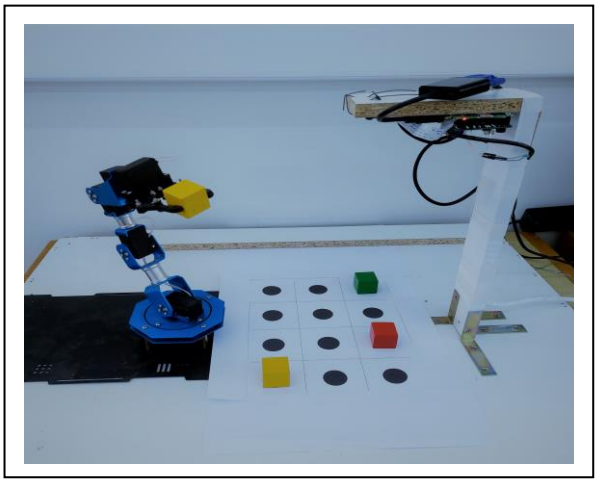
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ ΚΑΙ ΕΥΦΥΗ ΠΕΡΙΒΑΛΛΟΝΤΑ»

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη ρομποτικού βραχίονα ελεγχόμενου μέσω μικροελεγκτή



Μεταπτυχιακός Φοιτητής : Μιχάλης Λαζαρίδης, ΑΜ: msciot18002
Επιβλέπων : Γρηγόριος Καλτσάς, Καθηγητής

ΑΙΓΑΛΕΩ, ΦΕΒΡΟΥΑΡΙΟΣ 2023

UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

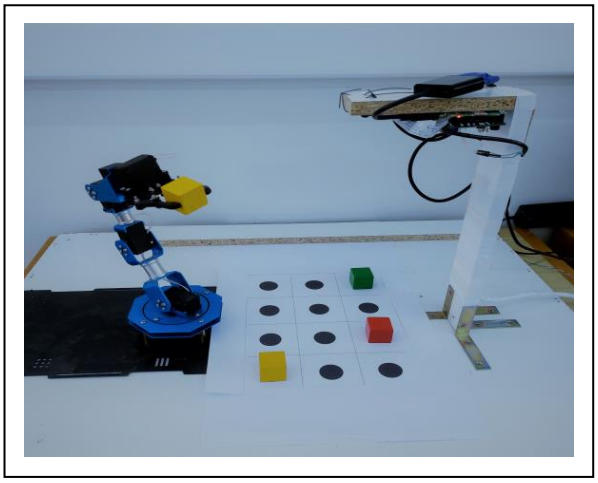
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Master of Science in

"INTERNET of THINGS AND INTELLIGENT ENVIRONMENTS"

MSc Thesis

Development of a robotic arm controlled by a microcontroller



Student: Lazaridis Michael, Registration Number msciot18002

MSc Thesis Supervisor: Grigoris Kaltsas, Professor

ATHENS-EGALEO, FEBRUARY 2023

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

Καλτσάς Γρηγόριος Καθηγητής (Επιβλέπων)	Αλεξανδρίδης Αλέξανδρος Καθηγητής (Μέλος)	Κανδρής Ξενοφών-Διονύσιος Καθηγητής (Μέλος)
 Digitally signed by Grigorios Kaltsas Date: 2023.02.22 20:44:32 +02'00'	 Digitally signed by Alexandros Alexandridis Date: 2023.02.20 10:49:46 +02'00'	XENOFON DIONYSIOS KANDRIS Digitally signed by XENOFON DIONYSIOS KANDRIS Date: 2023.02.20 10:38:48 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Λαζαρίδης Μιχάλης του Κωσταντίνου, με αριθμό μητρώου msciot18002, μεταπτυχιακός φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

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

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου».

Ο Δηλών

Μιχάλης Λαζαρίδης

ΠΕΡΙΛΗΨΗ

Η διπλωματική εργασία αυτή είναι αποτέλεσμα μίας γενικότερης προσπάθειας κατανόησης της λειτουργίας των ρομποτικών βραχιόνων και των εφαρμογών τους με αναφορές από τις πρότερες εφευρέσεις έως και σήμερα και ολοκληρώνεται με την υλοποίηση μίας πειραματικής διάταξης ρομποτικού βραχίονα που εκτελεί μία τυπική παραγωγική εργασία όπως αυτή της αναγνώρισης και της ταξινόμησης αντικειμένων στο χώρο.

Στα πλαίσια της εργασία θα πραγματοποιηθεί ιστορική περιγραφή των αρχικών προσπαθειών υλοποίησης ρομποτικών βραχιόνων και τα χρονικά σημεία που έγιναν τα μεγαλύτερα βήματα εξέλιξης της τεχνολογίας τους. Επίσης θα πραγματοποιηθεί περιγραφή των σημαντικότερων εφαρμογών που έχουν στους διάφορους τομείς της ανθρώπινης δραστηριότητας όπως είναι ο στρατιωτικός, ο διαστημικός, ο γεωργικός, ο ιατρικός τομέας αλλά και αυτός της βιομηχανικής παραγωγής που έπαιξε και τον καθοριστικότερο ρόλο στην δημιουργία και εξέλιξη των ρομποτικών βραχιόνων.

Εν συνεχεία παρουσιάζονται οι βασικές διαμορφώσεις ρομποτικών βραχιόνων και των τμημάτων που απαρτίζονται. Γίνεται περιγραφή των βασικών συστημάτων τους όπως της κίνησης, της αίσθησης και του ελέγχου τους καθώς και των χαρακτηριστικών μεγεθών που τους καθορίζουν. Ακόμη δίνεται έμφαση στο πεδίο της ρομποτικής όρασης που είναι ταχέως αναπτυσσόμενο και αποτελεί ένα σημαντικό κομμάτι στην ανάπτυξη νέων εφαρμογών.

Τέλος υλοποιείται μία πειραματική διάταξη ρομποτικού βραχίονα κάνοντας χρήση του μικροελεγκτή Raspberry Pi και μίας κάμερας όπου επιχειρείται η οπτική αναγνώριση μικρών αντικειμένων με τη χρήση του ελεύθερου λογισμικού OpenCV και ενός αλγορίθμου που αναπτύχθηκε αποκλειστικά για τις ανάγκες της εργασίας και που δίνει τη δυνατότητα μετακίνησης του βραχίονα κατά τρόπο που να συλλέγει αντικείμενα και να τα χωροθετεί.

ΛΕΞΕΙΣ – ΚΛΕΙΔΙΑ: Δομή Ρομποτικών Βραχιόνων, Εντοπισμός και Μετακίνηση Αντικειμένων με χρήση Ρομποτικού Βραχίονα, Έλεγχος και καθοδήγηση Ρομποτικού Βραχίονα, Ιστορία Ρομποτικών Βραχιόνων, Ρομποτική Όραση.

ABSTRACT

This dissertation is the result of a general effort to understand the operation and applications of robotic arms with historical references from previous inventions to the present day and is resulting to the implementation of a robotic arm experimental device that performs a typical productive task such as that of identifying and sorting objects in space.

The work will provide a historical description of the initial efforts to implement robotic arms and the time points that became the biggest steps in the evolution of their technology. There will also be a description of the most important applications in the various fields of human activity, such as the military, space, agriculture, medicine and industrial production, which played the most decisive role in the creation and development of robotic arms.

Then the basic configurations of robotic arms and the components that are made are presented. Their basic systems such as movement, sensation and control as well as the characteristic quantities that define them are described. Emphasis is also placed on the field of robotic vision which is rapidly developing and is an important part of developing new applications.

Finally, an experimental robotic arm assembly is implemented using the Raspberry Pi microcontroller and a camera where the visual recognition of small objects implemented using the free OpenCV software and an algorithm developed exclusively for the needs of the present thesis and which enables the movement of the arm in a way to collect objects and place them.

KEYWORDS: Robotic Arm Structure, Locating and Moving Objects Using a Robotic Arm, Robotic Arm Control and Guidance, Robotic Arm History, Robotic Vision.

Ευχαριστίες

Ευχαριστώ τον εισηγητή και επιβλέποντα της μεταπτυχιακής διπλωματικής μου εργασίας Καθηγητή Γρηγόριο Καλτσά, Διευθυντή του Μεταπτυχιακού Προγράμματος Σπουδών “Διαδίκτυο των Πραγμάτων και Ευφυή Περιβάλλοντα ” για την καθοδήγηση και αμέριστη υποστήριξή του στην ολοκλήρωση της διπλωματικής μου καθώς και τους φίλους και συναδέλφους Δανιήλ Κανταλέα και Πέτρο Πάνο για την παροχή συμβουλών και την παρέα τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Στην παρούσα πτυχιακή εργασία υλοποιείται μία πειραματική διάταξη Ρομποτικού Βραχίονα που με τη χρήση οπτικής κάμερας επιχειρεί την οπτική αναγνώριση και το χειρισμό αντικειμένων. Η συνδεσμολογία και το λογισμικό που χρησιμοποιούνται μας δίνουν πειραματικά δεδομένα και μας οδηγούν σε συμπεράσματα από τα οποία προκύπτουν δυνατότητες για βελτίωση της εφαρμογής αυτής. Επιχειρείται η περιγραφή της ιστορικής εξέλιξης της τεχνολογίας ρομποτικών βραχιόνων από την αρχαιοελληνική περίοδο μέχρι σήμερα. Προκειμένου να γίνει κατανοητή η σημασία της εξέλιξης τους περιγράφονται οι εφαρμογές που έχουν οι Ρομποτικοί Βραχίονες σε διάφορους τομείς αλλά και οι διαμορφώσεις τους από τεχνολογικής πλευράς. Εμφαίνετε ότι θα υπάρχει μεγάλη συμμετοχή της τεχνολογίας ρομποτικών διατάξεων όπως αυτή του ρομποτικού βραχίονα σχεδόν σε όλους τους τομείς του ανθρώπου και ότι αυτή θα αυξηθεί τα επόμενα χρόνια αποτελώντας μέρος της 4ης Βιομηχανικής επανάστασης [1].

Το πρώτο κεφάλαιο αναφέρεται στις πρώτες καταγεγραμμένες διατάξεις ρομποτικών βραχιόνων αλλά και στην ιστορική εξέλιξη τους από τα μέσα του περασμένου αιώνα μέχρι και το εγγύς μέλλον όπου αναμένεται να εμφανιστούν πολλά και πλήρως λειτουργικά μοντέλα που θα ενσωματώνονται όλο και περισσότερο στην παραγωγική δραστηριότητα. Γίνεται ακόμη μία εκτενής περιγραφή των εφαρμογών και των πλεονεκτημάτων που κατέχουν σε διάφορες περιοχές της ανθρώπινης δραστηριότητας όπου και ενσωματώνονται.

Το δεύτερο κεφάλαιο αναφέρεται στις βασικές διαμορφώσεις των ρομποτικών βραχιόνων όπου περιγράφονται η δομή και η λειτουργία τους, τα χαρακτηριστικά μεγέθη που τους καθορίζουν και τα συστήματα κίνησης, αίσθησης και ελέγχου που τους απαρτίζουν. Γίνεται ακόμη μία αναφορά στη ρομποτική όραση και πως αυτή συμβάλει στην επίλυση προβλημάτων αναγνώρισης και ταυτοποίησης.

Στο τρίτο κεφάλαιο πραγματοποιείται η περιγραφή μίας πειραματικής διάταξης ρομποτικού βραχίονα και των τμημάτων που αποτελείται καθώς επίσης και του λογισμικού που χρησιμοποιήθηκε για τον έλεγχό της. Αναφέρεται η διαδικασία που ακολουθήσαμε για να εξάγουμε πειραματικά αποτελέσματα και ειδικότερα η υλοποίηση ενός πρωτότυπου αλγορίθμου που κάνοντας χρήση βιβλιοθηκών ελεύθερων λογισμικών όπως το OpenCV καθώς και μίας οπτικής κάμερας συνδεδεμένης σε έναν μικροελεγκτή Raspberry Pi, εκτελεί διαδικασία εντοπισμού και χειρισμού μικρών αντικειμένων στο χώρο.

ΚΕΦΑΛΑΙΟ 1

Ιστορική εξέλιξη και εφαρμογές Ρομποτικών Βραχιόνων

Error! Reference source not found.

Η δημιουργία και ανάπτυξη του ρομποτικού βραχίονα μπορεί να θεωρηθεί ιστορικά ότι ξεκινάει στην αρχαία Ελλάδα με την κατασκευή διάφορων αυτομάτων και ανθρωπόμορφων μηχανών όπως αυτό της βαδίζουσας αυτόματης θεραπαινίδας του Φίλωνος του Βυζαντίου 3ος π.χ. (Σχήμα 1) η οποία σαν εργασία είχε να γεμίζει τα κύπελλα των καλεσμένων με κρασί και νερό [2].

Σχήμα 1: Η αυτόματη θεραπαινίς του Φίλωνος του Βυζαντίου (3ος αι. π.Χ.)

Στην σύγχρονη εποχή υπάρχει μια σαφής τάση στην κατασκευή ρομποτικών βραχιόνων να γίνονται ολοένα και ποιο ικανοί, με περισσότερους άξονες κίνησης (βαθμούς ελευθερίας) και δυνατότητες από αυτές του ανθρώπινου χεριού. Ο da Vinci σχεδίασε το πρώτο εξελιγμένο ρομποτικό βραχίονα το 1495 με τέσσερις βαθμούς ελευθερίας και έναν αναλογικό ελεγκτή που παρείχε ισχύ και προγραμματισμό.

Το 1954 σχεδιάστηκε το πρώτο προγραμματιζόμενο ρομπότ από τον George Devol ο οποίος επινόησε τον όρο Universal Automation. Ο ίδιος και ο μηχανικός Joseph Engelberger ίδρυσαν την πρώτη εταιρεία ρομπότ παγκοσμίως, την Unimation το 1956. Η Unimation αγοράστηκε από την Condec Corporation και ξεκίνησε η ανάπτυξη των συστημάτων ρομπότ Unimate το 1960. Το ίδιο έτος η American Machine and Foundry, αργότερα γνωστή ως AMF

Corporation, διέθεσε ένα ρομπότ, που ονομάζεται Versatran και σχεδιάστηκε από τον Harry Johnson και τον Veljko Milenkovic.

Το πρώτο βιομηχανικό ρομπότ που ενσωματώθηκε σε ένα εργοστάσιο αυτοκινήτων της General Motors στο New Jersey ήταν το UNIMATE του Devol και του Engelberger το 1962. Πραγματοποίησε επιτόπια συγκόλληση και εξήγαγε χυτά καλουπώματα.

Το 1969 η Nachi ξεκινά τη ρομποτική της επιχείρηση και το 1973 η γερμανική εταιρεία robotics KUKA, δημιουργεί το πρώτο βιομηχανικό ρομπότ με έξι ηλεκτρομηχανικούς άξονες με την ονομασία Famulus (Σχήμα 2β). Το 1974 ο David Silver φοιτητής στο MIT σχεδίασε ένα ρομποτικό βραχίονα (Silver Arm, Σχήμα 2γ) που πραγματοποιούσε συναρμολόγηση μικρών τεμαχίων χρησιμοποιώντας ανατροφοδότηση από αισθητήρες αφής και πίεσης.

Ο καθηγητής Scheinman, ο κύριος του έργου του Stanford Arm (Σχήμα 2α), δημιούργησε την Vicarm Inc η οποία διέθετε στην αγορά μια έκδοση του βραχίονα για βιομηχανικές εφαρμογές. Ο νέος βραχίονας ελεγχόταν από ένα μικροϋπολογιστή. Το 1974 τα βιομηχανικά ρομπότ αναπτύχθηκαν και εγκαταστάθηκαν στο εργοστάσιο της Fanuc. Η παραγωγή και πώληση των σερβοκινητήρων DC ξεκίνησε με την άδεια της GETTYS MANUFACTURING CO., INC [3].

(α) (β) (γ)

Σχήμα 2: α) Stanford Arm 1969, β) Famulus (Kuka 1973), γ) Silver Arm 1974

Το 1977 παρουσιάστηκε το Motoman L10 (Σχήμα 3β). Είχε πέντε άξονες και μέγιστο φόρτο εργασίας 10 κιλών, το οποίο περιλάμβανε και τη λαβή. Ζύγισε 470 κιλά. Το Motoman L10 ήταν το πρώτο ρομπότ που εισήγαγε η Yaskawa στην αγορά. Το 1977 η ASEA, μια ευρωπαϊκή εταιρεία ρομπότ, άρχισε να προσφέρει δύο μεγέθη ηλεκτροκίνητων βιομηχανικών ρομπότ. Και τα δύο ρομπότ χρησιμοποιούσαν για ελεγκτή έναν μικροϋπολογιστή που ήλεγχε τη λειτουργία και το προγραμματισμό τους. Η Unimation αγοράζει την Vicarm Inc. κατά τη διάρκεια αυτού του έτους. Το 1978 η Vicarm, Unimation δημιουργεί το ρομπότ PUMA (προγραμματιζόμενη γενική μηχανή για συναρμολόγηση) με υποστήριξη από τη General Motors (Σχήμα 3γ). Πολλά ερευνητικά εργαστήρια χρησιμοποιούν ακόμα αυτό το ρομπότ συναρμολόγησης. Το 1979 η Nachi ανέπτυξε τα πρώτα ρομπότ με κινητήρα με βασική εργασία τη σημειακή συγκόλληση (Σχήμα 3δ). Το 1980 η βιομηχανία ρομπότ ξεκινάει την ταχεία ανάπτυξή της, με ένα νέο ρομπότ ή μία νέα εταιρεία να εισέρχεται στην αγορά κάθε μήνα. Το 1981 η Takeo Kanade χτίζει τον άμεσο βραχίονα κίνησης. Είναι οι πρώτοι που έχουν τοποθετήσει μοτέρ απευθείας στις αρθρώσεις του βραχίονα. Αυτή η αλλαγή το κάνει πιο γρήγορο και πολύ πιο ακριβές από τους προηγούμενους ρομποτικούς βραχίονες.

Το 1985 ο OTC DAIHEN έγινε ο επίσημος προμηθευτής ρομπότ στην Miller Electric Company. Ο Miller επέλεξε να εκχωρήσει διαφορετικούς αριθμούς μοντέλων στα ρομπότ που πωλούνται στη βορειοαμερικανική αγορά. Το πρόθεμα των γραμμάτων στο μοντέλο με "MR"

για το Robot Miller. Τα Ιαπωνικά μοντέλα εμφάνισαν το δικό τους αριθμό και όνομα. Το 1987 η ASEA (Σχήμα 3α) από τη Βαστέρα της Σουηδίας (που ιδρύθηκε το 1883) και η BBC Brown Boveri Ltd απο τη Μπάντεν της Ελβετίας (που ιδρύθηκε το 1891) ανακοίνωσαν σχέδια για τη δημιουργία της ABB Asea Brown Boveri Ltd., με έδρα τη Ζυρίχη της Ελβετίας. Κάθε πλευρά θα κατείχε το 50% της νέας εταιρείας. Το 1988 το σύστημα ελέγχου Motoman ERC εισήχθη με την ικανότητα να ελέγχει μέχρι και 12 άξονες, περισσότερους από οποιονδήποτε άλλο ελεγκτή την εποχή εκείνη. Το 1989 το ιδρύθηκε η Nachi Technology Inc., ΗΠΑ [3].

(α) (β) (γ) (δ)

Σχήμα 3: α) ASEA 1975, (β) Motoman 1977, γ) PUMA 1978, δ) Nachi 1979

Το 1992 ιδρύθηκε το σχολείο ρομποτικής της FANUC. Το 1994 το σύστημα ελέγχου MRC Motoman (Σχήμα 4) εισήχθη με την ικανότητα ελέγχου έως 21 αξόνων. Θα μπορούσε επίσης να συγχρονίσει τις κινήσεις δύο ρομπότ. Το 1996 η Nachi επεκτείνει την ρομποτική εφαρμογή με το εργαλείο κοπής και άλλες σειρές προϊόντων. Το 1998 η εισαγωγή του ελεγκτή XRC επέτρεψε τον έλεγχο έως και 27 αξόνων και τον συγχρονισμένο έλεγχο τριών έως τεσσάρων ρομπότ. Η σειρά Motoman UP εισήγαγε ένα απλούστερο βραχίονα ρομπότ που ήταν πιο εύκολα προσβάσιμο για συντήρηση και επισκευή (Σχήμα 4).

Σχήμα 3: Ιστορία των ρομποτικών βραχιόνων Motoman της εταιρίας Yaskawa

Η Honda συνέβαλε αποφασιστικά στην ανάπτυξη τόσο της σειράς UP βραχιόνων όσο και του ελεγκτή του βραχίονα XRC. Το 2003 η OTC DAIHEN εισήγαγε τη σειρά Almega AX (Σχήμα 5α), μια σειρά από ρομπότ για συγκόλληση τόξου και χειρισμού αρκετά διαδεδομένα πλέον.

Ακολούθησαν πολλοί και εξειδικευμένοι ρομποτικοί βραχίονες με βελτιωμένη ακρίβεια, βάρος εργασίας, εμβέλεια βραχίονα και άλλα χαρακτηριστικά βελτιστοποιημένα χαρακτηριστικά (Σχήματα 5β,5γ) [3].

(α) (β) (γ)

Σχήμα 4: α) Almega AX (2005), β) Yumi dual arm robot (2015), γ) Future of robotic Arms

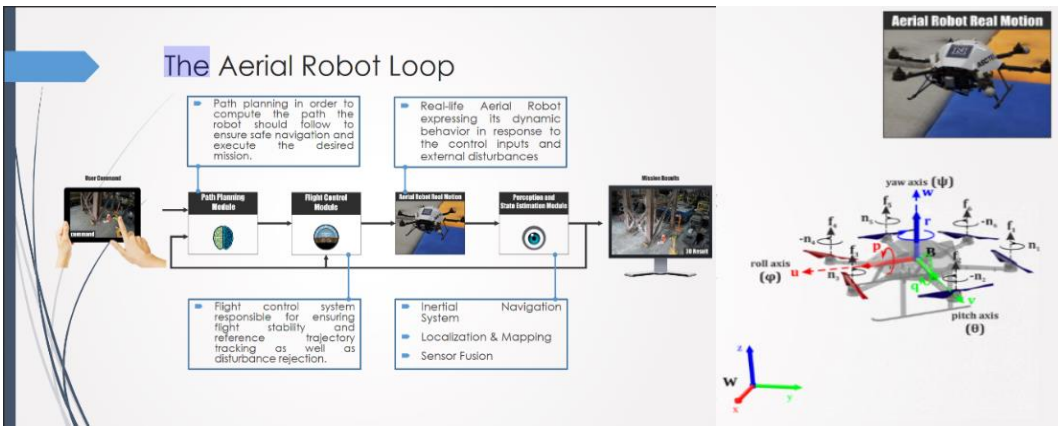
1.2 Εφαρμογές ρομποτικών βραχιόνων

Μπορούμε να πούμε ότι οι εφαρμογές της ρομποτικής και ειδικότερα των ρομποτικών βραχιόνων είναι πολυάριθμες αλλά και ποικίλες. Όπου ενέχεται ανθρώπινη εργασία μπορεί να γίνει μερική η ολική αντικατάσταση αυτής από ρομποτικές διατάξεις. Μερικοί από τους τομείς αυτούς αναφέρονται στη συνέχεια:

Εναέρια ρομποτική και μη επανδρωμένα αεροσκάφη (Aerial Robotics and Unmanned Aerial Vehicles)

Ένα μη επανδρωμένο εναέριο όχημα (UAV) (ή χωρίς πλήρωμα αεροσκάφος, ποιο ευρέως γνωστό ως drone) είναι ένα αεροσκάφος χωρίς ανθρώπινο πιλότο επί του σκάφους και ένα είδος μη επανδρωμένου οχήματος [4,5]. Τα UAV αποτελούν μέρος ενός συστήματος μη επανδρωμένων αεροσκαφών (UAS), τα οποία περιλαμβάνουν ένα UAV, έναν επίγειο ελεγκτή και ένα σύστημα επικοινωνίας μεταξύ των δύο. Η πτήση των UAVs μπορεί να λειτουργεί με διάφορους βαθμούς αυτονομίας: είτε με τηλεχειρισμό από έναν χειριστή είτε με αυτόνομο τρόπο από υπολογιστές επί του σκάφους [6]. Παρά τους περιορισμένους υπολογιστικούς πόρους των UAV ο βασικός βρόχος λειτουργίας τους φαίνεται στο Σχήμα 6, όπου παρουσιάζονται οι υπολογιστικές απαιτήσεις και το μοντέλο δυναμικής για την πτήση τους.

Σε σύγκριση με τα πληρώματα αεροσκάφους, τα UAV χρησιμοποιήθηκαν αρχικά για αποστολές πολύ «μικρής ορατότητας, μολυσματικών ή επικίνδυνων περιοχών» για τους ανθρώπους [7]. Ενώ προέρχονται κυρίως από στρατιωτικές εφαρμογές, η χρήση τους επεκτείνεται ταχέως σε εμπορικές, επιστημονικές, ψυχαγωγικές, γεωργικές και άλλες εφαρμογές όπως αστυνόμευση και επιτήρηση, παράδοση προϊόντων, αεροφωτογραφία, λαθρεμπόριο [8,9]. Οι μη στρατιωτικές μονάδες UAV υπερτερούν σε μεγάλο βαθμό των στρατιωτικών UAV, με πάνω από ένα εκατομμύριο πωλήσεις μέχρι το 2015.



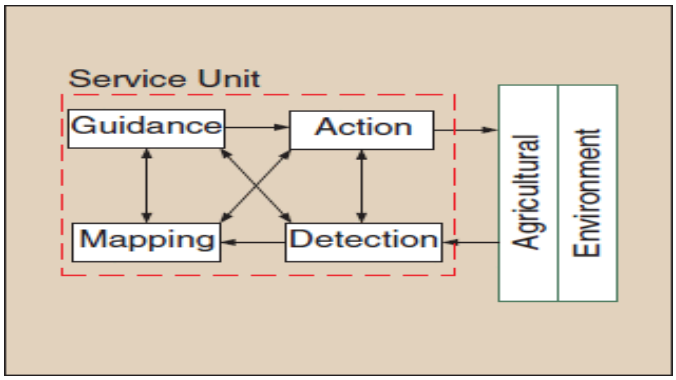
Σχήμα 5: Διάγραμμα λειτουργίας του βασικού βρόχου όλων των εναέριων ρομπότ [10].

Γεωργική Ρομποτική και Αυτοματισμοί (Agricultural Robotics and Automation)

Ένα γεωργικό ρομπότ είναι ένα ρομπότ που χρησιμοποιείται για γεωργικούς σκοπούς. Ο κύριος τομέας εφαρμογής των ρομπότ στη γεωργία σήμερα βρίσκεται στο στάδιο της συγκομιδής. Οι νεοεμφανιζόμενες εφαρμογές ρομπότ ή drones στη γεωργία περιλαμβάνουν τον έλεγχο των ζιζανίων, τη σπορά με ψεκασμό, τη φύτευση σπόρων, τη συγκομιδή, την παρακολούθηση του περιβάλλοντος και την ανάλυση του εδάφους. Σύμφωνα με μία έγκυρη έρευνα αγοράς, η αγορά των γεωργικών ρομπότ αναμένεται να φθάσει τα 11,58 δισεκατομμύρια δολάρια μέχρι το 2025.

Ο στόχος της αγροτικής ρομποτικής είναι κάτι περισσότερο από την εφαρμογή των ρομποτικών τεχνολογιών στη γεωργία. Επί του παρόντος, τα περισσότερα από τα αυτόματα γεωργικά οχήματα που χρησιμοποιούνται είναι επανδρωμένα. Μια αυτόνομη λειτουργία τέτοιων οχημάτων θα επιτρέψει τη συνεχή επίβλεψη του πεδίου, δεδομένου ότι οι πληροφορίες σχετικά με το περιβάλλον μπορούν να αποκτηθούν αυτόνομα και το όχημα μπορεί στη συνέχεια να εκτελέσει προκαθορισμένες ενέργειες.

Οι πιο σημαντικές τρέχουσες ικανότητες των αυτόματων γεωργικών οχημάτων μπορούν να ομαδοποιηθούν σε τέσσερις κατηγορίες: 1) καθοδήγηση (δηλαδή, ο τρόπος που το όχημα κινείται μέσα στο γεωργικό περιβάλλον) 2) ανίχνευση (εξόρυξη βιολογικών χαρακτηριστικών από το περιβάλλον) 3) δράση (εκτέλεση της εργασίας για την οποία σχεδιάστηκε το όχημα, π.χ. συλλογή) 4) χαρτογράφηση (κατασκευή ενός χάρτη του αγροτικού πεδίου με τα πιο σχετικά χαρακτηριστικά του) [10,11] (Σχήμα 7).



Σχήμα 6: Η σχέση των τεσσάρων ποιο βασικών ενσωματώσεων ενός αυτόνομου οχήματος για την εκτέλεση των κύριων ή δευτερευουσών γεωργικών εργασιών

Διαστημική ρομποτική (Space Robotics)

Το εξώτερο διάστημα είναι το ποιο απαιτητικό πεδίο για την εφαρμογή της ρομποτικής τεχνολογίας. Δεδομένου ότι το διάστημα είναι ένα σκληρό περιβάλλον με ακραίες θερμοκρασίες, κενό, ακτινοβολία, βαρύτητα και μεγάλες αποστάσεις, η πρόσβαση του ανθρώπου είναι πολύ δύσκολη και επικίνδυνη και ως εκ τούτου είναι περιορισμένη. Για την υποστήριξη των ανθρώπινων δραστηριοτήτων στο διάστημα όπως η κατασκευή και η διατήρηση διαστημικών μηχανισμών και δομών, οι ρομποτικοί χειριστές παίζουν βασικούς ρόλους στις αποστολές που εκτελούνται σε τροχιά [12]. Επιπλέον, διευρύνοντας τους ορίζοντες της εξερεύνησης πέρα από τις περιοχές πρόσβασης του ανθρώπου, τα ρομπότ που προσγειώνονται και ταξιδεύουν σε πλανητικές επιφάνειες έχουν συμβάλει σημαντικά στην αύξηση της γνώσης μας για το ηλιακό σύστημα.

Τα διαστημικά συντρίμμια έχουν εξελιχθεί σε μία αυξανόμενη ανησυχία τα τελευταία χρόνια [13]. Οι συγκρούσεις σε τροχιακές ταχύτητες μπορεί να είναι πολύ επιζήμιες για τους λειτουργικούς δορυφόρους και μπορούν επίσης να παράγουν ακόμη περισσότερα διαστημικά συντρίμμια. Τα θραύσματα από τέτοιες συγκρούσεις αποτελούν πλέον σοβαρό κίνδυνο για τους αστροναύτες στο ISS (διεθνής διαστημικός σταθμός) και ως εκ τούτου το ISS είναι θωρακισμένο για τον μετριασμό των ζημιών από τις συγκρούσεις μικρών σε διαστάσεις θραυσμάτων (λιγότερο από 1 cm), ενώ περιστασιακά κάνει ελιγμούς αποφυγής συγκρούσεων για μεγαλύτερα σώματα με μεγέθη μεγαλύτερα από 10 cm). Πρόσφατα, σημειώθηκε μεγάλη σύγκρουση μεταξύ ενός απενεργοποιημένου δορυφόρου Kosmos 2251 και ενός επιχειρησιακού δορυφόρου Iridium 33 στις 10 Φεβρουαρίου 2009. Η σχετική ταχύτητα πρόσκρουσης ήταν πάνω από 10 km / s και οι δύο δορυφόροι καταστράφηκαν δημιουργώντας και διασκορπίζοντας ένα σημαντικό ποσό νέων συντριμμιών. Μετά από αυτό το γεγονός, το διαστημικό λεωφορείο και το ISS είχαν αυξημένο αριθμό προειδοποιήσεων και ελιγμών αποφυγής.

Για να μετριαστεί η δημιουργία πρόσθετων θραυσμάτων, έχουν προταθεί διάφορα μέτρα. Αλλά η ποιο ελπιδοφόρα ιδέα είναι η ρομποτική αναδιάταξη τροχιάς (και όχι ο εκτροχιασμός). Η ρομποτική αναδιάταξη τροχιάς μπορεί να λειτουργήσει για να μειώσει την τροχιά των δορυφόρων σε χαμηλή τροχιά Γης (LEO) η αλλιώς να τους ωθήσει σε μια λεγόμενη ζώνη νεκροταφείου δορυφόρων που βρίσκεται σε γεωφυσική γήινη τροχιά (GEO). Μπορεί επίσης να λειτουργήσει για να αλλάξει το τροχιακό επίπεδο εάν υπάρχει αρκετή ικανότητα πρόωσης στο ρομπότ. Αυτή είναι μία από τις διαστημικές εφαρμογές συστημάτων αυτοματισμού ρομποτικής και μηχανικής νοημοσύνης (Automation Robotics and Machine Intelligence Systems A.R.A.M.I.S.) που εξετάζεται από τη Nasa (Σχήμα 8).

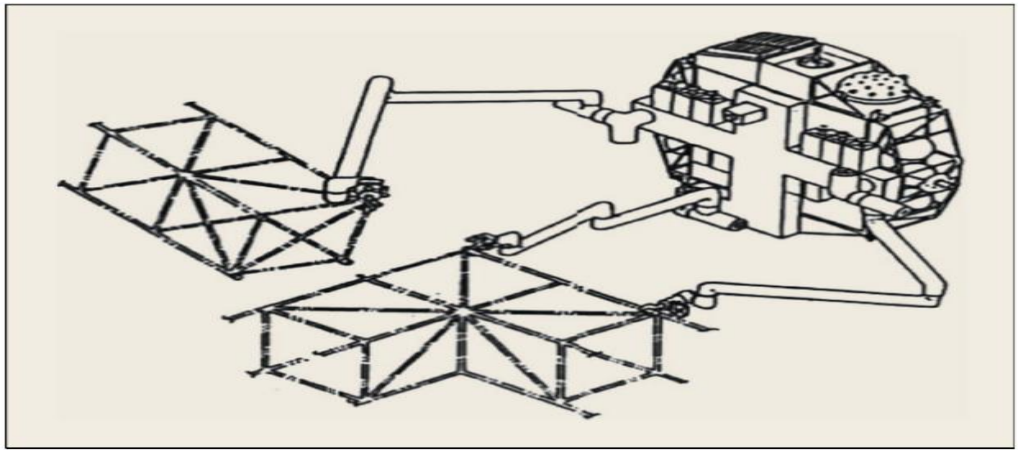
Αν και η σημασία αυτών των αποστολών επαναφοράς, επαναρρύθμισης ή διάσωσης έχει κατανοηθεί καλά για περισσότερο από μια δεκαετία, οι δυνατότητες επίδειξης τεχνολογίας εν πτήση είναι περιορισμένες. Η αποστολή του ETS-VII (Orihime και Hikoboshi) που διεξήχθη από την Εθνική Υπηρεσία Ανάπτυξης του Διαστήματος της Ιαπωνίας (NASDA) για την αποστολή 1997-1999 και του Orbital Express που πραγματοποιήθηκε από την DARPA και τη Boeing το 2007 είναι δύο εξαιρετικά παραδείγματα.

Το ETS-VII (Σχήμα 9) είναι ένα μη επανδρωμένο διαστημικό σκάφος που είναι εξοπλισμένο με ρομποτικό βραχίονα χειριστή μήκους 2 μέτρων και έξι βαθμών ελευθερίας (6-DoF). Αναπτύχθηκε και δρομολογήθηκε από την NASDA (επί του παρόντος JAXA) το Νοέμβριο του 1997. Στόχος της αποστολής του ETS-VII ήταν να δοκιμάσει την τεχνολογία ρομποτικών βραχιόνων σε ελεύθερη πτήση και να αποδείξει τη χρησιμότητά τους σε μη

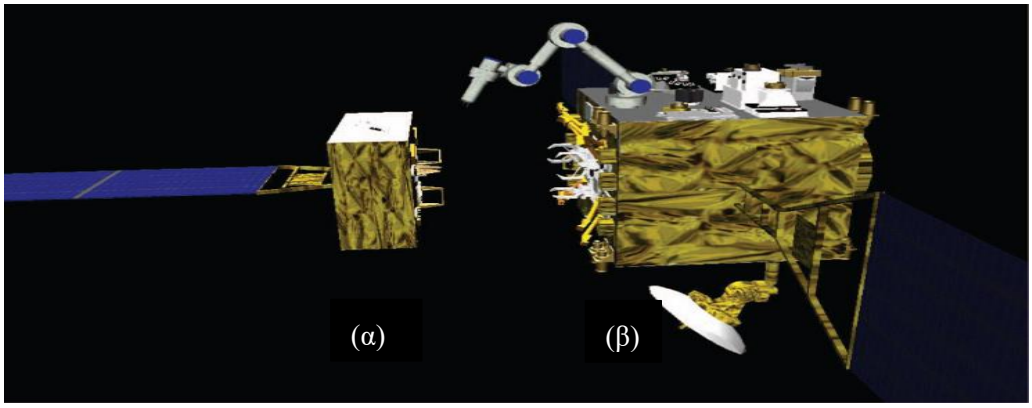
επανδρωμένες τροχιακές αποστολές. Η αποστολή του αποτελείται από δύο φάσεις: πειράματα αυτόνομης σύνδεσης / κουμπώματος (RVD) και ρομποτικά πειράματα (RBT).

Για τα πειράματα RVD, το ETS-VII χωρίστηκε σε δύο κομμάτια δορυφορικά σε τροχιά. Το μεγάλο κομμάτι ονομάστηκε Hikoboshi (πρίγκιπας σε ιαπωνικό παραδοσιακό παραμύθι) και λειτούργησε ως κυνηγός. Το μικρό κομμάτι ονομάστηκε Orihime (πριγκίπισσα στην ιαπωνική ιστορία) και έδρασε ως στόχος. Και τα δύο σενάρια προσέγγισης και κουμπώματος R-bar και V-bar διεξήχθησαν με επιτυχία χρησιμοποιώντας το παγκόσμιο σύστημα εντοπισμού θέσης (GPS), το ραντάρ λέιζερ σύνδεσης (RVR), τον οπτικό αισθητήρα εγγύτητας (PXS) και την αυτονομία του σκάφους.

Οι ρομποτικές δοκιμές RBT με τη χρήση βραχίονα χειριστή 6-DoF επί του σκάφους διεξήχθησαν κατά τη διάρκεια μιας διετούς περιόδου από πολλές οργανώσεις, συμπεριλαμβανομένων των NASDA, του Εθνικού Αεροδιαστημικού Εργαστηρίου (NAL) του Ηλεκτροτεχνικού εργαστηρίου (ETL) Ερευνητικού Εργαστηρίου Ερευνών (CRL) της Ευρωπαϊκής Διαστημικής Υπηρεσίας (ESA) του Γερμανικού Αεροδιαστημικού Κέντρου (DLR) του Ιδρύματος Τεχνολογίας της Ιαπωνίας και του Πανεπιστημίου του Κιότο.



Σχήμα 7: Ένα ρομπότ ελεύθερης πτήσης (telerobotic servicer) που συζητήθηκε στα αρχεία αυτοματισμού, ρομποτικής και συστημάτων μηχανικής ευφυίας (ARAMIS 1983) [12].



Σχήμα 8: Ιαπωνικός Δοκιμαστικός Δορυφόρος (ETS-VII): (α) Orihime και (β) Hikoboshi.

Ιατρική ρομποτική (Medical Robotics)

Ρομποτική σε εφαρμογές αποκατάστασης: Τα ρομποτικά συστήματα προσθετικής και ορθωτικής αναπτύσσονται για να αντικαταστήσουν τα χαμένα μέρη του σώματος, όπως τα χέρια ή τα πόδια και να παρέχουν βοήθεια σε τραυματισμένα άκρα. Τα προσθετικά είναι ηλεκτρομηχανικές συσκευές που χρησιμοποιούνται συχνά για να παρέχουν δυνατότητες κινητικότητας ή χειρισμού όταν χαθεί ένα άκρο. Ένα από τα εμπορικά διαθέσιμα προσθετικά χέρια είναι ο βραχίονας Utah (Motion Control, Inc., Salt Lake City, UT), ο οποίος είναι μια πρόθεση ελεγχόμενη από Η/Υ που αναπτύχθηκε από τον Jacobsen στο Πανεπιστήμιο της Γιούτα. Αυτός ο ρομποτικός βραχίονας ελέγχεται χρησιμοποιώντας ανατροφοδότηση ηλεκτρομυογραφίας (EMG) από αισθητήρες που μετρούν την απόκριση ενός μυς σε νευρική διέγερση [14]. Η συσκευή χρησιμοποιεί μετατροπείς για τον έλεγχο της ταχύτητας κίνησης, η οποία παρέχει τελικά την κατάλληλη ακρίβεια στις χαμηλές ταχύτητες και τον έλεγχο απόκρισης σε υψηλότερες ταχύτητες. Άλλες πτυχές αυτού του τεχνητού χεριού είναι η ακριβής τοποθέτηση του στον αγκώνα, η μίμηση της ικανότητας του φυσικού βραχίονα να πέφτει ελεύθερα με χαμηλή αντίσταση όταν οι μύες χαλαρώνουν και ιδιομορφία που είναι εγγενής στο σχεδιασμό [15]. Προσθετικά ρομποτικά συστήματα μπορούν επίσης να χρησιμοποιηθούν για την αντικατάσταση των κάτω άκρων. Το MIT LegLab έχει κατασκευάσει μια σειρά από ρομπότ με πόδια για πειράματα ενεργητικής ισορροπίας σε δυναμική κίνηση με πόδια. Αυτά τα ρομποτικά προσθετικά χρησιμοποιούνται για να επιτρέψουν στα άτομα με ακρωτηριασμούς πάνω από το γόνατο να περπατούν και να ανεβαίνουν σκάλες πιο φυσικά [16].

Ένα ορθωτικό σύστημα είναι ένας μηχανισμός που χρησιμοποιείται για να υποβοηθήσει ή να υποστηρίξει έναν ασθενή με αδύναμη άρθρωση, μυ ή άκρο. Πολλά ορθωτικά συστήματα χρησιμοποιούν τη ρομποτική τεχνολογία και συχνά παίρνουν τη μορφή ενός εξωσκελετού.

Οι εξωσκελετοί έχουν αρθρώσεις ή συνδέσμους που αντιστοιχούν σε εκείνους των ανθρώπων και ενεργοποιητές που βοηθούν τους ασθενείς να κινούν τα άκρα τους ή να ανυψώνουν εξωτερικά φορτία. Το Wrist-Hand Orthosis (WHO) είναι ένα από αυτά τα ρομποτικά ορθωτικά συστήματα. Χρησιμοποιεί ενεργοποιητές κράματος μνήμης σχήματος για να παρέχει μια λειτουργία κρατήματος στους τετραπληγικούς ασθενείς [17].

Θεραπευτικά ρομποτικά συστήματα: Ένα από τα πιο γνωστά θεραπευτικά ρομποτικά συστήματα είναι το MIT-Manus. Το πρωτότυπο αυτού του συστήματος δύο βαθμών ελευθερίας (δύο DOF) αναπτύχθηκε για τα προγράμματα αποκατάστασης ώμων και αγκώνα στο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης (MIT) και δοκιμάστηκε στο Burke Rehabilitation Hospital, White Plains, NY, ΗΠΑ. Οι συνεισφέροντες υποστήριξαν ότι μια τέτοια θεραπεία που υποστηρίζεται από ρομπότ δεν έχει παρενέργειες και μπορεί να βοηθήσει στις δυσλειτουργίες ελέγχου των άκρων [18]. Το 2007, οι Rosati et al. [19] πρότειναν ένα ρομπότ που βασίζεται σε κίνηση συρμάτων που ονομάζεται NeReBot (NeuroRehabilitation roBot) για να παρέχει πολλαπλούς βαθμούς ελευθερίας σε τέτοια ρομποτικά συστήματα αποκατάστασης [20].

Ρομποτική που εφαρμόζεται κατά τη διάρκεια χειρουργικών επεμβάσεων**:** Η χειρουργική και η ρομποτική έχουν συνδυαστεί για να δημιουργήσουν ένα νέο είδος επαυξημένου χειρουργείου [21]. Στις ρομποτικές χειρουργικές επεμβάσεις με τη βοήθεια ηλεκτρονικών υπολογιστών γίνεται ενίσχυση της ικανότητας του χειρουργού να πραγματοποιεί διάφορες χειρουργικές επεμβάσεις [22].

Τα χειρουργικά ρομπότ μπορούν να ταξινομηθούν με πολλούς τρόπους. Μερικές από τις κοινές ταξινομήσεις των χειρουργικών ρομποτικών συστημάτων είναι οι εξής: ανάλογα με σχεδιασμό στον χειριστή (π.χ. κινηματικά, ενεργοποιητές, βαθμοί ελευθερίας DOF), κατά επίπεδο αυτονομίας (όπως προ-προγραμματισμένο, καθοδηγούμενο με εικόνα, τηλεχειριζόμενο, συνεργατικό), ανάλογα με την επιθυμητή ανατομία / τεχνική (π.χ. καρδιακή, ενδοαγγειακή, δια δερμική, λαπαροσκοπική, μικροχειρουργική), από το προβλεπόμενο περιβάλλον λειτουργίας (π.χ. χειρουργείο, σαρωτής απεικόνισης, δάπεδο νοσοκομείου) και από το πλαίσιο του ρόλου τους σε συστήματα χειρουργικών επεμβάσεων με ενσωματωμένους υπολογιστές (π.χ. σχεδιασμός χειρουργείου, χειρουργικοί βοηθοί)[22].

Η πρώτη εφαρμογή της ρομποτικής στη χειρουργική ήταν στον τομέα της στερεοστατικής χειρουργικής εγκεφάλου χρησιμοποιώντας ένα βιομηχανικό ρομπότ που ονομάζεται Unimate Puma 560 (Puma, Advanced Research & Robotics, Οξφόρδη, CT) [23]. Άλλα βιομηχανικά ρομπότ που χρησιμοποιήθηκαν αρχικά στη νευροχειρουργική είναι το NeuroMate (Integrated Surgical Systems, Sacramento, CA) και η Minerva (Πανεπιστήμιο της Λωζάνης, Λωζάνη, Ελβετία) [24]. Το NeuroMate ήταν το πρώτο εμπορικά διαθέσιμο ρομπότ νευροχειρουργικής που εγκρίθηκε από την Υπηρεσία Τροφίμων και Φαρμάκων της Αμερικής (FDA). Χρησιμοποίησε ένα προεγχειρητικό σύστημα καθοδήγησης απεικόνισης και ένα βραχίονα για να εκτελέσει χειρουργικές επεμβάσεις. Η Minerva, που αναπτύχθηκε στο Πανεπιστήμιο της Λωζάνης, χρησιμοποίησε έναν πραγματικό 3D σαρωτή CT, ο οποίος παρείχε στον χειρουργό ακριβή θέση στην περιοχή της χειρουργικής επέμβασης, παρόλο που η κεφαλή ή ο εγκέφαλος είχαν μετατοπιστεί [23, 24]. Ένα νέο νευρορομποτικό σύστημα που αναπτύχθηκε πρόσφατα είναι το 6-DoF PathFinder που διατίθεται στην ευρωπαϊκή αγορά (Prosurgics, ΗΝΩΜΕΝΟ ΒΑΣΙΛΕΙΟ; πρώην Armstrong Healthcare). Χρησιμοποιώντας τόσο το ενσωματωμένο σύστημα κάμερας όσο και το λογισμικό προγραμματισμού, ανιχνεύει αυτόματα τους χαρακτηριστικούς δείκτες και μπορεί να διατάξει και να ευθυγραμμίσει τη θέση του οργάνου σύμφωνα με την προγραμματισμένη τροχιά με ακρίβεια 1 mm [25]. Το Cyberknife (Accuray, Inc., Sunnyvale, CA) είναι το πιο πρόσφατο εμπορικά διαθέσιμο νευροχειρουργικό σύστημα που χρησιμοποιείται ως ελάχιστα επεμβατική εναλλακτική λύση. Αποτελείται από ένα 6-DOF (6 βαθμών ελευθερίας) ρομποτικό χειριστή ελεγχόμενο από υπολογιστή και ειδικό σύστημα οπτικής καθοδήγησης ακτίνων Χ.

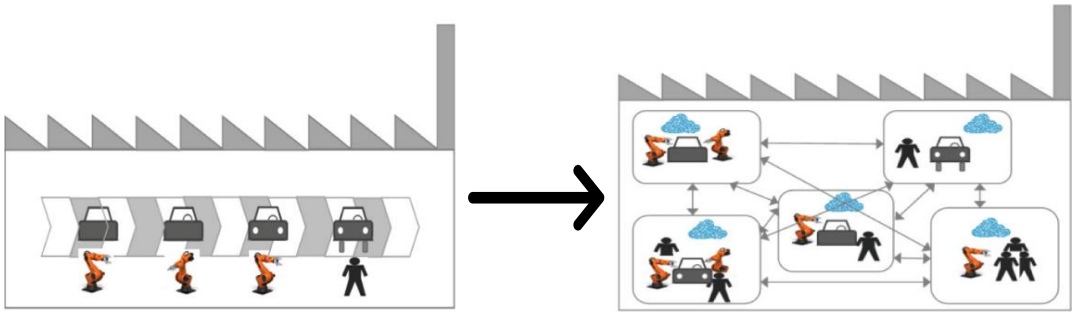
Βιομηχανικά ρομπότ

Η βιομηχανία βρίσκεται επί του παρόντος στο στάδιο της τέταρτης βιομηχανικής επανάστασης. Το Παγκόσμιο Οικονομικό Φόρουμ WEF (World Economic Forum) που πραγματοποιήθηκε στο Νταβός το 2016 χαρακτήρισε τις αλλαγές που πραγματοποιήθηκαν στην παγκόσμια βιομηχανική και ψηφιακή σκηνή ως την τέταρτη βιομηχανική επανάσταση. Ο όρος «Industry 4.0» εμφανίστηκε για πρώτη φορά το 2011 σε μία έκθεση του στο Ανόβερο της Γερμανίας. Το Industry 4.0 παρουσιάζει το όραμα της προηγμένης βιομηχανικής παραγωγής που ήδη εφαρμόζεται εν μέρει ως νέα τεχνολογία με τον αυτοματισμό των διαδικασιών κατασκευής, της ανταλλαγής δεδομένων και της επεξεργασίας δεδομένων. Μέσα στην τέταρτη βιομηχανική επανάσταση σχηματίζεται μάλιστα μία αλυσίδα που βασίζεται κυρίως στα κυβερνοφυσικά συστήματα τα οποία είναι επίσης το δεύτερο όνομα για το Διαδίκτυο των πραγμάτων και τις σχετικές υπηρεσίες που υλοποιούνται συχνότερα στο cloud (Cloud Computing) [28]

Η κλασική διαδικασία κατασκευής χρησιμοποιούσε τη γραμμή παραγωγής (Σχήμα 10α) και βιομηχανικά ρομπότ πρώτης γενιάς στη διαδικασία αυτοματισμού, τα οποία

διαχωρίστηκαν από τους εργάτες για να αποτρέψουν τραυματισμούς. Προκειμένου να αλλάξει το προϊόν, έπρεπε να επαναπρογραμματιστούν, κάτι που απαιτούσε πολύ χρόνο για την αναδιοργάνωση, δηλαδή ήταν αυστηρός αυτοματισμός. Το Industry4.0 άλλαξε τη διαδικασία παραγωγής σε κλειστό βρόχο όπου όλες οι συσκευές στη διαδικασία παραγωγής είναι διασυνδεδεμένες και έχουν επικοινωνία με M2M (machine to machine επικοινωνία περιλαμβάνει σειριακή σύνδεση μεταξύ δύο μηχανών και σύνδεση ρεύματος PLC ή ασύρματες επικοινωνίες όταν αναφερόμαστε στο IoT), αλληλοεπιδρούν και ανταλλάσσουν δεδομένα με άλλα κυβερνο-φυσικά συστήματα (Σχήμα 10β).

Η ραγδαία αύξηση του αριθμού των διπλωμάτων ευρεσιτεχνίας από το "Industry 4.0", και τις σχετικές τεχνολογίες: υπολογιστικό νέφος, ρομποτική & αυτοματοποίηση, έξυπνοι αισθητήρες, τρισδιάστατοι εκτυπωτές και αναγνώριση ραδιοσυχνοτήτων – RFID και την εφαρμογή τους στη διαδικασία κατασκευής, αύξησε την ενσωμάτωσή τους στο "Industry 4.0" [30,31].



(α) (β)

Σχήμα 10: α) Κλασική διαδικασία παραγωγής, β) Διαδικασία παραγωγής στη 4η Βιομηχανική επανάσταση

Η πρόοδος των βιομηχανικών ρομπότ είχε ως επακόλουθο τη διάδοση των ρομπότ σε διάφορες βιομηχανίες, από την αυτοκινητοβιομηχανία έως την βιομηχανία ιατρικού εξοπλισμού. Τα ρομπότ προσφέρουν ταχύτητα και ακρίβεια που δεν μπορεί να επιτευχθεί με ανθρώπινη εργασία. Τα ρομπότ μπορούν επίσης να μειώσουν το λειτουργικό κόστος, να μειώσουν τα απορρίμματα και να είναι ευέλικτα σε μελλοντικές αλλαγές. Οι δυνατότητες της ρομποτικής αυξήθηκαν με το χρόνο, ενώ το κόστος συνέχισε να μειώνεται. Οι μεγαλύτεροι κατασκευαστές ρομπότ αναβαθμίζουν συνεχώς τα ρομπότ τους με αυξημένη ικανότητα ωφέλιμου φορτίου, μεγαλύτερη ακρίβεια, αυξημένη εμβέλεια, βελτιωμένη ταχύτητα και επιτάχυνση, ταχύτερη επικοινωνία με τον εξωτερικό εξοπλισμό, καλύτερα χαρακτηριστικά ασφαλείας και χαμηλότερο λειτουργικό κόστος.

Τα πολλά οφέλη των ρομπότ φαίνεται να είναι πιο αισθητά στην παραγωγικότητα, την ασφάλεια και στην εξοικονόμηση χρόνου και χρήματος.

Σχετικά με τη παραγωγικότητα τα ρομπότ παράγουν πιο ακριβή και υψηλής ποιότητας εργασία. Τα ρομπότ σπάνια κάνουν λάθη και είναι πιο ακριβή από τους εργαζόμενους. Μπορούν να παράγουν μεγαλύτερη ποσότητα σε σύντομο χρονικό διάστημα. Μπορούν να εργαστούν με σταθερή ταχύτητα χωρίς διαλείμματα, ημέρες αργίας, ή ώρες διακοπών. Επίσης μπορούν να εκτελέσουν εφαρμογές με μεγαλύτερη επαναληψιμότητα από τους ανθρώπους.

Αναφορικά με την ασφάλεια τα ρομπότ αντικαθιστούν τους εργαζόμενους στην εκτέλεση επικίνδυνων εργασιών, είναι ικανά να ανυψώνουν βαριά φορτία χωρίς να τραυματίζονται ή

να κουράζονται. Τα ρομπότ αυξάνουν την ασφάλεια των εργαζομένων, αποτρέποντας ατυχήματα, καθώς οι άνθρωποι δεν εκτελούν τις επικίνδυνες εργασίες.

Όσον αφορά την εξοικονόμηση, τα ρομπότ εξοικονομούν χρόνο με την ικανότητα να παράγουν μεγαλύτερο πλήθος προϊόντων, μειώνουν το υλικό που χρησιμοποιήθηκε λόγω της ακρίβειας τους και εξοικονομούν μακροπρόθεσμα χρήματα στις εταιρείες με γρήγορες ROIs (επιστροφές επένδυσης), λιγότερους τραυματισμούς εργαζομένων (μειώνοντας ή εξαλείφοντας τις αποζημιώσεις του εργατικού δυναμικού) και με τη χρήση λιγότερων υλικών.

Πολλοί άνθρωποι έχουν την αντίληψη ότι τα ρομπότ αφαίρεσαν δουλειές από τους εργαζόμενους, αλλά αυτό δεν είναι απαραίτητα αλήθεια. Τα ρομπότ έχουν δημιουργήσει νέες δουλειές για όσους ήταν κάποτε σε γραμμές παραγωγής ελεγχόμενες από προγραμματισμό. Έχουν απορροφήσει υπαλλήλους από επαναλαμβανόμενες, μονότονες δουλειές και τους έβαλαν σε καλύτερες, και πιο απαιτητικές. Σήμερα τα ρομπότ είναι φιλικά προς το χρήστη, έξυπνα και οικονομικά προσιτά [31].

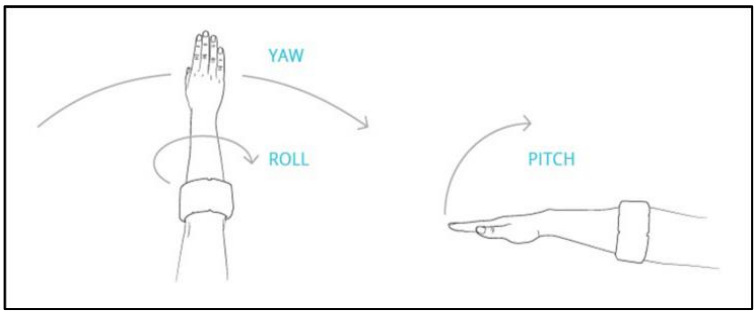
Το 2019 η Νότια Κορέα είχε εγκατεστημένα 855 βιομηχανικά ρομπότ ανά 10.000 υπαλλήλους, κάτι που οφείλεται κυρίως στην συνεχιζόμενη εγκατάσταση ρομπότ στις βιομηχανίες ηλεκτρικών και ηλεκτρονικών συσκευών. Η Γερμανία και η Ιαπωνία φημίζονται για τις αυτοκινητοβιομηχανίες τους και έχουν επίπεδα πυκνότητας περίπου 350 βιομηχανικά ρομπότ ανά 10.000 εργαζόμενους. Είναι ενδιαφέρον ότι η Ιαπωνία είναι ένας από τους κύριους παραγωγούς των βιομηχανικών ρομπότ, αντιπροσωπεύοντας πάνω από το ήμισυ της παγκόσμιας παραγωγής [32].

ΚΕΦΑΛΑΙΟ 2

Γεωμετρίες ρομποτικών βραχιόνων – Βασικές Διαμορφώσεις

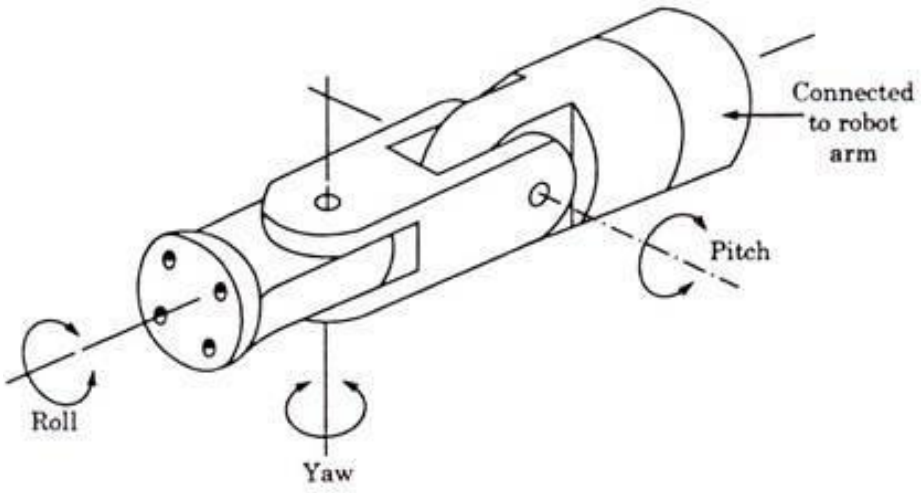
2.1 Βασική δομή και λειτουργία του ρομποτικού βραχίονα

Η μηχανική δομή του ρομπότ ή μηχανικός βραχίονας αποτελείται από μία αλυσίδα στερεών σωμάτων (των συνδέσμων) που συνδέονται μεταξύ τους με τις αρθρώσεις. Οι αρθρώσεις διακρίνονται σε δύο βασικούς τύπους: αυτές που επιτρέπουν σχετική περιστροφή των γειτονικών συνδέσμων γύρω από τον άξονα της άρθρωσης (περιστροφικές αρθρώσεις) και αυτές που επιτρέπουν σχετική μεταφορική κίνηση κατά μήκος του άξονα της άρθρωσης γραμμικά (πρισματικές αρθρώσεις). Οι δύο τύποι αρθρώσεων συνδυάζονται για να μας δώσουν τη συγκεκριμένη δομή ενός βραχίονα. Συνήθως οι τρείς πρώτες αρθρώσεις του βραχίονα προσδίδουν κινητικότητα στο μηχανισμό του ρομπότ και ο συνδυασμός του τύπου τους χαρακτηρίζει τη γεωμετρία του βραχίονα. Οι επόμενες των τριών πρώτων αρθρώσεις του βραχίονα έχουν συνήθως στόχο να προσδώσουν επιδεξιότητα στο μηχανισμό του ρομπότ και αποτελούν τον καρπό του. Στον καρπό προσαρτάται το άκρο του βραχίονα, με το οποίο εκτελείται η απαιτούμενη εργασία από το ρομπότ. Ο καρπός είναι υπεύθυνος για τον προσανατολισμό του άκρου του βραχίονα και γι' αυτό είναι συνήθως συμπαγής. Ο καρπός στην ανθρωπομορφική μορφή του (Σχήμα 11) αποτελείται από τρεις περιστροφικές αρθρώσεις που επιτρέπουν τη στροφή (roll) την κλίση (pitch) και την εκτροπή (yaw) όπως και στον άνθρωπο.



Σχήμα 9: Κινήσεις ανθρώπινου καρπού

Η μεγαλύτερη επιδεξιότητα του καρπού επιτυγχάνεται όταν οι άξονες των τριών περιστροφικών αρθρώσεων τέμνονται σε ένα σημείο τότε αποκαλείται σφαιρικός καρπός (Σχήμα 12).



Σχήμα 10: Κινήσεις σφαιρικού καρπού

Σε κάθε άρθρωση αντιστοιχεί ένας ενεργοποιητής που προκαλεί την κίνηση της άρθρωσης. Ο ενεργοποιητής μπορεί να είναι ένας κινητήρας ηλεκτρικός, υδραυλικός ή πνευματικός. Για τον έλεγχο των κινητήρων χρειάζονται μετρήσεις για τη θέση και ίσως και την ταχύτητα του. Με τον όρο θέση εννοούμε την απόσταση από κάποιο αυθαίρετα ορισμένο σημείο αναφοράς. Στην περίπτωση της περιστροφής η θέση αφορά τη γωνιακή θέση μετρούμενη σε rad. Διάφοροι τύποι αισθητηρίων (π.χ. ποτενσιόμετρο, ταχύμετρο) μπορούν να δώσουν μετρήσεις για τη θέση και την ταχύτητα της άρθρωσης. Το άκρο του βραχίονα εξαρτάται από την εφαρμογή στη οποία θα χρησιμοποιηθεί ο βραχίονας και μπορεί να είναι ένα εργαλείο ή μια αρπάγη. Υπάρχουν διάφοροι τύποι αρπάγης: αρπάγες για ειδικές εφαρμογές όπως μαγνητική αρπάγη ή αρπάγη κενού και αρπάγες για γενικές εφαρμογές που διαθέτουν μηχανικά δάχτυλα με τα οποία συλλαμβάνουν και χειρίζονται αντικείμενα. Ο απλούστερος τύπος γενικής αρπάγης έχει δύο δάχτυλα και δύο καταστάσεις (ανοιχτή και κλειστή) ενώ στους εξελιγμένους τύπους αρπάγης τα μηχανικά δάχτυλα είναι εφοδιασμένα με αισθητήρια (Σχήμα 13). Όσο μεγαλύτερο είναι το εύρος των εφαρμογών της αρπάγης τόσο μεγαλύτερο το κόστος της.

Σχήμα 11: Μηχανικά δάχτυλα με ενσωματωμένους αισθητήρες πίεσης

2.1.1 Σύστημα κίνησης και είδη μηχανισμών κίνησης

Υπάρχουν μηχανισμοί παθητικής και ενεργητικής κίνησης, καθώς και ευφυή σχέδια και για τους δύο. Σχετικά με τους μηχανισμούς ενεργητικής κίνησης κάποιοι από τους τρόπους για να κινήσουμε ένα ρομποτικό όργανο δράσης είναι οι εξής:

Ηλεκτροκινητήρες: Βασίζονται στο ηλεκτρικό ρεύμα και είναι οι ποιο διαδεδομένοι, οι ποιο φθηνοί και οι πιο εύχρηστοι ρομποτικοί μηχανισμοί κίνησης.

Υδραυλικοί μηχανισμοί: Οι συγκεκριμένοι μηχανισμοί κίνησης βασίζονται στις μεταβολές της πίεσης των υγρών. Οι υδραυλικοί μηχανισμοί είναι αρκετά ισχυροί και ακριβείς, αλλά είναι μεγάλοι σε μέγεθος, πιθανώς επικίνδυνοι, και πρέπει να είναι καλά συσκευασμένοι και απόλυτα στεγανοί.

Πνευματικοί μηχανισμοί: Οι πνευματικοί μηχανισμοί κίνησης βασίζονται στις μεταβολές της πίεσης του αέρα. Όπως και οι υδραυλικοί μηχανισμοί, είναι συνήθως μεγάλοι σε μέγεθος, πιθανώς επικίνδυνοι, και πρέπει να είναι απόλυτα στεγανοί.

Υλικά ευαίσθητα σε χημικά: Όπως υποδηλώνει και το όνομά τους, αυτά τα υλικά αντιδρούν σε συγκεκριμένες χημικές ουσίες. Ένα καλό παράδειγμα είναι ένα συγκεκριμένο είδος νήματος το οποίο συστέλλεται (μαζεύει) όταν τοποθετείται σε όξινο διάλυμα και επιμηκύνεται όταν τοποθετείται σε βασικό (αλκαλικό) διάλυμα. Τα υλικά αυτού του είδους μπορούν να χρησιμοποιηθούν ως μηχανισμοί γραμμικής κίνησης (linear actuators), οι οποίοι αυξάνουν ή μειώνουν το μήκος τους. Η γραμμική κίνηση είναι πολύ διαφορετική από την περιστροφική κίνηση των κινητήρων.

Υλικά ευαίσθητα στη θερμοκρασία: Αυτά τα υλικά αντιδρούν στις μεταβολές της θερμοκρασίας.

Πιεζοηλεκτρικά υλικά: Αυτά τα υλικά (συνήθως είδη κρυστάλλων) παράγουν ηλεκτρική τάση όταν δέχονται πίεση.

Η παραπάνω λίστα δεν είναι πλήρης, καθώς οι μηχανικοί αναπτύσσουν συνεχώς νέα είδη μηχανισμών κίνησης. Εξετάζοντας όμως τους απλούστερους και δημοφιλέστερους ρομποτικούς μηχανισμούς κίνησης θα καταλήγαμε στους κινητήρες.

Κινητήρες

Οι κινητήρες είναι οι πιο διαδεδομένοι μηχανισμοί κίνησης στη ρομποτική. Είναι ιδανικοί για να κινούν τους τροχούς ένα ιδιαίτερα δημοφιλές όργανο δράσης στη ρομποτική και γενικότερα στο έδαφος, καθώς τους παρέχουν περιστροφική κίνηση. Οι κινητήρες είναι επίσης πολύ χρήσιμοι για να κινούν και άλλα είδη οργάνων δράσης. Σε σύγκριση με κάθε άλλο μηχανισμό κίνησης, οι κινητήρες συνεχούς ρεύματος (DC) είναι απλοί, χαμηλού κόστους , εύχρηστοι και καθόλου δυσεύρετοι. Διατίθενται σε μεγάλη ποικιλία μεγεθών και συσκευασιών, ώστε να ανταποκρίνονται στις ανάγκες διαφορετικών ρομπότ και εργασιών. Αυτό είναι πολύ σημαντικό καθώς ένας καλός σχεδιαστής ρομπότ πρέπει να επιλέγει για κάθε ρομπότ τα κατάλληλα συστατικά (συμπεριλαμβανομένων των μηχανισμών κίνησης) σύμφωνα με την εργασία που πρέπει να εκτελέσει το ρομπότ. Γνωρίζουμε από τη φυσική ότι οι κινητήρες DC μετατρέπουν την ηλεκτρική ενέργεια σε μηχανική. Χρησιμοποιούν μαγνήτες, πηνία και ρεύμα για να δημιουργήσουν μαγνητικά πεδία, των οποίων η αλληλεπίδραση περιστρέφει τον άξονα του κινητήρα. Έτσι, η ηλεκτρομαγνητική ενέργεια μετατρέπεται σε κινητική, παράγοντας κίνηση. Το Σχήμα 14 παρουσιάζει έναν τυπικό κινητήρα DC. Αν και τα

συστατικά του κινητήρα είναι κρυμμένα μέσα στο περίβλημά του, μπορούμε να διακρίνουμε στα αριστερά τα καλώδια ρεύματος.

Σχήμα 12: Τυπικός κινητήρας σταθερής τάσης DC με ενσωματωμένο οδηγό αριστερά και δεξιά ακροδέκτες και διάγραμμα συνδέσεως με μικροεπεξεργαστή Arduino και εξωτερική πηγή ενέργειας.

Η εκκίνηση ενός κινητήρα απαιτεί την τροφοδότηση με ηλεκτρική ενέργεια στο κατάλληλο εύρος τάσης. Αν η τάση είναι μικρότερη και πάλι θα λειτουργήσει, αλλά με μειωμένη ισχύ. Αν η τάση είναι υπερβολικά υψηλή, η ισχύς του κινητήρα θα αυξηθεί, αλλά ο κινητήρας θα χαλάσει πιο γρήγορα λόγω της φθοράς. Είναι όπως όταν οδηγούμε ένα αυτοκίνητο σε χαμηλές και υψηλές στροφές. Αν το οδηγήσουμε σε χαμηλές στροφές αυτό πηγαίνει αργά αλλά σε υψηλές στροφές ο κινητήρας του αυτοκινήτου αυξάνει ταχύτητα, όσο περισσότερο οδηγούμε σε υψηλές στροφές τόσο πιο γρήγορα θα πέσει η απόδοση του κινητήρα.

Όταν ένας κινητήρας DC τροφοδοτείται με σταθερή τάση στο κατάλληλο εύρος, η ποσότητα ρεύματος που καταναλώνει είναι ανάλογη του έργου που παράγει. Το έργο, όπως ορίζεται στη φυσική, είναι το γινόμενο της δύναμης επί τη μετατόπιση, επομένως όταν η κίνηση ενός ρομπότ εμποδίζεται από έναν τοίχο, οι κινητήρες που κινούν τους τροχούς καταναλώνουν περισσότερο ρεύμα και εξαντλούν γρηγορότερα τις μπαταρίες απ’ ό,τι όταν το ρομπότ κινείται ελεύθερα, χωρίς εμπόδια στο δρόμο του. Ο λόγος για τον οποίο συμβαίνει αυτό είναι η αντίσταση που προβάλλει ο μη κινούμενος τοίχος, ο οποίος δεν επιτρέπει στο ρομπότ να μετακινηθεί με αποτέλεσμα να χρειάζεται να ασκήσει μεγαλύτερη δύναμη για την ίδια ποσότητα παραγόμενου έργου. Αν η αντίσταση είναι ιδιαίτερα μεγάλη (ο τοίχος δεν πρόκειται να φύγει από τη θέση του, όσο και να τον σπρώχνει το ρομπότ), ο κινητήρας χρησιμοποιεί τη μέγιστη διαθέσιμη ισχύ και μετά, αφού έχει εξαντλήσει κάθε περιθώριο, κολλάει (stall).

Όσο περισσότερο ρεύμα καταναλώνει ο κινητήρας, τόσο μεγαλύτερη ροπή (περιστροφική δύναμη) αναπτύσσεται στον άξονά του. Αυτό είναι σημαντικό, καθώς η ισχύς που παράγει ένας κινητήρας είναι ανάλογη της ροπής του και της ταχύτητας περιστροφής του άξονα. Ποιο συγκεκριμένα, η ισχύς ενός κινητήρα είναι ανάλογη προς το γινόμενο αυτών των δύο μεγεθών, δηλαδή της ροπής και της ταχύτητας περιστροφής. Όταν ο κινητήρας περιστρέφεται ελεύθερα, χωρίς ο άξονάς του να συνδέεται με κάποιο άλλο αντικείμενο, η ταχύτητα περιστροφής μεγιστοποιείται και η ροπή μηδενίζεται, οπότε η παραγόμενη ισχύς είναι μηδέν. Αντίθετα, όταν ο κινητήρας κολλάει , η ροπή μεγιστοποιείται και η ταχύτητα περιστροφής μηδενίζεται, οπότε η παραγόμενη ισχύς είναι πάλι μηδέν. Ανάμεσα στα δύο αυτά άκρα, την ελεύθερη περιστροφή και το κόλλημα, ο κινητήρας παράγει όντως χρήσιμο

έργο και είναι αποδοτικός. Το πόσο αποδοτικός είναι εξαρτάται από την ποιότητα του κινητήρα. Κάποιοι κινητήρες είναι πολύ αποδοτικοί, ενώ άλλοι σπαταλούν έως και τη μισή τους ενέργεια. Τα πράγματα γίνονται χειρότερα σε κινητήρες άλλου τύπου, όπως οι ηλεκτροστατικοί μικροκινητήρες, οι οποίοι χρησιμοποιούνται σε ρομπότ μινιατούρες.

Η ταχύτητα περιστροφής των περισσότερων κινητήρων DC χωρίς φορτίο (ελεύθερη περιστροφή) κυμαίνεται από 3000 έως 9000 στροφές ανά λεπτό (pm), δηλαδή από 50 έως 150 στροφές ανά δευτερόλεπτο. Αυτό σημαίνει ότι η ταχύτητα περιστροφής των περισσότερων κινητήρων DC είναι υψηλή, αλλά ταυτόχρονα η ροπή τους είναι χαμηλή. Επομένως, είναι κατάλληλοι για την κίνηση αντικειμένων με μικρό βάρος που περιστρέφονται με μεγάλη ταχύτητα, όπως για παράδειγμα τα πτερύγια του ανεμιστήρα. Αυτό έρχεται σε αντίθεση με τα ρομπότ που εκτελούν εργασίες και πρέπει να κινήσουν αντικείμενα σημαντικού βάρους, όπως το σώμα τους (οι τροχοί).

Σερβοκινητήρες (servo motors)

Οι κινητήρες DC είναι ιδανικοί για συνεχή λειτουργία προς μία κατεύθυνση. Ωστόσο, τα ρομπότ πρέπει συχνά να χρησιμοποιήσουν τον κινητήρα βηματικά, ώστε να μετακινήσει ένα όργανο δράσης που συνδέεται με αυτόν (π.χ., ένα βραχίονα) σε συγκεκριμένη θέση. Οι κινητήρες που μπορούν να θέσουν τον άξονά τους σε συγκεκριμένη θέση ονομάζονται σερβοκινητήρες (servo motors). Οι σερβοκινητήρες χρησιμοποιούνται πολύ σε παιχνίδια, όπως τα τηλεκατευθυνόμενα αυτοκίνητα (για τη ρύθμιση της κατεύθυνσης κίνησης) και τα τηλεκατευθυνόμενα αεροσκάφη (για τη ρύθμιση της θέσης των πτερυγίων). Παρόλο που φαίνεται να διαφέρουν οι κινητήρες DC και οι σερβοκινητήρες στην πραγματικότητα ένας σερβοκινητήρας είναι το αποτέλεσμα της προσθήκης σε έναν κινητήρα DC των εξής στοιχείων:

- Ενός μηχανισμού μείωσης της σχέσης μετάδοσης, για τους λόγους που προαναφέραμε.
- Ενός αισθητήρα θέσης του άξονα του κινητήρα, ο οποίος παρακολουθεί το βαθμό και τη φορά περιστροφής του άξονα.
- Ένα ηλεκτρονικό κύκλωμα, το οποίο ελέγχει τον κινητήρα ως προς το βαθμό και τη φορά περιστροφής του άξονα.

Σκοπός της λειτουργίας των σερβοκινητήρων είναι η περιστροφή του άξονα του κινητήρα στην επιθυμητή θέση. Η θέση αυτή μπορεί να βρίσκεται σε οποιοδήποτε σημείο εντός 180 μοιρών από το σημείο αναφοράς, προς οποιαδήποτε κατεύθυνση. Επομένως, το εύρος περιστροφής του σερβοκινητήρα δεν είναι συνήθως 360 μοίρες, αλλά 180 μοίρες.

Η περιστροφή του άξονα από 0 έως 180 μοίρες καθορίζεται από ένα ηλεκτρικό σήμα, το οποίο αποτελείται από μια ακολουθία παλμών. Κάθε φορά που ο κινητήρας λαμβάνει έναν παλμό, ο άξονάς του περιστρέφεται, ενώ όταν δεν λαμβάνει παλμό, παραμένει στάσιμος. Η αυξομείωση του ύψους των παλμών δημιουργεί ένα κυματοειδές μοτίβο, που ονομάζεται κυματομορφή (Waveform). Ο Βαθμός περιστροφής του άξονα κατά την αύξηση του παλμού εξαρτάται από τη διάρκεια του παλμού. Όσο μεγαλύτερη η διάρκεια του παλμού τόσο μεγαλύτερη η γωνία στροφής. Αυτό ονομάζεται διαμόρφωση εύρους παλμών (pulse-width modulation) καθώς το εύρος των παλμών είναι πολύ σημαντικός παράγοντας, γι' αυτό και πρέπει να ελέγχεται με ακρίβεια διαφορετικά ο κινητήρας δεν θα αποκριθεί σωστά, θα λειτουργήσει ακανόνιστα ή θα προσπαθήσει να περιστραφεί έξω από τα μηχανικά του όρια

(180 μοίρες). Αντίθετα, η διάρκεια μεταξύ των παλμών δεν είναι τόσο σημαντική, καθώς μεταξύ των παλμών ο άξονας παραμένει στάσιμος.

Ο έλεγχος θέσης (position control), κατά τον οποίο ο κινητήρας ελέγχει συνεχώς τη θέση του άξονά του, χρησιμοποιείται στους περισσότερους μηχανισμούς κίνησης των ρομπότ και επιτρέπει στους ηλεκτροκίνητους μηχανισμούς κίνησης να είναι πολύ ακριβείς, αλλά και πολύ άκαμπτοι. Στην πράξη, αυτό σημαίνει ότι ο μηχανισμός κίνησης κλειδώνει στην επιθυμητή θέση, προσπαθώντας σε κάθε περίπτωση να τη διατηρήσει, παράγοντας μεγάλες δυνάμεις. Αυτό όμως μπορεί να είναι αρνητικό για ένα ρομπότ. Μια εναλλακτική λύση είναι ο έλεγχος ροπής (torque control), κατά τον οποίο ο κινητήρας ελέγχει συνεχώς την επιθυμητή ροπή, ανεξάρτητα από τη θέση του άξονα. Ως αποτέλεσμα, ο μηχανισμός κίνησης λειτουργεί πολύ πιο ομαλά.

Μία τυπική περιγραφή ενός σερβο-μοτέρ του εμπορίου είναι και αυτή που ακολουθεί για το μοντέλο “KS-3620” (Σχήμα 15).

- Μοντέλο: KS-3620
- Ροπή(4.8V): 12.5 kg-cm
- Ροπή(6.6V): 15.0.0 kg-cm
- Ταχύτητα: 0.07sec (4.8V) | 0.66sec (6.0V)
- Τάση Λειτουργίας:4.8 ~ 6.6 DC Volts
- Βάρος: 55 g /Servo Motor
- Τύπος Ρουλεμάν: Ball Bearing x 2
- Τύπος μοτέρ: high-speed magnetic motor
- Θερμοκρασίες λειτουργίας: -20 ℃ -60 ℃
- Συχνότητα λειτουργίας:500μs / 2500hz

Σχήμα 13: Σέρβο-μοτέρ KS-3620 Κινεζικής προέλευσης

Όργανα δράσης

Τα όργανα δράσης (effectors) επιτρέπουν στο ρομπότ να δρα, δηλαδή να εκτελεί φυσικές ενέργειες. Είναι ό,τι καλύτερο μετά τα πόδια, τα πτερύγια, τα φτερά και διάφορα άλλα μέλη του σώματος των ζώων που τους επιτρέπουν να κινούνται. Τα όργανα δράσης χρησιμοποιούν υποκείμενους μηχανισμούς, όπως είναι οι μύες στα ζώα έτσι και οι κινητήρες στη ρομποτική που ονομάζονται μηχανισμοί κίνησης (actuators) κάνουν όλη τη δουλειά. Όπως και με τους αισθητήρες, τα όργανα δράσης και οι μηχανισμοί κίνησης των ρομπότ είναι πολύ διαφορετικοί από τους αντίστοιχους βιολογικούς. Χρησιμοποιούνται για δύο βασικές δραστηριότητες:

1. Μετακίνηση (locomotion) : Κίνηση στο χώρο, μετακίνηση σε επιθυμητά σημεία

2. Χειρισμός (manipulation) : Χρήση αντικειμένων.

Τα παραπάνω αντιστοιχούν στις δύο βασικές θεματικές υποπεριοχές της ρομποτικής:

1. Τη ρομποτική κίνησης (mobile robotics), η οποία ασχολείται με τα ρομπότ που κινούνται στο χώρο (κυρίως στο έδαφος, αλλά και στον αέρα και κάτω από το νερό).

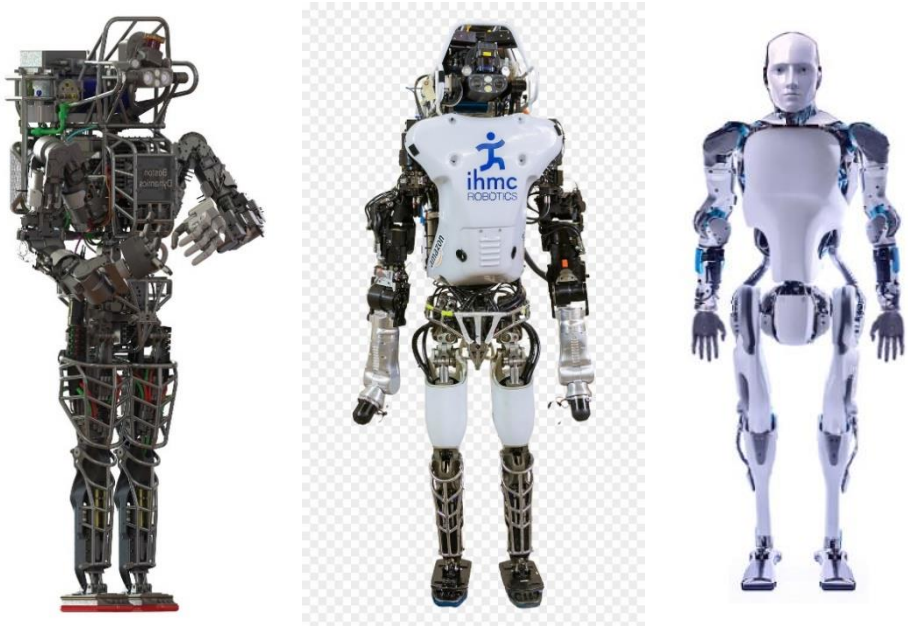
2. Τη ρομποτική χειρισμών (manipulation robotics), που ασχολείται κυρίως με διάφορους τύπους ρομποτικών βραχιόνων.

Τα κινούμενα ρομπότ χρησιμοποιούν μηχανισμούς μετακίνησης (όπως ρόδες, ερπύστριες και πόδια) και κινούνται συνήθως στο έδαφος. Τα υποβρύχια και τα ιπτάμενα

ρομπότ είναι και αυτά κινούμενα, με τη διαφορά ότι η κίνησή τους δεν περιορίζεται στο έδαφος και επομένως ο έλεγχός τους είναι ακόμη πιο δύσκολος.

Οι χειριστές (manipulators) αναφέρονται σε διάφορους ρομποτικούς βραχίονες και λαβίδες που μπορούν να κινούνται σε μία ή περισσότερες διαστάσεις. Υπάρχουν τουλάχιστον τέσσερις τύποι ρομποτικών λαβίδων όπως είναι οι λαβές κενού αέρος, πνευματικές λαβές, υδραυλικές λαβές και οι σέρβο-ηλεκτρικές λαβές. Η λαβή κενού αέρος αποτελεί την ποιο διαδεδομένη εφαρμογή στην κατασκευή λόγω του υψηλού επιπέδου ευελιξίας.

Ο διαχωρισμός μεταξύ ρομποτικής κίνησης και ρομποτικών χειρισμών σταδιακά εξαφανίζεται, καθώς αναπτύσσονται πιο σύνθετα ρομπότ, όπως τα ανθρωποειδή, τα οποία μπορούν να κινούνται στο χώρο και ταυτόχρονα να χειρίζονται αντικείμενα. Στο Σχήμα 16 παρουσιάζεται η τεχνολογική εξέλιξη ενός ρομπότ της εταιρίας ρομποτικών εφαρμογών Boston Dynamic το οποίο συνδυάζει κίνηση και χειρισμούς.



Σχήμα 14: Το ανθρωπόμορφο μοντέλο Atlas της Boston Dynamics κατά χρονική εξέλιξη.

2.1.2 Σύστημα αίσθησης

Οι αισθητήρες χωρίζονται σε δύο κατηγορίες

1) Αισθητήρες αυτοαντίληψης (proprioceptive sensors). Ανιχνεύουν στοιχεία της εσωτερικής κατάστασης του ρομπότ όπως οι θέσεις των τροχών οι γωνίες των αρθρώσεων των βραχιόνων και η κατεύθυνση στην οποία κοιτάει το κεφάλι.

2) Αισθητήρες εξωτερικής αντίληψης (exteroceptive sensors). Ανιχνεύουν τα στοιχεία του κόσμου γύρω από το ρομπότ όπως φωτεινότητα, αποστάσεις από αντικείμενα ήχους κλπ.

Μία άλλη διαφοροποίηση αισθητήρων γίνεται ανάλογα με το αν απαιτούν πρόσθετη τροφοδοσία για να λειτουργήσουν (ενεργοί αισθητήρες) ή είναι αυτόνομοι (παθητικοί αισθητήρες).

Οι ενεργοί αισθητήρες απαιτούν μια εξωτερική πηγή ισχύος για να λειτουργήσουν, η οποία παράγει κάποιο σήμα διέγερσης. Το σήμα εισάγεται από τον αισθητήρα για να παράγει ένα σήμα εξόδου. Για παράδειγμα, ένα θερμίστορ δεν παράγει κανένα ηλεκτρικό

σήμα, αλλά περνώντας ένα ηλεκτρικό ρεύμα μέσα από αυτό, η αντίστασή του μπορεί να μετρηθεί ανιχνεύοντας διακυμάνσεις στο ρεύμα ή την τάση στο θερμίστορ.

Οι παθητικοί αισθητήρες, αντίθετα παράγουν ηλεκτρικό σήμα σε απόκριση σε ένα εξωτερικό ερέθισμα και το χρησιμοποιούν για να παράγουν το σήμα εξόδου χωρίς την ανάγκη πρόσθετης πηγής ενέργειας. Τέτοια παραδείγματα είναι μια φωτοδίοδος και ένας πιεζοηλεκτρικός αισθητήρας, θερμοστοιχείο[38].

Ο αυξανόμενος ρυθμός της τεχνικής προόδου έχει φέρει τεχνικές καινοτομίες που αποσκοπούν στην κάλυψη των τωρινών αναγκών αλλά και των μελλοντικών καθώς νέες προκλήσεις αναδύονται στις ζωές των ανθρώπων. Ο συνδυασμός της ιατρικής επιστήμης και της μηχανικής έχει απλουστεύσει εργασίες όπως μία πολύπλοκη χειρουργική επέμβαση από ένα ρομποτικό χέρι. Για να καταγραφεί η κίνηση των ανθρώπινων άκρων, μπορούν να χρησιμοποιηθούν αισθητήρες. Ορισμένες εταιρείες έχουν σχεδιάσει διατάξεις, οι οποίες ενσωματώνουν επιταχυνσιόμετρα, γυροσκόπια, μαγνητόμετρα και μπορούν να τοποθετηθούν στα ανθρώπινα άκρα. Αυτές οι διατάξεις μπορούν να ενσωματωθούν σε ρούχα και να γίνουν φορετές προκειμένου να χρησιμοποιηθούν 1) στη μοντελοποίηση χαρακτήρων παιχνιδιών (Σχήμα 17α), 2) εικονική πραγματικότητα (Σχήμα 17β), 3) αναγνώριση δραστηριότητας (Σχήμα 17γ). Ένας αισθητήρας είναι μια συσκευή που μπορεί να μετρήσει κάποιο χαρακτηριστικό της κίνησης, που είναι ένα από τα τρία πρωτογενή χαρακτηριστικά της ρομποτικής (εκτός από το σχεδιασμό και τον έλεγχο), η αίσθηση παίζει σημαντικό ρόλο στις ρομποτικές εφαρμογές. Οι χειριστές ρομποτικών βραχιόνων μπορούν να έχουν διαφορετικές διαμορφώσεις και κινηματικούς περιορισμούς. Λίγοι από αυτούς τους περιορισμούς μπορούν να χαρτογραφηθούν αποτελεσματικά από τον τομέα των ανθρώπινων βραχιόνων στο περιορισμένο πεδίο κίνησης των συνδέσμων ενός ρομπότ.

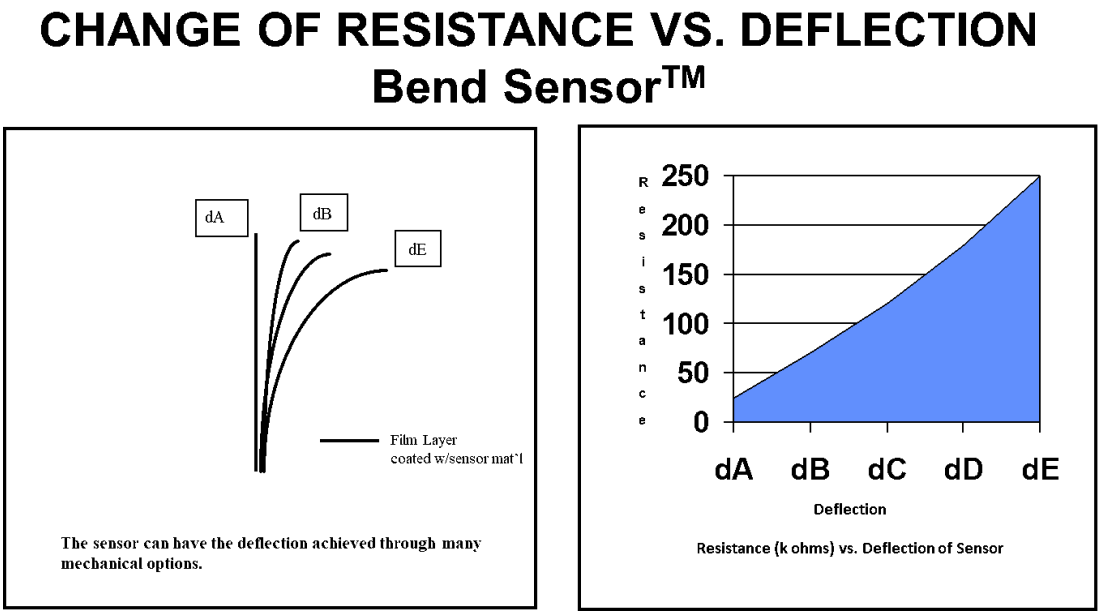
amount	Required data-measurement-packages per run
9	ActivePAL™ (AP1-9)
1	SmartCardia (SC1)
2	Myo-Armband (Myo1_blue, Myo2_red)
1	iPhone®
1	BPMS™(pressure mattress)
1	memorycard for handcamera
4	memorycards for wall cameras

(α) (β) (γ)

Σχήμα 17: α) Χρήση αισθητήρων για μοντελοποίηση χαρακτήρων, β) Χρήση αισθητήρων για εικονική πραγματικότητα, γ) Χρήση αισθητήρων για ανίχνευση δραστηριότητας.

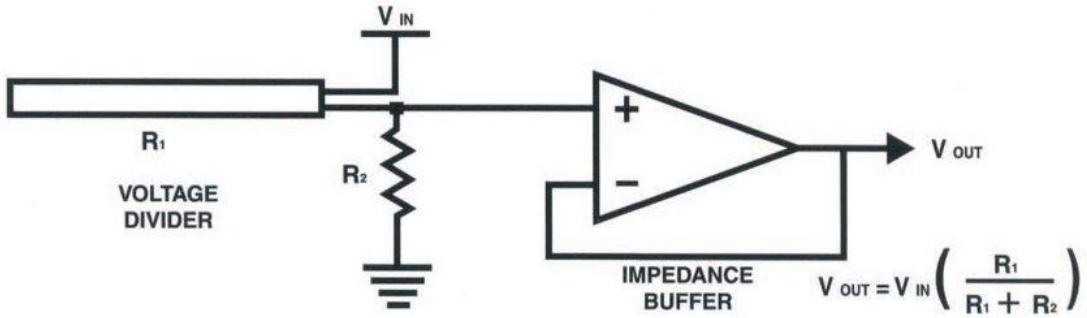
Αισθητήρες κάμψης

Οι αισθητήρες κάμψης είναι αναλογικές αντιστάσεις. Αυτές οι αντιστάσεις λειτουργούν ως μεταβλητοί διαιρέτες αναλογικών τάσεων. Μέσα στον αισθητήρα κάμψης υπάρχουν στοιχεία με αντίσταση άνθρακα σε λεπτό εύκαμπτο υπόστρωμα. Περισσότερος άνθρακας σημαίνει λιγότερη αντίσταση. Όταν το υπόστρωμα είναι λυγισμένο, ο αισθητήρας έχει αντίσταση ανάλογη με την ακτίνα καμπυλότητας και η αντίσταση εξόδου που παράγεται συσχετίζεται με την ακτίνα καμπής όπως φαίνεται στο Σχήμα 18.



Σχήμα 15: Μεταβολή αντίστασης με παράλληλη αλλαγή καμπυλότητας του αισθητήρα,

Το ρυθμιστικό εμπέδησης (Impedance Buffer) είναι ένας ενισχυτής που χρησιμοποιείται σε ένα κύκλωμα εύκαμπτου αισθητήρα όπως φαίνεται και στο Σχήμα 19. Το ρεύμα χαμηλής πόλωσης του ενισχυτή μειώνει το σφάλμα λόγω της σύνθετης αντίστασης του εύκαμπτου αισθητήρα ως διαιρέτη τάσης. Η κάμψη του αισθητήρα έχει ως αποτέλεσμα τη μεταβολή της ίδιας της αντίστασης. Το κύκλωμα επεξεργασίας σήματος χρησιμοποιείται για την ανάγνωση αυτών των αλλαγών αντίστασης και τις μεταφέρει σε μία διάταξη ADC (Analog to Digital Converter). Η διάταξη ADC μετατρέπει αυτές τις τιμές σε ισοδύναμες ψηφιακές τιμές [32].



Σχήμα 19: Βασικό κύκλωμα εύκαμπτου αισθητήρα

Μορφοτροπείς (transducers)

Οι μορφοτροπείς χρησιμοποιούνται συχνά στα συστήματα αυτοματισμού, μέτρησης και ελέγχου, όπου τα ηλεκτρικά σήματα μετατρέπονται από κάποιες φυσικές ποσότητες σε κάποιες άλλες (ενέργεια, δύναμη, ροπή, φως, κίνηση, θέση κ.λπ.). Η διαδικασία μετατροπής μιας μορφής ενέργειας σε άλλη είναι γνωστή ως μορφοτροπή. Οι μορφοτροπείς που μετατρέπουν τις φυσικές ποσότητες σε μηχανικές ποσότητες είναι γνωστοί ως μηχανικοί μορφοτροπείς. Οι μετατροπείς που μετατρέπουν τις φυσικές ποσότητες σε ηλεκτρικές ποσότητες είναι γνωστοί ως ηλεκτρικοί μορφοτροπείς. Παραδείγματα είναι ένα θερμοζεύγος που μετατρέπει τις διαφορές θερμοκρασίας σε μια μικρή τάση ή ο γραμμικός μεταβλητός

μετασχηματιστής διαφορικού (LVDT linear variable differential transformer) που χρησιμοποιείται για τη μέτρηση της μετατόπισης [33].

2.1.3 Σύστημα ελέγχου

Κάθε ρομποτικός χειριστής (manipulator) είναι ένα όργανο δράσης. Μπορεί να είναι οποιοσδήποτε τύπος λαβίδας, χεριού, βραχίονας η μέλος του σώματος που χρησιμοποιείται για να επηρεάσει και να μετακινήσει αντικείμενα στο περιβάλλον. Κατ' επέκταση ο όρος χειρισμός (manipulation) αναφέρεται σε κάθε κίνηση με συγκεκριμένο στόχο ενός οποιουδήποτε χειριστή. Οι χειριστές αποτελούνται συνήθως από ένα ή περισσότερα μέλη τα οποία συνδέονται μεταξύ τους με αρθρώσεις καθώς και με το τελικό όργανο δράσης. Τα μέλη του χειριστή (manipulator links) είναι τα επιμέρους στοιχεία τα οποία ελέγχονται ανεξάρτητα. Αν πάρουμε ως παράδειγμα τον ανθρώπινο βραχίονα το τμήμα από το ώμο έως τον αγκώνα είναι ένα μέλος ενώ ο πήχης είναι ένα άλλο. Το τελικό όργανο δράσης είναι το τμήμα του χειρισμού που επηρεάζει το περιβάλλον, στο χέρι για παράδειγμα μπορεί να είναι το δάχτυλο που ακουμπά ένα αντικείμενο. Για να μετακινηθεί το τελικό όργανο δράσης ενός χειριστή στο επιθυμητό σημείο θα πρέπει να υπολογίσουμε τις γωνίες των αρθρώσεων ενός χειριστή. Η μετατροπή αυτή από τις καρτεσιανές συντεταγμένες (x,y,z) στις γωνίες αρθρώσεων του χειριστή (πχ ενός βραχίονα) ονομάζεται ανάστροφη κινηματική ακριβώς επειδή αποτελεί το αντίστροφο της απλούστερης διαδικασίας της κινηματικής η οποία υπολογίζει τη θέση του άκρου του χειριστή με βάσει τις γωνίες όλων των αρθρώσεων.

Μία άλλη διάσταση ελέγχου είναι η δυναμική που αναφέρεται στις ιδιότητες της κίνησης και της ενέργειας ενός κινούμενου αντικειμένου. Η διαδικασία χειρισμού των ρομπότ αποτελεί αρκετά πολύπλοκο αντικείμενο και μερικοί τομείς που αποτελούν αντικείμενο έρευνας είναι α) η εύρεση των σημείων λαβής (grasp points) δηλαδή των σημείων που πρέπει να τοποθετηθούν τα δάκτυλα σε σχέση με το κέντρο βάρους την τριβή τα εμπόδια κλπ. β) η δύναμη στη λαβή δηλαδή πόσο δυνατή θα πρέπει να είναι η λαβή ώστε το ρομπότ να μην αφήσει το αντικείμενο να πέσει κάτω αλλά ούτε και να το σπάσει. γ) η συμμόρφωση (compliance) δηλαδή προσαρμογή στο περιβάλλον για την εκτέλεση εργασιών που απαιτείται επαφή (όπως ολίσθηση πάνω σε μία επιφάνεια). δ) η εκτέλεση εργασιών υψηλής δυναμικής (πχ ασκήσεις επιδεξιότητας).

Το σύστημα ελέγχου ρομπότ μας δίνει τη δυνατότητα να προγραμματίσουμε την εργασία του βραχίονα και ελέγχει τη κίνηση του βραχίονα κατά τη διάρκεια της εκτέλεσης της εργασίας που έχουμε προγραμματίσει. Αποτελείται από το υλικό τμήμα (hardware) και το λογισμικό τμήμα (software) [26,27].

Το υλικό τμήμα περιλαμβάνει: (α) έναν υπολογιστή στον οποίο εγκαθίσταται το λογισμικό του ρομποτικού ελεγκτή και ο οποίος αποθηκεύει το πρόγραμμα των εντολών που πρέπει να εκτελεστεί για μια συγκεκριμένη εργασία (β) τα ηλεκτρονικά ελέγχου, τα οποία υλοποιούν το βρόχο ελέγχου για τον κινητήρα κάθε άρθρωσης (γ) τα ηλεκτρονικά επικοινωνίας, τα οποία δίνουν τη δυνατότητα σύνδεσης του ελεγκτή με τα αισθητήρια των αρθρώσεων κ,,αι του άκρου και (δ) τους ενισχυτές ισχύος οι οποίοι χρησιμοποιούνται για την ενίσχυση του σήματος ελέγχου στο επίπεδο που απαιτείται από τους κινητήρες των αρθρώσεων.

Το λογισμικό τμήμα περιλαμβάνει: (α) ένα προγραμματιστικό περιβάλλον με χρήση κάποιας γλώσσας υψηλού επιπέδου (β) το κινηματικό λογισμικό το οποίο με τη θέση των αρθρώσεων καθορίζει τη θέση του άκρου και αντίστροφα, δηλαδή με δεδομένη την

επιθυμητή θέση του άκρου καθορίζει τις αντίστοιχες θέσεις κάθε κινητήρα άρθρωσης (γ) το λογισμικό ελέγχου το οποίο υλοποιεί τον αλγόριθμο ελέγχου κάθε άρθρωσης, δηλαδή με γνωστή την κατάσταση του κινητήρα (θέση, ταχύτητα), καθορίζει το απαιτούμενο σήμα ελέγχου ώστε να κινηθεί ο κινητήρας προς την επιθυμητή θέση (δ) το λογισμικό δυναμικών όρων, το οποίο χρησιμοποιεί δεδομένα για τις δυναμικές παραμέτρους του βραχίονα και την κατάστασή του, ώστε να προσαρμόσει το σήμα ελέγχου με στόχο την καλύτερη λειτουργία του βραχίονα και (ε) το βοηθητικό λογισμικό, το οποίο φροντίζει για την παρακολούθηση της λειτουργίας του βραχίονα και ενημερώνει το χρήστη για λάθη. Για την κατανόηση της συντονισμένης λειτουργίας του λογισμικού του ελεγκτή, ας θεωρήσουμε μια εντολή κίνησης και ας δούμε τα στάδια της επεξεργασίας και της εκτέλεσής της. Έστω λοιπόν η εντολή:

Εντολή: Κίνησε το άκρο από τη θέση Α που βρίσκεται στη θέση Β με επιθυμητή ταχύτητα V.

Αρχικά το σημείο Α είναι γνωστό στον ελεγκτή από τις θέσεις των μεταβλητών των αρθρώσεων. Έτσι, το λογισμικό του ελεγκτή έχει ως δεδομένα: (α) τις θέσεις Α, Β ως προς ένα ακίνητο σύστημα συντεταγμένων, τοποθετημένο π.χ. στη βάση του ρομπότ (β) τη διαδρομή που θα ακολουθηθεί, π.χ. ευθεία γραμμή μεταξύ Α και Β. η οποία συνήθως δηλώνεται από την επιλογή του τύπου της εντολής κίνησης και (γ) την ταχύτητα του άκρου κατά τη διάρκεια της κίνησης.

Όταν δοθεί η εντολή κίνησης, η γλώσσα προγραμματισμού περνά στο κινηματικό λογισμικό (1) τον κώδικα της εντολής που πρόκειται να εκτελεστεί (2) τον ορισμό του τρέχοντος ακίνητου συστήματος συντεταγμένων, (3) τον επιθυμητό στόχο (δηλ., το σημείο Β), (4) την ταχύτητα του άκρου και (5) τις παραμέτρους του προφίλ της ταχύτητας (δηλ. χρόνους επιτάχυνσης και επιβράδυνσης). Το κινηματικό λογισμικό υπολογίζει τις τιμές των μεταβλητών των αρθρώσεων για τη θέση Β με λύση του αντίστροφου κινηματικού προβλήματος, ελέγχει αν κάποιες από αυτές βρίσκονται έξω από τα όρια λόγω περιορισμών του μηχανισμού και στην περίπτωση αυτή, επιστρέφει στο βοηθητικό λογισμικό, το οποίο ειδοποιεί το χρήστη για το λάθος. Στην αντίθετη περίπτωση υπολογίζει το βήμα αύξησης της θέσης του άκρου στη μονάδα του χρόνου για τη δεδομένη ταχύτητα. Στα στάδια επιτάχυνσης και επιβράδυνσης το βήμα αυτό προσαρμόζεται για να δημιουργηθεί το επιθυμητό ομαλό προφίλ ταχύτητας . Στο σημείο αυτό ξεκινά η εκτέλεση της κίνησης, Υπολογίζεται η στιγμιαία επιθυμητή θέση του άκρου από την πρόσθεση του βήματος αύξησης με την τρέχουσα θέση και κατ’ αυτό τον τρόπο υπολογίζονται οι στιγμιαίες επιθυμητές τιμές των μεταβλητών των αρθρώσεων , οι οποίες και περνούν στο λογισμικό ελέγχου ως είσοδοι αναφοράς των ελεγκτών των αρθρώσεων. Ο κύκλος των τελευταίων βημάτων επαναλαμβάνεται έως το τέλος της κίνησης.

Έλεγχος κίνησης αρθρωτού βραχίονα: Το πρόβλημα του ελέγχου του αρθρωτού βραχίονα αναφέρεται στο καθορισμό των δυνάμεων και των ροπών που πρέπει να αναπτυχθούν ώστε να εξασφαλίζεται η εκτέλεση των εντολών των αρθρώσεων. Και για να εξασφαλίζεται η εκτέλεση των εντολών εργασίας του βραχίονα χρειάζεται η ταυτόχρονη ικανοποίηση των προδιαγραφών της λειτουργίας στη μεταβατική και μόνιμη κατάσταση. Η εργασία του βραχίονα αφορά την εκτέλεση προκαθορισμένων κινήσεων στον ελεύθερο χώρο και την εφαρμογή προκαθορισμένων δυνάμεων επαφής όταν ο βραχίονας αλληλοεπιδρά με το περιβάλλον του. Οι τεχνικές σχεδίασης τροχιάς δημιουργούν τις εισόδους αναφοράς στο σύστημα ελέγχου κίνησης του βραχίονα. Ελέγχοντας τους κινητήρες των αρθρώσεων κινούμε το βραχίονα. Ο αρθρωτός βραχίονας είναι μια αλυσίδα στερεών σωμάτων συνδεδεμένων μεταξύ τους, με αποτέλεσμα η κίνηση μιας άρθρωσης να μην κινεί μόνο το δικό της σύνδεσμο

αλλά και όλους τους επόμενους συνδέσμους, των οποίων οι αρθρώσεις επίσης κινούνται. Αυτή η σειριακή δομή του βραχίονα έχει έναν αριθμό από σημαντικές επιπτώσεις:

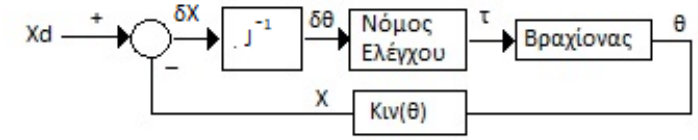
1) Η συνολική δύναμη αδρανείας και βαρύτητας που δρα στους κινητήρες αλλάζει κατά τη διάρκεια της κίνησης.

2) Λόγω των αλληλεπιδράσεων των αρθρώσεων, η κίνηση μιας άρθρωσης επιδρά σε όλες τις άλλες αρθρώσεις μέσω των δυνάμεων κεντρομόλου, Coriolis και τριβής.

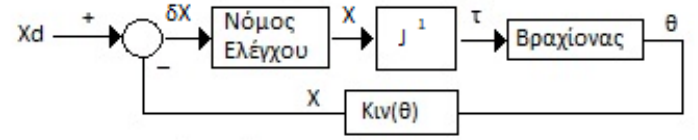
3) Όταν ο βραχίονας μεταφέρει αντικείμενα, αυτά αποτελούν μέρος του συστήματος, και επομένως αλλάζουν τις συνολικές δυναμικές ιδιότητες του βραχίονα.

Επομένως, οι ρομποτικοί βραχίονες είναι συστήματα μη γραμμικά, ισχυρά συζευγμένα και πολλών μεταβλητών με μεταβαλλόμενες παραμέτρους, γεγονός που τα καθιστά πολύ δύσκολα να αναλυθούν και να ελεγχθούν σε υψηλές ταχύτητες.

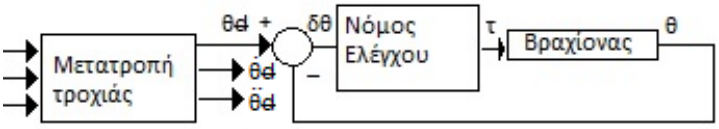
Οι τεχνικές ελέγχου ενός βραχίονα, καθώς και ο τρόπος υλοποίησής τους, επηρεάζουν την απόδοση του βραχίονα, καθώς και το εν δυνάμει εύρος των εφαρμογών τους. Από την άλλη πλευρά, το σύστημα οδήγησης των αρθρώσεων επηρεάζει την επιλογή της στρατηγικής ελέγχου που θα χρησιμοποιηθεί . Για παράδειγμα, η παρουσία μειωτήρων υψηλής σχέσης μειώνει τα αποτελέσματα των μη γραμμικοτήτων και έτσι τείνει να αποσυζεύξει τις αρθρώσεις και να γραμμικοποιήσει τη δυναμική του συστήματος. Στην περίπτωση αυτή είναι δυνατή η επιτυχής εφαρμογή αποκεντρωμένων στρατηγικών ελέγχου στις οποίες κάθε άρθρωση ελέγχεται ανεξάρτητα από τις άλλες. Το μειονέκτημα της εισαγωγής των μειωτήρων είναι η εμφάνιση ελαστικότητας και υστέρησης στις αρθρώσεις, καθώς και επιπλέον τριβών που περιορίζουν περαιτέρω την απόδοση του συστήματος. Όταν οι δυναμικές αλληλεπιδράσεις μεταξύ των αρθρώσεων δεν μπορούν να αγνοηθούν, επιλέγονται κεντρικές στρατηγικές ελέγχου οι οποίες λαμβάνουν υπόψη αυτές τις αλληλεπιδράσεις. Με βάση το γεγονός ότι η εργασία του βραχίονα περιγράφεται συνήθως στον λειτουργικό χώρο του βραχίονα (με προδιαγραφές για την κίνηση και δύναμη εφαρμογής της αρπάγης) ενώ οι δράσεις ελέγχου υλοποιούνται στο χώρο των αρθρώσεων, διακρίνουμε δύο βασικά σχήματα ελέγχου τα οποία διαφοροποιούνται από το χώρο στον οποίο δημιουργούνται τα σφάλματα και σχεδιάζονται οι νόμοι ελέγχου: το σχήμα ελέγχου στον λειτουργικό χώρο (Σχήμα 20 νούμερο 1 και 2 σχεδιαγράμματα) και το σχήμα ελέγχου στο χώρο των αρθρώσεων (Σχήμα 20 νούμερο 3 σχεδιάγραμμα).



1) Δημιουργία σφάλματος στον λειτουργικό χώρο και σχεδίαση ελέγχου στο χώρο των αρθρώσεων



2) Δημιουργία σφάλματος και σχεδίαση ελέγχου στον λειτουργικό χώρο



3) Δημιουργία σφάλματος και σχεδίαση ελέγχου στο χώρο των αρθρώσεων

τ η ροπή στον άξονα του συνδέσμου
θ η γωνιακή μετατόπιση του άξονα του συνδέσμου

Σχήμα 20 : Βρόχοι δημιουργίας και ελέγχου σφάλματος [27]

Το σχήμα ελέγχου στο χώρο των αρθρώσεων χρησιμοποιεί το αντίστροφο κινηματικό του βραχίονα για τη μετατροπή των επιθυμητών τροχιών του άκρου στο χώρο των αρθρώσεων. Οι είσοδοι αναφοράς αφορούν πλέον τις αρθρώσεις και έτσι ο νόμος ελέγχου για την παρακολούθησή τους σχεδιάζεται στο χώρο των αρθρώσεων (Σχήμα 20 σχεδιάγραμμα 3). Έτσι, το άκρο του βραχίονα ελέγχεται έμμεσα, με αποτέλεσμα οι όποιες αβεβαιότητες της δομής (π.χ. ελαστικότητα συνδέσμων) και ανακρίβειες στους κινηματικούς υπολογισμούς να επηρεάζουν αρνητικά την ακρίβεια των μεταβλητών του λειτουργικού χώρου.

Το σχήμα ελέγχου στον λειτουργικό χώρο εμπλέκει τη χρήση της Ιακωβιανής του βραχίονα στο βρόγχο ελέγχου, έχει όμως εν δυνάμει το πλεονέκτημα να ελέγχει άμεσα τις μεταβλητές του λειτουργικού χώρου. Στο σχεδιάγραμμα 1 (Σχήμα 20) τα σφάλματα ελέγχου δημιουργούνται στον λειτουργικό χώρο και απεικονίζονται στο χώρο των αρθρώσεων της αντίστροφης Ιακωβιανής, όπου και σχεδιάζεται ο νόμος ελέγχου. Το σχήμα αυτό παρουσιάζει προβλήματα στα ιδιάζοντα σημεία του βραχίονα στα οποία η Ιακωβιανή δεν αντιστρέφεται. Το πρόβλημα αυτό δεν παρουσιάζεται στο σχεδιάγραμμα 2 (Σχήμα 20) διότι τα σήματα ελέγχου σχεδιάζονται στον λειτουργικό χώρο του βραχίονα και απεικονίζονται στο χώρο των αρθρώσεων με την αντίστροφη Ιακωβιανή. Όταν η μέτρηση των μεταβλητών του λειτουργικού συστήματος δεν είναι διαθέσιμη (π.χ. με κάποιο οπτικό αισθητήριο) οι μεταβλητές αυτές υπολογίζονται από τις κινηματικές εξισώσεις του βραχίονα και τις μετρήσεις των μεταβλητών των αρθρώσεων όπως φαίνεται στο βρόχο ανάδρασης των σχεδιαγραμμάτων 1 και 2 με αποτέλεσμα να αποδυναμώνεται το πλεονέκτημα αυτού του σχεδιαγράμματος από τις ανακρίβειες των υπολογισμών [27].

2.2 Χαρακτηριστικά μεγέθη ρομποτικού βραχίονα

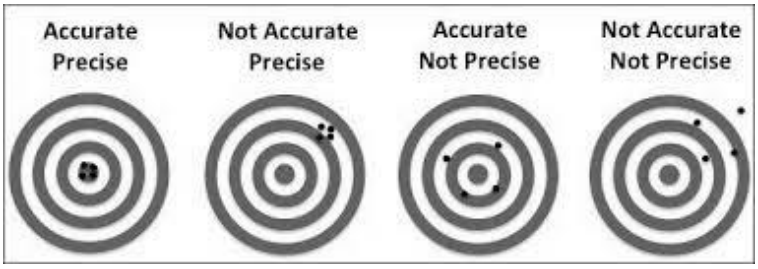
Ο αριθμός των ανεξάρτητων κινήσεων που μπορεί να κάνει ένα αντικείμενο στον τρισδιάστατο χώρο ονομάζεται βαθμός ελευθερίας (β.ε.) της κίνησής του. Ένα στερεό σώμα που κινείται ελεύθερα στο χώρο έχει έξι βαθμούς ελευθερίας, τρεις για τη θέση και τρεις για τον προσανατολισμό. Αν θεωρήσουμε ένα καρτεσιανό σύστημα συντεταγμένων ως σύστημα αναφοράς των θέσεων και του προσανατολισμού του σώματος, τότε οι τρείς ανεξάρτητες κινήσεις για τον προσανατολισμό μπορούν να οριστούν ως τρεις μεταφορικές κινήσεις κατά μήκος των αξόνων και ως τρείς περιστροφές γύρω από τους άξονες του συστήματος αναφοράς.

Οι ποιο συνηθισμένοι τύποι αρθρώσεων είναι:

- Η περιστροφική άρθρωση (rotary) πχ η σφαιροειδής (ball and socket) που παρέχει περιστροφική κίνηση γύρω από σταθερό άξονα.
- Η πρισματική άρθρωση που παρέχει γραμμική κίνηση .

Κάθε άρθρωση ενός ρομποτικού βραχίονα, είτε είναι πρισματική είτε περιστροφική, συμβάλλει με ένα βαθμό κινητικότητας, ο οποίος μπορεί να αντιπροσωπεύει ένα βαθμό ελευθερίας όταν η κίνησή του είναι ανεξάρτητη. Για παράδειγμα, οι πρώτες τρεις αρθρώσεις στις βασικές γεωμετρίες που αναφέρονται στην προηγούμενη παράγραφο υλοποιούν τους τρείς αναγκαίους βαθμούς ελευθερίας για την τοποθέτηση ενός σώματος στο χώρο ενώ οι τρείς περιστροφικές αρθρώσεις ενός ρομποτικού καρπού υλοποιούν τους τρεις αναγκαίους βαθμούς ελευθερίας για τον προσανατολισμό του σώματος. Για ελεύθερη κίνηση στο χώρο χρειαζόμαστε επομένως, ρομποτικούς βραχίονες με έξι βαθμούς ελευθερίας, οι οποίοι υλοποιούνται με έξι τουλάχιστον αρθρώσεις. Στην πραγματικότητα υπάρχουν ρομπότ με λιγότερες ή περισσότερες από έξι αρθρώσεις. Αυτά που έχουν λιγότερες αρθρώσεις έχουν περιορισμένη κίνηση, αλλά μπορεί να επαρκούν για κάποιες εργασίες. Για παράδειγμα ένα ρομπότ τεσσάρων βαθμών ελευθερίας με τις τρεις πρώτες αρθρώσεις σε γεωμετρία SCARA και με την τέταρτη άρθρωση περιστροφική και με άξονα περιστροφής τον άξονα της τρίτης (πρισματικής) άρθρωσης μπορεί να χρησιμοποιηθεί επιτυχώς σε εργασίες κάθετης συναρμολόγησης, πχ. Στην τοποθέτηση /συναρμολόγηση ολοκληρωμένων κυκλωμάτων σε τυπωμένες πλακέτες. Ένας βραχίονας με περισσότερες από έξι αρθρώσεις βελτιώνει την κινητική του ευελιξία αλλά παρουσιάζει περίπλοκο κινηματικό λογισμικό.

Δύο σημαντικά χαρακτηριστικά των ρομποτικών βραχιόνων είναι η ακρίβεια και η επαναληψιμότητα [26] [27]. Οι προδιαγραφές των βιομηχανικών βραχιόνων αναφέρονται συνήθως στην επαναληψιμότητα σε mm. Επαναληψιμότητα είναι η χαρακτηριστική διάσταση του όγκου μέσα στον οποίο το ρομπότ θα τοποθετήσει το άκρο του όταν κινηθεί κατ' επανάληψη σε μια επιθυμητή θέση από διάφορες αρχικές. Έτσι, ένας βραχίονας με επαναληψιμότητα 2 mm τοποθετεί το άκρο του μέσα σε σφαίρα διαμέτρου 2 mm (Σχήμα 21).



Σχήμα 21: Επαναληψιμότητα και ακρίβεια

Αν επιπρόσθετα το κέντρο της σφαίρας συμπίπτει ή είναι κοντά στη επιθυμητή θέση, τότε ο βραχίονας έχει ικανοποιητική ακρίβεια. Αν όμως αυτό δεν ισχύει, είτε διότι η επαναληψιμότητα είναι κακή είτε διότι το κέντρο της σφαίρας απέχει σημαντικά από την επιθυμητή θέση, τότε ο βραχίονας είναι ανακριβής (Σχήμα 21). Η ακρίβεια και η επαναληψιμότητα διαφέρουν για τις διάφορες θέσεις του άκρου στις οποίες μπορεί να κινηθεί ο βραχίονας. Οι διαφορές εξαρτώνται τόσο από την επιθυμητή θέση όσο και από την κατεύθυνση κίνησης του άκρου. Η ακρίβεια είναι, π.χ. μικρότερη για θέσεις κοντά στα όρια του άκρου εργασίας του βραχίονα όπου ο βραχίονας είναι σχεδόν εκτεταμένος.

Τέλος, ένα σημαντικό χαρακτηριστικό των βραχιόνων είναι το φορτίο. Το φορτίο είναι το μέγιστο βάρος του αντικειμένου που μπορεί να χειριστεί το ρομπότ με τις προδιαγραφές απόδοσης που δίνει ο κατασκευαστής. Το φορτίο στα βιομηχανικά ηλεκτρικά ρομπότ είναι της τάξεως των 100gr έως 80 kg ενώ τα υδραυλικά ρομπότ μπορούν να σηκώσουν βάρη που φτάνουν στα 300 kg και να τα μεταφέρουν με ταχύτητες έως 3m/s [27].

2.3 Ρομποτική όραση

Η μηχανική όραση ή αλλιώς όραση υπολογιστών είχε ραγδαία ανάπτυξη τις τελευταίες δεκαετίες, έχοντας φθάσει τα τελευταία χρόνια σε υψηλά επίπεδα αποτελεσματικότητας.

Όταν χρησιμοποιείται στις εφαρμογές των ρομποτικών χειριστών, καλείται ρομποτική όραση. Σε συνδυασμό με άλλα αισθητήρια συμβάλει στη λεγόμενη ενεργό αίσθηση (Active sensing) του περιβάλλοντος εργασίας από το ρομπότ. Από όλες τις τεχνικές της όρασης υπολογιστών, αυτές που έχουν ενδιαφέρον για τις ρομποτικές εφαρμογές είναι αυτές που συμβάλουν στην αναγνώριση τρισδιάστατων αντικειμένων στο χώρο και στη ακριβή καταγραφή της θέσης και των διαστάσεων τους.

Όταν υπάρχει έλλειψη πληροφορίας από αισθητήρες, ο μόνος τρόπος για να εκτελεστεί σωστά κάποια εργασία είναι η προσεκτική τοποθέτηση αντικειμένων σε γνωστές θέσεις, με την προϋπόθεση ότι οι γεωμετρικές ιδιότητες του χώρου και των αντικειμένων είναι επακριβώς γνωστές. Στο παράδειγμα σύλληψης αντικειμένων από τη μεταφορική ταινία υποθέτουμε ότι τι αντικείμενο είναι σε συγκεκριμένη θέση και στο σωστό προσανατολισμό, για να πιαστεί. Ο εξοπλισμός που απαιτείται, για να τοποθετηθούν τα αντικείμενα στους ακριβείς προσανατολισμούς είναι δαπανηρός, αλλά μπορεί να εξαιρεθεί, εάν χρησιμοποιούμε τεχνικές ρομποτικής όρασης για τον εντοπισμό του αντικειμένου. Η όραση μπορεί επίσης να χρησιμοποιηθεί για την ταυτοποίηση των διαφορετικών αντικειμένων, αυξάνοντας έτσι την ευελιξία του συστήματος. Κατά συνέπεια, η όραση μας δίνει σημαντικά κομμάτια πληροφορίας, τα οποία μπορούν να συνεργαστούν με το μοντέλο κίνησης για να αυξηθεί η ευελιξία του συστήματος ελέγχου μας.

Η ρομποτική όραση δεν ήταν παλαιότερα αναπόσπαστο κομμάτι ενός ρομποτικού συστήματος. Στην περίπτωση εκείνη, εάν κάποιος ήθελε να ενσωματώσει ρομποτική όραση στην εφαρμογή του, θα έπρεπε να το κάνει εξωτερικά. Δηλαδή, ένα πρόγραμμα όρασης υπολογιστή θα έτρεχε σε εξωτερικό υπολογιστή και τα αποτελέσματά του θα διαβιβάζονταν στο ρομποτικό σύστημα μέσω κατάλληλων δίαυλων επικοινωνίας. Στο Σχήμα 22 φαίνεται σε ελεύθερη σχεδίαση μία τέτοια εφαρμογή ρομποτικής όρασης σε ένα πείραμα συναρμολόγησης τρισδιάστατου puzzle. Στις πλευρές ενός τρισδιάστατου ξύλινου κύβου μπορούν να τοποθετηθούν ξύλινα αντικείμενα διαφόρων μεγεθών και σχημάτων, έτσι ώστε να συμπληρώνεται ένα 3D puzzle. Η τοποθέτηση γίνεται με τη βοήθεια του ρομποτικού

χειριστή ο οποίος “βλέπει” μέσω μίας κάμερας που ελέγχεται από ένα μικροεπεξεργαστή raspberry.

Σχήμα 162: 3-D puzzle και ρομποτικός βραχίονας καθοδηγούμενος από κάμερα Raspberry V2 ελεγχόμενη από μικροεπεξεργαστή Raspberry 3 b+.

Τα διάφορα αντικείμενα ρίχνονται με τυχαίο τρόπο πάνω στη τράπεζα εργασίας, κοντά στον κύβο. Μία κάμερα στην οροφή του χώρου εργασίας συλλαμβάνει την εικόνα των αντικειμένων και το πρόγραμμα όρασης αναγνωρίζει και ταυτοποιεί τα αντικείμενα, υπολογίζοντας ταυτόχρονα τις συντεταγμένες του κέντρου τους (οπή σημείου σύλληψης) και τον προσανατολισμό τους. Μετά τους απαραίτητους υπολογισμούς μετασχηματισμών συντεταγμένων το εξωτερικό πρόγραμμα επικοινωνεί μέσω μίας σειριακής θύρας με το ρομποτικό ελεγκτή δίνοντας οδηγίες για το πώς πρέπει να κινηθεί, ώστε να συλλάβει ένα-ένα τα αντικείμενα και να τα τοποθετήσει, περιστρέφοντάς τα κατάλληλα, στη σωστή θέση στο puzzle. Για να έχει επιτυχία το πείραμα, εκτός από αποτελεσματικούς αλγόριθμούς αναγνώρισης προτύπων, απαιτείται η σωστή βαθμονόμηση (calibration) της κάμερας στο πρόβλημα που χειρίζεται. Μια δεύτερη κάμερα, τοποθετημένη κατάλληλα στο άκρο του χειριστή, θα μπορούσε να συμβάλει στη αύξηση της ακρίβειας του πειράματος, δίνοντας τη δυνατότητα λήψης πρόσθετων εικόνων από κοντά, τη στιγμή που τα αντικείμενα προσεγγίζονται από το άκρο του ρομποτικού χειριστή.

Μία άλλη ιδιαίτερα σημαντική χρήση της ρομποτικής όρασης είναι η συνεισφορά της στη διαχείριση λαθών. Στη περίπτωση σύλληψης αντικειμένων επάνω σε μεταφορική ταινία, ένας αισθητήρας φωτεινής δέσμης θα μπορούσε να ανιχνεύσει την ύπαρξη ενός αντικειμένου στην ταινία. Η αξιοπιστία του συστήματος μπορεί να βελτιωθεί, αν γίνει ανίχνευση της παρουσίας του αντικειμένου με τη βοήθεια της όρασης και συγκριθεί αυτή με το σήμα από τον ανιχνευτή φωτεινής δέσμης, για έλεγχο λάθους και διόρθωση. Επιπλέον, το σύστημα όρασης μπορεί να καθορίσει τη θέση, τις διαστάσεις και τον προσανατολισμό του αντικειμένου. Το πρόγραμμα ρομπότ χρησιμοποιεί αυτές τις πληροφορίες θέσης, για να ορίσει τον προσανατολισμό της αρπάγης. Όταν το σύστημα εικόνας αναγνωρίζει ένα αντικείμενο, επιλέγει τον κατάλληλο σενάριο σύλληψης για εκείνο το αντικείμενο.

Κάποιες γλώσσες προγραμματισμού ρομπότ έχουν ενσωματωμένες εντολές για τη καταγραφή και την ανάλυση εικόνων και video. Η RAIL, για παράδειγμα έχει λειτουργίες για τον προσδιορισμό πολλών διαφορετικών χαρακτηριστικών (features) εικόνας. Η VAL II και η διάδοχός της V+ είναι εφοδιασμένες με εντολές για τη καταγραφή (VPICTURE) των εικόνων και video και τον εντοπισμό των αντικειμένων (VLOCATE). Η V+ συνεργάζεται στενά με το σύστημα AdeptVision, το οποίο έχει όλες τις ευκολίες βαθμονόμησης, σύλληψης, επεξεργασίας και χειρισμού εικόνων. Για να αναγνωρίσει τα αντικείμενα, εκτός από την

ανάλυση της εικόνας, ένα σύστημα εικόνας πρέπει να συσχετίζει τη θέση των επι μέρους λεπτομερειών της εικόνας με τη φυσική θέση των αντικειμένων σε σχέση με το ρομπότ. Πριν μπορέσουμε να χρησιμοποιήσουμε ένα σύστημα όρασης, πρέπει “να του διδάξουμε” αυτές τις λεπτομέρειες και να καθορίσουμε τη σχέση της εικόνας με το φυσικό χώρο.

Κατά τη διάρκεια της φάσης διδασκαλίας, ένας ή περισσότεροι δίσκοι γνωστής διαμέτρου τοποθετούνται σε γνωστές θέσεις στο πεδίο της όρασης της κάμερας. Η εικόνα αναλύεται και ένα σύνολο μετασχηματισμών αντικειμένων υπολογίζονται σε σχέση με τη κάτω αριστερή γωνία της εικόνας (πλαίσιο κάμερας). Αυτοί οι μετασχηματισμοί συνδυάζονται έπειτα με τους μετασχηματισμούς που περιγράφουν τις γνωστές φυσικές θέσεις των δίσκων, για να βαθμονομήσουν (calibrate) το σύστημα εικόνας. Κατά τη διάρκεια της ρύθμισης, υπολογίζουμε τη θέση του πλαισίου κάμερας, τον προσανατολισμό της εικόνας, και τις οριζόντιες και κάθετες κλίμακες, τα οποία συσχετίζουν τις διαστάσεις εικόνας με τις φυσικές διαστάσεις. Κατόπιν, τα αντικείμενα, που πρόκειται να αναγνωριστούν, “διδάσκονται” στο σύστημα μέσω παρουσίασης. Στο Σχήμα 23 παρουσιάζεται ένα ρομπότ τύπου SCARA σε εργασία υποβοηθούμενη από όραση. Η κάμερα είναι συνδεδεμένη με τον καρπό του ρομπότ και το πλαίσιο κάμερας καθορίζεται σε σχέση με το πλαίσιο καρπού, έτσι ώστε η θέση του πλαισίου κάμερας σε σχέση με τις βασικές συντεταγμένες να είναι το προϊόν του μετασχηματισμού του καρπού και του μετασχηματισμού της κάμερας. Η πρώτη εντολή στο ρομποτικό πρόγραμμα πρέπει να κινήσει τη κάμερα προς το χώρο της τράπεζας εργασίας. Στη συνέχεια, η εντολή VPICTURE συλλαμβάνει την εικόνα και η εντολή VLOCATE ψάχνει στην εικόνα για το αντικείμενο. Μόλις βρεθεί ένα αντικείμενο, υπολογίζεται η θέση του. Η αρπάγη κινείται έπειτα προς τη θέση σύλληψης

Σχήμα 173: Ρομπότ τύπου SCARA με ενσωματωμένη κάμερα.

Αμέσως παρακάτω δίνεται τμήμα απλουστευμένου κώδικα, που περιλαμβάνει τις εντολές. Για περισσότερες λεπτομέρειες χρήσης ενός συστήματος όρασης, όπως για παράδειγμα το Adept Vision, θα πρέπει κανείς να ανατρέξει στο εγχειρίδιο χρήσης του κατασκευαστή.

```
MOVE            table.ref           ; Μετακίνηση ρομπότ στην τράπεζα
VPICTURE                            ; Σύλληψη εικόνας
VLOCATE object2                     ; Αναγνώριση του αντικειμένου
MOVE            table.object2.ref   ; Μετακίνηση της αρπάγης στο αντικείμενο
```

ΚΕΦΑΛΑΙΟ 3

Υλοποίηση ρομποτικού βραχίονα με Μικροελεγκτή Raspberry

3.1 Σχεδιασμός και συναρμολόγηση πειραματικής διάταξης για τη μετακίνηση αντικειμένων με τη χρήση κάμερας.

Η κατασκευή του πρωτότυπου ρομποτικού μας βραχίονα, βασίστηκε σε προκατασκευασμένο εμπορικό σετ [34] που στη συνέχεια τροποποιήθηκε και προσαρμόστηκε στις ανάγκες του πειράματός μας. Η συναρμολόγηση και η σύνδεση των διάφορων περιφερειακών του (module) έγινε με βάση τις απαιτήσεις και τον αρχικό σχεδιασμό (Σχήμα 24).

Το σετ του βραχίονα αποτελείται από τη βάση, τις αρθρώσεις αλουμινίου, τα σερβομοτέρ με τον οδηγό τους, τη λαβίδα, και τη τροφοδοσία του ρεύματος ενώ προστέθηκε ο μικροελεγκτής Raspberry καθώς και τη κάμερα Pi. Η συναρμολόγηση έγινε με χρήση εργαλείων όπως σταυροκατσάβιδο, καστάνια και καρυδάκια. Η σύνδεση των σερβομοτέρ έγινε σειριακά και με συγκεκριμένη ακολουθία προκειμένου να τροφοδοτηθούν όλα από ένα καλώδιο το οποίο συνδέθηκε τελικώς στον οδηγό σερβομοτέρ της εταιρίας SB Components. Η παροχή ρεύματος προς τον μικροελεγκτή Raspberry Pi 3 B+ καθώς και τον συνδεδεμένο οδηγό σερβομοτέρ έγινε από μετασχηματιστή ρεύματος 5.1V - 2.5A προκειμένου να έχουμε σταθερή και αδιάληπτη λειτουργία. Βασικό στοιχείο της σωστής συναρμολόγησης ήταν η τήρηση των οδηγιών και η κατάλληλη ροπή συναρμογής προκειμένου να μην υπερβούμε τις πλαστικές αντοχές του αλουμινίου αλλά και να μην έχουμε φαινόμενα χαλάρωσης και υψηλών ανοχών κατά την κίνηση του βραχίονα.

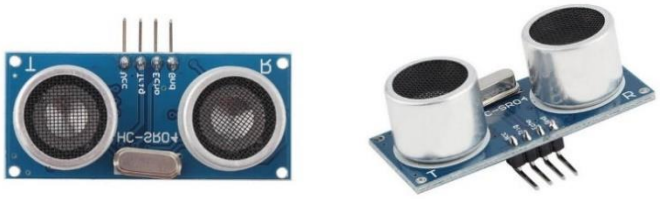
Σχήμα 184 : Εξαρτήματα του Ρομποτικού βραχίονα PiArm και οδηγίες συναρμολόγησης.

Οι σύνδεσμοι των αρθρώσεων του βραχίονα είναι περιστροφικοί και είναι κατασκευασμένοι από αλουμίνιο, σύνηθες υλικό κατασκευής για εφαρμογές ρομποτικών βραχιόνων ελαφριού τύπου.

Το τελικό στοιχείο δράσης είναι μία σέρβο-ηλεκτρική λαβίδα με λαστιχένιες άκρες. Και οι 6 βηματικοί κινητήρες που χρησιμοποιήθηκαν έχουν δυνατότητα περιστροφής 180^{o} Υπάρχουν διαφορετικοί ρυθμοί επικοινωνίας μεταξύ των σερβοκινητήρων και του μικροελεγκτή. Εδώ, τα σερβομοτέρ του ρομποτικού μας βραχίονα λειτουργούνε με 115200Hz συχνότητα επικοινωνίας (Σχήμα 25).

Σχήμα 25: Βηματικός κινητήρας (servo motor) διαδρομής 180^{0}

Ένα άλλο συμβατό περιφερειακό (Modules) που μπορεί να χρησιμοποιηθεί με το ρομποτικό μας βραχίονα είναι το αποστασιόμετρο HC-SR04 (Σχήμα 26) στην περίπτωση που θέλουμε να μετρήσουμε την απόσταση ενός αντικειμένου από το ρομποτικό μας βραχίονα. Εν προκειμένω δεν έγινε χρήση του αποστασιόμετρου αλλά θα μπορούσε για διαφορετικές εργασίες.



Σχήμα 26: Αποστασιόμετρο HC-SR04

Πρόκειται για μία μονάδα μέτρησης απόστασης με υπέρηχους η οποία έχει υψηλή ακρίβεια και τα τυφλά της σημεία ξεκινάνε σε πολύ κοντινή απόσταση (2εκ.).

Βασικές τεχνικές παράμετροι του περιφερειακού είναι:

1: Τάση: DC 5V

2: Στατικό ρεύμα: μικρότερο από 2mA

3: Ηλεκτρική έξοδος στάθμης: 5V υψηλή

4: Ηλεκτρική έξοδος στάθμης: 0V χαμηλή

5: Γωνία εισαγωγής: μέγιστη 15 μοίρες

6: Απόσταση ανίχνευσης: 2cm-450cm

7: Άλλα χαρακτηριστικά: 2mm Καλωδίωση: VCC, trig (πλευρά ελέγχου), echo (τερματικό λήψης), GND (Γείωση)

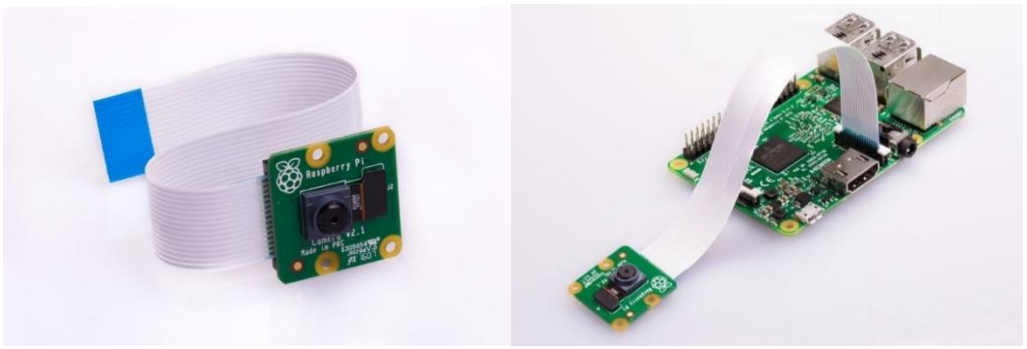
Βασικές λειτουργίες της περιφερειακού μέτρησης απόστασης είναι:

(1) Χρησιμοποιεί υψηλό σήμα ενεργοποίησης τουλάχιστον 10us

(2) Η μονάδα αποστέλλει αυτόματα οκτώ κύματα τετραγωνικής συχνότητας 40khz και ανιχνεύει αυτόματα αν υπάρχει σήμα επιστροφής.

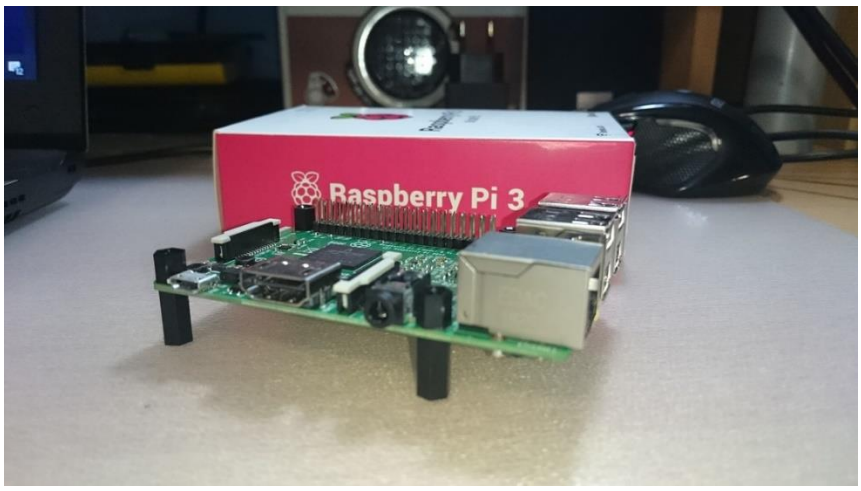
(3) Εάν υπάρχει σήμα επιστροφής, ορίζεται ο χρόνος της μέγιστης ηλεκτρικής στάθμης που ορίζεται ως ο χρόνος από τη έναρξη εκπομπής των υπερήχων μέχρι την επιστροφή του υπέρηχου. Έτσι η απόσταση του αντικειμένου στη δοκιμή μας ισούται με το χρόνο μέγιστης ηλεκτρικής στάθμης επί την ταχύτητα του ήχου (340m/sec) δια 2.

Ένα άλλο περιφερειακό που χρησιμοποιήθηκε είναι η Κάμερα Raspberry Pi3 V2 (Σχήμα 27). Η Μονάδα της κάμερας μπορεί να χρησιμοποιηθεί για λήψη βίντεο υψηλής ευκρίνειας, καθώς και για λήψη φωτογραφιών. Με τη χρήση βιβλιοθηκών μπορεί να γίνει χρήση ειδικών εφέ και υποστηρίζει λειτουργίες βίντεο 1080p30, 720p60 και VGA90, καθώς και καταγραφή. Συνδέεται μέσω καλωδίου με κορδέλα στη θύρα CSI του Raspberry Pi. Η κάμερα λειτουργεί με όλα τα μοντέλα Raspberry Pi 1, 2, 3 και 4. Είναι προσβάσιμη μέσω των οδηγών διασύνδεσης API MMAL και V4L και υπάρχουν πολλές βιβλιοθήκες τρίτων κατασκευαστών για αυτήν, συμπεριλαμβανομένης της βιβλιοθήκης Picamera Python.



Σχήμα 27: Η κάμερα Raspberry Pi Module v2 διαθέτει αισθητήρα Sony IMX219 8 megapixel.

Ο μικροελεγκτής Raspberry Pi που χρησιμοποιήθηκε είναι ένας μικρός υπολογιστής σε μέγεθος πιστωτικής κάρτας και δημιουργήθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation για την ευκολότερη εκμάθηση της επιστήμης των υπολογιστών στα σχολεία (Σχήμα 28).

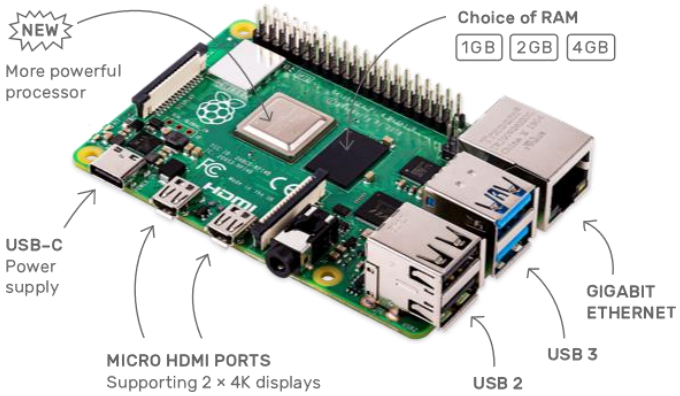


Σχήμα 28: Raspberry Pi 3

Ιστορική αναδρομή του μικροελεγκτή Raspberry Pi.

Η πρώτη γενιά (Raspberry Pi 1 model B) παρουσιάστηκε στο κοινό τον Φεβρουάριο του 2012. Το 2014 κυκλοφόρησε η νέα βελτιωμένη έκδοση του Raspberry Pi 1 η οποία ονομάστηκε RPi 1 model B+. Τον Απρίλιο του 2014 η εταιρία κυκλοφόρησε τον μικρότερο και οικονομικότερο υπολογιστή τσέπης. Το μοντέλο ονομάστηκε Raspberry Pi Zero και είχε κόστος μόλις 5 $. Τον Φεβρουάριο του 2015 κυκλοφόρησε το Raspberry Pi 2 το οποίο είχε την διπλάσια μνήμη RAM από τα προηγούμενα μοντέλα. Εν συνεχεία, τον Φεβρουάριο του 2016, κυκλοφόρησε στο Raspberry Pi 3 Model B το οποίο έχει ενσωματωμένο WiFi και Bluetooth και ακριβώς το ίδιο μέγεθος με το προηγούμενο. Φθάνοντας στο σήμερα τον Ιούνιο του 2019 ανακοινώθηκε το Raspberry Pi 4 Model B (Σχήμα 29)που θα τροφοδοτείται από usb type C. Το δημοφιλέστερο λειτουργικό σύστημα που χρησιμοποιείται από τον μικροεπεξεργαστή ονομάζεται Raspbian και βασίζεται στο Linux. Επίσης υπάρχουν και άλλα λειτουργικά συστήματα ειδικά σχεδιασμένα για το Raspberry Pi όπως Ubuntu, Windows 10 IOT Core, RISC OS καθώς και διάφορες άλλες εκδόσεις που προσομοιάζουν πλήρως ένα media center σύστημα [35]. Τεχνικά χαρακτηριστικά Raspberry Pi 3:

- Έχει 1.5GHz τετραπύρινο επεξεργαστή 64-bit ARM Cortex-A72 CPU (~3× performance)
- Μνήμη 1GB, 2GB, or 4GB of LPDDR4 SDRAM
- 1 Gigabit Ethernet Διπλής μπάντας 802.11ac ασύρματη δικτύωση
- Bluetooth 5.0
- Δύο θύρες USB 3.0 και δύο USB 2.0
- Υποστηρίζει διπλή οθόνη, με ανάλυση μέχρι και 4K
- Γραφικά VideoCore VI, υποστηρίζει OpenGL ES 3.x
- Αποκωδικοποίηση υλικού 4Kp60 βίντεο HEVC
- Πλήρης συμβατότητα με προηγούμενα προϊόντα Raspberry Pi



Σχήμα 29: Raspberry Pi 4 με περιγραφή των στοιχείων του

3.2 Προγραμματισμός της πειραματικής μας διάταξης

Στην παρούσα εργασία επιχειρήθηκε ο προγραμματισμός του ρομποτικού μας βραχίονα προκειμένου αυτός να μπορεί να αναγνωρίζει αντικείμενα με τη χρήση κάμερας, να τα πιάνει με τη χρήση μηχανικής αρπάγης και να τα εναποθέτει σε συγκεκριμένες θέσεις ανάλογα με τις ανάγκες της εργασίας μας.

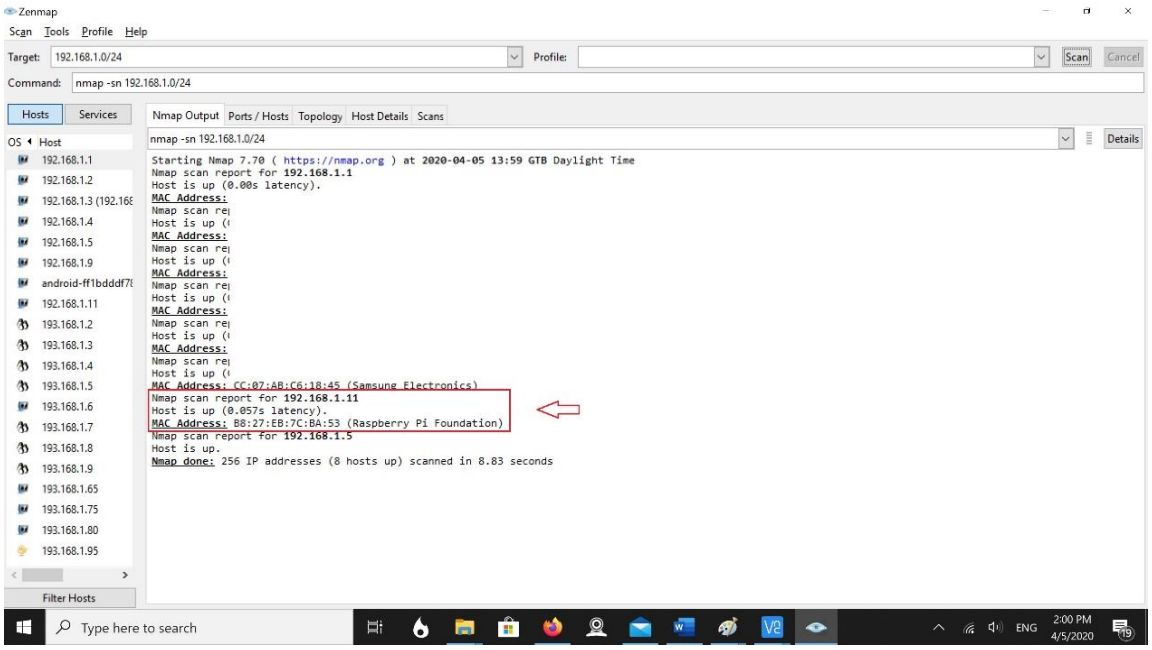
Επειδή οι επιτυχημένες προσπάθειες που έχουν δημοσιευτεί παγκοσμίως αναφορικά με τη χρήση εμπορικών μικροελεγκτών (όπως Raspberry, Arduino) στη καθοδήγηση ρομποτικών βραχιόνων με χρήση κάμερας είναι πολύ λίγες και αυτές που πραγματικά έχουν δώσει αποτελέσματα με ακριβή στοιχεία μετριούνται στα δάχτυλα των χεριών αναγκαστήκαμε στη παρούσα προσέγγιση του προβλήματος να συνθέσουμε ένα καινούργιο αλγόριθμο που χρησιμοποιεί πρωτότυπες τεχνικές ενσωματώνοντας χρήση βιβλιοθηκών ανοιχτού λογισμικού κάμερας και λογισμικού καθοδήγησης σερβομοτέρ.

3.2.1 Απομακρυσμένη διαχείριση ρομποτικού βραχίονα από υπολογιστή.

Υπάρχουν διάφοροι τρόποι να κατευθύνουμε και να αλληλοεπιδράσουμε με το ρομποτικό μας βραχίονα (Σχήμα 30). Ένας τρόπος που εδώ χρησιμοποιούμε είναι ο χειρισμός του ρομποτικού μας βραχίονα και ο έλεγχός του ασύρματα μέσω ενός υπολογιστή που συνδέεται διαδικτυακά με τον μικροελεγκτή Raspberry Pi. Ένας από τους τρόπους για να πετύχουμε την διαδικτυακή σύνδεση του Raspberry Pi με το Laptop είναι μέσω μίας εφαρμογής απομακρυσμένης διαχείρισης επιφάνειας εργασίας όπως η εμπορική VNC [36].

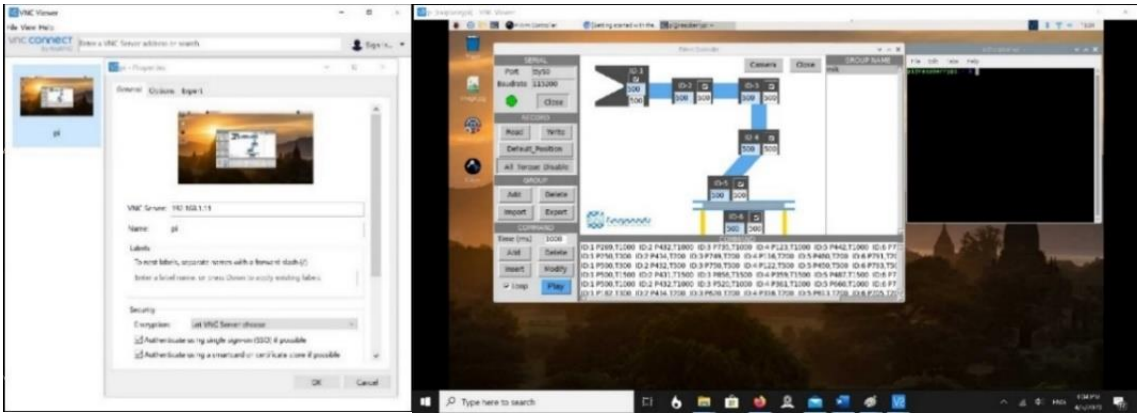
Σχήμα 30: Ενδεικτικοί τρόποι που μπορεί να γίνει η επικοινωνία του ρομποτικού μας βραχίονα με τον χειριστή

Τα βήματα που ακολουθούμε προκειμένου να εγκαταστήσουμε και να χρησιμοποιήσουμε αυτήν την εφαρμογή είναι δύο. 1ον Εκτελούμε εγκατάσταση της εφαρμογής VNC τόσο στον μικροελεγκτή Raspberry Pi όσο και στον υπολογιστή μας. 2ον Για να συνδεθούμε απομακρυσμένα χρειαζόμαστε να εισάγουμε τη διεύθυνση ip με την οποία επικοινωνεί ο μικροελεγκτής με το ασύρματο δίκτυο. Για να βρούμε την διεύθυνση ip αυτή χρησιμοποιούμε μία άλλη εφαρμογή την Nmap (Zenmap) [37] με την οποία εκτελούμε σάρωση όλων των ενεργών διευθύνσεων ip στο δίκτυό μας προκειμένου να εντοπίσουμε την ζητούμενη ip με την περιγραφή Raspberry Pi (Σχήμα 31).



Σχήμα 31: GUI της εφαρμογής Zenmap όπου διακρίνεται η διεύθυνση ip: 192.168.1.11 με την οποία είναι συνδεδεμένος ο μικροελεγκτής μας

Χρησιμοποιώντας τη διεύθυνση ip μέσω της εφαρμογής VNC επιτυγχάνουμε απομακρυσμένη σύνδεση του υπολογιστή μας με την επιφάνεια εργασίας του μικροελεγκτή μας (Σχήμα 32).



Σχήμα 32: GUI της εφαρμογής VNC όπου φαίνεται στην οθόνη του υπολογιστή στο δεξιό παράθυρο, η απομακρυσμένη επιφάνεια των Linux του Raspberry με ανοιχτή την εφαρμογή Pi Arm με την οποία γίνεται ο χειρισμός του ρομποτικού μας βραχίονα.

3.2.2 Λογισμικά που χρησιμοποιήθηκαν.

Το λογισμικό που χρησιμοποιήθηκε για την οπτική αναγνώριση των αντικειμένων βασίστηκε στις βιβλιοθήκες του ανοιχτού λογισμικού Open CV (Open Source Computer Vision Library) [38] το οποίο είναι μια βιβλιοθήκη λειτουργιών προγραμματισμού μηχανικής όρασης υπολογιστή σε πραγματικό χρόνο, ενώ το λογισμικό που απαιτήθηκε ως προς την κίνηση του βραχίονα βασίστηκε στο PiArm Control (λογισμικό της εταιρίας sb components που

ειδικεύεται σε ρομποτικά συστήματα) [34]. Αναφορικά με το Λογισμικό PiArm Control που χρησιμοποιήθηκε αυτό αποτελεί μία εφαρμογή για τον έλεγχο των κινήσεων του ρομποτικού βραχίονα και την υλοποίηση λειτουργιών του. Χάρη σε αυτό ο ρομποτικός μας βραχίονας μπορεί να κάνει διάφορες εργασίες όπως να επιλέγει και να τοποθετεί διάφορα αντικείμενα, να σχεδιάζει, να ανιχνεύει και να βλέπει τα αντικείμενα ή να εκτελέσει οποιαδήποτε άλλη μορφή κίνησης. Ο χειρισμός των κινήσεων του ρομποτικού βραχίονα έγινε κάνοντας χρήση τμήματος του συγκεκριμένου λογισμικού το οποίο μπορεί στην αρχική του μορφή να χρησιμοποιηθεί με τρείς τρόπους, α) με χρήση γραφικού περιβάλλοντος χρήστη (GUI), β) με χρήση χειριστηρίου joystick και γ) με φωνητικές οδηγίες.

Επιπρόσθετα λογισμικά που χρησιμοποιήθηκαν είναι τα Python Thonny και PyCharm για την επεξεργασία και την εκτέλεση αλγορίθμων αλλά και τη σύνθεση του κώδικά μας σε γλώσσα Python.

Έγινε χρήση της βιβλιοθήκης σχεδίασης Matplotlib σε γλώσσα προγραμματισμού Python καθώς της NumPy η οποία είναι μία βιβλιοθήκη που παρέχει υποστήριξη για μεγάλους, πολυδιάστατους πίνακες και περιλαμβάνει μια μεγάλη συλλογή μαθηματικών συναρτήσεων υψηλού επιπέδου για τη λειτουργία αυτών.

Άλλα προαιρετικά λογισμικά είναι το VNC [36] για την εξ αποστάσεως επικοινωνία του υπολογιστή μας με τον μικροελεγκτή Raspberry Pi, το λογισμικό Zenmap [37] που εντοπίζει τις ενεργές συνδέσεις συσκευών σε ένα δίκτυο

Τέλος, όλα τα παραπάνω συνδυάστηκαν προκειμένου να προκύψει ένα τελικό αρχείο που κάνει χρήση προκαθορισμένων σεναρίων και τρέχει σε πραγματικό χρόνο υλοποιώντας διάφορες εργασίες.

Εγκατάσταση και παραμετροποίηση των βιβλιοθηκών Open CV

Αναφορικά με την εγκατάσταση των βιβλιοθηκών εντολών Open CV (Computer Vision) [39] σε περιβάλλον Linux αυτό είναι ένα από τα δημοφιλέστερα και πληρέστερα λογισμικά οπτικής αναγνώρισης (Computer Vision). Προκειμένου να εγκαταστήσουμε τις βιβλιοθήκες του ανοιχτού λογισμικού Open CV το οποίο είναι ευρέως γνωστό για τους αλγόριθμους οπτικής αναγνώρισης και επεξεργασίας που χρησιμοποιεί τρέχουμε μία σειρά εντολών από το παράθυρο εντολών (terminal) του περιβάλλοντος Linux στο μικροϋπολογιστή Raspberry Pi. Συγκεκριμένα εκτελούμε τις εντολές: *sudo apt-get install git* # Εγκαθιστά το Git που είναι ένα δημόσιο αποθετήριο αρχείων. Η εντολή: *git clone https://github.com/opencv/opencv.git* # Πραγματοποιεί λήψη της ποιο πρόσφατης έκδοσης του opencv και την αποθηκεύει σε ένα φάκελο “opencv” στον τρέχοντα κατάλογο. Αφού μεταβούμε στο φάκελο “opencv” που έχουμε κατεβάσει τρέχουμε την εντολή: *cd opencv* δημιουργούμε ένα φάκελο “build” μέσω της εντολής: *mkdir build* και μεταβαίνουμε σε αυτόν: *cd build*

Τώρα που έχουμε όλα τα απαιτούμενα εξαρτώμενα αρχεία, εγκαθιστούμε το OpenCV. Η εγκατάσταση πρέπει να ρυθμιστεί με την εντολή CMake# καθορίζει ποιες μονάδες πρόκειται να εγκατασταθούν, τη διαδρομή της εγκατάστασης, ποιες πρόσθετες βιβλιοθήκες θα χρησιμοποιηθούν, εάν θα μεταγλωττιστούν τα βοηθητικά αρχεία και τα παραδείγματα κ.λπ. Οι περισσότερες από αυτές τις εργασίες γίνονται αυτόματα με διαμορφωμένες σωστά τις προεπιλεγμένες παραμέτρους. Αφού τρέξουμε την εντολή *cmake ../* # Η εντολή αυτή χρησιμοποιείται συνήθως για τη διαμόρφωση της βιβλιοθήκης OpenCV (εκτελείται από το φάκελο build). Εν συνεχεία δημιουργούμε τα αρχεία χρησιμοποιώντας την εντολή: *sudo make install*

Η εγκατάσταση έχει πλέον ολοκληρωθεί και η διαδρομή αποθήκευσης είναι συνήθως στο “/usr/local/” φάκελο.

Κατά τη χρήση του λογισμικού Open CV κάποιες βασικές εντολές για τον χειρισμό των εικόνων είναι:

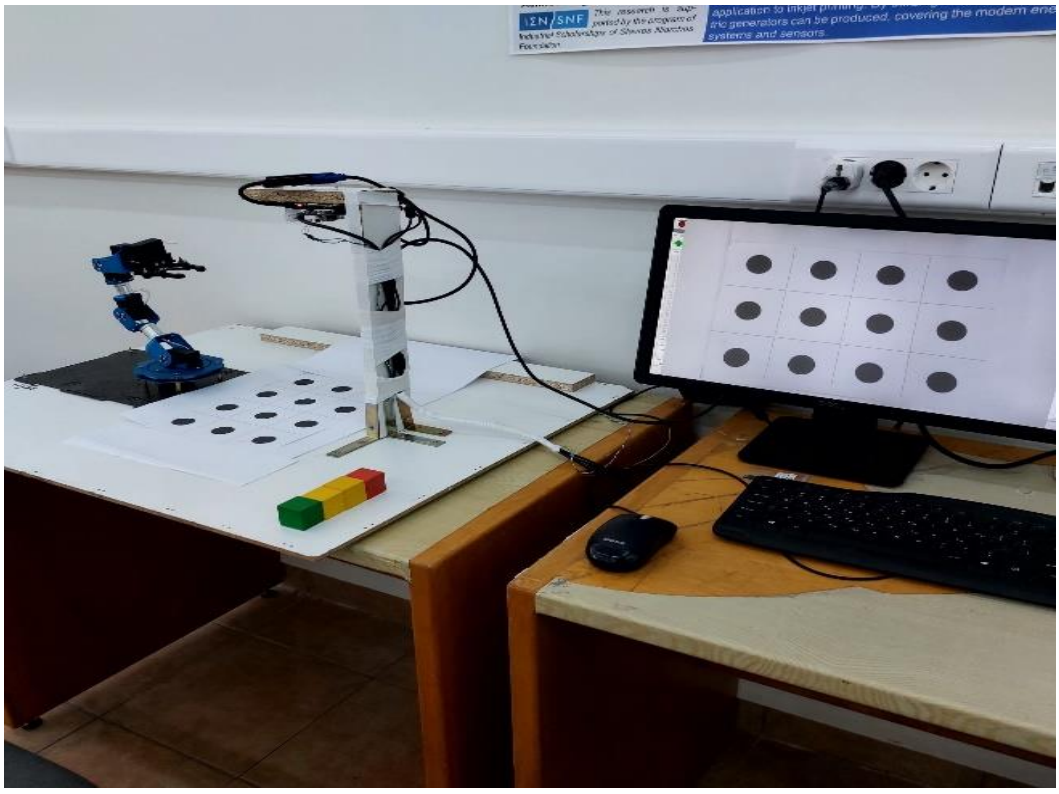
cv :: imread # Χρησιμοποιείται για να διαβάζουμε μια εικόνα από το αρχείο

cv :: imshow # Χρησιμοποιείται για να εμφανίζουμε μία εικόνα σε παράθυρο

cv :: imwrite # Χρησιμοποιείται για να γράφουμε μια εικόνα σε ένα αρχείο

3.2.3 Πρωτότυπος Αλγόριθμος και στάδια λειτουργίας του.

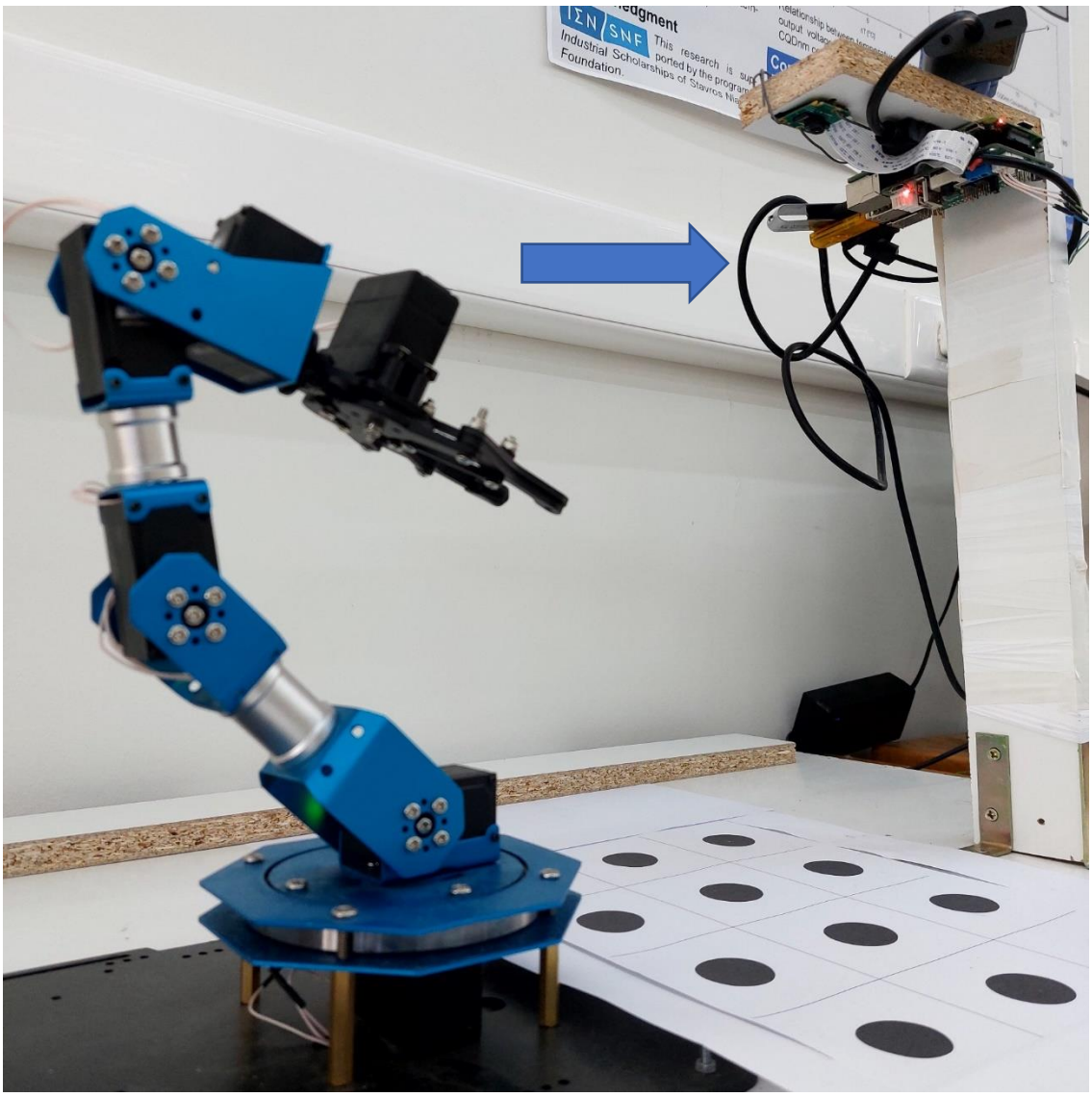
1) Επιλέχθηκαν 12 προκαθορισμένες θέσεις που θα μπορούν να τοποθετηθούν τα αντικείμενα (κύβοι εν προκειμένω) ως επιφάνειας εργασίας και σχεδιάστηκαν σε μία κόλλα Α4 οι θέσεις ως μαύροι κύκλοι. Στο Σχήμα 33 βλέπουμε την τελική διάταξη του πειράματος με την επιφάνεια εργασίας που αποτελείται από 12 θέσεις. Εκτελώντας αρχικά τον αλγόριθμο πραγματοποιείται λήψη μίας φωτογραφίας της επιφάνειας εργασίας (Σχήμα 33) χωρίς αντικείμενα από την κάμερά η οποία είναι σταθεροποιημένη στο βραχίονα παρατήρησης (Σχήμα 34).



Σχήμα 33: Ολόκληρη η διάταξη του πειράματος χωρίς αντικείμενα τοποθετημένα.

Η λήψη της φωτογραφίας όπου εδώ ονομάζουμε “Photo_Bcknd1” γίνεται με τη χρήση της εντολής :

```
img1 = cv2.imread('Photo_Bcknd1.jpg')
```

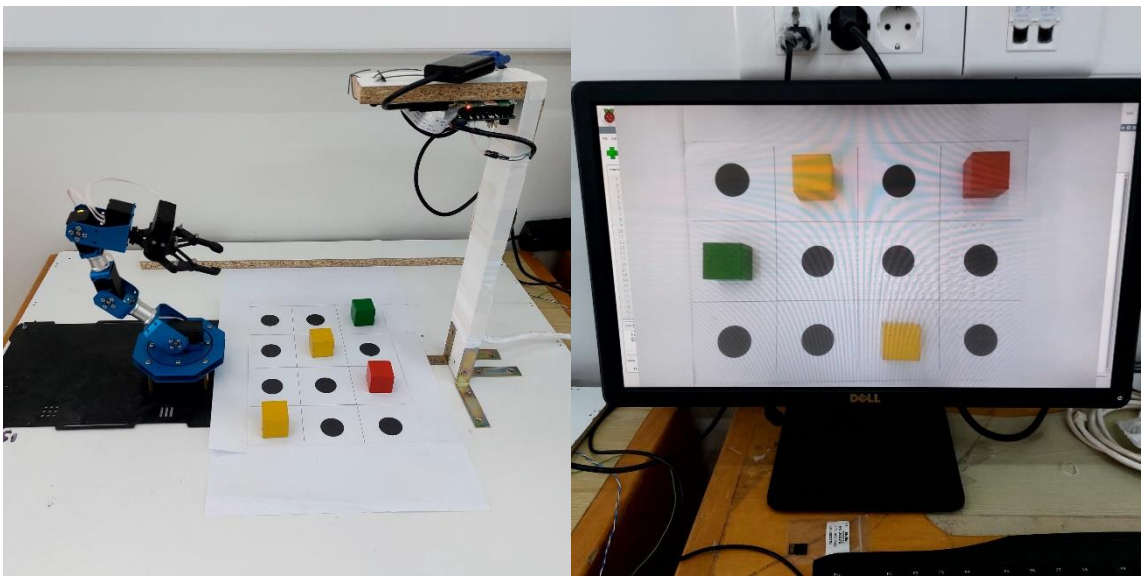


Σχήμα 34: Κάμερα Raspberry Pi Camera V2 τοποθετημένη σε σταθερό βραχίονα με το οπτικό της πεδίο να καταγράφει την επιφάνεια εργασίας (ενδεικτικό βέλος). Είναι εμφανής η συνδεσμολογία της με το μικροελεγκτή Raspberry Pi μέσω καλωδίου 15 ακίδων στη σειριακή διεπαφή CSI.

2) Αφού τοποθετηθούν τα αντικείμενα (κύβοι διαστάσεων 5cm x 5cm x 5cm) στις προκαθορισμένες θέσεις Σχήμα 35α, 35β, ο αλγόριθμος ζητάει έγκριση για να καταγράψει την εικόνα που βλέπει η κάμερα ώστε να παραχθεί η φωτογραφία των αντικειμένων σαν κάτοψη.

Η λήψη της φωτογραφίας όπου εδώ ονομάζουμε “AlmostX_Photo1” γίνεται με τη χρήση της εντολής :

```
img1 = cv2.imread('AlmostX_Photo1.jpg')
```



(α) (β)

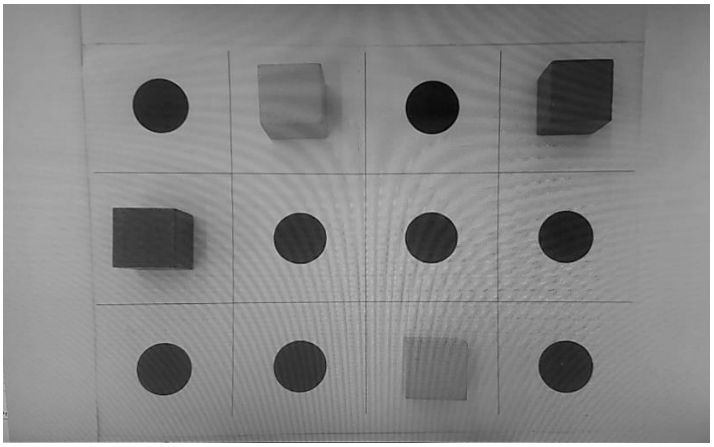
Σχήμα 35 α): Βλέπουμε ολόκληρη τη διάταξη του πειράματος με 4 αντικείμενα (κύβους) στην επιφάνεια εργασίας, β) Προβολή σε οθόνη της οριοθετημένης επιφάνειας εργασίας όπου είναι κατειλημμένες οι 4 από τις 12 πιθανές θέσεις των αντικειμένων.

3) Εν συνεχεία ο αλγόριθμος κάνει σύγκριση των δύο φωτογραφιών με χρήση της αντίστοιχης βιβλιοθήκης του Opencv και τη εντολής absdiff και βρίσκει τις διαφορές.

Αυτό επιτυγχάνεται με την χρήση της εντολής diff= cv2.absdiff(img1,img2) όπου img1 έχουμε δηλώσει την φωτογραφία Photo_Bcknd1 και img2 έχουμε δηλώσει την φωτογραφία AlmosX_Photo1.

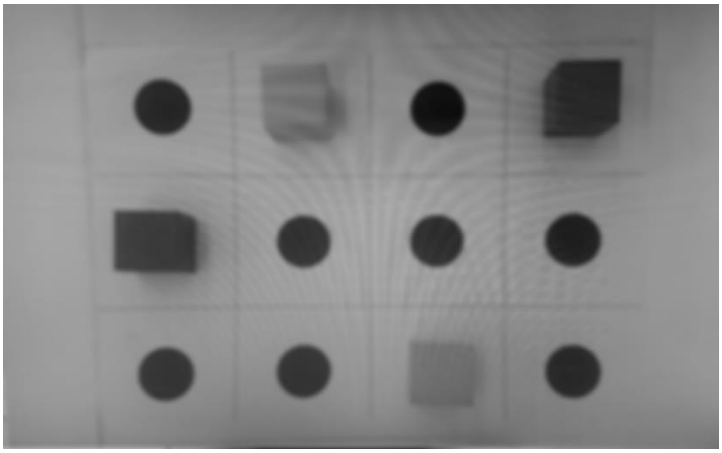
Εκτελεί οπτικές τροποποιήσεις βασισμένες σε τεχνικές οπτικής επεξεργασίας του OpenCV όπως:

α) Μετατρέπει σε ασπρόμαυρες τις φωτογραφίες (grayscale) προκειμένου να διευκολυνθεί η οπτική τμηματοποίηση, εδώ χρησιμοποιείται η εντολή gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY) (Σχήμα 36).



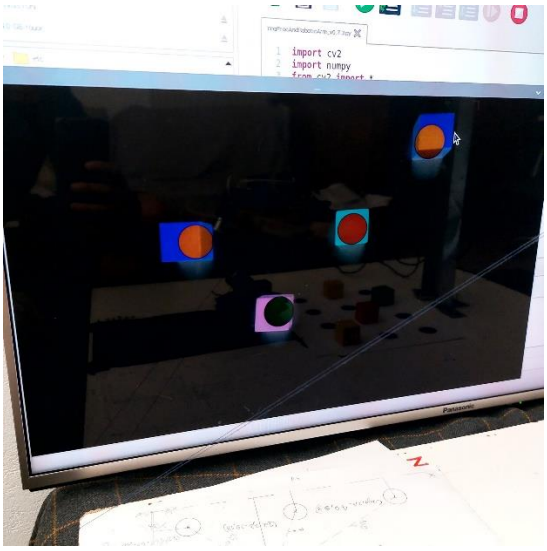
Σχήμα 36: Εφαρμογή φίλτρου εικόνας σε κλίμακα του γκρι

β) Θολώνει την εικόνα για καλύτερη επεξεργασία και τμηματοποίηση της φωτογραφίας, χρησιμοποιήθηκε η εντολή blur = cv2.GaussianBlur(gray, (5, 5), 0) (Σχήμα 37).



Σχήμα 37: Εφαρμογή φίλτρου θόλωσης εικόνας (blur)

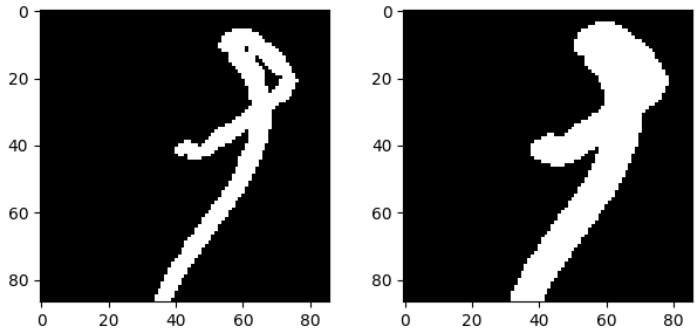
γ) Εφαρμόζει τη τεχνική thresholding με χρήση της εντολής _,thresh = cv2.threshold(blur,20,255,cv2.THRESH_BINARY) (Σχήμα 38). Η τεχνική thresholding είναι μια από τις πιο κοινές και βασικές τεχνικές τμηματοποίησης στην όραση υπολογιστή και μας επιτρέπει να διαχωρίσουμε το προσκήνιο (δηλαδή τα αντικείμενα που μας ενδιαφέρουν) από το φόντο της εικόνα. Είναι η δυαδοποίηση μιας εικόνας όπου επιδιώκουμε να μετατρέψουμε μια εικόνα σε κλίμακα του γκρι (grayscale) σε δυαδική εικόνα, όπου τα pixel είναι είτε 0 είτε 255.



Σχήμα 38: Εφαρμογή τεχνικής thresholding.

ε) Εκτέλεση συνάρτησης dilation από τις βιβλιοθήκες του OpenCV με την χρήση της εντολής dilated = cv2.dilate(thresh,None, iterations=5) . Η τεχνική dilate και erode είναι λειτουργίες μορφολογικής επεξεργασίας εικόνας. Η μορφολογική επεξεργασία εικόνας μέσω της εντολής dilate του OpenCV είναι μια διαδικασία όπου γίνεται τροποποίηση της γεωμετρικής δομής στην εικόνα. για να βρούμε το σχήμα το μέγεθος ή τη δομή ενός αντικειμένου (Σχήμα 39).

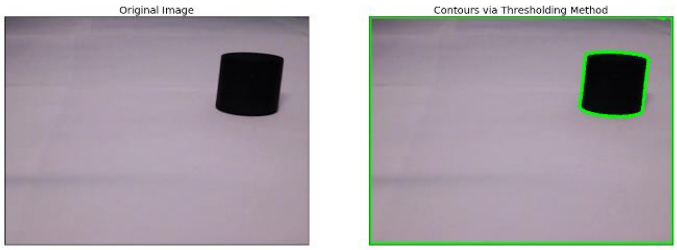
Η εντολή dilate προσθέτει εικονοστοιχεία στα όρια των αντικειμένων σε μια εικόνα, ενώ η erode αφαιρεί τα εικονοστοιχεία στα όρια των αντικειμένων. Ο αριθμός των pixel που προστίθενται ή αφαιρούνται από τα αντικείμενα μιας εικόνας εξαρτάται από το μέγεθος και το σχήμα του δομικού στοιχείου που χρησιμοποιείται για την επεξεργασία της εικόνας.



Σχήμα 39: Παράδειγμα εφαρμογής τεχνικής dilate.

ζ) Εντοπίζει την οριακή περιοχή που καταλαμβάνει το αντικείμενο/να, με την εντολή contours,_ = cv2.findContours(dilated,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE) (Σχήμα 40).

Η συνάρτηση findContours() της βιβλιοθήκης Open CV εφαρμόζεται για να βρούμε όλα τα οριακά σημεία(x,y) ενός αντικειμένου στην εικόνα. Τα περιγράμματα μπορούν να εξηγηθούν απλώς ως μια καμπύλη που ενώνει όλα τα συνεχόμενα σημεία (κατά μήκος του ορίου), έχοντας το ίδιο χρώμα ή ένταση.

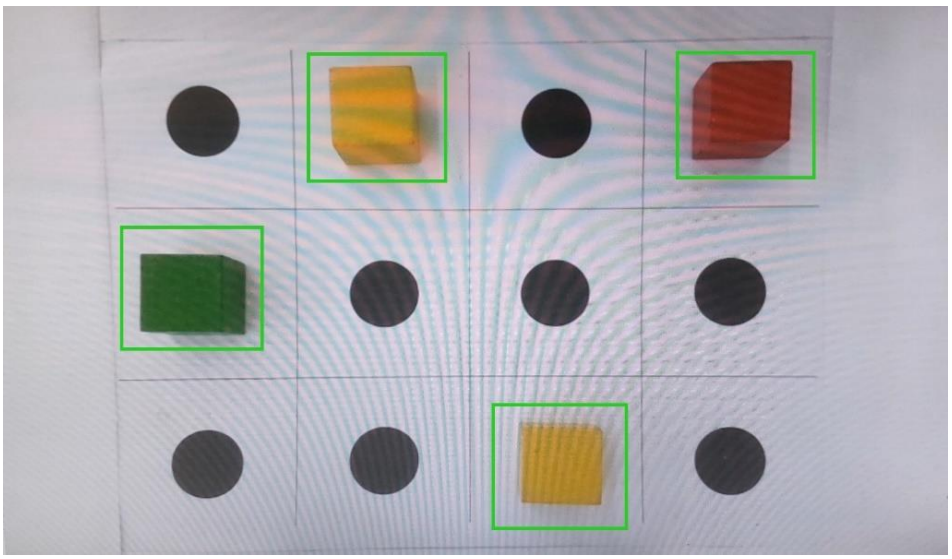


Σχήμα 40: Παράδειγμα εφαρμογής λειτουργίας Contours.

η) Σχεδιάζει τα πλαίσια (τετράγωνο 255mm εν προκειμένω) των αντικειμένων που εντοπίζονται, (Σχήμα 41) με χρήση της εντολής:

```
x,y,w,h =cv2.boundingRect(c)
   if x<=540 :
      cv2.rectangle(img2, (x,y),(x+w, y+h),(0, 255, 0), 2)
      print()
```

Η cv2 boundingrect() είναι μια συνάρτηση που χρησιμοποιείται για τη δημιουργία ενός κατά προσέγγιση ορθογωνίου μέσα στην κύρια εικόνα. Η κύρια χρήση αυτής της εντολής είναι η επισήμανση της περιοχής ενδιαφέροντος αφού ληφθεί το εξωτερικό σχήμα της εικόνας. Με τις κατάλληλες σημάνσεις, μπορούμε εύκολα να αναδείξουμε τις περιοχές ενδιαφέροντος σε μια εικόνα.



Σχήμα 41: Εύρεση αντικειμένων και σχεδιασμός πλαισίων τους στην επιφάνεια εργασίας

4) Καλεί το αρχείο Arm Positions με την εντολή

```
Pos=ArmPositions(x,y)
  print("Positions   x:", x,"y: ", y)
```

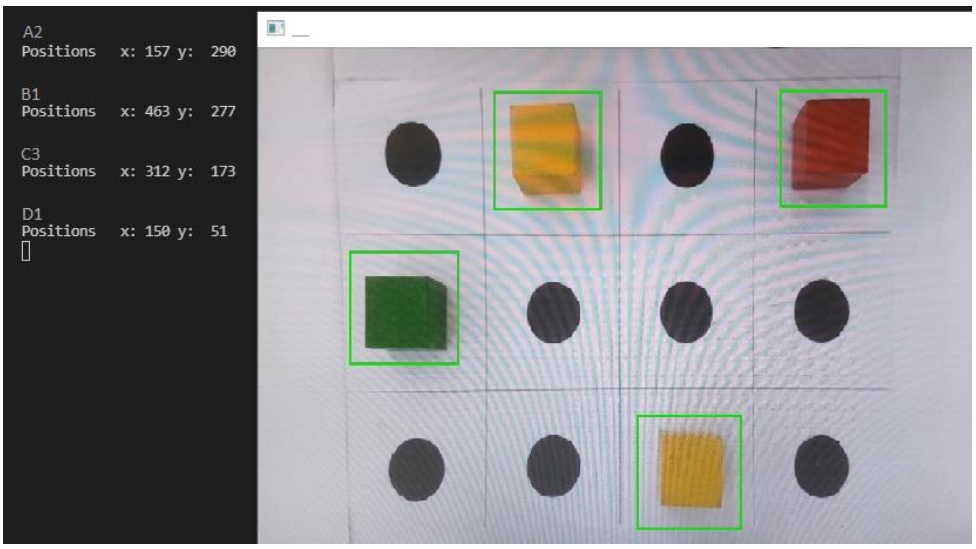
Η εντολή def ArmPositions(x,y) και είναι μία ρουτίνα (function) που εντοπίζει σε ποιο από τα διαστήματα συντεταγμένων που έχουμε χωρίσει την επιφάνεια εργασίας εν προκειμένω 12 διαστήματα, βρίσκονται οι x και y συντεταγμένες των αντικειμένων που τοποθετήσαμε στην επιφάνεια εργασίας και εμφανίζει τη θέση τους ώς αλφαριθμητικό χαρακτήρα (A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4,D1,D2,D3,D4). (Σχήμα 42)

```
def ArmPositions(x,y) :

    if (x>=410 and x<=580) and (y>=270 and y<=390) :
        print('A1')
        ArmFile= 'A1'
    elif (x>=270 and x<=430) and (y>=270 and y<=390) :
        print('A2')
        ArmFile= 'A2'
    elif (x>=80 and x<=290) and (y>=270 and y<=390) :
        print('A3')
        ArmFile= 'A3'
```

Σχήμα 42: Εύρεση των διαστημάτων συντεταγμένων που βρίσκονται τα αντικείμενα

5) Ακολούθως καλείται η ρουτίνα ArmPositions(x,y) η οποία απεικονίζει τις συντεταγμένες x, y που έχουμε υπολογίσει προηγουμένως (ειδικότερα οι συντεταγμένες x, y αναφέρονται στο σημείο της πάνω αριστερής και της πάνω δεξιάς γωνίας αντίστοιχα του πλαισίου που έχει σχεδιαστεί περιμετρικά των αντικειμένων (Σχήμα 43).

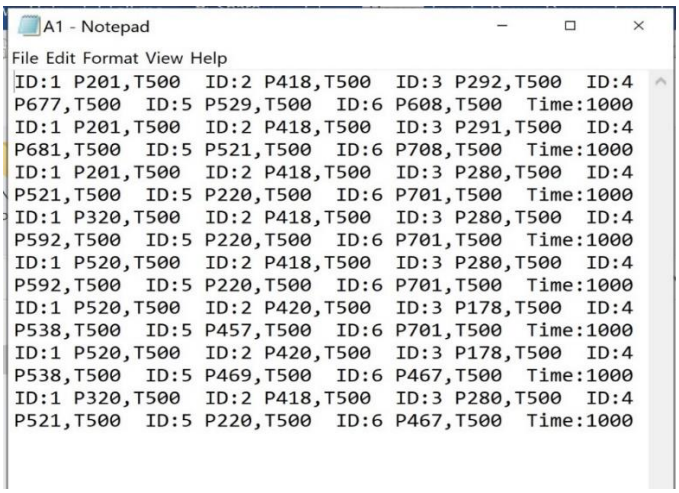


Σχήμα 43: Καθορισμός των θέσεων των αντικειμένων (εδώ έχουμε A2, B1, C3, D1)

6) Καλεί απο το φάκελο COORDS ο οποίος περιέχει τα σενάρια κίνησης του βραχίονα, το κατάλληλο σενάριο ανάλογα με τη θέση που βρίσκεται το αντικείμενο στις στήλες Ax,Bx,Cx, Dx για 0<x<3. Τα σενάρια τα οποία αποτελούν προσχεδιασμένες κινήσεις του βραχίονα όπου για κάθε μία θέση του αντικειμένου από τις επιτρεπόμενες της επιφάνειας εργασίας καθοδηγούν το βραχίονα στην αντίστοιχη θέση προκειμένου να συγκρατήσει το αντικείμενο, να το μετακινήσει στην επιθυμητή θέση και να το αποθέσει (Σχήμα 44). Τα σενάρια έχουν δημιουργηθεί με χειροκίνητο τρόπο και ειδικότερα αφού τοποθετήθεί ο βραχίονας στη θέση που επιθημούμε καταγράφουμε την θέση των 6 σερβομοτέρ του και εν συνεχεία μετακινούμε το βραχίονα προς τη θέση που θέλουμε να κινηθεί όπου επίσης καταγράφουμε τη θέση των 6 σερβομοτέρ του. Η διαδικασία επαναλήφθηκε για αριθμό θέσεων αντικειμένων στην επιφάνεια εργασίας και επιθυμητών κινήσεων και προσθέσαμε το παράγοντα της ταχύτητας κίνησης αυξομειώνοντάς τον ανάλογα με την ακρίβεια και τη σταθερότητα που επιθυμούσαμε να έχουμε.

Η επιφάνεια εργασίας αποτελείται συνολικά απο 12 επιτρεπόμενες θέσεις. Σε κάθε θέση αντιστοιχεί και ένα σενάριο κατά το οποίο δίνονται σε σειρά οι θέσεις και η ταχύτητες που θα κινηθούν τα 6 σερβομοτέρ του ρομποτικού βραχίονα (6 βαθμοί ελευθερίας). Αυτή η λειτουργία υλοποιείται με την εντολή:

```
file = open('/media/pi/KRD/ImgProcAndRoboticArm_v0.6.4/coords/' + Pos + '.txt', 'r')
file1 = open('/media/pi/KRD/ImgProcAndRoboticArm_v0.6.4/coords/dpos.txt', 'r')
```



Σχήμα 44: Σενάριο κίνησης βραχίονα A1 όπου ID: 1 είναι το σερβομοτέρ No 1, P520 είναι η θέση περιστροφής του σερβομοτέρ με επιτρεπόμενο εύρος τιμών 0< x <1000 (εδώ η θέση περιστροφής 520 αντιστοιχεί σε γωνία 70^0) και T500 η ταχύτητα που θα κινηθεί το σερβομοτέρ για να φτάσει στην επιθυμητή θέση με εύρος 0< x <1000 (εδώ η τιμή 500 αντιστοιχεί σε ταχύτητα 10cm/sec).

8) Καλείται το αρχείο Arm Movement με την εντολή:

ArmMovement(file)

ArmMovement(file1)

το οποίο συνδέεται με το αρχείο PiArm και υλοποιεί τα σενάρια κίνησης του βραχίονα που έχουν επιλεχθεί εκτελώντας τις κινήσεις του βραχίονα προκειμένου να εκτελέσουμε την εργασία που του έχουμε ορίσει (Σχήμα 45).

zipfile.py | ArmMov.py | piarm.py

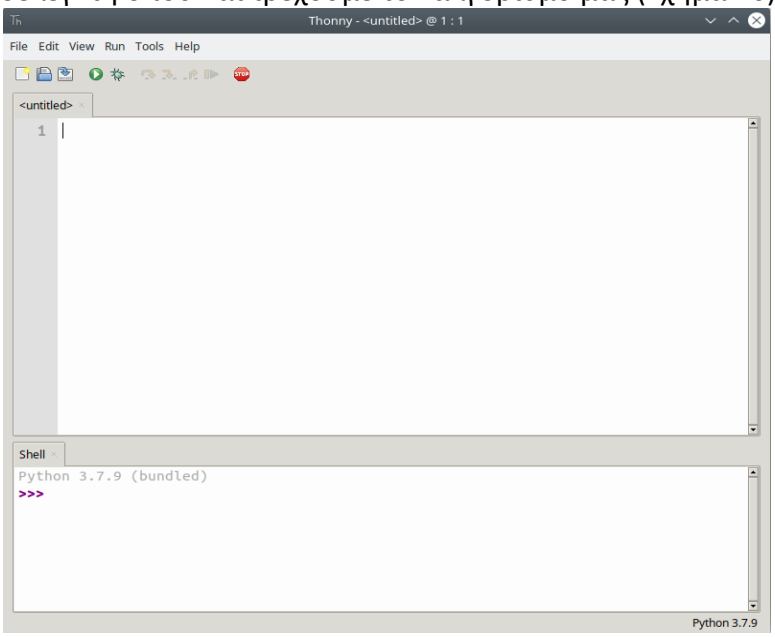
```
1  from piarm import PiArm
2  from time import sleep
3  import re
4  robo = PiArm()
5  robo.connect('/dev/ttyS0')
6
7
8
9  def ArmMovement(file):
10     command_data = []
11     for line in file:
12         command_data.append(line.strip())
13
14     size = len(command_data)
15     print(size)
16
17     for command in range(0,size):
18         raw_Data = command_data[command]
19         raw_Data = raw_Data.split('  ')
20         delay = raw_Data.pop()
21         delay = delay.split(':')
22
23         delay = int(int(delay[1])/1000)
24         print('Go- ', raw_Data, delay)
25
26         for value in range(6):
27             cmd_Data = re.findall('\d+',raw_Data[value])
28             print('Cmd Data- ', cmd_Data)
29             robo.servoWrite(int(cmd_Data[0]), int(cmd_Data[1]), int(cmd_Data[2]))
30
31         sleep(delay)
```

Σχήμα 45: Σενάριο κίνησης του βραχίονα για αντικείμενο στη θέση A1

3.3 Τελικός έλεγχος σε πραγματικές συνθήκες

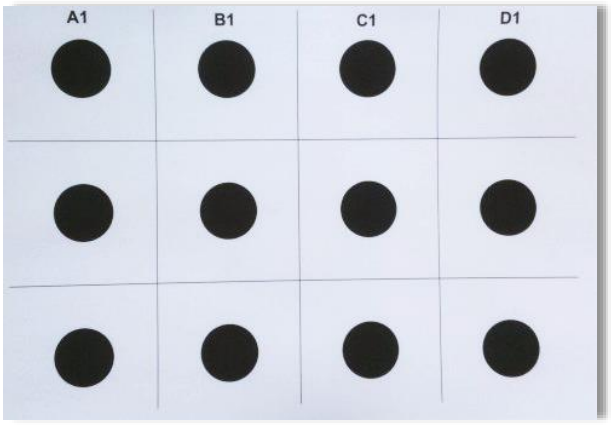
Αρχικά φροντίζουμε να βρίσκονται σε λειτουργία όλες οι διατάξεις εξασφαλίζοντας αδιάλειπτη παροχή ρεύματος και τη κατάλληλη συνδεσμολογία μεταξύ του ρομποτικού βραχίονα, του μικροελεγκτή Raspberry Pi αλλά και των περιφερειακών όπως κάμερα, πληκτρολόγιο και ποντίκι.

Εν συνεχεία ανοίγουμε τον επεξεργαστή κώδικα στην επιφάνεια εργασίας του μικροελεγκτή όπου και τρέχουμε τον αλγόριθμό μας (Σχήμα 46).



Σχήμα 46: Επεξεργαστής κώδικα Thonny σε λειτουργικό λογισμικό Linux

Η αρχική επιφάνεια εργασίας πρέπει να είναι άδεια προκειμένου να ληφθεί η αρχική φωτογραφία (Σχήμα 47).

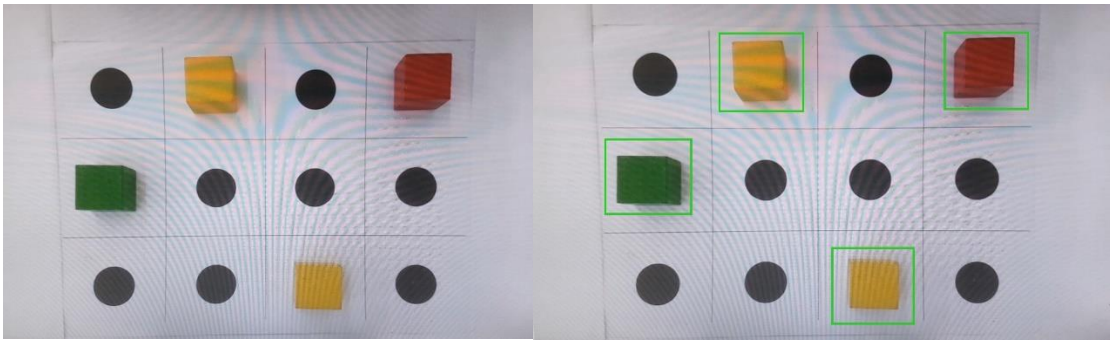


Σχήμα 47: Κενή επιφάνεια εργασίας με 12 πιθανές θέσεις αντικειμένων όπου για λόγους καλύτερης κατανόησης του αλγόριθμου και αντιστοίχισης των σεναρίων κίνησης του βραχίονα με τις θέσεις , έχουμε ονομάσει κάθε στήλη (A,B,C,D) και αριθμήσει κάθε γραμμή (1,2,3). Έτσι έχουμε 12 θέσεις με 12 σενάρια κίνησης του βραχίονα, ένα για κάθε θέση.

Τοποθετούμε τα αντικείμενα εν προκειμένω κύβους διαστάσεων 300mm x 300mm σε τυχαίες θέσεις στα 12 τετράγωνα. Εν συνεχεία θα κάνουμε λήψη μίας δεύτερης

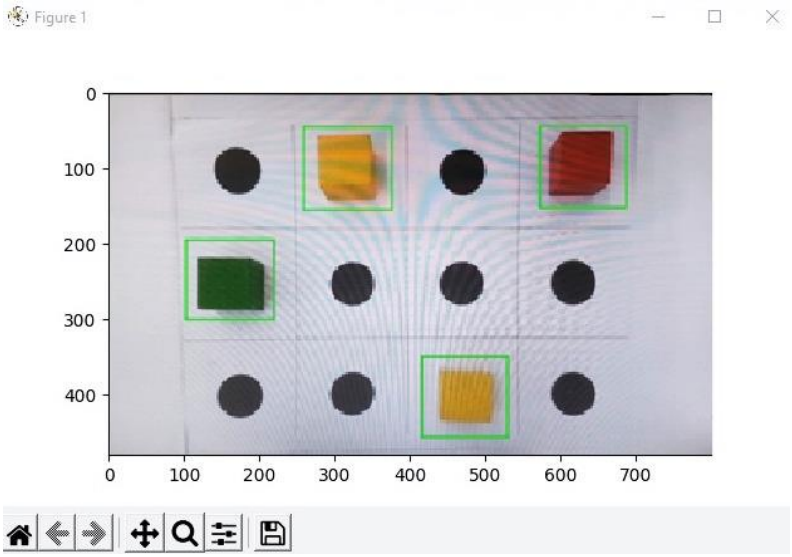
φωτογραφίας όπου θα φαίνεται η θέση και το πλήθος των αντικειμένων επάνω στην επιφάνεια εργασίας (Σχήμα 48α). Ο αλγόριθμός αφού εκτελέσει μία σειρά τεχνικών οπτικής επεξεργασίας των εικόνων συνεχίζει στην σχεδίαση των περιγραμμάτων των αντικειμένων και υπολογίζει τις συντεταγμένες τους.

Έτσι η δεύτερη φωτογραφία αυτή θα συσχετιστεί αλγοριθμικά με την αρχική κενή φωτογραφία προκειμένου να υπολογιστούν τα περιγράμματα (Σχήμα 48β) και τέλος οι συντεταγμένες των αντικειμένων (Σχήμα 49).



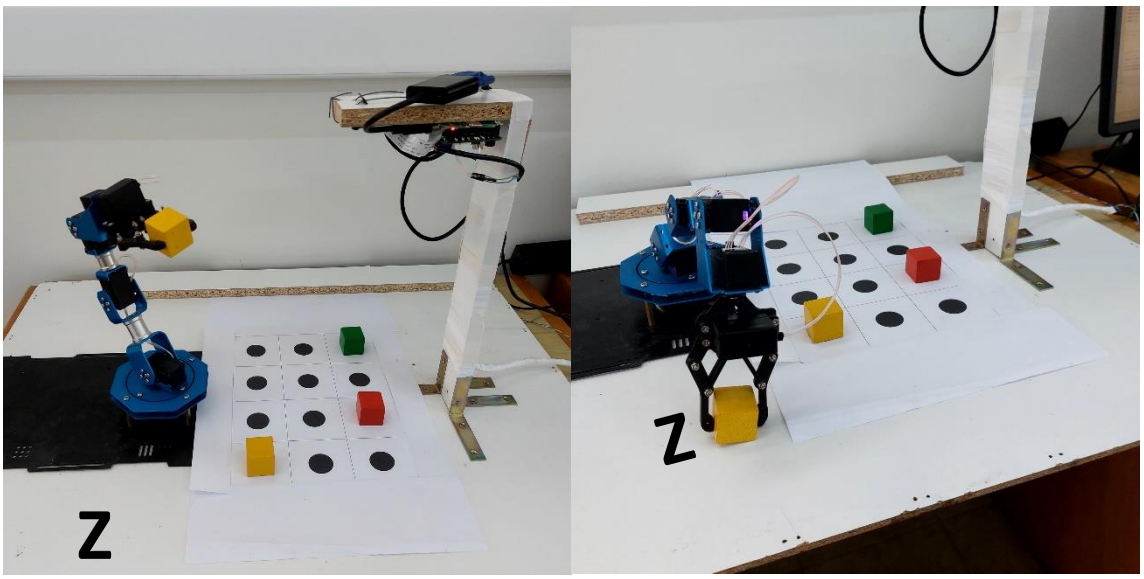
(α) (β)

Σχήμα 48: α) Επιφάνεια εργασίας με τέσσερις κύβους τοποθετημένους στις θέσεις A2,B1, C3,D1, β) Αυτόματη σχεδίαση των περιγραμμάτων των τεσσάρων αντικειμένων.



Σχήμα 49: Υπολογισμός των συντεταγμένων των αντικειμένων στις θέσεις A2,B1, C3,D1

Στη συνέχεια ο αλγόριθμος κάνει κλήση των συνδεδεμένων με τις θέσεις που βρίσκονται τα αντικείμενα σεναρίων τα οποία καθοδηγούν το βραχίονα στην υλοποίηση των προκαθορισμένων κινήσεων του έως ότου να μετακινηθούν όλα τα αντικείμενα από τις αρχικές θέσεις στη θέση **Z**. Η υλοποίηση ενός τέτοιου σεναρίου φαίνεται στα Σχήματα 50α, 50β.



(α) (β)

Σχήμα 50: α) Γενική εικόνα της ρομποτικής διάταξης όπου φαίνεται η υλοποίηση ενός σεναρίου κίνησης του ρομποτικού βραχίονα για την μετακίνηση του αντικειμένου, β) Εκτελώντας τις τελευταίες κινήσεις του σεναρίου κίνησης, ο ρομποτικός βραχίονας εναποθέτει το αντικείμενο (κύβος) στην επιθυμητή θέση Z.

Τελικώς ο βραχίονας επανέρχεται στην αρχική του θέση και ο αλγόριθμος τερματίζεται κλείνοντας όλα τα ενεργά παράθυρα.

3.4 Τελικά συμπεράσματα και προτάσεις βελτιστοποίησης

Η σύνθεση του αλγορίθμου εντοπισμού και χωροθέτησης μικρών αντικειμένων και η υλοποίησή του σε κώδικα στη γλώσσα Python έγινε αξιοποιώντας βιβλιοθήκες εντολών και διαδικασιών οπτικής αναγνώριση και επεξεργασίας που υπάρχουν διαθέσιμες στο διαδίκτυο. Σε επόμενα στάδια εξέλιξης θα μπορούσε να ενσωματωθεί τεχνητή νοημοσύνη A.I. προκειμένου να βελτιστοποιηθούν τα σενάρια εργασιών και να αυτοματοποιηθεί η οπτική αναγνώριση για εντοπισμό αντικειμένων στο χώρο σε οποιαδήποτε θέση.

Επιπλέον βήματα εξέλιξης τα οποία θα μπορούσαν γίνουν και τα οποία υλοποιήθηκαν σε ένα βαθμό ήταν η αρχική προσπάθεια για την προσθήκη γραφικού περιβάλλοντος χρήστη και η διόρθωση της παραμόρφωσης της ληφθείσας εικόνας.

3.4.1 Προσθήκη γραφικού περιβάλλοντος χρήστη

Σαν μελλοντικό βήμα αυτής της διαδικασίας μας αναμένεται να είναι η οπτική απεικόνιση του αλγορίθμου μας με προσθήκη ενός γραφικού περιβάλλοντος χρήστη.

Το γραφικό περιβάλλον χρήστη (*Graphical User Interface*, *GUI*, Σχήμα 51), το οποίο είναι ένα σύνολο εικονικών μενού, παραθύρων, κουμπιών, κουτιών και εικονιδίων που βελτιώνουν τη χρηστικότητα του ψηφιακού περιβάλλοντος που ελέγχει το ρομποτικό μας βραχίονα. Διευκολύνουν επίσης και επιταχύνουν την αλληλεπίδραση μεταξύ του χρήστη και

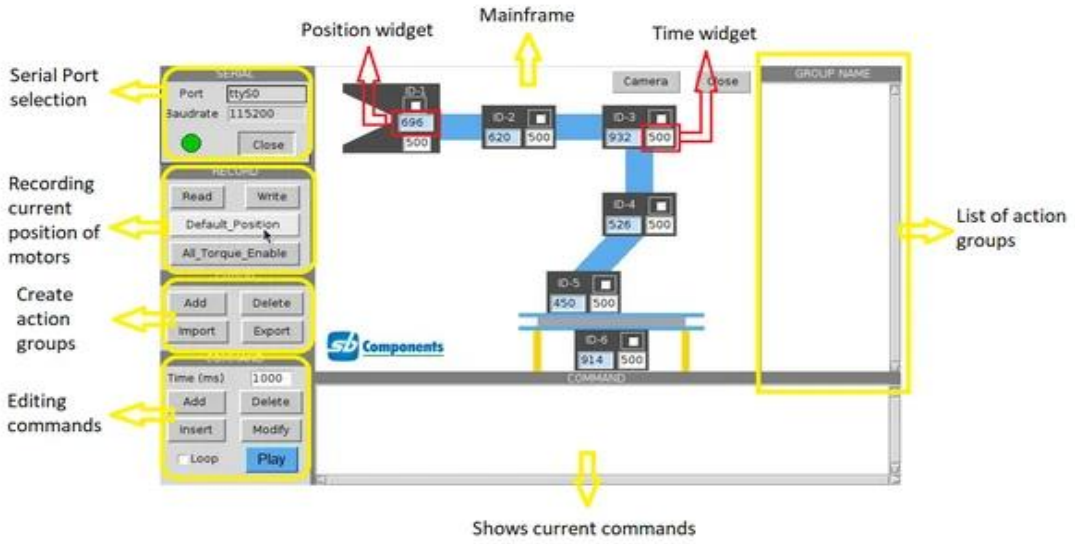
της ρομποτικής διάταξης παρέχοντας στον τελευταίο, μέσω εικόνων, ενδείξεις και εργαλεία προκειμένου αυτός, με απλές ενέργειες, να εκτελέσει συγκεκριμένες εργασίες.

Ένα παράδειγμα τέτοιας εγκατάστασης έτοιμου γραφικού περιβάλλοντος χρήστη GUI για το ρομποτικό βραχίονα έγινε με χρήση ελεύθερου λογισμικού που λήφθηκε από τη βιβλιοθήκη GitHub (αρχείο PiArm.git).[40]

Η υλοποίησή του έγινε με τη παρακάτω εντολή που εισήχθη στη γραμμή εντολών του μικροϋπολογιστή Raspberry Pi3 B+.

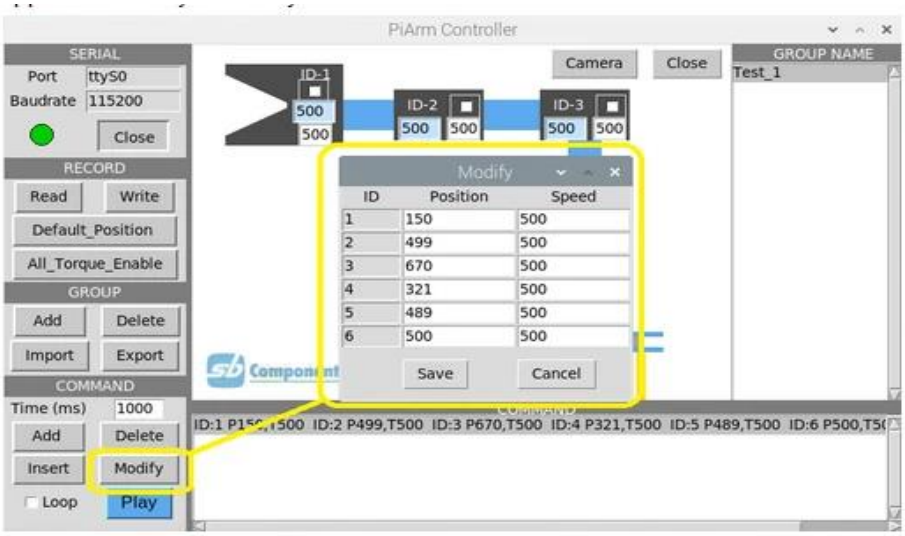
git clone https://github.com/sbcshop/PiArm.git

Το αποτέλεσμα αυτής της ενέργειας ήταν να δημιουργηθεί ένας φάκελος με την ονομασία ''Pi'' εντός του οποίου αποθηκεύτηκε η εφαρμογή PiArm control (το γραφικό περιβάλλον χρήστη που ελέγχει το ρομποτικό βραχίονα αποκλειστικά).



Σχήμα 51 : GUI της εταιρίας Sb Components που χρησιμοποιήθηκε για τον έλεγχο του ρομποτικού βραχίονα. Η επικοινωνία μεταξύ του σερβομοτέρ και του μικροελεγκτής έγινε σειριακά με τη χρήση ενός καλωδίου (εδώ επιλέχθηκε η σειριακή θύρα ttyS0 για την σύνδεση)

Μερικές από τις δυνατότητες του GUI είναι η καταγραφή των θέσεων των σερβομοτέρ και η αλλαγή της θέσεως τους όπως επίσης και της ταχύτητας που θα κινηθούν αυτά (Σχήμα 52)



Σχήμα 52: GUI της SbComponents – επιλογή επεξεργασίας της θέσης και της ταχύτητας κίνησης των σερβομοτέρ.

Το λογισμικό δίνει τη δυνατότητα να προστεθεί για κάθε μοτέρ ένα σενάριο (γκρουπ εντολών) που θα τρέχει σε προκαθορισμένη σειρά και θα καθορίζει τις ενδιάμεσες θέσεις και τις τελικές θέσεις των σερβομοτέρ και κατ' επέκταση του ρομποτικού βραχίονα. Επίσης μπορούν να χρησιμοποιηθούν έτοιμα σενάρια ή να γίνει καταγραφή των επιθυμητών θέσεων του βραχίονα και να αποθηκευτεί σαν σενάριο.

Επιπρόσθετα θα μπορούσε να χρησιμοποιηθεί κάποια φορετή διάταξη για τον έλεγχο των κινήσεων και των λειτουργιών της πειραματικής διάταξης ακόμα και εξ'αποστάσεως.

Εναλλακτικός τρόπος με τον οποίο μπορεί να ελεγχθεί ο ρομποτικός μας βραχίονας είναι με τη χρήση ενός εμπορικού χειριστηρίου (joystick) κονσόλας παιχνιδιών (Σχήμα 53). Αντίστοιχα η εγκατάσταση του χειριστηρίου μπορεί να γίνει με τη χρήση της εντολής : *pip install pygame* η οποία εγκαθιστά τη βιβλιοθήκη Pygame.

Επίσης μπορεί να γίνει αναβάθμιση και προεγκατάσταση ταυτόχρονα του αρχείου Pygame με την εντολή : *sudo pip3 install –upgrade pygame*

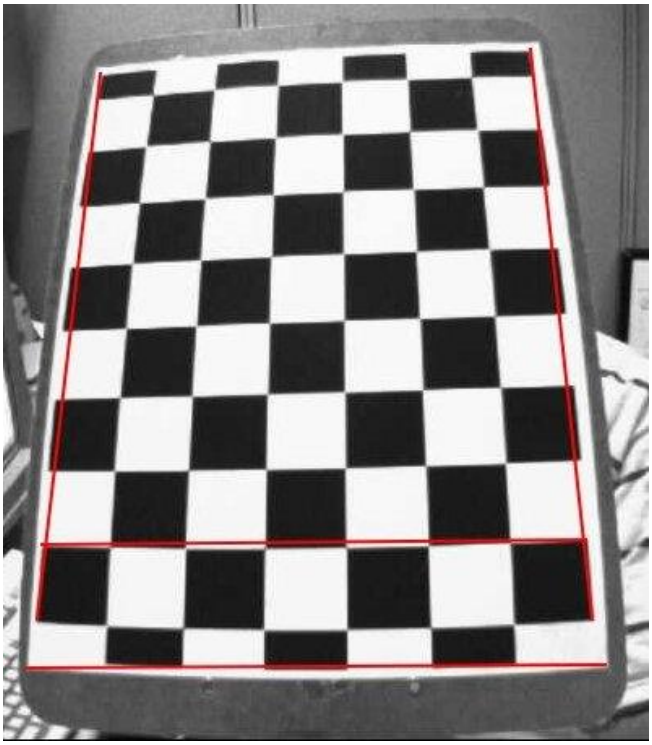
Σχήμα 53: Εμπορικά χειριστήρια κατάλληλα για οδήγηση ρομποτικών διατάξεων.

Τα περισσότερα χειριστήρια έχουν ένα σύμβολο σε κάθε κουμπί. Αυτά τα σύμβολα χρησιμοποιούνται για τον προσδιορισμό του πλήκτρου που πιέζεται. Τα αριστερά πλήκτρα των μοχλών χειρισμού αντιστοιχούν στην επιλογή του άξονα ενώ τα υπόλοιπα πλήκτρα έχουν

αριθμητική τιμή. Οι τιμές των πλήκτρων ενδέχεται να διαφέρουν ανάλογα με τα χειριστήρια διαφορετικών κατασκευαστών. Μία σχετική βιβλιοθήκη η Pygame χρησιμοποιείται για τη διαχείριση μιάς μονάδας χειριστηρίου σε υπολογιστή[40](Παράρτημα).

3.4.2 Διόρθωση της παραμόρφωσης της ληφθείσας εικόνας από την κάμερα.

Ένα από τα μεγαλύτερα προβλήματα των σύγχρονων οικονομικών καμερών είναι η παραμόρφωση της εικόνας που προκαλούν. Υπάρχουν δύο είδη παραμόρφωσης η ακτινική και η εφαπτομενική παραμόρφωση. Λόγω της ακτινικής παραμόρφωσης οι ευθείες γραμμές εμφανίζονται σαν καμπύλες. Το πρόβλημα διογκώνεται καθώς απομακρυνόμαστε από το κέντρο της εικόνας. Για παράδειγμα, εμφανίζεται μια εικόνα παρακάτω (Σχήμα 54), όπου δύο άκρα μιας σκακιέρας σημειώνονται με κόκκινες γραμμές. Όπως μπορούμε να δούμε το περίγραμμα δεν είναι ευθεία και δεν ταιριάζει με την κόκκινη γραμμή. Όλες οι αναμενόμενες ευθείες γραμμές είναι διογκωμένες.



Σχήμα 54: Ακτινική παραμόρφωση εικόνας όπου σημειώνονται τα δύο άκρα της με κόκκινες γραμμές.

Αυτή η παραμόρφωση επιλύεται μαθηματικά ως εξής:

$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$

$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$

Παρομοίως, μια άλλη παραμόρφωση είναι η εφαπτομενική παραμόρφωση που συμβαίνει επειδή ο φακός λήψης της εικόνας δεν ευθυγραμμίζεται απόλυτα παράλληλα με το επίπεδο απεικόνισης. Έτσι, ορισμένες περιοχές στην εικόνα μπορεί να φαίνονται πιο κοντά από το αναμενόμενο.

Αυτή η παραμόρφωση επιλύεται ως εξής:

$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$

$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$

Εν ολίγοις, πρέπει να βρούμε πέντε παραμέτρους, γνωστές ως συντελεστές παραμόρφωσης που δίνονται από τους:

Συντελεστές Παραμόρφωσης = $(k_1\ k_2\ p_1\ p_2\ k3)$

Εκτός από αυτό, πρέπει να βρούμε μερικές ακόμη πληροφορίες, όπως οι εσωτερικές και εξωτερικές παράμετροι της κάμερας. Οι εσωτερικές παράμετροι είναι συγκεκριμένες για μια κάμερα και περιλαμβάνουν πληροφορίες όπως το εστιακό μήκος (f_x, f_y) εκφρασμένο σε μονάδες pixel και τα οπτικά κέντρα (c_x,c_y) που βρίσκονται συνήθως στο κέντρο του αντικειμένου. Αυτά απεικονίζονται σε ένα πίνακα που ονομάζεται πίνακας κάμερας. Αυτός εξαρτάται μόνο από την κάμερα, οπότε μόλις υπολογιστεί, μπορεί να αποθηκευτεί για μελλοντικούς σκοπούς. Εκφράζεται ως μήτρα 3x3:

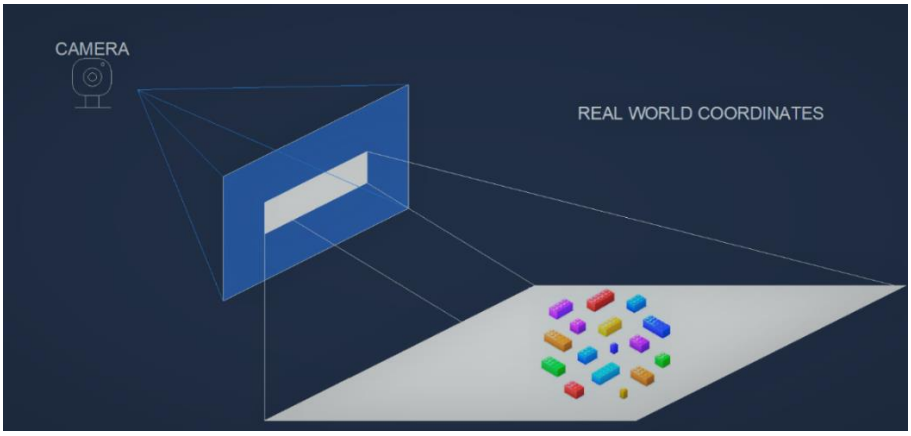
$$\boldsymbol{\pi ίνακας\ κάμερας} = \begin{vmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{vmatrix}$$

Οι εξωγενείς παράμετροι αντιστοιχούν σε διανύσματα περιστροφής και μετασχηματισμού που μετατρέπουν τις συντεταγμένες ενός τρισδιάστατου 3D σημείου σε ένα σύστημα συντεταγμένων.

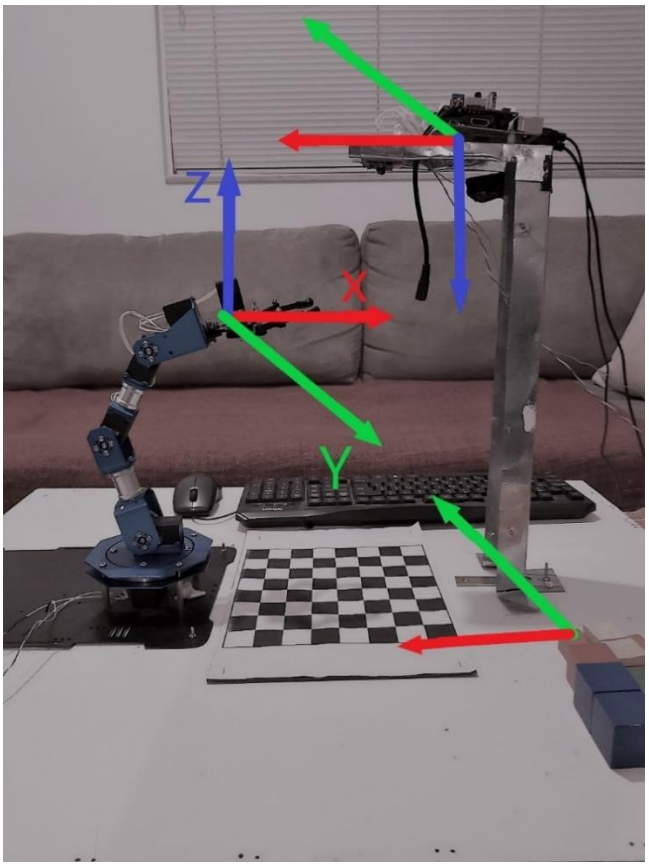
Για δισδιάστατες εφαρμογές, αυτές οι παραμορφώσεις πρέπει πρώτα να διορθωθούν. Για να βρεθούν όλες αυτές οι παράμετροι πρέπει να φορτωθούν δείγματα εικόνων ενός καλά καθορισμένου μοτίβου (π.χ. σκακιέρα). Να βρεθούν κάποια συγκεκριμένα σημεία σε αυτό (τετράγωνες γωνίες στη σκακιέρα). Να γνωρίζουμε τις συντεταγμένες στον πραγματικό χώρο και τις συντεταγμένες του στην εικόνα. Με αυτά τα δεδομένα, αυτό το μαθηματικό πρόβλημα επιλύεται στο παρασκήνιο για τη λήψη των συντελεστών παραμόρφωσης. Ενδεικτικά χρειάζονται τουλάχιστον 10 δείγματα δοκιμών [41,42].

3.4.3 Διαμόρφωση του αλγορίθμου για εντοπισμό αντικειμένων στο χώρο

Κατά την ενσωμάτωση της οπτικής αναγνώρισης αντικειμένων με χρήση κάμερας στο ρομποτικό βραχίονα θεωρήθηκε ότι τα αντικείμενα τα οποία είναι συγκεκριμένων διαστάσεων βρίσκονται όλα σε συγκεκριμένη στάθμη αυτή του πάγκου εργασίας και άρα η τιμή του άξονα Z στον τρισδιάστατο χώρο παίρνει τιμή 0. Στην περίπτωση που θα θέλαμε να υλοποιήσουμε εντοπισμό αντικειμένων σε άλλη στάθμη θα έπρεπε να τοποθετηθεί και δεύτερη κάμερα διαφορετικά να διαμορφωθεί ο αλγόριθμος ώστε να διαβάζει την εικόνα που λαμβάνεται με τρόπο που να αξιολογεί τις αλλαγές στον άξονα ύψους Z (Σχήμα 56).



Σχήμα 55: Αναπαράσταση της μετατροπής των συντεταγμένων της εικόνας σε συντεταγμένες του πραγματικού κόσμου.



Σχήμα 56: Πλαίσια συντεταγμένων ρομπότ, κάμερας και σχεδίου. Το κόκκινο είναι ο άξονας X, το πράσινος ο άξονας Y και το μπλε ο άξονας Z, τα βέλη δείχνουν προς την κατεύθυνση των θετικών αυξήσεων. Αυτή η απεικόνιση είναι σημαντική για την κατανόηση του κώδικα και του τρόπου με τον οποίο μπορούμε να τον προσαρμόσουμε. Ενδέχεται να διαμορφώνονται διαφορετικά πλαίσια συντεταγμένων για άλλες εφαρμογές του βραχίονα (Παράρτημα).

3.4.4 Σύνοψη - Συμπεράσματα

Διαφαίνεται ότι το πεδίο των ρομποτικών βραχιόνων που καθόρισε τη βιομηχανική παραγωγή και πλέον ενσωματώνεται σε ολοένα και περισσότερους τομείς θα διαδραματίσει βασικό ρόλο στις ζωές των ανθρώπων.

Στα πλαίσια αυτής της προσέγγισης η παρούσα εργασία έθεσε σαν στόχο την ανάπτυξη ενός οικονομικού ρομποτικού μοντέλου βραχίονα που θα μπορεί να συνδυάζει την οπτική αναγνώριση κάνοντας χρήση ανοιχτών λογισμικών για την πραγματοποίηση εργασιών όπως η μετακίνηση αντικειμένων από ένα χώρο σε κάποιο σημείο.

Η συγκεκριμένη εργασία επεκτείνεται σε τομείς όπως η ηλεκτρονική, η μηχανολογία και ο προγραμματισμός και συνεπώς απαιτούνται συνδυαστικές γνώσεις και από τα τρία αυτά πεδία.

Μετά από έρευνα και πολλές δοκιμές βγήκε το συμπέρασμα ότι ήταν απαραίτητο να δημιουργηθεί ένας καινούργιος κώδικας ο οποίος θα αναλάμβανε να εξυπηρετήσει τους στόχους που είχαμε θέσει για το ρομποτικό μας βραχίονα.

Στα πλαίσια τη συγκεκριμένης εργασίας αντιμετωπίστηκαν επιτυχώς αρκετές τεχνικές προκλήσεις όπως ο έλεγχος των ανοχών των συνδέσμων του βραχίονα για να πετύχουμε υψηλή επαναληψιμότητα, η στόχευση του οπτικού πεδίου της κάμερας στον ωφέλιμο χώρο του μπροστινού μέρους του βραχίονα, η σωστή ρύθμιση όλης της κατασκευής.

Ήταν πολύ μεγάλη πρόκληση η ανάπτυξη τεχνογνωσίας για την χρήση του ρομποτικού βραχίονα και της καθοδήγησή του από την κάμερα με τη χρήση ενός εμπορικού μικροελεγκτή. Εν προκειμένω αν και επιχειρήθηκε αρχικά η χρήση έτοιμων αλγορίθμων από το διαδίκτυο αποδείχθηκε ότι αυτοί δεν ήταν λειτουργικοί και χρησίμευσαν περισσότερο ως συμπλήρωμα στις υπάρχουσες γνώσεις μας.

Αφού ολοκληρώθηκε με επιτυχία το πρώτο πείραμα που είχαμε ορίσει αρχικά όπου θα γινόταν ο εντοπισμός και η μετακίνηση ενός αντικειμένου από την επιφάνεια εργασίας μας σε κάποιο σημείο, περάσαμε στη δεύτερη φάση όπου θα μπορούσαν να εντοπιστούν πολλαπλά αντικείμενα και να μετακινηθούν ταυτόχρονα κάτι που επίσης ολοκληρώθηκε με επιτυχία.

Σαν μελλοντική εξέλιξη της εργασίας αυτής μπορούμε να ορίσουμε τον εντοπισμό αντικειμένων διαφορετικών διαστάσεων, χρωμάτων και υλικών και τη τοποθέτησή τους σε κατάλληλες θέσεις ανάλογα με τα επιθυμητά κριτήρια.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

[1]. International Federation of Robotics (IFR), Press Releases, <https://ifr.org/ifr-press-releases/news/china-leads-post-pandemic-recovery>, 01.07.2021 (accessed 01.04.22)

[2]. Jacob Heffernan IPT project, <http://iptmajorprojectjacobheffernan.weebly.com/history-of-the-robotic-arm.html>, (accessed 01.04.22)

[3]. Michael E. Moran, Evolution of robotic arms, Journal of Robotic Surgery volume 1, Published online 1 May 2007 (2007) 103–111, doi:10.1007/s11701-006-0002-x

[4]. International Federation of Robotics (IFR), History of Robotic Systems, <https:// ifr.org/robot-history>, (accessed 01.03.22)

[5]. Online Encyclopedia hosted by the Wikimedia Foundation, Post regarding Unmanned aerial vehicle, <https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle>, (accessed 01.03.22)

[6]. Vertical Technologies, product DeltaQuad Pro VIEW VTOL Fixed wing surveillance UAV, <https://www.deltaquad.com/vtol-drones/view/>, (accessed 01.04.22)

[7]. Tice, Brian P, Unmanned Aerial Vehicles – The Force Multiplier of the 1990s, Airpower Journal, (Spring 1991)

[8]. Franke, Ulrike Esther, Civilian Drones: Fixing an Image Problem, ISN Blog. International Relations and Security Network. Retrieved 5 March 2015, 26 January 2015.

[9]. ICAO, ICAO's circular 328 AN/190 : Unmanned Aircraft Systems (UAS), 3 February 2016

[10]. K. Alexis, C. Papachristos, Drones Demystified, Autonomous Robots Lab, University of Nevada, RenoA. Tzes, Autonomous Robots & Intelligent Systems Lab, NYU Abu Dhabi (2016)

[11]. Fernando Alfredo Auat Cheein; Ricardo Carelli, Agricultural Robotics Unmanned Robotic Service Units in Agricultural Tasks, IEEE Industrial Electronics Magazine, Volume: 7 Issue: 3, SEPTEMBER 2013

[12]. D. Akin, M. Minsky, E. Thiel, and C. Kurtzman, Space applications of automation, robotics and machine intelligence systems (ARAMIS), phase II, Rep. NASA CR-3735 (1983)

[13]. Kazuya Yoshida, Achievements in space robotics, IEEE Robotics & Automation Magazine, Volume: 16 , Issue: 4 , December 2009, 20-28

[14]. Speich, J. and Rosen, J., Medical Robotics. In: G. Wnek and G. Bowlin (Eds), Encyclopedia of Biomaterials and Biomedical Engineering, New York, USA: Marcel Dekker (2004) 983–993

[15]. Iversen E., Sears H.H. and Jacobsen S.C., Artificial Arms Evolve from Robots or Vice Versa, IEEE Control Systems Magazine 25 (2005) 16–20

[16]. Najarian S, Fallahnezhad M, Afshari E, Advances in medical robotic systems with specific applications in surgery—a review, Pages 19-33, Received 25 Aug 2010, Accepted 22 Oct 2010, Published online: 09 Dec 2010, doi:10.3109/03091902.2010.535593

[17]. Makaran J., Dittmer D., Buchal R. and MacArthur D, The SMART(R) wrist-hand orthosis (WHO) for quadriplegic patients, JPO Journal of Prosthetics and Orthotics 5 (1993) 73–76, doi:10.1097/00008526-199307000-00002

[18]. Krebs H.I., Volpe B.T., Williams D., Celestino J., Charles S.K., Lynch D. And Hogan N., Robot-Aided Neurorehabilitation: A Robot for Wrist Rehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering 15 (1993) 327–335

[19]. Rosati G., Gallina P. and Masiero S., Design, Implementation and Clinical Tests of a Wire-Based Robot for Neurorehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering 15 (2007) 560–569

[20]. Howe R.D. and Matsuoka Y., Robotics for surgery, Annu Rev Biomed Eng. 1 (1999) 211–240

[21]. Varkarakis I.M., Rais-Bahrami S., Kavoussi L.R. and Stoianovici D., Robotic surgery and telesurgery in urology. Urology 65 (2005) 840–846

[22]. Villorte N., Glauser D., Flury P. and Burckhardt C.W., Conception of stereotactic instruments for the neurosurgical robot Minerva, 14th Annual International Conference of the IEEE Engineers in Medicine and Biology Society (1992)

[23]. Burckhardt C.W., Flury P. and Glauser D., Stereotactic brain surgery. IEEE Engineering in Medicine and Biology 14 (1995) 314– 317

[24]. Eljamel M.S., Validation of the PathFinder neurosurgical robot using a phantom. Int. J. of Med Robotics Comput Assist Surg. (2007) 372–377

[25]. John McDougall, Larry B. Lessard and Vincent Hayward, applications of advanced materials to robotic design: The freedom-7 haptic hand controller, McGill University, Canada, Proceedings of ICCM–11, Gold Coast, Australia, 14th-18th July 1997

[26]. Γιάννης Σ. Μπουτάλης, Ρομποτική ανάλυση, έλεγχος και προγραμματισμός ρομποτικών χειριστών σταθερής βάσης, 2015.

[27]. Ζωή Δούλγερη, Ρομποτική κινηματική, δυναμική και έλεγχος αρθρωτών βραχιόνων, Αθήνα εκδόσεις Κριτική, 2007.

[28]. Isak Karabegović, Edina Karabegović, Mehmed Mahmić and Ermin Husak, Implementation of Industry 4.0 and Industrial Robots in the Manufacturing Processes, Technical Faculty Bihać, University of Bihać, Ul Irfana Ljubujankića bb, 77000 Bihać, Bosnia and Herzegovina, Received in July 2020, Revised in August 2020, Accepted in October 2020, doi: 10.24874/mvm.2021.47.02.02

[29]. Συντακτική Ομάδα της εταιρίας EMEA Media, <https://emea.gr/epilogi-syntakton/611950/se-poia-chora-kyriarchoyn-ta-viomichanika-rompot-ergates/>, 11/2020 (accessed 01.04.2022)

[30]. Link Labs, Written by Brian Ray, <https://www.link-labs.com/blog/what-is-m2m>, 14 February 2018 (accessed 01.04.2022)

[31]. Singh B, Sellappan N, Kumaradhas P., Evolution of industrial robots and their applications, International Journal of Emerging Technology and Advanced Engineering, Mechanical Engineering Section, Salalah College of Technology, Salalah, Sultanate of Oman, Certified Journal,Volume 3, Issue 5, May 2013

[32]. Abidhusain Syed, Zamrrud Taj H. Agasbal, Thimmannagouday Melligeri, Bheemesh Gudur, Flex Sensor Based Robotic Arm Controller Using Micro Controller, Department of Electronics and Communication, BLDEA College of Engg Bijapur-3, India; Department of Electronics and Communication, KBN College of Engg Gulbarga-4, India, Received September 6th, 2011; revised March 10th, 2012; accepted April 3rd, 2012.

[33]. Online Encyclopedia hosted by the Wikimedia Foundation, Post regarding transducer, <https://en.wikipedia.org/wiki/Transducer>, (accessed 01.04.2022)

[34]. SB Components UK Company, <https://shop.sb-components.co.uk/collections/robotic-arm/products/piarm-the-diy-robotic-arm-for-raspberry-pi>, (accessed 01.04.22)

[35]. MathWorks Inc, <https://www.mathworks.com/help/supportpkg/raspberrypiio/ref/working-with-raspberry-pi-hardware.html>, (accessed 01.04.22)

[36]. Online Encyclopedia hosted by the Wikimedia Foundation, Post regarding RealVNC, <https://en.wikipedia.org/wiki/RealVNC>, (accessed 01.04.2022)

[37]. Online Encyclopedia hosted by the Wikimedia Foundation, Post regarding Nmap, <https://el.wikipedia.org/wiki/Nmap>, (accessed 01.04.2022)

[38]. Online Encyclopedia hosted by the Wikimedia Foundation, Post regarding OpenCV, <https://opencv.org>, (accessed 01.04.2022)

[39]. OpenCV, Python tutorials <https://docs.opencv.org/master/d2/de6/tutorial_py_setup_in_ubuntu.html>, (accessed 01.04.22)

[40]. Open source developer platform, sbcshop project, <https://github.com/sbcshop/PiArm>, (accessed 01.04.2022)

[41]. FDXlabs Texas corporation, <https://www.fdxlabs.com/calculate-x-y-z-real-world-coordinates-from-a-single-camera-using-opencv/> (accessed 01.04.22)

[42]. Francisco Garcia, https://github.com/pacogarcia3/hta0-horizontal-robot-arm (accessed 01.04.22)

ΠΑΡΑΡΤΗΜΑΤΑ

Αλγόριθμος Ρομποτικού Βραχίονα «ImgProcAndRoboticArm_v0.7.3»

```
import cv2
import numpy
from cv2 import *
from numpy import diff
import picamera
from ObjectsPositions import ObjPositions
from PiArmDataServoWrite import PiArmPositionObjectMovement
import matplotlib.pyplot as plt
import os
import sys

#python -m pip install -U matplotlib
#pip install opencv-contrib-python==3.4.13.47
#pip install numpy

###SETUP Pi Camera######################
#create object for PiCamera class
VidCap = picamera.PiCamera()
#set resolution
VidCap.resolution = (800, 480)
VidCap.brightness = 60

# Exit if video not opened.
if not VidCap.start_preview():
   print("Could not open Camera")
   sys.exit()

#####READ IMAGE#################################
```

```
#PhotoOfTheBackgroundEmpty = cv2.imread('Photo_Empty_Background.jpg') #photo of Background

while True:
   input("Press Enter to take photo without objects ...")
   VidCap.start_preview()
   VidCap.capture('Photo_Empty_Background.jpg')
   VidCap.stop_preview()
   PhotoOfTheBackgroundEmpty = cv2.imread('Photo_Empty_Background.jpg') #photo of Background
   cv2.imshow('___', PhotoOfTheBackgroundEmpty)
   cv2.waitKey(2000)
   cv2.destroyAllWindows()

   input("Press Enter to take photo with objects ...")
   VidCap.start_preview()
   VidCap.capture('PhotoWObjects.jpg')
   VidCap.stop_preview()
   PhotoWithObjects=cv2.imread('PhotoWObjects.jpg')
   cv2.imshow('___', PhotoWithObjects)
   cv2.waitKey(2000)
   cv2.destroyAllWindows()

   diff= cv2.absdiff(PhotoOfTheBackgroundEmpty,PhotoWithObjects) #briskei tis diafores meta3i twn 2 eikonwn
   cv2.imshow('___', diff)
   cv2.waitKey(2000)
   cv2.destroyAllWindows()

   gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY) #corvert to grayscale
   blur = cv2.GaussianBlur(gray, (5, 5), 0)
   _,thresh = cv2.threshold(blur,20,255,cv2.THRESH_BINARY) #convert 0 1
   dilated = cv2.dilate(thresh,None, iterations=5)
   contours,_ = cv2.findContours(dilated,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

   for c in contours:
      if cv2.contourArea(c) <180 : #for calibrate #poso megala i mikra antikeimena 8a pianei
         continue

      x,y,w,h =cv2.boundingRect(c)
```

```
      if x<=620 :
         cv2.rectangle(PhotoWithObjects, (x,y),(x+w, y+h),(0, 255, 0), 2) #draw rectangle

         print("Positions   x:", x,"y: ", y)
         PositionsOfObjects = ObjPositions(x,y)
         if PositionsOfObjects != '' :
            file=open('/media/pi/288C-
8C2E/ImgProcAndRoboticArm_v0.6.4/coords/'+PositionsOfObjects+'.txt', 'r')
            #ReturnFile                                                    =
open('/home/pi/Desktop/ImgProcAndRoboticArm_v0.6.4/coords/Return.txt', 'r')
            PiArmPositionObjectMovement(file) #trabaei to antistoixo arxeio me tis kineiseis tou
xeriou
           # PiArmPositionObjectMovement(ReturnFile)#epistrefei to xeri stin arxiki 8esi
         else :
            print("It looks something goes wrong...")
            plt.imshow(PhotoWithObjects)
            plt.show()
            sys.exit()
         #PiArmPositionObjectMovement(file) #trabaei to antistoixo arxeio me tis kineiseis tou
xeriou
         #PiArmPositionObjectMovement(ReturnFile)
   plt.imshow(PhotoWithObjects)
   plt.show()
   cv2.imshow('___', PhotoWithObjects)
   cv2.waitKey(5000)
   cv2.destroyAllWindows()
   sys.exit()
```

Συνοδευτικός αλγόριθμος εύρεσης διαστημάτων θέσεων αντικειμένων «ObjectsPositions»

```
def ObjPositions(x,y) :
    CoordsNameFile= ''
    if (x>=100 and x<=224) and (y>=1 and y<=99) :
        print('A1')
        CoordsNameFile= 'A1'
    elif (x>=100 and x<=224) and (y>=100 and y<=200) :
        print('A2')
        CoordsNameFile= 'A2'
    elif (x>=100 and x<=224) and (y>=201 and y<=310) :
        print('A3')
        CoordsNameFile= 'A3'
    elif (x>=225 and x<=330) and (y>=1 and y<=99) :
        print('B1')
        CoordsNameFile= 'B1'
    elif (x>=225 and x<=330) and (y>=100 and y<=200) :
        print('B2')
        CoordsNameFile= 'B2'
    elif (x>=225 and x<=330) and (y>=201 and y<=310) :
        print('B3')
        CoordsNameFile= 'B3'
    elif (x>=451 and x<=580) and (y>=1 and y<=99) :
        print('C1')
        CoordsNameFile= 'C1'
    elif (x>=451 and x<=580) and (y>=100 and y<=200) :
        print('C2')
        CoordsNameFile= 'C2'
    elif (x>=451 and x<=580) and (y>=201 and y<=390) :
        print('C3')
        CoordsNameFile= 'C3'
    else :
        CoordsNameFile= ''

    return CoordsNameFile
```

Συνοδευτικός αλγόριθμος κίνησης του βραχίονα «PiArmDataServoWrite»

```
from piarm import PiArm
from time import sleep
import re
robo = PiArm()
robo.connect('/dev/ttyS0')

def PiArmPositionObjectMovement(file):
   command_data = []
   for line in file:
      command_data.append(line.strip())

   size = len(command_data)
   print(size)

   for command in range(0,size):
      raw_Data = command_data[command]
      raw_Data = raw_Data.split('  ')
      delay = raw_Data.pop()
      delay = delay.split(':')

      delay = int(int(delay[1])/1000)
      print('Go- ', raw_Data, delay)

      for value in range(6):
         cmd_Data = re.findall('\d+',raw_Data[value])
         print('Cmd Data- ', cmd_Data)
         robo.servoWrite(int(cmd_Data[0]), int(cmd_Data[1]), int(cmd_Data[2]))

      sleep(delay)
```

Προγραμματισμός Χειριστηρίου joystick.

Για να εισάγουμε την ενότητα pygame τρέχουμε στην γραμμή εντολών την εντολή : *import pygame*

Για να αρχικοποιήσουμε την μονάδα χειριστηρίου και να λάβουμε μία λίστα περιπτώσεων χρησιμοποιούμε τον ακόλουθο κώδικα:

pygame.joystick.init()

joysticks = [pygame.joystick.Joystick(x)

for x in range(pygame.joystick.get_count())]

Αφού τρέξουμε τον κώδικα εμφανίζεται το ακόλουθο μήνυμα: JOYAXISMOTION JOYBALLMOTION JOYBUTTONDOWN JOYBUTTONUP JOYHATMOTION

Για να σταματήσουμε να αρχικοποιούμε τη μονάδα του joystick εκτελούμε: *pygame.joystick.quit()*

Μπορούμε να βρούμε τον αριθμό των joystick που είναι συνδεδεμένα με το σύστημα μας χρησιμοποιώντας την εντολή: *pygame.joystick.get_count()*

Για να ελέγξουμε τη θέση του άξονα του joystick χρησιμοποιούμε την get_axis λειτουργία η οποία επιστρέφει μία κυμαινόμενη τιμή μεταξύ του -1 και του 1 με την τιμή 0 να σημαίνει ότι έχει κεντραριστεί: *pygame.joystick.Joystick.get_axis(axis_number)*

Μπορούμε να χρησιμοποιήσουμε το Module του pygame για να εντοπίσουμε τη δραστηριότητα του χρήστη με την εντολή: *pygame.event.get()*

Η οποία επιστρέφει τη δραστηριότητα που εκτελείται από το χρήστη. Αυτά μπορούν να συγκριθούν με τις πιθανές θέσεις του joystick.

event.type == pygame.JOYBUTTONDOWN

Or

event.type == pygame.JOYBUTTONUP

Και έτσι μπορούμε να δούμε τί πλήκτρο πατήθηκε από το χρήστη. Στο χειριστήριο όλα τα πλήκτρα έχουν ένα συγκεκριμένο αριθμό που μπορεί να χρησιμοποιηθεί για να συγκρίνουμε τί πατήθηκε από το χρήστη.

event.button == 1

Υπάρχουν πολλές λειτουργίες και χαρακτηριστικά διαθέσιμα στην pygame.joystick βιβλιοθήκη η οποία μπορεί να χρησιμοποιηθεί για να προσθέσουμε περισσότερα χαρακτηριστικά του Ρομποτικού μας βραχίονα.

Μπορούμε να χρησιμοποιήσουμε το pygame με δύο τρόπους: 1)Αναπαραγωγή της εντολής από μια επιλεγμένη ομάδα.

2)Περιστρέφοντας συγκεκριμένους σερβοκινητήρες του PiArm πατώντας τα πλήκτρα πάνω, κάτω, αριστερά και δεξιά.

Εάν θέλουμε μια συγκεκριμένη κίνηση από το PiArm που έχουμε καταγράψει νωρίτερα, μπορούμε να τρέξουμε κάθε διαφορετική εντολή στο πάτημα του πλήκτρου.

Παράδειγμα: Εάν θέλουμε να επιλέξουμε ένα αντικείμενο, μπορούμε να επιλέξουμε μια ομάδα που επιλέγει αντικείμενα και να την τρέξουμε. Αν θέλουμε να επιλέξουμε ένα αντικείμενο, πατάμε το πλήκτρο προώθησης. Το PiArm επιλέγει το αντικείμενο αν θέλουμε να αντιστρέψουμε τη δράση μπορούμε να πατήσουμε το πλήκτρο προς τα πίσω για να αναπαραγάγουμε την προηγούμενη εντολή.

Τα βήματα για την αναπαραγωγή εντολών στο πάτημα πλήκτρων από μια ομάδα: κάνουμε Read all όλα τα αρχεία από το ~ / PiArm / Example και αποθηκεύουμε σε μια μεταβλητή λίστα.

Ο κώδικας Joystick.py αυτούσιος με μερική επεξήγηση τμημάτων του. Αρχικά γίνεται εισαγωγή των απαιτούμενων βιβλιοθηκών για την εγκατάσταση του Piarm, re, time, pygame.

```
import re
import pygame
import piarm
from time import sleep

class Joystick_Controller(object):
  """
  Joystick controller class
  """

  def __init__(self):
    self.servo_POS_error = False
    self.joystick_keypress_status = False
    self.step = 50
    self.servo_position = {
          1: 500,
          2: 500,
          3: 500,
          4: 500,
          5: 500,
          6: 500,
      }
    self.button_status = {
            0: 0,
            1: 0,
```

```
            2: 0,
            3: 0,
            4: 0,
            5: 0,
            6: 0,
        }
```

Από εδώ γίνεται μία αρχικοποίηση του χειριστηρίου με την εντολή “Pygame.init()”

```
    #  Initialize Joystick
    pygame.init()
    try:
        self.controller = pygame.joystick.Joystick(0)
        self.controller.init()
        print('Joystick initialized')
    except pygame.error as pygame_err:
        print(pygame_err)
    self.read_servo_position()
    print(self.servo_position)
```

Μετά την αρχικοποίηση όλων των εισηγμένων λειτουργιών με την εντολή pygame.init() μπορούμε να ξεκινήσουμε το χειρισμό.

Αφού αρχικοποιήσουμε όλες τις λειτουργίες μπορούμε να δούμε την τρέχουσα κατάσταση όπως κουμπί πατημένο ή ελεύθερο.

*JOYBUTTONDOWN – ελέγχουμε αν είναι πατημένο το κουμπί η όχι

*JOYBUTTONUP – ελέγχουμε αν το κουμπί είναι απελευθερωμένο η όχι.

Για να δούμε την κατάσταση του πλήκτρου τρέχουμε τη εντολή “pygame.event.get()”. Η εντολή αυτή λαμβάνει τα γεγονότα στη σειρά. Εν συνεχεία προγραμματίζει το Piarm καταλλήλως.

```
  def listen(self):
    """
    Listen for events to happen
    """
    while True:
        for event in pygame.event.get():
            #  Button Pressed
            if event.type == pygame.JOYBUTTONDOWN:
                # set Servo to default
                if event.button == 9:
```

```
            if robot.alive:
                for ID in range(1, 7):
                    self.servo_position[ID] = 500
                    robot.servoWrite(ID, self.servo_position[ID], 1500)
            else:
                print('Comm port is not conected')

        # keypress
        elif event.button in range(7):
            self.button_status[event.button] = 1
            print('button {} pressed: '.format(event.button),
                  self.button_status[event.button])
        print(self.button_status)

    #  Button Released
    elif event.type == pygame.JOYBUTTONUP:
        if event.button in range(7):
            self.button_status[event.button] = 0
            print('button {} released: '.format(event.button),
                  self.button_status[event.button])

    #  If facing buttons are pressed reset button position
    if self.button_status[0] == self.button_status[2]:
        self.button_status[0] = 0
        self.button_status[2] = 0
    if self.button_status[1] == self.button_status[3]:
        self.button_status[1] = 0
        self.button_status[3] = 0
```

Σύμφωνα με τη κατάσταση του κουμπιού προγραμματίζονται οι θέσεις των σερβομοτέρ. Σε περίπτωση σειριακού σφάλματος πρέπει να ελέγχουμε τη θύρα σύνδεσης ότι είναι η ttyS0

```
    # Change servo position
    for button, status in self.button_status.items():
        if status:
            #  Move UP
            if button == 0:
                if self.servo_position[5] - self.step in range(101, 999):
                    self.servo_position[5] -= self.step
```

```
            if self.servo_position[4] + self.step in range(1, 999):
                self.servo_position[4] += (self.step + 20)
            if self.servo_position[3] - self.step in range(52, 970):
                self.servo_position[3] -= (self.step + 40)

        #  Move Left
        elif button == 1:
            if self.servo_position[6] - self.step in range(10, 980):
                self.servo_position[6] += self.step

        #  Move Down
        elif button == 2:
            if self.servo_position[5] + self.step in range(101, 999):
                self.servo_position[5] += self.step
            if self.servo_position[4] - self.step in range(1, 999):
                self.servo_position[4] -= (self.step + 20)
            if self.servo_position[3] + self.step in range(52, 970):
                self.servo_position[3] += (self.step + 40)

        #  Move Right
        elif button == 3:
            if self.servo_position[6] + self.step in range(10, 980):
                self.servo_position[6] -= self.step

        #  Claw Open
        elif button == 4:
            if self.servo_position[1] - self.step in range(144, 710):
                self.servo_position[1] -= (self.step + 20)

        #  Claw Close
        elif button == 5:
            if self.servo_position[1] + self.step in range(144, 710):
                self.servo_position[1] += (self.step + 20)
        self.joystick_keypress_status = True

    #  Write current positions
    if self.joystick_keypress_status:
        if robot.alive:
            for ID in range(1, 7):
```

```
                robot.servoWrite(ID, self.servo_position[ID], 1000)
                print("Writing to servo Id {} position: ".format(ID), self.servo_position[ID])
            self.joystick_keypress_status = False
        else:
            print('Comm port is not conected')
        sleep(1)

    def read_servo_position(self):
        '''
        This funciton read current servo position
        '''
        if robot.alive:
            try:
                #  Read Positions of motors one at a time
                for ID in range(1, 7):
                    response = robot.positionRead(ID)
                    pos = int.from_bytes(response[5]+response[6], byteorder='little')
                    #  Button Position to variable
                    self.servo_position[ID] = pos

                if self.servo_POS_error:
                    print("Servo Error", "Servo " + str(ID) +
                                ' - Position Out of Range..!')
                else:
                    print("Motor position Read Done Successfully")

            except TypeError:
                print("Servo Error", "Servo " + str(ID) +
                                ' - Not Responding')

if __name__ == "__main__":
    robot = piarm.PiArm()
    # write your serial comm
    robot.connect("/dev/ttyS0")
    joystick = Joystick_Controller()
    #  Start Joystick
    try:
        joystick.listen()
    #  Set Motors to Default at KeyboardInterrupt
```

```
    except KeyboardInterrupt:
        pass
        for ID in range(1, 7):
            robot.servoWrite(ID, 500, 1500)
```

Κώδικας: Initial calibration camera

```
#https://docs.opencv.org/3.3.0/dc/dbb/tutorial_py_calibration.html

import numpy as np
import cv2
import glob
import time

workingdir="/home/pi/Desktop/Captures/"
savedir="camera_data/"

# termination criteria

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((7*7,3), np.float32)

#add 2.5 to account for 2.5 cm per square in grid
objp[:,:2] = np.mgrid[0:7,0:7].T.reshape(-1,2)*2.5

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.
images = glob.glob('calibration_images/*.jpg')

win_name="Verify"
cv2.namedWindow(win_name, cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty(win_name,cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FULLSCREEN)

print("getting images")
for fname in images:
    img = cv2.imread(fname)
    print(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (7,7), None)
    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)
```

```
        corners2=cv2.cornerSubPix(gray,corners, (11,11), (-1,-1), criteria)
        imgpoints.append(corners)
        # Draw and display the corners
        cv2.drawChessboardCorners(img, (7,7), corners2, ret)
        cv2.imshow(win_name, img)
        cv2.waitKey(500)
    img1=img

cv2.destroyAllWindows()

print(">==> Starting calibration")
ret, cam_mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)

#print(ret)
print("Camera Matrix")
print(cam_mtx)
np.save(savedir+'cam_mtx.npy', cam_mtx)

print("Distortion Coeff")
print(dist)
np.save(savedir+'dist.npy', dist)

print("r vecs")
print(rvecs[2])

print("t Vecs")
print(tvecs[2])

print(">==> Calibration ended")

h,  w = img1.shape[:2]
print("Image Width, Height")
print(w, h)
#if using Alpha 0, so we discard the black pixels from the distortion.  this helps make the entire region of interest is the full dimensions of the image (after undistort)
#if using Alpha 1, we retain the black pixels, and obtain the region of interest as the valid pixels for the matrix.
#i will use Apha 1, so that I don't have to run undistort.. and can just calculate my real world x,y
newcam_mtx, roi=cv2.getOptimalNewCameraMatrix(cam_mtx, dist, (w,h), 1, (w,h))

print("Region of Interest")
print(roi)
np.save(savedir+'roi.npy', roi)

print("New Camera Matrix")
#print(newcam_mtx)
np.save(savedir+'newcam_mtx.npy', newcam_mtx)
```

```
print(np.load(savedir+'newcam_mtx.npy'))

inverse = np.linalg.inv(newcam_mtx)
print("Inverse New Camera Matrix")
print(inverse)


# undistort
undst = cv2.undistort(img1, cam_mtx, dist, None, newcam_mtx)

# crop the image
#x, y, w, h = roi
#dst = dst[y:y+h, x:x+w]
#cv2.circle(dst,(308,160),5,(0,255,0),2)
cv2.imshow('img1', img1)
cv2.waitKey(5000)
cv2.destroyAllWindows()
cv2.imshow('img1', undst)
cv2.waitKey(5000)
cv2.destroyAllWindows()
```

initial_perspective_calibration.py

```
#https://docs.opencv.org/3.3.0/dc/dbb/tutorial_py_calibration.html

import numpy as np
import cv2
import glob
import camera_realworldxyz

cameraXYZ=camera_realworldxyz.camera_realtimeXYZ()

calculatefromCam=True

imgdir="/home/pi/Desktop/Captures/"

writeValues=False

#test camera calibration against all points, calculating XYZ

#load camera calibration
savedir="camera_data/"
cam_mtx=np.load(savedir+'cam_mtx.npy')
dist=np.load(savedir+'dist.npy')
```

```
newcam_mtx=np.load(savedir+'newcam_mtx.npy')
roi=np.load(savedir+'roi.npy')

#load center points from New Camera matrix
cx=newcam_mtx[0,2]
cy=newcam_mtx[1,2]
fx=newcam_mtx[0,0]
print("cx: "+str(cx)+",cy "+str(cy)+",fx "+str(fx))

#MANUALLY INPUT YOUR MEASURED POINTS HERE
#ENTER (X,Y,d*)
#d* is the distance from your point to the camera lens. (d* = Z for the camera center)
#we will calculate Z in the next steps after extracting the new_cam matrix

#world center + 9 world points

total_points_used=10

X_center=10.9
Y_center=10.7
Z_center=43.4
worldPoints=np.array([[X_center,Y_center,Z_center],
                      [5.5,3.9,46.8],
                      [14.2,3.9,47.0],
                      [22.8,3.9,47.4],
                      [5.5,10.6,44.2],
                      [14.2,10.6,43.8],
                      [22.8,10.6,44.8],
                      [5.5,17.3,43],
                      [14.2,17.3,42.5],
                      [22.8,17.3,44.4]], dtype=np.float32)

#MANUALLY INPUT THE DETECTED IMAGE COORDINATES HERE

#[u,v] center + 9 Image points
imagePoints=np.array([[cx,cy],
                      [502,185],
                      [700,197],
                      [894,208],
                      [491,331],
                      [695,342],
                      [896,353],
                      [478,487],
                      [691,497],
                      [900,508]], dtype=np.float32)
```

```
#FOR REAL WORLD POINTS, CALCULATE Z from d*

for i in range(1,total_points_used):
   #start from 1, given for center Z=d*
   #to center of camera
   wX=worldPoints[i,0]-X_center
   wY=worldPoints[i,1]-Y_center
   wd=worldPoints[i,2]

   d1=np.sqrt(np.square(wX)+np.square(wY))
   wZ=np.sqrt(np.square(wd)-np.square(d1))
   worldPoints[i,2]=wZ

print(worldPoints)



#print(ret)
print("Camera Matrix")
print(cam_mtx)
print("Distortion Coeff")
print(dist)

print("Region of Interest")
print(roi)
print("New Camera Matrix")
print(newcam_mtx)
inverse_newcam_mtx = np.linalg.inv(newcam_mtx)
print("Inverse New Camera Matrix")
print(inverse_newcam_mtx)
if writeValues==True: np.save(savedir+'inverse_newcam_mtx.npy', inverse_newcam_mtx)

print(">==> Calibration Loaded")



print("solvePNP")
ret, rvec1, tvec1=cv2.solvePnP(worldPoints,imagePoints,newcam_mtx,dist)

print("pnp rvec1 - Rotation")
print(rvec1)
if writeValues==True: np.save(savedir+'rvec1.npy', rvec1)

print("pnp tvec1 - Translation")
print(tvec1)
if writeValues==True: np.save(savedir+'tvec1.npy', tvec1)
```

```
print("R - rodrigues vecs")
R_mtx, jac=cv2.Rodrigues(rvec1)
print(R_mtx)
if writeValues==True: np.save(savedir+'R_mtx.npy', R_mtx)

print("R|t - Extrinsic Matrix")
Rt=np.column_stack((R_mtx,tvec1))
print(Rt)
if writeValues==True: np.save(savedir+'Rt.npy', Rt)

print("newCamMtx*R|t - Projection Matrix")
P_mtx=newcam_mtx.dot(Rt)
print(P_mtx)
if writeValues==True: np.save(savedir+'P_mtx.npy', P_mtx)

#[XYZ1]

#LETS CHECK THE ACCURACY HERE

s_arr=np.array([0], dtype=np.float32)
s_describe=np.array([0,0,0,0,0,0,0,0,0,0],dtype=np.float32)

for i in range(0,total_points_used):
   print("=======POINT # " + str(i) +" ========================")

   print("Forward: From World Points, Find Image Pixel")
   XYZ1=np.array([[worldPoints[i,0],worldPoints[i,1],worldPoints[i,2],1]], dtype=np.float32)
   XYZ1=XYZ1.T
   print("{{-- XYZ1")
   print(XYZ1)
   suv1=P_mtx.dot(XYZ1)
   print("//-- suv1")
   print(suv1)
   s=suv1[2,0]
   uv1=suv1/s
   print(">==> uv1 - Image Points")
   print(uv1)
   print(">==> s - Scaling Factor")
   print(s)
   s_arr=np.array([s/total_points_used+s_arr[0]], dtype=np.float32)
   s_describe[i]=s
   if writeValues==True: np.save(savedir+'s_arr.npy', s_arr)

   print("Solve: From Image Pixels, find World Points")

   uv_1=np.array([[imagePoints[i,0],imagePoints[i,1],1]], dtype=np.float32)
```

```
    uv_1=uv_1.T
    print(">==> uv1")
    print(uv_1)
    suv_1=s*uv_1
    print("//-- suv1")
    print(suv_1)

    print("get camera coordinates, multiply by inverse Camera Matrix, subtract tvec1")
    xyz_c=inverse_newcam_mtx.dot(suv_1)
    xyz_c=xyz_c-tvec1
    print("      xyz_c")
    inverse_R_mtx = np.linalg.inv(R_mtx)
    XYZ=inverse_R_mtx.dot(xyz_c)
    print("{{-- XYZ")
    print(XYZ)

    if calculatefromCam==True:
        cXYZ=cameraXYZ.calculate_XYZ(imagePoints[i,0],imagePoints[i,1])
        print("camXYZ")
        print(cXYZ)


s_mean, s_std = np.mean(s_describe), np.std(s_describe)

print(">>>>>>>>>>>>>>>>>>>>> S RESULTS")
print("Mean: "+ str(s_mean))
#print("Average: " + str(s_arr[0]))
print("Std: " + str(s_std))

print(">>>>>> S Error by Point")

for i in range(0,total_points_used):
    print("Point "+str(i))
    print("S: " +str(s_describe[i])+" Mean: " +str(s_mean) + " Error: " + str(s_describe[i]-s_mean))
```