προτεραιότητας

Η εντολή εκτελείται όταν το επίπεδο της εισόδου αλλάζει από 0 σε 1, ή από 1 σε 0.

Η προτεραιότητα πρέπει να είναι ένας αριθμός μεταξύ 1 και 127, ενώ το επίπεδο προτεραιότητας από προεπιλογή είναι 1.

Εντολή REACTI

Η εντολή αυτή είναι η ίδια με την προηγούμενη με την μόνη διαφορά ότι η εκτέλεση του υποπρογράμματος θα πραγματοποιηθεί **ακριβώς μόλις η αντίστοιχη είσοδος γίνει θετική(1)**.

Π.χ. **REACTI σήμα, πρόγραμμα,** επίπεδο προτεραιότητας

Π.χ
```
MOVE a
BREAK
REACTI 1001, SP1
MOVE b
MOVE c
BREAK
MOVE d
```

Το ρομπότ θα κινηθεί στο σημείο a, στη συνέχεια ελέγχει εάν το σήμα 1001 είναι θετικό. Εάν είναι, τότε το κυρίως πρόγραμμα θα καλέσει το

υποπρόγραμμα SP1 και όταν εκτελέσει όλες τις εντολές του υποπρογράμματος θα συνεχίσει τις υπόλοιπες εντολές του κυρίως προγράμματος(κίνηση στα σημεία b, c, d). Εάν το σήμα είναι αρνητικό τότε το κυρίως πρόγραμμα θα εκτελέσει τις επόμενες εντολές, δηλαδή κίνηση στα σημεία a, b, c.

1. Οι πρώτες 12 είσοδοι(**1001-1012**) είναι οι μόνες που επιτρέπονται για σήματα που αφορούν τις εντολές **REACT, REACTI**.
2. **Το σύστημα δεν θα αναγνωρίσει τις αλλαγές στα επίπεδα των εισόδων(είτε από 0 σε 1, είτε από 1 σε 0), μέχρις ότου αυτά μείνουν σταθερά για τουλάχιστον 18ms.**

Εντολή IGNORE

Διαγράφει την παρακολούθηση της αντίστοιχης εισόδου και στη συνέχεια ακυρώνει την επίδραση των εντολών **REACT, REACTI**.

Π.χ. **IGNORE σήμα**

Π.χ.
```
MOVE a
BREAK
REACTI 1001, SP1
MOVE b
MOVE c
MOVE d
IGNORE 1001
MOVE e
```

Σε αυτήν την περίπτωση το πρόγραμμα θα καλέσει το υποπρόγραμμα SP1, κατά την διάρκεια στην οποία το σήμα 1001 γίνεται θετικό(1) και κατά την διάρκεια των κινήσεων στα σημεία b, c, d.

2.5.8 Εντολές DO READY και BREAK

Εντολή DO READY

Μεταφέρει τον ρομποτικό βραχίονα σε μια συγκεκριμένη θέση που έχει οριστεί ως προεπιλογή. Η θέση αυτή είναι η ακόλουθη για τις 6 αρθρώσεις:

Joint 1: **0**^{o}, Joint 2: **-90**^{o}, Joint 3: 9**0**^{o}, Joint 4: **0**^{o}, Joint 5: **0**^{o}, Joint 6: **0**^{o}

Πρόκειται για τη θέση ασφαλείας που τοποθετούμε το ρομπότ πάντα πριν απενεργοποιήσουμε τον ρομποτικό βραχίονα.

Εντολή BREAK

Αυτή η εντολή καθυστερεί την εκτέλεση του προγράμματος μέχρις ότου το ρομπότ φτάσει στον προορισμό του.

Η εντολή **BREAK** προτιμάται όταν ο χρήστης θέλει να αποφύγει τυχόν υπολογισμούς ή την διαχείρηση εντολών για τις εισόδους/εξόδους όταν η κίνηση του ρομπότ πραγματοποιείται μεταξύ δύο σημείων.

Π.χ. **MOVE a**
MOVE b
BREAK
MOVE c
MOVE d

Σε αυτήν την περίπτωση το ρομπότ θα κινηθεί στα σημεία a και b. Όμως, επειδή υπάρχει η εντολή **BREAK**, αμέσως μετά το ρομπότ θα φτάσει πρώτα την θέση b και στη συνέχεια θα κινηθεί στις θέσεις c και d. Έτσι εξασφαλίζεται ότι ρομπότ θα περάσει από την θέση b. Παρατηρείται λοιπόν, ότι εάν δεν υπάρχει η εντολή **BREAK** μετά από κάθε μετακίνηση του ρομπότ σε κάποιο σημείο, το ρομπότ υπάρχει περίπτωση να μην περάσει από το συγκεκριμένο σημείο αλλά να περάσει πολύ κοντά από αυτό. Αυτό μπορεί να οφείλεται στην ταχύτητα που έχουμε δώσει στο ρομπότ, οπότε για να φτάσει σύμφωνα με αυτή στον προορισμό του υπάρχει περίπτωση να κάνει μια μικρή παράκαμψη.

2.6 Χειριστήριο (Teach Pendant)

Ο ελεγκτής επιτρέπει στον χρήστη να ελέγξει τον ρομποτικό βραχίονα μέσω του χειριστηρίου (Teach Pendant). Τα κουμπιά του χειριστηρίου θα αναλυθούν στην συνέχεια.

Εικόνα 2.8: Χειριστήριο ρομποτικού βραχίονα Staubli RX90L

2.6.1 Function keys

Τα function keys είναι τα κουμπιά εκείνα που επιτρέπουν στον χρήστη να ξεκινήσει συγκεκριμένες λειτουργίες. Αυτά είναι τα **EDIT, DISP, CLR ERR, CMD** και **PROG SET.** Κάθε κουμπί έχει ένα μικρό λαμπάκι πάνω αριστερά.

Κουμπί EDIT

Το κουμπί αυτό επιτρέπει στον χρήστη να αλλάξει τις πραγματικές μεταβλητές ή τις μεταβλητές θέσης (real variables and location variables), χρησιμοποιώντας τα αριθμητικά κουμπιά του χειριστηρίου.

Κουμπί DISP

Το συγκεκριμένο κουμπί χρησιμοποιείται για να εμφανίσει στην οθόνη του χειριστηρίου (κατά σειρά από αριστερά προς τα δεξιά) τις λέξεις: γωνίες των

αρθρώσεων (joint values), την καρτεσιανή θέση του ρομπότ(world location), την κατάσταση του συστήματος (status), την κατάσταση των εισόδων/εξόδων (binary I/O), και τέλος το τελευταίο λάθος που προέκυψε από την χρήση του ρομπότ (last error).

Κουμπί CLR ERR

Κάθε φορά που παρατηρείται ένα λάθος, το χειριστήριο εμφανίζει μια προειδοποίηση στην οθόνη και ανάβει το λαμπάκι του κουμπιού **CLR ERR.** Για να διαγραφεί το μήνυμα και να επιστρέψει το χειριστήριο στην προηγούμενη κατάσταση πρέπει να πατηθει το κουμπί **CLR ERR**.

Κουμπί CMD

Το κουμπί αυτό χρησιμοποιείται για να εμφανίσει στην οθόνη του χειριστηρίου (κατά σειρά από αριστερά στα δεξιά) την αυτόματη εκκίνηση (AUTO START), το calibration (CALIB), την αποθήκευση όλης την μνήμης στο δίσκο (STORE ALL), ή για να ενεργοποιήσει τα προγράμματα CMD1 και CMD2.

Κουμπί PROG SET

Το συγκεκριμένο κουμπί χρησιμοποιείται από τον χρήστη για να εμφανίσει στην οθόνη (κατά σειρά από αριστερά στα δεξιά) τις λέξεις: ένα νέο πρόγραμμα(NEW), τον καθορισμό του αριθμού των βημάτων(1 STEP), τον καθορισμό του αριθμού των κύκλων του προγράμματος (1 CYCLE), την προσαρμογή της ονομαστικής ταχύτητας (SPEED 50) και τέλος την εκκίνηση του προγράμματος που βρίσκεται στην μνήμη (START). Για να κάνει ο χρήστης την επιλογή που θέλει, χρησιμοποιεί τα user keys, που εξετάζονται παρακάτω.

2.6.2 User keys

Τα συγκεκριμένα κουμπιά χρησιμεύουν στο να κάνει ο χρήστης την επιλογή που θέλει από την οθόνη του χειριστηρίου πατώντας το αντίστοιχο κουμπί που βρίσκεται κάτω από την εκάστοτε επιλογή.

2.6.3 Data entry keys

Τα κουμπιά αυτά επιτρέπουν στον χρήστη να απαντά σε μηνύματα που εμφανίζονται στην οθόνη του χειριστηρίου. Τα κουμπιά αυτά είναι : +/NAI(+/YES), -/OXI(-/NO), διαγραφή(DELETE), οι αριθμοί 0-9 και το σημείο στίξης(.).

2.6.4 Mode selection keys

Τα συγκεκριμένα κουμπιά είναι τα μόνα κουμπιά στο χειριστήριο που επιτρέπουν στον χρήστη να αλλάξει mode. Για εύκολη αναγνώριση, όταν χρησιμοποιούνται ανάβει το αντίστοιχο λαμπάκι στην πάνω αριστερά γωνία του εκάστοτε κουμπιού. Τα κουμπιά αυτά είναι : κουμπί έκτακτης ανάγκης (**EMERGENCY STOP** key), κουμπί **RUN/HOLD**, κουμπί **DIS PWR**, κουμπί **COMP PWR** και τέλος το κουμπί **MAN/HALT**, τα οποία θα αναλυθούν παρακάτω.

Κουμπί έκτακτης ανάγκης (EMERGENCY STOP key)

Το κουμπί αυτό σταματά το πρόγραμμα χρησιμοποιώντας μια εντολή **HALT** και σταματά το ρομπότ κόβοντας το ρεύμα του ρομποτικού βραχίονα και ενεργοποιώντας τα φρένα. Το συγκεκριμένο κουμπί χρησιμοποιείται μόνο σε περίπτωση έκτακτης ανάγκης. Το κουμπί αυτό έχει δύο λειτουργίες, πρώτον τα πάτημα του κουμπιού (για ενεργοποίηση του **EMERGENCY STOP**) και δεύτερον το δεξιόστροφο γύρισμα του κουμπιού (για απενεργοποίηση του **EMERGENCY STOP**) Για να συνεχίσει ο χρήστης να χρησιμοποιεί το ρομπότ πρέπει να γυρίσει το κουμπί δεξιόστροφα και στην συνέχεια να πατήσει το κουμπί **COMP PWR** που θα εξηγηθεί παρακάτω, και τέλος να πατήσει το κουμπί **ARM POWER ON** που αναβοσβήνει στο μπροστινό πάνελ (όπως εξηγήθηκε στο **υποενότητα 2.4.2**)

Κουμπί RUN/HOLD

Το κουμπί αυτό προκαλεί το άμεσο σταμάτημα του ρομποτικού βραχίονα. Η ακολουθία του προγράμματος διακόπτεται και το μήνυμα 'HOLD' εμφανίζεται στην οθόνη του χειριστηρίου. Εάν πατηθεί ξανά, το πρόγραμμα συνεχίζει από εκεί που σταμάτησε.

Κουμπί DIS PWR

Το συγκεκριμένο κουμπί κόβει το ρεύμα του ρομποτικού βραχίονα.

Κουμπί COMP PWR

Εάν το ρεύμα του ρομποτικού βραχίονα έχει απενεργοποιηθεί (πατώντας το κουμπί έκτακτης ανάγκης είτε στο χειριστήριο είτε στο μπροστινό πάνελ, είτε εάν έχει προκύψει κάποιο λάθος), τότε πρέπει να πατηθεί το συγκεκριμένο κουμπί για να να ενεργοποιηθεί το ρεύμα. Αυτό τοποθετεί το σύστημα σε computer mode (COMP) σε προετοιμασία για την ενεργοποίηση του ρομποτικού βραχίονα. Όταν πατηθεί το κουμπί από τον χρήστη, το κουμπι **ARM POWER ON** αρχίζει να αναβοσβήνει και το σύστημα περιμένει το πάτημα του κουμπιού για 15 δευτερόλεπτα. Μετά από αυτόν το χρόνο το κουμπί σταματά να αναβοσβήνει και η διαδικασία πρέπει να ξαναγίνει από την αρχή. Τέλος, το κουμπί αυτό χρησιμοποιείται όταν ο χρήστης θέλει να εκτελέσει ένα πρόγραμμα, οπότε πριν την εκτέλεση του προγράμματος από τον υπολογιστή ο χρήστης πρέπει να πατήσει το κουμπί **COMP PWR.**

Κουμπί MAN/HALT

Το κουμπί αυτό προκαλεί το άμεσο σταμάτημα του ρομποτικού βραχίονα. Η εκτέλεση του προγράμματος διακόπτεται και εμφανίζεται στην οθόνη το μήνυμα 'E-STOP'. Τέλος, όταν ο χρήστης επιθυμεί να μετακινήσει τον ρομποτικό βραχίονα χειροκίνητα τότε πρέπει να αλλαχθεί το mode από **COMP**(κουμπί COMP PWR) σε **MAN**(κουμπί MAN/HALT).

Manual mode selection

Το manual mode χρησιμοποιείται συχνά για να μετακινηθεί ο ρομποτικός βραχίονας σε μια συγκεκριμένη θέση. Όταν το σύστημα βρίσκεται σε manual mode τότε το λαμπάκι του κουμπιού **MAN/HALT** είναι αναμένο καθώς και ένα από τα λαμπάκια του manual mode. Τα λαμπάκια του «manual mode» υποδεικνύουν στον χρήστη κάθε φορά το mode στο οποίο βρίσκεται ο ρομποτικός βραχίονας. Αυτά είναι : **WORLD, TOOL, JOINT, FREE.** Για να αλλάξει ο χρήστης mode, π.χ. από **WORLD** σε **TOOL** ή από **TOOL** σε **JOINT** αρκεί το πάτημα του κουμπιού **MAN/HALT**.

2.6.5 Manual control keys

Τα κουμπιά στο δεξί μέρος του χειριστηρίου είναι τα manual control keys. Όταν ο χρήστης βρίσκεται σε manual mode, αυτά τα κουμπιά επιτρέπουν στον χρήστη να επιλέξει την κατεύθυνση της κάθε άρθρωσης. Τα κουμπιά είναι : **X/1, Y/2, Z/3, RX/4, RY/5, RZ/6.** Τα κουμπιά X,Y,Z,RX,RY,RZ αφορούν την κίνηση του ρομποτικού βραχίονα όταν ο χρήστης βρίσκεται σε **WORLD MODE**, ενώ τα ίδια κουμπιά 1,2,3,4,5,6 όταν ο χρήστης βρίσκεται σε **JOINT MODE** (εδώ μπορεί ο χρήστης να κινήσει την κάθε άρθρωση ξεχωριστά).

Speed bars

Οι μπάρες ταχύτητας επιτρέπουν στον χρήστη να επιλέξειτην ταχύτητα και την κατεύθυνση του ρομποτικού βραχίονα (στα θετικά ή στα αρνητικά είτε πρόκειται για γωνία άρθρωσης είτε για σημείο στον τρισδιάστατο χώρο).Ο χρήστης μπορεί να επιλέξει αργή ή γρήγορη ταχύτητα όσο απομακρύνεται ο αντίχειρας του από τα σύμβολα +/- σε όποια από τις δύο κατευθύνσεις επιθυμεί (**Εικόνα 2.6.1**).

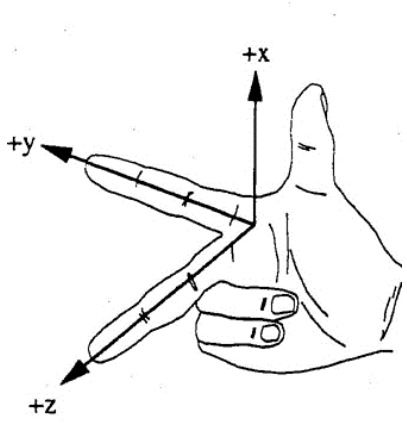
Κουμπί SLOW

Το συγκεκριμένο κουμπί επιτρέπει στον χρήστη να χρησιμοποιήσει δύο διαφορετικά εύρη ταχύτητας, το αυτόματο που είναι προεπιλεγμένο ή το mode πάρα πολύ αργής ταχύτητας πατώντας το κουμπί **SLOW.**

2.7 Σύστημα συντεταγμένων

Για να εντοπιστεί ένα σημείο στον χώρο (3D), χρησιμοποιείται το σύστημα συντεταγμένων.

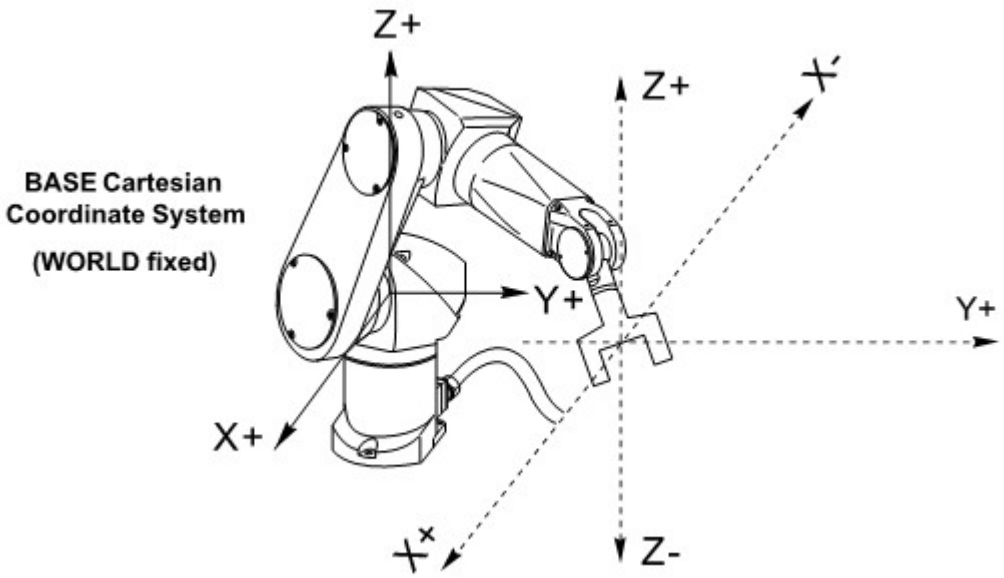
- Ένα σύστημα συντεταγμένων αναπαριστάται από μια αρχή των αξόνων και 3 άξονες που ονομάζονται X, Y, Z όλοι κάθετοι μεταξύ τους.
  Ο προσανατολισμός του κάθε άξονα ακολουθεί τον κανόνα του δεξιού χεριού, όπου X : αντίχειρας , Y : δείκτης, Z : μέσος



Εικόνα 2.9: Ο κανόνας των 3 δακτύλων του δεξιού χεριού

Συντεταγμένες συστήματος σε WORLD mode

Όλες οι κινήσεις είναι παράλληλες στις WORLD συντεταγμένες. Οι περιστροφές RX, RY, RZ είναι τέτοιες ώστε να συνάδουν με τις WORLD συντεταγμένες. Οι κινήσεις πραγματοποιούνται με το πάτημα των κουμπιών +/- στα speed bar keys(όπως αναλύθηκε στην **υποενότητα 2.6.5**).



Εικόνα 2.10: Βασικό σύστημα συντεταγμένων(WORLD mode)

Ένα σημείο στο χώρο καθορίζεται από 6 παραμέτρους, X, Y, Z, y, p, r.

- Οι τιμές των X, Y, Z εκφράζονται σε χιλιοστά και συμβολίζουν την απόσταση μεταξύ της $6^{ης}$ άρθρωσης ή του εργαλείου και της αρχής των αξόνων του βασικού συστήματος συντεταγμένων. Αυτές οι τιμές απεικονίζουν την ακριβή θέση του ρομποτικού βραχίονα.
- Οι τιμές y, p, r εκφράζονται σε μοίρες και συμβολίζουν τον προσανατολισμό της $6^{ης}$ άρθρωσης ή του εργαλείου.

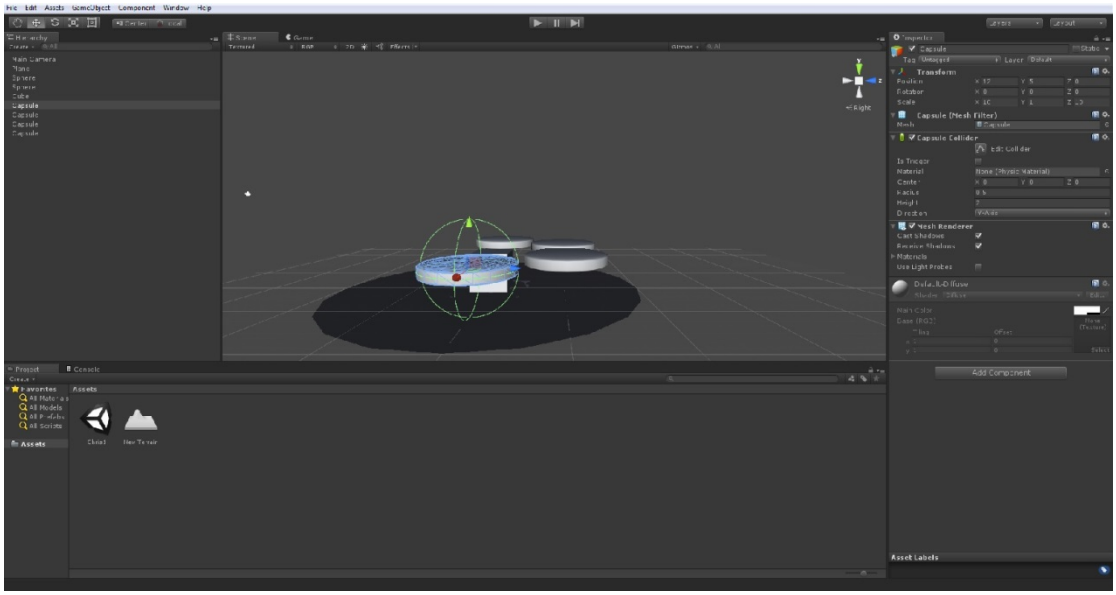
Για τη συγκεκριμένη ενότητα, οι πληροφορίες συντέθηκαν από τα manual της Adept Technology Inc και συγκεκριμένα από τα [18]-[20]. Για περισσότερες διευκρινίσεις και πληροφορίες μπορεί ο αναγνώστης να ανατρέξει στη συγκεκριμένη βιβλιογραφία.

3

Το περιβάλλον του Unity3d

3.1 Περιγραφή

Το λογισμικό Unity3d είναι μια πλατφόρμα ανάπτυξης βιντεοπαιχνιδιών, η οποία δημιουργήθηκε από την εταιρία Unity Technologies, με σκοπό την ανάπτυξη βιντεοπαιχνιδιών για ηλεκτρονικού υπολογιστές, κονσόλες, κινητές συσκευές και ιστοσελίδες. Η πρώτη έκδοση του Unity ανακοινώθηκε μόνο για Mac OS X στο παγκόσμιο συνέδριο εφαρμογών της Apple το 2005, και έχει έκτοτε επεκταθεί ώστε να η χρήση του να στοχεύει σε 21 πλατφόρμες. [1] Το λογισμικό Unity3d διαθέτει ρυθμίσεις για ανάλυση και συμπίεση κειμένου για κάθε πλατφόρμα που υποστηρίζει η παιχνιδομηχανή, και μπορεί να δεχθεί εικονικά μοντέλα που έχουν κατασκευαστεί με χρήση άλλου λογισμικού σε μορφή obj. και fbx.. [23]



Εικόνα 3.1: Περιβάλλον Unity3d

Για την ανάπτυξη των εφαρμογών υποστηρίζει τις ακόλουθες γλώσσες προγραμματισμού:

- **JavaScript**
- **C#**
- **Boo**(η οποία είναι βασισμένη σε γλώσσα **Python**)
- **Unity Script**

Το συγκεκριμένο λογισμικό επιτρέπει την χρήση διαφορετικού εξωτερικού περιβάλλοντος **IDE** (Intergrated Development Environment). Έχει ως προεπιλογή το MonoDevelop το οποίο χρησιμοποιείται στην παρούσα διπλωματική εργασία μαζί με την γλώσσα προγραμματισμού C# για τον προγραμματισμό της συμπεριφοράς του ανθρώπου (Avatar), ο οποίος θα συνεργάζεται με τον ρομποτικό βραχίονα Staubli RX90L.

3.2 Γλώσσα προγραμματισμού C#

Η επιλογή μιας αντικειμενοστραφούς γλώσσας προγραμματισμού είναι αρκετά σημαντική για την ανάπτυξη εφαρμογών στο Unity3d. Η **C#** είναι μια ολοκληρωμένη αντικειμενοστραφής γλώσσα προγραμματισμού σχεδιασμένη για την δημιουργία λογισμικού σε .Net Framework[3].

Ο κώδικας που γράφεται σε C# οργανώνεται σε Namespaces. Ένα Namespace μπορει να περιέχει ένα από τα ακόλουθα: [6] Namespaces, Classes, Delegates, Enums, Structs, Interfaces.

Ένας ακόμη λόγος που χρησιμοποιούνται οι αντικειμενοστραφείς γλώσσες προγραμματισμού είναι ότι επιτρέπουν την αξιοποίηση εννοιών, όπως είναι η κληρονομικότητα (inheritance) και η αφαιρετική γραφή κώδικα (abstraction), δυνατότητες που έχουν πολλαπλά οφέλη όπως είναι, για παράδειγμα, η εύκολη

[3] Net Framework είναι ένα πλαίσιο λογισμικού, το οποίο αναπτύχθηκε από την Microsoft και τρέχει κυρίως σε Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη κλάσεων γνωστή ως Framework Class Library(FCL) και παρέχει interoperability(κάθε γλώσσα μπορεί να γραφεί σε κώδικα από διαφορετικές γλώσσες) μεταξύ πολλών γλωσσών προγραμματισμού. Τα προγράμματα τα οποία είναι γραμμένα σε .Net Framework εκτελούνται σε ένα περιβάλλον λογισμικού γνωστό ως Common Language Runtime(CLR), μια εφαρμογή εικονικής μηχανής η οποία παρέχει υπηρεσίες όπως η ασφάλεια, η διαχείρηση μνήμης, και η διαχείρηση εξαιρέσεων. Τα FCL, CLR μαζί αποτελούν το .Net Framework.

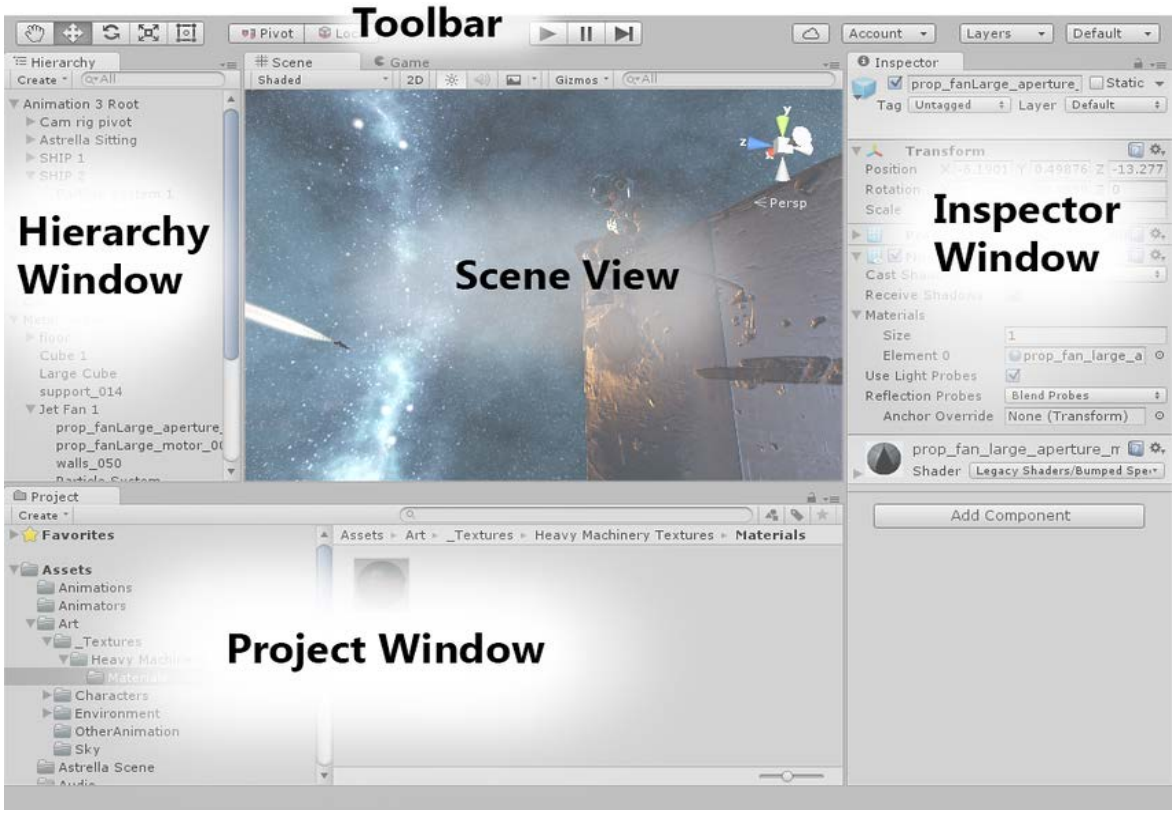
ανάγνωση και τροποποίηση προγραμμάτων και η επαναχρησιμοποίηση των κλάσεων. Δεν θα αναλυθεί περαιτέρω η γλώσσα προγραμματιμού και τα χαρακτηριστικά της, καθώς αυτό δεν αποτελεί στόχο της παρούσας διπλωματικής εργασίας και ούτως ή άλλως δεν μπορεί να περιοριστεί σε ένα και μόνο κεφάλαιο. Όποτε όμως χρειάζεται θα αναλύονται συγκεκριμένες εντολές και ολόκληρος ο κώδικας που θα χρησιμοποιηθεί.

3.3 Λειτουργία του λογισμικού

Το Unity3d παρέχει στον χρήστη την δυνατότητα να δημιουργεί **Projects**, καθένα από τα οποία αποτελείται από έναν αριθμό 'σκηνών'(scene). Η κάθε σκηνή απαρτίζεται από έναν αριθμό **Game Objects** τα οποία αποτελούν μια ομάδα **Components**. [8], [9]

Τα **Components** είναι αντικείμενα κάποιας κλάσης και το καθένα έχει τις δικές του μεθόδους και ιδιότητες. Το Unity3d ως λογισμικό παρέχει έτοιμα **Components** και υπάρχει η δυνατότητα από τον χρήστη να δημιουργήσει νέες κλάσεις, τα αντικείμενα των οποίων μπορούν να είναι **Components**. Η αξιοποίηση αυτών των κλάσεων πραγματοποιείται με την προσάρτηση τους σε κάποιο **Game Object**. [8], [9]

Ουσιαστικά τα **Game Objects** είναι κλάσεις (containers) οι οποίες αποκτούν την συμπεριφορά από τα **Components** που έχουν προσαρτημένα πάνω τους. Εφόσον όμως είναι και αντικείμενα κλάσεων μπορούν να έχουν και σχέση κληρονομικότητας (inheritance) μεταξύ τους. Το περιβάλλον του Unity3d είναι δομημένο με τέτοιο τρόπο έτσι ώστε να απεικονίζει άμεσα τις σχέσεις των αντικειμένων που υπάρχουν σε μια σκηνή και να δίνει μεγάλο πλήθος **Game Objects** με προσαρτημένα πάνω τους τα κατάλληλα **Components**. Αυτό φαίνεται στην εικόνα που ακολουθεί (**Εικόνα 3.3**).



Εικόνα 3.2 : Διαμόρφωση GUI στο Unity3d

Στα αριστερά της **Εικόνας 3.3** βρίσκεται το ιεραρχικό παράθυρο(Hierarchy Window) στο οποίο τοποθετούνται όλα τα **Game Objects** της σκηνής. Τα **Game Objects** απεικονίζονται δενδρικά, με αυτόν τον τρόπο παρουσιάζεται η σχέση μεταξύ των αντικειμένων. Στο **κεντρικό παράθυρο (scene view)** βρίσκεται η οθόνη της εικονικής σκηνής, στην οποία τοποθετούνται τα διάφορα αντικείμενα στον χώρο και με βάση τη **γραμμή εργαλειών (Toolbar)** που βρίσκεται στο πάνω μέρος της **Εικονας 3.3** αυτά μπορούν να τροποποιηθούν. Στο παράθυρο αριστερά βρίσκεται το **παράθυρο επίβλεψης (Inspector Window)**, στο οποίο φαίνονται για το επιλεγμένο Game Object τα αντίστοιχα Components που το απαρτίζουν. Στο κάτω μέρος της εικόνας βρίσκεται το **παράθυρο των Project (Project Window)**, στο οποίο φαίνονται όλα τα αρχεία του συγκεκριμένου **Project** που χρησιμοποιεί ο χρήστης οργανωμένα σε φακέλους και υποφακέλους. [23]

3.4 Βασικά Game Objects και Components

Σε αυτή την υποενότητα παρουσιάζονται τα σημαντικότερα **Game Objects** που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία. Αρχικά, στο Unity3d χρησιμοποιούνται κατά κόρον τα παρακάτω γεωμετρικά σχήματα: Κύβος (Cube), Σφαίρα (Sphere), Κάψουλα (Capsule),Κύλινδρος (Cylinder), Επίπεδο (Plane). Υπάρχουν επίσης **Game Objects** που συνδέονται με τον φωτισμό της σκηνής όπως: Directional light,Point light, Spot light, Area light Το σημαντικότερο **Game Object** το οποίο χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία είναι η **κάμερα**, ένα **Game Object** το οποίο είναι απαραίτητο σε κάθε σκηνή για την εκτέλεση οποιασδήποτε εφαρμογής.

Έχοντας επιλέξει ένα **Game Object** μπορούμε να δούμε στο παράθυρο επίβλεψης (Inspector Window) τέσσερα πολύ βασικά Components. Αυτά είναι τα ακόλουθα: Transform, Mesh Filter, Box Collider, Mesh Renderer. Το **Transform component** καθορίζει την θέση, τον προσανατολισμό και τις διαστάσεις του εκάστοτε αντικειμένου. Το **Mesh Filter component** συνδέει το αντικείμενο με ένα πλέγμα έτσι ώστε να μπορεί να απεικονιστεί η μορφή του και οι επιφάνειες του. Το **Box Collider component** έπαιξε καθοριστικό ρόλο στην παρούσα διπλωματική εργασία καθώς είναι μια κάψουλα η οποία περιβάλλει το χώρο γύρω από το αντικείμενο και μπορεί να εντοπίσει αντικείμενα που εισέρχονται, εξέρχονται ή βρίσκονται μέσα σε αυτό. Το **Mesh Renderer component** είναι υπεύθυνο για το πώς θα αποτυπώνονται σην οθόνη κατά την εκτέλεση της εφαρμογής οι μορφές που συνδέονται με το κάθε **Game Object**. Τέλος, προσθέτοντας το **Rigid Body component** σε ένα **Game Object** δίνεται η δυνατότητα να προστεθούν σε αυτό φυσικές ιδιότητες όπως για παράδειγμα η μάζα, να ασκηθούν πάνω του διάφορες δυνάμεις και να υπολογιστεί η επιτάχυνση που προκύπτει από τα παραπάνω.

3.5 Scripts-Behaviour Components

Προκειμένου η σκηνή και τα αντικείμενα που την απαρτίζουν να είναι διαδραστικά είναι αναγκαία η δημιουργία **Script Components** που να λαμβάνουν κάποια είσοδο και να παράγουν κάποια έξοδο στη σκηνή κατά την

εκτέλεση ενός προγράμματος. Αυτός είναι, εξ άλλου, και ο λόγος που επιλέχθηκε το συγκεκριμένο λογισμικό.

Κάθε **Component** που βρίσκεται στη σκηνή συνοδεύεται από μεθόδους που δίνουν πρόσβαση στις μεταβλητές δημόσιας πρόσβασης (public variables) και με αυτόν τον τρόπο η παραμετροποίηση του μπορεί να γίνει δυναμικά κατά την εκτέλεση ενός ή περισσοτέρων **Script Components**.

Τα **Script Components** είναι κλάσεις - παιδιά της MonoBehaviour κλάσης του Unity3d. Αυτό σημαίνει ότι κληρονομούνται η μέθοδος Start( ) που χρησιμοποιείται στην αρχικοποίηση των παραμέτρων του **Component** καθώς και η μέθοδος Update( ) η οποία εκτελείται σε κάθε frame και η OnGUI( ). [7]

Απαραίτητη θεωρείται η χρήση του namespace **UnityEngine** με σκοπό να είναι ορατές οι κλάσεις του Unity3d με τις μεθόδους που τη συνοδεύουν. Μια ειδική κατηγορία μεθόδων είναι οι events functions(συναρτήσεις γεγονότων), οι οποίες περιμένουν κάποιο συγκεκριμένο γεγονός για να εκτελέσουν το τμήμα του κώδικα που είναι γραμμένος σε αυτές. Τέτοιες είναι οι **OnTriggerEnter( )**, **OnCollisionEnter** και άλλες. Τέλος, κάθε **GameObject** έχει την δυνατότητα να έχει την ιδιότητα της ετικέτας(Tag) και με την συνάρτηση **FindGameObjectWithTag(''name'')** δίνεται η δυνατότητα στον προγραμματιστή να βρει το συγκεκριμένο αντικείμενο με την ετικέτα. [11], [25]

3.6 Assets

Τα στοιχεία που χρησιμοποιούνται για την δημιουργία ενός project στο Unity3d (σκηνές, εικόνες, Script Components, κ.α.) ονομάζονται **Assets** και βρίσκονται στο παράθυρο των **Projects** κάτω αριστερά στην **Εικόνα 3.3** σε ένα σύστημα με φακέλους και υποφακέλους.

Υπάρχει η δυνατότητα εισαγωγής Assets από εξωτερικά προγράμματα ή ακόμη και από άλλο project του Unity3d κάτι που επιταχύνει τον χρόνο ανάπτυξης μιας εφαρμογής.

3.7 Κάμερα Kinect

Η **Kinect** (με την κωδικη ονομασία Project Natal κατά την διάρκεια της ανάπτυξης) είναι μια συσκευή εισόδων ανίχνευσης κίνησης που δημιουργήθηκε από τη Microsoft για τις κονσόλες βιντεοπαιχνιδιών Xbox 360 και Xbox One καθώς και για τα Windows PCs. Βασίζεται σε μια περιφερειακή κάμερα, επιτρέπει στους χρήστες να ελέγχουν και να αλληλεπιδρούν με την κονσόλα ή τον υπολογιστή τους χωρίς την ανάγκη για έναν ελεγκτή παιχνιδιών, μέσω μιας φυσικής διασύνδεσης του χρήστη χρησιμοποιώντας σήματα και προφορικές εντολές. Η πρώτη γενιά **Kinect** εισήχθη για πρώτη φορά το Νοέμβριο του 2010, σε μια προσπάθεια να διευρύνει το κοινό του Xbox 360 πέραν του τυπικού gamer του. Μια έκδοση για Windows κυκλοφόρησε την 1η Φεβρουαρίου, 2012. Η **Kinect** ανταγωνίζεται διάφορους ελεγκτές κίνησης άλλων παιχνιδομηχανών, όπως το Wii Remote Plus για το Wii και το Wii U, το Playstation Move/Playstation Eye για το Playstation 3 και το Playstation Camera για το Playstation 4. Η Microsoft κυκλοφόρησε την πρώτη έκδοση του λογισμικού Kinect για Windows 7 στις 16 Ιουνίου, 2011. Αυτό το SDK είχε σκοπό να επιτρέψει στους προγραμματιστές να γράψουν εφαρμογές Kinect σε άλλες γλώσσες προγραμματισμου, όπως η C++/CLI, C# ή Visual Basic .NET. [43]

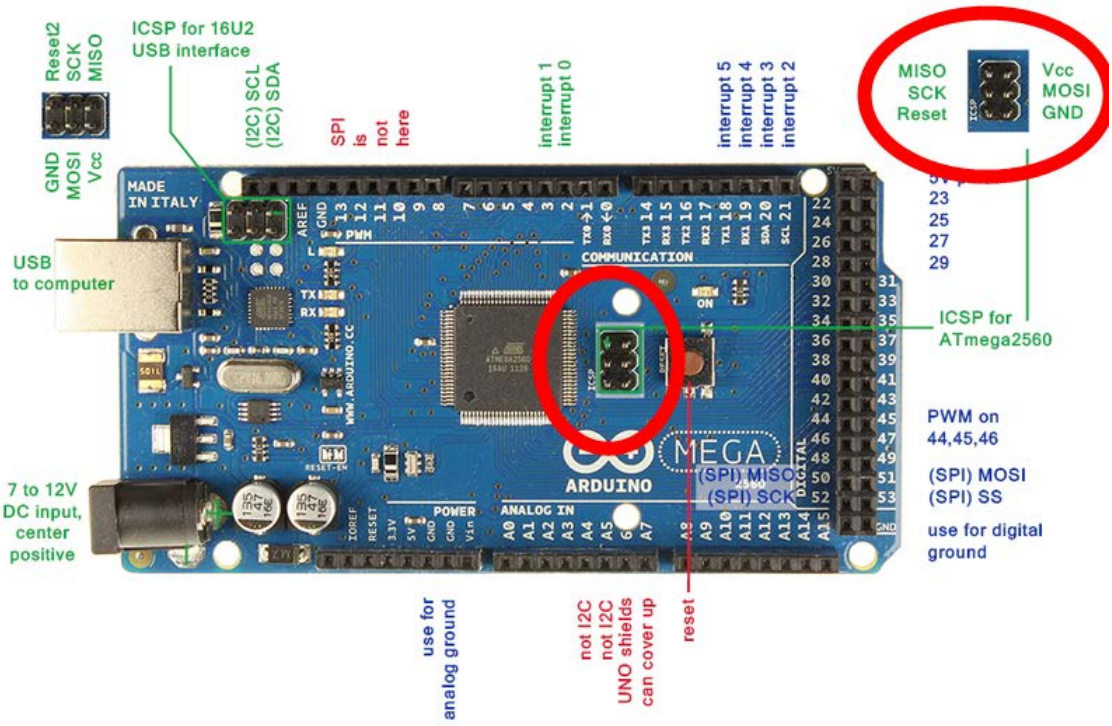
Εικόνα 3.3 : Κάμερα Kinect

4

Μικροελεγκτής

4.1 ArduinoMega

Το **ArduinoMega** είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοιχτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για μια γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. [2]



Εικόνα 4.1: Πλακέτα ArduinoMega

Μια πλακέτα Arduino αποτελείται από έναν μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ρυθμιστή τάσης 5Volts και έναν κρυσταλλικό ταλαντωτή 16MHZ (ή κεραμικό ανηχητή σε κάποιες παραλλαγές). Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος σε bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. [2]

Σε εννοιολογικό επίπεδο, στην χρήση του Arduino software stack, όλα τα boards προγραμματίζονται με μια RS-232 σειριακή σύνδεση, αλλά ο τρόπος που επιτυγχάνεται αυτό διαφέρει σε κάθε hardware εκδοχή. Οι σειριακές πλάκες Arduino περιέχουν ένα απλό level shifter κύκλωμα για να μετατρέπει μεταξύ του σήματος επιπέδου RS-232 και TTL. Τα σύγχρονα Arduino προγραμματίζονται μέσω USB.[23]

Τα χαρακτηριστικά του ArduinoMega είναι :

- **Μικροελεγκτής ATmega2560**
- **Τάση λειτουργίας 5Volts**
- **Προτεινόμενη τάση εισόδων 7-12Volts (όριο 6-20Volts)**
- **54 ψηφιακά πινάκια εισόδων/εξόδων(από τα οποία τα 15 χρησιμοποιούνται ως PWM έξοδοι)**
- **16 αναλογικές είσοδοι**
- **4 UARTS(σειριακές θύρες)**

Η πλακέτα Arduino μπορεί να δέχεται ηλεκτρικά σήματα(τάσεις) εντός των αποδεκτών ορίων και να παρέχει τις αντίστοιχες εξόδους που προκύπτουν από το πρόγραμμα που εκτελείται. Οι έξοδοι μπορούν να είναι από 0-5Volts που παρέχονται από τους ψηφιακούς I/O ακροδέκτες(0-25 για ArduinoMega).

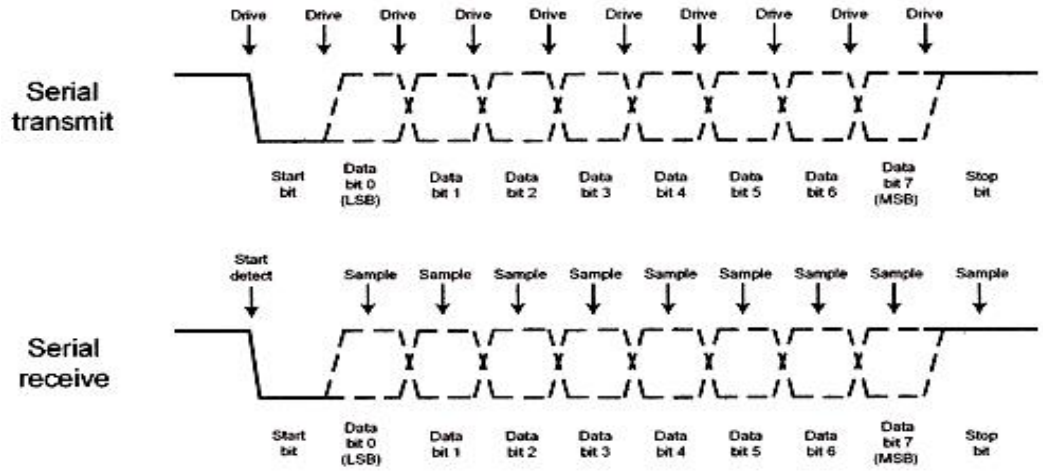
Δυνατότητα παραγωγής αναλογικού σήματος εξόδου δεν υπάρχει, εφόσον δεν μπορεί να παραχθεί σαν έξοδος τάση μεταξύ των 0 και 5 volts. Για την προσέγγιση τέτοιων σημάτων υποστηρίζεται από τους ακροδέκτες PWM(Pulse Width Modulation).

Στην περίπτωση των ακροδεκτών PWM του Arduino η μέγιστη τάση που μπορεί να παραχθεί είναι 5Volts και η ελάχιστη 0Volts, επομένως η τελική σχέση που περιγράφει την ισοδύναμη τάση εξόδου είναι:

$$V_{PWM} = 5 * D$$

όπου D ο κύκλος λειτουργίας(Duty Cycle), ο οποίος περιγράφει το ποσοστό του χρόνου που ο παλμός έχει την υψηλή τιμή του ανά περίοδο.

Επιπλέον, υπάρχει η δυνατότητα να στέλνονται bytes σε ηλεκτρονικό υπολογιστή μέσω θύρας USB. Αυτό συμβαίνει καθώς όλες οι πλακέτες Arduino διαθέτουν τουλάχιστον μια σειριακή θύρα η οποία επικοινωνεί εκτός από τη θύρα USB και με τους ψηφιακούς ακροδέκτες 0(RX) και 1(TX) (σε ArduinoMega υπάρχουν στα πινάκια από 14-19 οι επιπλέον ψηφιακοί ακροδέκτες RX1, TX1, RX2, TX2, RX3, TX3). Το πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την μεταφορά των δεδομένων από τη μια συσκευή στην άλλη είναι το πρωτόκολλο UART. [23]

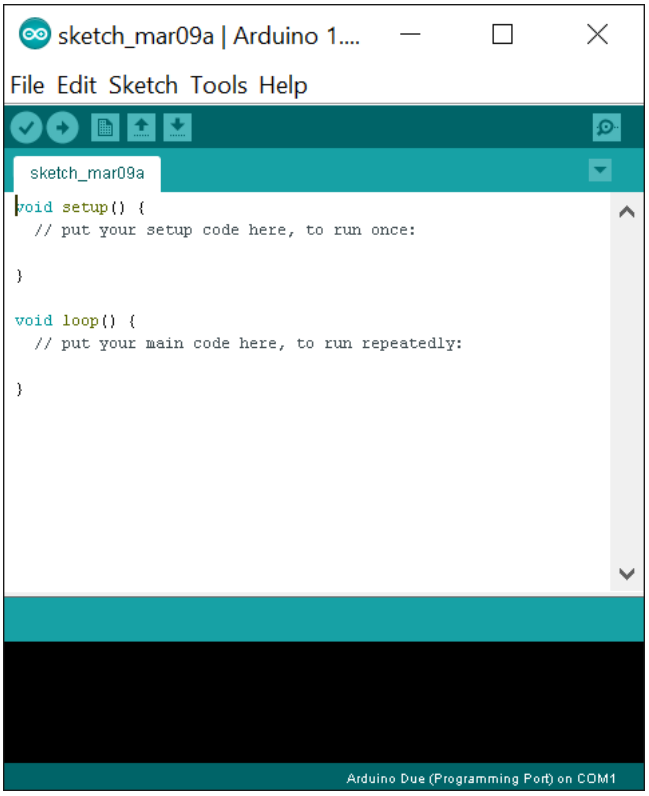


Εικόνα 4.2: Πρωτόκολλο UART

Οι ψηφιακοί ακροδέκτες μπορούν να χρησιμοποιηθούν επίσης ως είσοδοι, αλλά μπορούν να αντιληφθούν μόνο δύο τιμές τάσης, δηλαδή 'υψηλή' και 'χαμηλή'. Άλλοι είσοδοι είναι οι αναλογικοί ακροδέκτες(ANALOG IN) που φαίνονται στην κάτω πλευρά της **Εικόνας 4.1**.

Το πρόγραμμα που θα κληθεί να εκτελέσει ο μικροελεγκτής συντάσσεται σε ηλεκτρονικό υπολογιστή, χρησιμοποιώντας το IDE (Intergrated Development Environment) του Arduino.

Στο συγκεκριμένο πρόγραμμα καθορίζονται ποιοι ψηφιακοί ακροδέκτες θα χρησιμοποιηθούν και με ποιον τρόπο(είσοδοι, απλές έξοδοι ή PWM έξοδοι), ποιες αναλογικές θύρες θα είναι ανοιχτές και τι δεδομένα θα σταλούν από τον ηλεκτρονικό υπολογιστή μέσω USB. Οι εντολές που χρησιμοποιούνται στην παρούσα διπλωματικη εργασία παρατίθενται στην επόμενη ενότητα. [23]



Εικόνα 4.3: Arduino IDE

Μετά τη σύνταξη του προγράμματος και αφού έχει συνδεθεί η πλακέτα Arduino στον υπολογιστή, πατώντας το κουμπί Upload που είναι το δεύτερο κουμπί πάνω αριστερά στην **Εικόνα 4.3** πραγματοποιείται η φόρτωση του προγράμματος στην μνήμη του επεξεργαστή. Το πρόγραμμα διατηρείται στη μνήμη του επεξεργαστή ακόμη και εάν διακοπεί η παροχή ισχύος. Όταν αυτή επανέλθει το πρόγραμμα θα επανεκκινήσει.

Τέλος, η πλακέτα Arduino όταν είναι συνδεδεμένη με τον ηλεκτρονικό υπολογιστή μέσω του καλωδίου USB, τότε εκλαμβάνεται αυτή ως πηγή ισχύος. Διαφορετικά μπορεί να γίνει χρήση της υποδοχής εισόδου συνεχούς ρεύματος(DC jack) για παροχή ρεύματος στη πλακέτα από κάποια εξωτερική συσκευή. [23]

4.2 Προγραμματισμός ελεγκτή Arduino

Η δομή ενός προγράμματος Arduino ακολουθεί τη λογική και τη σύνταξη της γλώσσας C, και ο χρήστης που θα προγραμματίσει την πλακέτα

πρέπει να συπληρώσει τις βασικές συναρτήσεις **void setup( ) { …}** και **void loop( ) { …} .** [16]

Ο κώδικας που είναι γραμμένος στην συνάρτηση **void setup( )** εκτελείται μια φορά κατά την εκκίνηση του προγράμματος, οπότε το τμήμα αυτό του προγράμματος χρησιμοποιείται για αρχικοποίηση. Στη συγκεκριμένη συνάρτηση ορίζονται ποιοι ακροδέκτες θα χρησιμοποιηθούν, γίνεται ο ορισμός των μεταβλητών που θα χρησιμοποιηθούν και τέλος αρχικοποιείται η σειριακή επικοινωνία(ρυθμός αποστολής δεδομένων μέσω USB). Στη συνέχεια, εκτελείται ο κώδικας που είναι γραμμένος στη συνάρτηση **void loop( ),** που αποτελεί το κύριο μέρος του προγράμματος, μέχρι να εκτελεστούν όλες οι εντολές που περιλαμβάνει η συνάρτηση. Μόλις τελειώσει η πρώτη επανάληψη ξαναεκτελείται η συνάρτηση **void loop( )** επ' άπειρον. Ο μόνος τρόπος να σταματήσει η εκτέλεση του προγράμματος είναι η διακοπή της τροφοδοσίας της πλακέτας Arduino(είτε μέσω της αποσύνδεσης του καλωδίου USB είτε μεσω της αποσύνδεσης της εξωτερικής DC τροφοδοσίας). Για να επανεκκινήσει το πρόγραμμα από την αρχή υπάρχει συγκεκριμένο κουμπί **Reset** στη πλακέτα Arduino. [16]

Ακολουθούν βασικές συναρτήσεις [16]:

- **pinMode(pinNumber, mode):** Ορίζει εάν ο ψηφιακός ακροδέκτης που αντιστοιχίζεται στον ακέραιο αριθμό του πρώτου ορίσματος θα χρησιμοποιηθεί ως είσοδος ή έξοδος χρησιμοποιώντας ως δεύτερο όρισμα τις παραμέτρους **INPUT** ή **OUTPUT.**
- **Serial.begin(baudrate):** Ορίζει την έναρξη της σειριακής επικοινωνίας(μέσω της οποίας στέλνονται δεδομένα στον υπολογιστή), με παράμετρο το ρυθμό των δεδομένων (baudrate: bits ανά δευτερόλεπτο). Γίνεται χρήση των ακόλουθων αριθμών: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200. Συνηθισμένη για σημαντικό πλήθος εφαρμογών είναι η χρήση της τιμής **9600**. Επιπλέον, εάν χρησιμοποιηθεί οποιαδήποτε συνάρτηση της βιβλιοθήκης Serial, τότε δεν μπορούν να χρησιμοποιηθούν ως είσοδοι ή έξοδοι οι ακροδέκτες RX, TX.
- **digitalWrite(pinNumber, state):** Ορίζει εάν ο ψηφιακός ακροδέκτης που αντιστοιχεί στο πρώτο όρισμα θα έχει την τιμή 0 ή 5 Volts χρησιμοποιώντας ως δεύτερο όρισμα τις παραμέτρους **LOW** ή **HIGH.** Ισοδύναμα μπορούν να χρησιμοποιηθούν ως δεύτερο όρισμα οι ακέραιοι αριθμοί 0 (για LOW) ή 1 (για HIGH).

- **digitalRead(pinNumber):** Επιστρέφει την τιμή **LOW** ή **HIGH**, που αντιστοιχεί στον ψηφιακό ακροδέκτη του ορίσματος της συνάρτησης.
- **analogWrite(pinNumber, value):** Δημιουργεί με παλμό PWM μια τάση ως έξοδο στον ψηφιακό ακροδέκτη τυ πρώτου ορίσματος. Το δεύτερο όρισμα θέτει τον κύκλο λειτουργίας του παλμού PWM και λαμβάνει ακέραιες τιμές από 0 έως 255 που αντιστοιχούν σε κύκλο λειτουργίας 0-100%.
- **analogRead(pinNumber):** Επιστρέφει την τιμή από τον αναλογικό ακροδέκτη που υποδεικνύει το όρισμα της συνάρτησης, χρησιμοποιώντας έναν αναλογικό-ψηφιακό μετατροπέα 10bit. Αυτό σημαίνει ότι τάση από 0 έως 5 Volts αντιστοιχίζεται σε έναν ακέραιο αριθμό μεταξύ 0 και 1023.
- **delay(parameter)/delayMicroseconds(parameter):** Σταματά την εκτέλεση του προγράμματος για τον αριθμό millisecond και microseconds αντίστοιχα, που δίνεται στο όρισμα της συνάρτησης.
- **Serial.print(variable):** Επιστρέφει την τιμή της μεταβλητής που χρησιμοποιείται ως όρισμα της συνάρτησης.
- **Serial.available(variable):** Επιστρέφει τον αριθμό των bytes που είναι διαθέσιμα προς ανάγνωση στη σειριακή θύρα. Τα δεδομένα αυτά είναι αποθηκευμένα σε ένα buffer χωρητικότητας 64 bytes.

Οι συναρτήσεις **pinMode( )** και **Serial.begin( )** χρησιμοποιούνται κυρίως για αρχικοποίηση εντός της συνάρτησης **void setup( )**.

Υπάρχει μεγάλο πλήθος διαθέσιμων συναρτήσεων, ωστόσο κρίνεται ότι οι προηγούμενες συναρτήσεις που αναλύθηκαν αρκούν για μια πρώτη προσέγγιση του προγραμματισμού της πλακέτας Arduino.

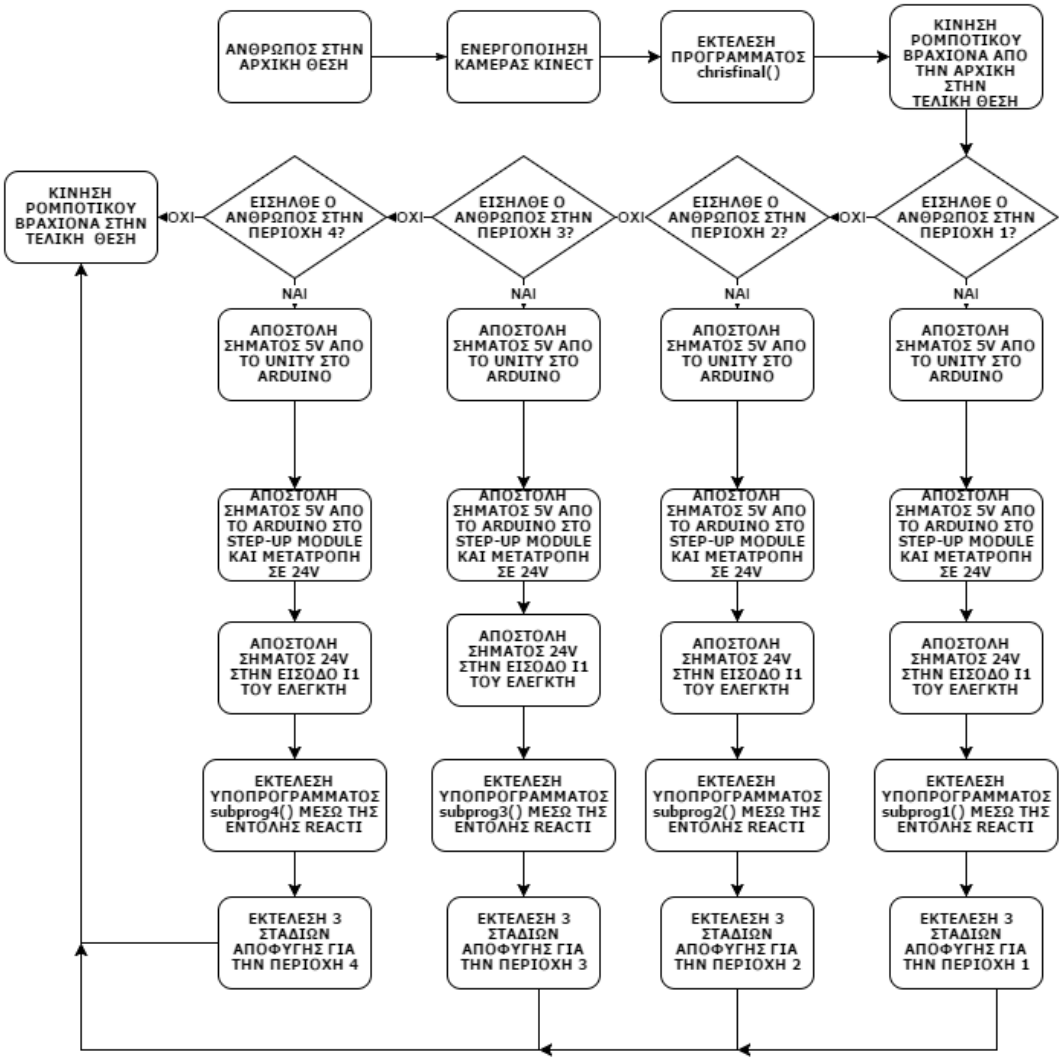
5

Απλή αποφυγή του ανθρώπου από το ρομπότ

5.1 Σκοπός και προσέγγιση

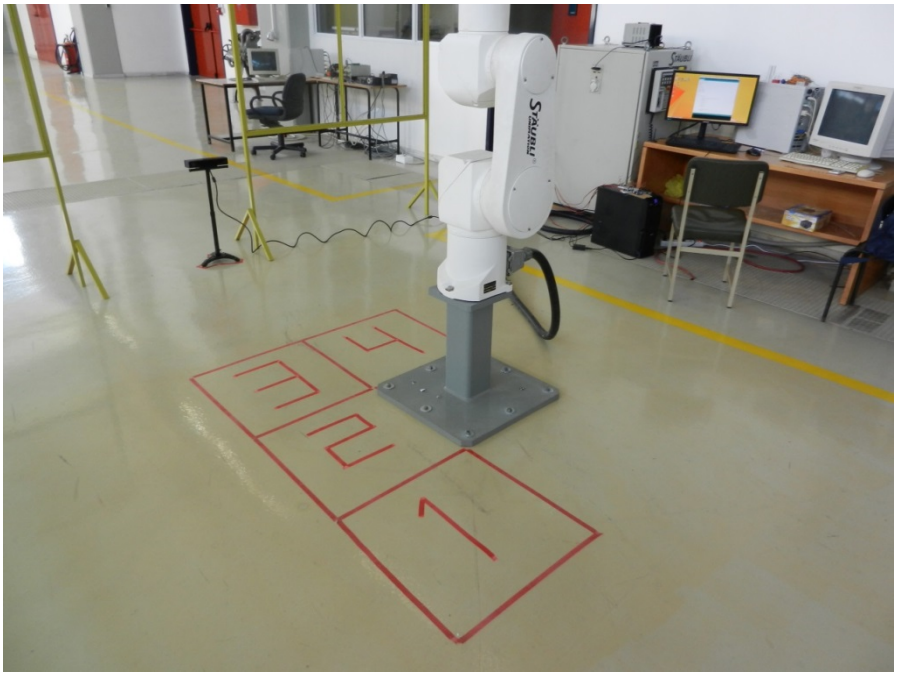
Ο σκοπός της πρώτης εφαρμογής είναι ο καταρχήν προγραμματισμός του ρομποτικού βραχίονα Staubli RX90L με σκοπό να αποφεύγει τον άνθρωπο, όταν εκτελεί συγκεκριμένη κίνηση σε όλο το εύρος της $1^{ης}$ άρθρωσης βάσει ενός αρχικού προγράμματος, Αυτό εμπεριέχει ως δευτερεύοντες στόχους τον προγραμματισμό του Unity3d, ώστε όταν λαμβάνει σήμα από την κάμερα Kinect να στέλνει σήμα στην πλακέτα Arduino, και τον προγραμματισμό της πλακέτας Arduino με σκοπό την αποστολή σήματος στις εισόδους του ελεγκτή του ρομποτικού βραχίονα όταν λάβει κατάλληλο σήμα από το λογισμικό Unity3d, έτσι ώστε να γίνει παράκαμψη του αρχικού προγράμματος.

Παρακάτω παρουσιάζεται ένα διάγραμμα ροής για την $1^{η}$ εφαρμογή, όπου φαίνονται όλες οι αλληλεπιδράσεις ανάμεσα στον άνθρωπο και το ρομποτικό βραχίονα(**Εικόνα 5.1**).



Εικόνα 5.1 : Διάγραμμα ροής 1ης εφαρμογής

Στην **Εικόνα 5.2** φαίνεται ο ρομποτικός βραχίονας Staubli RX90L, οι τέσσερις περιοχές, στις οποίες όταν υπάρχει άνθρωπος εκτελείται παράκαμψη του αρχικού προγράμματος. Στο βάθος φαίνονται, η κάμερα Kinect, ο ελεγκτής του ρομποτικού βραχίονα και οι 2 υπολογιστές από τους οποίους γίνεται ο έλεγχος των προγραμμάτων στο λογισμικό Unity3d καθώς και ο έλεγχος του ρομποτικού βραχίονα.



Εικόνα 5.2: Ο ρομποτικός βραχίονας και οι 4 περιοχές ενεργοποίησης αποφυγής

5.2 Προγραμματισμός ρομποτικού βραχίονα

Σημαντικό ρόλο στον προγραμματισμό έπαιξε η εντολή **REACTI**, η οποία όταν έχει ενεργοποιηθεί μια είσοδος του ελεγκτή του ρομποτικού βραχίονα καλεί ένα υποπρόγραμμα και πραγματοποιείται παράκαμψη του αρχικού προγράμματος με σκοπό την εκτέλεση ενός υποπρογράμματος. Εφόσον λοιπόν ο ρομποτικός βραχίονας έχει προγραμματιστεί να εκτελέσει μια κίνηση όπου η άρθρωση βάσης σαρώνει εύρος 280 μοιρών, όταν βρίσκεται ο άνθρωπος μέσα σε κάποια από τις 4 περιοχές (όπως φαίνεται στην **Εικόνα 5.2**) γίνεται παράκαμψη με στόχο την αποφυγή του ανθρώπου. Δηλαδή, καθώς η κάμερα εντοπίζει τον άνθρωπο, μέσω των προγραμμάτων στο Unity3d και στην πλακέτα ArduinoMega δημιουργείται κατάλληλο σήμα στην είσοδο του ελεγκτή του ρομποτικού βραχίονα και το ρομπότ τροποποιεί την πορεία του κατά προκαθορισμένο τρόπο. Τα προγράμματα στο Unity3d και της πλακέτας ArduinoMega θα αναλυθούν σε επόμενες υποενότητες.

Αρχικά, ο άνθρωπος παίρνει θέση στον χώρο (πατώντας τις 2 κόκκινες λωρίδες μπροστά από την θέση 1 όπως φαίνεται στην **Εικόνα 5.3 (α)** και στην συνέχεια ένας χειριστής ενεργοποιεί την κάμερα μέσω του προγράμματος του Unity3d στον πρώτο υπολογιστή. Αυτό το βήμα είναι ιδιαίτερα σημαντικό,

καθώς γίνεται η αρχικοποίηση της αρχικής θέσης του χρήστη στην εικονική σκηνή του Unity3d.

Ο χειριστής εκτελεί το πρόγραμμα **chrisfinal( )** στον δεύτερο υπολογιστή. Ο ρομποτικός βραχίονας ξεκινά την κίνηση των 280 μοιρών. Το σενάριο της εφαρμογής αυτής έχει ως εξής: όταν ο ρομποτικός βραχίονας πλησιάζει τον άνθρωπο, εκείνος μπαίνει μέσα στην περιοχή 1 (**Εικόνα 5.3 (γ)**). Τότε, η κάμερα εντοπίζει τον άνθρωπο, στέλνει σήμα μέσω του προγράμματος στο Unity3d στην πλακέτα Arduino και αυτή με τη σειρά της στέλνει σήμα στην είσοδο του ελεγκτή του ρομποτικού βραχίονα.

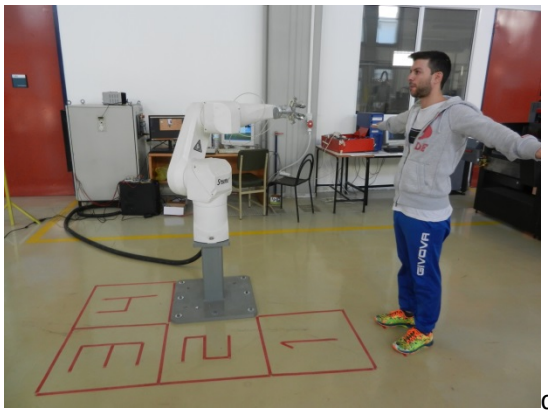
Έτσι, η εντολή **REACTI 1001, subprog1** γίνεται αληθής και καλείται το υποπρόγραμμα **subprog1** (όλοι οι κώδικες βρίσκονται στο **Παράρτημα: Κώδικες – Απλή αποφυγή**).

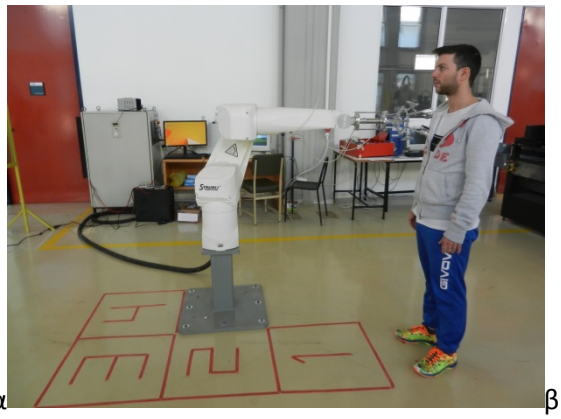
Το υποπρόγραμμα **subprog1** εκτελεί τα 3 στάδια της αποφυγής καθώς και τη συνέχιση της κίνησης στην προκαθορισμένη τελική θέση (όπως φαίνεται στις **Εικόνα 5.3 (δ), (ε), (ζ), (η), (θ)**).

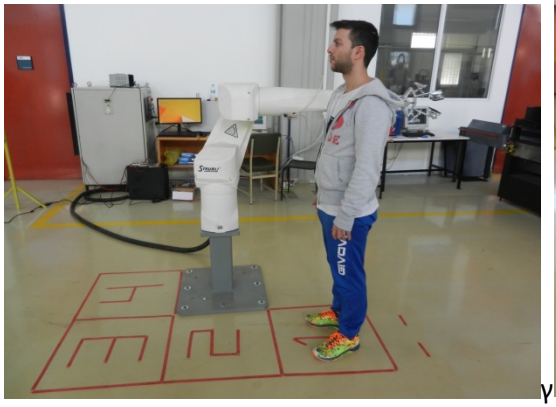
Αρχικά, στο πρώτο στάδο της αποφυγής ο ρομποτικός βραχίονας σταματά και αρχίζει να κινείται προς τα πίσω στο πρώτο σημείο το οποίο βρίσκεται ένα μέτρο περίπου μακριά από την περιοχή 1 (**Εικόνα 5.3 (δ)**). Στη συνέχεια, στο δεύτερο στάδιο της αποφυγής ο ρομποτικός βραχίονας αρχίζει να κινείται κατακόρυφα έτσι ώστε να παρακάμψει ένα όρθιο άνθρωπο ύψους 1,80μ. Οι συντεταγμένες του δεύτερου σημείου είναι τέτοιες ώστε ο ρομποτικός βραχίονας να βρίσκεται σε ασφαλές ύψος αλλά ακόμη να μην έχει αποφύγει τον άνθρωπο (**Εικόνα 5.3 (ε)**). Τέλος, στο τρίτο και τελευταίο στάδιο της αποφυγής ο ρομποτικός βραχίονας αρχίζει και κινείται στο τρίτο σημείο το οποίο βρίσκεται σε ασφαλές ύψος έχοντας αποφύγει πλήρως τον άνθρωπο (**Εικόνα 5.3 (ζ)**). Η συγκεκριμένη εφαρμογή εξετάζει την περίπτωση που ο άνθρωπος εισέρχεται στην περιοχή 1. Επομένως, τα ίδια βήματα ακολουθούνται και για τις υπόλοιπες περιοχές 2, 3 και 4 με κάποιες τροποποιήσεις που φαίνονται στους κώδικες(**Παράρτημα: Κώδικες – Απλή αποφυγή**).

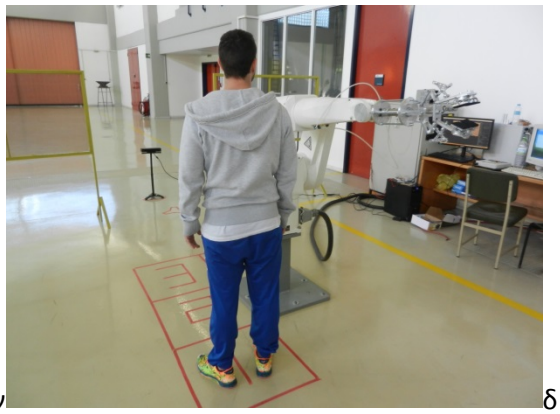
Η βασική τροχιά που θα ακολουθήσει ο ρομποτικός βραχίονας εάν δεν εντοπίσει η κάμερα Kinect τον άνθρωπο θα είναι μια ευθεία κίνηση 280 μοιρών από την αρχική θέση έως την τελική θέση (οι συντεταγμένες των σημείων βρίσκονται στο **Παράρτημα: Κώδικες – Απλή αποφυγή**).

Οι τροχιές που αναλύθηκαν στην παρούσα διπλωματική εργασία χρησιμοποιήθηκαν διότι υπήρχε ελεύθερος χώρος γύρω από τον ρομποτικό βραχίονα, έτσι τα συγκεκριμένα σημεία (αρχικά, τελικά και εκείνα για τα 3 στάδια της αποφυγής) πάρθηκαν αυθαίρετα. Όμως, όπως βλέπει κανείς και από τους κώδικες στο **Παράρτημα: Κώδικες – Απλή αποφυγή** καταλαβαίνει ότι η μεθοδολογία που ακολουθήθηκε γενικεύεται εάν κάποιος θέλει να αλλάξει τα σημεία που χρησιμοποιήθηκαν στη συγκεκριμένη διπλωματική εργασία, ακόμη και να προσθέσει νέα σημεία εάν επρόκειτο να υπάρχουν εμπόδια σε κάποιον άλλον χώρο εργασίας.


α


β


γ


δ


ε


ζ


η

θ

Εικόνα 5.3 : (α) Η αρχική θέση που παίρνει ο άνθρωπος και ο ρομποτικός βραχίονας (β) Ο ρομποτικός βραχίονας πλησιάζει τον άνθρωπο (γ) Ο άνθρωπος εισέρχεται στην περιοχή 1 (δ) Ο ρομποτικός βραχίονας βρίσκεται στην 1η θέση της αποφυγής (ε) Ο ρομποτικός βραχίονας βρίσκεται στην 2η θέση της αποφυγής (ζ) Ο ρομποτικός βραχίονας βρίσκεται στην 3η θέση της αποφυγής (η) Ο ρομποτικός βραχίονας έχει αποφύγει τον άνθρωπο (θ) Ο ρομποτικός βραχίονας βρίσκεται στην τελική του θέση

Ομοίως, γίνεται και η αποφυγή όταν ο χρήστης βρίσκεται στις περιοχές 2, 3 και 4 μέσω των εντολών **REACTI 1002, subprog2**, **REACTI 1003, subprog3**, **REACTI 1004, subprog4** αντίστοιχα.

Ακολουθεί ο κώδικας του βασικού προγράμματος **chrisfinal( )**, ο οποίος έχει γραφτεί σε γλώσσα προγραμματισμού V+ :

PROGRAM chrisfinal( )

SET #startpoint = #PPOINT(140,-67,157,0,0,0) //ορισμός αρχικής θέσης

SET #finalpoint = #PPOINT(-140,-67,157,0,0,0) //ορισμός τελικής θέσης

SPEED 30 ALWAYS // ορισμός ταχύτητας

MOVE #startpoint // κίνηση στην αρχική θέση

BREAK // δεσμεύει την κίνηση στην αρχική θέση

MOVE #finalpoint // κίνηση στην τελική θέση

REACTI 1001, subprog1 // έλεγχος για ύπαρξη σήματος στην είσοδο 1001

REACTI 1002, subprog2 // έλεγχος για ύπαρξη σήματος στην είσοδο 1002

REACTI 1003, subprog3 // έλεγχος για ύπαρξη σήματος στην είσοδο 1003

REACTI 1004, subprog4 // έλεγχος για ύπαρξη σήματος στην είσοδο 1004

Η θέση των εντολών **REACTI** μέσα στον κώδικα δεν παίζει κανένα ρόλο, καθώς η ενεργοποίηση τους γίνεται όταν ο ελεγκτής λάβει σήμα 24V σε κάποια από τις εισόδους του. Έτσι οποιαδήποτε στιγμή κατά την εκτέλεση του προγράμματος η εντολη **REACTI** γίνεται αληθής εάν λάβει σήμα ο ελεγκτής και στη συνέχεια καλείται το αντίστοιχο υποπρόγραμμα ανάλογα με την είσοδο.

Ακολουθεί ο κώδικας του υποπρογράμματος **subprog1( )**, ο οποίος έχει γραφτεί σε γλώσσα προγραμματισμού V+ , και αναφέρεται στην περιοχή 1:

PROGRAM subprog1( )

SET #a = #PPOINT(90,-67,157,0,0,0) //ορισμός της 1ης θέσης της αποφυγής

SET #b = #PPOINT(90,-85,125,0,0,0) // ορισμός της 2ης θέσης της αποφυγής

SET #c = #PPOINT(30,-85,125,0,0,0) // ορισμός της 3ης θέσης της αποφυγής

SET #fp1 = #PPOINT(-140,-67,157,0,0,0) //ορισμός τελικής θέσης

MOVE #a // κίνηση στην 1η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 1η θέση της αποφυγής

MOVE #b // κίνηση στην 2η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 2η θέση της αποφυγής

MOVE #c // κίνηση στην 3η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 3η θέση της αποφυγής

MOVE #fp1 // κίνηση στην τελική θέση εφόσον έχει ενεργοποιηθεί κάποια από τις εντολές REACTI

Οι κώδικες **subprog2( ), subprog3( ), subprog4( )** είναι όμοιοι με τον κώδικα **subprog1( )** και βρίσκονται στο Παράρτημα.

5.3 Προγραμματισμός στο λογισμικό Unity3d

Για την ολοκλήρωση της παρούσας εργασίας χρησιμοποιήθηκε από τη διδακτορική διατριβή του Ηλία Μάτσα [21] το Game Object **"liakous2Avatar"**, το οποίο έχει προσαρτημένο το Component **Avatar Controller(script).** Επιπλέον, χρησιμοποιήθηκε από την διπλωματική εργασία του Σ. Μίχα [44] το Game Object **"Staubli RX90Avatar"** απλά και μόνο για να παρουσιαστούν οι διαστάσεις του ρομποτικού βραχίονα Staubli RX90L στην εικονική σκηνή και με βάση αυτό να σχεδιαστούν οι κύβοι 1, 2, 3 και 4 στην εικονική σκηνή που αντιπροσωπεύουν τις περιοχές 1, 2, 3 και 4 στον πραγματικό χώρο.Στο ιεραρχικό παράθυρο (**Εικόνα 5.4**) φαίνονται όλα τα Game Object της σκηνής :

- **3rd person camera**
- **Base**
- **Liakous2(avatar)**
- **Staubli RX90L**
- **Cube (1)**
- **Cube (2)**
- **Cube (3)**
- **Cube (4)**

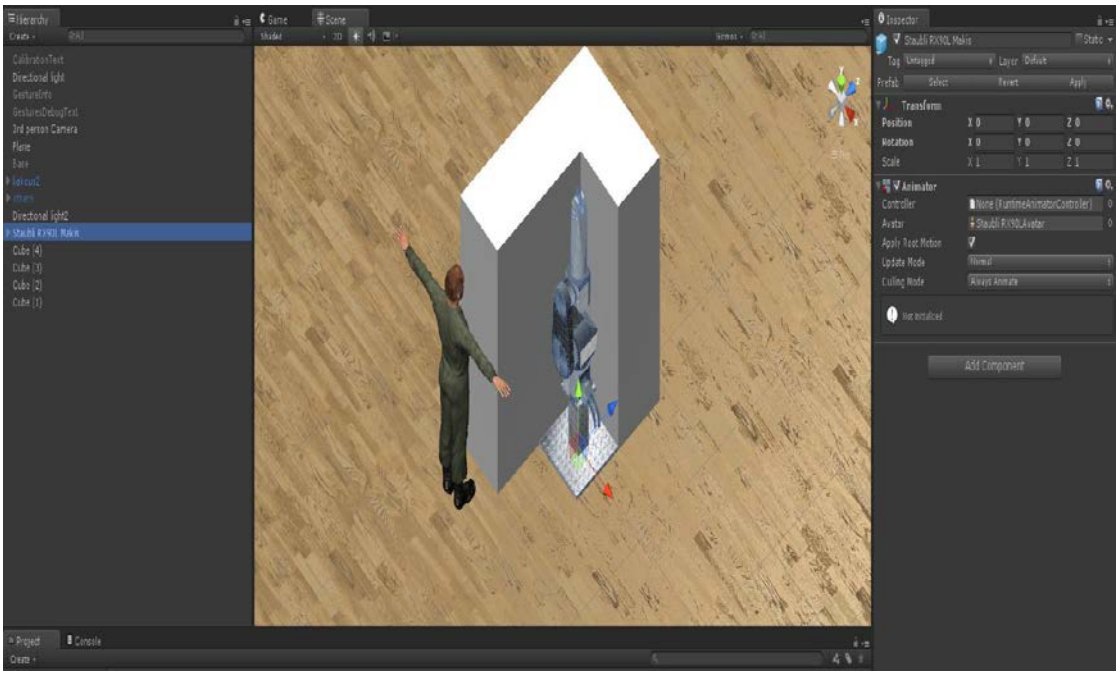
Στην **Εικόνα 5.4** φαίνεται το Game Object **Staubli RX90L**, που οι διαστάσεις του είναι 1m*1m, καθώς και **η βάση της σκηνής(base)**.

Στην **Εικόνα 5.5** φαίνεται το Game Object **κάμερα 3ου προσώπου(3rd person camera)**, όπου παρουσιάζονται τα χαρακτηριστικά της όπως είναι **η θέση της(position), η περιστροφή της(rotation)** και **το μέγεθός της(scale)**.

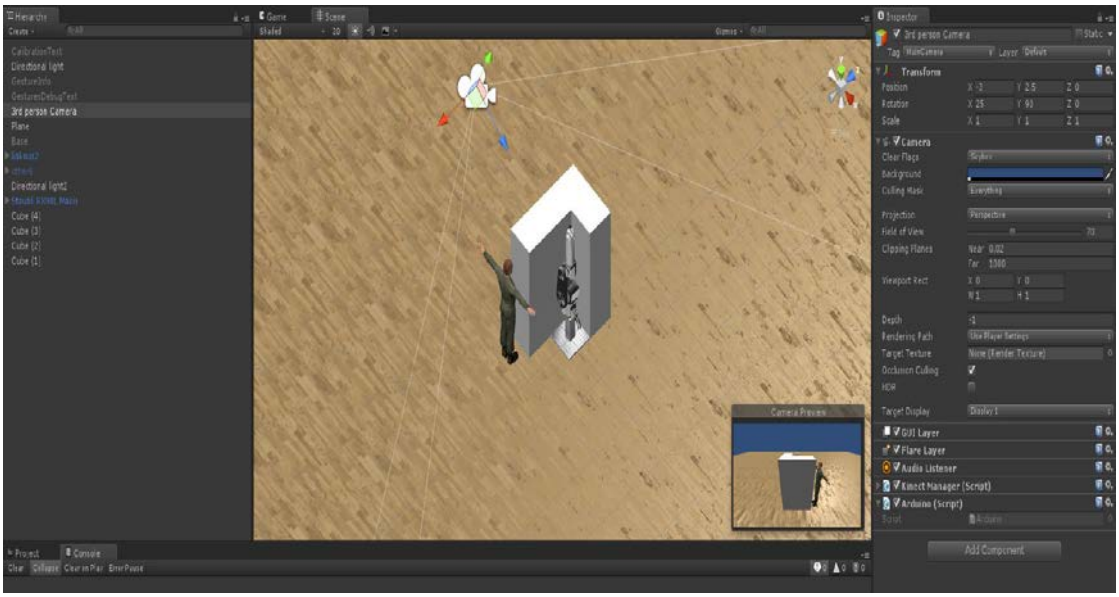
Στην **Εικόνα 5.5** φαίνονται τα δύο Components που είναι προσαρτημένα πάνω στο Game Object **3rd Person camera**. Το πρώτο είναι ο κώδικας **Kinect Manager(C# script)**[4], ο οποίος είναι ο απαραίτητος κώδικας που σχετίζεται με την λειτουργία της κάμερας **Kinect**, ενώ το δεύτερο είναι ο κώδικας **Arduino(C# script)**, που έχει να κάνει με τα σήματα που στέλνονται

[4] Ο κώδικας Kinect Manager αναπτύχθηκε στη διδακτορική διατριβή του Ηλία Μάτσα[21], όπως και το Game Object Liakous2(Avatar)

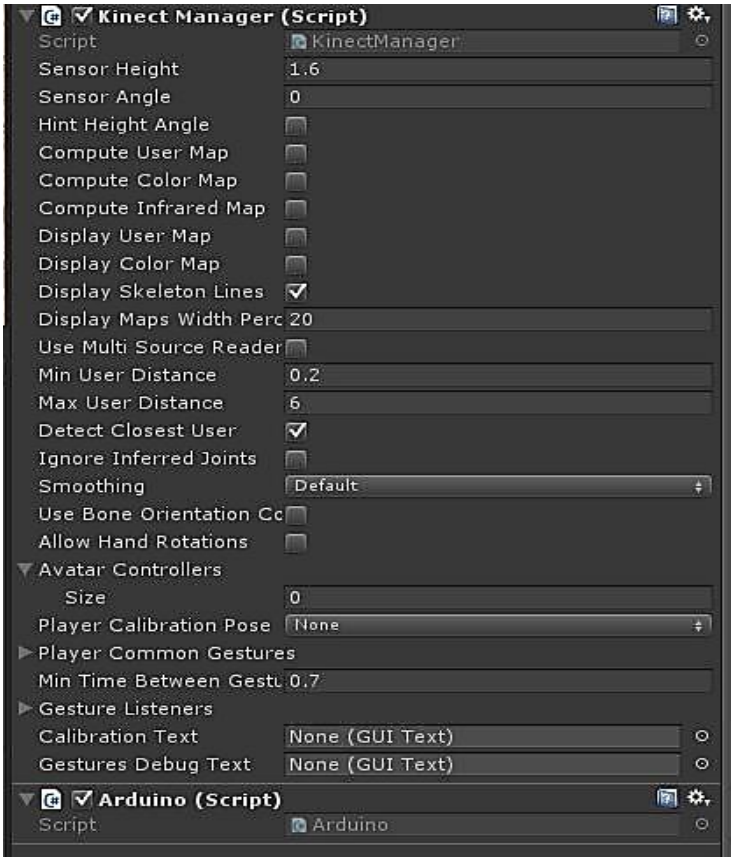
στην **πλακέτα ArduinoMega** όταν η κάμερα εντοπίσει ότι ο άνθρωπος βρίσκεται μέσα σε κάποια από τις περιοχές 1,2,3 ή 4. Όταν ο άνθρωπος βρίσκεται ανάμεσα σε 2 περιοχές (π.χ. τις περιοχές 1 και 2), τότε η κάμερα **Kinect** τον εντοπίζει πρώτα στην περιοχή 1 και μετέπειτα στην περιοχή 2. Επομένως, ο ρομποτικός βραχίονας ενώ αρχίζει να εκτελεί τις 3 αποφυγές για την περιοχή 1, σταματά και αρχίζει να εκτελεί τις 3 αποφυγές για την περιοχή 2. Ομοίως, και για τα άλλα ζευγάρια περιοχών (2-3, 3-4).



Εικόνα 5.4 : Το Game Object Staubli RX90L

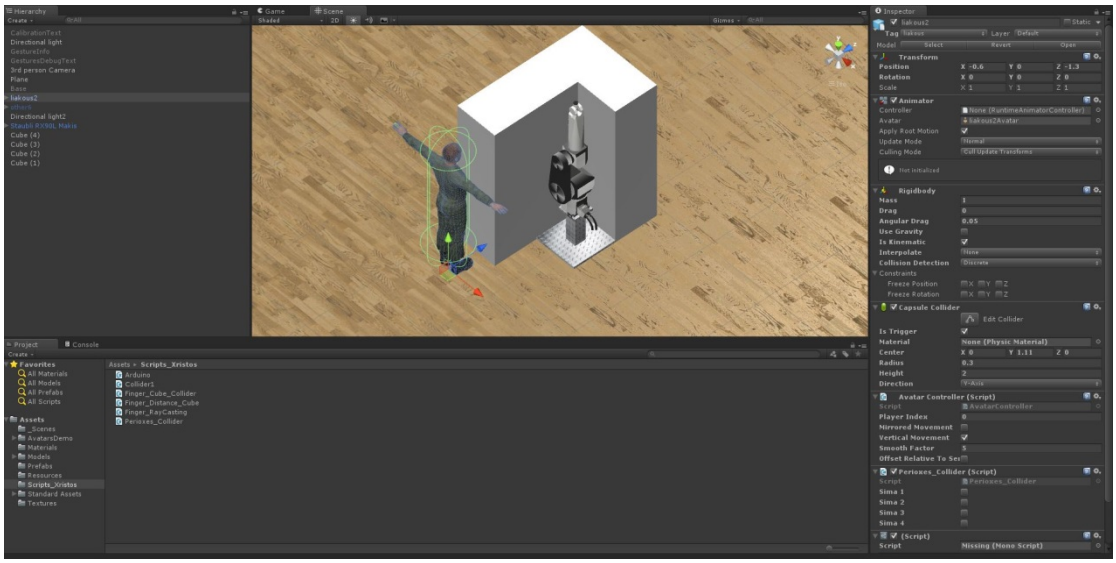


Εικόνα 5.5 : Το Game Object 3rd Person Camera



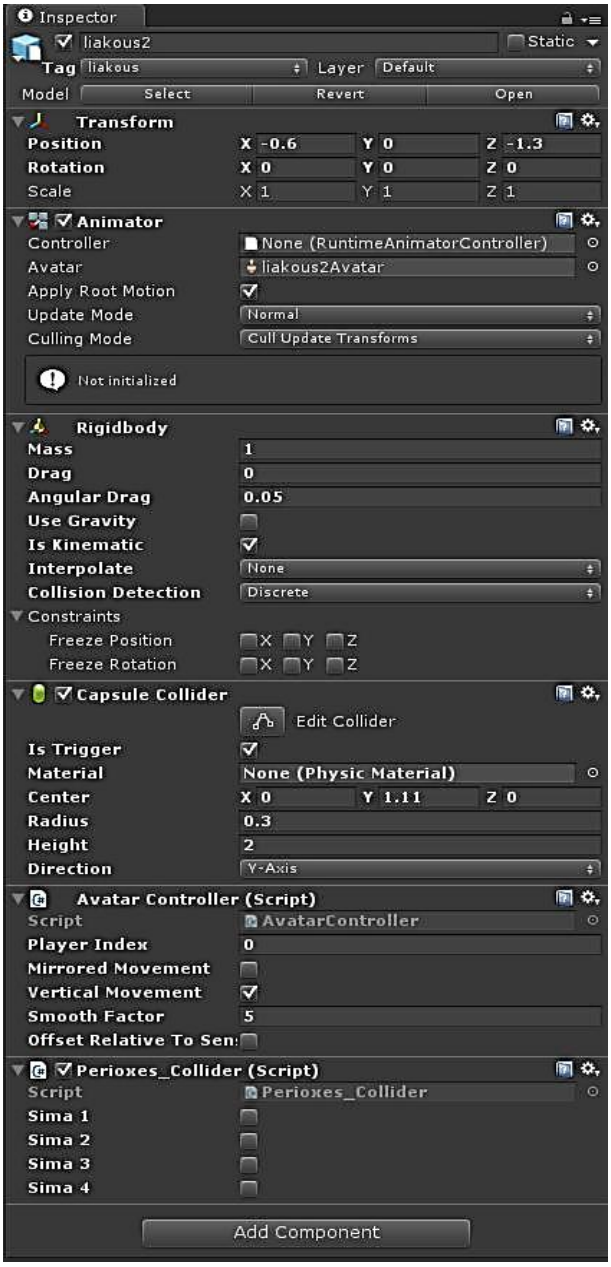
Εικόνα 5.6 : Τα Components Kinect Manager και Arduino

Στην **Εικόνα 5.6** φαίνεται το Game Object **Liakous2(Avatar)**, όπου παρουσιάζονται τα χαρακτηριστικά του συγκεκριμένου Game Object όπως είναι **η θέση του (position)** και **το μέγεθος του (scale)**.



Εικόνα 5.7 : Το Game Object Liakous2(Avatar)

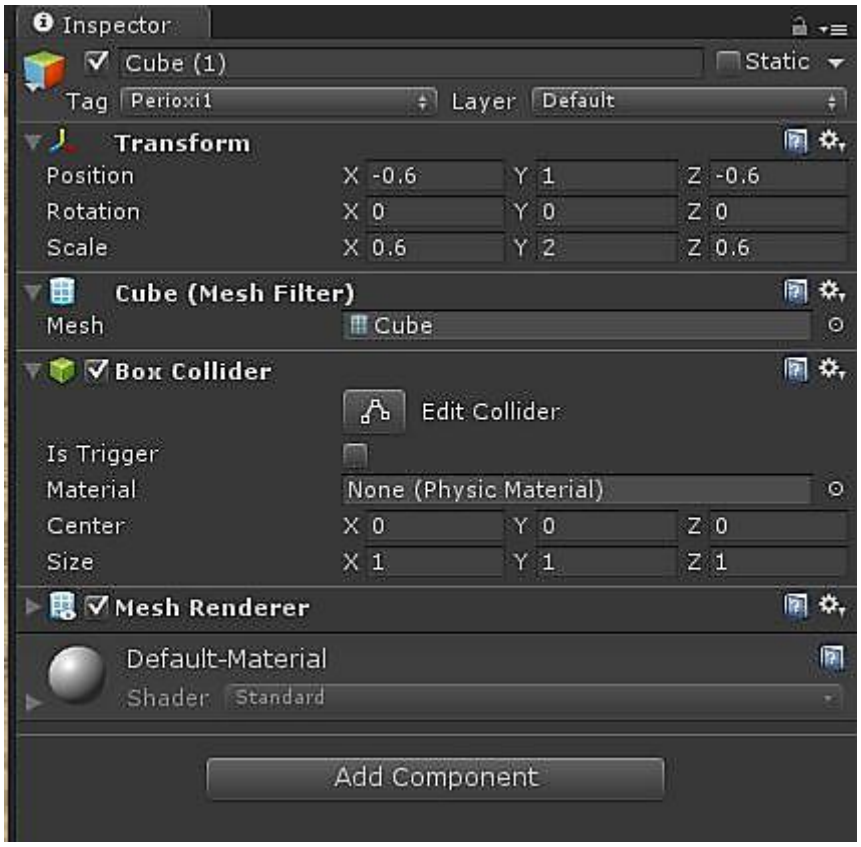
Στην **Εικόνα 5.7** φαίνονται τα **Rigidbody**, **Capsule Collider** καθώς επίσης και ο κώδικας **Avatar Controller(script)**, που συνδέεται με τον έλεγχο του Game Object **Liakous2(Avatar)** [21] και ήταν προσαρτημένος σε αυτό, και ο κώδικας **Perioxes_Collider(script)**, που συνδέεται με τον προγραμματισμό του κώδικα για τον έλεγχο του **Liakous2(Avatar)** όταν εισέρχεται σε κάποιον από τους κύβους 1, 2, 3 και 4, που αντιπροσωπεύουν τις περιοχές 1,2,3 και 4 στον πραγματικό χώρο όπως φαίνονται στην **Εικόνα 5.2**. Τέλος, φαίνεται ότι έχει προστεθεί η ετικέτα(Tag) "liakous", κάτι που χρησιμοποιήθηκε για το γράψιμο του κώδικα και τον έλεγχο της διεπαφής Liakous2(Avatar)-Box Collider(Cubes 1, 2, 3, 4).



Εικόνα 5.8 : Ο inspector του Game Object Liakous2(Avatar)

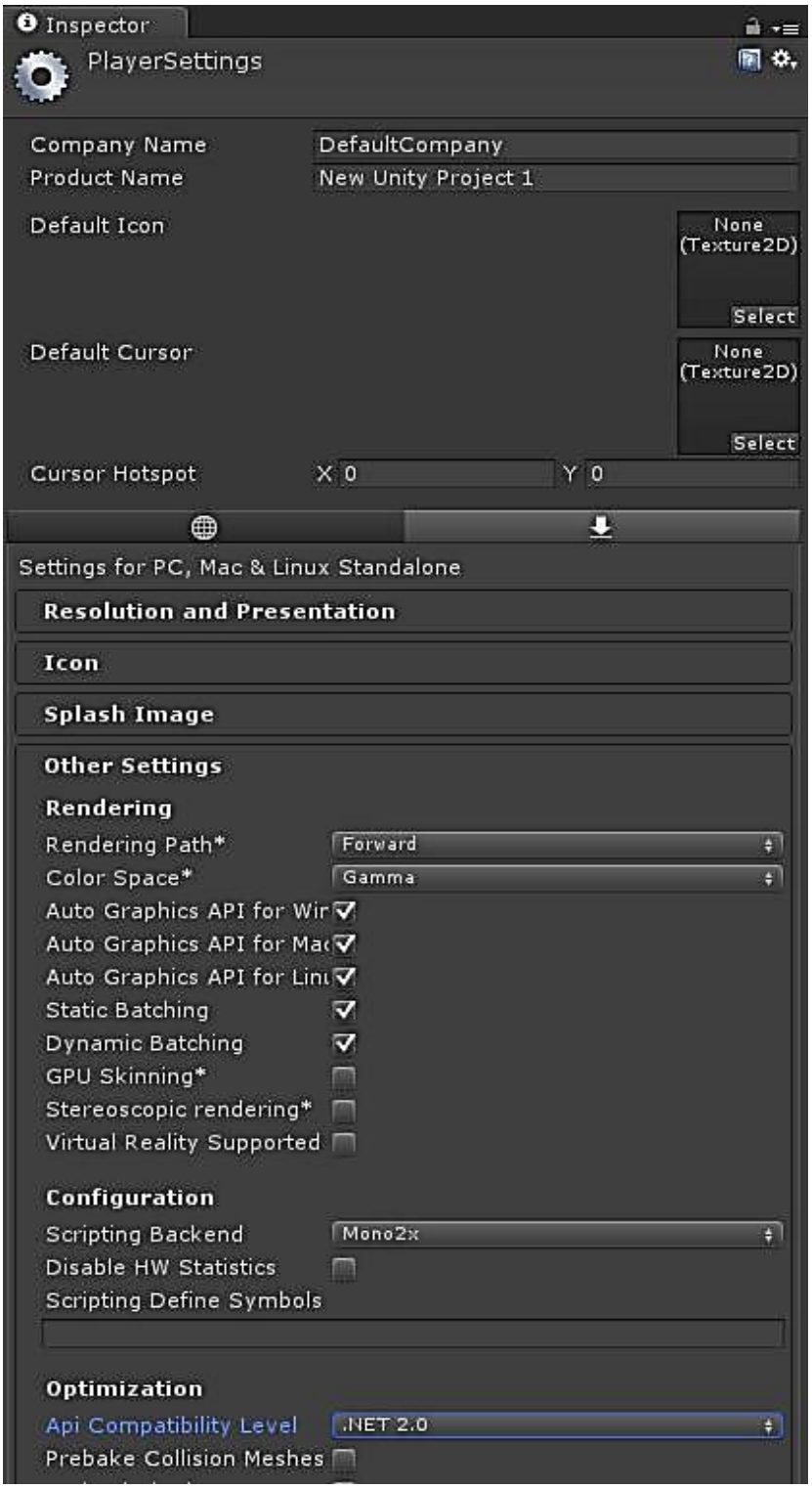
Οι δύο βασικοί κώδικες **Arduino(script)** και **Perioxes_Collider(script)** θα αναλυθούν παρακάτω.

Στην **Εικόνα 5.9** φαίνεται ο inspector του πρώτου κύβου (Cube (1)), όπου παρουσιάζονται **η διάσταση του (scale), η θέση του στον χώρο (position),** καθώς και τα **Box Collider, Mesh Filter, Mesh Renderer Components του** (έχουν αναλυθεί στην **υποενότητα 3.4**). Επίσης, φαίνεται ότι έχει προστεθεί η ετικέτα (Tag) "Perioxi 1", κάτι που χρησιμοποιήθηκε για τον γράψιμο του κώδικα και τον έλεγχο της διεπαφής Liakous2 (Avatar) - Box Collider. Τα ίδια ισχύουν για τους κύβους 2, 3 και 4.



Εικόνα 5.9 : Ο inspector του 1ου κύβου

Σε αυτό το σημείο πρέπει να σημειωθεί ότι είναι αναγκαία η ανάπτυξη της εφαρμογής στο Unity3d που να επικοινωνεί μέσω σειριακής θύρας με την πλακέτα ArduinoMega. Αυτό γίνεται με τα ακόλουθα βήματα. Αρχικά από την γραμμή εργαλειών επιλέγουμε File -> Build Settings -> Player Settings και στο πεδίο Api Compatibility Level αλλάζουμε την προεπιλεγμένη επιλογή σε **.NET 2.0** (βλ. **Εικόνα 5.10**). Αυτό γίνεται καθώς, προκειμένου να δημιουργηθεί αντικείμενο της κλάσης **SerialPort**, πρέπει να χρησιμοποιηθεί το **namespace System.IO.Ports**, το οποίο δεν περιλαμβάνεται στην επιλογή **.NET 2.0 Subset**.



Εικόνα 5.10 : Η επιλογή σε .NET 2.0

Κατά την διάρκεια εκτέλεσης της εφαρμογής δεν υπάρχει δυνατότητα άλλο πρόγραμμα στον υπολογιστή να χρησιμοποιήσει την θύρα **COM 4** που χρησιμοποιήθηκε στον κώδικα **Arduino(script).**

Ακολουθεί η ανάλυση των βασικών προγραμμάτων **Arduino( )** , **Perioxes_Collider( ).**

Program Perioxes_Collider( )

```
void Start ( )                    {
        sima1 = false;                    //ορισμός της μεταβλητής sima1 ως ψευδής
        sima2 = false;                    //ορισμός της μεταβλητής sima2 ως ψευδής
        sima3 = false;                    //ορισμός της μεταβλητής sima3 ως ψευδής
        sima4 = false;          }         //ορισμός της μεταβλητής sima4 ως ψευδής
void OnTriggerEnter(Collider other)       {
        if (other.gameObject.tag == "Perioxi1")           {
                sima1 = true;             //ορισμός της μεταβλητής sima1 ως αληθής
                Debug.Log ("Eimai stin perioxi 1");       }
        if (other.gameObject.tag == "Perioxi2")           {
                sima2 = true;             //ορισμός της μεταβλητής sima2 ως αληθής
                Debug.Log ("Eimai stin perioxi 2");       }
        if (other.gameObject.tag == "Perioxi3")           {
                sima3 = true;             //ορισμός της μεταβλητής sima3 ως αληθής
                Debug.Log ("Eimai stin perioxi 3");       }
        if (other.gameObject.tag == "Perioxi4")           {
                sima4 = true;             //ορισμός της μεταβλητής sima4 ως αληθής
                Debug.Log ("Eimai stin perioxi 4");       }
}
```

Η συνάρτηση **OnTriggerEnter(Collider other)** καλείται όταν η Collider other γίνεται Trigger(enters the Trigger). Αυτό το μήνυμα στέλνεται στην Trigger Collider και στο Rigidbody στο οποίο η Trigger Collider ανήκει, καθώς και στο Rigidbody το οποίο έρχεται σε επαφή με την Trigger.

Στην ουσία ενεργοποιείται ο **Collider** του **liakous2** (γίνεται Triggered), στη συνέχεια ελέγχεται εάν ο άνθρωπος(το Game Object **liakous2** στην εικονική σκηνή) έχει μπει στην περιοχή 1 στον πραγματικό χώρο (περιοχή με την ετικέτα **"Perioxi 1"** στην εικονική σκηνή) (**if (other.gameObject.tag == "Perioxi1")**). Εάν βρίσκεται μέσα στην συγκεκριμένη περιοχή (δηλαδή υπάρχει επαφή στους **Colliders** του **liakous2**{ετικέτα **"liakous"**} και **Cube (1)**{ετικέτα **"Perioxi 1"**}, τότε εκτελούνται οι εντολές **sima1 = true, Debug.Log ("Eimai stin perioxi 1"),** δηλαδή το **sima1** γίνεται αληθές και εμφανίζεται στην κονσόλα η ένδειξη **Eimai stin perioxi 1.** Όμοια γίνεται ο έλεγχος και για τις υπόλοιπες περιοχές.

Program Arduino( )

```
using UnityEngine;
using SystemCollections;
using System.IO.Ports;                    //ορισμός της βιβλιοθήκης System.IO.Ports
public class Arduino : MonoBehaviour      {       //ορισμός κλάσης Arduino
// Αρχικοποίηση
private SerialPort sp;                    //ορισμός μεταβλητής sp ως σειριακή θύρα
void Start ( )            {
        sp = new SerialPort("COM4", 9600);        //ορισμός σειριακής θύρας sp
        sp.Open ( );   }                          //ενεργοποίηση της θύρας sp
void Update ( )           {
        char[ ] s = new char[ 1 ];                //ορισμός πίνακα τύπου χαρακτήρα
        bool person_found = false;                //ορισμός person_found ως ψευδής
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Peri
        oxes_Collider>( ).sima1)      {
        person_found = true;                      //ορισμός person_found ως αληθής
        s[ 0 ] = "1" ;                            //ορισμός του s[ 0 ] ως ο χαρακτήρας 1
        sp.Write(s, 0 , 1);       }               //αποστολή του s στην σειριακή θύρα sp
```

```
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Perioxes_Collider>( ).sima2)     {
        person_found = true;                    //ορισμός person_found ως αληθής
        s[ 0 ] = "2" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 2
        sp.Write(s, 0 , 1);       }             //αποστολή του s στην σειριακή θύρα sp
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Perioxes_Collider>( ).sima3)     {
        person_found = true;                    //ορισμός person_found ως αληθής
        s[ 0 ] = "3" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 3
        sp.Write(s, 0 , 1);       }             //αποστολή του s στην σειριακή θύρα sp
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Perioxes_Collider>( ).sima4)     {
        person_found = true;                    //ορισμός person_found ως αληθής
        s[ 0 ] = "4" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 4
        sp.Write(s, 0 , 1);       }             //αποστολή του s στην σειριακή θύρα sp
        if(!person_found)     {         //έλεγχος για το αν το person_found είναι ψευδής
        s[ 0 ] = "0" ;                  //ορισμός του s[ 0 ] ως ο χαρακτήρας 0
        sp.Write(s, 0 , 1);       }     //αποστολή του s στην σειριακή θύρα sp
        }
}
```

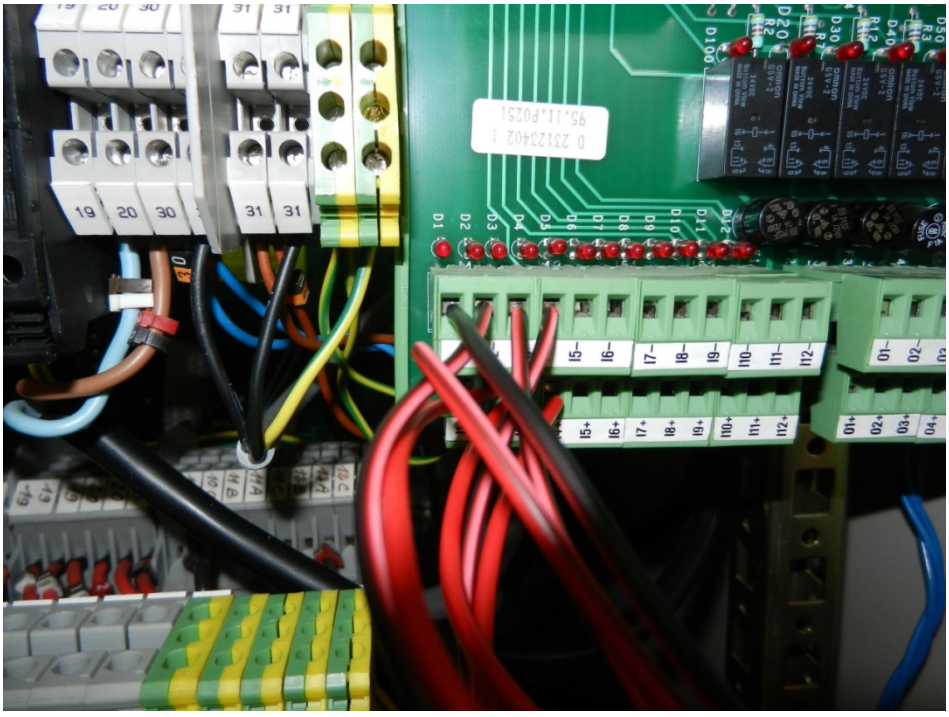
Η εντολή **if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Perioxes_Collider>( ).sima1)** είναι η πιο βασική εντολή του συγκεκριμένου κώδικα, καθώς ελέγχει εάν στο Game Object που έχει την ετικέτα(Tag) **"liakous"**, και συγκεκριμένα στο Component του **Perioxes_Collider(scipt)**, είναι αληθές το **sima1.** Επομένως, η εντολή αυτή τρέχει τον κώδικα **Perioxes_Collider** και εάν το **sima1** είναι αληθές τότε εκτελεί τις εντολές **person_found = true, s[0] = "1" , sp.Write(s, 0 , 1)** , εάν το **sima1** είναι ψευδές τότε προχωράει στην επόμενη **if.** Ομοίως συμβαίνει και για τις επόμενες **if** με τα αντίστοιχα σήματα **sima2, sima3** και **sima4.**

Τέλος, εάν η μεταβλητή **person_found** είναι ψευδής, τότε εκτελούνται οι εντολές **s[0] = "0" , sp.Write(s, 0 , 1).**

Στην συνέχεια, θα αναλυθεί ο κώδικας που προγραμματίστηκε για τον έλεγχο της πλακέτας **ArduinoMega**, η οποία επικοινωνεί με το λογισμικό Unity3d μέσω της σειριακής θύρας **COM4** του ηλεκτρονικού υπολογιστή καθώς και με τον ελεγκτή του ρομποτικού βραχίονα **Staubli RX90L.**

5.4 Προγραμματισμός ArduinoMega

Στο κεφάλαιο αυτό θα αναλυθεί το ηλεκτρικό κύκλωμα που χρησιμοποιήθηκε για την επικοινωνία της πλακέτας ArduinoMega με τον ελεγκτή του ρομποτικού βραχίονα Staubli RX90L, καθώς και ο κώδικας που προγραμματίστηκε στο λογισμικό Arduino IDE.

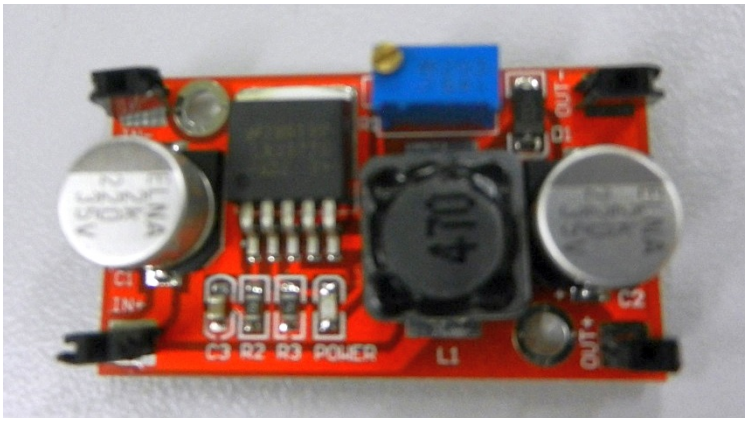


Εικόνα 5.11 : Είσοδοι του ελεγκτή του ρομποτικού βραχίονα(I1-I4)

Αρχικά, είναι απαραίτητο να τονιστεί ότι οι εισόδοι του ελεγκτή του ρομποτικού βραχίονα Staubli RX90L(βλ. **Εικόνα 5.11**) παραλαμβάνουν σήματα 24 Volts, επομένως χρησιμοποιήθηκαν 4 **Step-Up Module**[5] (βλ.

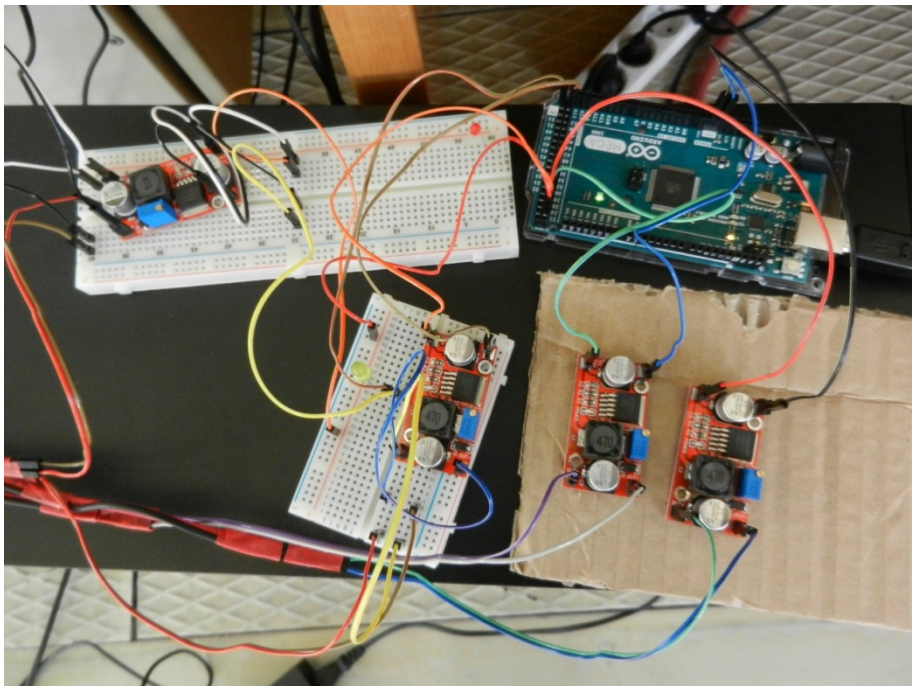
[5] Πρόκειται για μια πλακέτα η οποία παραλαμβάνει είσοδο(3.3-5 Volts), την οποία μετατρέπει μέσω ενός ρυθμιστή τάσης σε έξοδο (5 – 35 Volts).

Εικόνα 5.12) για την μετατροπή του σήματος (που παραλαμβάνουν από την πλακέτα ArduinoMega) σε 24 Volts, τα οποία δέχεται ως είσοδο ο ελεγκτής.

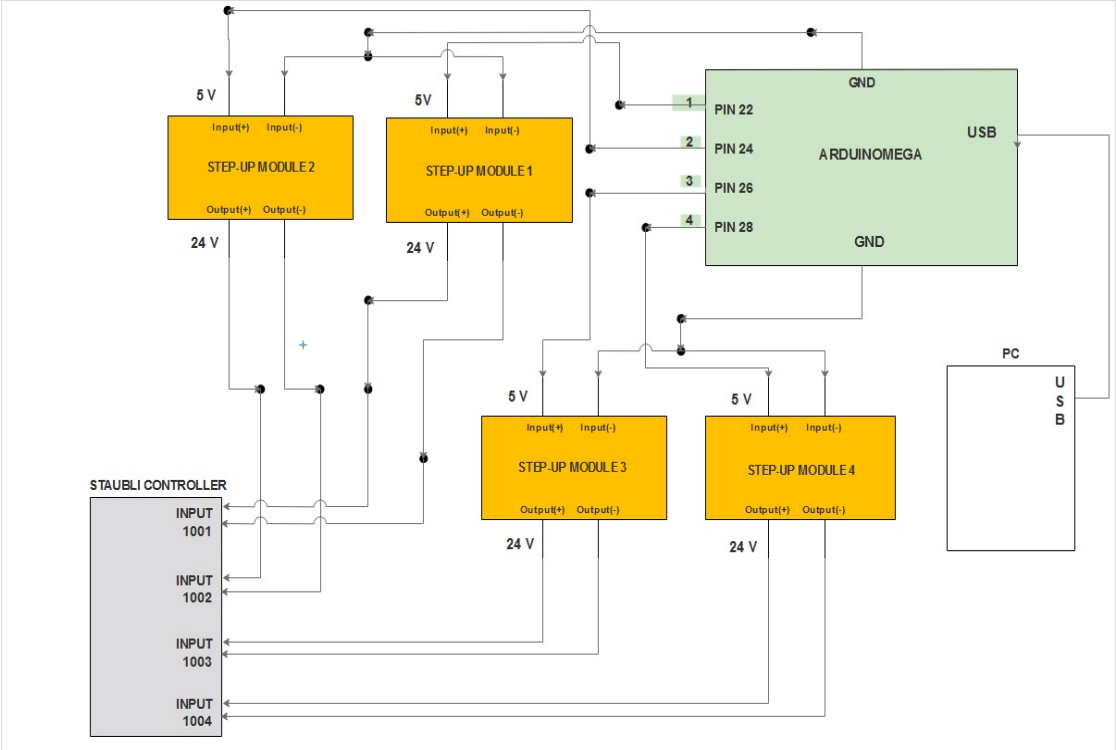


Εικόνα 5.12 : Step-Up Module

Το κύκλωμα που χρησιμοποιήθηκε για την επικοινωνία της πλακέτας ArduinoMega με τον ελεγκτή του ρομποτικού βραχίονα φαίνεται στην **Εικόνα 5.13**, ενώ στην **Εικόνα 5.14** φαίνεται η σχηματική απεικόνιση του ηλεκτρικού κυκλώματος. Στο ηλεκτρικό κύκλωμα παρατηρείται ότι υπάρχουν τα 4 **Step-Up Module** τα οποία μετατρέπουν το σήμα και στη συνέχεια αυτό οδηγείται στον ελεγκτή του ρομποτικού βραχίονα. Τα σήματα έρχονται από τα πινάκια της πλακέτας ArduinoMega 22, 24, 26 και 28 (για τα τις εισόδους του ελεγκτή I1, I2, I3 και I4 αντίστοιχα).



Εικόνα 5.13 : Το ηλεκτικό κύκλωμα επικοινωνίας ArduinoMega με τον ελεγκτή



Εικόνα 5.14 : Σχηματική απεικόνιση του ηλεκτρικού κυκλώματος

Ακολουθεί η ανάλυση του κώδικα της πλακέτας ArduinoMega που προγραμματίστηκε σε C στο λογισμικό Arduino IDE.

Finalprogram( )

```
#inlude <SoftwareSerial.h>

void setup( )          {          //συνάρτηση setup( )
      int pin1 = 22;               //ορισμός για το πινάκι 22 της πλακέτας ως pin1
      int pin2 = 24;               //ορισμός για το πινάκι 24 της πλακέτας ως pin2
      int pin3 = 26;               //ορισμός για το πινάκι 26 της πλακέτας ως pin3
      int pin4 = 28;               //ορισμός για το πινάκι 28 της πλακέτας ως pin4
      Serial.begin (9600);               //ορισμός του baudrate ως 9600
      pinMode (pin1, OUTPUT);            //ορισμός της μεταβλητής pin1 ως έξοδο
      pinMode (pin2, OUTPUT);            //ορισμός της μεταβλητής pin2 ως έξοδο
      pinMode (pin3, OUTPUT);            //ορισμός της μεταβλητής pin3 ως έξοδο
      pinMode (pin4, OUTPUT);    }       //ορισμός της μεταβλητής pin4 ως έξοδο
void loop( )           {                //συνάρτηση loop( )
```

```
        if (Serial.available( ) > 0)     {        //έλεγχος για τα σειριακά δεδομένα
                char incoming_char = (char)Serial.read ( );
                if (incoming_char == '1')    {
                        writeToPin1( );        }//κάλεσμα της συνάρτησης writeToPin1()
                if (incoming_char == '2')    {
                        writeToPin2( );        }//κάλεσμα της συνάρτησης writeToPin2()
                if (incoming_char == '3')    {
                        writeToPin3( );        }//κάλεσμα της συνάρτησης writeToPin3()
                if (incoming_char == '4')    {
                        writeToPin4( );        }//κάλεσμα της συνάρτησης writeToPin4()
                if (incoming_char == '0')    {
                        nullAllPins( );        }//κάλεσμα της συνάρτησης nullAllPins( )
        }                       //τέλος της if(Serial.available ( ) > 0)
}                               //τέλος της συνάρτησης void loop
void writeToPin1( )             {       //συνάρτηση writeToPin1( )
        nullAllPins( );                 //κάλεσμα της συνάρτησης nullAllPins( )
digitalWrite(pin1, HIGH);       }       //μεταφορά στο pin1 τάσης 5 Volts

void writeToPin2( )             {       //συνάρτηση writeToPin2( )
        nullAllPins( );                 //κάλεσμα της συνάρτησης nullAllPins( )
digitalWrite(pin2, HIGH);       }       //μεταφορά στο pin2 τάσης 5 Volts

void writeToPin3( )             {       //συνάρτηση writeToPin3( )
        nullAllPins( );                 //κάλεσμα της συνάρτησης nullAllPins( )
digitalWrite(pin3, HIGH);       }       //μεταφορά στο pin3 τάσης 5 Volts

void writeToPin4( )             {       //συνάρτηση writeToPin4( )
```

```
        nullAllPins( );                 //κάλεσμα της συνάρτησης nullAllPins( )
digitalWrite(pin4, HIGH); }             //μεταφορά στο pin4 τάσης 5 Volts

void nullAllPins( )             {       // συνάρτηση nullAllPins( )
digitalWrite(pin1, LOW);                //μεταφορά στο pin1 τάσης 0 Volts
digitalWrite(pin2, LOW);                //μεταφορά στο pin2 τάσης 0 Volts
digitalWrite(pin3, LOW);                //μεταφορά στο pin3 τάσης 0 Volts
digitalWrite(pin4, LOW); }              //μεταφορά στο pin4 τάσης 0 Volts
```

Στην αρχή του προγράμματος(συνάρτηση **setup( )**) παρατηρείται ότι γίνεται ο ορισμός μεταβλητών για τα πινάκια 22, 24, 26 και 28, ο ορισμός του ρυθμού που στέλνονται τα δεδομένα από την πλακέτα ArduinoMega(baudrate), καθώς και ο ορισμός για τα πινάκια 22, 24, 26 και 28 ως έξοδοι της πλακέτας ArduinoMega. Όλα τα παραπάνω αναλύθηκαν στη **υποενότητα 4.2.** Στην συνάρτηση **loop( )** ελέγχεται εάν έχουν σταλεί δεδομένα μέσω της σειριακής θύρας USB (**if (Serial.available( ) > 0)**) και στη συνέχεια γίνεται ο ορισμός της μεταβλητής **incoming_char**, η οποία ορίζεται ως ο χαρακτήρας που λαμβάνει η πλακέτα ArduinoMega από τη σειριακή θύρα USB(**char incoming_char = (char)Serial.read ( )**) δηλαδή από το λογισμικό Unity3d. Εάν ο επερχόμενος χαρακτήρας είναι ο αριθμός **1**, τότε καλείται η συνάρτηση **writeToPin1( )**. Όμοια, εάν οι επερχόμενοι χαρακτήρες είναι οι αριθμοί **2**, **3** και **4** καλούνται οι συναρτήσεις **writeToPin2( )**, **writeToPin3( )**, **writeToPin4( )**. Εάν ο επερχόμενος χαρακτήρας είναι ο αριθμός **0** καλείται η συνάρτηση **nullAllPins( ).**

Η συνάρτηση **writeToPin1( )** καλεί την συνάρτηση **nullAllPins( )** και στη συνέχεια στέλνει τάση **5 Volts** στο pin1(**digitalWrite(pin1, HIGH)**). Όμοίως, για τις συναρτήσεις **writeToPin2( )**, **writeToPin3( )** και **writeToPin4( )**.

Τέλος, η συνάρτηση **nullAllPins( )** στέλνει τάση **0 Volts** στα pin1, pin2, pin3 και pin4(**digitalWrite(pin1, LOW)**, **digitalWrite(pin2, LOW)**, **digitalWrite(pin3, LOW)**, **digitalWrite(pin4, LOW)**).

Καταλαβαίνουμε, λοιπόν, ότι όταν ο άνθρωπος έχει εισέλθει, για παράδειγμα, στην περιοχή 1, η κάμερα Kinect το αντιλαμβάνεται και έτσι μέσω του κώδικα

στο λογισμικό Unity3d στέλνεται ο χαρακτήρας 1 στην πλακέτα ArduinoMega. Όπως αναφέρθηκε παραπάνω, υπάρχουν σειριακά δεδομένα, η πλακέτα ArduinoMega τα ελέγχει, και επειδή ο χαρακτήρας που λαμβάνει είναι ο αριθμός 1 καλεί την συνάρτηση **writeToPin1( )**, η οποία με τη σειρά της στέλνει στην έξοδο(πινάκι 22 στην περίπτωση αυτή) τάση **5 Volts**. Η τάση αυτή μετατρέπεται σε **24 Volts** μέσω του **Step-Up Module** και κατευθύνεται στην είσοδο του ελεγκτή του ρομποτικού βραχίονα(είσοδος **I1** **Εικόνα 5.17**). Αυτό έχει ως αποτέλεσμα την παράκαμψη του προγράμματος **chrisfinal( )** καλώντας το υποπρόγραμμα **subprog1**(επειδή ο χρήστης βρίσκεται στην συγκεκριμένη περίπτωση στην περιοχή 1) μέσω της εντολής **REACTI 1001, subprog1** (το σήμα στην είσοδο 1001 είναι αληθές).

6

Σύνθετη αποφυγή του ανθρώπου από το ρομπότ

6.1 Σκοπός και Προσέγγιση

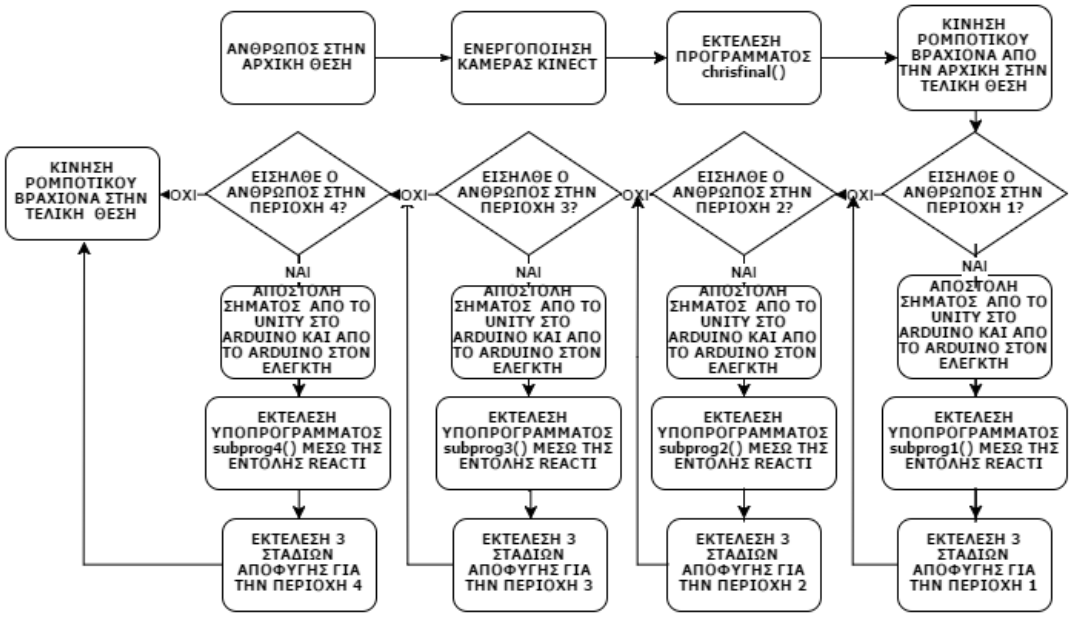
Ο σκοπός της δεύτερης εφαρμογής είναι ίδιος με της πρώτης, με τη διαφορά ότι πραγματοποιείται δεύτερη αποφυγή κατά τη διάρκεια της εκτέλεσης της πρώτης αποφυγής. Δηλαδή, έστω ότι ο άνθρωπος βρίσκεται ήδη στην περιοχή 1 και πραγματοποιείται η αποφυγή για την συγκεκριμένη περιοχή (**βλ. Κεφάλαιο 5, Εικόνα 5.6**). Κατά την εκτέλεση της αποφυγής ο χρήστης εισέρχεται στην περιοχή 2, έτσι ο ρομποτικός βραχίονας Staubli RX90L το αντιλαμβάνεται και πραγματοποιεί την αποφυγή για την περιοχή 2. Τα προγράμματα του λογισμικού Unity3d, καθώς και της πλακέτας ArduinoMega παραμένουν ίδια. Η διαφορά με την πρώτη εφαρμογή είναι οι κώδικες που αφορούν τον προγραμματισμό του ρομποτικού βραχίονα Staubli RX90L.

6.2 Προγραμματισμός ρομποτικού βραχίονα

Στην **Εικόνα 6.1** παρουσιάζεται το διάγραμμα ροής για την 2η εφαρμογή, όπου φαίνονται όλες οι αλληλεπιδράσεις μεταξύ ανθρώπου και ρομποτικού βραχίονα.

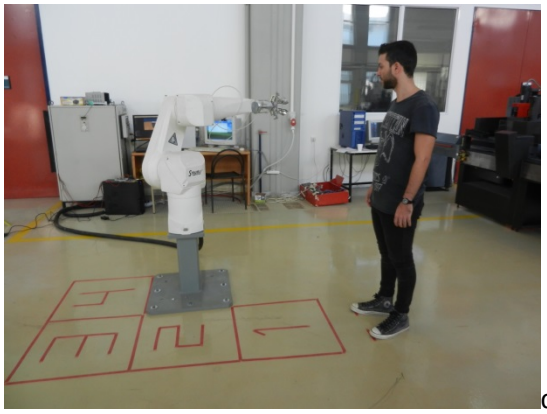
Για την ολοκλήρωση της συγκεκριμένης εφαρμογής έπαιξε σημαντικό ρόλο η εντολή **RETURN** (πέραν της εντολής **REACTI**), η οποία όταν χρησιμοποιείται σε κάποιο σημείο του κώδικα ενός υποπρογράμματος διακόπτει την εκτέλεση του υποπρογράμματος και επιστρέφει στην τελευταία

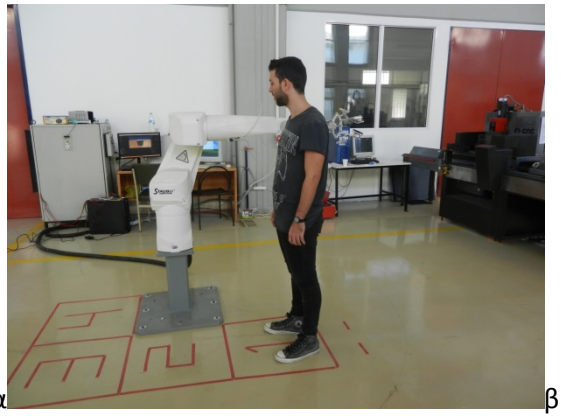
εντολή του κυρίως προγράμματος. Εφόσον εκτελείται το κυρίως πρόγραμμα **chrisfinal( )**, ο χρήστης εισέρχεται στην περιοχή 1(**Εικόνα 6.2 (α), (β)**) και πραγματοποιείται η αποφυγή για την συγκεκριμένη περιοχή καλώντας το υποπρόγραμμα **subprog1( )** (**Εικόνα 6.2 (γ)**). Κατά την διάρκεια της εκτέλεσης του υποπρογράμματος ο χρήστης εισέρχεται στην περιοχή 2(**Εικόνα 6.2 (δ)**), επομένως χρησιμοποιώντας την εντολή **RETURN** επιστρέφουμε στο κυρίως πρόγραμμα, στο οποίο το σήμα **1002** είναι αληθές και έτσι εκτελείται η εντολή **REACTI 1002, subprog2** καλώντας το υποπρόγραμμα **subprog2( )**. Έτσι, πραγματοποιείται η αποφυγή για την περιοχή 2 (**Εικόνα 6.2 (ε), (ζ), (η), (θ), (ι)**) (όλοι οι κώδικες βρίσκονται στο **Παράρτημα : Κώδικες – Σύνθετη αποφυγή**).

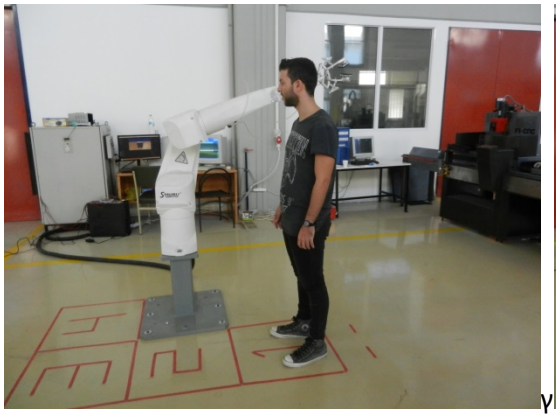


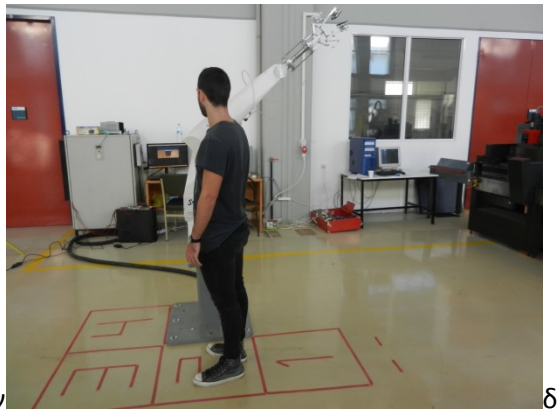
Εικόνα 6.1 : Διάγραμμα ροής 2ης εφαρμογής

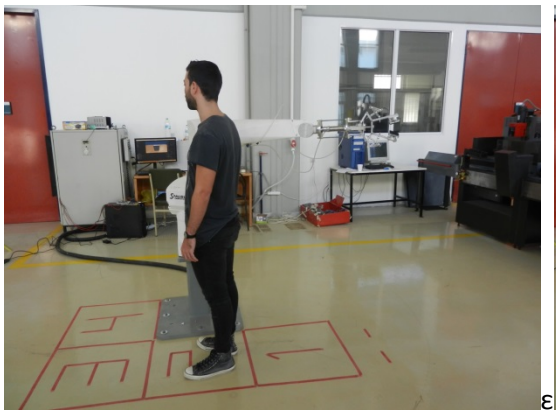
Αυτό μπορεί να πραγματοποιηθεί εάν ο άνθρωπος εισέλθει είτε στην περιοχή 2, είτε στην περιοχή 3, είτε στην περιοχή 4 και γενικότερα σε οποιαδήποτε περιοχή με την προϋπόθεση ότι το ρομπότ βρίσκεται σε άλλη περιοχή πριν από αυτές και πρόκειται να μπεί σε αυτές στη συνέχεια της πορείας του. Για παράδειγμα, για την περιοχή 2 ο χρήστης μπορεί να εισέλθει είτε στην περιοχή 3, είτε στην περιοχή 4. Τέλος, για την περιοχή 3 ο χρήστης μπορεί να εισέλθει στην περιοχή 4.

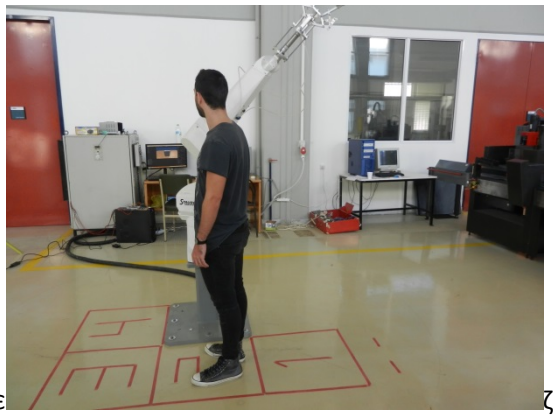

α

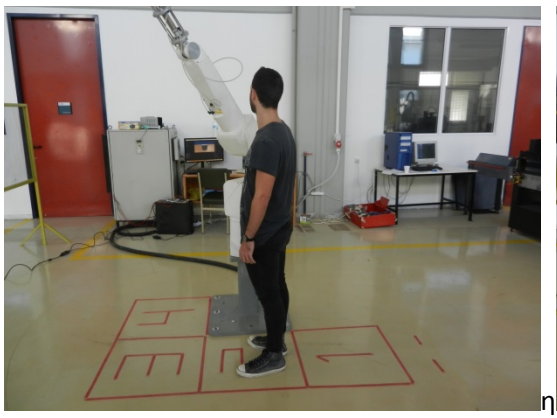

β

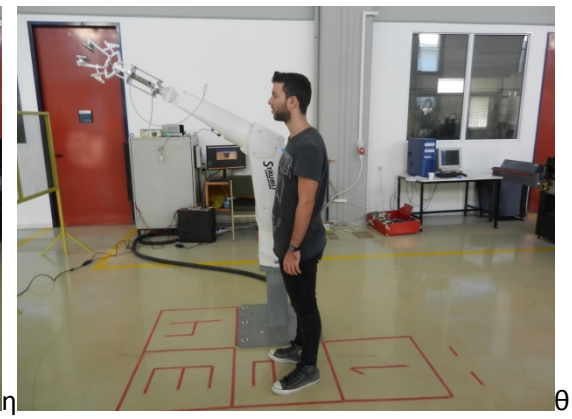

γ

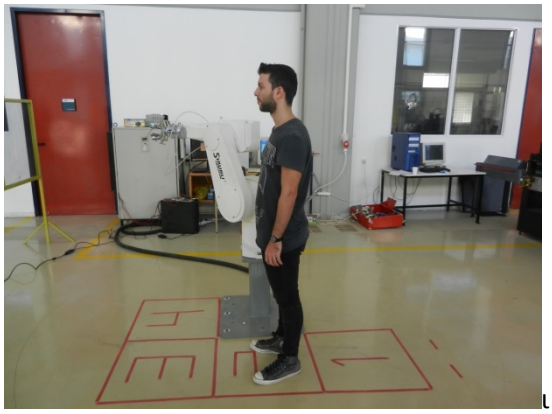

δ


ε


ζ


η


θ



Εικόνα 6.2 : (α) Η αρχική θέση του ανθρώπου και του ρομποτικού βραχίονα (β) Ο άνθρωπος εισέρχεται στην περιοχή 1 (γ) Ο ρομποτικός βραχίονας πηγαίνει στην 1η θέση της 1ης αποφυγής (δ) Ο χρήστης εισέρχεται στην περιοχή 2 (ε) Ο ρομποτικός βραχίονας βρίσκεται στην 1η θέση της 2ης αποφυγής (ζ) Ο ρομποτικός βραχίονας βρίσκεται στην 2η θέση της 2ης αποφυγής (η) Ο ρομποτικός βραχίονας βρίσκεται στην 3η θέση της 2ης αποφυγής (θ) Ο ρομποτικός βραχίονας πηγαίνει προς την τελική θέση (ι) Ο ρομποτικός βραχίονας βρίσκεται στη τελική θέση

Στη συγκεκριμένη ανάλυση παρουσιάζονται τα βήματα που ακολουθούνται όταν ο άνθρωπος εισέρχεται αρχικά στην Περιοχή 1. Ο άνθρωπος όμως μπορεί να εισέλθει είτε στην Περιοχή 2 είτε στην Περιοχή 3 και στη συνέχεια τα βήματα που ακολουθούνται είναι ίδια με αυτά που παρουσιάστηκαν για την Περιοχή 1 παραπάνω. Ο άνθρωπος μπορεί να εισέλθει και στην Περιοχή 4 αρχικά, αλλά επειδή είναι η τελευταία θέση ο κώδικας θα εκτελέσει μόνο τα 3 στάδια αποφυγής για τη θέση αυτή και στη συνέχεια θα σταματήσει μόλις φτάσει στην τελική θέση.

Ο κώδικας του βασικού προγράμματος **chrisfinal2( )** είναι ίδιος με το **chrisfinal( )**. Τα υποπρογράμματα **subprog1(), subprog2(), subprog3()** και **subprog4()** είναι διαφορετικά από ότι στην πρώτη εφαρμογή.

Παρακάτω ακολουθεί ο κώδικας του πρώτου υποπρογράμματος **subprog1()** για την δεύτερη εφαρμογή(τα υποπρογράμματα **subprog2( ), subprog3( )** και **subprog4( )** βρίσκονται στο **Παράρτημα : Κώδικες – Σύνθετη αποφυγή**)

PROGRAM SUBPROG1( )

SET #a = #PPOINT(90,-67,157,0,0,0) //ορισμός της 1ης θέσης της αποφυγής

SET #b = #PPOINT(90,-85,125,0,0,0) // ορισμός της 2ης θέσης της αποφυγής

SET #c = #PPOINT(30,-85,125,0,0,0) // ορισμός της 3ης θέσης της αποφυγής

SET #fp1 = #PPOINT(-140,-67,157,0,0,0) //ορισμός τελικής θέσης

MOVE #a // κίνηση στην 1η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 1η θέση της αποφυγής

IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN

    RETURN

END

MOVE #b // κίνηση στην 2η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 2η θέση της αποφυγής

IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN

    RETURN

END

MOVE #c // κίνηση στην 3η θέση της αποφυγής

BREAK // δεσμεύει την κίνηση στην 3η θέση της αποφυγής

IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN

    RETURN

END

MOVE #fp1 // κίνηση στην τελική θέση

IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN

    RETURN

END

Παρατηρείται ότι μετά από κάθε κίνηση του ρομποτικού βραχίονα έχει προστεθεί στον κώδικα ο έλεγχος των σημάτων 1002, 1003 και 1004(**IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN**). Εάν κάποιο από αυτά είναι αληθές τότε εκτελείται η εντολή **RETURN**, με την οποία διακόπτεται η

εκτέλεση του υποπρογράμματος και πραγματοποιείται η επιστροφή στην τελευταία εντολή του κυρίως προγράμματος. Επομένως, στην συνέχεια θα πραγματοποιηθεί η εκτέλεση της αντίστοιχης εντολής **REACTI** ανάλογα με το σήμα που είναι αληθές(δηλαδή ανάλογα σε ποια περιοχή βρίσκεται ο χρήστης την δεδομένη στιγμή). Με τον συγκεκριμένο προγραμματισμό επιτυγχάνεται η συνεχής παρακολούθηση της κίνησης του ανθρώπου και οι κατάλληλες με βάση αυτή αποφυγές του ρομποτικού βραχίονα Staubli RX90L.

7

Σταμάτημα ρομποτικού βραχίονα όσο ο άνθρωπος βρίσκεται στο χώρο εργασίας

7.1 Σκοπός και προσέγγιση

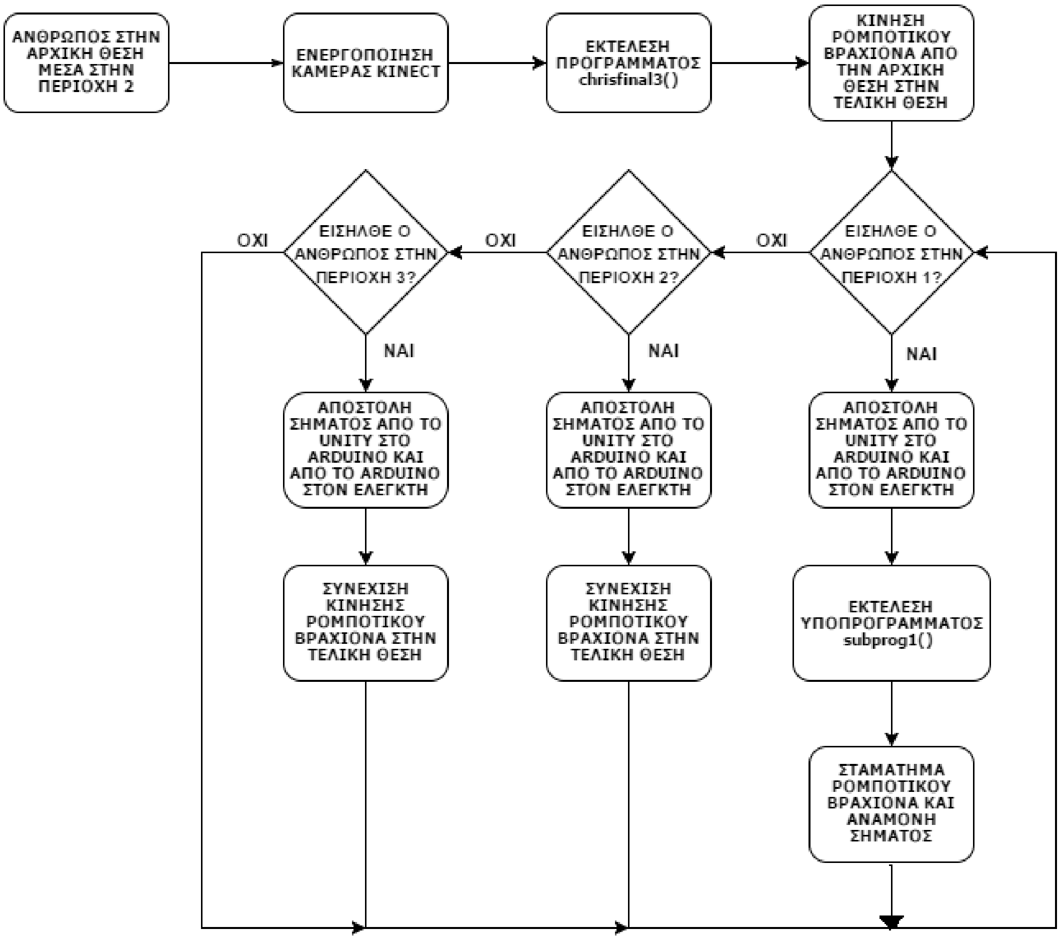
Ο σκοπός της τρίτης εφαρμογής είναι το σταμάτημα του ρομποτικού βραχίονα όσο ο άνθρωπος βρίσκεται μέσα στον χώρο εργασίας του ρομπότ, μέχρις ότου εκείνος εξέλθει από τον χώρο εργασίας. Τα προγράμματα του λογισμικού Unity3d, της πλακέτας ArduinoMega καθώς και οι κώδικες που αφορούν τον προγραμματισμό του ρομποτικού βραχίονα Staubli RX90L έχουν κάποιες διαφορές σε σχέση με την πρώτη και δεύτερη εφαρμογή. Θα αναλυθούν στις επόμενες υποενότητες.

7.2 Προγραμματισμός ρομποτικού βραχίονα

Για την ολοκλήρωση της συγκεκριμένης εφαρμογής έπαιξαν σημαντικό ρόλο οι εντολές **RETURN**, **REACTI** όπως και στην δεύτερη εφαρμογή. Έστω ότι ο άνθρωπος βρίσκεται έξω από τον χώρο εργασίας του ρομπότ και εκτελείται το πρόγραμμα **chrisfinal3()**. Ο ρομποτικός βραχίονας ξεκινά να εκτελέσει την κίνηση του από την θέση A στην θέση B (εύρος 280^o). Εάν ο άνθρωπος εισέλθει μέσα στον χώρο εργασίας του ρομπότ, τότε ο ρομποτικός βραχίονας σταματάει την κίνησή του και περιμένει το κατάλληλο σήμα για να συνεχίσει. Μέχρι να λάβει αυτό το σήμα ο ρομποτικός βραχίονας μένει ακινητοποιημένος. Εάν ο άνθρωπος εξέλθει από το χώρο εργασιας του ρομπότ

τότε το κατάλληλο σήμα φτάνει στον ελεγκτή του ρομποτικού βραχίονα και έτσι ο ρομποτικός βραχίονας συνεχίζει την κίνηση του μέχρι να φτάσει στην τελική θέση Β. Τα συγκεκριμένα βήματα επαναλαμβάνονται εάν ο άνθρωπος εισέλθει πάλι στον χώρο εργασίας του ρομπότ. Το πρόγραμμα σταματά όταν ο ρομποτικός βραχίονας φτάσει στην τελική θέση Β. (όλοι οι κώδικες βρίσκονται στο **Παράρτημα : Κώδικες – Σταμάτημα ρομποτικού βραχίονα**).

Παρακάτω παρουσιάζεται το διάγραμμα ροής για την 3η εφαρμογή, όπου φαίνονται όλες οι αλληλεπιδράσεις μεταξύ ανθρώπου και ρομποτικού βραχίονα.



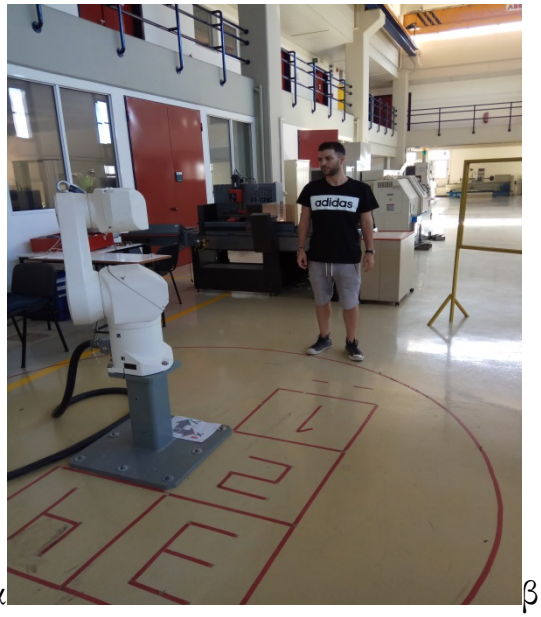
Εικόνα 7.1: Διάγραμμα ροής 3ης εφαρμογής

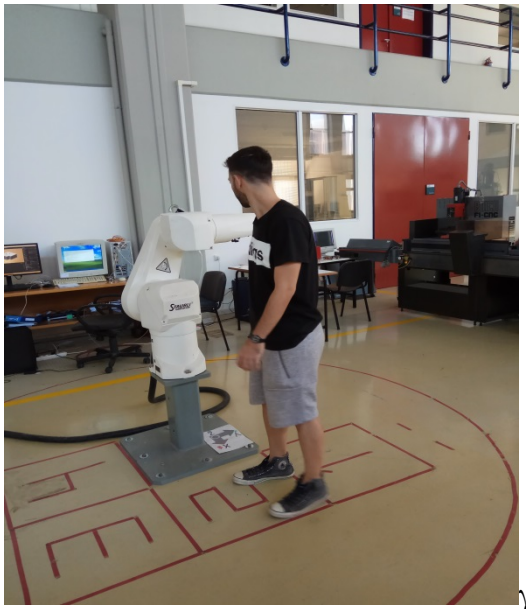
Παρακάτω ακολουθεί ο κώδικας του προγράμματος και του υποπρογράμματος για την τρίτη εφαρμογή:

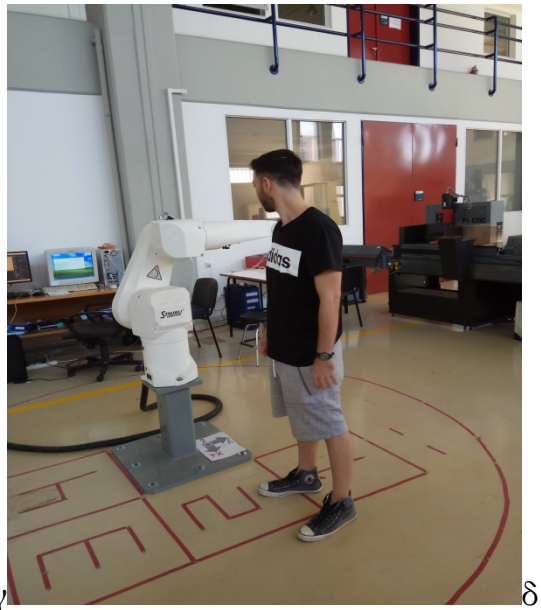
```
PROGRAM CHRISTASK3( )
        SET #sp = #PPOINT(140,-67,157,0,0,0)  //ορισμός της αρχικής θέσης
        SET #fp = #PPOINT(-140,-67,157,0,0,0)  //ορισμός τελικής θέσης
        SPEED 10 ALWAYS
        MOVE #sp          // κίνηση στην αρχική θέση
        BREAK             // δεσμεύει την κίνηση στην αρχική θέση
1       MOVE #fp          // κίνηση στην τελική θέση
        REACTI 1001, stop //έλεγχος σήματος 1001
        GOTO 1            //εκτέλεση της εντολής στη γραμμή 1

PROGRAM STOP()
        WAIT (SIG(1002) OR SIG(1003))  //αναμονή σήματος 1002,1003
RETURN
```

Ο άνθρωπος βρίσκεται έξω από τον χώρο εργασίας του ρομποτικού βραχίονα(**Εικόνα 7.2 (β)**), καθώς πηγαίνει ο ρομποτικός βραχίονας από την αρχική θέση στην τελική. Εάν ο άνθρωπος εισέλθει μέσα στον χώρο εργασίας τότε το σήμα 1001 γίνεται θετικό και καλείται το υποπρόγραμμα **stop()**, το οποίο περιμένει να γίνει θετικό το σήμα 1002 ή το 1003. Τα σήματα 1002, 1003 γίνεται θετικά όταν ο άνθρωπος εξέλθει από το χώρο εργασίας του ρομποτικού βραχίονα. Έτσι, το πρόγραμμα συνεχίζει στην επόμενη εντολή, η οποία είναι η **GOTO 1**, και εκτελείται η εντολή της γραμμής 1 δηλαδή η κίνηση στην τελική θέση. Η εκτέλεση του υποπρογράμματος μπορεί να πραγματοποιηθεί πολλές φορές μέσα στο πρόγραμμα, με σκοπό ο άνθρωπος να μπορεί να εισέρχεται με ασφάλεια στον χώρο εργασίας του ρομπότ.


α

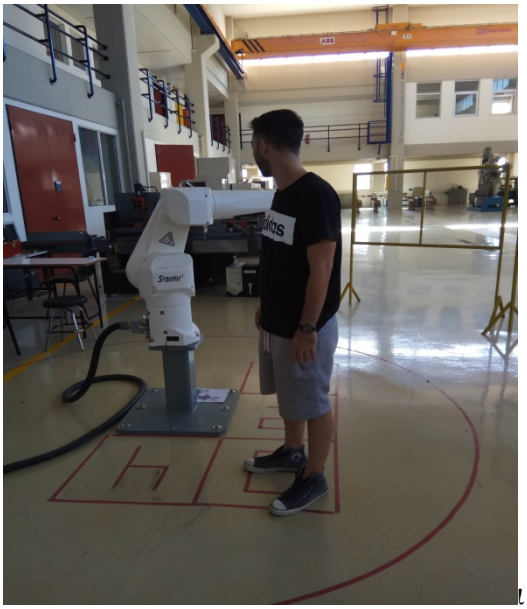

β


γ


δ


ε

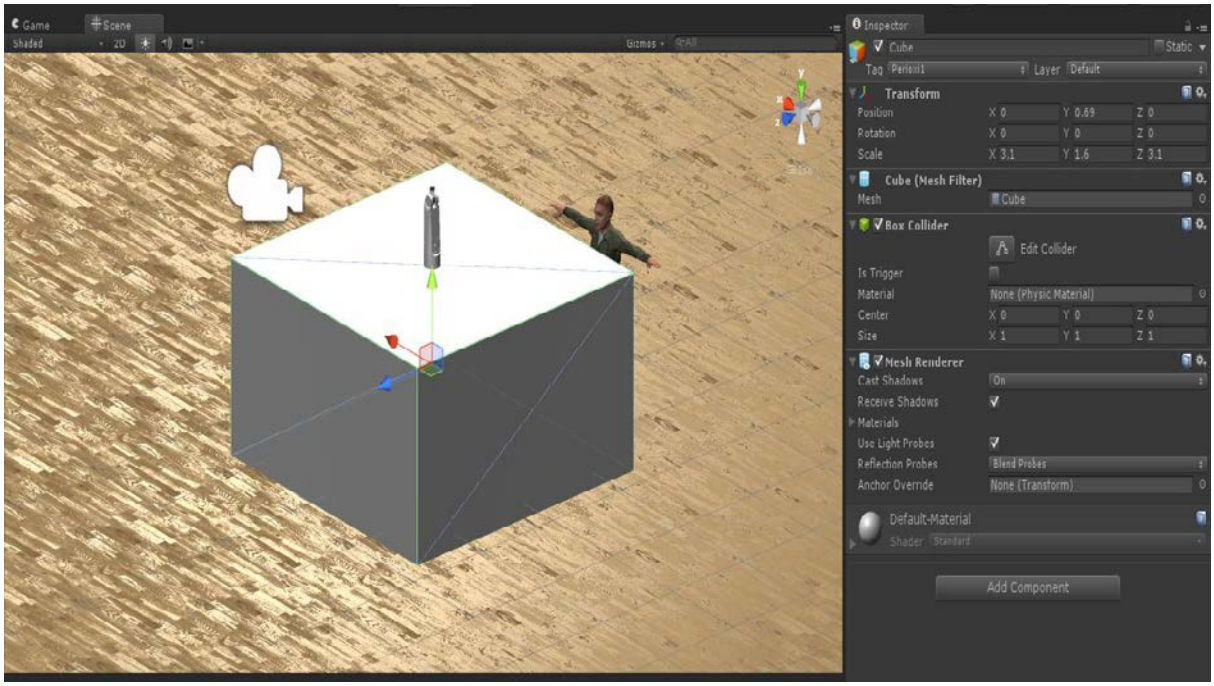

ζ



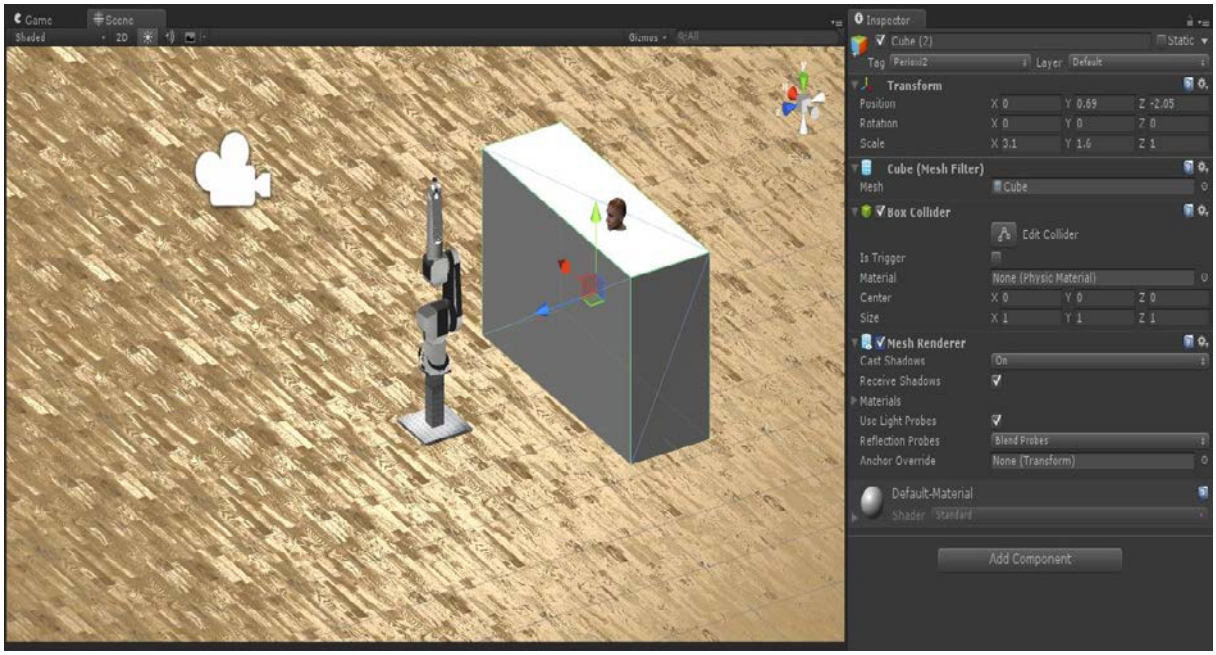
Εικόνες : 7.2 (α) Ο άνθρωπος βρίσκεται στην αρχική θέση για ενεργοποίηση του προγράμματος του Unity3d (β) Ο άνθρωπος βρίσκεται έξω από τον χώρο εργασίας του ρομποτικού βραχίονα (γ), (ζ) Ο άνθρωπος εισέρχεται στον χώρο εργασίας και ο ρομποτικός βραχίονας σταματά (δ), (ι) Ο άνθρωπος κινείται μέσα στον χώρο εργασίας και ο ρομποτικός βραχίονας παραμένει ακινητοποιημένος (ε) Ο άνθρωπος βρίσκεται έξω από τον χώρο εργασίας του ρομποτικού βραχίονα

7.3 Προγραμματισμός στο λογισμικό Unity3d

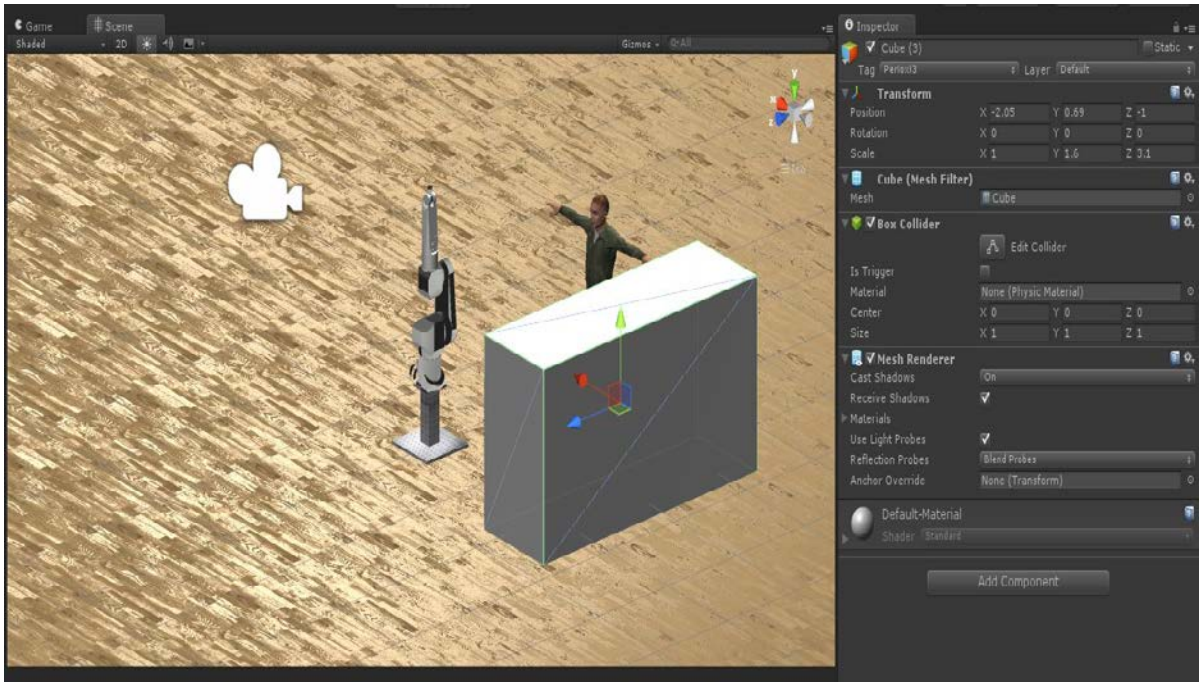
Η συγκεκριμένη εφαμογή έχει αρκετές ομοιότητες με τις προηγούμενες δύο εφαρμογές. Στις παρακάτω εικόνες φαίνονται οι Περιοχές 1, 2 και 3 που χρησιμοποιήθηκαν.



Εικόνα 7.3: Περιοχή 1 ως χώρος εργασίας



Εικόνα 7.3: Περιοχή 2 εκτός χώρου εργασίας



Εικόνα 7.3: Περιοχή 3 εκτός χώρου εργασίας

Ακολουθεί η ανάλυση των βασικών προγραμμάτων **Arduino( )** , **Perioxes_Collider( ).**

Program Perioxes_Collider( )

```
void Start ( )                    {
        sima1 = false;                    //ορισμός της μεταβλητής sima1 ως ψευδής
        sima2 = false;                    //ορισμός της μεταβλητής sima2 ως ψευδής
        sima3 = false;            }       //ορισμός της μεταβλητής sima3 ως ψευδής


void OnTriggerEnter(Collider other)       {
        if (other.gameObject.tag == "Perioxi1")           {
                sima1 = true;             //ορισμός της μεταβλητής sima1 ως αληθής
                Debug.Log ("Eimai ston xwro ergasias tou robot");       }
        if (other.gameObject.tag == "Perioxi2")           {
                sima2 = true;             //ορισμός της μεταβλητής sima2 ως αληθής
```

```
            Debug.Log ("Eimai eksw apo ton xwro ergasias tou robot");    }
        if (other.gameObject.tag == "Perioxi3")            {
            sima3 = true;          //ορισμός της μεταβλητής sima3 ως αληθής
            Debug.Log ("Eimai eksw apo ton xwro ergasias tou robot");    }
}
```

Program Arduino( )

```
using UnityEngine;
using SystemCollections;
using System.IO.Ports;                  //ορισμός της βιβλιοθήκης System.IO.Ports
public class Arduino : MonoBehaviour    {       //ορισμός κλάσης Arduino
private SerialPort sp;                  //ορισμός μεταβλητής sp ως σειριακή θύρα
void Start ( )          {
        sp = new SerialPort("COM4", 9600);      //ορισμός σειριακής θύρας sp
        sp.Open ( );   }                        //ενεργοποίηση της θύρας sp
void Update ( )         {
        char[ ] s = new char[ 1 ];              //ορισμός πίνακα s με ένα στοιχείο
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Peri
        oxes_Collider>( ).sima1)    {
        s[ 0 ] = "1" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 1
        sp.Write(s, 0 , 1);     }               //αποστολή του s στην σειριακή θύρα sp
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Peri
        oxes_Collider>( ).sima2)    {
        s[ 0 ] = "2" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 2
        sp.Write(s, 0 , 1);     }               //αποστολή του s στην σειριακή θύρα sp
        if(GameObject.FindGameObjectWithTag("liakous").GetComponent<Peri
        oxes_Collider>( ).sima3)    {
        s[ 0 ] = "3" ;                          //ορισμός του s[ 0 ] ως ο χαρακτήρας 3
```

```
        sp.Write(s, 0 , 1);      }              //αποστολή του s στην σειριακή θύρα sp
}
```

Ισχύουν ακριβώς τα ίδια όπως και στην 1η εφαρμογή, απλά στους συγκεκριμένους κώδικες έχουμε τις Περιοχές 1, 2 και 3. Η Περιοχή 1 αντιπροσωπεύει τον χώρο εργασίας του ρομπότ, ενώ οι Περιοχές 2 και 3 τους χώρους έξω από τον χώρο εργασίας του ρομπότ.

7.4 Προγραμματισμός ArduinoMega

Η σχηματική απεικόνιση του ηλεκτρικού κυκλώματος για την συγκεκριμένη εφαρμογή είναι ίδια με αυτή της 1ης εφαρμογής(**Εικόνα 5.13**), με την μόνη διαφορά ότι παραλείπεται εδώ το πινάκι 28, το Step-Up Module 4 καθώς και η είσοδος 1004 του ελεγκτή του ρομπότ.

Ακολουθεί ο κώδικας της πλακέτας ArduinoMega που προγραμματίστηκε σε C στο λογισμικό Arduino IDE.

Program Task3( )

```
#inlude <SoftwareSerial.h>
void setup( )          {          //συνάρτηση setup( )
        int pin1 = 22;             //ορισμός για το πινάκι 22 της πλακέτας ως pin1
        int pin2 = 24;             //ορισμός για το πινάκι 24 της πλακέτας ως pin2
        int pin3 = 26;             //ορισμός για το πινάκι 26 της πλακέτας ως pin3
        Serial.begin (9600);              //ορισμός του baudrate ως 9600
        pinMode (pin1, OUTPUT);           //ορισμός της μεταβλητής pin1 ως έξοδο
        pinMode (pin2, OUTPUT);           //ορισμός της μεταβλητής pin2 ως έξοδο
        pinMode (pin3, OUTPUT);           //ορισμός της μεταβλητής pin3 ως έξοδο
}
void loop( )           {                  //συνάρτηση loop( )
if (Serial.available( ) > 0)     {        //έλεγχος για τα σειριακά δεδομένα
        char incoming_char = (char)Serial.read ( );
```

```
        if (incoming_char == '1')    {
        digitalWrite(pin1, HIGH);          //μεταφορά στο pin1 τάσης 5 Volts
        digitalWrite(pin2, LOW);           //μεταφορά στο pin2 τάσης 0 Volts
        digitalWrite(pin3, LOW);    }      //μεταφορά στο pin3 τάσης 0 Volts
        if (incoming_char == '2')    {
        digitalWrite(pin2, HIGH);          //μεταφορά στο pin2 τάσης 5 Volts
        digitalWrite(pin1, LOW);           //μεταφορά στο pin1 τάσης 0 Volts
        digitalWrite(pin3, LOW);    }      //μεταφορά στο pin3 τάσης 0 Volts
        if (incoming_char == '3')    {
        digitalWrite(pin3, HIGH);          //μεταφορά στο pin3 τάσης 5 Volts
        digitalWrite(pin1, LOW);           //μεταφορά στο pin1 τάσης 0 Volts
        digitalWrite(pin2, LOW);    }      //μεταφορά στο pin2 τάσης 0 Volts
        }                          //τέλος της if(Serial.available ( ) > 0)
}                                  //τέλος της συνάρτησης void loop
```

8

Παραλαβή συντεταγμένων ανθρώπου και ρομποτικού βραχίονα

8.1 Σκοπός και προσέγγιση

Στο κεφάλαιο αυτό, σκοπός είναι η παραλαβή των συντεταγμένων του ανθρώπου από το λογισμικό Unity3d στην πλακέτα του Arduino, καθώς και των συντεταγμένων του ρομποτικού βραχίονα από τον ελεγκτή του ρομποτικού βραχίονα στην πλακέτα του Arduino. Με αυτόν τον τρόπο μπορεί κάποιος να κάνει έλεγχο του ρομποτικού βραχίονα με πολύ μεγάλη ακρίβεια. Οι κώδικες αναλύονται στις επόμενες υποενότητες.

8.2 Συντεταγμένες ρομποτικού βραχίονα

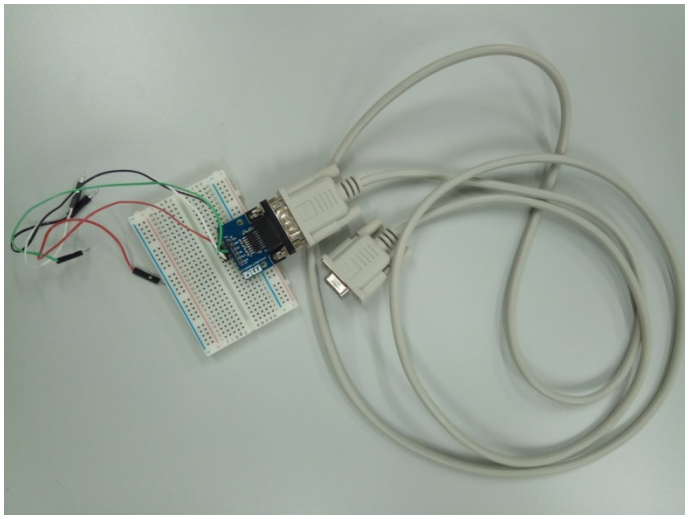
8.2.1 Προγραμματισμός ρομποτικού βραχίονα

Στη συγκεκριμένη υποενότητα αναλύεται ο κώδικας για την αποστολή των συντεταγμένων του ρομποτικού βραχίονα στην πλακέτα του Arduino. Για τη συγκεκριμένη εφαρμογή σημαντικό ρόλο έπαιξαν οι εντολές **ATTACH, DETACH** και **DECOMPOSE x[ ]**. Ο ρομποτικός βραχίονας ξεκινά να κινείται από την αρχική θέση στην τελική. Με την εντολή **ATTACH** δεσμεύεται η σειριακή θύρα του ελεγκτή του ρομποτικού βραχίονα. Η εντολή **DECOMPOSE x=[ ]= b1** αποθηκεύει τις συντεταγμένες της μεταβλητής **b1** στον πίνακα **x[ ]**. Η εντολή **HERE** αποθηκεύει σε μια μεταβλητή τις συντεταγμένες του ρομποτικού βραχίονα τη στιγμή που εκτελείται η εντολή. Ο αριθμός της πρώτης **FOR** (10000) έχει χρησιμοποιηθεί αυθαίρετα, έτσι ώστε μέχρι ο ρομποτικός βραχίονας να τελειώσει την κίνηση του να στέλνει συνέχεια

τις συντεταγμένες στη σειριακή θύρα. Δεν έχει χρησιμοποιηθεί εδώ η εντολή **BREAK** , καθώς δεν είναι αναγκαία η δέσμευση κάποιας κίνησης. Τέλος, με την εντολή **WRITE (slun) x[i]** πραγματοποιείται η αποστολή των στοιχείων του πίνακα **x[ ]** . Η δεύτερη **FOR** καλύπτει τις τιμές από 0 έως 2 επειδή ο χρήστης του Arduino θέλει να παραλάβει τις συντεταγμένες (x, y, z). Εάν ο χρήστης επιθυμούσε να παραλάβει μόνο τις συντεταγμένες (x,y) τότε η **FOR** θα κάλυπτε τις τιμές από 0 έως 1. Παρακάτω ακολουθεί ο κώδικας :

```
Program serialdata ( )
AUTO slun          //ορισμός της μεταβλητής slun
AUTO $text         //ορισμός μεταβλητής τύπου χαρακτήρα
SET #sp = #PPOINT(140,-67,157,0,0,0)  //ορισμός της αρχικής θέσης
SET #fp = #PPOINT(-140,-67,157,0,0,0)      //ορισμός τελικής θέσης
SPEED 10 ALWAYS
MOVE #sp           //κίνηση στην αρχική θέση
MOVE #fp           //κίνηση στην τελική θέση
ATTACH  (slun, 4)  "SERIAL: 2"     //χρησιμοποίηση της σειριακής θύρας 2
FOR  y = 0 TO 10000
        HERE b1            //ορισμός συντεταγμένων (x, y, z) ως b1
        DECOMPOSE x[ ] = b1 // εισαγωγή συντεταγμένων στον πίνακα x[ ]
        FOR i = 0 TO 2
                WRITE (slun) x[i] //αποστολή στη σειριακή θύρα x[0], x[1], x[2]
        END
END
DETACH  (slun) //τερματισμός της χρησιμοποίησης της σειριακής θύρας 2
```

Για τη μεταφορά των δεδομένων από τον ελεγκτή του ρομποτικού βραχίονα στην πλακέτα Arduino χρησιμοποιήθηκε το καλώδιο σειριακής επικοινωνίας που φαίνεται στην **Εικόνα 8.1**.



Εικόνα 8.1 : RS-232 Serial Cable(Male to Female)

Η μία άκρη του καλωδίου συνδέεται με τη σειριακή θύρα 2(RS-232) του ελεγκτή (**Εικόνα 8.2**), ενώ η άλλη άκρη του καλωδίου συνδέεται με μια δοκιμαστική πλακέτα(breadboard) (**Εικόνα 8.1**).

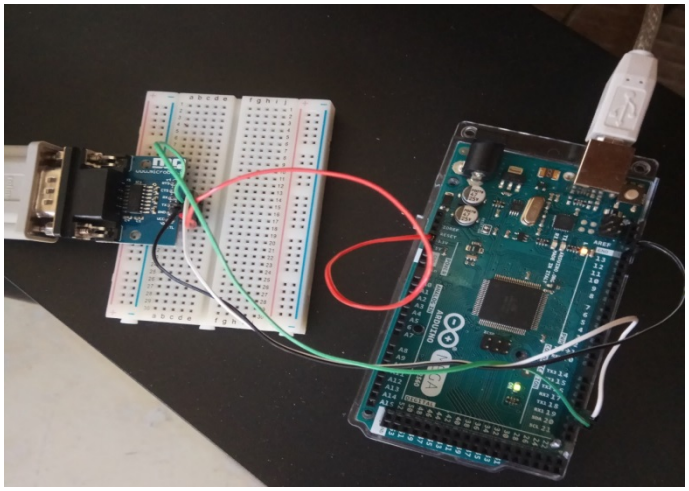
Εικόνα 8.2 : (α) Το μπροστινό μέρος του ελεγκτή του ρομποτικού βραχίονα (β) Οι σειριακές θύρες RS - 232

8.2.2 Προγραμματισμός πλακέτας Arduino

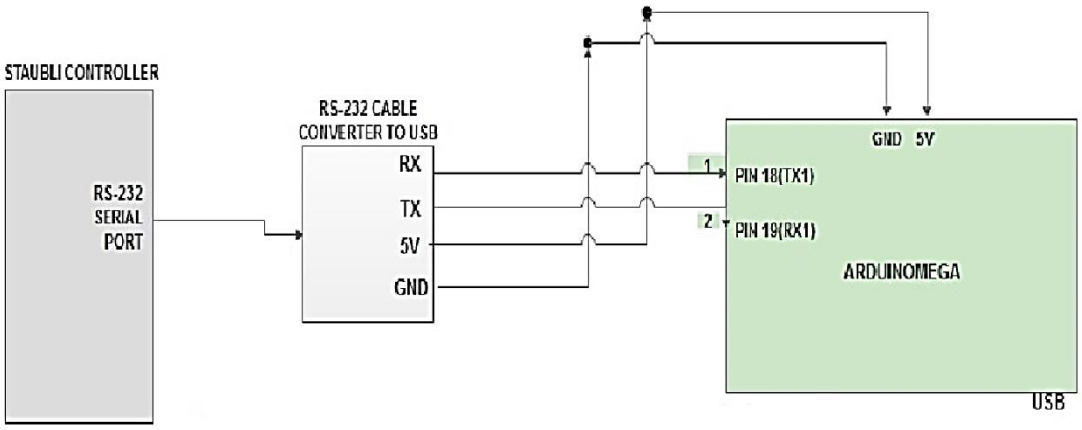
Στο κεφάλαιο αυτό θα αναλυθεί το ηλεκτρικό κύκλωμα που χρησιμοποιήθηκε για την επικοινωνία της πλακέτας ArduinoMega με τον ελεγκτή του ρομποτικού βραχίονα Staubli RX90L, καθώς και ο κώδικας που προγραμματίστηκε στο λογισμικό Arduino IDE.

Για την πραγματοποίηση της συγκεκριμένης εφαρμογής χρησιμοποιήθηκε το καλώδιο (**Εικόνα 8.1**), με το οποίο ήταν δυνατή η μεταφορά δεδομένων από τον ελεγκτή του ρομποτικού βραχίονα στην πλακέτα Arduino μέσω της σειριακής θύρας 1. Το ArduinoMega που χρησιμοποιήθηκε έχει 4 σειριακές θύρες(πινάκια 0 – 1, 18 – 19, 16 – 17, 14 – 15). Για τα πινάκια 18 – 19 χρησιμοποιείται η Serial1.begin( ) , Serial1.available( ) , η Serial1.read( ) κτλπ. Για τα πινάκια 16 – 17 χρησιμοποιείται αντίστοιχα ο αριθμός 2 , για τα πινάκια 14 – 15 ο αριθμός 3, και για τα πινάκια 0 – 1 κανένας αριθμός.

Το κύκλωμα που χρησιμοποιήθηκε για την επικοινωνία της πλακέτας ArduinoMega με τον ελεγκτή του ρομποτικού βραχίονα φαίνεται στην **Εικόνα 8.3** , ενώ στην **Εικόνα 8.4** φαίνεται η σχηματική απεικόνιση της σχετικής συνδεσμολογίας.



Εικόνα 8.3 : Συνδεσμολογία Arduino με ελεγκτή μέσω RS-232 Cable



Εικόνα 8.4 : Σχηματική απεικόνιση συνδεσμολογίας με RS232

Ακολουθεί η ανάλυση του κώδικα της πλακέτας ArduinoMega που προγραμματίστηκε σε C στο λογισμικό Arduino IDE.

Robot_position( )

```
#inlude <SoftwareSerial.h>

String inString = '''';

float pos[3];

int i;


void setup( )          {              //συνάρτηση setup( )
        Serial.begin (9600);                    //ορισμός του baudrate ως 9600
        Serial1.begin(9600);  //ορισμός του baudrate ως 9600 στη σειριακή θύρα 1
void loop( )           {                        //συνάρτηση loop( )
        while (Serial1.available( ) > 0)        { //έλεγχος για τα σειριακά δεδομένα
                char  inChar = (char)Serial1.read ( ); //ορισμός μεταβλητής inChar
                if (inChar == 'x')      { // έλεγχος της μεταβλητής inChar
                        i = 0;          } //ορισμός της μεταβλητής i ως 0
```

```
            else if (inChar == '\n')      { // έλεγχος της μεταβλητής inChar
                inString +=  inChar    }//πρόσθεση της inChar στη συμβολοσειρά
            else    {
                float num = inString.toFloat(); //μετατροπή της inString σε float
                pos[i] = num;          //εισαγωγή της num στη θέση i του πίνακα
                inString = '''';       // μηδενισμός της συμβολοσειράς inString
                i = i + 1;       }//αύξηση της μεταβλητής i κατά 1
            if (i == 3)      {
                Serial.println("robot X coordinate is :");
                Serial.println(pos[1], 4);
                Serial.println("robot Y coordinate is :");
                Serial.println(pos[2], 4);
                Serial.println("robot  Z coordinate is :");
                Serial.println(pos[3],4);
            }
        }                       //τέλος της if(Serial.available ( ) > 0)
}                               //τέλος της συνάρτησης void loop
```

Στον παραπάνω κώδικα χρησιμοποιείται η συμβολοσειρά **inString** , με σκοπό να αποθηκευτούν οι χαρακτήρες σε αυτή. Επίσης χρησιμοποιείται ο πίνακας **pos[3]** με 3 θέσεις αποθήκευσης, καθώς και ο μετρητής **i**. Τέλος, χρησιμοποιείται η μεταβλητή τυπου char **inChar**, στην οποία αποθηκεύεται κάθε φορά ο χαρακτήρας που διαβάζει το Arduino με την εντολή **char inChar = (char)Serial1.read( );**. Η εντολή **inString += inChar;** προσθέτει στην συμβολοσειρά τον χαρακτήρα **inChar**.

Ο ελεγκτής του ρομποτικού βραχίονα στέλνει αδιάκοπα τα ακόλουθα δεδομένα που αλλάζουν με το χρόνο π.χ. : x \n 700,25 \n 650,75 \n 1500,00 \n . Οπότε, η πλακέτα Arduino λαμβάνει πρώτα τον χαρακτήρα **x** και τον αποθηκεύει στην πρώτη θέση του πίνακα **x[0]**, αφού πρώτα λάβει τον χαρακτήρα **\n**(χαρακτήρας Newline). Στη συνέχεια, μηδενίζει την

συμβολοσειρά **inString** και αυξάνει τον μετρητή **i** κατά 1. Αντιστοίχως, λαμβάνει τις συντεταγμένες x, y, και z και τις αποθηκεύει στις θέσεις 1, 2 ,και 3 αντίστοιχα. Τέλος, εφόσον ο μετρητής **i** έχει την τιμή 3 εκτελείται η τελευταία if, με την οποία τυπώνονται στην σειριακή οθόνη(Serial Monitor) οι συντεταγμένες του ρομποτικού βραχίονα. Αυτό συνεχίζεται μέχρις ότου ο ρομποτικός βραχίονας να φτάσει στην τελική του θέση. Έτσι, έρχονται συνέχεια σετ συντεταγμένων (x, y, z) .

8.3 Συντεταγμένες ανθρώπου

8.3.1 Προγραμματισμός λογισμικού Unity3d

Στο κεφάλαιο αυτό θα αναλυθεί ο κώδικας που χρησιμοποιήθηκε στο λογισμικό Unity3d για την αποστολή των συντεταγμένων του ανθρώπου από την κάμερα MicrosoftXboxOne(μέσω υπολογιστή) στην πλακέτα Arduino.

Για την πραγματοποίηση της συγκεκριμένης εφαρμογής χρησιμοποιήθηκε ειδικό καλώδιο (USB to TTL serial cable, **Εικόνα 8.5**), με το οποίο ήταν δυνατή η μεταφορά δεδομένων από τον υπολογιστή στην πλακέτα Arduino μέσω της σειριακής θύρας 1. Παρακάτω αναλύεται ο κώδικας που χρησιμοποιήθηκε για τη συγκεκριμένη εφαρμογή .

Program Avatar_Position( )

```
using UnityEngine;
using System.Collections;
using System.IO;
using System.IO.Ports;
public class Avatar_Position : Monobehaviour    {
        private SerialPort sp;
        public GameObject avatar_center;
        public GameObject robot_center;
```

```
public Vector3 avatar_Pos;
public Vector3 robot_Pos;
public Vector3 relative_Pos;
void Start ( )        {
        sp = new SerialPort ("COM6" , 9600);
        sp.ReadTimeout = 50;
        sp.Open ( );        }
void Update ( )    {
        Coordinates ( );    }
void Coordinates ( )      {
        int[ ] xcoordinates;
        int[ ] ycoordinates;
        int[ ] zcoordinates;
        int lengthOfArray = 100;
        xcoordinates = new int[lengthOfArray];
        ycoordinates = new int[lengthOfArray];
        zcoordinates = new int[lengthOfArray];
        for (int i = 0; i < lengthOfArray; i++)        {
                avatar_Pos = avatar_center.transform.position;
                robot_Pos = robot_center.transform.position;
                relative_Pos = -1000.0f * (avatar_Pos – robot_Pos);
                int positionx = (int)relative_Pos.x;
                int positiony = (int)relative_Pos.z;
                int positionz = (int)relative_Pos.y;
                xcoordinates[i] = positionx;
```

```
                ycoordinates[i] = positiony;
                zcoordinates[i] = positionz;
                if (i == 99) {
                        positionx = xcoordinates[i];
                        positiony = ycoordinates[i];
                        positionz = zcoordinates[i];
                        string _x = positionx.ToString( );
                        string _y = positiony.ToString( );
                        string _z = positionz.ToString( );
                        Debug.Log ( _x + ',' + _y + ',' + _z);
                        sp.Write ("x");
                        sp.Write (_x);
                        sp.Write ("y");
                        sp.Write (_y);
                        sp.Write ("z");
                        sp.Write (_z);
                        sp.Write ("e");
                } //τέλος της if(i == 99)
        }//τέλος της for(int i = 0; i < lengthOfArray; i++)
} //τέλος της Void Coordinates ( )
}//τέλος της Avatar_Position : Monobehaviour
```

Ο συγκεκριμένος κώδικας υπολογίζει τις συντεταγμένες του ανθρώπου μέσω της κάμερας Microsoft Kinect Xbox One. Επειδή όμως το σύστημα συντεταγμένων στο λογισμικό Unity3d είναι διαφορετικό από το σύστημα συντεταγμένων του πραγματικού χώρου στο εργαστήριο, ήταν αναγκαία η

μετατροπή αυτή. Έτσι, ορίζονται 2 μεταβλητές **avatar_Pos** και **robot_Pos**. Η πρώτη ορίζεται ως διάνυσμα θέσης(Vector3) μέσω της εντολής **avatar_Pos = avatar_center.transform.position;** και είναι οι συντεταγμένες του κέντρου βάρους του avatar Liakous, ενώ η δεύτερη ορίζεται αντίστοιχα μέσω της εντολής **robot_Pos = robot_center.transform.position;** και είναι οι συντεταγμένες του κέντρου βάρους του ρομποτικού βραχίονα Staubli RX90L. Στη συνέχεια, ορίζεται η σχετική θέση ρομποτικού βραχίονα-ανθρώπου με την εντολή **relative_Pos = -1000.0f * (avatar_Pos – robot_Pos);** . Εδώ σημαντικό ρόλο παίζει το μείον με το οποίο μετατρέπεται το ένα σύστημα συντεταγμένων στο άλλο, και γίνεται ο πολλαπλασιασμός με 1000 γιατί το Unity3d υπολογίζει όλα τα μεγέθη σε μέτρα, οπότε λαμβάνονται εν τέλει οι συντεταγμένες σε χιλιοστά. Με την εντολή **int positionx = (int)relative_Pos.x;** ορίζεται η ακέραια μεταβλητή που αφορά την συντεταγμένη x. Έπειτα μεταφέρονται οι συντεταγμένες στους αντίστοιχους πίνακες με την εντολή **positionx = xcoordinates[i];** . Τέλος, εάν ο μετρητής **i** γίνει 99 τότε οι συντεταγμένες αποθηκεύονται σε μεταβλητές τύπου συμβολοσειράς (string) με την εντολή **string _x = positionx.ToString( );** έτσι ώστε να είναι δυνατή η αποστολή τους και στέλνονται οι χαρακτήρες x, y, z και e ανάμεσα στους οποίους στέλνεται κάθε φορά και η αντίστοιχη μεταβλητή για τις συντεταγμένες **_x**, **_y**, και **_z**. Ο λόγος που δεν στέλνονται κατευθείαν οι συντεταγμένες αλλά ανά 100 είναι η ταχύτητα με την οποία το Arduino αποθηκεύει δεδομένα στο buffer. Στην αρχή, ο κώδικας είχε υλοποιηθεί χωρίς τις 100 επαναλήψεις και παρατηρήθηκε ότι γινόταν υπερχείλιση του buffer της πλακέτας Arduino(buffer overflow), με συνέπεια να μην λαμβάνονται οι επιθυμητές τιμές των συντεταγμένων.

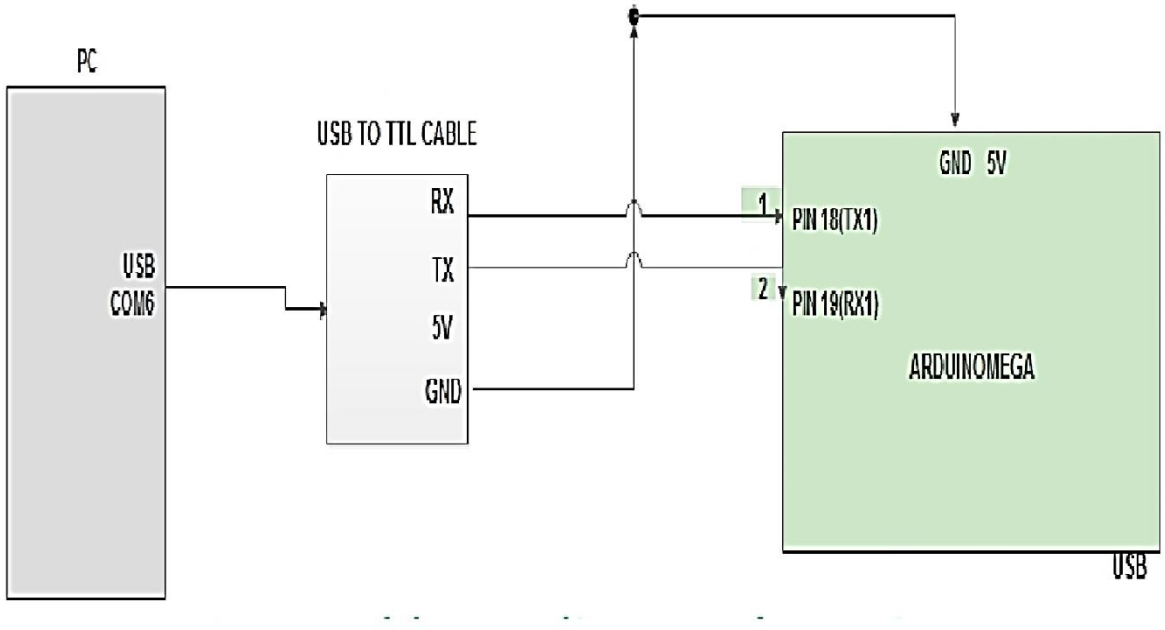
8.3.2 Προγραμματισμός πλακέτας Arduino

Στο κεφάλαιο αυτό θα αναλυθεί το ηλεκτρικό κύκλωμα που χρησιμοποιήθηκε για την επικοινωνία της πλακέτας ArduinoMega με τον ελεγκτή του ρομποτικού βραχίονα Staubli RX90L, καθώς και ο κώδικας που προγραμματίστηκε στο λογισμικό Arduino IDE.

Για την πραγματοποίηση της συγκεκριμένης εφαρμογής χρησιμοποιήθηκε το καλώδιο (USB to TTL cable) το οποίο φαίνεται στην **Εικόνα 8.5**, με το οποίο ήταν δυνατή η μεταφορά δεδομένων από τον υπολογιστή στην πλακέτα Arduino. . Τα πινάκια του καλωδίου (USB to TTL

Cable) είναι τα ακόλουθα : το RX είναι το πράσινο πινάκι, το TX το άσπρο πινάκι, η τάση 5V είναι το κόκκινο πινάκι και το GND είναι το μάυρο πινάκι. Στην **Εικόνα 8.6** φαίνεται η συνδεσμολογία της εφαρμογής.

Εικόνα 8.5 : USB to TTL Cable



Εικονα 8.6 : Σχηματική απεικόνιση συνδεσμολογίας με usb to ttl

Ακολουθεί η ανάλυση του κώδικα της πλακέτας ArduinoMega που προγραμματίστηκε σε C στο λογισμικό Arduino IDE.

Human_position( )

```
#inlude <SoftwareSerial.h>
String inString = '''';          //ορισμός της συμβολοσειράς inString
boolean x = false;      //ορισμός της μεταβλητής x ως ψευδής
boolean y = false;      //ορισμός της μεταβλητής y ως ψευδής
boolean z = false;      //ορισμός της μεταβλητής z ως ψευδής
int xcoord;             //ορισμός της συντεταγμένης x
int ycoord;             //ορισμός της συντεταγμένης y
int zcoord;             //ορισμός της συντεταγμένης z
void setup( )           {          //συνάρτηση setup( )
        Serial.begin (9600);    //ορισμός του baudrate ως 9600
        Serial1.begin(9600);   //ορισμός του baudrate ως 9600 στη σειριακή θύρα 1
void loop( )            {                          //συνάρτηση loop( )
        while (Serial1.available( ) > 0)        { //έλεγχος για τα σειριακά δεδομένα
                char  inChar = (char)Serial1.read ( ); //ορισμός μεταβλητής inChar
                if (inChar == 'x')      { // έλεγχος της μεταβλητής inChar
                        x = true;       } //ορισμός της μεταβλητής x ως αληθής
                else if (inChar == 'y')         { // έλεγχος της μεταβλητής inChar
                        y = true;       //ορισμός της μεταβλητής y ως αληθής
                        x = false;      //ορισμός της μεταβλητής x ως ψευδής
                        xcoord = inString.toInt ( );
                        inString = '''';    }//μηδενισμός της inString
                else if  (inChar == 'z') {      // έλεγχος της μεταβλητής inChar
                        z = true;       //ορισμός της μεταβλητής z ως αληθής
                        y = false;      //ορισμός της μεταβλητής y ως ψευδής
                        ycoord = inString.toInt ( );
                        inString = '''';     }//μηδενισμός της inString
```

```
            else if ((inChar != 'x') && (inChar != 'y') && (inChar != 'z') &&
(inChar != 'e'))        {       // έλεγχος της μεταβλητής inChar
                        if (x == true)  { // έλεγχος της μεταβλητής x
                                inString += inChar;  }
                        if (y == true)  { // έλεγχος της μεταβλητής y
                                inString += inChar;  }
                        if (z == true)  { // έλεγχος της μεταβλητής z
                                inString += inChar;  }
                }
                else if  (inChar == 'e') { // έλεγχος της μεταβλητής inChar
                        z = false;      //ορισμός της μεταβλητής z ως ψευδής
                        zcoord = inString.toInt ( );
                        inString = '''';    //μηδενισμός της inString
                        Serial.println("human X coordinate is :");
                        Serial.println(xcoord);
                        Serial.println("human Y coordinate is :");
                        Serial.println(ycoord);
                        Serial.println("human Z coordinate is :");
                        Serial.println(zcoord);
                }
        }                       //τέλος της if(Serial.available ( ) > 0)
}                               //τέλος της συνάρτησης void loop
```

Ο παραπάνω κώδικας λαμβάνει τα δεδομένα που του στέλνει ο υπολογιστής μέσω του λογισμικού Unity3d και τα τυπώνει στην Serial Monitor του Arduino. Αρχικά, ορίζεται η μεταβλητή τύπου συμβολοσειράς με την εντολή **String inString = '''';** , οι 3 μεταβλητές για τις συντεταγμένες **x**, **y** και **z**, και οι 3 ακέραιες μεταβλητές **xcoord**, **ycoord** και **zcoord** . Στη συνέχεια, ελέγχει εάν υπάρχουν δεδομένα στον buffer του Arduino. Εάν υπάρχουν τότε

τοποθετεί τον πρώτο χαρακτήρα σε μια μεταβλητή τύπου χαρακτήρα με την εντολή **char inChar = (char)Serial1.read ( );** και ελέγχει κάθε φορά το συγκεκριμένο χαρακτήρα. Δεδομένου ότι το Unity3d στέλνει δεδομένα όπως π.χ. : x640y1150z1200e , με την εντολή **else if ((inChar != 'x') && (inChar != 'y') && (inChar != 'z') && (inChar != 'e'))** ελέγχεται ότι ο εισερχόμενος χαρακτήρας δεν είναι ο x, y, z και e , και έπειτα ελέγχεται ποια μεταβλητή είναι αληθής από τις **boolean x, y** και **z** . Όσο είναι αληθής κάποια από τις **boolean x, y** και **z** προστίθεται κάθε φορά στην συμβολοσειρά **inString** ο εισερχόμενος χαρακτήρας **inChar** με την εντολή **inString += inChar;** . Προηγουμένως, εάν ο εισερχόμενος χαρακτήρας είναι ο x, τότε η **boolean x** γίνεται αληθής, εάν ο εισερχόμενος χαρακτήρας είναι ο y, τότε η **boolean y** γίνεται αληθής, ενώ η **boolean x** ψευδής, αποθηκέυεται στην **xcoord** η μετατροπή της συμβολοσειράς σε ακέραιο(εντολή **xcoord = inString.toInt ( );**), εφόσον έχουμε λάβει τις συντεταγμένες x, και μηδενίζεται η συμβολοσειρά **inString**. Αντίστοιχα, εάν ο εισερχόμενος χαρακτήρας είναι ο z, τότε η **boolean z** γίνεται αληθής, ενώ η **boolean y** ψευδής, αποθηκέυεται στην **ycoord** η μετατροπή της συμβολοσειράς σε ακέραιο(εντολή **ycoord = inString.toInt ( );**), εφόσον έχουμε λάβει τις συντεταγμένες y, και μηδενίζεται η συμβολοσειρά **inString**. Εάν ο εισερχόμενος χαρακτήρας είναι ο e, τότε οι **boolean x, y, z** γίνονται ψευδείς, αποθηκέυεται στην **zcoord** η μετατροπή της συμβολοσειράς σε ακέραιο(εντολή **zcoord = inString.toInt ( );**), εφόσον έχουμε λάβει τις συντεταγμένες z , μηδενίζεται η συμβολοσειρά **inString** και τυπώνονται στην Serial Monitor του Arduino οι 3 συντεταγμένες **xcoord, ycoord** και **zcoord**.

9

Συμπεράσματα

9.1 Επισκόπηση

Η παρούσα διπλωματική εργασία πραγματεύεται τη συνεργασία Ανθρώπου-Ρομποτικού βραχίονα στα ρομποτικά συστήματα παραγωγής, και την αποτελεσματικότητα, την ασφάλεια και την αποδοχή αυτής της συνεργασίας από τον άνθρωπο. Επιπλέον, προτείνεται η χρήση της Εικονικής Πραγματικότητας ως εργαλείου υλοποίησης της παραπάνω συνεργασίας.

Η τεχνική υποχώρησης που χρησιμοποιήθηκε στοχεύει στη δημιουργία προσαρμοστικής κινηματικής συμπεριφοράς από τον ρομποτικό βραχίονα Staubli RX90L, και εστιάζει στην αποφυγή της σύγκρουσης μεταξύ ανθρώπου-ρομπότ, χωρίς να πραγματοποιείται διακοπή της κίνησης του ρομποτικού βραχίονα.

Η λογική της τεχνικής υποχώρησης βασίζεται στην υπόθεση ότι από μόνη της η υποχώρηση του ρομποτικού βραχίονα αρκεί για να αποφευχθούν συγκρούσεις ή επικίνδυνες επαφές μεταξύ ανθρώπου-ρομπότ. Όσον αφορά την ασφάλεια, τα πειράματα που διεξήχθησαν έδειξαν ότι η χρησιμοποιούμενη τεχνική ασφαλούς συνεργασίας είναι αποτελεσματική.

Δημιουργήθηκαν όλοι οι κατάλληλοι κώδικες προγραμματισμού για τα λογισμικά Unity3d, Arduino IDE καθώς και για τον έλεγχο του ρομποτικού βραχίονα Staubli RX90L. Συνολικά αναπτύχθηκαν τρεις περιπτώσεις εφαρμογής (use cases). Η πρώτη περίπτωση εφαρμογής (**Ενότητα 5**) αναπαριστά την τεχνική αποφυγής του ανθρώπου όταν αυτός εισέρχεται σε κάθεμια από τις περιοχές 1, 2, 3 και 4 ξεχωριστά. Η δεύτερη περίπτωση εφαρμογής (**Ενότητα 6**) αναπαριστά την σύνθετη τεχνική αποφυγής όσο ο άνθρωπος εισέρχεται στις περιοχές 1, 2, 3 και 4 με συνδιασμό της εισόδου στις διάφορες περιοχές, π.χ. 1-2, 2-3, 1-3-4 κ.λ.π.. Η τρίτη περίπτωση εφαρμογής

(**Ενότητα 7**) αναπαριστά την ακινητοποίηση του ρομποτικού βραχίονα όσο ο άνθρωπος βρίσκεται μέσα στον χώρο εργασίας του ρομποτικού βραχίονα, με σκοπό ο άνθρωπος να μπορεί να κάνει εργασίες με την κατάλληλη ασφάλεια. Τέλος, αναπτύχθηκε και λειτουργικότητα παραλαβής συντεταγμένων του ανθρώπου και του ρομποτικού βραχίονα (**Ενότητα 8**), η οποία είναι βοηθητική για μελλοντική χρήση.

Όπως αναφέρθηκε και στην ενότητα 5.2, στην παρούσα διπλωματική εργασία η τροχιά του ρομποτικού βραχίονα, καθώς και οι συντεταγμένες των σημείων για τις 3 αποφυγές για τις περιοχές 1, 2, 3 και 4 χρησιμοποιήθηκαν με 'τυχαίο' τρόπο. Επομένως, η μεθοδολογία που ακολουθήθηκε γενικεύεται εάν κάποιος επιθυμεί να τροποποιήσει τα σημεία που χρησιμοποιήθηκαν στη συγκεκριμένη διπλωματική εργασία, ακόμη και να προσθέσει νέα σημεία εάν επρόκειτο να υπάρχουν εμπόδια σε κάποιον άλλον χώρο εργασίας.

9.2 Μελλοντικές επεκτάσεις

Οι τρεις εφαρμογές που αναπτύχθηκαν στην παρούσα διπλωματική εργασία είναι ολοκληρωμένες και λειτουργικές, και προσφέρονται ως μια πλατφόρμα δοκιμών της συνεργασίας ανθρώπου-ρομποτικού βραχίονα. Μπορεί, λοιπόν, να γίνει πληθώρα νέων δοκιμών με νέα σενάρια εργασίας ανθρώπου-ρομποτικού βραχίονα. Επίσης, αναπτύχθηκαν οι κώδικες για την παραλαβή των συντεταγμένων του ανθρώπου και του ρομποτικού βραχίονα. Επομένως, χρησιμοποιώντας τους συγκεκριμένους κώδικες μπορεί κάποιος να κάνει πιο ακριβή έλεγχο όσον αφορά την συνεργασία ανθρώπου – ρομποτικού βραχίονα.

Επιπλέον, μπορούν να γίνουν δοκιμές στο πλάισιο παρακείμενου κύτταρου κατεργασιών όταν το ρομπότ φορτοεκφορτώνει τεμάχια από ταινιόδρομο σε ενδιάμεση αποθήκη και από αυτή στον τόρνο και τη φρέζα CNC που αποτελούν τμήματα του κυττάρου ενώ ο άνθρωπος φορτοεκφορτώνει τεμάχια μεταξύ μιας κεντρικής αποθήκης τεμαχίων και ταινιοδρόμου.

Ακόμα, μπορεί να διερευνηθεί η χρήση νέων συσκευών(αισθητήρες) που προσφέρουν βελτιωμένες δυνατότητες αναγνώρισης κίνησης του ανθρώπου(π.χ. αισθητήρες δύναμης-ροπής του ανθρώπου {άρθρα [26], [27]}, αισθητήρες υπέρυθρων ακτίνων, ρομποτική όραση με κάμερες, κ.α.), ώστε να είναι δυνατή

η αναγνώριση και η παρακολούθηση όλων των κινήσεων των χεριών είτε ολόκληρου του σώματος.

Τέλος, πρέπει να λαμβάνονται υπόψη στην κινηματική του ρομπότ τα ιδιόμορφα σημεία (singularities) και ιδανικά να αποφεύγονται, για την ασφαλέστερη συνεργασία ανθρώπου-ρομποτικού βραχίονα. Αυτό θα μπορούσε να υλοποιηθεί είτε παθητικά (π.χ. να προειδοποείται ο χρήστης όταν ο ρομποτικός βραχίονας πλησιάζει ένα ιδιόμορφο σημείο), ή ενεργητικά (να υπολογίζονται τα ιδιόμορφα σημεία στο εικονικό περιβάλλον και να σχεδιάζεται ανάλογα η συνεργατική τροχιά).

Βιβλιογραφία

[1] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine)

[2] Wikipedia. [Online]. Available: https://el.wikipedia.org/wiki/Arduino

[3] Wikipedia. [Online]. Available: https://el.wikipedia.org/wiki/Ρομποτική

[4] Isaac Asimov. Three laws of Robotics. [Online]. Available: http://users.sch.gr/jenyk/index.php/robotics/robotics-historicalreview/28-the3rulesofrobotics

[5] Εισαγωγή στη Ρομποτική (για αρχάριους) Δημήτρης Πιπερίδης Διαδραστική Έκθεση Επιστήμης & Τεχνολογίας Ίδρυμα Ευγενίδου. [Online]. Available: http://docplayer.gr/2288524-Eisagogi-sti-rompotiki-gia-arharioys-dimitris-piperidis-diadrastiki-ekthesi-epistimis-tehnologias-idryma-eygenidoy.html

[6] Unity Documentation. [Online]. Available: https://docs.unity3d.com/Manual/Namespaces.html

[7] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[8] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/GameObject.html

[9] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/Component.html

[10] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/MeshCollider.html

[11] Unity Documentation. [Online]. Available: https://docs.unity3d.com/Manual/ControllingGameObjectsComponents.html

[12] Unity Documentation. [Online]. Available: https://docs.unity3d.com/Manual/Coroutines.html

[13] Unity Documentation. [Online]. Available: https://docs.unity3d.com/Manual/NullReferenceException.html

[14] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/CapsuleCollider.html

[15] Adept Technology Inc, "eV+/V+ Language Reference Guide." [Online]. Available: http://www1.adept.com/main/KE/DATA/V Plus/V Language Reference/vlangTOC.html.

[16] Arduino Language Reference. [Online]. Available: https://www.arduino.cc/en/Reference/HomePage

[17] J. J. Craig, "Introduction to Robotics," *IEEE Expert*, vol. 1, no. 2, 1986

[18] Adept Technology Inc, "V + Operating System User ' s Guide," 1997.

[19] Adept Technology Inc, "V + Language," 1997.

[20] Stäubli, "Arm - RX series 90B family Characteristics," 2004.

[21] Μάτσας, Ηλίας, "Ανάπτυξη και αξιολόγηση διαδραστικών μοντέλων εικονικής πραγματικότητας για το σχεδιασμό ρομποτικών συστημάτων παραγωγής.", Διδακτορική Διατριβή, Σχολή Μηχανολόγων Μηχανικών, 2016.

[22] Μάνου, Ευγενία. "Προγραμματισμός ρομπότ σε περιβάλλον εικονικής πραγματικότητας με τη βοήθεια μαγνητικού αισθητήρα έξι βαθμών ελευθερίας", Διπλωματική Εργασία, Σχολή Μηχανολόγων Μηχανικών, 2015.

[23] Ευάγγελος, Τζήμας. "Υποστηριξη διαδικασιών προετοιμασίας εργαλειομηχανών με τη βοήθεια επαυξημένης πραγματικότητας", Διπλωματική Εργασία, Σχολή Μηχανολόγων Μηχανικών, 2015.

[24] Κουρής, Αλέξανδρος. "Εντοπισμός συγκρούσεων και διάκριση από επιθυμητές επαφές κατά τη συνεργασία ανθρώπου-ρομπότ", Διπλωματική Εργασία, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Πατρών, 2016.

[25] Unity Documentation. [Online]. Available: https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html

[26] J. N. Pires, J. Ramming, S. Rauch, and R. Arajo, "Force/torque sensing applied to industrial robotic deburring," Sensor Review, vol. 22, no. 3,

pp. 232–241, 2002.

[27] S. Kuhn, T. Gecks, and D. Henrich, "Velocity control for safe robot guidance based on fused vision and force/torque data," pp. 485 –492, Sept. 2006.

[28] Vasic, Milos, and Aude Billard. "Safety issues in human-robot interactions." *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013.

[29] Kerezovic, Tanja, et al. "HUMAN SAFETY IN ROBOT APPLICATIONS-REVIEW OF SAFETY TRENDS." *Acta Technica Corviniensis-Bulletin of Engineering* 6.4 (2013): 113.

[30] Eder, Kerstin, Chris Harper, and Ute Leonards. "Towards the safety of human-in-the-loop robotics: Challenges and opportunities for safety assurance of robotic co-workers'." *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014.

[31] Haddadin, Sami, Alin Albu-Schäffer, and Gerd Hirzinger. "Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing." *Robotics: Science and Systems*. Vol. 3. 2007.

[32] Marvel, Jeremy A., Joe Falco, and Ilari Marstio. "Characterizing Task-Based Human–Robot Collaboration Safety in Manufacturing." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.2 (2015): 260-275.

[33] Weng, Yueh-Hsuan, Chien-Hsun Chen, and Chuen-Tsai Sun. "Toward the human–robot co-existence society: on safety intelligence for next generation robots." International Journal of Social Robotics 1.4 (2009): 267-282.

[34] Goodrich, Michael A., and Alan C. Schultz. "Human-robot interaction: a survey." Foundations and trends in human-computer interaction 1.3 (2007): 203-275.

[35] Cherubini, Andrea, et al. "Collaborative manufacturing with physical human–robot interaction." Robotics and Computer-Integrated Manufacturing 40 (2016): 1-13.

[36] De Santis, Agostino, et al. "An atlas of physical human–robot interaction." Mechanism and Machine Theory 43.3 (2008): 253-270.

[37] Kulić, Dana, and Elizabeth Croft. "Pre-collision safety strategies for human-robot interaction." Autonomous Robots 22.2 (2007): 149-164.

[38] Shackleford, William, et al. "Performance Evaluation of Human Detection Systems for Robot Safety." Journal of Intelligent & Robotic Systems (2016): 1-19.

[39] Zhang, Ping, et al. "Ensuring safety in human-robot coexisting environment based on two-level protection." Industrial Robot: An International Journal 43.3 (2016).

[40] Bicchi, Antonio, Michael A. Peshkin, and J. Edward Colgate. "Safety for physical human–robot interaction." Springer handbook of robotics. Springer Berlin Heidelberg, 2008. 1335-1348.

[41] Cordero, Cristina Alén, et al. "Experimental tests in human–robot collision evaluation and characterization of a new safety index for robot operation." Mechanism and machine theory 80 (2014): 184-199.

[42] Definition of robot – Robot Institute of America. [Online]. Available : https://www.wyzant.com/resources/lessons/english/etymology/words-mod-robots

[43] Wikipedia. [Online]. Available : https://en.wikipedia.org/wiki/Kinect

[44] Μίχας, Σεραφείμ. "Προγραμματισμός και τηλεπαρακολούθηση λειτουργίας ρομποτικού βραχίονα με τη βοήθεια εικονικής πραγματικότητας και αισθητήρων ηλεκτρονικών συσκευών ευρείας κατανάλωσης" Διπλωματική Εργασία, Σχολή Μηχανολόγων Μηχανικών, 2015.

Παράρτημα: Κώδικες

Απλή αποφυγή

Κώδικες ρομπότ

```
PROGRAM chrisfinal( )
SET #startpoint = #PPOINT(140,-67,157,0,0,0)
SET #finalpoint = #PPOINT(-140,-67,157,0,0,0)
SPEED 30 ALWAYS
MOVE #startpoint
BREAK
MOVE #finalpoint
REACTI 1001, subprog1
REACTI 1002, subprog2
REACTI 1003, subprog3
REACTI 1004, subprog4

PROGRAM subprog1( )
SET #a = #PPOINT(90,-67,157,0,0,0)
SET #b = #PPOINT(90,-85,125,0,0,0)
SET #c = #PPOINT(30,-85,125,0,0,0)
SET #fp1 = #PPOINT(-140,-67,157,0,0,0)
MOVE #a
BREAK
MOVE #b
BREAK
MOVE #c
BREAK
MOVE #fp1

PROGRAM subprog2( )
SET #d = #PPOINT(65,-67,157,0,0,0)
SET #e = #PPOINT(65,-85,125,0,0,0)
SET #f = #PPOINT(-45,-85,125,0,0,0)
SET #fp2 = #PPOINT(-140,-67,157,0,0,0)
MOVE #d
BREAK
MOVE #e
BREAK
MOVE #f
BREAK
MOVE #fp2

PROGRAM subprog3( )
SET #g = #PPOINT(10,-67,157,0,0,0)
SET #h = #PPOINT(10,-85,125,0,0,0)
SET #i = #PPOINT(-90,-85,125,0,0,0)
SET #fp3 = #PPOINT(-140,-67,157,0,0,0)
MOVE #g
BREAK
MOVE #h
```

```
BREAK
MOVE #i
BREAK
MOVE #fp3

PROGRAM subprog4( )
SET #j = #PPOINT(-25,-67,157,0,0,0)
SET #k = #PPOINT(-25,-85,125,0,0,0)
SET #l = #PPOINT(-150,-85,125,0,0,0)
SET #fp4 = #PPOINT(-140,-67,157,0,0,0)
MOVE #j
BREAK
MOVE #k
BREAK
MOVE #l
BREAK
MOVE #fp4
```

Κώδικες Unity3d

```
Program Perioxes_Collider( )
void Start ( )
{
        sima1 = false;
        sima2 = false;
        sima3 = false;
        sima4 = false;
}
void OnTriggerEnter(Collider other)
{
        if (other.gameObject.tag == “Perioxi1”)
        {
                sima1 = true;
                Debug.Log (“Eimai stin perioxi 1”);
        }
        if (other.gameObject.tag == “Perioxi2”)                 {
                sima2 = true;
                Debug.Log (“Eimai stin perioxi 2”);
        }
        if (other.gameObject.tag == “Perioxi3”)
        {
                sima3 = true;
                Debug.Log (“Eimai stin perioxi 3”);
        }
        if (other.gameObject.tag == “Perioxi4”)
        {
                sima4 = true;
                Debug.Log (“Eimai stin perioxi 4”);
        }
}

Program Arduino( )
using UnityEngine;
using SystemCollections;
using System.IO.Ports;
```

```
public class Arduino : MonoBehaviour
{
	private SerialPort sp;
	void Start ( )
	{
		sp = new SerialPort(“COM4”, 9600);
		sp.Open ( );
	}
	void Update ( )
	{
		char[ ] s = new char[ 1 ];
		bool person_found = false;
		if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Pe
		rioxes_Collider>( ).sima1)
		{
			person_found = true;
			s[ 0 ] = “1” ;
			sp.Write(s, 0 , 1);
		}
		if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Pe
		rioxes_Collider>( ).sima2)
		{
			person_found = true;
			s[ 0 ] = “2” ;
			sp.Write(s, 0 , 1);
		}
		if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Pe
		rioxes_Collider>( ).sima3)
		{
			person_found = true;
			s[ 0 ] = “3” ;
			sp.Write(s, 0 , 1);
		}
		if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Pe
		rioxes_Collider>( ).sima4)
		{
			person_found = true;
			s[ 0 ] = “4” ;
			sp.Write(s, 0 , 1);
		}
		if(!person_found)
	{
		s[ 0 ] = “0” ;
		sp.Write(s, 0 , 1);		}
	}
}
```

Κώδικας ArduinoMega

```
Finalprogram( )
#inlude <SoftwareSerial.h>
void setup( )
{
	int pin1 = 22;
	int pin2 = 24;
	int pin3 = 26;
```

```
        int pin4 = 28;
        Serial.begin (9600);
        pinMode (pin1, OUTPUT);
        pinMode (pin2, OUTPUT);
        pinMode (pin3, OUTPUT);
        pinMode (pin4, OUTPUT);
}
void loop( )
{
        if (Serial.available( ) > 0)
        {
                char incoming_char = (char)Serial.read ( );
                if (incoming_char == '1')
                {
                        writeToPin1( );
                }
                if (incoming_char == '2')
                {
                        writeToPin2( );
                }
                if (incoming_char == '3')
                {
                        writeToPin3( );
                }
                if (incoming_char == '4')
                {
                        writeToPin4( );
                }
                if (incoming_char == '0')
                {
                        nullAllPins( );
                }
        }
}
void writeToPin1( )
{
        nullAllPins( );
        digitalWrite(pin1, HIGH);
}
void writeToPin2( )
{
        nullAllPins( );
        digitalWrite(pin2, HIGH);
}
void writeToPin3( )
{
        nullAllPins( );
        digitalWrite(pin3, HIGH);
}
void writeToPin4( )
{
        nullAllPins( );
        digitalWrite(pin4, HIGH);
}
void nullAllPins( )
{
```

```
        digitalWrite(pin1, LOW);
        digitalWrite(pin2, LOW);
        digitalWrite(pin3, LOW);
        digitalWrite(pin4, LOW);        }
}
```

Σύνθετη αποφυγή

Κώδικες ρομπότ

```
PROGRAM chrisfinal( )
SET #startpoint = #PPOINT(140,-67,157,0,0,0)
SET #finalpoint = #PPOINT(-140,-67,157,0,0,0)
SPEED 30 ALWAYS
MOVE #startpoint
BREAK
MOVE #finalpoint
REACTI 1001, subprog1
REACTI 1002, subprog2
REACTI 1003, subprog3
REACTI 1004, subprog4

PROGRAM subprog1( )
SET #a = #PPOINT(90,-67,157,0,0,0)
SET #b = #PPOINT(90,-85,125,0,0,0)
SET #c = #PPOINT(30,-85,125,0,0,0)
SET #fp1 = #PPOINT(-140,-67,157,0,0,0)
MOVE #a
BREAK
IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN
        RETURN
END
MOVE #b
BREAK
IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN
        RETURN
END
MOVE #c
BREAK
IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN
        RETURN
END
MOVE #fp1
IF SIG(1002) OR SIG(1003) OR SIG(1004) THEN
        RETURN
END

PROGRAM subprog2( )
SET #d = #PPOINT(65,-67,157,0,0,0)
SET #e = #PPOINT(65,-85,125,0,0,0)
SET #f = #PPOINT(-45,-85,125,0,0,0)
SET #fp2 = #PPOINT(-140,-67,157,0,0,0)
MOVE #d
BREAK
IF SIG(1003) OR SIG(1004) THEN
```

```
        RETURN
END
MOVE #e
BREAK
IF  SIG(1003) OR SIG(1004) THEN
        RETURN
END
MOVE #f
BREAK
IF  SIG(1003) OR SIG(1004) THEN
        RETURN
END
MOVE #fp2
IF  SIG(1003) OR SIG(1004) THEN
        RETURN
END


PROGRAM subprog3( )
SET #g = #PPOINT(10,-67,157,0,0,0)
SET #h = #PPOINT(10,-85,125,0,0,0)
SET #i = #PPOINT(-90,-85,125,0,0,0)
SET #fp3 = #PPOINT(-140,-67,157,0,0,0)
MOVE #g
BREAK
IF  SIG(1004) THEN
        RETURN
END
MOVE #h
BREAK
IF  SIG(1004) THEN
        RETURN
END
MOVE #i
BREAK
IF SIG(1004) THEN
        RETURN
END
MOVE #fp3
IF  SIG(1004) THEN
        RETURN
END

PROGRAM subprog4( )
SET #j = #PPOINT(-25,-67,157,0,0,0)
SET #k = #PPOINT(-25,-85,125,0,0,0)
SET #l = #PPOINT(-150,-85,125,0,0,0)
SET #fp4 = #PPOINT(-140,-67,157,0,0,0)
MOVE #j
BREAK
MOVE #k
BREAK
MOVE #l
BREAK
MOVE #fp4
```

Σταμάτημα ρομποτικού βραχίονα

Κώδικες ρομπότ

```
PROGRAM CHRISTASK3( )

SET #sp = #PPOINT(140,-67,157,0,0,0)

SET #fp = #PPOINT(-140,-67,157,0,0,0)

SPEED 10 ALWAYS

MOVE #sp

BREAK

1       MOVE #fp

        REACTI 1001, stop

GOTO 1


PROGRAM STOP()

        WAIT (SIG(1002) OR SIG(1003))

RETURN
```

Κώδικες Unity3d

Program Perioxes_Collider( )

```
void Start ( )                  {

        sima1 = false;

        sima2 = false;

        sima3 = false;          }

void OnTriggerEnter(Collider other)   {

        if (other.gameObject.tag == "Perioxi1")                 {

                sima1 = true;

                Debug.Log ("Eimai ston xwro ergasias tou robot");    }

        if (other.gameObject.tag == "Perioxi2")                 {

                sima2 = true;

                Debug.Log ("Eimai eksw apo ton xwro ergasias tou robot");  }
```

```
        if (other.gameObject.tag == “Perioxi3”)                    {
                sima3 = true;
                Debug.Log (“Eimai eksw apo ton xwro ergasias tou robot”);  }
}
Program Arduino( )
using UnityEngine;
using SystemCollections;
using System.IO.Ports;
public class Arduino : MonoBehaviour{
private SerialPort sp;
void Start ( )              {
        sp = new SerialPort(“COM4”, 9600);
        sp.Open ( );     }
void Update ( ) {
        char[ ] s = new char[ 1 ];
        if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Perioxes_
        Collider>( ).sima1)      {
        s[ 0 ] = “1” ;
        sp.Write(s, 0 , 1);         }
        if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Perioxes_
        Collider>( ).sima2)      {
        s[ 0 ] = “2” ;
        sp.Write(s, 0 , 1);         }
        if(GameObject.FindGameObjectWithTag(“liakous”).GetComponent<Perioxes_
        Collider>( ).sima3)      {
        s[ 0 ] = “3” ;
        sp.Write(s, 0 , 1);         }
}
```

Κώδικας ArduinoMega

```
Program Task3( )
```

```
#inlude <SoftwareSerial.h>
void setup( )                {
        int pin1 = 22;
        int pin2 = 24;
        int pin3 = 26;
        Serial.begin (9600);
        pinMode (pin1, OUTPUT);
        pinMode (pin2, OUTPUT);
        pinMode (pin3, OUTPUT);
}
void loop( )                 {
if (Serial.available( ) > 0)         {
        char incoming_char = (char)Serial.read ( );
        if (incoming_char == '1')        {
        digitalWrite(pin1, HIGH);
        digitalWrite(pin2, LOW);
        digitalWrite(pin3, LOW);         }
        if (incoming_char == '2')        {
        digitalWrite(pin2, HIGH);
        digitalWrite(pin1, LOW);
        digitalWrite(pin3, LOW);         }
        if (incoming_char == '3')        {
        digitalWrite(pin3, HIGH);
        digitalWrite(pin1, LOW);
        digitalWrite(pin2, LOW);         }
        }
}
```

Παραλαβή συντεταγμένων ρομποτικού βραχίονα

Κώδικας ρομπότ

```
Program serialdata ( )

AUTO slun

AUTO $text

SET #sp = #PPOINT(140,-67,157,0,0,0)

SET #fp = #PPOINT(-140,-67,157,0,0,0)

SPEED 10 ALWAYS

MOVE #sp

MOVE #fp

ATTACH  (slun, 4)  "SERIAL: 2"

FOR    y = 0 TO 10000

        HERE b1

        DECOMPOSE x[ ] = b1

        FOR i = 0 TO 2

                WRITE (slun) x[i]

        END

END

DETACH  (slun)
```

Κώδικας ArduinoMega

```
Robot_position( )

#inlude <SoftwareSerial.h>

String inString = '''';

float pos[3];

int i;


void setup( )              {
```

```
        Serial.begin (9600);
        Serial1.begin(9600);
void loop( )            {
        while (Serial1.available( ) > 0)  {
                char  inChar = (char)Serial1.read ( );
                if (inChar == 'x')        {
                        i = 0;              }
                else if (inChar == '\n')          {
                      inString +=  inChar;    }
                else      {
                        float num = inString.toFloat();
                          pos[i] = num;
                          inString = '''';
                          i = i + 1;          }
                if (i == 3)        {
                          Serial.println("robot X coordinate is :");
                          Serial.println(pos[1], 4);
                          Serial.println("robot Y coordinate is :");
                          Serial.println(pos[2], 4);
                          Serial.println("robot     Z coordinate is :");
                          Serial.println(pos[3],4);
                }
        }
}
```

Παραλαβή συντεταγμένων ανθρώπου

Κώδικας Unity3d

Program Avatar_Position( )

```
using UnityEngine;
```

```
using System.Collections;
using System.IO;
using System.IO.Ports;
public class Avatar_Position : Monobehaviour {
        private SerialPort sp;
        public GameObject avatar_center;
        public GameObject robot_center;
        public Vector3 avatar_Pos;
        public Vector3 robot_Pos;
        public Vector3 relative_Pos;
        void Start ( )    {
                sp = new SerialPort ("COM6" , 9600);
                sp.ReadTimeout = 50;
                sp.Open ( );              }
        void Update ( ) {
                Coordinates ( );          }
        void Coordinates ( )    {
                int[ ] xcoordinates;
                int[ ] ycoordinates;
                int[ ] zcoordinates;
                int lengthOfArray = 100;
                xcoordinates = new int[lengthOfArray];
                ycoordinates = new int[lengthOfArray];
                zcoordinates = new int[lengthOfArray];
                for (int i = 0; i < lengthOfArray; i++)   {
                        avatar_Pos = avatar_center.transform.position;
                        robot_Pos = robot_center.transform.position;
                        relative_Pos = -1000.0f * (avatar_Pos – robot_Pos);
                        int positionx = (int)relative_Pos.x;
```

```
                int positiony = (int)relative_Pos.z;
                int positionz = (int)relative_Pos.y;
                xcoordinates[i] = positionx;
                ycoordinates[i] = positiony;
                zcoordinates[i] = positionz;
                if (i == 99) {
                        positionx = xcoordinates[i];
                        positiony = ycoordinates[i];
                        positionz = zcoordinates[i];
                        string _x = positionx.ToString( );
                        string _y = positiony.ToString( );
                        string _z = positionz.ToString( );
                        Debug.Log ( _x + ',' + _y + ',' + _z);
                        sp.Write ("x");
                        sp.Write (_x);
                        sp.Write ("y");
                        sp.Write (_y);
                        sp.Write ("z");
                        sp.Write (_z);
                        sp.Write ("e");
                }
        }
}
}
```

Κώδικας ArduinoMega

Human_position( )

```
#inlude <SoftwareSerial.h>
String inString = '''';
boolean x = false;
```

```
boolean y = false;
boolean z = false;
int xcoord;
int ycoord;
int zcoord;
void setup( )              {
        Serial.begin (9600);
        Serial1.begin(9600);
void loop( )               {
        while (Serial1.available( ) > 0)  {
                char  inChar = (char)Serial1.read ( );
                if (inChar == 'x')        {
                        x = true;         }
                else if (inChar == 'y')  {
                        y = true;
                        x = false;
                        xcoord = inString.toInt ( );
                        inString = '''';     }
                else if   (inChar == 'z') {
                        z = true;
                        y = false;
                        ycoord = inString.toInt ( );
                        inString = '''';     }
                else if  ((inChar != 'x') && (inChar != 'y') && (inChar != 'z') && (inChar
!= 'e'))  {
                        if (x == true)    {
                                inString += inChar;     }
                        if (y == true)    {
                                inString += inChar;     }
```

```
                if (z == true)    {
                        inString += inChar;     }
        }
        else if   (inChar == 'e') {
                z = false;
                zcoord = inString.toInt ( );
                inString = '''';
                Serial.println("human X coordinate is :");
                Serial.println(xcoord);
                Serial.println("human Y coordinate is :");
                Serial.println(ycoord);
                Serial.println("human Z coordinate is :");
                Serial.println(zcoord);
        }
    }
}
```