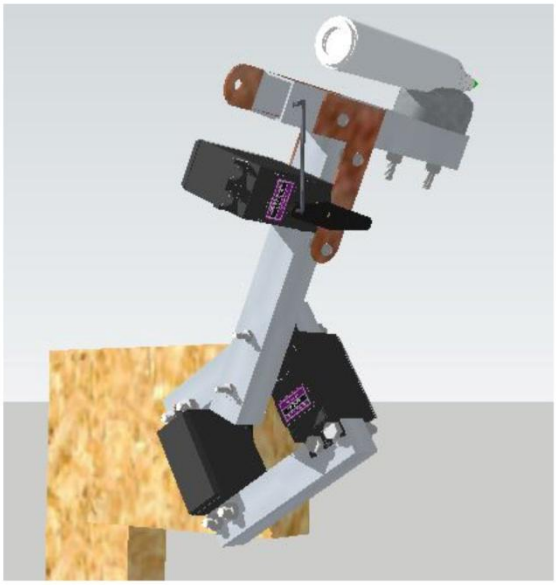
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΚΟΠΗΣ & ΚΑΤΑΣΚΕΥΑΣΤΙΚΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

ΣΧΕΔΙΑΣΜΟΣ, ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ



ΔΕΛΗΚΩΝΣΤΑΝΤΙΝΟΥ ΒΑΣΙΛΕΙΟΣ

ΕΠΙΒΛΕΠΩΝ: ΑΡΙΣΤΟΜΕΝΗΣ ΑΝΤΩΝΙΑΔΗΣ
ΚΑΘΗΓΗΤΗΣ

ΑΡ. ΔΙΠΛ. : 56

ΧΑΝΙΑ 2016

"*Οι τρείς νόμοι της Ρομποτικής:*

Το ρομπότ δε θα κάνει κακό σε άνθρωπο, ούτε με την αδράνειά του θα βλάψει ανθρώπινο όν.

Το ρομπότ πρέπει να υπακούει τις διαταγές που του δίνουν οι άνθρωποι, εκτός αν αυτές οι διαταγές έρχονται σε αντίθεση με τον πρώτο νόμο.

Το ρομπότ οφείλει να προστατεύει την ύπαρξή του, εφόσον αυτό δε συγκρούεται με τον πρώτο και τον δεύτερο νόμο."

Ισαάκ Ασίμωφ, (1942), "Runaround"

Η παρούσα διπλωματική εργασία αποτέλεσε για εμένα ένα ταξίδι στη γνώση που άπτεται του φάσματος της ρομποτικής, αλλά, παράλληλα αποτέλεσε και ένα ταξίδι στην αυτογνωσία. Η εργασία κάτω από την πίεση του χρόνου, η έρευνα, τα πειράματα, οι δυσκολίες και τα προβλήματα, δημιούργησαν στιγμές έντονου άγχους και στρες, οι οποίες βέβαια λειτούργησαν ως εργαλείο παρατήρησης και συνειδητοποίησης του εαυτού μου.

Με την περάτωση αυτής της εργασίας οφείλω να ευχαριστήσω κάποιους ανθρώπους οι οποίοι με βοήθησαν είτε με τις γνώσεις τους στο αντικείμενο της εργασίας, είτε σε ψυχολογικό επίπεδο, το οποίο ήταν εξίσου σημαντικό για εμένα.

Συνεπώς θα ήθελα να ευχαριστήσω μέσα από την καρδία μου την οικογένειά μου. Τους γονείς μου, Αγάπιο και Ματίνα Δεληκωνσταντίνου, που πιστεύουν και στηρίζουν ακατάπαυστα όλες τις επιλογές μου, την αδερφή μου, Μαρία Δεληκωνσταντίνου και τη σύντροφό μου, Παναγιώτα Καραφίνα, οι οποίες είναι δίπλα μου και με στηρίζουν πάντα. Επίσης όλους τους φίλους μου που με πιστεύουν και είναι δίπλα μου.

Ευχαριστώ ιδιαίτερα τον κ. Αναστάσιο Χαραλαμπάκη, εξαίρετο ψυχοθεραπευτή, ο οποίος δρα καταλυτικά στο δρόμο προς την αυτογνωσία μου. Επίσης ευχαριστώ μέσα από την καρδιά μου τον κ. Γιώργο Μποζιονέλο, ο οποίος διέθεσε πολύτιμο χρόνο προκειμένου να μου μεταβιβάσει τις γνώσεις του σε πολλούς τομείς στην εργασία.

Επίσης ευχαριστίες και ευχές για τη σταδιοδρομία τους, στους Δημήτρη Βακόνδιο, Χαρά Ευσταθίου, Αντώνη Λυρώνη και Κώστα Σοφιάκη, οι οποίοι ως μέλη του Εργαστηρίου Μικροκοπής και Κατασκευαστικής Προσομοίωσης Μ3 με βοήθησαν με τις ιδέες τους.

Τέλος θα ήθελα να απονείμω θερμές ευχαριστίες στον καθηγητή μου, Δρ. Αριστομένη Αντωνιάδη, έναν άνθρωπο με όραμα και θετική ενέργεια για τους γύρω του, ο οποίος με πίστεψε, με στήριξε και με βοήθησε καθ'όλη τη διάρκεια της εκπόνησης της εργασίας.

Βασίλης Δεληκωνσταντίνου
Χανιά, Σεπτέμβριος 2016

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο εργασίας

Στην παρούσα διπλωματική εργασία παρουσιάζεται ο σχεδιασμός, η κατασκευή και ο προγραμματισμός ενός ρομποτικού συστήματος, το οποίο έχει ως κύρια εργασία τη γραφή λέξεων που εισάγονται από τον χρήστη. Το ρομποτικό, αυτό, σύστημα αποτελείται από δύο υποσυστήματα, τα οποία επικοινωνούν μεταξύ τους. Το ένα υποσύστημα είναι ένας επίπεδος βραχίονας δύο περιστροφικών αρθρώσεων, ο οποίος γράφει τις λέξεις που εισάγονται, ενώ το άλλο υποσύστημα αποτελείται από έναν άξονα, με ενσωματωμένο σφουγγάρι, ο οποίος είναι συνδεδεμένος με έναν σερβοκινητήρα, έτσι ώστε να περιστρέφεται στο επίπεδο του πίνακα γραφής και να σαρώνει την επιφάνειά του, καθαρίζοντάς τον. Ο έλεγχος και ο προγραμματισμός του συστήματος γίνεται από την πλατφόρμα Arduino και η γλώσσα προγραμματισμού του είναι η Wiring C. Ο στόχος της παρούσας εργασίας, είναι η κατασκευή του προαναφερθέντος ρομποτικού συστήματος με τα εξής κριτήρια: την πρωτοτυπία του σχεδίου, την απλότητα στην κατασκευή και τη λειτουργία του, τη χρήση, κυρίως, εργαλείων χειρός, την επαναχρησιμοποίηση αντικειμένων και εξαρτημάτων, τα οποία βρίσκονται στα περισσότερα σπίτια, καθώς και το χαμηλό κόστος κατασκευής. Ο σκοπός της παρούσας εργασίας είναι εκπαιδευτικός και αφορά στην κατανόηση της λειτουργίας των ρομποτικών συστημάτων.

1.2 Δομή εργασίας

Το **κεφάλαιο 1** αποτελεί μια εισαγωγή στην εργασία μέσω της παρουσίασης του αντικειμένου και της δομής της.

Στο **κεφάλαιο 2** παρουσιάζεται η απαραίτητη βιβλιογραφική έρευνα για την ιστορία και τη λειτουργία των ρομποτικών συστημάτων, καθώς και των εξαρτημάτων που χρησιμοποιούνται στην κατασκευή της παρούσας εργασίας, όπως το Arduino και οι σερβοκινητήρες.

Το **κεφάλαιο 3** αποτελεί μία εκτενή παρουσίαση του ρομποτικού συστήματος, στο οποίο αναφέρεται η εργασία. Παρουσιάζεται βήμα προς βήμα ο σχεδιασμός, η κατασκευή και ο προγραμματισμός του.

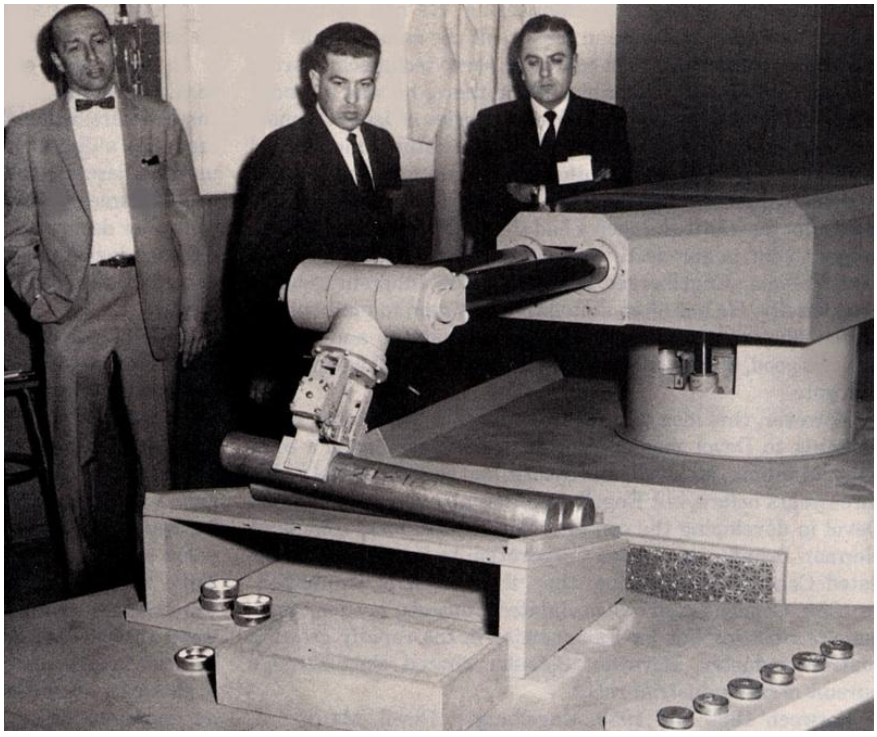
Στο **κεφάλαιο 4** αναφέρονται τα συμπεράσματα, οι προτάσεις για τη βελτίωση του ρομποτικού συστήματος, καθώς και το κόστος υλοποίησης της εργασίας.

2. ΣΤΑΘΜΗ ΓΝΩΣΕΩΝ

2.1 Ρομποτική και ρομποτικοί βραχίονες

2.1.1 Ιστορία και τομείς εφαρμογής

Ρομποτική, καλείται ο κλάδος της επιστήμης, ο οποίος ασχολείται με το σχεδιασμό και την ανάπτυξη ρομποτικών συστημάτων. Ρομποτικά συστήματα ή ρομπότ, καλούνται τα αυτοματοποιημένα συστήματα τα οποία έχουν τη δυνατότητα να αντικαθιστούν τον άνθρωπο σε αρκετές εργασίες, είτε ελεγχόμενα από αυτόν, είτε αυτόνομα με τη χρήση της τεχνητής νοημοσύνης (AI). Ο όρος ρομπότ πρωτοεμφανίστηκε το 1921 στο θεατρικό έργο επιστημονικής φαντασίας "R.U.R." (Rossum's Universal Robots) του Τσέχου συγγραφέα **Κάρελ Τσάπεκ**, στο οποίο σατιρίζεται η εξάρτηση της κοινωνίας από τους μηχανικούς εργάτες (ρομπότ) της τεχνολογικής εξέλιξης, οι οποίο τελικά εξοντώνουν τους δημιουργούς τους. Σύμφωνα βέβαια, με τον ορισμό του **Ινστιτούτου Ρομπότ των ΗΠΑ**, το ρομπότ είναι μια επαναπρογραμματιζόμενη πολυλειτουργική χειριστική διάταξη, σχεδιασμένη για τη μετακίνηση υλικών, εξαρτημάτων, εργαλείων και εξειδικευμένων διατάξεων, μέσω μεταβλητών, προγραμματισμένων κινήσεων για την εκτέλεση μιας σειράς εργασιών. Η προσπάθεια για βελτιστοποίηση της παραγωγικής διαδικασίας, σε βιομηχανίες μαζικής παραγωγής προϊόντων, όσον αφορά τον όγκο, την ποιότητα, το έλεγχο των γραμμών παραγωγής και το κέρδος των βιομηχανιών παραγωγής, οδήγησε στο σχεδιασμό και στη δημιουργία τέτοιων αυτοματοποιημένων συστημάτων. Το 1961 συναντάται η κατασκευή και η χρήση του πρώτου βιομηχανικού ρομπότ, με υδραυλικούς ενεργοποιητές, από την Unimation με κύρια λειτουργία τη μεταφορά αντικειμένων από ένα σημείο σε ένα άλλο, μέγιστης απόστασης τεσσάρων μέτρων, όπως φαίνεται στο σχήμα 2.1. Η εταιρία Unimation ιδρύθηκε το 1956 από τους **George Devol** και **Joseph F. Engelberger**, με βάση το δίπλωμα ευρεσιτεχνίας για τα ρομπότ που κατείχε ο Devol. Μετά από μία δεκαετία περίπου, αρκετοί όμιλοι ανά τον κόσμο άρχισαν να δείχνουν το ενδιαφέρον τους στην κατασκευή βιομηχανικών ρομπότ, επεκτείνοντας το εύρος λειτουργιών τους.



Σχήμα 2.1: Το πρώτο βιομηχανικό ρομπότ από τη Unimation το 1961

Το 1969 ο **Victor Scheinman** στο Πανεπιστήμιο του Στάνφορντ δημιούργησε το "βραχίονα του Στάνφορντ", όπως φαίνεται στο σχήμα 2.2.

Σχήμα 2.2: Ο βραχίονας του Στάνφορντ το 1969

Ένας πλήρως ηλεκτρικός βραχίονας με έξι αρθρώσεις , οι οποίες του έδιναν τη δυνατότητα να κινείται με ακρίβεια σε πολλά σημεία στο χώρο. Συνεπώς οι δυνατότητες των ρομπότ άρχισαν να επεκτείνονται, με αποτέλεσμα τη χρήση αυτών σε πιο εξελιγμένες εφαρμογές, όπως η συναρμολόγηση και συγκόλληση. Ο Scheinman σχεδίασε κι ένα δεύτερο βραχίονα για το εργαστήριο Τεχνητής Νοημοσύνης του MIT. Στη συνέχεια έλαβε μια υποτροφία από την Unimation για να εξελίξει τα σχέδια του, τα οποία πούλησε στην ίδια εταιρία, ενώ συνέχισε να τα εξελίσσει με την υποστήριξη της General Motors. Το αποτέλεσμα αυτών των συνεργασιών ήταν η καθολικά προγραμματιζόμενη μηχανή για συναρμολόγηση (PUMA), όπως φαίνεται στο σχήμα 2.3.

Σχήμα 2.3: Ο Scheinman και ο βραχίονας PUMA

Στην Ευρώπη, η ρομποτική βιομηχανία αναπτύχθηκε πολύ γρήγορα, τόσο από την ABB Robotics, όσο και από την KUKA Robotics, όπου διέθεσαν ρομπότ στην αγορά το 1973. Η ABB Robotics ήταν μεταξύ των πρώτων εταιριών στον κόσμο, που διέθεσαν εξολοκλήρου ηλεκτρικά ρομπότ ,στον εμπόριο, τα οποία ελέγχονταν από μικροεπεξεργαστή. Επίσης, το 1973 η KUKA Robotics δημιούργησε το πρώτο ρομπότ, γνωστό ως FAMULUS, το ποίο όπως και το PUMA ήταν ένα από τα πρώτα αρθρωτά ρομπότ που δούλευαν με έξι ηλεκτρομηχανικούς άξονες.

Αξίζει να σημειωθεί ότι οι ABB και η KUKA Robotics πρωταγωνιστούν ακόμα στο χώρο της ρομποτικής, μαζί με αρκετές ακόμη εταιρίες, όπως η Automatrix, η Adept Technology, η Stäubli-Unimation, η Swedish-Swiss και άλλες, κυρίως Ιαπωνικές, όπως η FANUC LTD.

Στο σχήμα 2.4 παρατηρείται ένας βραχίονας της KUKA Robotics, τελευταίας τεχνολογίας, ο οποίος χρησιμοποιείται αρκετά από τις αυτοκινητοβιομηχανίες.

Σχήμα 2.4: Βραχίονας KUKA Robotics του 2015

Η χρήση των ρομπότ εκτός από τη βιομηχανία, έχει αρχίσει να γίνεται επιτυχημένα και σε άλλους τομείς, όπως την ιατρική για χειρουργικές επεμβάσεις και διαγνώσεις, την αεροναυπηγική, στις γραμμές παραγωγής αεροσκαφών , την αεροδιαστημική, όπως τα αυτοματοποιημένα διαστημόπλοια (μη επανδρωμένα), που χρησιμοποιούνται για διαστημικές έρευνες, την έρευνα και διάσωση θυμάτων φυσικών καταστροφών κ.ά., γεγονός που δίνει σημαντική ώθηση στην περαιτέρω ανάπτυξη της βιομηχανίας των ρομπότ. Οι κυριότερες εφαρμογές των βιομηχανικών ρομπότ, μέχρι σήμερα, ήταν οι ηλεκτροσυγκολλήσεις, οι εφαρμογές σε εργασίες πρεσαρίσματος, οι συναρμολογήσεις, οι βαφές με ψεκασμό και η επεξεργασία επιφανειών σε τροφοδοτήσεις εργαλειομηχανών, σε μορφοποιήσεις πλαστικών σε μήτρες κ.ά. Από τα μέσα περίπου της δεκαετίας του 1980 η χρήση των ρομπότ γενικεύτηκε στο πλαίσιο της ανάπτυξης των Ολοκληρωμένων Συστημάτων Παραγωγής (Computer-Integrated Manufacturing), αυτοματοποιημένων και ευέλικτων εργοστασίων, στα οποία οι εργαλειομηχανές έχουν τη δυνατότητα να επαναπρογραμματίζονται ταχύτατα για την παραγωγή νέων ή διαφοροποιημένων προϊόντων.

Επίσης, σε αισιόδοξο στάδιο βρίσκεται τα τελευταία χρόνια και η παραγωγή οικιακών ρομπότ για τις διάφορες εργασίες του σπιτιού, όπως καθάρισμα, σερβίρισμα κ.α. Στο σχήμα 2.5 φαίνεται το Zenbo, το οικιακό ρομπότ που παρουσίασε η Asus στην Computex στην Ταϊβάν, τον Ιούνιο του 2016 και προορίζεται ως κέντρο ψυχαγωγίας του σπιτιού.

Σχήμα 2.5: Το οικιακό ρομπότ της Asus με το όνομα Zembo

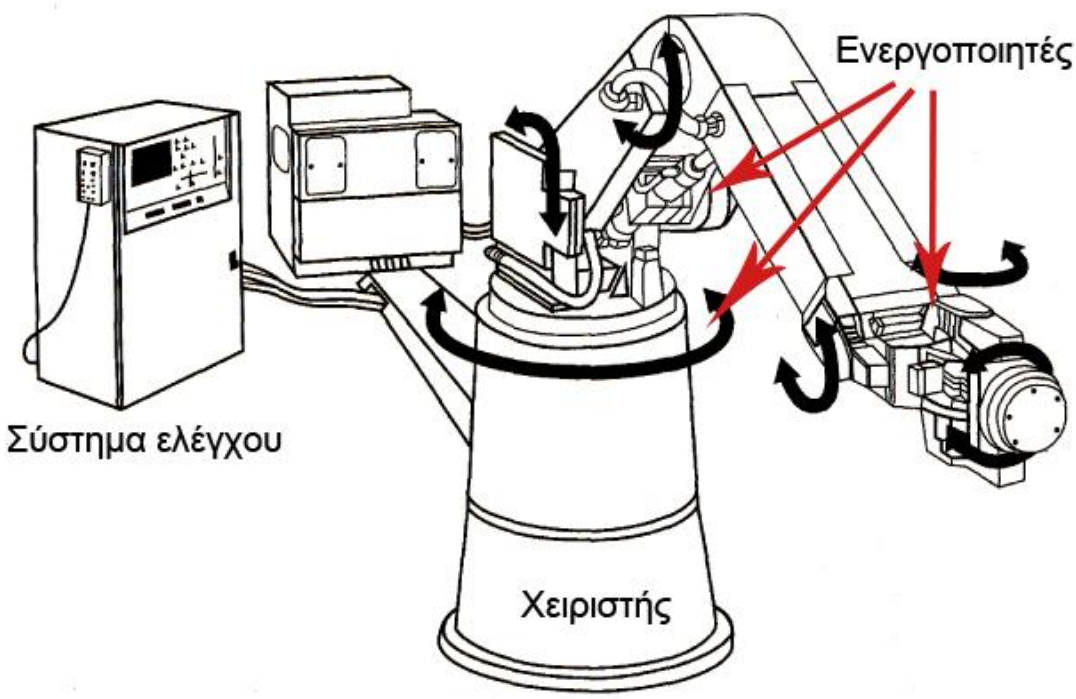
Η ανάπτυξη του κλάδου της τεχνητής νοημοσύνης (artificial intelligence) κατά τη δεκαετία του 1980 άνοιξε ευρύτατες προοπτικές εφαρμογής της στη ρομποτική. Η τεχνητή νοημοσύνη αποτελεί ένα από τα σημαντικότερα πεδία έρευνας της πληροφορικής και αφορά την κατασκευή συστημάτων αυτοματισμού εφοδιασμένων με ικανότητα μάθησης, δυνατότητα κατανόησης της φυσικής γλώσσας, ικανότητα αξιολόγησης στοιχείων, λήψης αποφάσεων κ.ά. Οι σχετικές έρευνες στον τομέα της ρομποτικής αφορούν την κατασκευή ρομπότ τα οποία πέρα από τις βασικές αισθήσεις, όπως η αφή και η όραση, θα είναι εφοδιασμένα με αντιληπτικές ικανότητες (για παράδειγμα, αντίληψη σχημάτων, μορφών, εικόνων κ.λπ.), με ικανότητα διεξαγωγής λογικών συνειρμών και εξαγωγής

συμπερασμάτων, καθώς και με δυνατότητες ανακατανομής δεδομένων ανάλογα με τη χρήση για την οποία ζητούνται και με ικανότητα αυτοδιόρθωσης. Η ανάπτυξη της προηγμένης τεχνολογίας ρομπότ αναμένεται ότι θα βοηθήσει σημαντικά στην επίλυση προβλημάτων και στην ολοκλήρωση εργασιών σε χώρους που είναι δύσκολα προσπελάσιμοι για τον άνθρωπο.

2.1.2 Βασική δομή ρομποτικού βραχίονα

Ως ρομποτικός βραχίονας θεωρείται το ρομπότ εκείνο το οποίο είναι σχεδιασμένο, κατασκευασμένο και προγραμματισμένο έτσι ώστε να προσομοιώνει το ανθρώπινο χέρι. Στη βιβλιογραφία συναντάται συχνά με τη γενική ονομασία **βιομηχανικό ρομπότ**. Η βασική δομή ενός τέτοιου ρομπότ περιλαμβάνει τουλάχιστον τα παρακάτω τρία δομικά μέρη, όπως φαίνεται και στο σχήμα 2.6 :

- Το κινούμενο μηχανικό τμήμα που ονομάζεται **χειριστής.**
- Τα στοιχεία που ενεργοποιούν το χειριστή και ονομάζονται **στοιχεία δράσης** ή **ενεργοποιητές (actuators).**
- Το **σύστημα ελέγχου**, που αποθηκεύει και εκτελεί τα προγράμματα εργασίας και ελέγχει τις κινήσεις του ρομπότ.



Σχήμα 2.6: Η βασική δομή ενός βιομηχανικού ρομποτικού βραχίονα

Χειριστής: Ο χειριστής σε ένα ρομποτικό βραχίονα δομείται όπως το ανθρώπινο χέρι. Το ανθρώπινο χέρι αποτελείται από αρθρώσεις, τμήματα που συνδέουν τις αρθρώσεις αυτές

(μύες) και τα δάκτυλα, έτσι και ένας ρομποτικός βραχίονας είναι σχεδιασμένος κατά αυτόν τον τρόπο. Συνεπώς ένας βραχίονας αντίστοιχα, αποτελείται από τις **αρθρώσεις (joints)**, οι οποίες αναλύονται παρακάτω, τα τμήματα που συνδέουν τις αρθρώσεις και ονομάζονται **σύνδεσμοι (links)** και το **τελικό σημείο δράσης (tool)**, το οποίο μπορεί να είναι μία αρπάγη, ένα εργαλείο συγκόλλησης, ένα εργαλείο βαφής ή οποιοδήποτε άλλο εργαλείο το οποίο είναι απαραίτητο στην εκάστοτε κατεργασία.

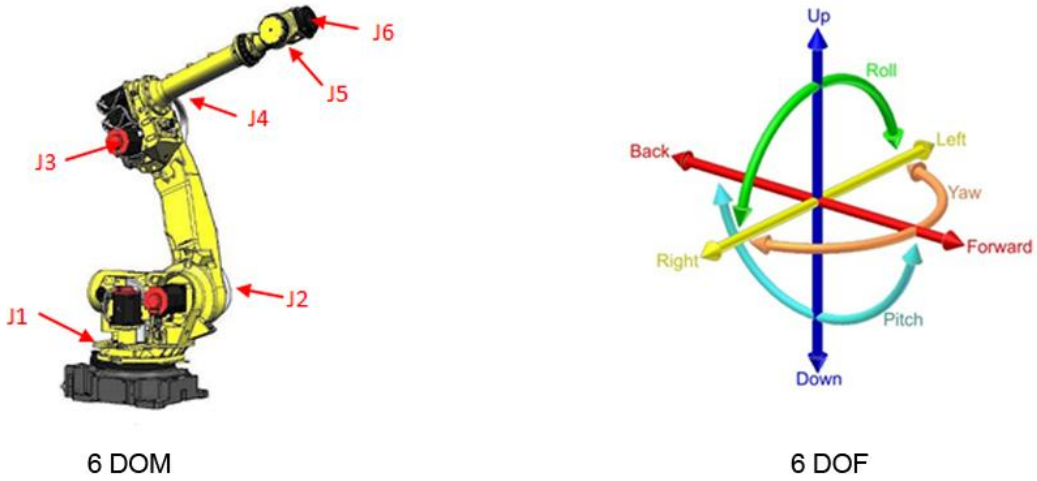
Ενεργοποιητές: Η μηχανική κίνηση του βραχίονα επιτυγχάνεται μέσω των ενεργοποιητών οι οποίοι μπορεί να είναι κινητήρες, υδραυλικά συστήματα κ.α. Στην περίπτωση ενός ηλεκτρικού βραχίονα οι ενεργοποιητές είναι ηλεκτρικοί κινητήρες, όπως παραδείγματος χάρη, οι σερβοκινητήρες, οι οποίοι αναλύονται παρακάτω.

Σύστημα ελέγχου: Το σύστημα ελέγχου είναι ο εγκέφαλος του ρομπότ. Είναι ένα υπολογιστικό σύστημα, όπως παραδείγματος χάρη το Arduino, το οποίο έχει τη δυνατότητα να λαμβάνει δεδομένα από το χρήστη ή από αισθητήρες και να εκτελεί τα αντίστοιχα προγράμματα για την επιθυμητή κίνηση του ρομπότ.

2.1.3 Βαθμοί ελευθερίας (DOF) και βαθμοί κινητικότητας (DOM)

Βαθμός ελευθερίας και βαθμός κινητικότητας είναι δύο έννοιες που συχνά ταυτίζονται, εντούτοις υπάρχει σαφής διαχωρισμός μεταξύ τους. Το σταθερό πλήθος των αρθρώσεων ενός βραχίονα κατά την κατασκευή του, αντικατοπτρίζει τους **βαθμούς κινητικότητάς (Degrees of Manipulandum)** του. Συνεπώς ένας βραχίονας ο οποίος από την κατασκευή του έχει π.χ. 5 αρθρώσεις, έχει αντίστοιχα και 5 βαθμούς κινητικότητας. Ενώ οι **βαθμοί ελευθερίας (Degrees of Freedom)** είναι συνδεδεμένοι με την εκάστοτε εργασία, την οποία καλείται να ολοκληρώσει ο βραχίονας. Συνεπώς, το σύνολο των ανεξαρτήτων μεταβλητών με βάση τις οποίες περιγράφεται πλήρως η θέση των υλικών σημείων του συστήματος, αποτελεί τους βαθμούς ελευθερίας του συστήματος.

Έστω ότι μια εργασία του βραχίονα απαιτεί την τοποθέτηση ενός υλικού στον τρισδιάστατο χώρο, τότε οι βαθμοί ελευθερίας που απαιτούνται είναι 3. Συνεπώς αν ο βραχίονας έχει 3 βαθμούς κινητικότητας ή και παραπάνω, θα μπορεί να ανταπεξέλθει. Αν όμως χρειαστεί να γίνει και ο προσανατολισμός του υλικού στο χώρο σύμφωνα με ένα σύστημα αναφοράς, τότε οι βαθμοί ελευθερίας που απαιτεί η εργασία, αυξάνονται σε έξι. Οπότε θα χρειαστεί ένας βραχίονας με 6 βαθμούς κινητικότητας, προκειμένου να μπορεί να εκτελέσει αυτήν την εργασία, όπως φαίνεται στο σχήμα 2.7.



Σχήμα 2.7: Οι 6 βαθμοί κινητικότητας και οι 6 βαθμοί ελευθερίας

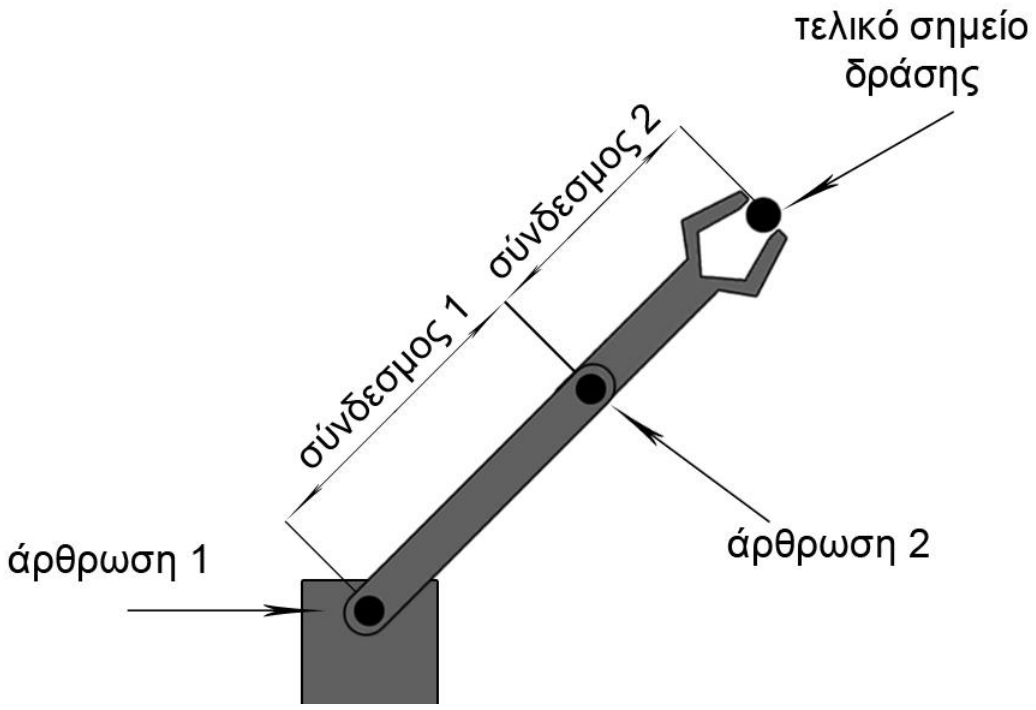
2.1.4 Αρθρώσεις

Όπως έχει προαναφερθεί, ένας βραχίονας αποτελείται από συνδέσμους οι οποίοι ενώνονται ανά δύο μεταξύ τους και οι διαδοχική αυτή ένωση σχηματίζει μια κινηματική αλυσίδα. Οι μηχανισμοί που ενώνουν του συνδέσμους και προσδίδουν κίνηση στην κινηματική αλυσίδα, ονομάζονται αρθρώσεις.

Οι αρθρώσεις αυτές χωρίζονται σε τρεις κύριες κατηγορίες:

- Περιστροφικές αρθρώσεις
- Πρισματικές ή γραμμικές ή τηλεσκοπικές αρθρώσεις
- Σύνθετες αρθρώσεις

Ένας τυπικό σχέδιο βραχίονα 2 βαθμών κινητικότητας ή ελευθερίας με 2 περιστροφικές αρθρώσεις είναι αυτό που φαίνεται στο σχήμα 2.8.

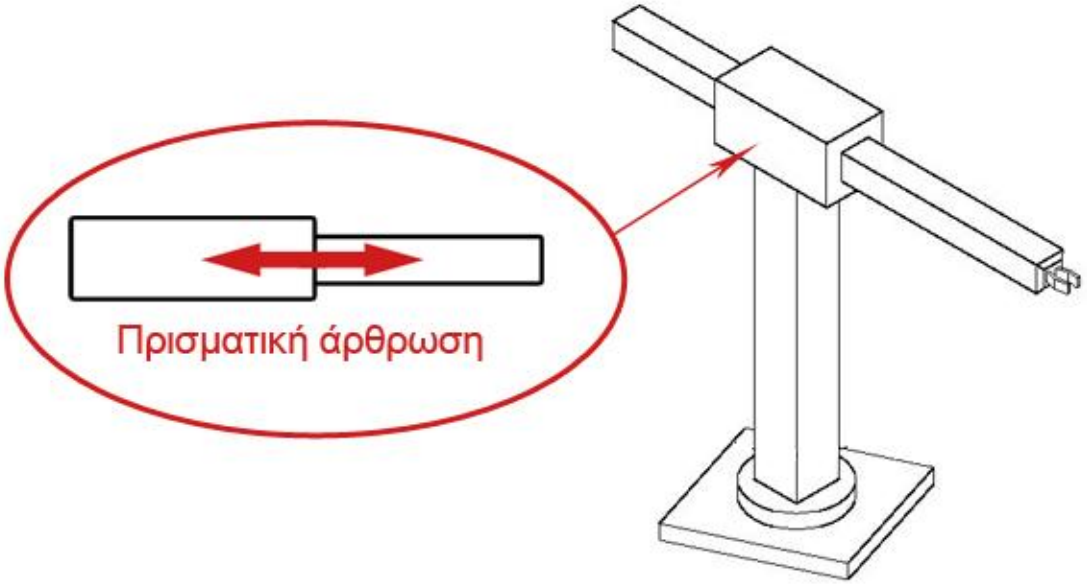


Σχήμα 2.8: Τυπικό σχέδιο βραχίονα με 2 περιστροφικές αρθρώσεις

Περιστροφική άρθρωση: Η περιστροφική άρθρωση ή revolute joint στα Αγγλικά, με συμβολισμό R, είναι άρθρωση που επιτρέπει τη σχετική περιστροφική κίνηση δύο διαδοχικών συνδέσμων. Δίνει ένα βαθμό ελευθερίας στο βραχίονα, επιτρέποντας στο σώμα του να περιστραφεί σε ένα επίπεδο, ενώ ταυτόχρονα αποκόπτει κάθε άλλη δυνατότητα κίνησης. Η μεταβλητή που ορίζει την κίνησή της είναι η γωνία θ ή q, η οποία θα αναλυθεί παρακάτω, στην παράγραφο της κινηματικής των βραχιόνων.

Πρισματική άρθρωση: Η πρισματική (ή γραμμική ή τηλεσκοπική) άρθρωση ή prismatic joint στα Αγγλικά, με συμβολισμό P, είναι άρθρωση που επιτρέπει σχετική μετατόπιση (σε ευθεία γραμμή) μεταξύ δύο διαδοχικών συνδέσμων. Δίνει ένα βαθμό ελευθερίας στο βραχίονα επιτρέποντας στο σώμα του να μετατοπίζεται στη διεύθυνση του άξονα στον

οποίο είναι προσανατολισμένη, ενώ αποκόπτει κάθε άλλη δυνατότητα κίνησης. Η μεταβλητή που ορίζει την κίνησή της είναι η μετατόπιση d, η οποία θα αναλυθεί παρακάτω, στην παράγραφο της κινηματικής των βραχιόνων. Στο σχήμα 2.9 φαίνεται μια πρισματική άρθρωση.



Σχήμα 2.9: Πρισματική άρθρωση

Σύνθετες αρθρώσεις: Σύνθετες αρθρώσεις είναι αυτές οι οποίες αναλύονται γεωμετρικά σε υπέρθεση δύο ή περισσότερων από τις βασικές αρθρώσεις (περιστροφική και πρισματική). Σύνθετες αρθρώσεις είναι:

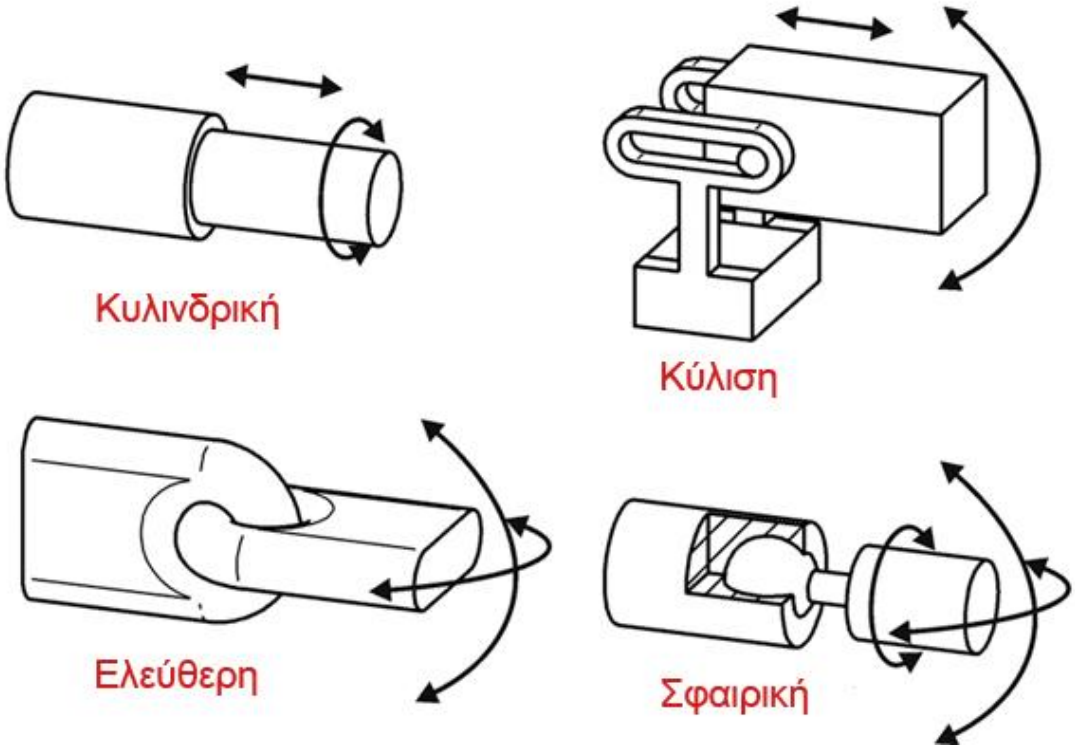
- Η **κυλινδρική άρθρωση** δίνει δύο βαθμούς ελευθερίας στο βραχίονα και επιτρέπει στο σώμα του να εκτελεί μία μεταφορική κίνηση στη διεύθυνση ενός άξονα και μία περιστροφική γύρω από τον άξονα αυτό.

- Η **άρθρωση της κύλισης** δίνει και αυτή δύο βαθμούς ελευθερίας, δηλαδή μία μεταφορική και μία περιστροφική κίνηση, άλλα σε αυτή την περίπτωση ο άξονας της περιστροφικής κίνησης είναι κάθετος στη διεύθυνση του άξονα που πραγματοποιείται η μεταφορική κίνηση.

- Η **ελεύθερη άρθρωση** δίνει δύο βαθμούς ελευθερίας επιτρέποντας δύο περιστροφικές κινήσεις και εμποδίζοντας όλες τις υπόλοιπες.

- Η **σφαιρική άρθρωση** δίνει τρεις βαθμούς ελευθερίας αφήνοντας και τις τρεις περιστροφικές κινήσεις ελεύθερες και εμποδίζοντας όλες τις μεταφορικές.

Στο σχήμα 2.10 φαίνονται οι σύνθετες αρθρώσεις.



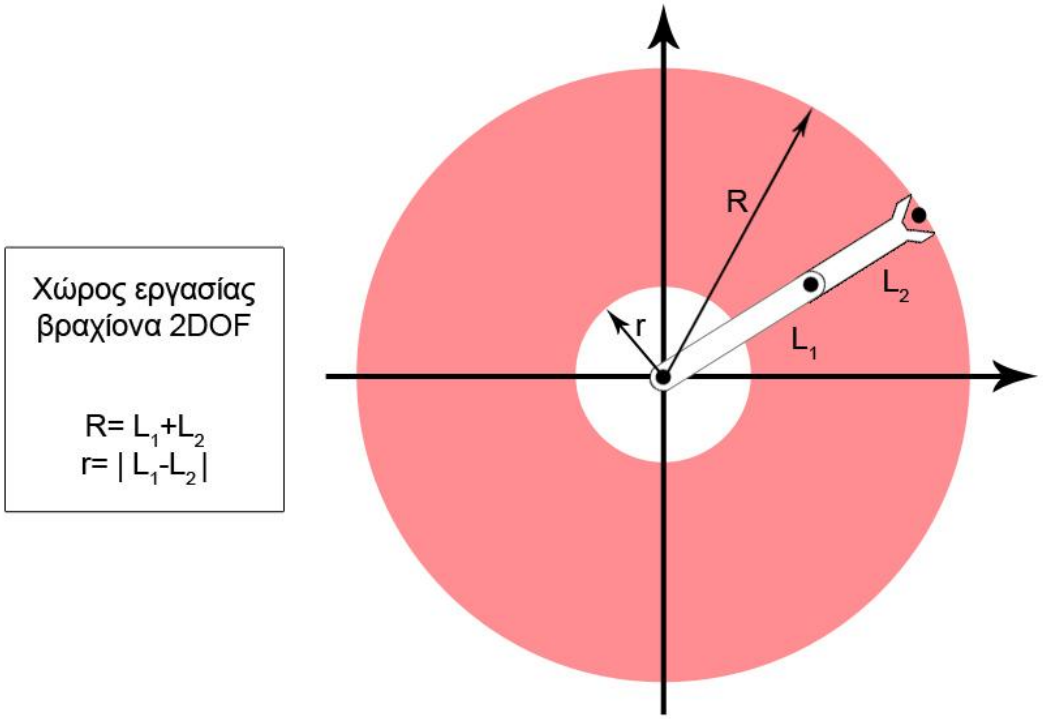
Σχήμα 2.10: Σύνθετες αρθρώσεις

2.1.5 Ρομποτικός χώρος εργασίας

Ως **χώρος εργασίας** ενός βραχίονα ορίζεται ο μέγιστος χώρος τον οποίο μπορεί να σαρώσει το τελικό σημείο δράσης του, σε ένα τρισορθογώνιο σύστημα αξόνων. Το μέγεθος και η γεωμετρική μορφή αυτού του χώρου εξαρτώνται από την κατασκευαστική δομή του βραχίονα, δηλαδή από την ποσότητα και το είδος των αρθρώσεων, καθώς και από τα μήκη των συνδέσμων από τα οποία αποτελείται.

Επίσης ο χώρος εργασίας ενός ρομπότ μπορεί να διακριθεί σε δύο μέρη, στον **προσβάσιμο** χώρο εργασίας και στο **επιδέξιο** χώρο εργασίας. Προσβάσιμος χώρος εργασίας είναι ο γεωμετρικός τόπος των σημείων, όπου το τελικό σημείο δράσης του ρομποτικού μηχανισμού μπορεί να προσεγγίσει έστω και με έναν προσανατολισμό της κινηματικής αλυσίδας. Ενώ ο επιδέξιος χώρος εργασίας, ο οποίος είναι υποσύνολο του προσβάσιμου, είναι ο γεωμετρικός τόπος των προσεγγίσιμων σημείων με όλους τους δυνατούς προσανατολισμούς της κινηματικής αλυσίδας. Αξίζει να σημειωθεί ότι ο επιδέξιος χώρος εργασίας υφίσταται μόνο σε χειριστές από 3 βαθμούς κινητικότητας ή ελευθερίας και πάνω, οι οποίοι μπορούν να έχουν διαφορετικούς προσανατολισμούς.

Στο σχήμα 2.11 φαίνεται ο χώρος εργασίας ενός επίπεδου (planar) βραχίονα 2DOF, με 2 περιστροφικές αρθρώσεις και με μήκη συνδέσμων L_1 L_2, όπου ισχύει $L_1 > L_2$. Αυτός ο βραχίονας δεν έχει επιδέξιο χώρο εργασίας.



Σχήμα 2.11: Χώρος εργασίας για χειριστή 2DOF με περιστροφικές αρθρώσεις

2.1.6 Ταξινόμηση βιομηχανικών ρομπότ

Προκειμένου να επιτευχθεί η βέλτιστη επιλογή ενός ρομπότ για μια συγκεκριμένη εφαρμογή, είναι πολύ βασική η σύγκριση ρομπότ παρόμοιων ιδιοτήτων με βάση κάποια συγκεκριμένα κριτήρια. Τα κυριότερα κριτήρια, όπως αυτά αναφέρονται στη βιβλιογραφία, είναι: η αρχή λειτουργίας, η μέθοδος ελέγχου της κίνησης και η γεωμετρική διαμόρφωση.

Αρχή λειτουργίας: Ως προς την αρχή λειτουργίας τους, τα ρομπότ ταξινομούνται σε σταθερής στάσης (fixed stop) (μη σερβοελεγχόμενα ή ανοικτού βρόχου) και σε σερβοελεγχόμενα (servo controlled).

- **Ρομπότ σταθερής στάσης:** Το ρομπότ κατά τη μετακίνηση του έχει έλεγχο μόνο των σημείων στάσης, αλλά όχι έλεγχο των ενδιάμεσων σημείων της τροχιάς. Κάθε άξονας έχει ένα προκαθορισμένο (ή καθοριζόμενο από τον ελεγκτή κάθε φορά) σταθερό μηχανικό όριο στο κάθε άκρο της έκτασής του και μπορεί να σταματάει μόνο στα όρια αυτά. Ρομπότ αυτού του τύπου είναι εξειδικευμένων εφαρμογών και ως εκ τούτου όχι διαδεδομένα.

- **Σερβοελεγχόμενα ρομπότ:** Ο τύπος αυτός κινείται από σερβομηχανισμούς, όπως αυτούς στο κεφάλαιο 3. Ένα τέτοιο ρομπότ έχει τη δυνατότητα να κινείται μέσα από έναν πρακτικά άπειρο αριθμό σημείων κατά την εκτέλεση μίας προγραμματισμένης ακολουθίας. Το σύνολο των σημείων προκύπτει από το συνδυασμό των επιτεύξεων μετατοπίσεων των αρθρώσεων. Εάν, για παράδειγμα, ένα ρομπότ έχει τρεις αρθρώσεις και κάθε άρθρωση μπορεί να επιτύχει 100 διαφορετικές μετατοπίσεις, τότε το άκρο του χειριστή μπορεί να επιτύχει 106 διαφορετικές τοποθετήσεις. Τα ρομπότ αυτά είναι πιο ακριβά και πιο σύνθετα στη

λειτουργία, τον προγραμματισμό και τη συντήρηση από τα ρομπότ σταθερής στάσης.

Μέθοδος ελέγχου κίνησης: Σε σχέση με τη μέθοδο ελέγχου κίνησης, τα ρομπότ ταξινομούνται σε ρομπότ σημείου-προς-σημείο (point-to-point) και ρομπότ συνεχούς τροχιάς (continuous path).

- **Ρομπότ σημείου-προς-σημείο:** Με αυτή τη μέθοδο ελέγχου της κίνησης, ένα ρομπότ προγραμματίζεται για την εκτέλεση μια κίνησης από ένα σημείο σε ένα άλλο, χωρίς να προσδιορίζεται η τροχιά που ακολουθεί. Συνεπώς, η τροχιά του ρομπότ η ταχύτητα κατά τη μετακίνηση από τη μία θέση στην επόμενη, εν γένει δεν έχει σημασία, και συνήθως δεν προγραμματίζεται, αλλά αποτελεί εσωτερική λειτουργία του ελεγκτή. Αυτή η μέθοδος εφαρμόζεται σε όλα τα ρομπότ σταθερής στάσης και σε ορισμένα σερβοελεγχόμενα. Οι περισσότερες εργασίες χειρισμού αντικειμένων, αλλά και αρκετές εργασίες χειρισμού εργαλείων, εκτελούνται σύμφωνα με αυτή τη μέθοδο.

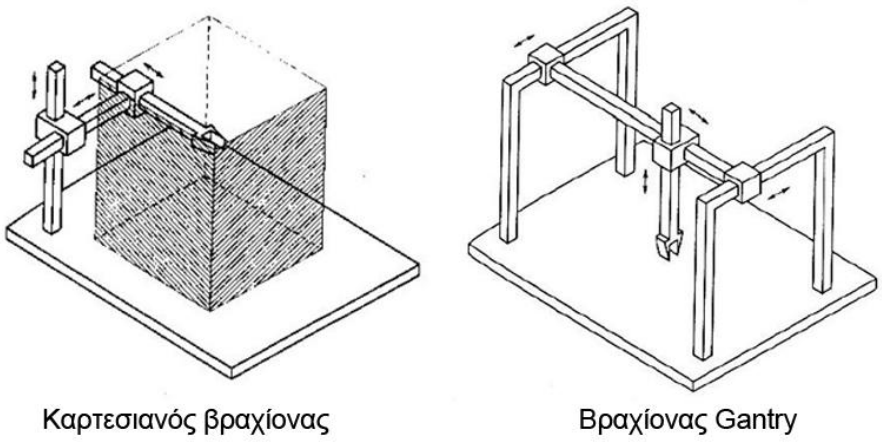
- **Ρομπότ συνεχούς τροχιάς:** Η μέθοδος ελέγχου συνεχούς τροχιάς είναι ένας τύπος ρομποτικού ελέγχου κατά τον οποίο το ρομπότ επαναλαμβάνει την κίνηση μέσα από διδαγμένα σημεία σε μικρή απόσταση μεταξύ τους, και τα οποία έχουν προγραμματισθεί σε μία σταθερή χρονική βάση κατά τη διάρκεια της διδασκαλίας. Τα σημεία διδασκαλίας καταγράφονται από τη μονάδα ελέγχου καθώς το ρομπότ οδηγείται μέσα από μία επιθυμητή τροχιά, διαβάζοντας τους κωδικοποιητές των αρθρώσεων, δηλ. τη θέση του κάθε άξονα σε σταθερή χρονική βάση. Ακολούθως, ένας αλγόριθμος επανάληψης επιχειρεί να επαναλάβει την κίνηση αυτή. Έλεγχος συνεχούς τροχιάς μπορεί επίσης να επιτευχθεί με την παρεμβολή μίας επιθυμητής καμπύλης τροχιάς μεταξύ των διδαγμένων σημείων. Η τροχιά του βραχίονα προγραμματίζεται με άμεσο τρόπο (καθοδήγηση μέσα από ενδιάμεσα σημεία - lead-through) ή υπολογίζει ο ελεγκτής τη διαδρομή μεταξύ διαδοχικών σημείων. Ορισμένες εργασίες χειρισμού αντικειμένων και εργαλείων εκτελούνται κατ' αυτόν τον τρόπο. Οι λειτουργίες συνεχούς τροχιάς μπορούν να εκτελεσθούν μόνο από σερβοελεγχόμενα ρομπότ.

Γεωμετρική διαμόρφωση: Στους περισσότερους βιομηχανικούς ρομποτικούς βραχίονες 6DOF, οι τρεις πρώτες αρθρώσεις συνήθως χρησιμοποιούνται για την τοποθέτηση του άκρου του βραχίονα σε ένα σημείο στο χώρο, ενώ οι τελευταίες αρθρώσεις σχηματίζουν τον καρπό (wrist), ο οποίος είναι υπεύθυνος για τον προσανατολισμό του άκρου ή του τελικού στοιχείου δράσης. Ανάλογα με το σύστημα συντεταγμένων των τριών πρώτων βαθμών ελευθερίας τα ρομπότ ταξινομούνται σε καρτεσιανά ή ορθογωνικά, κυλινδρικά, σφαιρικά ή πολικά, και αρθρωτά. Ωστόσο, ανάλογα με τη δομή και τη γεωμετρία του συνολικού τους μηχανισμού, τα ρομπότ μπορούν να χαρακτηρισθούν ως ανθρωπομορφικά, SCARA ή Gantry.

- **Καρτεσιανά ή ορθογωνικά:** Ένα καρτεσιανό ρομπότ (βραχίονας) προκύπτει από τη συναρμολόγηση τριών συνδέσμων και τριών πρισματικών αρθρώσεων στο κύριο σώμα του. Τα πλεονεκτήματα των καρτεσιανών ρομπότ είναι η μεγάλη ακρίβεια, η εύκολη αποφυγή εμποδίων και η μη επίδραση φορτίων βαρύτητας στην ακρίβεια, που συνεπάγεται ευκολία στον έλεγχο της κίνησης των αρθρώσεων. Ενώ τα βασικά μειονεκτήματά των καρτεσιανών ρομπότ είναι ο μεγάλος χώρος τον οποίο απαιτεί η δομή τους, ο σχετικά περιορισμένος χώρος εργασίας, η δυσκολία στη συνεργασία με άλλους ρομποτικούς βραχίονες και η

πολυπλοκότητα του μηχανικού τους σχεδιασμού για τις τρεις γραμμικές κινήσεις στους άξονες X-Y-Z. Επίσης στους καρτεσιανούς βραχίονες συγκαταλέγονται και οι βραχίονες **Gantry**. Τα ρομπότ αυτά είναι ορθογωνικά με τρεις βαθμούς (DOF) ελευθερίας κατ' ελάχιστο και η βασική διαφορά με τους καρτεσιανούς είναι ότι συνήθως τοποθετούνται στην οροφή ενός χώρου και η προσέγγιση του αντικειμένου γίνεται από πάνω. Ένα ρομπότ Gantry μπορεί να κινηθεί κατά τους άξονες X και Y διανύοντας σχετικά μεγάλες αποστάσεις με υψηλές ταχύτητες, ενώ ταυτόχρονα παρέχει και πολύ υψηλό βαθμό ακρίβειας τοποθέτησης των αντικειμένων. Τα χαρακτηριστικά του ρομπότ Gantry περιλαμβάνουν μεγάλους χώρους εργασίας, ικανότητα ανύψωσης μεγάλων φορτίων, κινητή τοποθέτηση στην οροφή, και τη δυνατότητα και ευελιξία λειτουργίας σε ένα χώρο εργασίας ισοδύναμο με αυτόν πολλών ρομπότ δαπέδου.

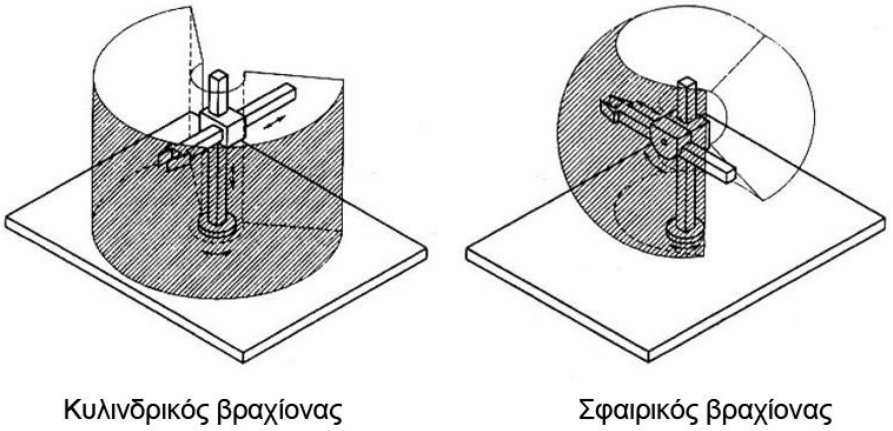
Στο σχήμα 2.12 φαίνεται η δομή και ο χώρος εργασίας ενός τυπικού καρτεσιανού βραχίονα και ενός Gantry.



Σχήμα 2.12: Καρτεσιανός βραχίονας και βραχίονας Gantry

- **Κυλινδρικά ρομπότ:** Τα κυλινδρικά ρομπότ είναι γνωστά και ως ρομπότ κυλινδρικών συντεταγμένων ή ρομπότ στήλης. Η δομή τους προκύπτει από τη συναρμολόγηση μιας βάσης, τουλάχιστον τριών αρθρώσεων και τουλάχιστον δύο αξόνων. Η μία άρθρωση είναι περιστρεφόμενη και συνδέει τη βάση με τον πρώτο άξονα. Οι άλλες δύο αρθρώσεις είναι πρισματικές και προσαρμοσμένες έτσι ώστε ο δεύτερος βραχίονας να κινείται πάνω κάτω και μπρος πίσω. Συνεπώς το ρομπότ κινείται σύμφωνα με ένα κυλινδρικό σύστημα συντεταγμένων, στο οποίο η θέση κάθε σημείου προσδιορίζεται συναρτήσει της γωνίας περιστροφής της βάσης, της ακτινικής διάστασης και του ύψους από το επίπεδο αναφοράς. Ο χώρος εργασίας του έχει κυλινδρική μορφή. Οι δυνατές κινήσεις αυτών των ρομπότ είναι η έκταση και η περιστροφή. Τα βασικά πλεονεκτήματα των κυλινδρικών ρομπότ είναι η πολύ μικρή εξάρτηση από τα φορτία βαρύτητας που δεν επηρεάζει την ακρίβεια του χειριστή και ο απλούστερος μηχανικός σχεδιασμός σε σχέση με τα καρτεσιανά ρομπότ. Στα μειονεκτήματά τους συμπεριλαμβάνονται η περιορισμένη συμβατότητα συνεργασίας με άλλους χειριστές σε κοινό χώρο εργασίας και η μικρότερη ακρίβεια και διακριτική ικανότητα σε σύγκριση με τα ορθογωνικά

ρομπότ. Στο σχήμα 2.13 φαίνεται η τυπική δομή ενός κυλινδρικού βραχίονα και ενός σφαιρικού βραχίονα, ο οποίος αναλύεται παρακάτω.



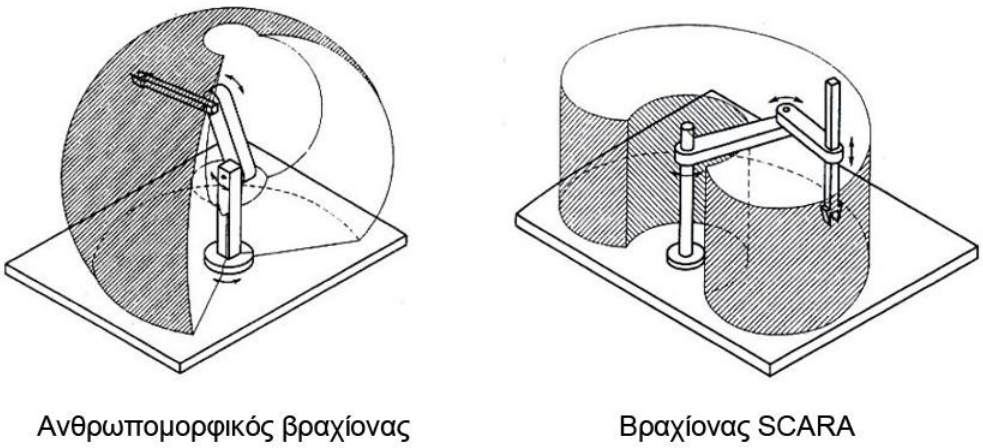
Σχήμα 2.13: Κυλινδρικός βραχίονας και σφαιρικός βραχίονας

- **Σφαιρικά ρομπότ:** Τα σφαιρικά ρομπότ ή ρομπότ σφαιρικών συντεταγμένων ή πολικά ρομπότ εργάζονται σε σφαιρικό χώρο εργασίας. Η δομή τους μοιάζει με αυτήν των κυλινδρικών ρομπότ και η διαφορά τους έγκειται στην προσθήκη μιας περιστροφικής άρθρωσης, προκειμένου ο δεύτερος άξονας να μπορεί να περιστρέφεται. Συνεπώς ο χώρος εργασίας που σχηματίζεται έχει τη μορφή ενός τμήματος σφαίρας. Τα βασικά πλεονεκτήματα των σφαιρικών ρομπότ είναι η μικρή πολυπλοκότητα της δομής τους, το σχετικά χαμηλό βάρος τους, η δυνατότητα συνεργασίας τους με άλλα ρομπότ και εργαλειομηχανές σε κοινό χώρο εργασίας, η καλή διακριτική ικανότητα τους, καθώς και το ότι η εκτέλεση πολλών κινήσεων απαιτεί μικρή διαδρομή των αρθρώσεων. Στα μειονεκτήματα τους συγκαταλέγονται η περιορισμένη δυνατότητα αποφυγής συγκρούσεων με εμπόδια, το μεγάλο σφάλμα τοποθέτησης εξαιτίας των περιστροφικών κινήσεων, το οποίο είναι ανάλογο της ακτίνας (δηλ. της απόστασης του εργαλείου από τη βάση), και ότι απαιτούνται μεγάλες και μεταβλητές ροπές στη δεύτερη και τρίτη άρθρωση, προκαλώντας έτσι πρόβλημα εξισορρόπησης.

- **Αρθρωτά ρομπότ:** Τα αρθρωτά ρομπότ εκτελούν κινήσεις όμοιες με εκείνες ενός ανθρώπινου χεριού (καρπού). Για το λόγο αυτό είναι γνωστά και ως **ανθρωπομορφικά**. Η δομή ενός αρθρωτού ρομπότ απαιτεί οι πρώτες τρεις αρθρώσεις του να είναι περιστροφικές. Ο πρώτος σύνδεσμος είναι τοποθετημένος σε μία περιστρεφόμενη βάση ενώ στην άλλη άκρη του υπάρχει μία άρθρωση, η οποία καλείται ώμος, που συνδέει το δεύτερο σύνδεσμο. Η επόμενη άρθρωση καλείται αγκώνας και συνδέει τον τρίτο σύνδεσμο. Επομένως υπάρχουν τρεις κινούμενοι άξονες. Η κίνηση αυτή των αξόνων σχηματίζει έναν σφαιρικό χώρο εργασίας. Τα κύρια πλεονεκτήματα των αρθρωτών ρομπότ είναι η ευελιξία προσέγγισης πάνω ή κάτω από ένα αντικείμενο και η συμβατότητα συνεργασίας με άλλα ρομπότ σε κοινό χώρο εργασίας. Τα βασικά μειονεκτήματά τους είναι η χαμηλή διακριτική ικανότητα και ακρίβεια, συνεπώς και μεγάλο σφάλμα τοποθέτησης, η περιορισμένη δυνατότητα αποφυγής εμποδίων, οι μεγάλες ροπές

αδρανείας και επιδράσεις φορτίων βαρύτητας, που προκαλούν δυναμική αστάθεια, αλλά και το γεγονός ότι δημιουργείται πρόβλημα εξισορρόπησης εξαιτίας των μεγάλων και μεταβλητών ροπών στις αρθρώσεις. Στους αρθρωτούς βραχίονες συγκαταλέγονται και οι βραχίονες **SCARA**. Το ρομπότ τύπου SCARA (Selective Compliance Arm for Assembly - Ρομποτικός Βραχίονας Συναρμολόγησης με Επιλεκτική Συμμόρφωση), είναι ένας ρομποτικός σχηματισμός οριζόντιας περιστροφής ο οποίος σχεδιάστηκε στο Πανεπιστήμιο Yamamachi της Ιαπωνίας. Είναι κατά βάση ανθρωπομορφική δομή (αρθρωτός βραχίονας), με τέσσερις ή πέντε βαθμούς ελευθερίας και λειτουργεί σε ένα οριζόντιο επίπεδο. Έχει συνήθως δύο ή τρεις οριζόντιες σερβοελεγχόμενες αρθρώσεις (ώμο, αγκώνα, και ορισμένες φορές καρπό), και έναν κάθετο, σερβοελεγχόμενο ή μη άξονα. Η συγκεκριμένη γεωμετρία παρέχει μεγάλη δυσκαμψία σε κατακόρυφη φόρτιση και ελαστικότητα σε οριζόντια. Βασικό μειονέκτημα όμως, είναι το γεγονός ότι η ακρίβεια τοποθέτησης του καρπού μειώνεται με την αύξηση της απόστασης του, από τον άξονα της πρώτης άρθρωσης.

Στο σχήμα 2.14 φαίνεται η τυπική δομή ενός ανθρωπομορφικού βραχίονα και ενός βραχίονα SCARA.



Σχήμα 2.14: Ανθρωπομορφικός βραχίονας και βραχίονας SCARA

2.1.7 Ωφέλιμο φορτίο, επαναληψιμότητα, ακρίβεια

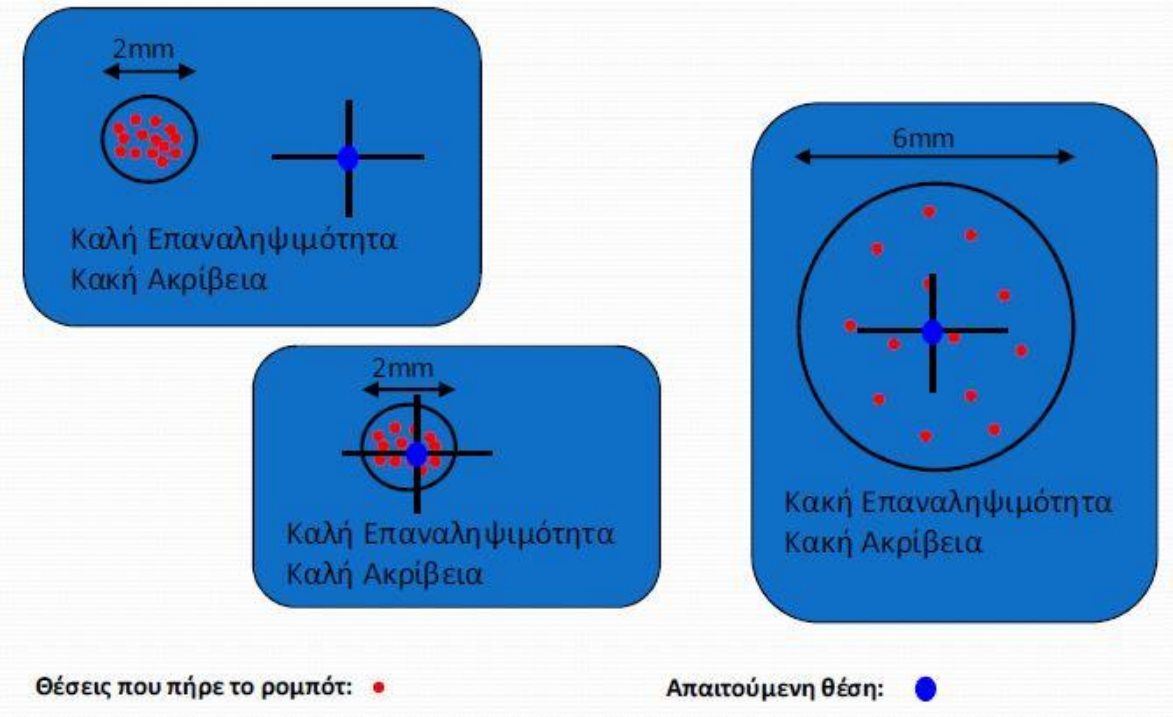
Τρία πολύ σημαντικά μεγέθη ενός βιομηχανικού βραχίονα είναι το ωφέλιμο φορτίο, η επαναληψιμότητα και η ακρίβεια. Πιο συγκεκριμένα:

- **Ωφέλιμο φορτίο:** Είναι το μέγιστο βάρος το οποίο μπορεί να σηκώσει το άκρο του βραχίονα. Το προδιαγραφόμενο αυτό φορτίο μπορεί να διαφοροποιείται με την ταχύτητα που κινείται ο βραχίονας.

- **Επαναληψιμότητα:** Εκφράζει τη δυνατότητα του βραχίονα να επιστρέφει σε ένα συγκεκριμένο σημείο μετά από αρκετές επαναλήψεις. Στις επαναλήψεις αυτές είναι σχεδόν σίγουρο ότι θα υπάρχει απόκλιση στη θέση του βραχίονα, για το λόγο αυτό η επαναληψιμότητα δίνεται ως εύρος. Η διδαχή ενός ρομπότ με αλγόριθμους διόρθωσης τείνει να ελαχιστοποιήσει κατά πολύ την απόκλιση αυτή.

- **Ακρίβεια:** Είναι η ικανότητα του ρομπότ να προσεγγίζει με ακρίβεια το προγραμματισμένο σημείο. Η ακρίβεια εξαρτάται από τα συστήματα ελέγχου, τη μηχανολογική σύνδεση των μελών και την ακρίβεια των σερβοκινητήρων.

Στο σχήμα 2.15 φαίνεται συσχέτιση της επαναληψιμότητας και της ακρίβειας για ένα βραχίονα με επαναληψιμότητα 2mm.



Σχήμα 2.15: Επαναληψιμότητα και ακρίβεια

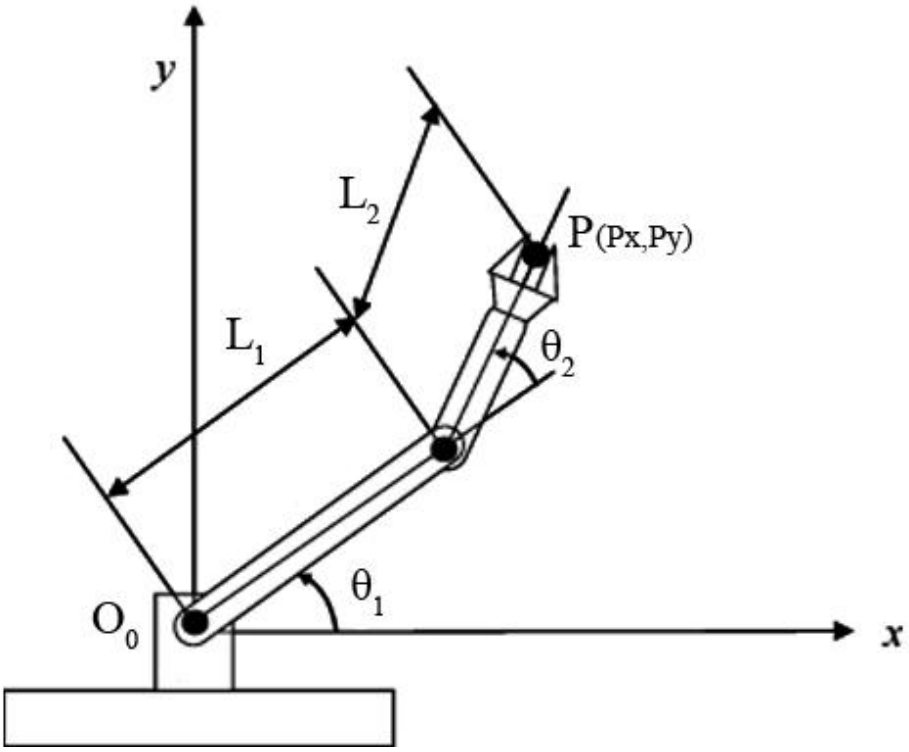
2.1.8 Κινηματική ανάλυση

Η κινηματική (kinematics) ασχολείται με τη βασική γεωμετρία των μηχανισμών και μελετά την κίνηση του ρομπότ στον χώρο χωρίς να εξετάζει τις δυνάμεις οι οποίες την προκαλούν. Τα δύο προβλήματα της κινηματικής των βραχιόνων είναι το ευθύ και το αντίστροφο κινηματικό πρόβλημα και τα οποία αναλύονται στη συνέχεια. Ένας αρθρωτός βραχίονας μπορεί να θεωρηθεί ως ένας μηχανισμός που παράγει μία θέση και ένα προσανατολισμό. Συνεπώς οι σχέσεις μεταξύ αυτών των ποσοτήτων και των μετατοπίσεων των αρθρώσεων, δηλαδή των μετρήσιμων και ελέγξιμων μεταβλητών, πρέπει να προσδιορισθούν. Ως θέση θεωρείται η θέση του άκρου του βραχίονα στο χώρο, ενώ ως προσανατολισμός, η κατεύθυνση της προσέγγισης του τελευταίου συνδέσμου ή του τελικού στοιχείου δράσης. Ενώ η θέση είναι εύκολα αντιληπτή και σε χώρους μεγαλύτερης διάστασης, η έννοια του προσανατολισμού, όπως και της περιστροφής, περιπλέκονται.

Στη συνέχεια αναλύεται ένας επίπεδος (planar) ρομποτικός βραχίονας με δύο βαθμούς ελευθερίας, όσοι δηλαδή απαιτούνται για την παραγωγή οποιασδήποτε θέσης μέσα σε ένα δεδομένο χώρο δύο διαστάσεων. Ενώ η επίτευξη του προσανατολισμού απαιτεί και έναν τρίτο βαθμό ελευθερίας. Ο βραχίονας που κατασκευάστηκε για τη συγκεκριμένη εργασία είναι όμοιος με αυτόν που αναλύεται παρακάτω.

2.1.9 Ευθύ κινηματικό πρόβλημα

Το ευθύ κινηματικό πρόβλημα ασχολείται με την εύρεση της θέσης και του προσανατολισμού του τελικού σημείου δράσης του βραχίονα, με δεδομένες τις μεταβλητές που ορίζουν τις κινήσεις των αρθρώσεών του, όπως για παράδειγμα τις γωνίες κίνησης των δύο περιστροφικών αρθρώσεων του βραχίονα 2DOF που φαίνεται στο σχήμα 2.14.



Σχήμα 2.16: Επίπεδος βραχίονας 2DOF

Συνεπώς οι σχέσεις που προκύπτουν για την ευθεία κινηματική ανάλυση του παραπάνω βραχίονα είναι οι εξής:

$$P_x = L_1 \cos\theta_1 + L_2 \cos(\theta_1+\theta_2)$$

$$P_y = L_1 \sin\theta_1 + L_2 \sin(\theta_1+\theta_2)$$

όπου τα $\cos\theta_i$ και $\sin\theta_i$,συμβολίζουν το συνημίτονο και το ημίτονο μίας γωνίας θ_i , αντίστοιχα.

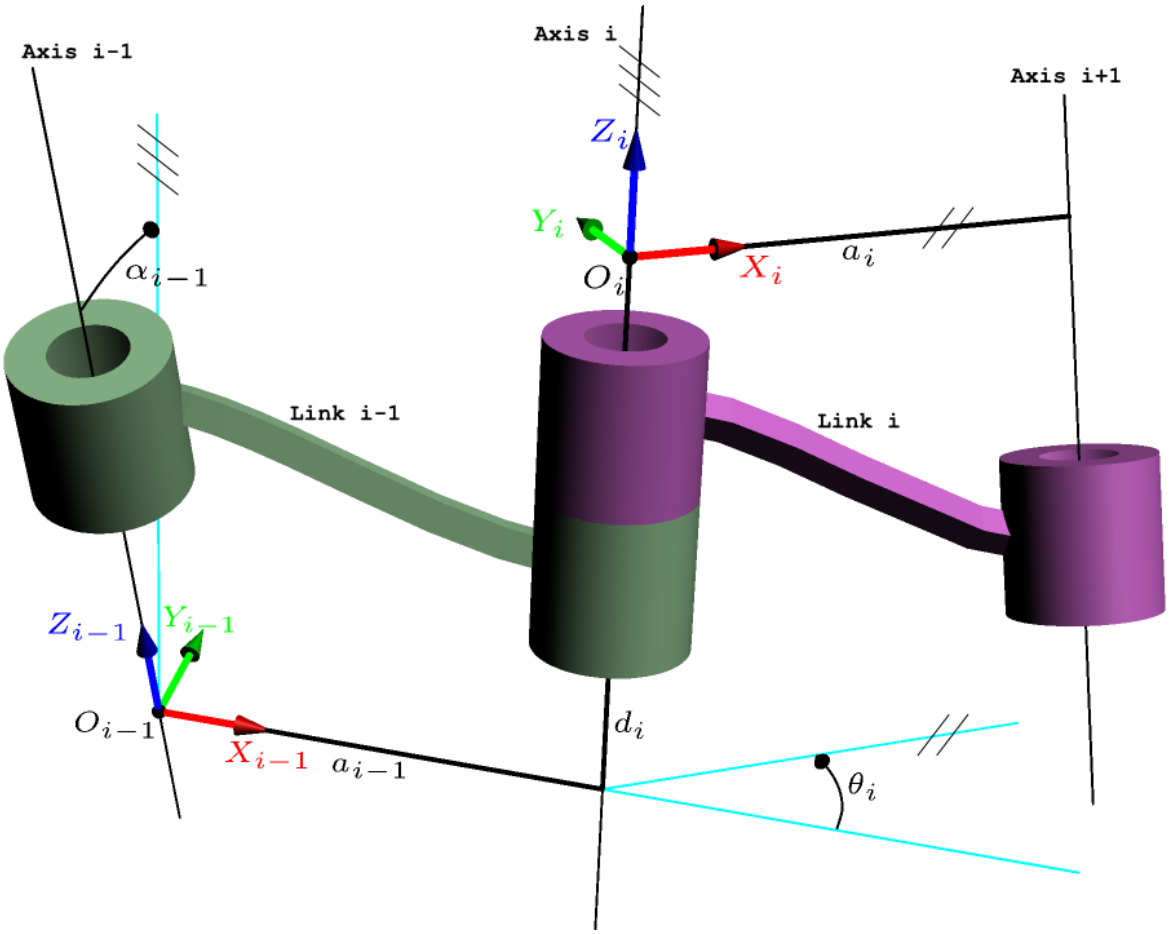
Γενικά, όπως έχει αναφερθεί, ένας βραχίονας αποτελείται από διαδοχικούς συνδέσμους οι οποίοι συνδέονται με αρθρώσεις, από τη βάση μέχρι το τελικό εργαλείο δράσης. Ένα κινηματικό μοντέλο που μπορεί να χρησιμοποιηθεί στην επίλυση του ευθέος κινηματικού προβλήματος των βραχιόνων, είναι αυτό των **Denavit** και **Hartenberg**. Η σύμβαση Denavit - Hartenberg, στην οποία χρησιμοποιούνται τέσσερις παράμετροι, είναι η πιο κοινή μέθοδος για την περιγραφή του κινηματικού προβλήματος ενός ρομπότ ν (ν=1,2,3…) αρθρώσεων. Ένα σύστημα συντεταγμένων επισυνάπτεται σε κάθε άρθρωση και καθορίζει τις D-H παραμέτρους. Βάσει της μεθόδου αυτής, επιλέγονται με συγκεκριμένο τρόπο τα συστήματα συντεταγμένων που είναι προσαρμοσμένα σε κάθε άρθρωση του ρομποτικού βραχίονα, εκτελώντας τα βήματα του αλγορίθμου, όπως

φαίνεται παρακάτω. Έτσι προσδιορίζεται η θέση και ο προσανατολισμός του άκρου εργασίας ως προς το ακίνητο σύστημα συντεταγμένων.

2.1.10 Αλγόριθμος Denavit - Hartenberg

Ο αλγόριθμος των Denavit - Hartenberg (D-H), είναι μια διαδικασία σύμφωνα με την οποία γίνεται η τοποθέτηση ορθοκανονικών και δεξιόστροφων Συστημάτων Συντεταγμένων στις αρθρώσεις του βραχίονα. Η συστηματική αυτή μέθοδος βοηθά στον ορισμό της σχετικής θέσης και του προσανατολισμού δύο διαδοχικών συνδέσμων. Βασικό στοιχείο της μεθόδου αποτελεί ο σωστός ορισμός του πλαισίου συντεταγμένων κάθε άρθρωσης, όπως φαίνονται στο σχήμα 2.17, ακολουθώντας τους παρακάτω τρεις κανόνες:

1. Επιλογή του άξονα Z_i κατά μήκος του άξονα της άρθρωσης i+1
2. Επιλογή του άξονα X_0 κατά μήκος της κοινής καθέτου των αξόνων Z_{i-1} και Z_i με φορά από την άρθρωση i στην άρθρωση i+1.
3. Επιλογή του άξονα Y_i , ώστε να έχουμε δεξιόστροφο σύστημα συντεταγμένων για το πλαίσιο i.



Σχήμα 2.17: Τα πλαίσια συντεταγμένων και οι παράμετροι D-H

Στη συνέχεια ακολουθεί ο προσδιορισμός των τεσσάρων παραμέτρων της σύμβασης D-H, οι οποίες είναι:

a_i = η απόσταση μεταξύ των Z_i και Z_{i+1} μετρούμενη κατά μήκος του X_i

α_i = η γωνία μεταξύ των Z_i και Z_{i+1} μετρούμενη ως προς τον X_i
d_i = η απόσταση μεταξύ των X_{i-1} και X_i μετρούμενη κατά μήκος του Z_i
θ_i = η γωνία μεταξύ των X_{i-1} και X_i μετρούμενη ως προς Z_i

Εφόσον έχουν ορισθεί τα πλαίσια των αρθρώσεων και οι παράμετροι της σύμβασης D-H, ακολουθώντας τα παρακάτω βήματα μπορούν να ορισθούν η θέση και ο προσανατολισμός του πλαισίου i ως προς το πλαίσιο i-1.

Βήμα 1: Περιστροφή του πλαισίου i-1 γύρω από τον άξονα Z_{i+1} κατά γωνία **θ_i**
Βήμα 2: Μετατόπιση **d_i** του πλαισίου i-1 κατά μήκος του άξονα Z_{i-1}
Βήμα 3: Μετατόπιση **a_i** (μήκος της κοινής καθέτου) κατά τον άξονα X_0
Βήμα 4: Περιστροφή γύρω από τον άξονα X_i κατά γωνία **α_i**

Συνεπώς από το βήμα 1 και βήμα 2 προκύπτει η μήτρα μετασχηματισμού:

$$T_{\Sigma_i}^{i-1} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Από το βήμα 3 και 4 προκύπτει:

$$T_i^{\Sigma_i} = \begin{bmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Οπότε συνολικά η μήτρα μετασχηματισμού η οποία δίνει τη σχετική θέση και τον προσανατολισμό του πλαισίου i ως προς το πλαίσιο i-1,είναι η ακόλουθη:

$$T_i^{i-1} = T_{\Sigma_i^{i-1}} * T_i^{\Sigma_i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & \alpha_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & \alpha_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Στο σχήμα 2.18 είναι ο πίνακας που περιέχει όλες τις απαραίτητες παραμέτρους για τη δημιουργία των ομογενών μετασχηματισμών μεταξύ των πλαισίων.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

Σχήμα 2.18: Πίνακας παραμέτρων Denavit-Hartenberg.

2.1.11 Αντίστροφο κινηματικό πρόβλημα

Κατά την ευθεία κινηματική ανάλυση προσδιορίστηκε η θέση και ο προσανατολισμός του τελικού σημείου δράσης $P=(P_x,P_y)$ με δεδομένες τις μεταβλητές των αρθρώσεων, θ_1 και θ_2. Στην περίπτωση όμως, κατά την οποία είναι δεδομένο το σημείο τοποθέτησης του άκρου $P=(P_x,P_y)$ του χειριστή, θα πρέπει να εφαρμοστεί η αντίστροφη κινηματική ανάλυση, συνεπώς να επιλυθεί το αντίστροφο κινηματικό πρόβλημα, προς εύρεση των μεταβλητών των αρθρώσεων, θ_1 και θ_2. Ενώ ο ευθύς υπολογισμός της θέσης του άκρου με δεδομένες τις μεταβλητές των αρθρώσεων είναι σχετικά απλός, η επίλυση του αντίστροφου κινηματικού προβλήματος είναι εν γένει δύσκολη και συχνά αδύνατη για χειριστές με πολλούς συνδέσμους. Ο λόγος για τον οποίο συμβαίνει αυτό, είναι επειδή οι κινηματικές εξισώσεις είναι κατά κανόνα μη γραμμικές και επομένως μία λύση δεν είναι πάντοτε εύκολα προσδιορίσιμη, ενώ σχεδόν ποτέ δεν είναι και μοναδική. Για τον επίπεδο χειριστή του σχήματος 2.19, ισχύει ότι κάθε σημείο $P=(P_x,P_y)$ στο χώρο εργασίας του, το οποίο απέχει από τη βάση O_0 απόσταση $r=\sqrt{P_x^2+P_y^2}$,ισχύει ότι:

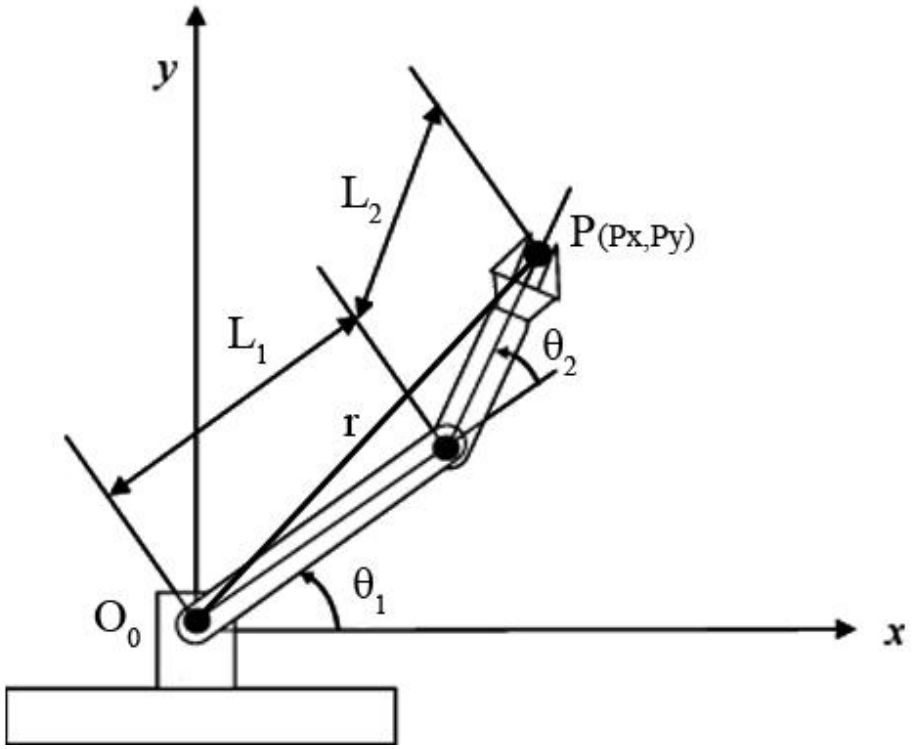
$$r^2=P_x^2+P_y^2=L_1^2+L_2^2+2L_1L_2\cos\theta_2 \Rightarrow \cos\theta_2=\frac{P_x^2+P_y^2-L_1^2-L_2^2}{2L_1L_2}=\sigma$$

Επομένως το ημίτονο της γωνίας θ_2 θα είναι:

$$\sin\theta_2=\pm\sqrt{1-\sigma^2}$$

Συνεπώς η γωνία θ_2 υπολογίζεται από τη σχέση:

$$\theta_2=\mathrm{ATAN2}(\pm\sqrt{1-\sigma^2}\ ,\ \sigma)$$

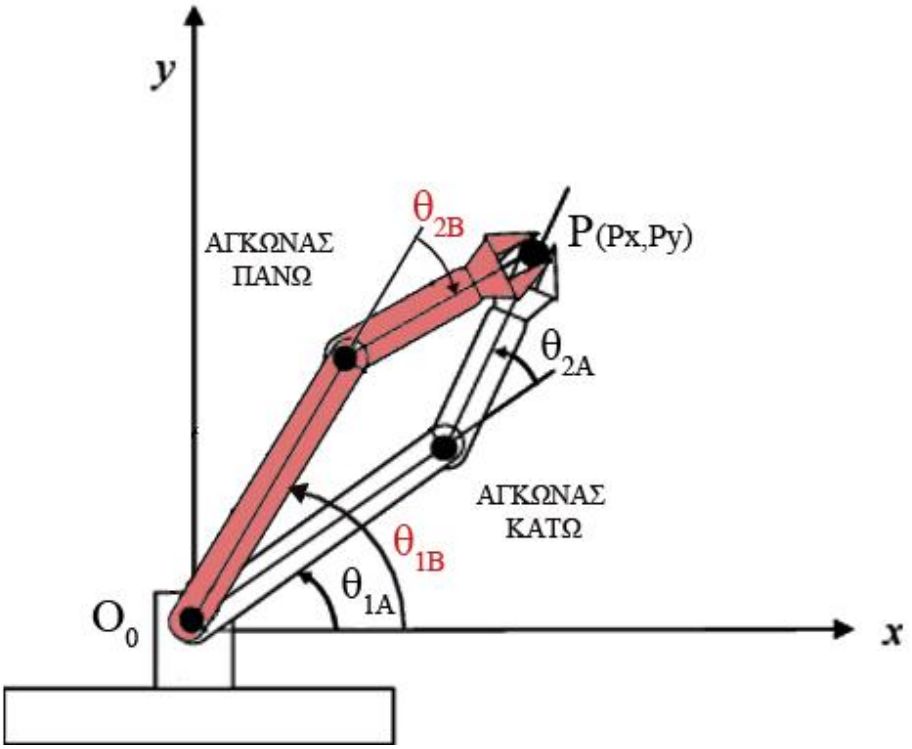


Σχήμα 2.19: Επίπεδος βραχίονας με $r^2=P_x^2+P_y^2$

Η συνάρτηση ATAN2 εκφράζει την αντίστροφη εφαπτομένη με δύο ορίσματα, το πρώτο εκ των οποίων αντιστοιχεί στην τιμή του ημιτόνου της γωνίας, ενώ το δεύτερο στην τιμή του συνημιτόνου. Κατ' αυτόν τον τρόπο είναι δυνατός ο μονοσήμαντος υπολογισμός της ζητούμενης γωνίας, καθότι είναι γνωστό και το τεταρτημόριο στο οποίο βρίσκεται. Αντίθετα, η γνωστή συνάρτηση ATAN ($\tan^{-1}$) δίνει δύο λύσεις που διαφέρουν κατά 180 μοίρες. Στο συγκεκριμένο πρόβλημα υπάρχουν, στη γενική περίπτωση, δύο λύσεις για τη γωνία θ_2 με ίσο μέτρο, αλλά αντίθετο πρόσημο. Επίσης για κάθε τιμή το θ_2 προκύπτει μία τιμή για το θ_1. Αυτό απορρέει από την επίλυση του συστήματος των εξισώσεων της ευθείας κινηματικής ανάλυσης από το οποίο προκύπτει η τιμή του θ_1:

$$\theta_1=\text{ATAN2}\,[-L_2\sin\theta_2 P_x+(L_1+L_2\cos\theta_2)P_y\ ,\ L_2\sin\theta_2 P_y+(L_1+L_2\cos\theta_2)P_x\ ]$$

Όπως προκύπτει από τα παραπάνω, στη γενική περίπτωση επίλυσης του αντίστροφου κινηματικού προβλήματος υπάρχουν δύο λύσεις για τις γωνίες θ_1 και θ_2. Αυτές οι δύο λύσεις, δηλαδή τα δύο ζεύγη (θ_1, θ_2), ονομάζονται ΑΓΚΩΝΑΣ ΠΑΝΩ (ELBOW UP) και ΑΓΚΩΝΑΣ ΚΑΤΩ (ELBOW DOWN) και ο αντίκτυπος στο βραχίονα φαίνεται στο σχήμα 2.20.



Σχήμα 2.20: Οι θέσεις του αγκώνα σύμφωνα με τα ζεύγη λύσεων για τις θ_1 και θ_2

Ωστόσο οι παραπάνω στις παραπάνω εξισώσεις υπάρχει ο περιορισμός $|\sigma| \leq 1$,αλλιώς δεν υπάρχει πραγματική λύση. Αυτό σημαίνει ότι το σημείο που έχει προγραμματιστεί να προσεγγίσει το άκρο του βραχίονα, βρίσκεται εκτός του χώρου εργασίας του.

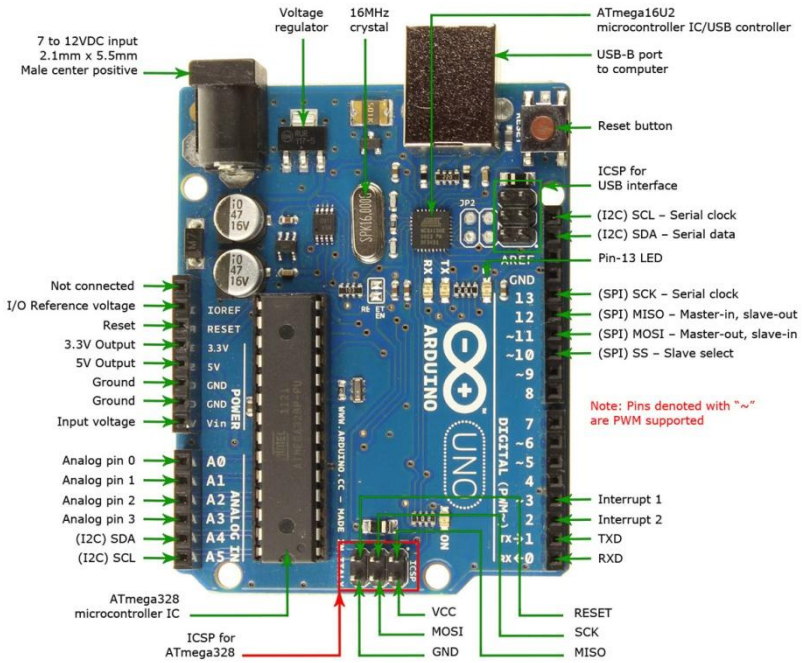
2.2 Η πλατμόρμα Arduino

Το Arduino είναι μια υπολογιστική πλατφόρμα, που αποτελείται από μια μητρική πλακέτα ανοικτού κώδικα με έναν ενσωματωμένο μικροελεγκτή και διάφορες εισόδους/εξόδους. Η σχεδίαση και η παραγωγή του ξεκίνησε το 2005 στην Ιβρέα της Ιταλίας, από τους **Massimo Banzi** και **David Cueartielles,** που ονόμασαν το σχέδιο Arduin of Ivrea, επηρεασμένοι από ένα μπαρ στην Ιβρέα, το οποίο είχε ονομασθεί έτσι προς τιμήν του ομώνυμου Βασιλιά της Ιταλίας το 1002 μ.Χ. (https://arduino.cc/). Το Arduino έχει τη δυνατότητα να αντλεί δεδομένα από το περιβάλλον μέσω αισθητήρων, να τα επεξεργάζεται και να αντιδρά σύμφωνα με τον προγραμματισμό του. Ο προγραμματισμός του μπορεί να γίνει με τη γλώσσα Wiring. Πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++. Προκειμένου να επιτευχθεί ο προγραμματισμός του, πραγματοποιείται σύνδεση με υπολογιστή, μέσω USB, στον οποίο έχει εγκατασταθεί το ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) του Arduino με άδεια χρήσης GPL.

Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων (stand-alone), αλλά και ελεγχόμενων από συμβατά προγράμματα υπολογιστή, διαδραστικών εφαρμογών. Από το 2005 έως σήμερα, υπάρχουν αρκετές συναρμολογημένες εκδόσεις του Arduino στο εμπόριο, όπως η Arduino Uno, Mini, Duemilanove, Mega, Due, Micro, Leonardo κ.α. Όμως υπάρχει και η δυνατότητα κατασκευής από οποιονδήποτε, καθώς το κύκλωμα της πλακέτας διατίθεται δωρεάν στην ιστοσελίδα του Arduino (https://www.arduino.cc/) με άδεια χρήσης Creative Commons.

2.2.1 Υλικό και τροφοδοσία πλατφόρμας Arduino Uno

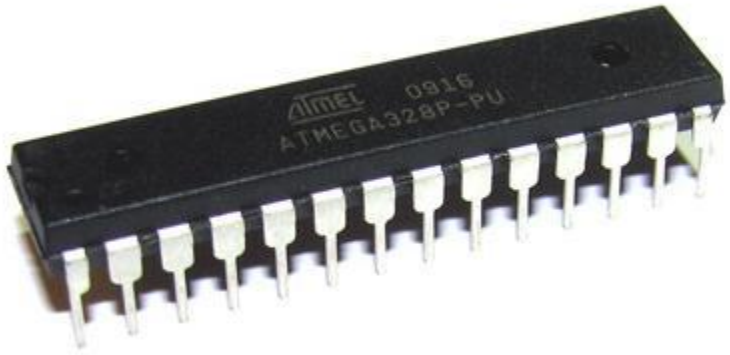
Η έκδοση του Arduino που χρησιμοποιείται στην παρούσα εργασία, είναι το Arduino Uno Rev3. Στο σχήμα 2.21. διακρίνονται τα εξαρτήματα του Arduino Uno, που θα αναλυθούν παρακάτω.



Σχήμα 2.21: Τα εξαρτήματα του Arduino Uno

2.2.2 Μικροελεγκτής και Μνήμη

Μία πλακέτα Arduino βασίζεται σε ένα μικροελεγκτή Atmel ATmega328, όπως αυτόν στο σχήμα 2.22, καθώς και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωσή του σε άλλα κυκλώματα.



Σχήμα 2.22: Μικροελεγκτής ATmega328

Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz. Το Arduino Uno μπορεί να συνδεθεί και να προγραμματιστεί από υπολογιστή, μέσω της θύρας USB που διαθέτει και με τη βοήθεια ενός μετατροπέα FTDI FT232 της θύρας από USB σε σειριακή. Γενικά όλες οι πλακέτες είναι προγραμματισμένες

μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για τη μετατροπή σημάτων επιπέδων RS-232 και TTL. Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- **SRAM**: Η μνήμη SRAM, 2Kb , είναι η ωφέλιμη μνήμη που χρησιμοποιείται από τα προγράμματα, προκειμένου να αποθηκεύσουν μεταβλητές, πίνακες κ.α. κατά την εκτέλεσή τους (runtime). Με τη διακοπή παροχής ρεύματος στο Arduino ή με reset, αυτή η μνήμη χάνει τα δεδομένα της.

- **EEPROM**: Η μνήμη EEPROM, 1Kb, μπορεί να χρησιμοποιηθεί για άμεση εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματα κατά την εκτέλεσή τους. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset οπότε είναι το ανάλογο του σκληρού δίσκου.

- **FLASH**: Η μνήμη Flash, 32Kb, χρησιμοποιείται για το firmware του Arduino και την αποθήκευση των προγραμμάτων που συντάσσονται στον υπολογιστή. Πιο συγκεκριμένα, τα 2Kb χρησιμοποιούνται από το firmware του Arduino, το οποίο έχει εγκατασταθεί ήδη κατά την κατασκευή του. Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς να χρειάζεται εξωτερικός προγραμματιστής (hardware programmer). Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή μέσω του IDE του Arduino. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset. Επίσης, η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime μέσα από τα προγράμματα, λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), ωστόσο έχει σχεδιαστεί μια βιβλιοθήκη που επιτρέπει την χρήση όσου χώρου περισσεύει (30Kb μείον το μέγεθος του προγράμματος σε μεταγλωττισμένη μορφή).

2.2.3 Pins - Είσοδοι/Έξοδοι

Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino, αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα ενώ αυτό εκτελείται. Επιπλέον υπάρχουν 14 θηλυκά pins, αριθμημένα από 0 ως 13, τα οποία μπορούν να λειτουργήσουν ως ψηφιακές είσοδοι και έξοδοι. Λειτουργούν στην τάση των 5V και καθένα μπορεί να παρέχει ή να δεχτεί ρεύμα εντάσεως έως και 40 mA. Ως ψηφιακή έξοδος, ένα από αυτά τα pins μπορεί να τεθεί από το πρόγραμμα σε κατάσταση HIGH ή LOW. Στην κατάσταση HIGH το Arduino διοχετεύει ρεύμα στο συγκεκριμένο pin, ενώ στην κατάσταση LOW, όχι. Με αυτόν τρόπο, παραδείγματος χάρη, μπορεί να αναβοσβήνει ένα LED. Επίσης, αν και εφόσον το pin οριστεί ως ψηφιακή είσοδος μέσα από το πρόγραμμα, υπάρχει η δυνατότητα να επιστραφεί η κατάσταση του pin, HIGH ή LOW, οπότε και εξετάζεται αν το pin διαπερνάται ή όχι από ρεύμα, όταν αυτό έχει συνδεθεί με κάποια εξωτερική συσκευή. Επί παραδείγματι, μπορεί να εξεταστεί η κατάσταση ενός διακόπτη συνδεδεμένου στο συγκεκριμένο pin. Μερικά από αυτά τα 14 pins, εκτός από ψηφιακές είσοδοι/έξοδοι έχουν και δεύτερη λειτουργία. Συγκεκριμένα:

- Τα pins 0 και 1 λειτουργούν ως RX και TX της σειριακής θύρας, όταν αυτό απαιτείται από το πρόγραμμα. Κατά αυτόν τον τρόπο, αν το πρόγραμμα στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB, αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1).

- Τα pins 2 και 3 λειτουργούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Συγκεκριμένα μπορούν να ρυθμιστούν μέσα από το πρόγραμμα, ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

- Τα pins 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation). Στην παρούσα εργασία χρησιμοποιείται η τεχνική PWM για την οδήγηση των σερβοκινητήρων, οπότε θα αναλυθεί παρακάτω.

Στην απέναντι πλευρά του Arduino υπάρχει μια σειρά από 6 pins με τη σήμανση ANALOG IN, αριθμημένα από το 0 ως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στον μικροελεγκτή. Για παράδειγμα, ένα από αυτά τα pins μπορεί να τροφοδοτηθεί με μια τάση, η οποία, με τη χρήση ενός ποτενσιόμετρου, μπορεί να κυμανθεί από 0V ως μια τάση αναφοράς Vref. Η προκαθορισμένη τάση είναι στα 5V. Μέσα από το πρόγραμμα μπορεί να διαβάζεται η τιμή του pin ως ένας ακέραιος αριθμός ανάλυσης 10-bit, από 0, όταν η τάση στο pin είναι 0V, μέχρι 1023, όταν η τάση στο pin είναι 5V. Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή μεταξύ 2 και 5V, τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτηθεί το pin AREF με 3.3V και στην συνέχεια διαβαστεί κάποιο pin αναλογικής εισόδου στο οποίο εφαρμόζεται τάση 1.65V, το Arduino θα επιστρέψει την τιμή 512. Επιπροσθέτως, το καθένα από τα 6 αυτά pins, με κατάλληλη εντολή μέσα από το πρόγραμμα μπορεί να μετατραπεί σε ψηφιακό pin εισόδου/εξόδου, όπως τα 14 που βρίσκονται στην απέναντι πλευρά και τα οποία περιγράφηκαν πριν. Σε αυτή την περίπτωση τα pins μετονομάζονται από 0~5 σε 14~19 αντίστοιχα.

2.2.4 Κουμπιά και LEDs

Στην πλακέτα του Arduino, υπάρχουν ενσωματωμένα τέσσερα μικροσκοπικά LED επιφανειακής στήριξης και ένας πιεζοηλεκτρικός διακόπτης micro-switch. Ο διακόπτης λειτουργεί ως κουμπί για reset. Για το λόγο αυτό έχει και την αντίστοιχη σήμανση. Το ένα εκ των τεσσάρων LED, με την ένδειξη power, βρίσκεται προκειμένου να δείχνει αν στο Arduino παρέχεται ρεύμα και λειτουργεί. Τα δύο εκ των τεσσάρων LED, με σήμανση Tx και Rx αντίστοιχα, χρησιμοποιούνται ως ένδειξη λειτουργίας του σειριακού interface, καθώς ανάβουν όταν το Arduino στέλνει ή λαμβάνει (αντίστοιχα) δεδομένα μέσω USB. Τα LED αυτά ελέγχονται από τον ελεγκτή Serial-over-USB και συνεπώς δεν λειτουργούν όταν η σειριακή επικοινωνία γίνεται αποκλειστικά μέσω των ψηφιακών pins 0 και 1.

Τέλος, υπάρχει το LED με την σήμανση L. Αυτό είναι ένα LED συνδεδεμένο από κατασκευής με το ψηφιακό pin 13, προκειμένου να είναι εφικτή η άμεση δοκιμή της πλακέτας με ένα πρόγραμμα που ελέγχει τη λειτουργία ενός LED. Παραδείγματος χάρη,

μπορεί ένα πρόγραμμα να αναθέτει την τιμή HIGH ή LOW στο LED και αυτό να αναβοσβήνει αντίστοιχα.

2.2.5 Τροφοδοσία του Arduino

Το Arduino μπορεί να τροφοδοτηθεί με δύο βασικούς τρόπους, όπως φαίνονται στο σχήμα 2.23. Ο πρώτος τρόπος είναι μέσω της θύρα USB, ενώ είναι συνδεδεμένο με υπολογιστή και επομένως η παρεχόμενη τάση είναι 5V. Και ο δεύτερος τρόπος είναι από εξωτερική τροφοδοσία μέσω του θηλυκού φις 2,1mm που διαθέτει. Η εξωτερική τροφοδοσία προτείνεται να είναι από 7 ως 12V και μπορεί να προέρχεται από ένα κοινό μετασχηματιστή AC/DC, από μπαταρίες ή οποιαδήποτε άλλη πηγή DC.

Σχήμα 2.23: Τρόποι τροφοδοσίας του Arduino

2.2.6 Τροφοδοσία συνδεδεμένων εξαρτημάτων

Στο Arduino είναι δυνατόν να συνδεθούν διάφορα εξαρτήματα, τα οποία απαιτούνται από το εκάστοτε σχέδιο και πρόγραμμα, όπως LEDs, αισθητήρες, κινητήρες και άλλα. Η παροχή ρεύματος σε αυτά τα εξαρτήματα μπορεί να γίνει με δύο τρόπους. Ο πρώτος τρόπος είναι απευθείας από το Arduino, μέσω των κατάλληλων pins που διαθέτει. Ενώ ο δεύτερος τρόπος είναι με εξωτερική τροφοδοσία από μετασχηματιστή ή μπαταρίες, μέσω ενός εξωτερικού κυκλώματος. Το κύκλωμα αυτό μπορεί να είναι τυπωμένο σε πλακέτα ή σε breadboard. Ο συγκεκριμένος τρόπος χρησιμοποιείται στην παρούσα εργασία και θα αναλυθεί παρακάτω. Στην περίπτωση που ακολουθείται ο πρώτος τρόπος τροφοδοσίας, χρησιμοποιείται η συστοιχία των pins με τη σήμανση POWER η οποία φαίνεται στο σχήμα 2.24. Η λειτουργία του καθενός από αυτά τα pins είναι ως εξής:

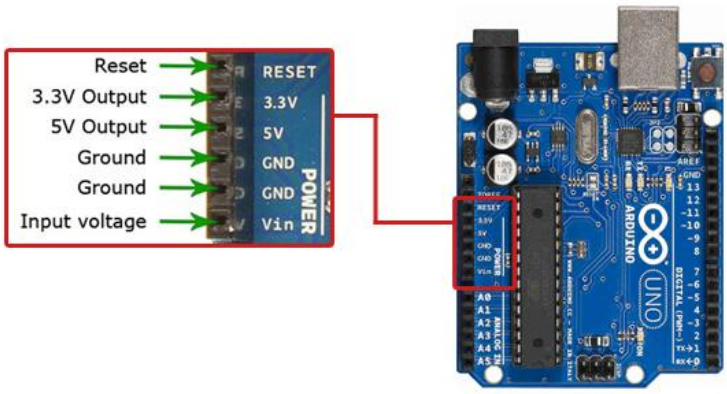
- Το pin με την ένδειξη RESET έχει ως αποτέλεσμα την επανεκκίνηση (reset) του Arduino, όταν γειωθεί σε οποιοδήποτε από τα 3 pin με την ένδειξη GND.

- Το pin με την ένδειξη 3.3V παρέχει τάση 3.3V σε οποιοδήποτε συνδεδεμένο εξάρτημα με αυτό το pin. Η τάση αυτή παράγεται από τον ελεγκτή Serial-over-USB και η μέγιστη παρεχόμενη ένταση είναι μόλις 50mA.

- Το pin με την ένδειξη 5V παρέχει τάση 5V σε οποιοδήποτε συνδεδεμένο εξάρτημα σε αυτό. Ανάλογα με τον τρόπο τροφοδοσίας του ίδιου του Arduino, η τάση αυτή προέρχεται είτε άμεσα από την θύρα USB, που ούτως ή άλλως λειτουργεί στα 5V, είτε από την εξωτερική τροφοδοσία αφού αυτή περάσει από ένα ρυθμιστή τάσης προκειμένου να σταθμιστεί στα 5V.

- Τα επόμενα δύο pins με την ένδειξη GND είναι γειώσεις.

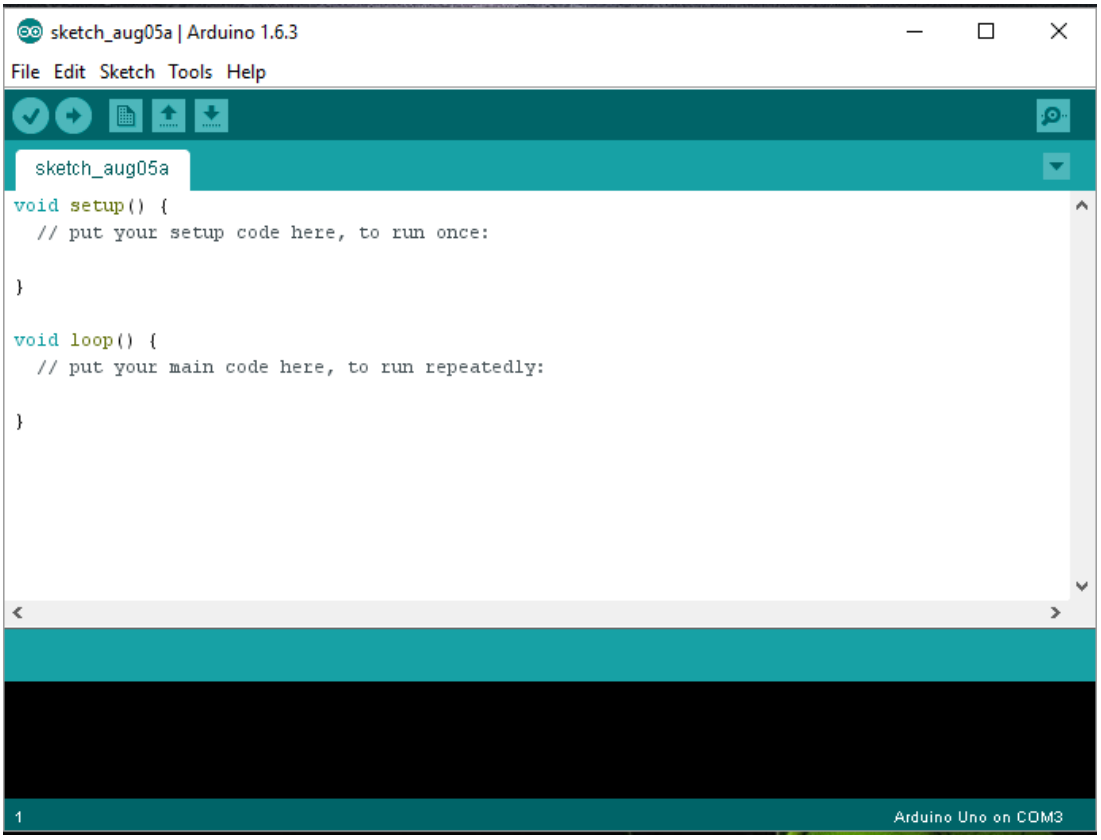
- Το τελευταίο pin με την ένδειξη Vin έχει διπλή λειτουργία. Η μία λειτουργία του αποτελεί έναν δευτερεύοντα τρόπο εξωτερικής τροφοδοσίας του Arduino, αν και εφόσον συνδυαστεί με το διπλανό pin γείωσης, στην περίπτωση που δεν παρέχεται τροφοδοσία από το φις των 2.1mm. Στην περίπτωση όμως που υπάρχει συνδεδεμένη εξωτερική τροφοδοσία (7~12V), μέσω του φις, τότε η άλλη λειτουργία του pin αυτού είναι η παροχή τάσης σε εξαρτήματα που είναι συνδεδεμένα σε αυτό. Η παρεχόμενη, αυτή, τάση είναι η πλήρης τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης, όπως συμβαίνει με το pin των 5V, που περιγράφηκε πριν.



Σχήμα 2.24: Συστοιχία των pins με σήμανση POWER

2.2.7 Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino

Όλα τα απαραίτητα εργαλεία για τον προγραμματισμό του Arduino παρέχονται από την ιστοσελίδα **https://www.arduino.cc**, δωρεάν. Το περιβάλλον ανάπτυξης (IDE) του Arduino, όπως φαίνεται στο σχήμα 2.25, είναι μία πολυπλατφορμική εφαρμογή προγραμματισμένη με Java και βασίζετε στο περιβάλλον της γλώσσας προγραμματισμού Processing (http://processing.org/). Περιλαμβάνει έναν πρακτικό επεξεργαστή κώδικα (editor) για τη συγγραφή προγραμμάτων, τον compiler για τη μεταγλώττισή τους, τη σειριακή οθόνη που παρακολουθεί την επικοινωνία της σειριακής θύρας (USB) για την απευθείας διαχείριση των προγραμμάτων και είναι ιδιαίτερα χρήσιμη για το debugging τους, καθώς και βιβλιοθήκες για την εύκολη διαχείριση των συνδεδεμένων εξαρτημάτων. Επίσης έχει την επιλογή της απευθείας φόρτωσης του μεταγλωττισμένου προγράμματος στην πλακέτα.



Σχήμα 2.25: Το IDE του Arduino

2.2.8 Γλώσσα προγραμματισμού

Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR, όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc. Λόγω της καταγωγής της από την C, στην γλώσσα του Arduino χρησιμοποιούνται ουσιαστικά οι ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, τους ίδιους τύπους δεδομένων και τους ίδιους τελεστές όπως και στην C.

Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino. Στο παράρτημα, που βρίσκεται στο τέλος της εργασίας, υπάρχουν οι δομές και λειτουργίες που χρησιμοποιούνται στη συγγραφή ενός προγράμματος(sketch) Arduino.

2.2.9 Δομή προγράμματος

Στη γλώσσα του Arduino κάθε πρόγραμμα αποτελείται από τις δύο βασικές ρουτίνες και έχει της εξής δομή:

```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών...

void setup()
{
  // ...
```

```
}

void loop()
{
  // ...
}

// Υπόλοιπες συναρτήσεις
```

Η βασική ρουτίνα **setup()** εκτελείται στην αρχή του προγράμματος και για μία μόνο φορά. Χρησιμοποιείται για τις αρχικοποιήσεις των μεταβλητών, τις δηλώσεις των pins (αν είναι είσοδος ή έξοδος) και τις αρχικοποιήσεις των βιβλιοθηκών. Ενώ η ρουτίνα **loop()** επαναλαμβάνει συνεχώς των κώδικα που είναι γραμμένος μέσα της. Οι τιμές των μεταβλητών των συναρτήσεων που καλούνται στη loop() έχουν τη δυνατότητα να αλλάζουν και το Arduino να ανταποκρίνεται ανάλογα.

2.2.10 Υποστηριζόμενες βιβλιοθήκες

Όπως αναφέρθηκε και προηγουμένως, το περιβάλλον του Arduino υποστηρίζει διάφορες βιβλιοθήκες, προκειμένου η διαχείριση και ο προγραμματισμός των πρόσθετων εξαρτημάτων και υλικού να γίνονται πιο εύκολα και πιο λειτουργικά. Αρκετές βασικές βιβλιοθήκες περιέχονται στην εγκατάσταση του IDE του Arduino, όπως η Servo, για την οδήγηση των σερβοκινητήρων, η Stepper, για την οδήγηση των βηματικών κινητήρων, και άλλες. Στην περίπτωση που χρειάζεται να χρησιμοποιηθεί κάποια άλλη βιβλιοθήκη, αρκεί να τοποθετηθεί μέσα στο φάκελο libraries, που βρίσκεται στο φάκελο του προγράμματος του Arduino στο σκληρό δίσκο του υπολογιστή.

Προκειμένου να χρησιμοποιηθεί μια βιβλιοθήκη, θα πρέπει να δηλωθεί στην αρχή του κώδικα με το tag: #include και τη βιβλιοθήκη. Για παράδειγμα η βιβλιοθήκη Servo, θα δηλωθεί ως εξής:

```
#include Servo
void setup()
{
        }
 void loop()
{
        }
```

Αξίζει να σημειωθεί ότι η κάθε βιβλιοθήκη που δηλώνεται στο Sketch, μεταφορτώνεται και στην πλακέτα, οπότε και καταλαμβάνει των αντίστοιχο χώρο. Για το λόγο αυτό, οι βιβλιοθήκες που είναι περιττές, θα πρέπει να σβήνονται από τον κώδικα του Sketch. Μερικές από τις βιβλιοθήκες που υποστηρίζει το Arduino είναι οι εξής:

Επικοινωνίας (δικτύωση και πρωτόκολλα):

- **Messenger** - για την επεξεργασία κειμένου με βάση τα μηνύματα από τον υπολογιστή.
- **PS2Keyboard** - διαβάζει χαρακτήρες από ένα πληκτρολόγιο PS2.
- **Simple Message System** - στέλνει μηνύματα μεταξύ Arduino και του υπολογιστή
- **SSerial2Mobile** - αποστολή μηνυμάτων κειμένου ή e-mail χρησιμοποιώντας ένα κινητό τηλέφωνο (μέσω εντολών AT μέσω σειράς λογισμικού)

- **X10** - Αποστολή σημάτων X10 μέσω γραμμών εναλλασσόμενου ρεύματος
- **Servo** – για τον έλεγχο κινητήρων τύπου Servo.

Ανίχνευσης:

- **Capacitive Sensing** - δύο ή περισσότερες ακίδες σε αισθητήρες πυκνωτή
- **Debounce** - για την ανάγνωση θορυβώδη ψηφιακών εισόδων (π.χ. από τα κουμπιά).

Εμφάνιση και LED:

- **Improved LCD library** - διορθώνει σφάλματα αρχικοποίησης LCD στην επίσημη Arduino LCD βιβλιοθήκη.
- **GLCD** – γραφικές ρουτίνες για LCD με βάση την KS0108 ή ισοδύναμο chipset.
- **LedControl** - για τον έλεγχο των LED ή επτά τμημάτων οθόνες με MAX7221 ή MAX7219.
- **LedDisplay** - τον έλεγχο της HCMS-29xx οθόνη LED.

Παραγωγή ήχου:

- **Tone** - αναπαράγει κύματα ήχου συχνότητας στο παρασκήνιο σε κάθε καρφίτσα του μικροελεγκτή.

Χρονοδιάγραμμα:

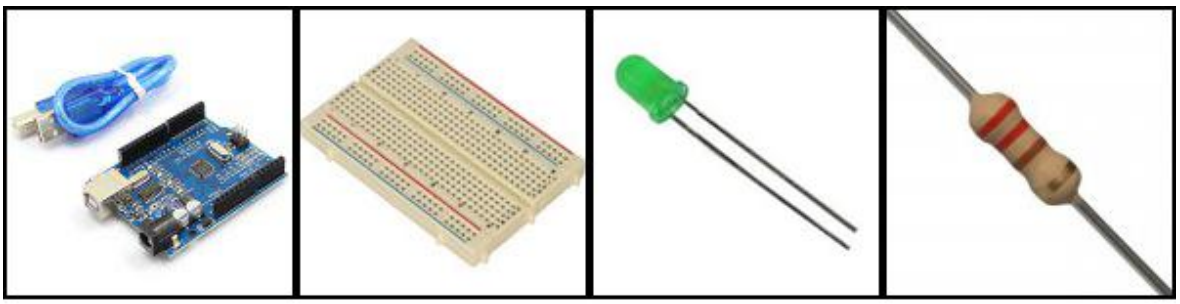
- **DateTime** - για την παρακολούθηση της τρέχουσας ημερομηνίας και ώρας.

2.2.11 Παράδειγμα - Εφαρμογή σε LED

Προκειμένου να κατανοηθούν τα παραπάνω, ακολουθεί ένα απλό παράδειγμα εφαρμογής του Arduino σε σύνδεση με ένα LED, το οποίο θα προγραμματιστεί να αναβοσβήνει με περίοδο 2 δευτερολέπτων.

Υλικά: Καταρχάς παρατίθενται τα υλικά που θα χρησιμοποιηθούν κατά την εφαρμογή.
- Πλακέτα Arduino Uno με καλώδιο σύνδεσης για υπολογιστή
- Breadboard
- LED
- Αντίσταση 220Ω

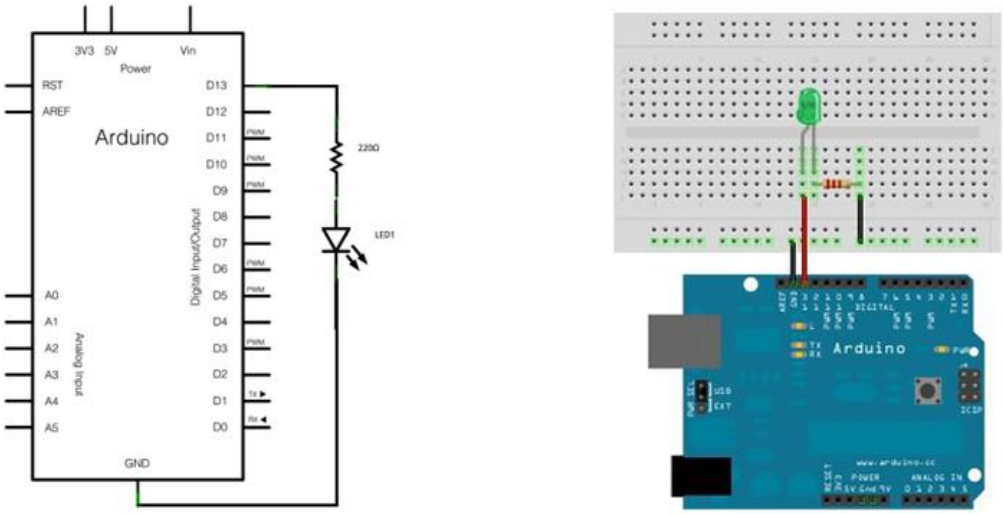
Επίσης θα χρειαστούν και κάποια καλώδια για τις συνδέσεις. Τα βασικά υλικά φαίνονται στο σχήμα 2.26.



Arduino Uno | Breadboard | LED | R 220Ω

Σχήμα 2.26: Τα υλικά της εφαρμογής

Κύκλωμα: Όπως φαίνεται στο σχήμα 2.27, το ένα άκρο της αντίστασης πρέπει συνδεθεί με το pin 13 της πλακέτας και το άλλο άκρο της να συνδεθεί με την άνοδο (μακρύ άκρο) του LED. Στη συνέχεια συνδέεται η κάθοδος του LED με τη γείωση της πλακέτας.

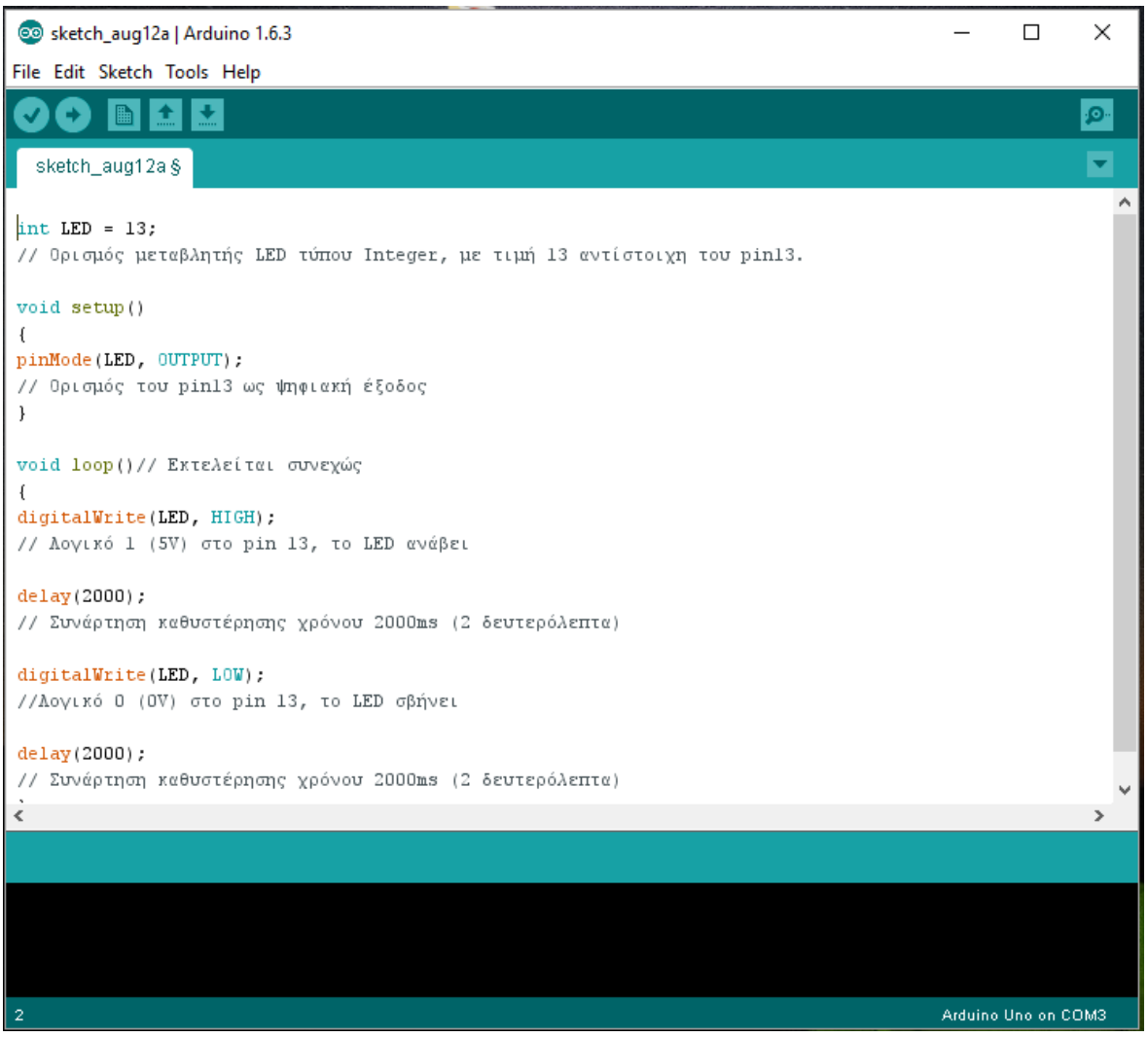


Θεωρητικό κύκλωμα

Προσομοίωση του κυκλώματος μέ το πρόγραμμα Fritzing

Σχήμα 2.27: Θεωρητικό κύκλωμα και η προσομοίωσή του με το πρόγραμμα Fritzing.

Προγραμματισμός: Εφόσον οι συνδέσεις των εξαρτημάτων και της πλακέτας με τον υπολογιστή, έχουν γίνει σωστά, ακολουθεί ο κώδικας της εφαρμογής, όπως φαίνεται στο σχήμα 2.28. Στη συνέχεια, αρκεί να μεταφορτωθεί το πρόγραμμα στην πλακέτα με το κατάλληλο κουμπί, ώστε να αρχίσει να αναβοσβήνει το LED.



Σχήμα 2.28: Κώδικας για LED που αναβοσβήνει

2.2.12 Επεκτάσεις πλακέτας - Shields

Η πλακέτα του Arduino έχει τη δυνατότητα επέκτασης με άλλες πλακέτες, οι οποίες στο εμπόριο ονομάζονται Shields (ασπίδες). Η πλακέτες αυτές έχουν σχεδιαστεί έτσι ώστε να εφαρμόζουν επάνω στην εκάστοτε πλακέτα Arduino. Για το λόγο αυτό το κάθε η κάθε έκδοση Arduino έχει και τα αντίστοιχα shields. Οι ασπίδες αυτές, που αποτελούνται από ενσωματωμένα κυκλώματα, παρέχουν στο χρήστη του Arduino, την ευκολία να διαχειρίζεται εξαρτήματα συνδεδεμένα στην πλακέτα, όπως κινητήρες, αισθητήρες και άλλα, αλλά και να επεκτείνει τις λειτουργίες της βασικής πλακέτας, όπως η σύνδεση με WiFi ή Bluetooth και άλλα. Μερικά από τα πιο δημοφιλή shields που κυκλοφορούν στο εμπόριο για το Arduino είναι:

- **Ethernet shield:** Δίνει στο Arduino την δυνατότητα να δικτυωθεί σε ένα LAN ή στο internet μέσω ενός τυπικού καλωδίου Ethernet.

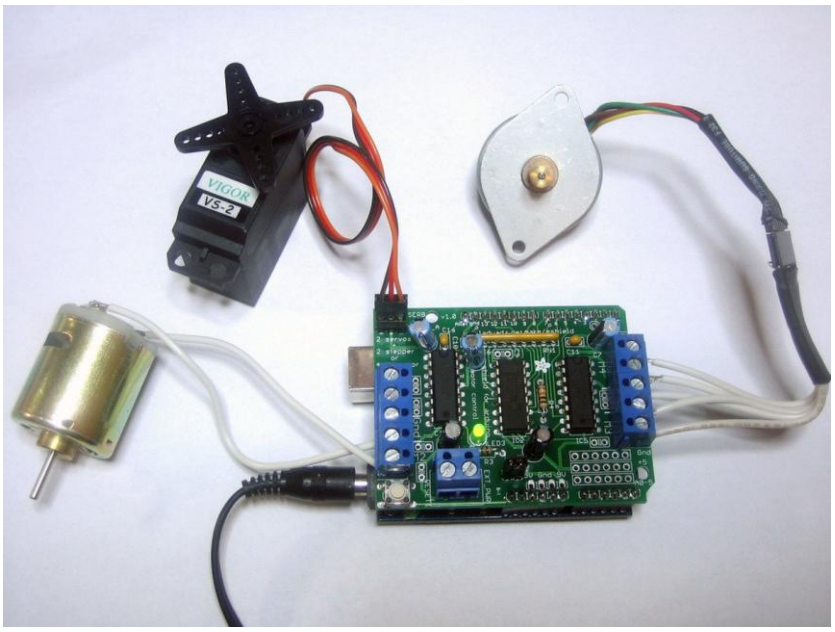
- **WiFi shield:** Δίνει στο Arduino την δυνατότητα να δικτυωθεί σε ένα LAN ή στο internet, χωρίς καλώδιο.

- **Wave shield:** Δίνει στο Arduino την δυνατότητα να παίζει ήχους/μουσική από κάρτες SD.

- **GPS shield:** Προσθέτει GPS δυνατότητες στο Arduino (εντοπισμό στίγματος).

- **Motor Shields**: Φαίνεται στο σχήμα 2.29. Καθιστά πιο εύκολη την οδήγηση μοτέρ διάφορων τύπων (απλά DC, servo, stepper κ.λπ.) από το Arduino.

- **ProtoShield:** Μια προσχεδιασμένη πλακέτα πρωτοτυποποίησης, συμβατή στις διαστάσεις του Arduino και χωρίς εξαρτήματα για κατασκευή ενός custom shield.



Σχήμα 2.29: Motor Shield επάνω σε Arduino Uno και διάφορα συνδεδεμένα μοτέρ

Αξίζει να σημειωθεί ότι οι ασπίδες αυτές περιέχουν και δικές τους βιβλιοθήκες, προκειμένου να καθιστούν πιο εύκολο και τον προγραμματισμό τους. Στην παρούσα εργασία δε χρησιμοποιείται κανένα Shield.

2.3 Σερβοκινητήρες

Οι σερβοκινητήρες είναι συστήματα κινητήρων, τα οποία ανήκουν στην κατηγορία των συστημάτων αυτομάτου ελέγχου κλειστού βρόγχου ή αλλιώς των συστημάτων ελέγχου με ανάδραση. Γενικά κάθε μηχανισμός που λειτουργεί με ανάδραση για τον έλεγχο κάποιας μεταβλητής, μπορεί να θεωρηθεί ως σερβομηχανισμός. Το σύστημα ανάδρασης στους σερβοκινητήρες χρησιμοποιείται σε συνδυασμό με ένα σερβομηχανισμό οδήγησης προκειμένου να είναι δυνατός ο ψηφιακός ή αναλογικός έλεγχος της ροπής, της ταχύτητας και της θέσης τους. Τα κύρια χαρακτηριστικά των σερβοκινητήρων είναι η υψηλή ροπή, η άμεση απόκριση στις εντολές του συστήματος ελέγχου και το μεγάλο εύρος ελέγχου της ταχύτητας. Η υψηλή ροπή, η οποία οφείλεται στο σύστημα οδοντωτών τροχών που περιέχει και λειτουργεί σαν μειωτήρας στροφών, είναι ικανή να διατηρεί ένα φορτίο, πολύ μεγαλύτερο από το βάρος του σερβοκινητήρα, σε δεδομένη θέση με μηδενική ταχύτητα. Επιπλέον οι εκκινητές των σερβοκινητήρων προσφέρουν τον ακριβέστερο έλεγχο της κίνησης, με αποτέλεσμα καθίστανται ικανοί να προσαρμόζουν τις σύνθετες εντολές και τα σχεδιαγράμματα κινήσεων εύκολα και άμεσα.

Τα δύο βασικά είδη σερβοκινητήρων είναι οι σερβοκινητήρες συνεχούς ρεύματος (DC servomotor) και οι σερβοκινητήρες εναλλασσόμενου ρεύματος, τριφασικοί ή μονοφασικοί (AC servomotor). Οι σερβοκινητήρες εναλλασσόμενου ρεύματος που είναι πιο

διαδεδομένοι, συνοδεύονται με τα εξής εξαρτήματα: ένα μετασχηματιστή, έναν ανορθωτή τάσης και έναν πυκνωτή. Ο μετασχηματιστής παίρνει την τάση του δικτύου AC και τη μετασχηματίζει με τη βοήθεια του ανορθωτή σε τάση DC κατάλληλου μεγέθους και έπειτα η τάση αποθηκεύεται στον πυκνωτή. Η χρήση των σερβοκινητήρων εναλλασσόμενου ρεύματος δεν συνιστάται σε συστήματα ελέγχου θέσης ή ταχύτητας λόγω της ροπής αδράνειας αλλά και των επιπρόσθετων απωλειών από τα εξαρτήματα τους. Ωστόσο οι σερβοκινητήρες συνεχούς ρεύματος χρησιμοποιούνται περισσότερο σε εφαρμογές όπως: σερβοσυστήματα καθορισμού θέσης (positioning), συστήματα ελέγχου ταχύτητας, συστήματα ελέγχου βραχίονα (π.χ. ρομπότ) και στο μοντελισμό (RC-servo). Στην παρούσα εργασία χρησιμοποιούνται σερβοκινητήρες μοντελισμού και ρομπότ, συνεχούς ρεύματος, όπως αυτόν στο σχήμα 2.30.

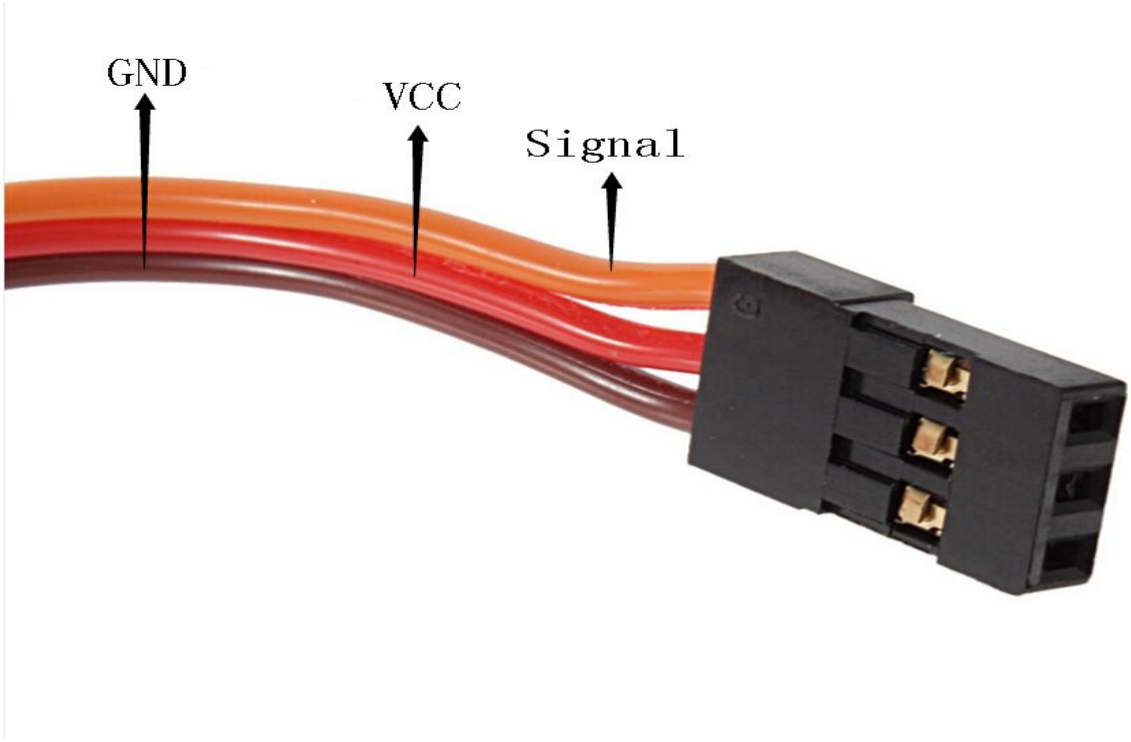
Σχήμα 2.30: Servo MG996R TowerPro

2.3.1 Λειτουργία σερβοκινητήρων

Προκειμένου να καταστεί κατανοητή η λειτουργία ενός σερβοκινητήρα, θα πρέπει να αναλυθεί το εσωτερικό του. Συνεπώς θα πρέπει να γίνουν γνωστά τα μέρη από τα οποία αποτελείται και το πώς αυτά συνεργάζονται έτσι ώστε να παράγεται το επιθυμητό αποτέλεσμα. Ένας σερβοκινητήρας (σέρβο ή servo) αποτελείται από ένα μικρό και απλό κινητήρα DC, ο οποίος είναι συνδεδεμένος με ένα σύστημα οδοντωτών τροχών (γρανάζια), ένα ποτενσιόμετρο κι ένα κύκλωμα ελέγχου, όπως φαίνεται στο σχήμα 2.31.

Σχήμα 2.31: Τα μέρη ενός σερβοκινητήρα

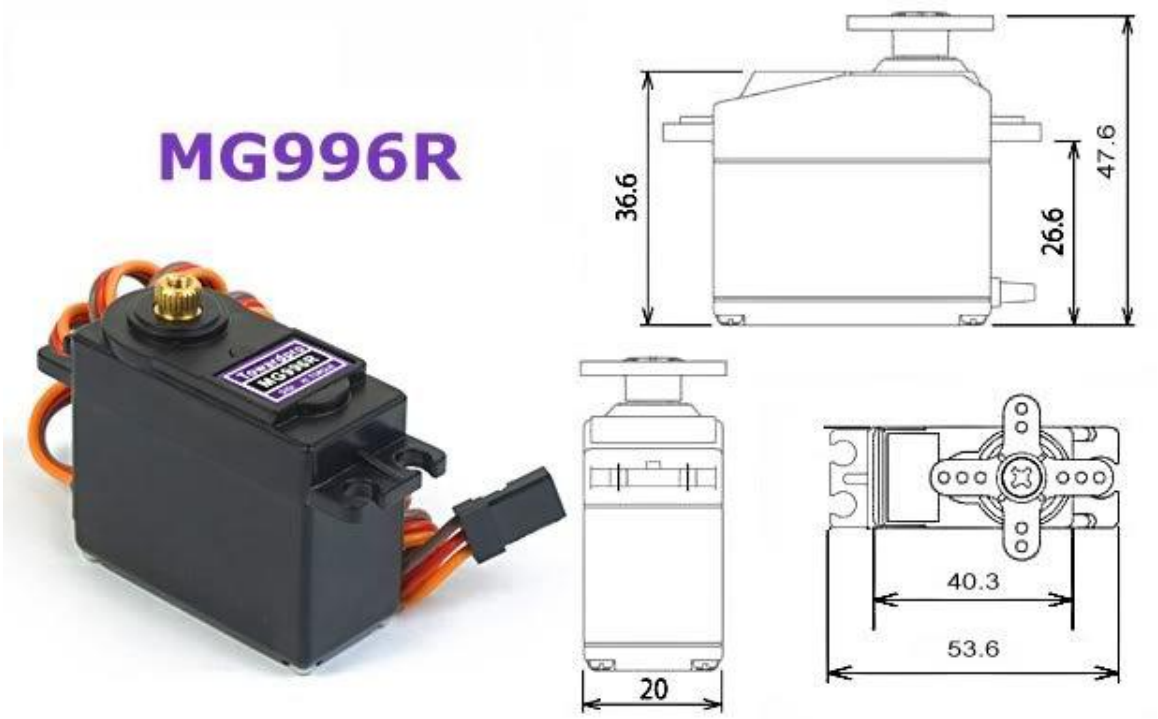
Ο σερβοκινητήρας έχει τρία καλώδια, όπως φαίνονται στο σχήμα 2.32, τα χρώματα των οποίων είναι προκαθορισμένα στο εμπόριο, ώστε να αποφευχθούν λάθη στις συνδέσεις. Το κόκκινο είναι για την παροχή τάσης (VCC), το μαύρο ή καφέ είναι η γείωση (GND) και το κίτρινο ή πορτοκαλί (Signal) είναι το καλώδιο σήματος, για την παροχή εντολών. Τα τρία αυτά καλώδια είναι συνδεδεμένα με το κύκλωμα ελέγχου. Στο κύκλωμα ελέγχου είναι συνδεδεμένος ο κινητήρας και το ποτενσιόμετρο. Καθώς ο κινητήρας περιστρέφεται, η αντίσταση του ποτενσιομέτρου μεταβάλλεται έτσι ώστε το κύκλωμα ελέγχου να έχει τη δυνατότητα να προσδιορίζει με ακρίβεια την ταχύτητα της κίνησης, την κατεύθυνση της περιστροφής και τη θέση που βρίσκεται ο άξονας. Όταν ο άξονας του κινητήρα είναι στην επιθυμητή θέση, η ισχύς που παρέχεται στον κινητήρα διακόπτεται. Η επιθυμητή θέση στέλνεται με τη μορφή ηλεκτρικών παλμών μέσω του καλωδίου σήματος. Αυτή είναι η τεχνική PWM, οποία θα αναλυθεί παρακάτω. Η ταχύτητα του κινητήρα είναι ανάλογη της διαφοράς μεταξύ της πραγματικής θέσης του και την επιθυμητή θέση. Έτσι, αν ο κινητήρας είναι κοντά στην επιθυμητή θέση, η ταχύτητα περιστροφής θα είναι χαμηλή, αλλιώς θα περιστραφεί με μεγαλύτερη ταχύτητα. Ο κινητήρας DC μπορεί να περιστρέφεται συνεχόμενα, εντούτοις οι περισσότεροι σερβοκινητήρες που παράγονται έχουν ένα συγκεκριμένο εύρος περιστροφής, συνήθως από 0 έως 180 μοίρες. Υπάρχει βέβαια και ένα μεγάλο ποσοστό σερβοκινητήρων, που κινούνται μεταξύ 0 και 120 μοίρες και επίσης υπάρχουν και αυτά που δεν έχουν όριο στην περιστροφή, τα λεγόμενα "continuous rotation servo".



Σχήμα 2.32: Τα καλώδια του σερβοκινητήρα

2.3.2 Χαρακτηριστικά του σερβοκινητήρα

Οι σερβοκινητήρες παράγονται με διάφορα κατασκευαστικά χαρακτηριστικά, τα οποία πρέπει να λαμβάνονται υπόψη κατά την επιλογή τους σε μια εφαρμογή. Καταρχάς, οι διαστάσεις ενός σερβοκινητήρα είναι πολύ σημαντικό κριτήριο για την επιλογή του. Συνεπώς ο κατασκευαστής οφείλει να παραθέτει το μηχανολογικό σχέδιο με διαστασιολόγηση, όπως φαίνεται στο σχήμα 2.33, για τον σερβοκινητήρα MG996R.



Σχήμα 2.33: Οι διαστάσεις του σερβοκινητήρα MG996R σε mm

Τα υπόλοιπα χαρακτηριστικά που πρέπει να λαμβάνονται υπόψη και δίνονται πάντα στην περιγραφή του προϊόντος είναι (οι τιμές αφορούν στο σερβοκινητήρα MG996R):

- **Rotation range:** 120º
  Οι μοίρες αντιστοιχούν στο εύρος περιστροφής του άξονα του σερβοκινητήρα.

- **Design:** Double ball bearing
  Σημαίνει ότι ο άξονας του σερβοκινητήρα μπορεί να κινείται και προς τις 2 κατευθύνσεις (60º, -60º)

- **Weight:** 55g
  Αυτό είναι το βάρος του σερβοκινητήρα.

- **Stall torque:** 9.4 kgF·cm (4.8 V), 11 kgF·cm (6 V)
  Είναι η ροπή του σερβοκινητήρα, για τη ελάχιστη τάση και την καλύτερη τάση, όταν η ταχύτητά του είναι μηδενική. Το 9.4 kgF·cm στα 4.8V, σημαίνει ότι ο σερβοκινητήρας, στο οποίο εφαρμόζεται τάση 4.8V, μπορεί να έχει στον άξονα τη δύναμη των 9.4 κιλών σε απόσταση ενός εκατοστού ή τη δύναμη του 1 κιλού σε απόσταση 9.8 εκατοστών κ.ο.κ. Αντίστοιχα συμβαίνει και με τις άλλες τάσεις.

- **Operating speed:** 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
  Είναι η μέγιστη ταχύτητα που μπορεί να έχει ο σερβοκινητήρας στις εκάστοτε τάσεις.

- **Operating voltage:** 4.8 V to 7.2 V
  Το εύρος της τάσης που μπορεί να εφαρμοστεί με ασφάλεια.

- **Running Current:** 500 mA – 900 mA (6V)
  Η ένταση του ρεύματος που διαπερνά τον σερβοκινητήρα, όταν αυτό κινείται.

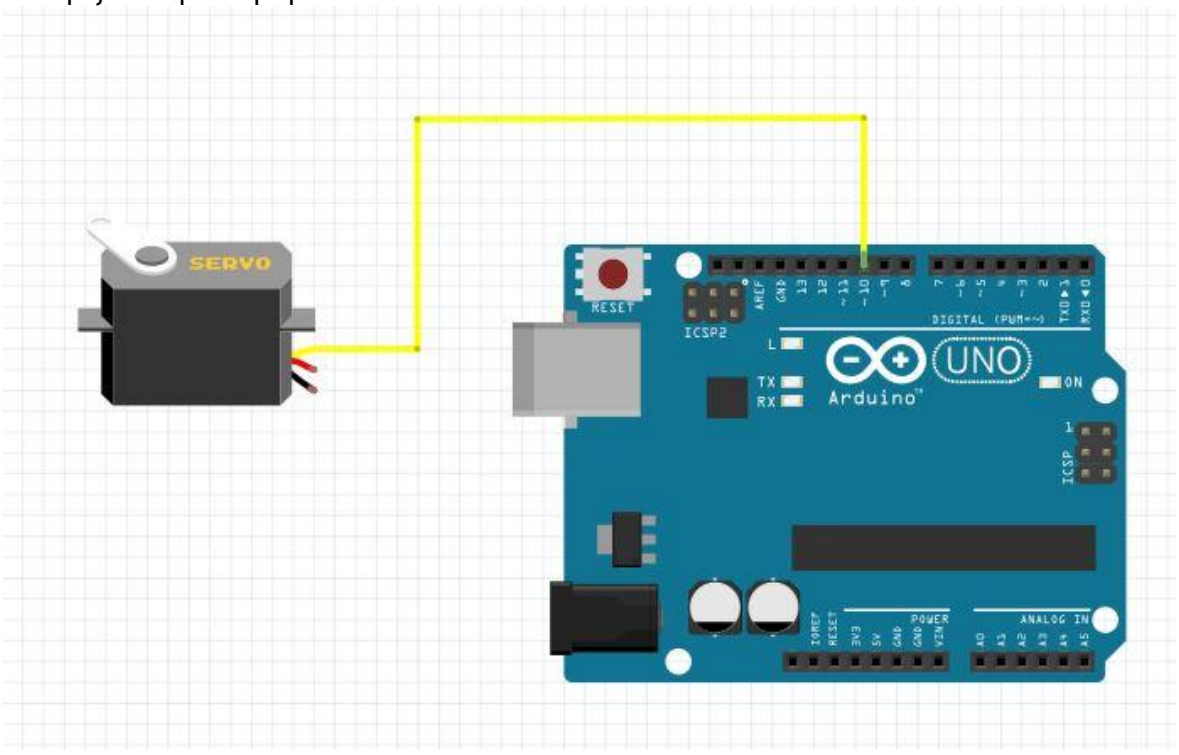
- **Stall Current:** 2.5 A (6V)
  Η ένταση του ρεύματος που διαπερνά τον σερβοκινητήρα όταν αυτό είναι φορτωμένο και έχει μηδενική ταχύτητα. Ουσιαστικά είναι η μέγιστη ένταση ρεύματος που μπορεί να δεχτεί με ασφάλεια ο σερβοκινητήρας.

- **Dead band width:** 5 μs
  Το νεκρό πλάτος παλμού σε μικροδευτερόλεπτα (μs). Είναι το πλάτος παλμού σήματος στο οποίο ο σερβοκινητήρας δεν αντιδρά. Παρακάτω αναλύεται η τεχνική ελέγχου του σερβοκινητήρα PWM, οπότε και θα γίνει κατανοητό.

- **Temperature range:** 0 ºC – 55 ºC
  Το εύρος θερμοκρασίας που μπορεί να αναπτύξει κατά τη λειτουργία του, ο σερβοκινητήρας, με ασφάλεια.

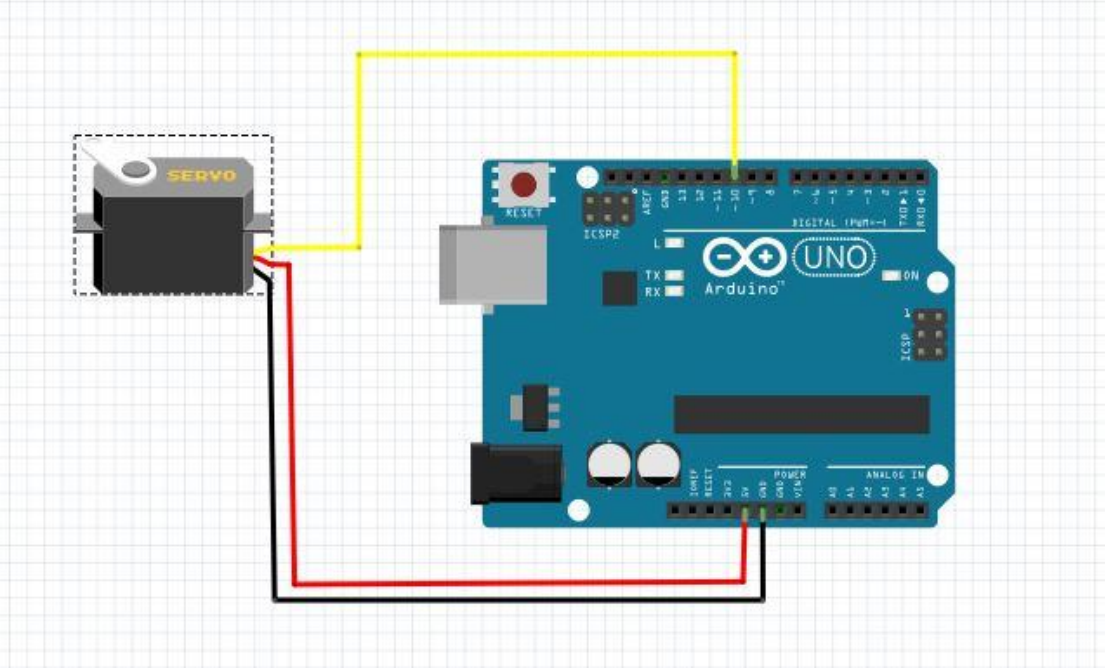
2.3.3 Σύνδεση και τροφοδοσία

Προκειμένου να μπορεί να λειτουργήσει ένας σερβοκινητήρας, θα πρέπει να συνδεθεί σε έναν υπολογιστή ή ελεγκτή. Στην προκειμένη περίπτωση ο ελεγκτής είναι το Arduino. Οπότε και αναλύεται η σύνδεση και ο έλεγχος ή η οδήγηση του σερβοκινητήρα, με το Arduino. Οι σερβοκινητήρες, όπως έχει προαναφερθεί, έχουν τρία καλώδια. Το ένα είναι για την τροφοδοσία (κόκκινο), το άλλο (μαύρο ή καφέ) για τη γείωση και το τρίτο (κίτρινο ή πορτοκαλί) για το σήμα ελέγχου. Το τελευταίο, το καλώδιο δηλαδή του σήματος ελέγχου, θα πρέπει οπωσδήποτε να συνδεθεί σε ένα PWM pin του Arduino, όπως φαίνεται στο σχήμα 2.34, ώστε να μπορεί να δέχεται τα παλμικά σήματα από την πλακέτα που καθορίζουν την κίνησή του.



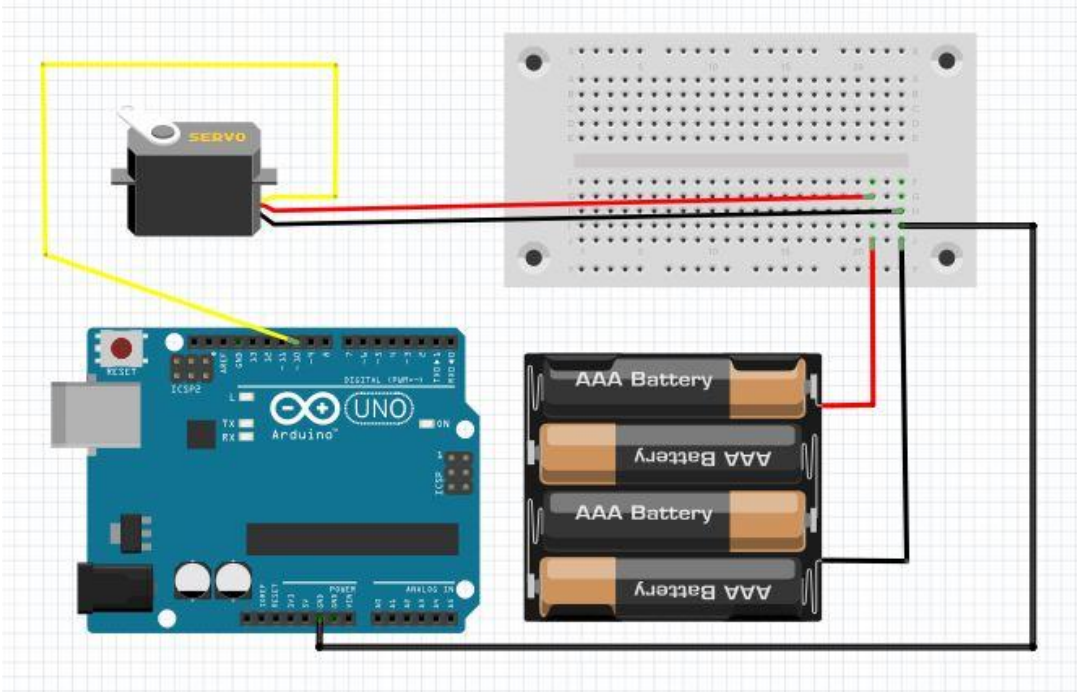
Σχήμα 2.34: Σύνδεση καλωδίου σήματος με PWM pin του Arduino

Η τροφοδοσία του σερβοκινητήρα μπορεί να γίνει με δύο τρόπους. Ο ένας τρόπος είναι με παροχή τάσης 5V, απευθείας από την πλακέτα του Arduino, όπως φαίνεται στο σχήμα 2.35.



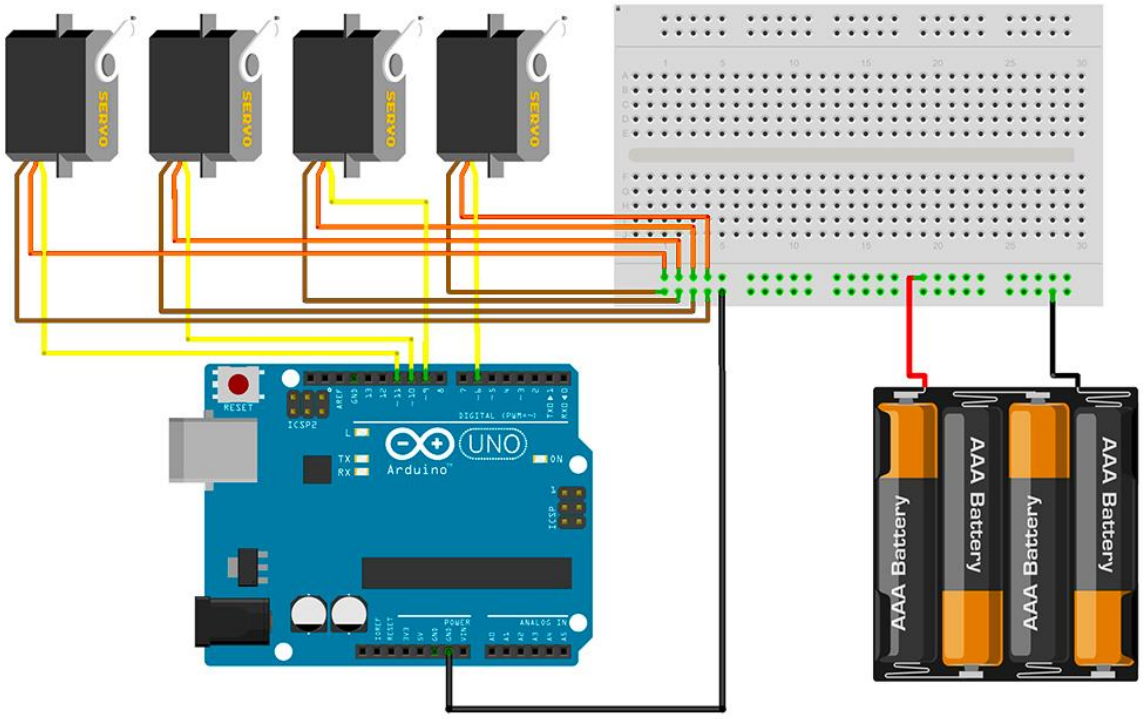
Σχήμα 2.35: Σερβοκινητήρας συνδεδεμένος απευθείας στο Arduino

Ενώ ο δεύτερος τρόπος είναι με εξωτερική τροφοδοσία από μπαταρίες ή τροφοδοτικό, μέσω ενός breadboard, όπως φαίνεται στο σχήμα 2.36.



Σχήμα 2.36: Σερβοκινητήρας με εξωτερική τροφοδοσία από μπαταρίες

Η ίδια λογική στις συνδέσεις εφαρμόζεται και για τη σύνδεση πολλαπλών σερβοκινητήρων. Η τροφοδοσία τους, όμως, θα πρέπει να είναι εξωτερική και πάντα να έχουν την ίδια γείωση με την πλακέτα, όπως δείχνει το σχήμα 2.37.

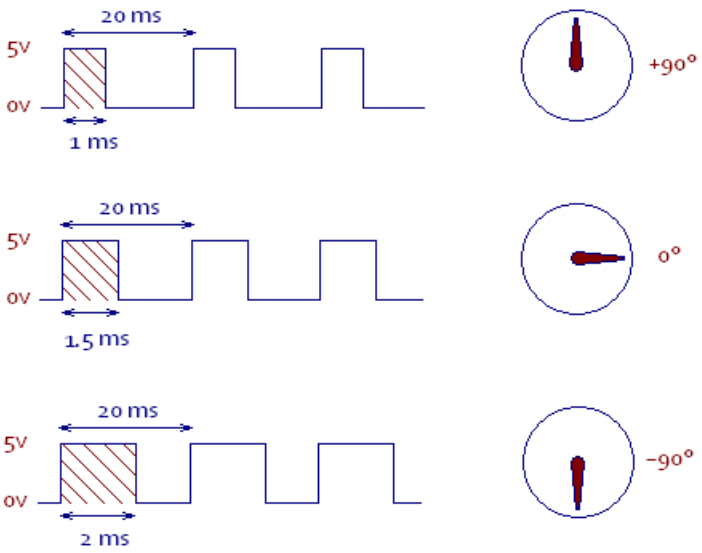


Σχήμα 2.37: Σύνδεση τεσσάρων σερβοκινητήρων σε πλακέτα Arduino Uno

2.3.4 Έλεγχος σερβοκινητήρα με PWM

Εφόσον έχει γίνει σωστά η σύνδεση του σερβοκινητήρα με την πλακέτα, ακολουθεί ο έλεγχος της λειτουργίας του. Οι σερβοκινητήρες ελέγχονται με την αποστολή ηλεκτρικών παλμών σταθερής συχνότητας (PWM – pulse width modulation) των 50Hz, τετραγωνικής μορφής και μεταβλητού κύκλου εργασίας (duty cycle), μέσω του καλωδίου σήματος από το Arduino. Υπάρχει ένας ελάχιστος, σε πλάτος, παλμός, ένας μέγιστος και μια περιοδικότητα των 20 ms. Όπως έχει προαναφερθεί, οι περισσότεροι σερβοκινητήρες που υπάρχουν στο εμπόριο περιστρέφουν τον άξονά τους έως και 90 μοίρες και προς τις δύο κατευθύνσεις (CW,CCW). Συνολικά λοιπόν ο άξονας μπορεί να περιστραφεί έως 180 μοίρες. Η ουδέτερη θέση του σερβοκινητήρα θεωρείται αυτή που είναι στη μέση του συνολικού εύρους περιστροφής. Το PWM αποστέλλεται στον κινητήρα και καθορίζει τη θέση του άξονα, έπειτα, ανάλογα με τη διάρκεια (πλάτος) του παλμού, ο άξονας θα στραφεί προς την επιθυμητή θέση. Ο σερβοκινητήρας αναμένει την εμφάνιση ενός παλμού κάθε 20 χιλιοστά του δευτερολέπτου (ms), ενώ το μήκος του παλμού θα καθορίσει το πόσο θα περιστραφεί ο άξονας. Για παράδειγμα, ένας παλμός με πλάτος 1ms θα αναγκάσει τον κινητήρα να περιστρέψει των άξονα στη θέση των 90 μοιρών. Αν ο παλμός έχει πλάτος 1.5ms, τότε ο άξονας θα περιστραφεί στις 0 μοίρες, ενώ αν το πλάτος του είναι 2ms, ο άξονας θα φτάσει στη θέση των -90 μοιρών, όπως ακριβώς δείχνει το σχήμα 2.38. Βέβαια σε κάθε σερβοκινητήρα τα όρια αλλάζουν ανάλογα την κατασκευή του. Αντίστοιχα όλες οι θέσεις που μπορεί να βρεθεί ο άξονας του σερβοκινητήρα, καθορίζονται από το πλάτος του εκάστοτε παλμού του εισερχόμενου σήματος. Αυτό πρακτικά συμβαίνει με τη βοήθεια της βιβλιοθήκης Servo, που διαθέτει το προγραμματικό περιβάλλον του Arduino, οποία μεταφράζει τις γωνίες περιστροφής που δίνονται από το

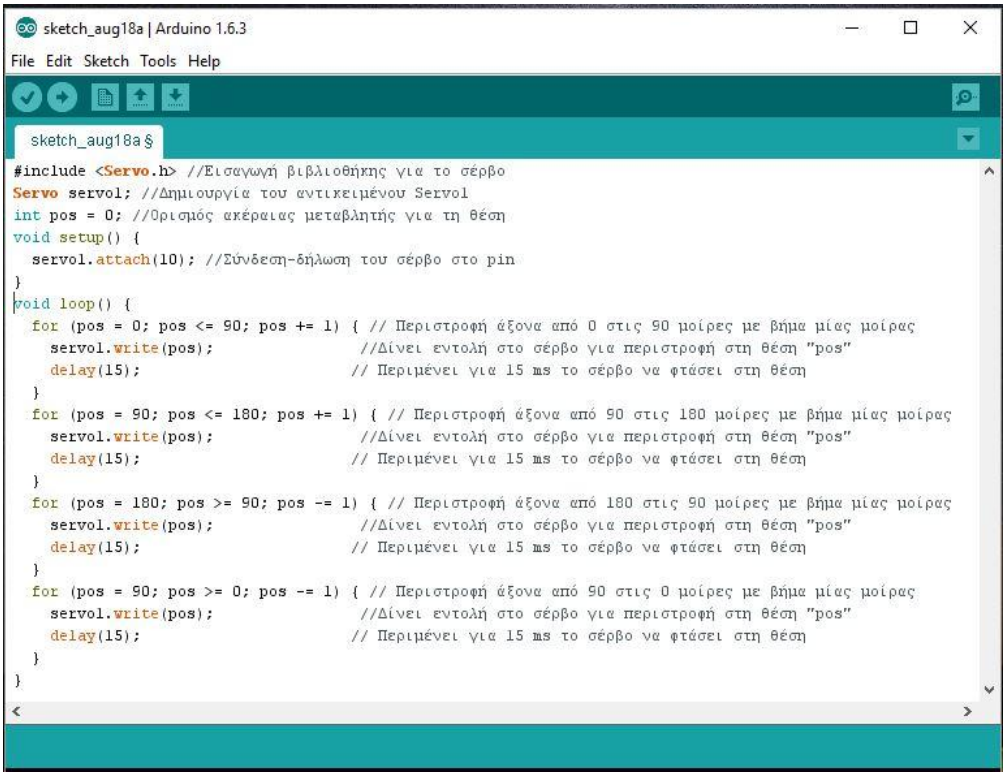
πρόγραμμα, σε παλμικά σήματα στον σερβοκινητήρα. Παρακάτω ακολουθεί ένα παράδειγμα κώδικα περιστροφής του άξονα ενός σερβοκινητήρα.



Σχήμα 2.38: PWM και οι θέσεις του άξονα του σερβοκινητήρα

2.3.5 Παράδειγμα κώδικα οδήγησης σερβοκινητήρα

Το παράδειγμα που ακολουθεί είναι μια απλή εφαρμογή των προαναφερθέντων, με στόχο την περιστροφή του άξονα ενός σερβοκινητήρα σε τρεις θέσεις και προς τις δύο κατευθύνσεις. Συνεπώς η πορεία του άξονα θα είναι από τις 0 στις 90 και στις 180 μοίρες και από τις 180, στις 90 και στις 0 μοίρες. Ο κώδικας της εφαρμογής φαίνεται στο σχήμα 2.39.




```
#include <Servo.h> //Εισαγωγή βιβλιοθήκης για το σέρβο
Servo servo1; //Δημιουργία του αντικειμένου Servo1
int pos = 0; //Ορισμός ακέραιας μεταβλητής για τη θέση
void setup() {
  servo1.attach(10); //Σύνδεση-δήλωση του σέρβο στο pin
}
void loop() {
  for (pos = 0; pos <= 90; pos += 1) { // Περιστροφή άξονα από 0 στις 90 μοίρες με βήμα μίας μοίρας
    servo1.write(pos);              //Δίνει εντολή στο σέρβο για περιστροφή στη θέση "pos"
    delay(15);                      // Περιμένει για 15 ms το σέρβο να φτάσει στη θέση
  }
  for (pos = 90; pos <= 180; pos += 1) { // Περιστροφή άξονα από 90 στις 180 μοίρες με βήμα μίας μοίρας
    servo1.write(pos);              //Δίνει εντολή στο σέρβο για περιστροφή στη θέση "pos"
    delay(15);                      // Περιμένει για 15 ms το σέρβο να φτάσει στη θέση
  }
  for (pos = 180; pos >= 90; pos -= 1) { // Περιστροφή άξονα από 180 στις 90 μοίρες με βήμα μίας μοίρας
    servo1.write(pos);              //Δίνει εντολή στο σέρβο για περιστροφή στη θέση "pos"
    delay(15);                      // Περιμένει για 15 ms το σέρβο να φτάσει στη θέση
  }
  for (pos = 90; pos >= 0; pos -= 1) { // Περιστροφή άξονα από 90 στις 0 μοίρες με βήμα μίας μοίρας
    servo1.write(pos);              //Δίνει εντολή στο σέρβο για περιστροφή στη θέση "pos"
    delay(15);                      // Περιμένει για 15 ms το σέρβο να φτάσει στη θέση
  }
}
```

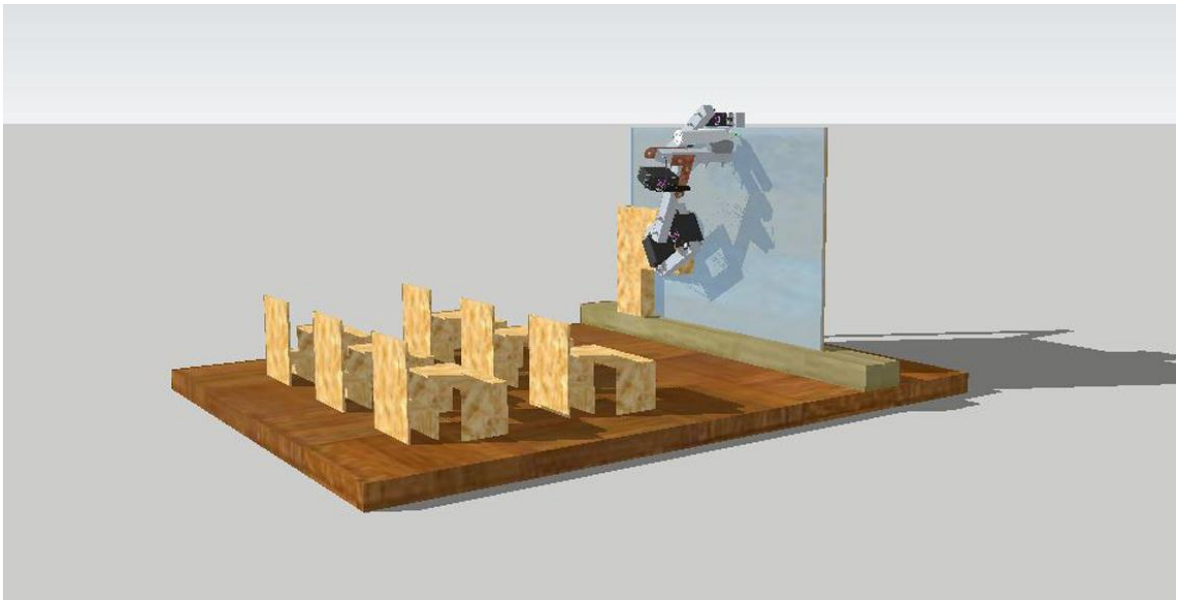
Σχήμα 2.39: Κώδικας περιστροφής σερβοκινητήρα με γωνίες 0, 90 και 180 μοίρες

Ωστόσο το Arduino είναι συνδεδεμένο στο υπολογιστή μέσω USB. Ενώ ο σερβοκινητήρας τροφοδοτείται απευθείας από την πλακέτα με τάση 5V και το καλώδιο σήματός του είναι συνδεδεμένο στο PWM pin 10 της πλακέτας.

3. ΣΧΕΔΙΑΣΜΟΣ ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

3.1 Σχεδιασμός

Όπως έχει προαναφερθεί στην εισαγωγή, στην παρούσα εργασία παρουσιάζεται ο σχεδιασμός, η κατασκευή και ο προγραμματισμός ενός ρομποτικού συστήματος, το οποίο έχει σαν κύρια εργασία τη γραφή λέξεων σε πίνακα και έπειτα τον καθαρισμό του πίνακα. Η τελική μορφή της κατασκευής, θα μπορούσε να ειπωθεί, ότι μοιάζει με μία ρομποτική αίθουσα μαθήματος. Συνεπώς η κατασκευή αποτελείται από δύο βασικά μέρη. Το ρομποτικό σύστημα και τη μακέτα της αίθουσας. Το ρομποτικό σύστημα, το οποίο είναι προσαρμοσμένο στον πίνακα, αποτελείται από δύο μέρη τα οποία επικοινωνούν μεταξύ τους, τον βραχίονα και τον καθαριστήρα. Η μακέτα της αίθουσας αποτελείται από το δάπεδο, τον πίνακα που είναι κάθετος στο δάπεδο και από έξι τραπεζοκαθίσματα. Το κύριο μέρος της κατασκευής είναι το ρομποτικό σύστημα και είναι αυτό που θα αναλυθεί εκτενώς στη συνέχεια. Ωστόσο τα κριτήρια στα οποία βασίστηκε ο σχεδιασμός του συστήματος είναι η πρωτοτυπία του σχεδίου, η απλότητα της κατασκευής και της λειτουργίας της, εφόσον πρόκειται για εκπαιδευτικό σκοπό, η χρήση μόνο εργαλείων χειρός, η επαναχρησιμοποίηση αντικειμένων και εξαρτημάτων, τα οποία βρίσκονται στα περισσότερα σπίτια, καθώς και το χαμηλό κόστος κατασκευής. Ο σχεδιασμός σε μοντέλο τριών διαστάσεων έγινε με το πρόγραμμα **SketchUp**. Το σχήμα 3.1 αποτελεί ένα στιγμιότυπο του προγράμματος σχεδιασμού και απεικονίζει την τελική μορφή της κατασκευής.



Σχήμα 3.1: Η κατασκευή σε 3D μοντέλο από το SketchUp

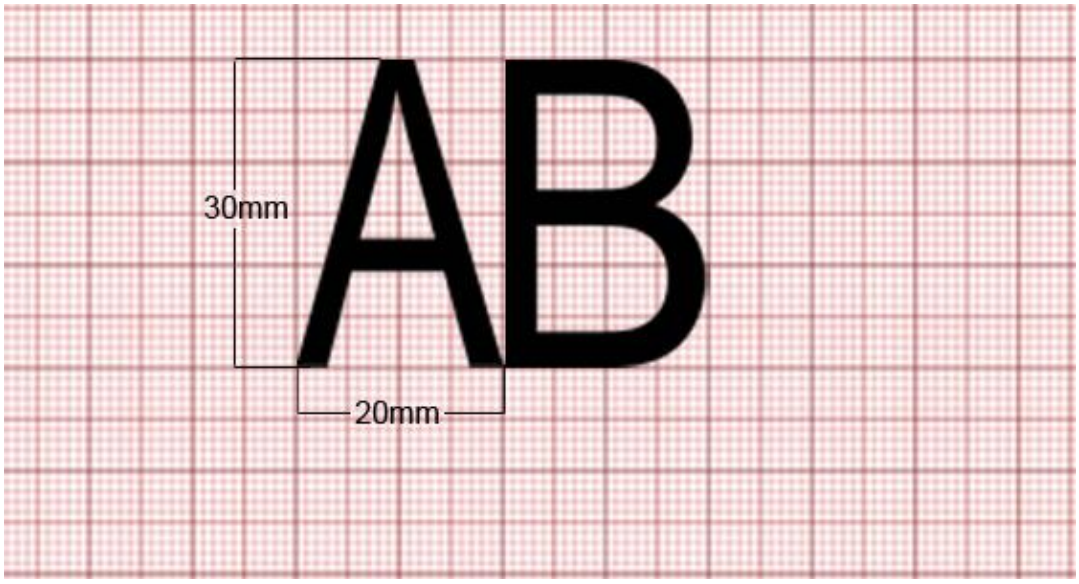
3.1.1 Σχεδιασμός ρομποτικού βραχίονα

Καταρχάς, στα κριτήρια, στα οποία βασίζεται η κατασκευή, συγκαταλέγεται η πρωτοτυπία. Συνεπώς ο βραχίονας σχεδιάστηκε χωρίς κάποιο συγκεκριμένο πρότυπο, αλλά με βάση κάποιους περιορισμούς. Οι περιορισμοί που τέθηκαν κατά τον σχεδιασμό του, είναι η ικανότητα του βραχίονα να μπορεί να γράφει λέξεις με επτά γράμματα σε μία σειρά, τα κριτήρια της απλότητας της κατασκευής και λειτουργίας του και το κόστος κατασκευής. Επομένως τα τρία ζητήματα που απορρέουν από τους παραπάνω περιορισμούς είναι: οι ελάχιστοι βαθμοί ελευθερίας, λόγω της απλότητας, που απαιτούνται για το κύριο σώμα

του χειριστή, τα μήκη των συνδέσμων προκειμένου ο χώρος εργασίας του να καλύπτει το εμβαδό των λέξεων και οι κατάλληλοι σερβοκινητήρες.

- **Βαθμοί ελευθερίας:** Η βασική εργασία του βραχίονα, όπως έχει αναφερθεί, είναι η γραφή λέξεων σε έναν πίνακα. Αυτή η εργασία προϋποθέτει την ικανότητα ενός βραχίονα να προσεγγίζει με το τελικό σημείο δράσης όλα τα σημεία ενός προκαθορισμένου χώρου εργασίας. Ο βραχίονας ο οποίος έχει την ικανότητα να προσεγγίζει όλα τα σημεία σε ένα προκαθορισμένο χώρο ενός επιπέδου X-Y, είναι ένας επίπεδος βραχίονας με δύο περιστροφικές αρθρώσεις κατά το ελάχιστο δυνατό. Συνεπώς η εργασία αυτή απαιτεί δύο βαθμούς ελευθερίας τουλάχιστον..

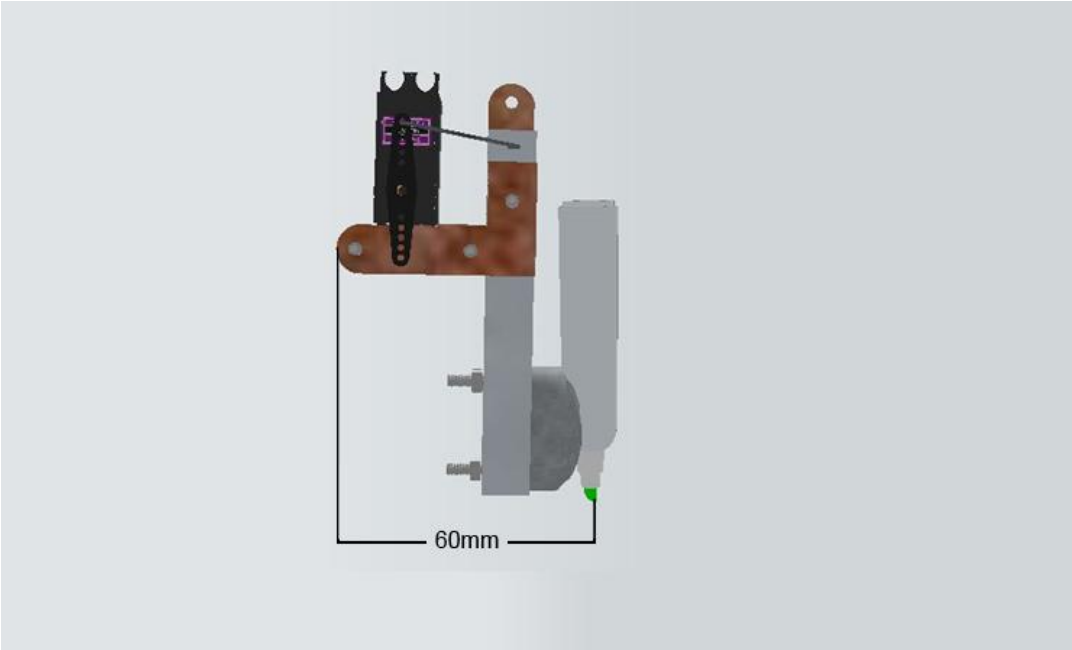
- **Σύνδεσμοι:** Ο στόχος του βραχίονα είναι να έχει τη δυνατότητα να γράφει λέξεις των επτά γραμμάτων σε μία σειρά. Για να γίνει αυτό προγραμματιστικά, θα πρέπει ο βραχίονας να γνωρίζει τις θέσεις αρχής και τέλους του πρώτου, δεύτερου, τρίτου κ.ο.κ γράμματος στο χώρο εργασίας. Συνεπώς θα πρέπει να οριστεί ένα μέγιστο πλάτος γράμματος για ένα σταθερό ύψος. Το σταθερό ύψος των γραμμάτων ορίσθηκε στα 30mm, ενώ το μέγιστο πλάτος στα 20mm, όπως φαίνεται στο σχήμα 3.2.



Σχήμα 3.2: Διαστάσεις γραμμάτων

Επομένως θα πρέπει να προσδιοριστούν τα μήκη των συνδέσμων του βραχίονα L_1 και L_2. Το L_2 συμπεριλαμβάνει και τον καρπό. Συνεπώς, θεωρήθηκε σωστό να σχεδιαστεί πρώτα ο **καρπός**, προκειμένου να γίνουν γνωστά τα γεωμετρικά του χαρακτηριστικά. Σημειώνεται εδώ, ότι τα υλικά που χρησιμοποιήθηκαν, όπως π.χ. η θήκη του μαρκαδόρου, οι γωνίες και οι άξονες, ο σερβοκινητήρας, προϋπήρχαν του σχεδιασμού, οπότε ο σχεδιασμός βασίστηκε στη γεωμετρία αυτών. Στη συνέχεια παρουσιάζονται εκτενώς τα υλικά, στο υποκεφάλαιο που αναφέρεται στην κατασκευή. Το σύστημα του καρπού περιλαμβάνει μία θέση συγκράτησης του μαρκαδόρου, η οποία είναι προσαρμοσμένη σε έναν κάθετο άξονα, που με τη βοήθεια ενός σερβοκινητήρα σηκώνει και κατεβάζει το μαρκαδόρο στις αντίστοιχες θέσης, γραφής και μη γραφής. Το τελικό μήκος του καρπού ανήλθε στα 60mm, σύμφωνα με το σχεδιασμό του. Ο καρπός, όπως σχεδιάστηκε, φαίνεται στο σχήμα 3.3. Επομένως εφόσον ο καρπός είναι 60mm, απομένει ο προσδιορισμός του

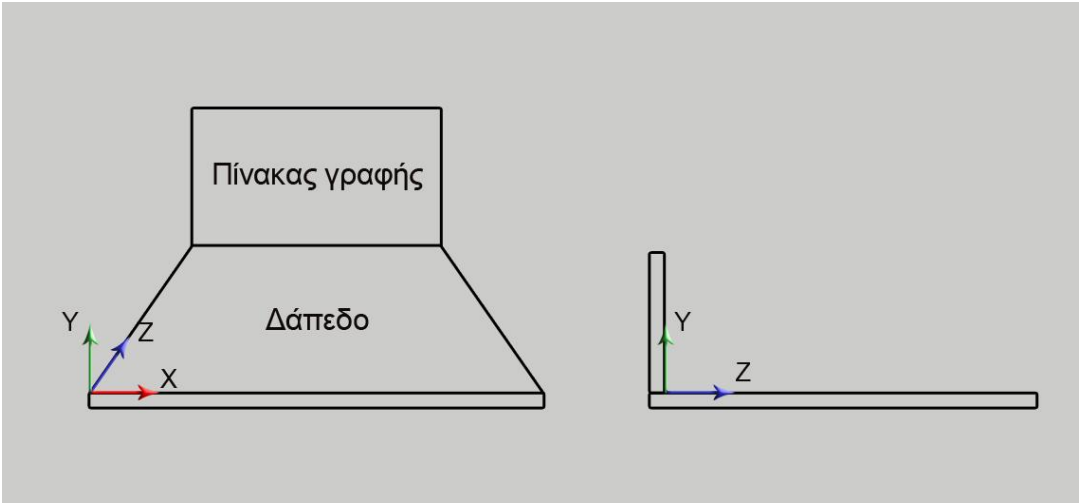
μήκους των υπόλοιπων αξόνων του βραχίονα. Μια αυθαίρετη, αλλά και αποτελεσματική προσέγγιση, όπως αποδείχτηκε στη λειτουργία, είναι η αναλογία $L_1/L_2=1/2$, δηλαδή $L_2=2L_1$. Συνεπώς ο **L_1=60mm** και ο **L_2=120mm**, μαζί με τον καρπό. Με αυτά τα δεδομένα συνεχίστηκε ο καθορισμός του χώρου στον οποίο θα πρέπει να γράφει ο βραχίονας. Σύμφωνα με την αρχική ιδέα, ο βραχίονας θα πρέπει να γράφει με τον αγκώνα κάτω, στο πρώτο τεταρτημόριο 0-90 μοίρες, επτά γράμματα στη σειρά με τα παραπάνω χαρακτηριστικά. Επομένως θα χρειαστεί να βρεθεί ένας ορθογώνιος χώρος με ύψος 30mm και μήκος 140mm, μέσα στο χώρο εργασίας του βραχίονα. Αυτός ο χώρος είναι το ορθογώνιο που ορίζεται από τα σημεία Α(0,10), Β(14,10), Γ(14,7), Δ(0,7). Πράγματι, με μια απλή προσομοίωση του βραχίονα με αυτές τις διαστάσεις, η οποία έγινε με τη βοήθεια του **Microsoft Excel** και εφαρμόστηκε η αντίστροφη κινηματική ανάλυση, παρατηρήθηκε ότι ο βραχίονας είχε την ικανότητα να προσεγγίζει όλα τα σημεία του χώρου που καθορίστηκε για τη γραφή των λέξεων. Ένα στιγμιότυπο από αυτήν την προσομοίωση φαίνεται στο σχήμα 3.4.



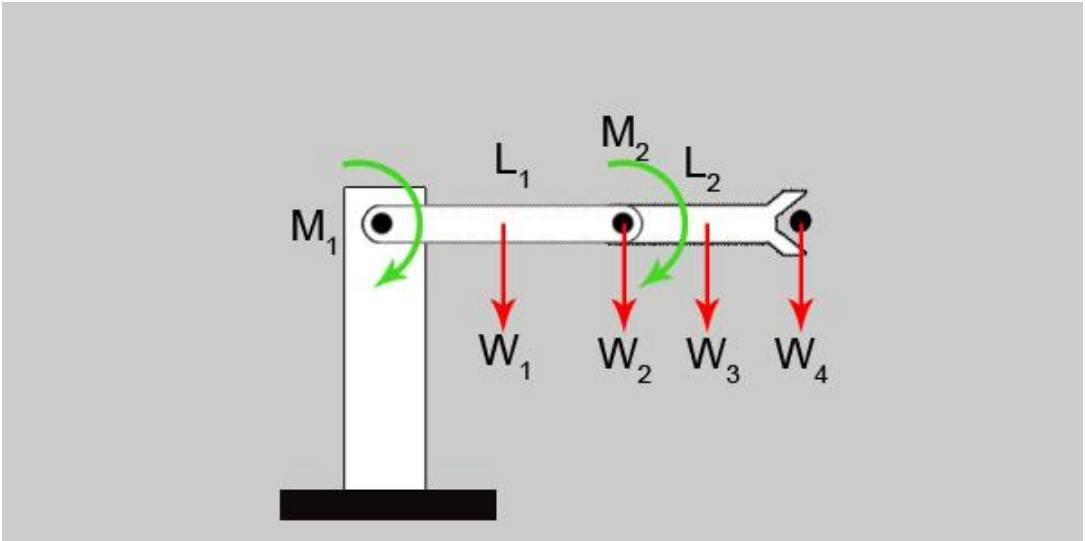
Σχήμα 3.3 Ο καρπός με το τελικό σημείο δράσης

Σχήμα 3.4: Προσομοίωση βραχίονα με το Microsoft Excel

Σερβοκινητήρες: Όπως παρατηρείται στο προηγούμενο σχήμα 3.1, αλλά και στο σχήμα 3.5 που ακολουθεί, το επίπεδο εργασίας X-Y του βραχίονα, είναι κάθετο στο επίπεδο X-Z του δαπέδου, εφόσον ο πίνακας γραφής είναι κάθετος στο δάπεδο. Το ζήτημα το οποίο προκύπτει από αυτήν τη διάταξη, είναι αυτό των βαρυτικών δυνάμεων των εξαρτημάτων, οι οποίες δημιουργούν ροπές κατά τη διεύθυνση περιστροφής των αρθρώσεων, όπως φαίνονται στο σχήμα 3.6. Οι ροπές αυτές, που αυξάνουν με το μέγεθος των συνδέσμων και το βάρος των εξαρτημάτων, επιβαρύνουν την κίνηση στις αρθρώσεις, μειώνουν την ακρίβεια του βραχίονα και καταπονούν τους σερβοκινητήρες. Η επίλυση του προβλήματος αυτού, βασίζεται στη σωστή επιλογή των κατάλληλων σερβοκινητήρων με το ελάχιστο κόστος και με κύριο χαρακτηριστικό τη ροπή τους. Σημειώνεται ότι καθώς αυξάνει η ροπή στα χαρακτηριστικά των σερβοκινητήρων, αυξάνει και το κόστος τους. Συνεπώς το επόμενο βήμα είναι ο υπολογισμός των ροπών στις αρθρώσεις με δεδομένα τα μήκη των συνδέσμων και τα βάρη. Μερικά από τα βάρη των εξαρτημάτων είναι εμπειρικώς υπολογισμένα.



Σχήμα 3.5: Ο πίνακας γραφής είναι κάθετος στο δάπεδο



Σχήμα 3.6: Βάρη και ροπές από τα στοιχεία του βραχίονα

Η εξίσωση της ροπής για την πρώτη άρθρωση είναι:

$$M_1=\left(\frac{L_1}{2}\times W_1\right)+(L_1\times W_2)+\left(\left(L_1+\frac{L_2}{2}\right)\times W_3\right)+((L_1+L_2)\times W_4)$$

Ενώ για τη δεύτερη είναι:

$$M_2=\left(\frac{L_2}{2}\times W_3\right)+(L_2\times W_4)$$

Όπου τα M_1 και M_2 είναι οι ροπές στις αρθρώσεις 1 και 2 αντίστοιχα. Τα L_1 και L_2 είναι τα μήκη των συνδέσμων, με το L_2 να συμπεριλαμβάνει και τον καρπό με το τελικό σημείο δράσης. Τα W_1 και W_3 δυνάμεις βάρους των συνδέσμων L_1 και L_2 αντίστοιχα, με σημείο εφαρμογής το μέσον των αξόνων και τέλος τα W_2 και W_4 είναι δυνάμεις βάρους της δεύτερης άρθρωσης και του καρπού.

Υπολογίζοντας τις ροπές, οι οποίες είναι περίπου 17kg.cm για τον σερβοκινητήρα της βάσης και 7kg.cm για τον σερβοκινητήρα του αγκώνα και με σημαντικά κριτήρια το βάρος, το οποίο δεν πρέπει να ξεπερνάει τα 70g, καθώς και το κόστος των σερβοκινητήρων, επιλέχθηκαν τελικά οι σερβοκινητήρες CYS S8201, όπως αυτόν που φαίνεται στο σχήμα 3.7.

Σχήμα 3.7: Ο σερβοκινητήρας CYS S8201 και το servo horn του.

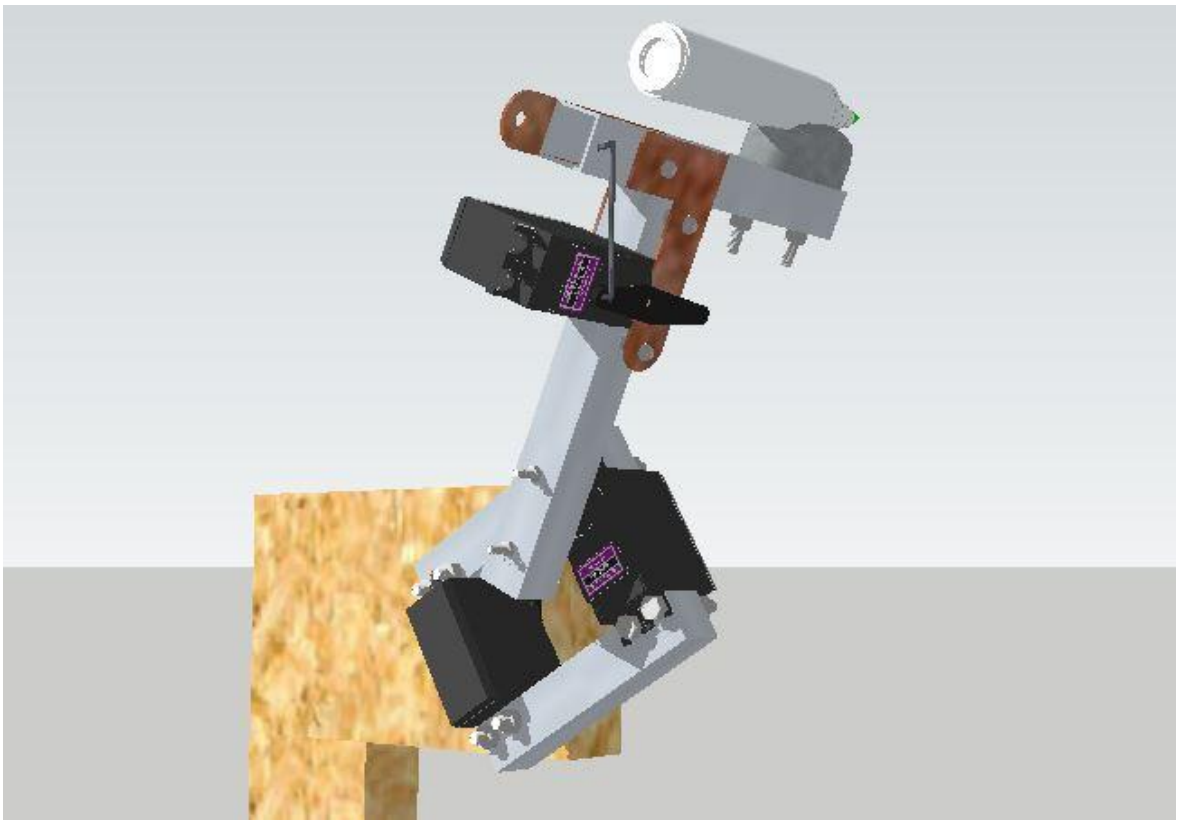
Τα χαρακτηριστικά του, οι διαστάσεις του και η τιμή του, είναι:

Type: Digital
Weight: 64g
Size: 40.1 x 20.1 x 38.8mm
Operating Voltage: 6.0~7.4Volts
Operating Temperature Range: (-)10 to +50 degree C
Operating Speed (6.0V): 0.16sec/60° at no load
Operating Speed (7.4V): 0.13sec/60° at no load
Stall Torque (6.0V): 18kg.cm

Stall Torque (7.4V): 20kg.cm
Stall Current: 2.3A/2.5A
Dead Band Width: 4usec
Bearing Type: Dual Ball Bearing
Gear Type: Metal
Price: 35€

Σημειώνεται ότι ο σερβοκινητήρας του καρπού προϋπήρχε του σχεδιασμού, πληροί της προϋποθέσεις για την κατασκευή και είναι ο MG996R, του οποίου τα χαρακτηριστικά περιγράφονται στην παράγραφο 2.3.2 του προηγούμενου κεφαλαίου.

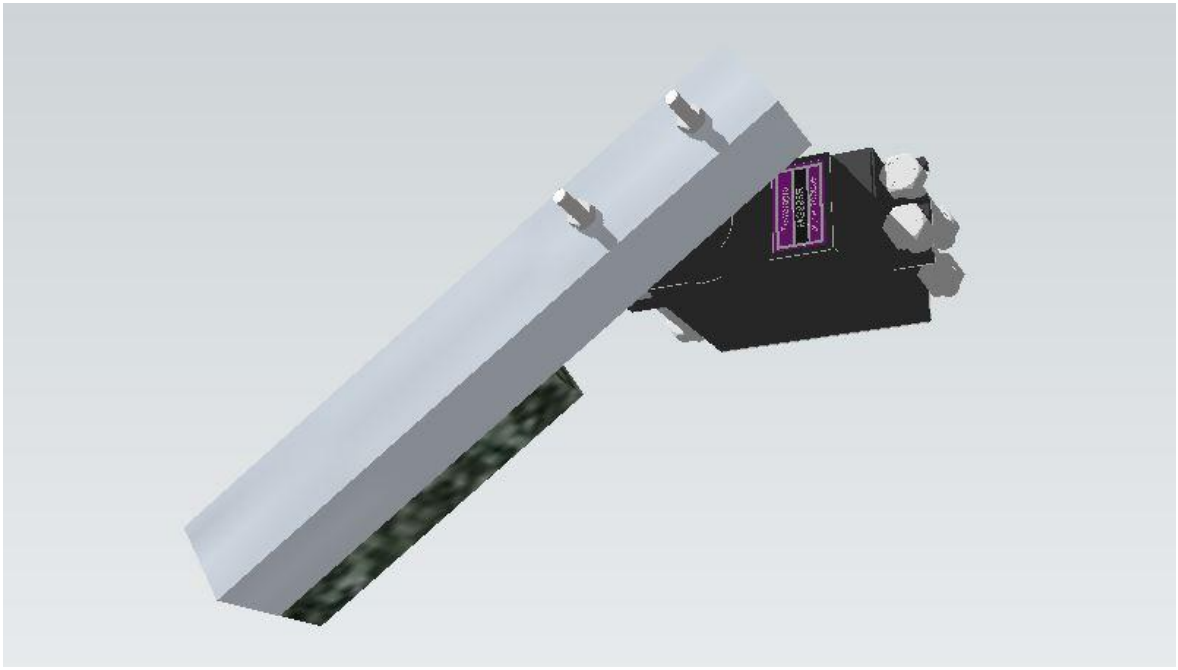
Εφόσον επιλέχθηκαν οι βαθμοί ελευθερίας, δηλαδή οι αρθρώσεις, τα μήκη των συνδέσμων και οι σερβοκινητήρες με παραπάνω ροπή από την απαιτούμενη, για την αποφυγή καταπόνησής τους, ο τελικός σχεδιασμός του βραχίονα, της εργασίας, είναι όπως απεικονίζεται σε τρισδιάστατο μοντέλο στο σχήμα 3.8.



Σχήμα 3.8: Ο τελικός βραχίονας σε τρισδιάστατο μοντέλο

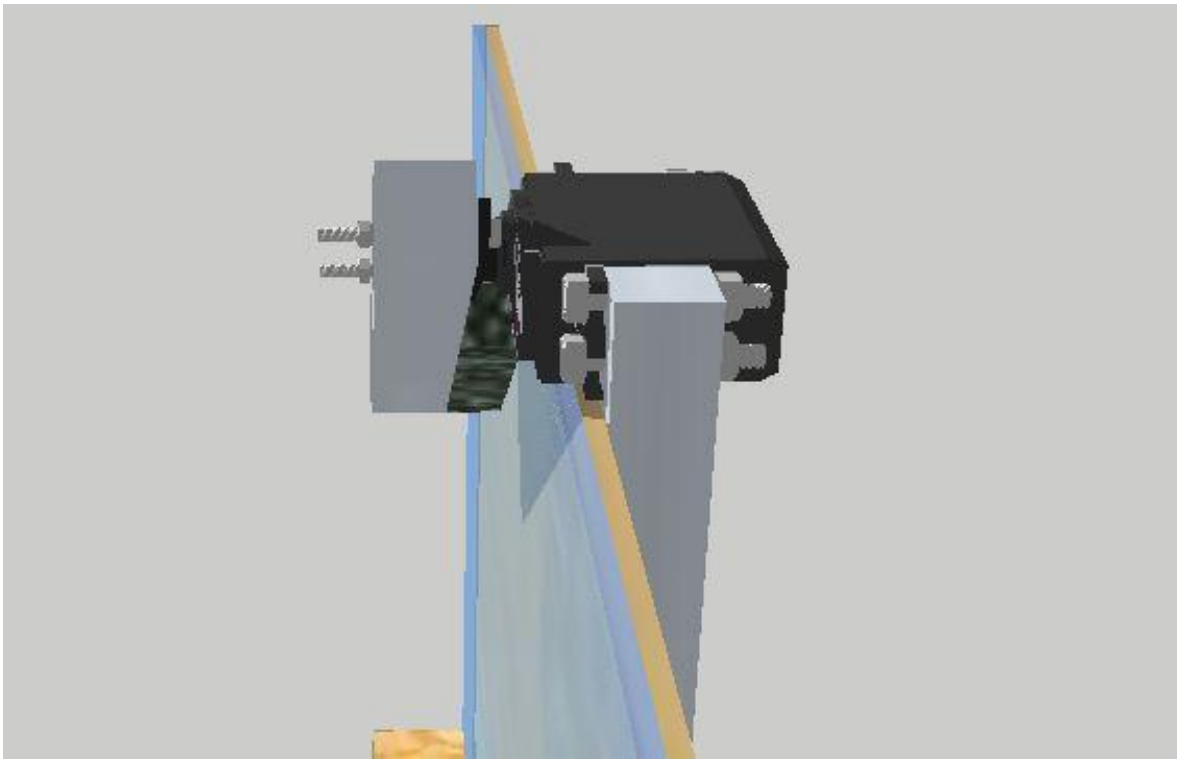
3.1.2 Σχεδιασμός καθαριστήρα

Ο καθαριστήρας είναι ένα απλό σύστημα βασισμένο στη λογική και τη λειτουργία ενός υαλοκαθαριστήρα αυτοκινήτου. Τα δύο βασικά μέρη από τα οποία αποτελείται, είναι ένας σερβοκινητήρας και ένας άξονας με ενσωματωμένο σφουγγάρι. Θα μπορούσε να ειπωθεί ότι είναι και αυτός ένας βραχίονας ενός βαθμού ελευθερίας, με μία περιστροφική άρθρωση. Η τελική συναρμολόγησή του απεικονίζεται στο σχήμα 3.9.



Σχήμα 3.9: Η συναρμολόγηση του καθαριστήρα

Στη συνέχεια ο σερβοκινητήρας κοχλιώνεται σε δύο άξονες με μήκος 254mm, οι οποίοι είναι παράλληλοι και κολλημένοι στην πίσω πλευρά του πίνακα, έτσι ώστε ο καθαριστήρας να βρίσκεται στο πάνω μέρος του πίνακα και να σαρώνει με το σφουγγάρι, την απαιτούμενη κυκλική επιφάνεια στο τζάμι του πίνακα, όπως φαίνεται στο σχήμα 3.10. Ο σερβοκινητήρας που χρησιμοποιείται είναι ο MG996R και ο άξονας έχει μήκος 120mm, έτσι ώστε να καλύπτει κατά πολύ το χώρο στον οποίο γράφει ό βραχίονας.

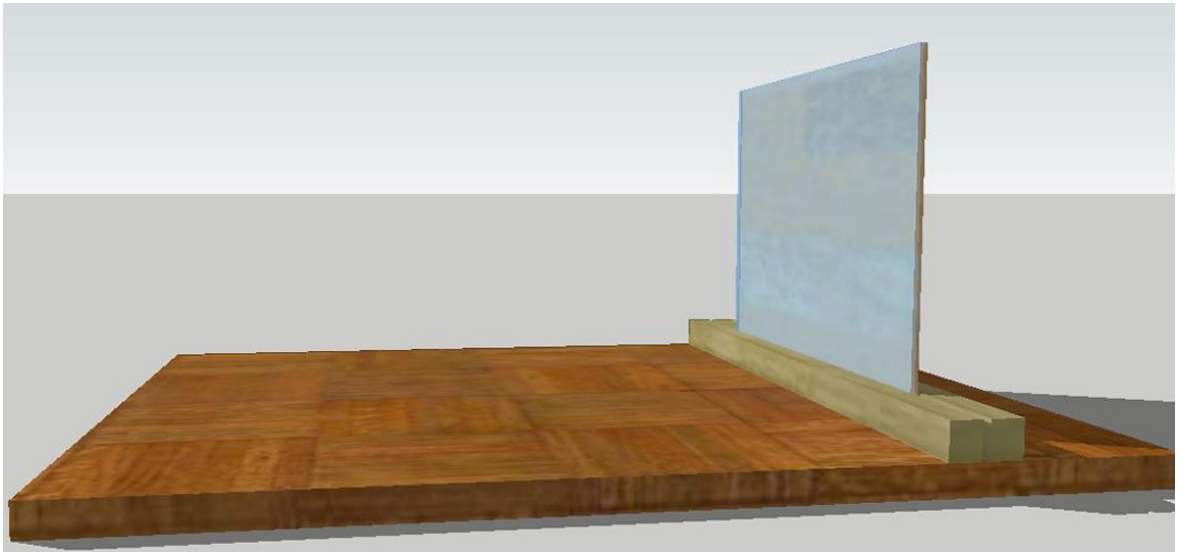


Σχήμα 3.10: Προσαρμογή καθαριστήρα στον πίνακα

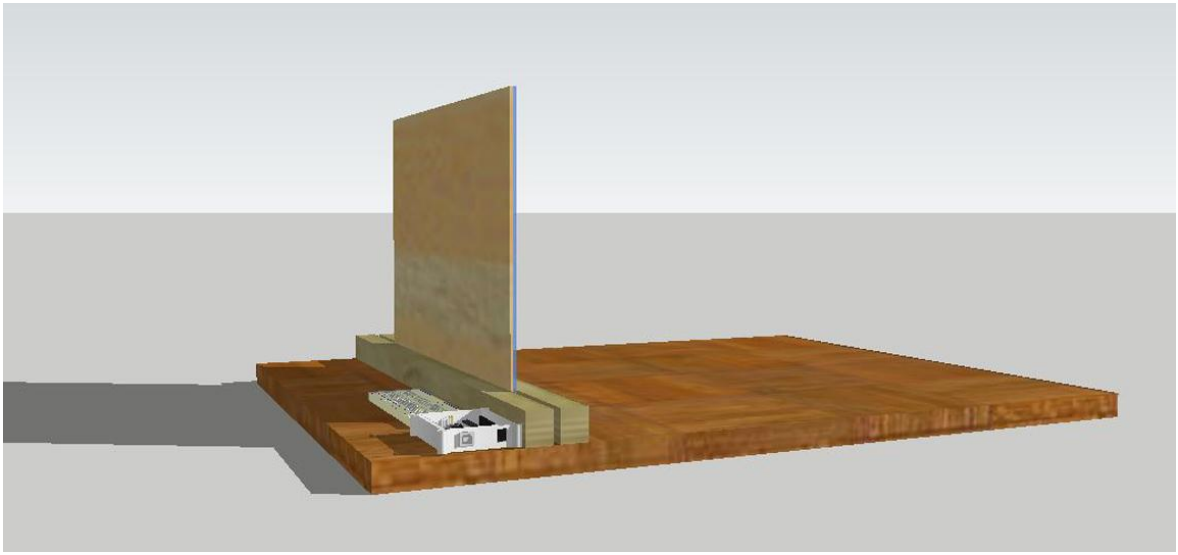
3.1.3 Σχεδιασμός αίθουσας

Η αίθουσα αποτελείται από τρία βασικά μέρη, το δάπεδο, τον πίνακα και τα έξι τραπεζοκαθίσματα. Ο πίνακας ανήκει στα υλικά και εξαρτήματα τα οποία προϋπήρχαν του σχεδιασμού, με συνέπεια τα γεωμετρικά του χαρακτηριστικά να είναι δεδομένα. Σε αυτά τα χαρακτηριστικά βασίστηκε και ο υπόλοιπος σχεδιασμός της αίθουσας. Ο πίνακας έχει διαστάσεις 400mm στο μήκος, 300mm στο ύψος (εφόσον είναι κάθετος στο δάπεδο) και 5mm πάχος. Επομένως το δάπεδο αποφασίστηκε να έχει διαστάσεις 600mm στο μήκος (σε άξονα παράλληλο του μήκους του πίνακα) και 800mm στο πλάτος, προκειμένου να υπάρχει η σχετική άνεση χώρου. Ο πίνακας συγκρατείται κάθετα στο δάπεδο, σφηνωμένος σε μία βάση, η οποία αποτελείται από δύο ράβδους τετραγωνικής διατομής 25mm, όπως φαίνεται στο σχήμα 3.11. Ο πίνακας είναι τοποθετημένος στη μέση της βάσης και συνεπώς στη μέση του δαπέδου κατά μήκος. Ενώ κατά το πλάτος του δαπέδου, ο πίνακας είναι τοποθετημένος στα 600mm, έτσι ώστε να δημιουργείται χώρος διαστάσεων 600mm x 200mm, προκειμένου να μπορούν να τοποθετηθούν τα απαραίτητα ηλεκτρονικά μέρη, όπως φαίνεται στο σχήμα 3.12.

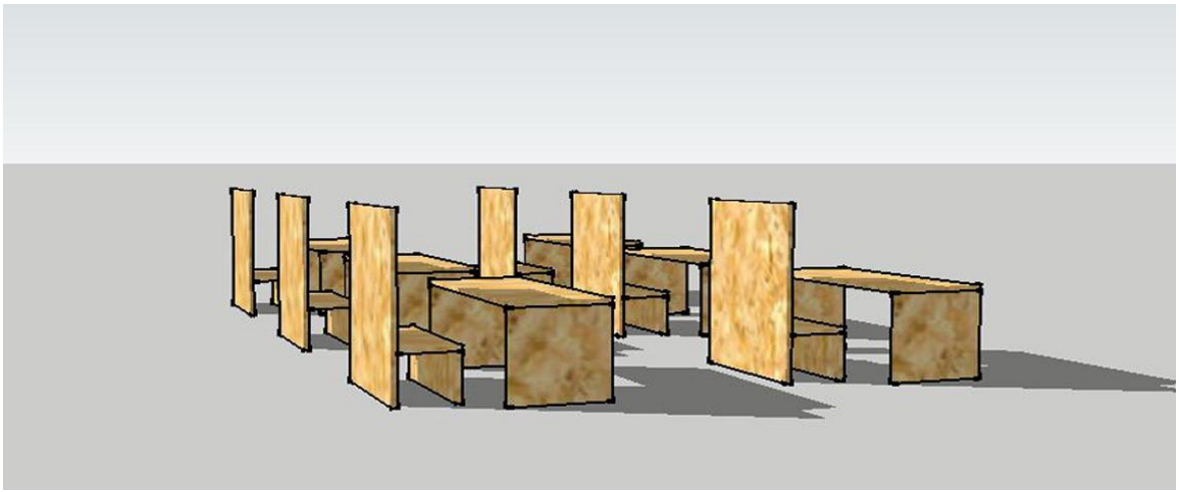
Τέλος ο σχεδιασμός περιλαμβάνει και έξι τραπέζια και έξι ενιαία καθίσματα των δύο ατόμων, τοποθετημένες προς την κατεύθυνση του πίνακα, για μια πιο ρεαλιστική απεικόνιση της αίθουσας. Τα τραπέζια έχουν διαστάσεις (μ x π x υ) 120mm x 50mm x 53mm, ενώ το κάθε κάθισμα έχει διαστάσεις (μ x π x υ) 80mm x 33mm x 100mm, στο ύψος της συμπεριλαμβάνεται και το ύψος της πλάτης. Τα καθίσματα και τα τραπέζια είναι τοποθετημένα σε δύο συστοιχίες των τριών τραπεζοκαθισμάτων, όπως φαίνεται στο σχήμα 3.13. Ο τελικός σχεδιασμός της αίθουσας απεικονίζεται στο σχήμα 3.14.



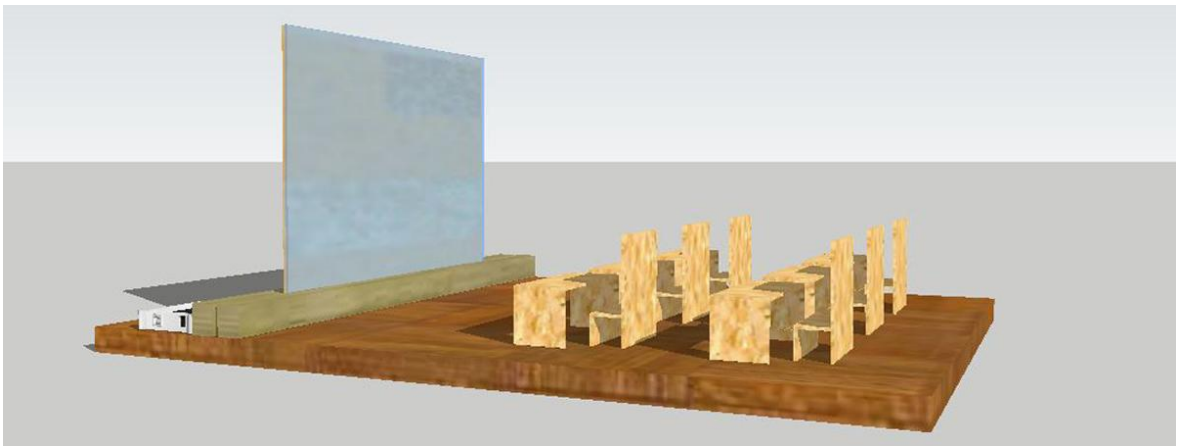
Σχήμα 3.11: Τοποθέτηση του πίνακα στη βάση



Σχήμα 3.12: Ο χώρος πίσω από τον πίνακα για τα ηλεκτρονικά εξαρτήματα



Σχήμα 3.13: Οι συστοιχίες των τραπεζοκαθισμάτων



Σχήμα 3.14: Ο τελικός σχεδιασμός της αίθουσας

3.2 Κατασκευή

Μετά το πέρας του σχεδιασμού, ακολουθεί η κατασκευή. Η ίδια σειρά, με την οποία σχεδιάστηκαν τα μέρη της διάταξης, ακολουθείται στην κατασκευή της. Συνεπώς το πρώτο μέρος της κατασκευής είναι ο βραχίονας, το δεύτερο μέρος είναι ο καθαριστήρας και το τρίτο μέρος η αίθουσα. Τα κριτήρια τα οποία αφορούν στην κατασκευή, όπως έχουν προαναφερθεί, είναι:

- Η κατασκευή να είναι, κατά ένα μεγάλο ποσοστό, χειροποίητη, δηλαδή να χρησιμοποιηθούν εργαλεία χειρός, είτε αυτά είναι ηλεκτρικά, είτε μηχανικά.
- Να γίνει επαναχρησιμοποίηση διαφόρων υλικών του σπιτιού.
- Το κόστος κατασκευής να είναι χαμηλό.

Οι κατεργασίες οι οποίες πραγματοποιήθηκαν σε όλη τη διαδικασία κατασκευής ήταν, κυρίως, κοπές, λειάνσεις και διατρήσεις. Στον πίνακα 3.1 παρουσιάζονται αναλυτικά τα εργαλεία και στον πίνακα 3.2, τα υλικά που χρησιμοποιήθηκαν, σε όλη την κατασκευή. Ωστόσο υπάρχουν κάποια υλικά τα οποία χρησιμοποιήθηκαν για την κατασκευή μόνο συγκεκριμένων τμημάτων, οπότε αυτά θα παρουσιαστούν στις αντίστοιχες παραγράφους.

Εργαλείο	Κατεργασίες	Εικόνα
Ηλεκτρικό δράπανο χειρός 500W με τρυπάνια για μέταλλα και ξύλα διαμέτρου 3mm, 4mm και 8mm.	Διάτρηση	
Ηλεκτρικό πολυεργαλείο μοντελισμού	Διάτρηση Λείανση	
Ηλεκτρική σέγα χειρός 400W	Κοπή	
Πριόνι χειρός	Κοπή	
Κοπίδι μοντελισμού	Κοπή	

Κόφτης χειρός για κοχλίες	Κοπή	
Κοχλιοστρόφια σταυρωτά και ευθεία	Σύσφιξη κοχλιών	
Κλειδί περικοχλίων	Σύσφιξη περικοχλίων	

Πίνακας 3.1: Τα εργαλεία που χρησιμοποιήθηκαν στην κατασκευή

Υλικό	Περιγραφή	Εικόνα
Σωλήνες αλουμινίου τετραγωνικής διατομής.	Η εξωτερική διατομή τους είναι 16mm x16mm, ενώ η εσωτερική τους 14mm x 14mm. Είναι υλικό επαναχρησιμοποίησης, από παλιά κεραία τηλεόρασης	
Φύλλα από ξύλο μπάλσα	Ένα φύλλο πάχους 10mm και ένα πάχους 3mm	
Ράβδοι από ξύλο μπάλσα	Μία έτοιμη κομμένη ράβδος τετραγωνικής διατομής 10mm x 10mm, από την αγορά	
Σερβοκινητήρες και servo horns	(2x) CYS 8201 (2x) MG996R με τα χαρακτηριστικά, όπως παρουσιάστηκαν παραπάνω	

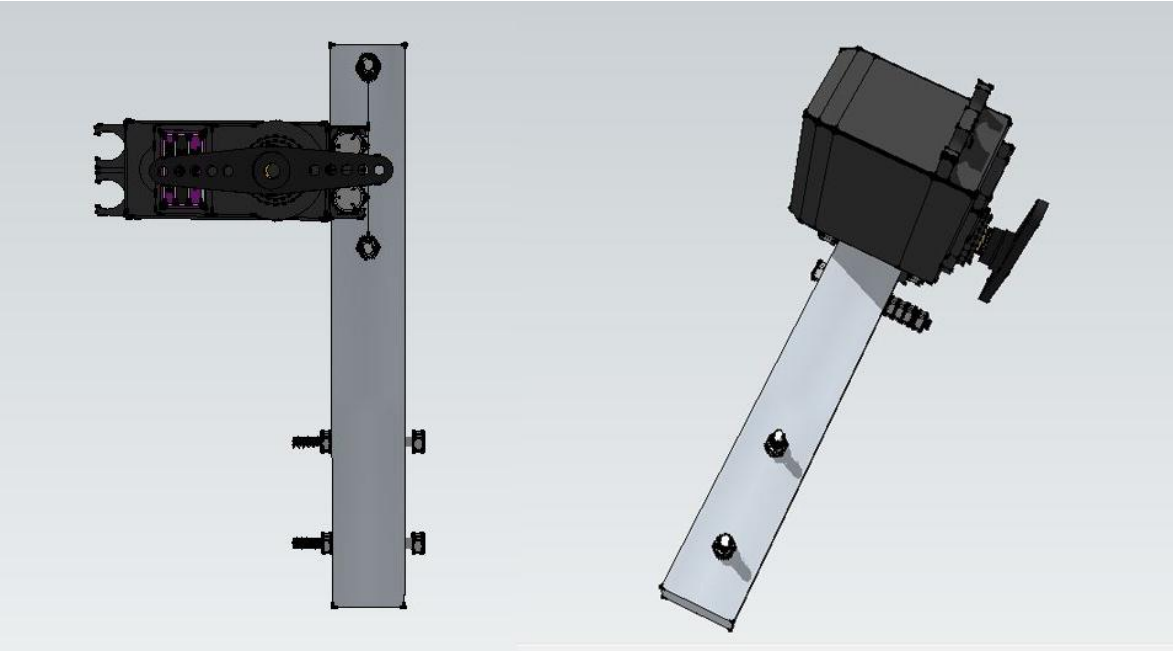
Κοχλίες για ξύλο, κοχλίες με περικόχλιο και ροδέλες.	(12x) Κοχλίες για ξύλο (6x) Κοχλίες με περικόχλιο 3mm (7x) Κοχλίες με (10x) περικόχλια 4mm (2x) απλές ροδέλες 4mm και (2x) ροδέλες με λάστιχο 4mm	
Γωνίες στήριξης	(2x) Μεταλλικές γωνίες στήριξης 90^{o}, διαστάσεων 60mm x 60mm x 15mm	
Κομμάτια ξύλου	Περίπου 1,5m^{2} ενιαία μοριοσανίδα επενδεδυμένη με μελαμίνη συνολικού πάχους 2,5mm	
Καρφιά	(12x) Μεταλλικά καρφιά ξύλου 40mm	
Κόλλα	1 σωληνάριο εποξική κόλλα	
Σύρμα	Ένα κομμάτι μήκους 50mm δύσκαμπτο σύρμα διαμέτρου 1mm	
Σφουγγάρι	Σφουγγάρι μπάνιου, είναι υλικό επαναχρησιμοποίησης	

Πίνακας 3.2: Τα υλικά που χρησιμοποιήθηκαν στην κατασκευή

3.2.1 Κατασκευή βραχίονα

Μετά τον αναλυτικό σχεδιασμό του βραχίονα, ακολουθεί η κατασκευή του, η οποία είναι βασισμένη στα εξαγόμενα δεδομένα, σε ότι αφορά στη γεωμετρία του και στη μορφή του.

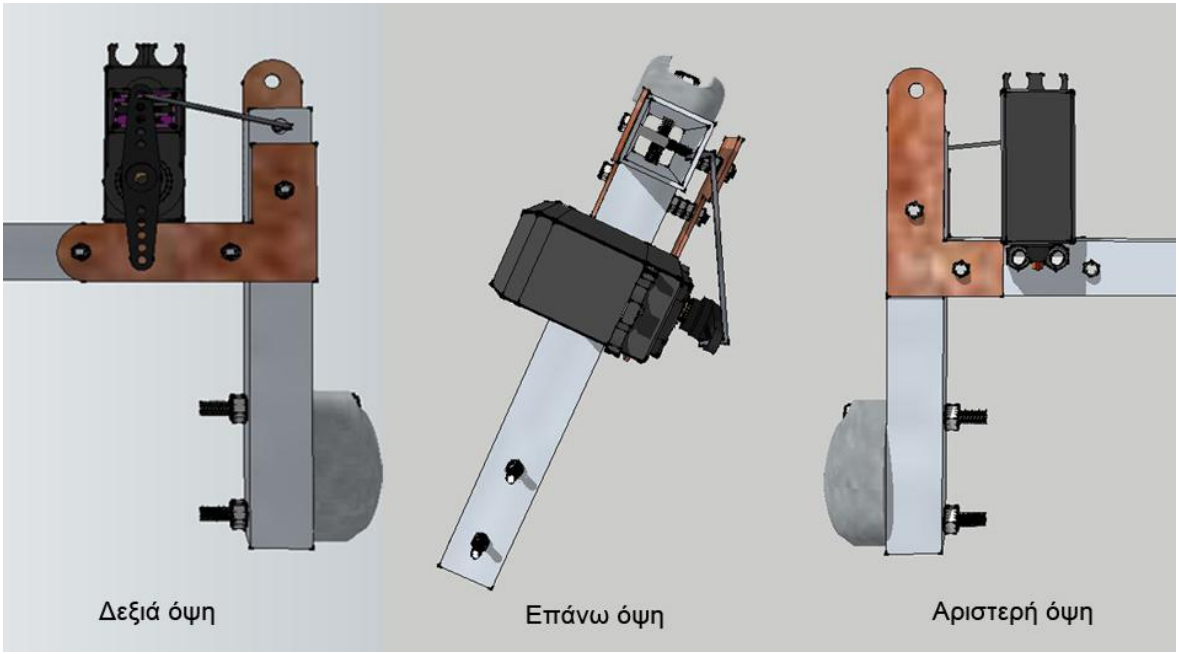
Η κατασκευή του βραχίονα ξεκίνησε από τον καρπό και τον δεύτερο σύνδεσμο (L_2). Για τη δημιουργία του άξονα του συνδέσμου L_2 χρειάστηκε να γίνει η κοπή ενός τμήματος από τη ράβδο αλουμινίου, μήκους 115mm. Η κοπή αυτή, όπως και όλες οι κοπές των αξόνων, έγινε με το πριόνι και τη σέγα χειρός. Επίσης, οι διατομές μετά την κοπή υπέστησαν λείανση με το ηλεκτρικό πολυεργαλείο μοντελισμού. Μετά την κοπή του άξονα, ακολουθεί η δημιουργία οπών για τις κοχλίες, σύμφωνα με το σχεδιασμό. Στον άξονα αυτόν χρειάστηκαν τέσσερις οπές διαμέτρου 4mm και δύο οπές διαμέτρου 3mm. Στις δύο οπές διαμέτρου 3mm κοχλιώθηκε ο ένας σερβοκινητήρας, όπως ακριβώς απεικονίζεται στο σχήμα 3.15.



Σχήμα 3.15: Ο δεύτερος άξονας με τις οπές για τις κοχλίες και τον σερβοκινητήρα

Στη συνέχεια φτιάχτηκε ο άξονας του τελικού σημείου δράσης, οποίος είναι και αυτός παράγωγο κοπής της ράβδου αλουμινίου, με μήκος επίσης 115mm. Στον άξονα αυτόν χρειάστηκαν τέσσερις οπές διαμέτρου 4mm, η κάθε μία. Οι δύο οπές χρησιμεύουν στην κοχλίωση της βάσης του μαρκαδόρου, η οποία είναι μία πλαστική βάση για το τηλέφωνο του μπάνιου και αποτελεί υλικό επαναχρησιμοποίησης με προκαθορισμένη γεωμετρία. Η άλλη οπή χρησιμεύει στη δημιουργία μιας περιστροφικής άρθρωσης, η οποία έχει μία κοχλία σαν άξονα περιστροφής, προκειμένου να μπορεί το τελικό σημείο δράσης να διαγράφει μια τροχιά ενός κυκλικού τομέα 30^o κάθετη στο επίπεδο γραφής. Ενώ η τέταρτη οπή είναι για τη σύνδεση του άξονα με το σερβοκινητήρα, μέσω του κομματιού του σύρματος, προκειμένου να δημιουργείται η απαιτούμενη ροπή η οποία χρειάζεται για να στρέψει τον άξονα. Ωστόσο, όπως ειπώθηκε, ως άξονας περιστροφής της άρθρωσης αυτής, είναι μία κοχλία διαμέτρου 4mm. Η κοχλία αυτή στηρίζεται σε δύο γωνίες, οι οποίες είναι τοποθετημένες στις δύο πλευρές του δεύτερου άξονα (L_2) και κοχλιωμένες στις δύο οπές διαμέτρου 4mm του και του άξονα του καρπού. Προκειμένου να μπορούν να τοποθετηθούν σωστά και να μην παρεμποδίζουν της κινήσεις του καρπού, οι δύο αυτές γωνίες, υπέστησαν κατεργασίες κοπής με κόφτη. Η αριστερή γωνία κόπηκε κατά μήκος και η δεξιά κατά ύψος. Επίσης στη δεξιά, πριν την κοχλίωσής της τοποθετήθηκαν ένα περικόχλιο, μία ροδέλα με λάστιχο και μία απλή ροδέλα, προκειμένου η γωνία να καλύψει τα κεφάλια των κοχλιών του σερβοκινητήρα. Η τελική συναρμολόγηση του καρπού φαίνεται στο σχήμα 3.16, σε τρισδιάστατο μοντέλου από το πρόγραμμα σχεδιασμού. Ενώ

στο σχήμα 3.17 απεικονίζεται η συναρμολόγηση του καρπού για τις τρεις όψεις, όπως κατασκευάστηκε στην πραγματικότητα.



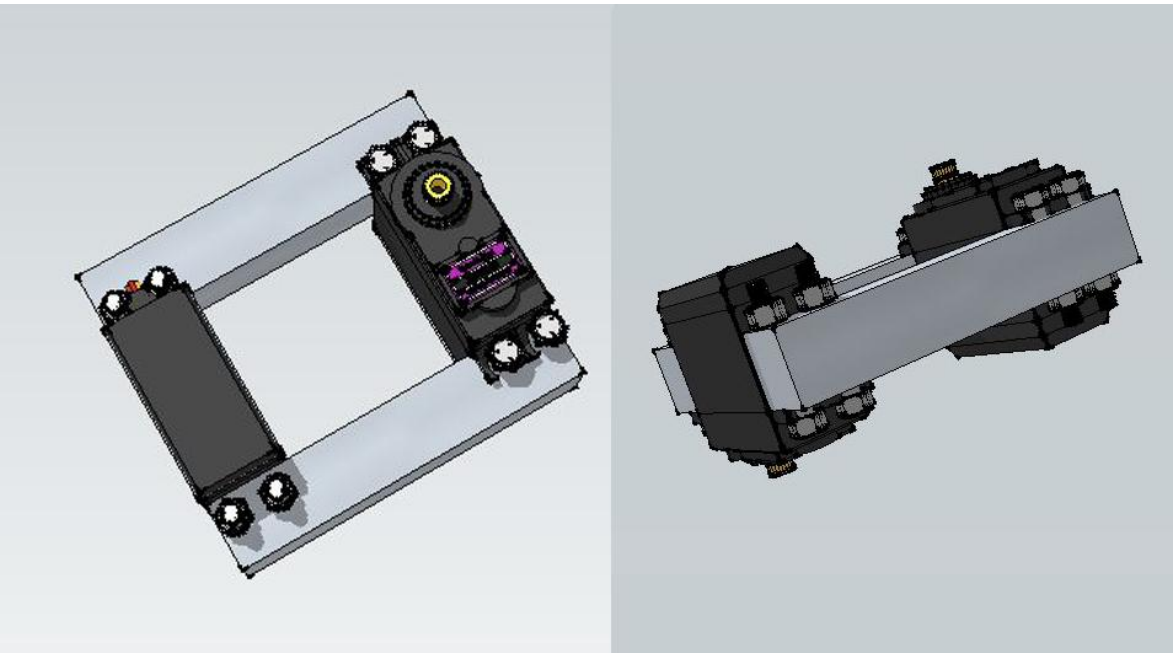
Σχήμα 3.16: Η συναρμολόγηση του καρπού σε 3D μοντέλο

Σχήμα 3.17: Η συναρμολόγηση του καρπού στην πραγματικότητα

Στις δύο ελεύθερες οπές του δεύτερου άξονα, του συνδέσμου L_2, που κατασκευάστηκε, θα πρέπει να κοχλιώσουμε το servo horn, προκειμένου να ενωθεί ο σερβοκινητήρας με αυτό, ώστε να μπορεί να μεταδίδει την κίνηση στον άξονα. Πριν, όμως γίνει αυτό, θα πρέπει να έχει κατασκευαστεί ο υπόλοιπος βραχίονας. Οπότε η σύνδεση αυτή, πραγματοποιείται στο τέλος της κατασκευής του βραχίονα. Συνεπώς ακολουθεί η κατασκευή του άξονα του συνδέσμου L_1 και της βάσης. Ο άξονας του συνδέσμου L_1, είναι αυτός που συνδέει τους

δύο σερβοκινητήρες. Στην πραγματικότητα ο σύνδεσμος L_1 δεν αποτελείται από έναν άξονα, αλλά από δύο ταυτόσημους και παράλληλους άξονες. Αυτό συμβαίνει λόγω της γεωμετρίας των συγκεκριμένων σερβοκινητήρων και των προεξοχών κοχλίωσής τους. Οπότε κατασκευάστηκαν δύο ταυτόσημοι άξονες. Για την κατασκευή των αξόνων αυτών, χρησιμοποιήθηκε ράβδος ξύλου μπάλσα. Σημειώνεται εδώ, ότι στον αρχικό σχεδιασμό οι άξονες αυτοί είναι από ράβδους αλουμινίου, αλλά επειδή το μήκος του συνδέσμου είναι σχετικά μικρό, προτιμήθηκε το ξύλο μπάλσα, για τη μείωση του βάρους. Το μπάλσα είναι γνωστό ότι έχει πολύ μεγάλη αντοχή στην κάμψη σχετικά με το βάρος του.

Ο σύνδεσμος L_1 έχει μήκος 60mm σύμφωνα με τον σχεδιασμό του, όμως ο L_1 είναι η απόσταση του άξονα του ενός σερβοκινητήρα, από τον άξονα του άλλου. Εφόσον ο άξονας τον σερβοκινητήρων βρίσκεται στο μέσον του πλάτους τους, δηλαδή στα 10,5mm, επομένως ο άξονας, του συνδέσμου L_1, με κοχλιωμένους τους δύο σερβοκινητήρες επάνω του, θα πρέπει να έχει ελάχιστο μήκος 81mm. Συνεπώς, από τη ράβδο του ξύλου μπάλσα, κόπηκαν δύο τμήματα μήκους 81mm. Στη συνέχεια γίνεται η κοχλίωση των σερβοκινητήρων στους άξονες, με τις κοχλίες ξύλου. Ο σερβοκινητήρας της βάσης είναι με τον άξονα προς τα κάτω, ώστε να ενωθεί με το servo horn που είναι κοχλιωμένο στη βάση, ενώ ο άλλος σερβοκινητήρας είναι με τον άξονα προς τα πάνω, προκειμένου να συνδεθεί με τον L_2 και τον καρπό. Όμως τα ακραία σημεία των αξόνων και των δύο σερβοκινητήρων θα πρέπει να είναι συνευθειακά σε ευθεία παράλληλη του L_1. Η συναρμολόγιση του άξονα του συνδέσμου L_1, απεικονίζεται σε δύο όψεις και σε τρισδιάστατο μοντέλο στο σχήμα 3.18.



Σχήμα 3.18: Τρισδιάστατη απεικόνιση του άξονα του συνδέσμου L_1

Έπειτα θα πρέπει να ενωθούν οι σερβοκινητήρες με τα αντίστοιχα servo horns, ώστε να ολοκληρωθεί η κατασκευή του βραχίονα. Η ένωση όμως των σερβοκινητήρων δεν γίνεται τυχαία, όσον αφορά στη γωνία τοποθέτησης. Επειδή οι σερβοκινητήρες έχουν ένα συγκεκριμένο εύρος περιστροφής, θα πρέπει να είναι γνωστή και να ληφθεί υπόψη η αρχική θέση του άξονα του σερβοκινητήρα, προκειμένου οι γωνίες που παράγονται κατά τη διάρκεια της εργασίας του βραχίονα, να μην είναι εκτός του εύρους των σερβοκινητήρων. Αυτό το ζήτημα έχει δύο σκέλη, την εύρεση των μέγιστων γωνιών που

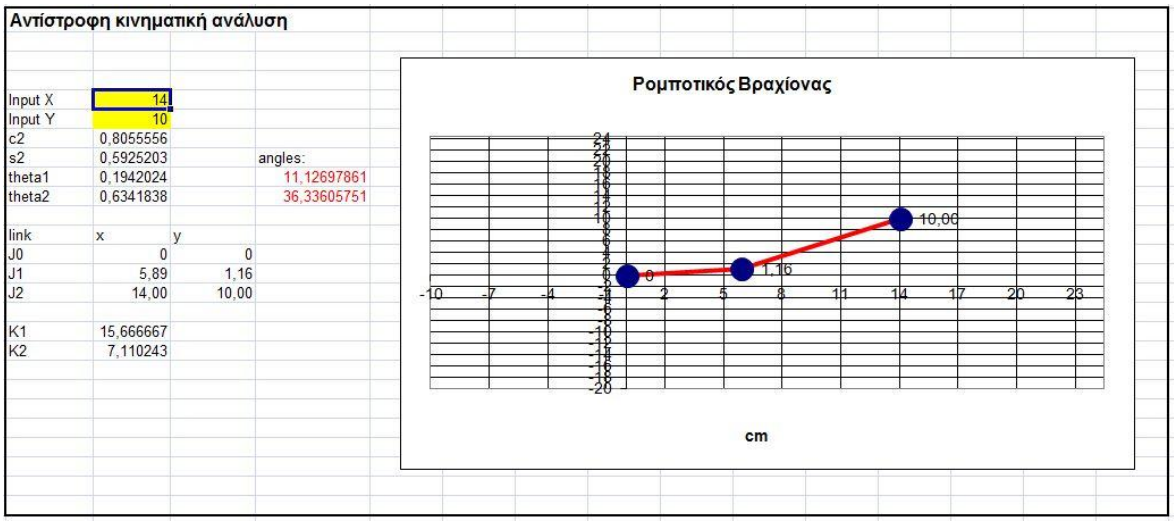
μπορούν να παραχθούν κατά την εργασία, με φορά όπως οι δείκτες του ρολογιού (CW) και την αντίθετη (CCW), καθώς και τη ρύθμιση της αρχικής θέσης του άξονα του κάθε σερβοκινητήρα, στην οποία θα ενωθεί το servo horn. Η επίλυση του πρώτου σκέλους γίνεται με τη προσομοίωση στο Microsoft Excel που έχει προαναφερθεί. Εφόσον είναι γνωστός ο ορθογώνιος χώρος γραφής των γραμμάτων, θα γίνει προσομοίωση ώστε να βρεθούν τα σημεία, αυτού του χώρου, τα οποία παράγουν τις μέγιστες γωνίες για CCW (θετικές γωνίες) και CW (αρνητικές γωνίες), για το θ_1 και το θ_2, με τον αγκώνα κάτω. Οπότε:

Γωνία θ_1

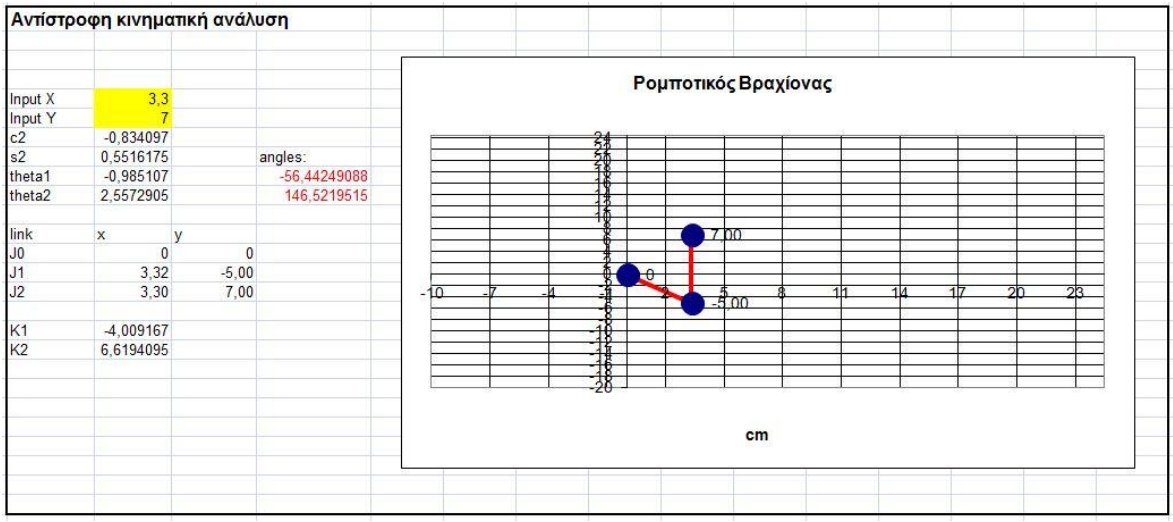
CCW: Το σημείο x=14 και y=10 δίνει θ_1=11,12697861 μοίρες, φαίνεται στο σχήμα 3.19.
CW: Το σημείο x=3,3 και y=7 δίνει θ_1=56,44249088 μοίρες, φαίνεται στο σχήμα 3.20.

Γωνία θ_2

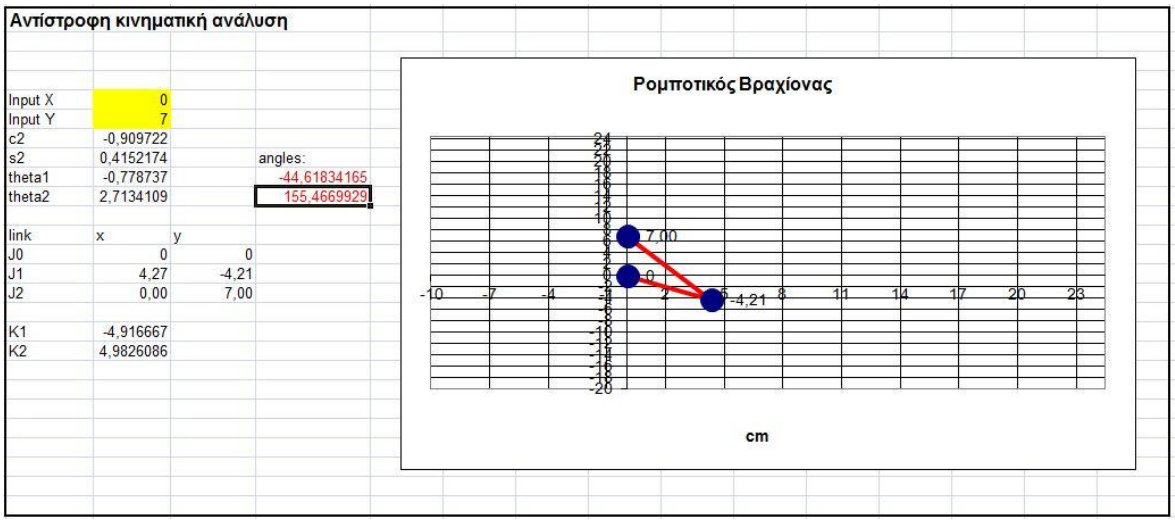
CCW: Το σημείο x=0 και y=7 δίνει θ_1= 155,4669929 μοίρες, φαίνεται στο σχήμα 3.21.
CW: Εφόσον οι κινήσεις το βραχίονα γίνονται κατά σύμβαση με τον αγκώνα κάτω, δεν υπάρχει αρνητική γωνία για το θ_2.



Σχήμα 3.19: Η μέγιστη θετική (CCW), σύμφωνα με τον άξονα xx', γωνία θ_1



Σχήμα 3.20: Η μέγιστη αρνητική (CW), σύμφωνα με τον άξονα xx', γωνία θ_1

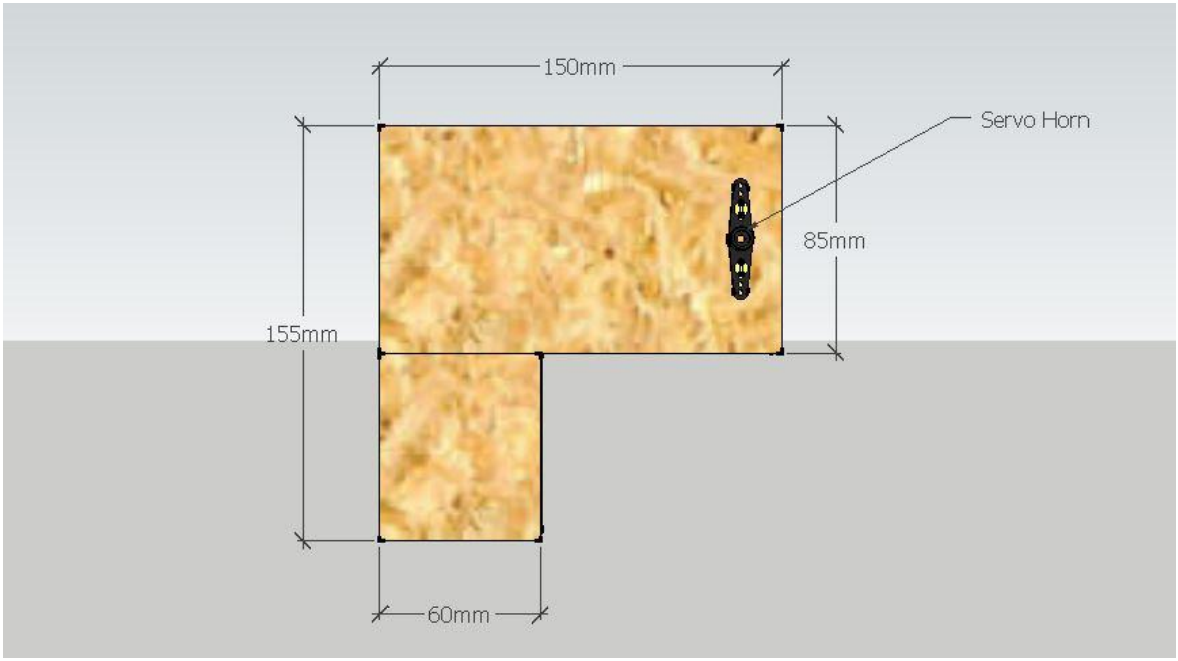


Σχήμα 3.21: Η μέγιστη θετική (CCW), σύμφωνα με τον άξονα L_1, γωνία θ_2

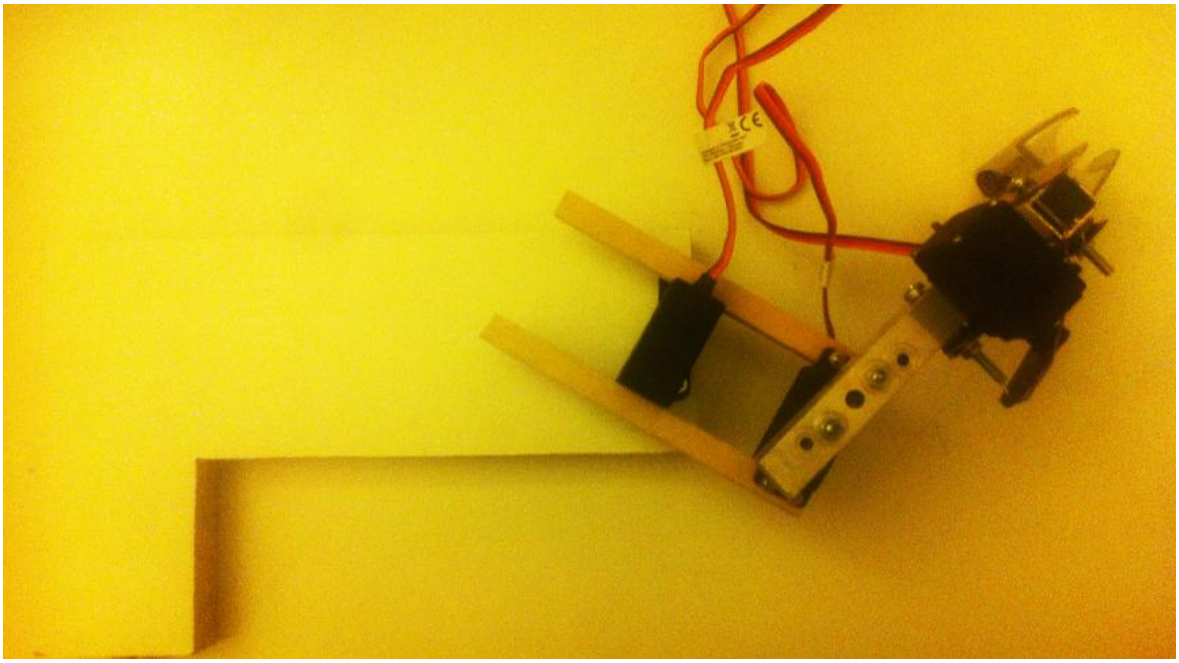
Το εύρος της γωνίας περιστροφής των σερβοκινητήρων CYS S8201 είναι 180 μοίρες, σύμφωνα με τον κατασκευαστή, οπότε καλύπτουν τις παραπάνω γωνίες. Όμως από το έλεγχο που έγινε στους σερβοκινητήρες, παρατηρήθηκε ότι οι σερβοκινητήρες αυτοί, λειτουργούν καλύτερα, όσον αφορά στην ακρίβεια, στην ταχύτητα και στη ροπή τους, από τις 10 έως τις 170 μοίρες. Γενικά για τους περισσότερους σερβοκινητήρες καλό είναι να αποφεύγονται οι ακραίες τιμές των γωνιών τους. Συνεπώς, σύμφωνα με τα παραπάνω, μπορεί να γίνει η τοποθέτηση του σερβοκινητήρα της βάσης στο servo horn της βάσης, όταν αυτός βρίσκεται στις 90 μοίρες. Ενώ η τοποθέτηση του άξονα του συνδέσμου L_2 στον σερβοκινητήρα του αγκώνα μπορεί να γίνει όταν αυτός βρίσκεται στις 10 μοίρες. Αυτό είναι το δεύτερο σκέλος του ζητήματος. Για να επιτευχθεί λοιπόν η θέση των σερβοκινητήρων ακολουθείται μια διαδικασία που ονομάζεται **calibration**. Το calibration μπορεί να γίνει εύκολα χρησιμοποιώντας το παράδειγμα της παραγράφου 2.3.5 για τον κάθε σερβοκινητήρα ξεχωριστά, θέτοντας στον κώδικα τις επιθυμητές γωνίες. Ωστόσο για να φαίνεται σε ποιες γωνίες πηγαίνει ο άξονας του σερβοκινητήρα, αρκεί να ενωθεί με ένα servo horn τοποθετημένο έτσι ώστε στις 0 μοίρες να είναι παράλληλος με το μήκος του σερβοκινητήρα. Στο σχήμα 3.22 φαίνεται η διαδικασία για τις -45, 0 και 45 μοίρες.

Σχήμα 3.22: Το calibration των σερβοκινητήρων

Ωστόσο η βάση του βραχίονα είναι κατασκευασμένη από μοριοσανίδα ξύλου επενδεδυμένη με μελαμίνη συνολικού πάχους 25mm, κομμένη από εξωτερικό συνεργάτη με μηχανή CNC, ενώ η γεωμετρία της φαίνεται στο σχήμα 3.23. Επομένως η τελική συναρμολόγηση του βραχίονα είναι γεγονός και απεικονίζεται σε πραγματικό μοντέλο, στο σχήμα 3.24.



Σχήμα 3.23: Η βάση του βραχίονα με κοχλιωμένο servo horn



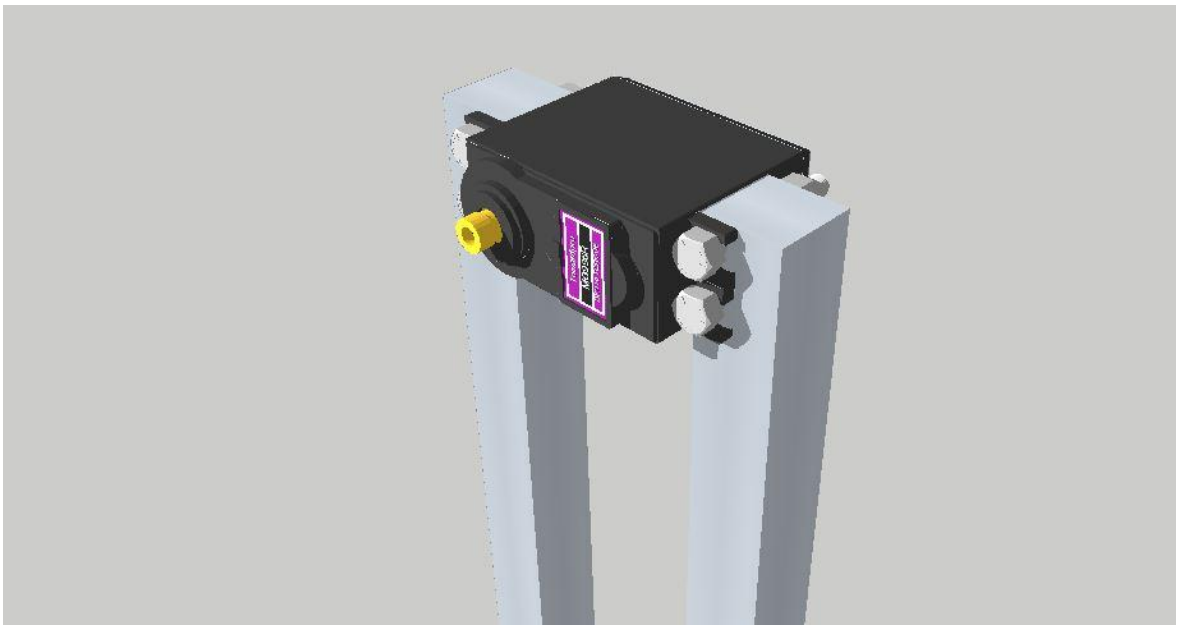
Σχήμα 3.24: Η τελική πραγματική συναρμολόγηση του βραχίονα

3.2.2 Κατασκευή καθαριστήρα

Μετά το δύσκολο κομμάτι της κατασκευής και της συναρμολόγισης του βραχίονα, έπεται η κατασκευή του καθαριστήρα. Σύμφωνα με το σχεδιασμό του καθαριστήρα, θα χρειαστούν:

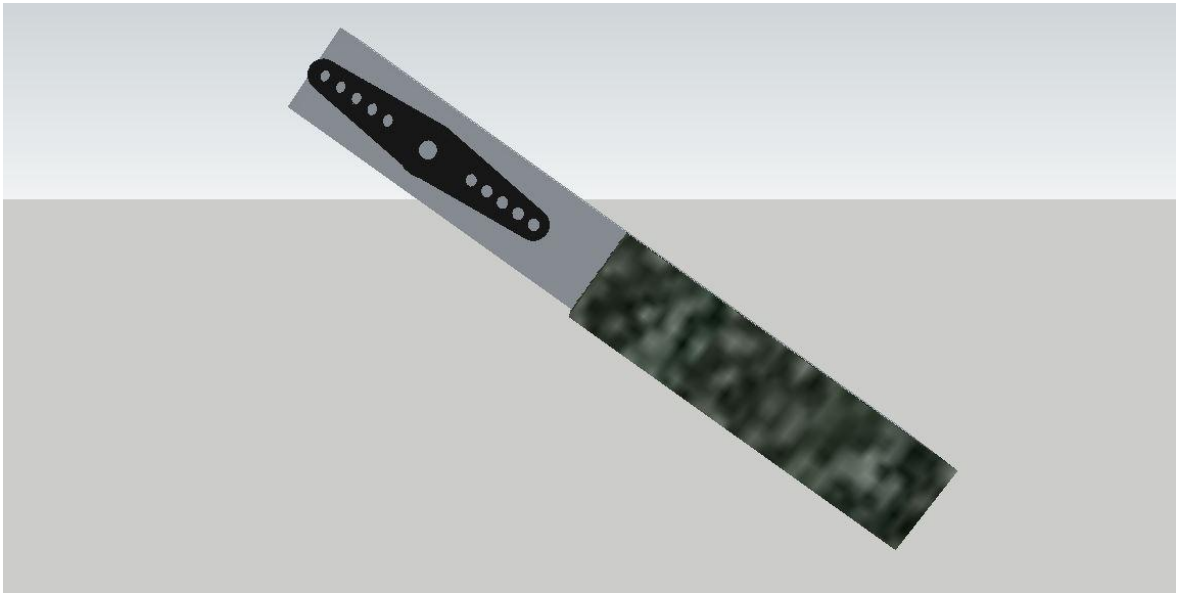
- Ένας σερβοκινητήρας με ένα servo horn.
- Δύο άξονες από τη ράβδο αλουμινίου μήκους 254mm ο καθένας.
- Ένα κομμάτι από φύλλο μπάλσα πάχους 10mm, μήκους 120mm και πλάτους 20mm.
- Τέσσερις κοχλίες με περικόχλια διαμέτρου 3mm.
- Κόλλα
- Σφουγγάρι μήκους 90mm, πλάτους 20mm και πάχους 20mm.

Με κοπή, με τη βοήθεια του πριονιού και της σέγας χειρός, δημιουργούνται οι δύο άξονες αλουμινίου, οι οποίοι χρησιμοποιούνται στη στήριξη του καθαριστήρα στον πίνακα γραφής. Ωστόσο οι άξονες αυτοί, όπως και όλα τα τμήματα, υπόκεινται σε λείανση, στις επιφάνειες κοπής, με τη βοήθεια του πολυεργαλείου μοντελισμού. Στη συνέχεια γίνεται διάτρηση για τη δημιουργία δύο οπών διαμέτρου 3mm στον κάθε άξονα, προκειμένου να κοχλιωθεί ο σερβοκινητήρας, με τις κοχλίες και τα περικόχλια, όπως φαίνεται στο σχήμα 3.25.

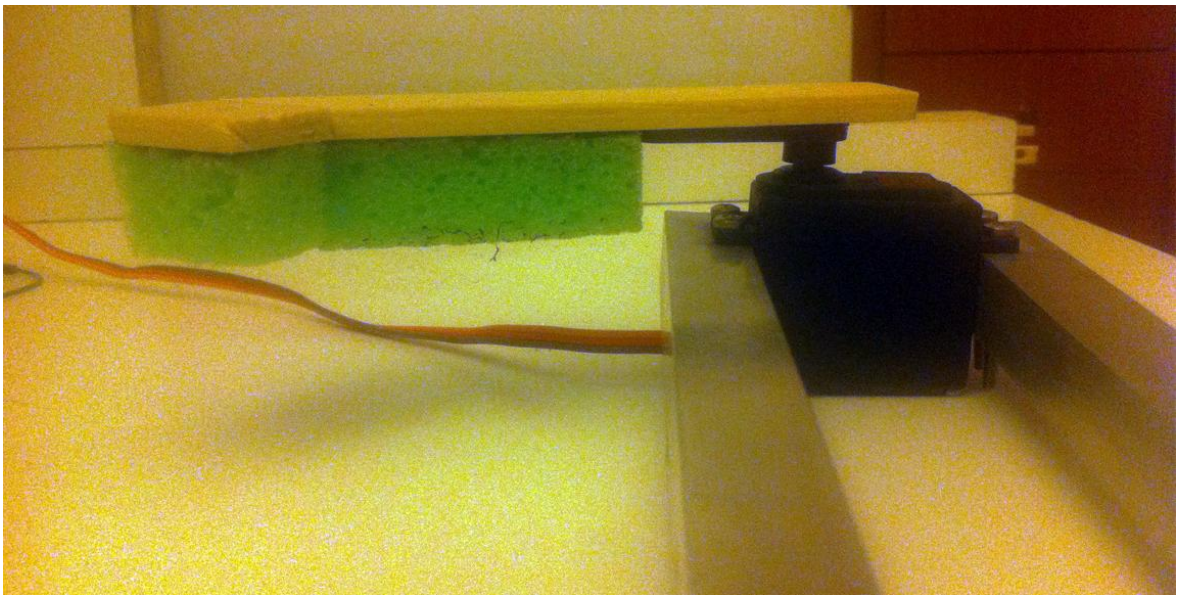


Σχήμα 3.25: Ο σερβοκινητήρας κοχλιωμένος στους δύο άξονες αλουμινίου

Έπειτα με τη βοήθεια του κοπιδιού, δημιουργείται το τμήμα από ξύλο μπάλσα και το τμήμα από το σφουγγάρι. Στο τμήμα από μπάλσα συγκολλείται το σφουγγάρι και το servo horn, με εποξική κόλλα, ώστε να προκύψει ο άξονας του καθαριστήρα, όπως φαίνεται στο σχήμα 3.26. Εδώ σημειώνεται ότι στον αρχικό σχεδιασμό του το τμήμα αυτό ήταν από αλουμίνιο, αλλά για τη μείωση του βάρους επιλέχθηκε το μπάλσα. Στη συνέχεια συνδέεται το servo horn με τον σερβοκινητήρα, ενώ ο σερβοκινητήρας είναι στη θέση με 0 μοίρες. Για το λόγο αυτό καλό είναι να προηγηθεί το calibration και αυτού του σερβοκινητήρα, όπως έγινε στην κατασκευή του βραχίονα, παραπάνω. Μετά τη σύνδεση του σερβοκινητήρα στη σωστή θέση με το servo horn, η τελική συναρμολόγιση του καθαριστήρα είναι στην πραγματικότητα, όπως φαίνεται στο σχήμα 3.27.



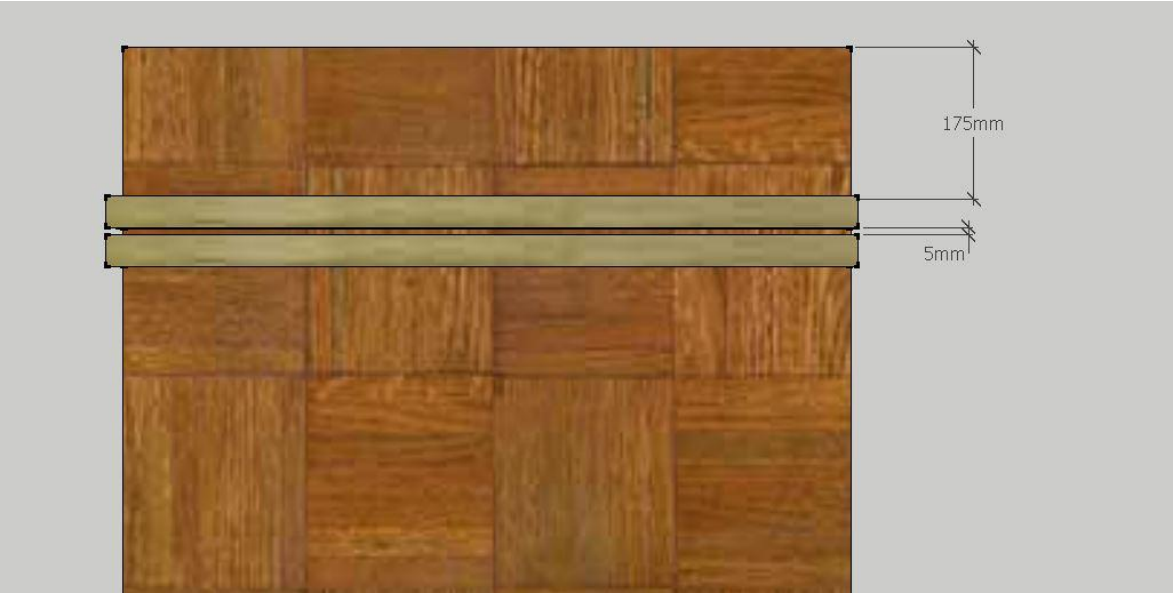
Σχήμα 3.26: Ο άξονας του καθαριστήρα



Σχήμα 3.27: Η πραγματική τελική συναρμολόγηση του καθαριστήρα

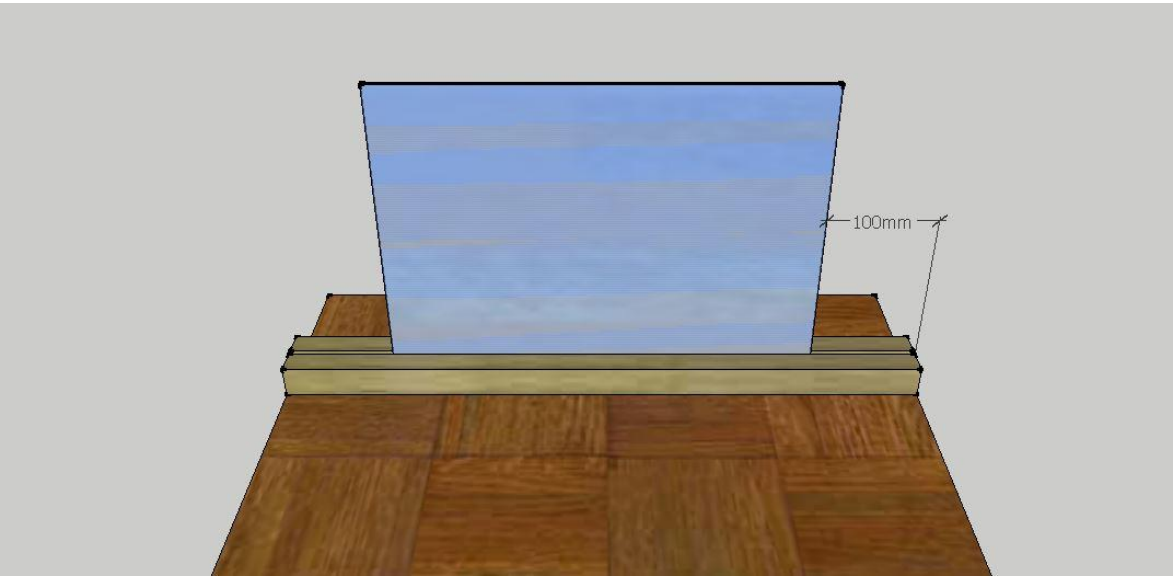
3.2.3 Κατασκευή αίθουσας

Η κατασκευή της αίθουσας, όπως και ο σχεδιασμός της, χωρίζεται σε τρία βασικά μέρη, το δάπεδο με τη βάση του πίνακα, τον πίνακα και τα έξι τραπεζοκαθίσματα. Το δάπεδο, όπως και η βάση του πίνακα, είναι κατασκευασμένα από μοριοσανίδα ξύλου επενδεδυμένη με μελαμίνη, συνολικού πάχους 25mm, κομμένα βάσει σχεδίου, από εξωτερικό συνεργάτη, σε μηχανή CNC. Ο πίνακας είναι προϊόν επαναχρησιμοποίησης. Πρόκειται για μια κορνίζα φωτογραφιών αποτελούμενη από τζάμι και πεπιεσμένο χαρτί. Τα τραπεζοκαθίσματα είναι κατασκευασμένα εξολοκλήρου από μπάλσα, ενώ η κατασκευή τους περιγράφεται παρακάτω. Συνεπώς η κατασκευή της αίθουσας ολοκληρώνεται σε τρία βήματα. Το πρώτο βήμα αφορά στην κατασκευή της βάσης του πίνακα. Συνεπώς τα δύο τμήματα που αποτελούν τη βάση, καρφώνονται στο δάπεδο, με έξι καρφιά ξύλου το κάθε ένα, με κενό 5mm μεταξύ τους, όπως φαίνεται στο σχήμα 3.28.



Σχήμα 3.28: Η βάση τοποθετημένη στο δάπεδο

Στο δεύτερο βήμα γίνεται η τοποθέτηση του πίνακα ανάμεσα στις δύο δοκούς της βάσης, όπως φαίνεται στο σχήμα 3.29.



Σχήμα 3.29: Η στήριξη του πίνακα στη βάση

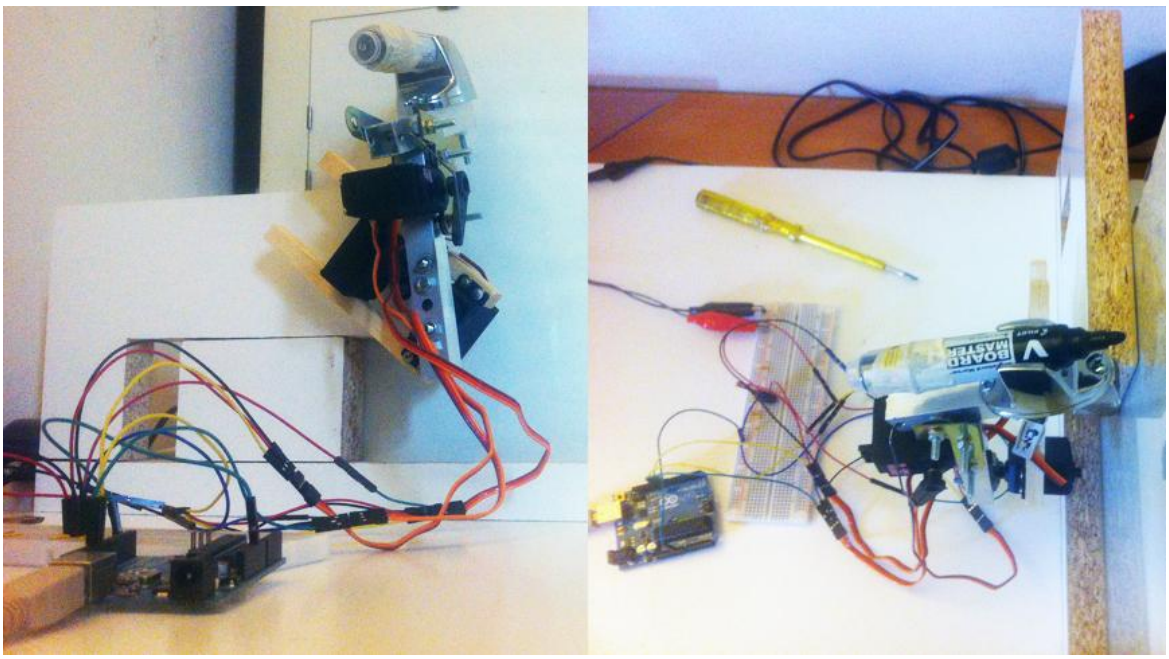
Στο τρίτο και τελευταίο βήμα κατασκευάζονται τα τραπέζια και τα καθίσματα. Η κατασκευή ενός τραπεζιού απαιτεί την κοπή, με κοπίδι, ενός ορθογώνιου τμήματος, από το φύλλο μπάλσα πάχους 3mm, με τις διαστάσεις που δίνονται στο σχεδιασμό και την κοπή δύο τετραγωνικών τμημάτων, για τα πλαϊνά στηρίγματα του τραπεζιού, με διαστάσεις 50mm x 50mm. Τα καθίσματα αποτελούνται από δύο τμήματα από φύλλο μπάλσα πάχους 3mm, διαστάσεων 30mm x 80mm, ενωμένα, με κόλλα, κάθετα μεταξύ τους ώστε να σχηματίζουν ένα ισόπλευρο "Γ" και ένα τμήμα από το ίδιο φύλλο μπάλσα 80mm x 100mm, για την πλάτη. Εφόσον έχουν κατασκευαστεί έξι τραπέζια και έξι καθίσματα, στη συνέχεια

συγκολλούνται με δάπεδο με κόλλα σε δύο συστοιχίες. Η τελική πραγματική συναρμολόγηση της αίθουσας απεικονίζεται στο σχήμα 3.30.

Σχήμα 3.30: Η πραγματική συναρμολόγηση της αίθουσας

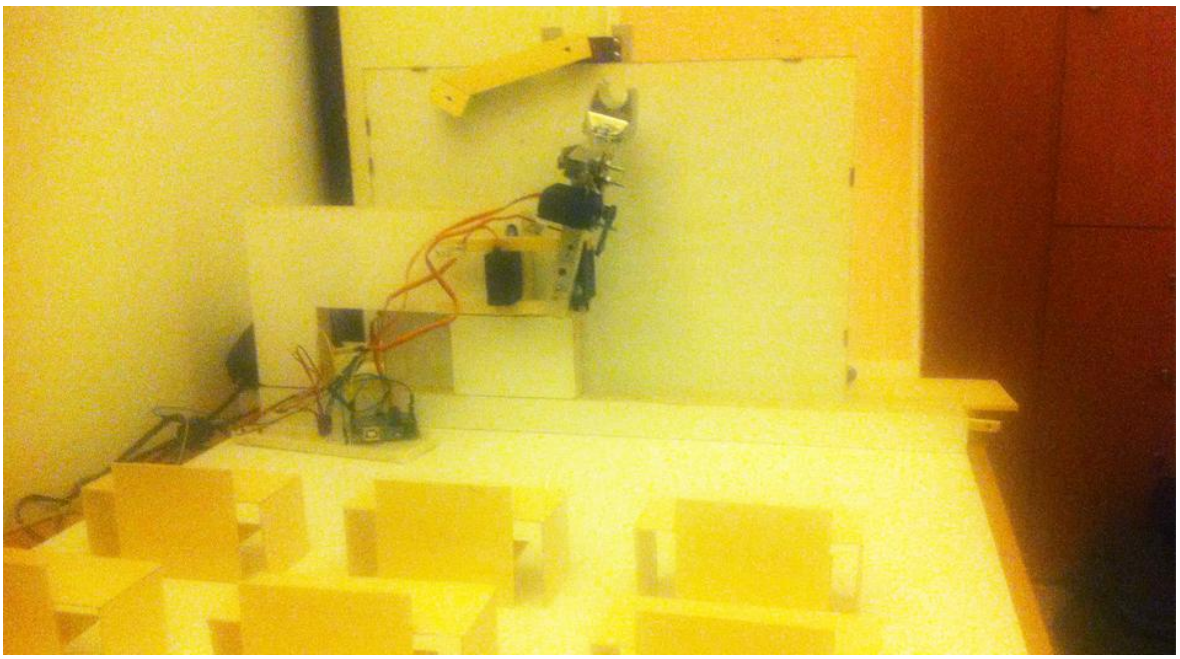
3.2.4 Συναρμολόγηση διάταξης

Εφόσον έχουν κατασκευαστεί όλα τα επιμέρους τμήματα της διάταξης, στη συνέχεια γίνεται η τελική της συναρμολόγηση. Η συναρμολόγηση αυτή περιλαμβάνει την τοποθέτηση του βραχίονα και την τοποθέτηση του καθαριστήρα στην αίθουσα. Σύμφωνα με τον εικονικό σχεδιασμό τοποθετείται η βάση του βραχίονα επάνω στη βάση του πίνακα. Η τοποθέτηση αυτή γίνεται εύκολα, χρησιμοποιώντας δύο ξύλινες καβίλιες και κόλλα. Οι καβίλιες που θα χρειαστούν έχουν διάμετρο διατομής 8mm και μήκος 40mm. Συνεπώς θα χρειαστεί να γίνει διάτρηση σε δύο σημεία της κάτω πλευράς της βάσης, με τρυπάνι 8mm για ξύλο. Θα πρέπει να γίνουν επίσης οι αντίστοιχες οπές και στον έναν άξονα της βάσης του πίνακα. Έπειτα τοποθετείται κόλλα στις οπές κατά μήκος και σε όλο το πλάτος, των δύο, προς ένωση, πλευρών. Οι καβίλιες προσαρμόζονται κατά το ήμισυ στις οπές της βάσης του βραχίονα και στις οπές του άξονα της βάσης του πίνακα, προκειμένου να δημιουργηθεί το αποτέλεσμα του σχήματος 3.31. Στη συνέχεια γίνεται η τοποθέτηση του καθαριστήρα με κόλλα, στο μέσον του πίσω μέρους του πίνακα, όπως φαίνεται στο σχεδιασμό του και όπως απεικονίζεται στο σχήμα 3.32. Η τελική συναρμολόγηση απεικονίζεται όπως είναι στην πραγματικότητα στο σχήμα 3.33.



Σχήμα 3.31: Η τοποθέτηση του βραχίονα

Σχήμα 3.32: Η τοποθέτηση του καθαριστήρα

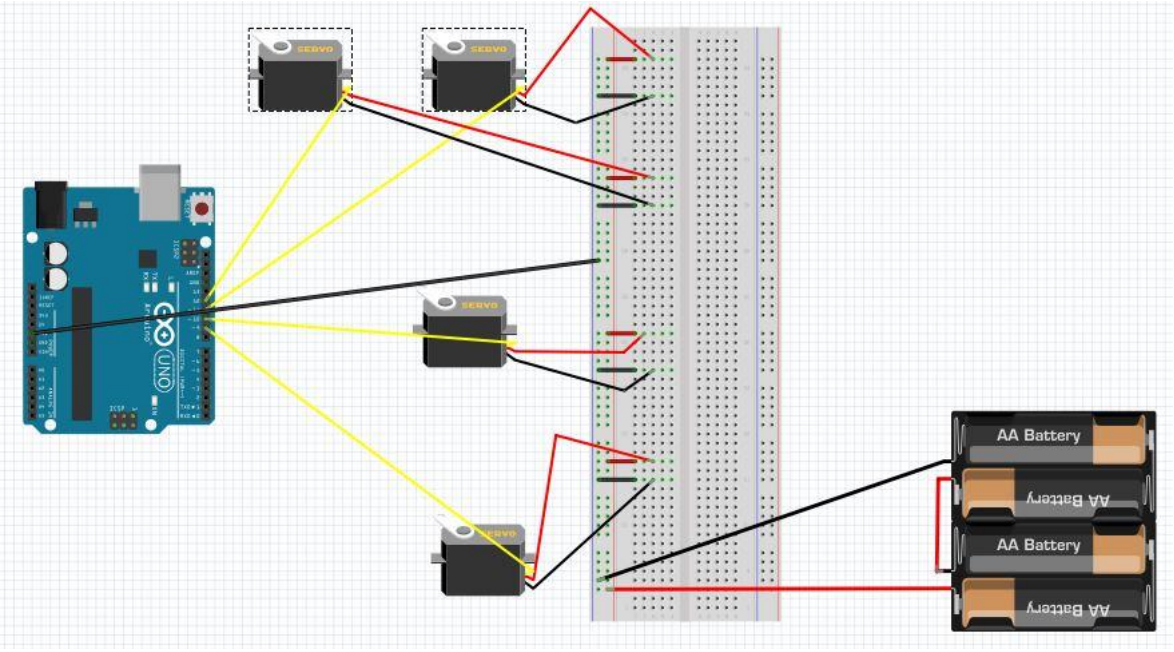


Σχήμα 3.33: Η τελική συναρμολόγηση της διάταξης

3.3 Προγραμματισμός

Το επόμενο στάδιο μετά την τελική συναρμολόγηση της διάταξης, είναι η λειτουργία της, η οποία στηρίζεται στο προγραμματισμό του ρομποτικού συστήματος που κατασκευάστηκε, προκειμένου, αυτό να είναι ικανό να εκτελέσει τις βασικές του εργασίες. Οι βασικές εργασίες του ρομποτικού συστήματος, όπως έχουν προαναφερθεί, είναι η γραφή λέξεων σε πίνακα, οι οποίες εισάγονται μέσω υπολογιστή από το χρήστη και ο καθαρισμός του πίνακα γραφής. Η γραφή των λέξεων επιτυγχάνεται από το ρομποτικό βραχίονα, ενώ ο καθαρισμός του πίνακα επιτυγχάνεται από τον καθαριστήρα. Συνεπώς το σύστημα αυτό χωρίζεται και προγραμματιστικά σε αυτά τα δύο επιμέρους συστήματα, το οποία όμως, επικοινωνούν μεταξύ τους και συνεργάζονται. Ο έλεγχος και η επικοινωνία του συνολικού συστήματος επιτυγχάνεται μέσω της πλατφόρμας Arduino Uno, η οποία είναι συνδεδεμένη στον υπολογιστή.

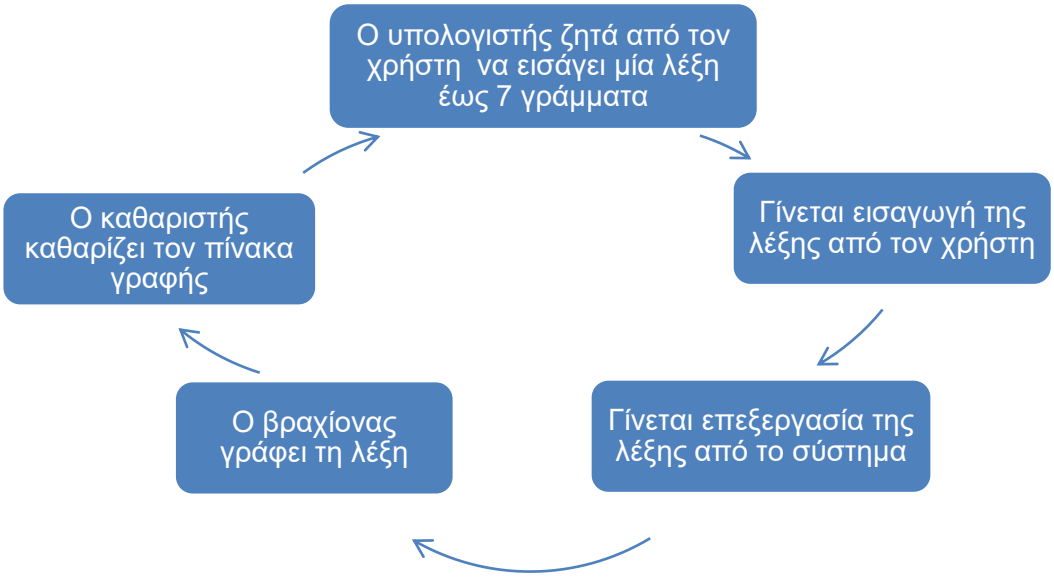
Επομένως, για να υπάρξει η δυνατότητα προγραμματισμού του συστήματος, θα πρέπει πρώτα να γίνει η σύνδεση των επιμέρους συστημάτων με την πλατφόρμα Arduino Uno και να εξασφαλιστεί η παροχή ρεύματος σε αυτά. Όπως έχει προαναφερθεί, οι σερβοκινητήρες των αρθρώσεων χρειάζονται τάση 6V κατά το ελάχιστο, ενώ οι άλλοι δύο σερβοκινητήρες λειτουργούν με τάση από 4,8V έως 7,2V. Συνεπώς η συνολική τάση θα πρέπει να είναι για ασφάλεια στα 6V σε όλο το σύστημα, κάτι που δε γίνεται μέσω του Arduino, αλλά μόνο με εξωτερική τροφοδοσία. Επίσης, όπως αναλύθηκε στη θεωρία, η σύνδεση όλων των σερβοκινητήρων με το Arduino, θα γίνει μέσω των PWM pins. Στο σχήμα 3.34 απεικονίζεται, σε σχέδιο εξαγόμενο από το πρόγραμμα **Fritzing**, όλη η συνδεσμολογία.



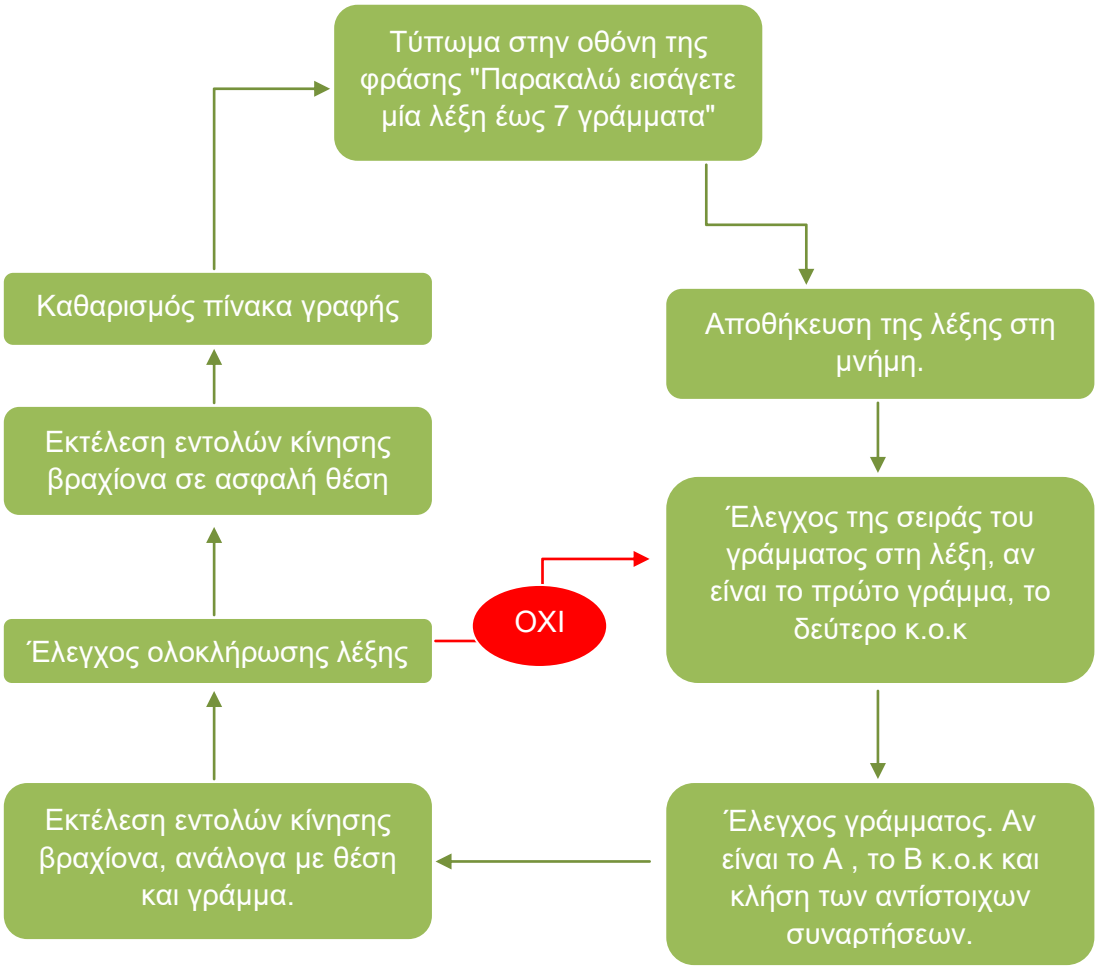
Σχήμα 3.34: Η τελική συνδεσμολογία του συστήματος

3.3.1 Δομή προγράμματος

Εφόσον η ολοκλήρωση της συνδεσμολογίας του συστήματος είναι επιτυχής, ακολουθεί ο προγραμματισμός του. Συνεπώς θα πρέπει να συνταχθεί ένα πρόγραμμα σε γλώσσα Wiring, το οποίο θα ελέγχει όλες τις λειτουργίες του συστήματος και το οποίο θα εκτελείται από το Arduino και εν συνεχεία από το σύστημα. Επομένως θα πρέπει να οριστούν, πιο ειδικά, οι λειτουργίες της διάταξης, ώστε να οριστεί και η λογική που θα ακολουθηθεί στο πρόγραμμα. Σύμφωνα με τη φυσική ροή των εργασιών μπορεί να δημιουργηθεί ένα απλό διάγραμμα ροής με τα γεγονότα όπως αυτά πρόκειται να συμβούν στην πραγματικότητα. Στο σχήμα 3.35, είναι το διάγραμμα της φυσικής ροής των γεγονότων. Επομένως ο κώδικας συντάσσεται έτσι, ώστε να συνάδει με κάθε στάδιο του διαγράμματος φυσικής ροής γεγονότων. Στο σχήμα 3.36, απεικονίζεται ένα απλό διάγραμμα ροής της λογικής που ακολουθεί ο κώδικας.



Σχήμα 3.5: Διάγραμμα φυσικής ροής γεγονότων



Σχήμα 3.36: Διάγραμμα λογικής ροής του κώδικα

3.3.2 Συγγραφή προγράμματος

Βάσει των παραπάνω διαγραμμάτων ροής γίνεται η συγγραφή του προγράμματος με γλώσσα Wiring στο IDE του Arduino, σε επιμέρους τμήματα τα οποία παρουσιάζονται στη συνέχεια σε βήματα.

Βημα 1ο: Το πρόγραμμα θα πρέπει να τυπώνει στη σειριακή οθόνη του IDE, τη φράση: "Parakalw eisagete mia leksi ews 7 grammata". Σημειώνεται εδώ ότι το IDE του Arduino δέχεται μόνο λατινικούς χαρακτήρες. Στη συνέχεια το πρόγραμμα αναμένει την εισαγωγή της λέξης από τον χρήστη, προκειμένου να προχωρήσει στο έλεγχό της. Συνεπώς ο κώδικας είναι:

```
Serial.println("Parakalw eisagete mia leksi ews 7 grammata:\n");
while (Serial.available()==0) {
}
```

Εφόσον ο χρήστης εισάγει τη λέξη, το πρόγραμμα οφείλει να κάνει τον έλεγχο, αν η λέξη έχει 7 γράμματα. Αυτό μπορεί να γίνει με τη δήλωση της λέξης ως διάνυσμα χαρακτήρων στην αρχή του προγράμματος. Συνεπώς η λέξη δηλώνεται ως εξής:

```
char leksi[6];
```

Ενώ διαβάζεται και αποθηκεύεται από το πρόγραμμα με την εντολή:
int byteAvailable = Serial.available();

Βήμα 2ο: Το πρόγραμμα πρέπει να αναγνωρίζει σε ποια θέση, μέσα στη λέξη, είναι το γράμμα που πρόκειται να γραφεί, καθώς επίσης και ποιο γράμμα της αλφαβήτου είναι αυτό. Συνεπώς η θέση και το γράμμα θα πρέπει να ορίζονται βάσει μίας κοινής μεταβλητής, η οποία θα αλλάζει ανάλογα με τις επαναλήψεις του ελέγχου. Αυτή η μεταβλητή δηλώνεται στην αρχή του προγράμματος ως: int i=0 ή σαν όρισμα μιας συνάρτησης, οποία δηλώνεται επίσης στην αρχή του προγράμματος, δηλαδή πριν την κύρια συνάρτηση void setup(), όπως στην προκειμένη περίπτωση. Η συνάρτηση η οποία σαν όρισμα έχει τη μεταβλητή (int i), είναι η συνάρτηση ελέγχου της θέσης του γράμματος:

void startpoint(int i)

Ονομάστηκε startpoint, για το λόγω του ότι αποτελεί τη συνάρτηση που ορίζει στο βραχίονα το αρχικό σημείο, κατά τον άξονα XX', γραφής του κάθε γράμματος. Επίσης το (int i), δηλώνει και ποιο γράμμα είναι σε κάθε επανάληψη του ελέγχου με την εντολή:

leksi[i] = Serial.read();

Επομένως αρκεί ένας κώδικας επανάληψης για το (int i), ο οποίος να περιέχει τις παραπάνω συναρτήσεις, καθώς και τις περιπτώσεις των γραμμάτων, προκειμένου να καλούνται κάθε φορά οι αντίστοιχες συναρτήσεις κίνησης του βραχίονα, προκειμένου να γράφει το σωστό γράμμα. Συνεπώς ο κώδικας αυτός έχει τη μορφή:

```
for(int i=0; i<7; i++){
        startpoint(i);
        delay(3000);
        leksi[i] = Serial.read();
        switch (leksi[i])
        {
          case 'A':
          Serial.print("Parakalw perimenete na zwgrafisw to A\n");
                        delay(1000);
                        Adraw();
                        delay(5000);
                        break;
          case 'B':
          Serial.print("Parakalw perimenete na zwgrafisw to B\n");
                       delay(1000);
                        Bdraw();
                        delay(5000);
                        break;
        }
}
```

Παρατηρείται εδώ ότι ο κώδικας περιέχει μόνο τις περιπτώσεις για τα γράμματα Α και Β. Θα πρέπει να γραφούν οι περιπτώσεις και για τα 24 γράμματα τις αλφαβήτου. Επιπλέον κάθε φορά που ο βραχίονας γράφει ένα γράμμα, τυπώνεται στην οθόνη τη μήνυμα:

"Parakalw perimenete na zwgrafisw to (εκάστοτε γράμμα)"

Επίσης οι καθυστερήσεις που εισάγονται στον κώδικα, είναι σημαντικές και προκύπτουν κυρίως πειραματικά. Οι συναρτήσεις Adraw(); και Bdraw(); θα αναλυθούν στο τρίτο βήμα.

Βήμα 3ο: Για την κίνηση του βραχίονα χρειάστηκε να κατασκευαστούν κάποιες βασικές εντολές κίνησης, των οποίων τα ονόματα είναι επηρεασμένα από τον κώδικα G των εργαλειομηχανών CNC. Η πρώτη εντολή που κατασκευάζεται είναι η G00 και είναι αυτή που αναγκάζει το βραχίονα να κινείται σε κάποιο σημείο με μέγιστη ταχύτητα και χωρίς να γράφει. Όπως παραδείγματος χάρη σε μία θέση ασφαλείας. Η εντολή G00 είναι η εξής:

```
void G00(float x_in, float y_in){
      Penup();
      delay(3000);
      FixCoordinates(x_in,y_in);
      InverseKin();
      Servo1.write(Servo1_angle,20,true);
      Servo2.write(Servo2_angle,20,true);
}
```

Η δεύτερη εντολή που κατασκευάζεται είναι η G01 και είναι αυτή που αναγκάζει το βραχίονα να κινείται ευθύγραμμα, από το σημείο που βρίσκεται σε κάποιο άλλο, με ταχύτητα που ορίζεται από τη ρουτίνα κάθε γράμματος. Η εντολή G01 είναι η εξής:

```
void G01(float x_in ,float y_in ,float steps,float e,float u,int speed1,int speed2,float vima){
     for (float dx=0; dx<=steps; dx=dx+vima){
         FixCoordinates(x_in +(e*dx),y_in +(u*dx));
         InverseKin();
         Servo1.write(Servo1_angle,speed1,true);
        Servo2.write(Servo2_angle,speed2,false);
        }
}
```

Η τρίτη και τελευταία βασική εντολή κίνησης, που κατασκευάζεται είναι η G02 και είναι αυτή που αναγκάζει το βραχίονα να διαγράφει κυκλική τροχιά CW, από το σημείο που βρίσκεται σε κάποιο άλλο, με προκαθορισμένη ταχύτητα ,ενώ η ακτίνα της ορίζεται από τη ρουτίνα κάθε γράμματος. Η εντολή G02 είναι η εξής:

```
void G02(float CircleCenterX,float CircleCenterY, float Radius_x, float Radius_y, float
Cfrom, float Cto){
    for (float z = Cfrom; z>-Cto; z=z-0.2){
     x = Radius_x * cos(z *(pi/180)) + CircleCenterX;
     y = Radius_y * sin(z *(pi/180)) + CircleCenterY;
     FixCoordinates(x,y);
     InverseKin();
     Servo1.write(Servo1_angle,80,true);
     Servo2.write(Servo2_angle,70,true);
      }
}
```

Στις παραπάνω εντολές κίνησης, παρατηρείται η συνάρτηση InverseKin(). Είναι η συνάρτηση που επιλύει το αντίστροφο κινηματικό πρόβλημα, μόνο για τις λύσεις

ΑΓΚΩΝΑΣ ΚΑΤΩ, υπολογίζει τις γωνίες θ_1 και θ_2 και ορίζει τις γωνίες περιστροφής των σερβοκινητήρων (Servo_angle). Σημειώνεται εδώ, όπως φαίνεται και στον παρακάτω κώδικα της InverseKin(), ότι η γωνίες των σερβοκινητήρων διαφέρουν από τις γωνίες θ_1 και θ_2, λόγω του calibration σερβοκινητήρων. Επομένως ο κώδικας της InverseKin(), είναι ο εξής:

```
void InverseKin(){
  float c;
  float s;
  float v;
  float w;
  float th_1;
  float th_2;
  c=(sqrt(sq(x)+sq(y)));
  s= atan2(y,x);
  v= acos((sq(x)+sq(y)+sq(a)-sq(b))/(2*a*c));
  w= acos((sq(x)+sq(y)-sq(a)-sq(b))/(2*a*b));

  th_2=(180/pi)* w;
  th_1=(180/pi)*(s-v);

  Servo1_angle = (th_1+96);
  Servo2_angle = (th_2+18);
}
```

Στη συνέχεια θα πρέπει να κατασκευαστούν οι εντολές που ανεβοκατεβάζουν το μαρκαδόρο, προκειμένου να γράφει ή όχι. Σύμφωνα με τη ρύθμιση του σερβοκινητήρα του καρπού, ο μαρκαδόρος σηκώνεται στη θέση με γωνία 130 μοίρες και κατεβαίνει στη θέση με γωνία 110 μοίρες. Οπότε οι εντολές αυτές είναι:

```
void Penup()
{
 Servo3.write(130);
 delay(500);
}

void Pendown()
{
  Servo3.write(110);
  delay(500);
}
```

Ωστόσο, στο δεύτερο βήμα, αναφέρθηκε η συνάρτηση void startpoint(int i). Η συνάρτηση αυτή συγκεκριμένα, ορίζει σε κάθε επανάληψη του (int i), την απόσταση κατά την οποία πρέπει να μετακινηθεί το νέο σημείο γραφής του επόμενου γράμματος. Συνεπώς πρέπει να οριστεί μία νέα μεταβλητή, η οποία θα αλλάζει ανάλογα με τις τιμές του (int i). Αυτή η μεταβλητή ορίζεται ως float xstart=0, στην αρχή του προγράμματος. Επομένως η συνάρτηση void startpoint(int i), είναι η εξής:

```
void startpoint(int i){
                if (i==0) {
```

```
                xstart=xstart;
                }
                    else if (i==1) {
                    xstart=xstart+2;
                    }
                    else if (i==2) {
                    xstart=xstart+2;
                    }
                    else if (i==3) {
                    xstart=xstart+2;
                    }
                    else if (i==4) {
                    xstart=xstart+2;
                    }
                    else if (i==5) {
                    xstart=xstart+2;
                    }
            else if (i==6) {
                    xstart=xstart+2;
                    }
}
```

Μετά την κατασκευή όλων των παραπάνω, ακολουθεί η κατασκευή των συναρτήσεων που ορίζουν τις κινήσεις γραφής κάθε γράμματος. Για το γράμμα Α η συνάρτηση Adraw() είναι η εξής:

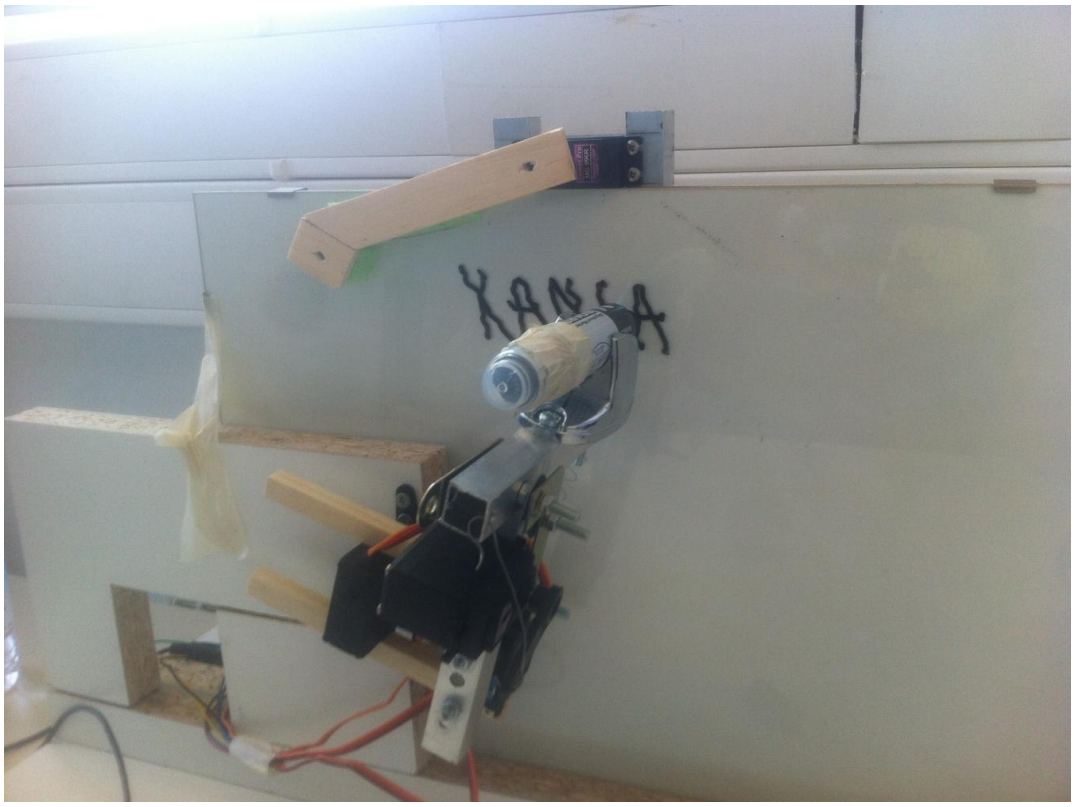
```
void Adraw(){
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0.166666,1,40,40,0.01);
  delay(3000);
  G01(xstart+1,10,3,0.166666,-1,40,40,0.01);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.75,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.75,8.5,1.5,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);}
```

Όπως έχει προαναφερθεί, μετά το τέλος της γραφής κάθε γράμματος, ο κώδικας κάνει επανάληψη για την εύρεση του επόμενου γράμματος. Στην περίπτωση που δεν υπάρχει άλλο γράμμα ή η λέξη που έχει γραφεί έχει ήδη επτά γράμματα, το πρόγραμμα γραφής τερματίζεται και ο βραχίονας απομακρύνεται σε μία προκαθορισμένη θέση ασαφαλείας, που ορίζεται από τη συνάρτηση safepos().

Βήμα 4ο: Μετά τη θέση ασφαλείας, αναλαμβάνει ο καθαριστήρας να σβήσει τον πίνακα. Συνεπώς πρέπει να κατασκευαστεί μία συνάρτηση που θα αναγκάζει τον σερβοκινητήρα του καθαριστήρα, να περιστρέφεται, όπως έχει οριστεί στο σχεδιασμό του. Η συνάρτηση αυτή είναι:

```
void Clearscreen(){
  Servo4.attach(12,771,1798);
  Servo4.write(0,25);
  delay(5000);
  Servo4.write(130,25);
  delay(5000);
  Servo4.write(0,40);
  delay(5000);
  Servo4.detach(); }
```

Βήμα 5ο: Το πέμπτο και τελευταίο βήμα της συγγραφής του κώδικα της διάταξης, είναι ένωση όλων των συναρτήσεων των παραπάνω βημάτων, σε ένα ενιαίο πρόγραμμα, σύμφωνα με τη δομή των προγραμμάτων για το Arduino. Συνεπώς το πρόγραμμα θα πρέπει να ξεκινά με την εισαγωγή βιβλιοθηκών, τη δήλωση των global μεταβλητών και συναρτήσεων, τη void setup(), τη void loop() και τις υπόλοιπες συναρτήσεις. Όλος ο κώδικας της εργασίας βρίσκεται στο παράρτημα. Επίσης, αξίζει να σημειωθεί, ότι η επιλογή της βιβλιοθήκης **VarSpeedServo** έναντι της Servo, έγινε λόγω του ότι η VarSpeedServo δίνει τη δυνατότητα στο χρήστη της επιλογής της ταχύτητας του σερβοκινητήρα. Το σχήμα 3.37 αποτελεί ένα στιγμιότυπο της γραφής της λέξης "ΧΑΝΙΑ" από το βραχίονα, ενώ το σχήμα 3.38 φαίνεται ο καθαρισμός του πίνακα, από τον καθαριστήρα.



Σχήμα 3.37: Στιγμιότυπο από τη γραφή της λέξης "ΧΑΝΙΑ"

Σχήμα 3.38: Στιγμιότυπο από τον καθαρισμό του πίνακα

4. ΣΥΜΠΕΡΑΣΜΑΤΑ - ΠΡΟΟΠΤΙΚΕΣ-ΚΟΣΤΟΣ

Μετά από αρκετά πειράματα και διορθώσεις, το βασικό εξαγόμενο συμπέρασμα είναι ότι ο στόχος της εργασίας καλύφθηκε πλήρως. Είναι μία σχετικά απλής λειτουργίας χειροποίητη ρομποτική κατασκευή, με χαμηλό κόστος και επαναχρησιμοποίηση υλικών, η οποία διαθέτει έναν βραχίονα, που είναι σε θέση να γράφει λέξεις έως επτά γράμματα και έναν καθαριστήρα, που καθαρίζει τον πίνακα γραφής. Όμως η κατασκευή αυτή, μειονεκτεί έναντι μίας αντίστοιχης κατασκευής του εμπορίου, όσον αφορά στην ποιότητα των γραμμάτων που γράφει. Το μειονέκτημα αυτό οφείλεται στην χαμηλή ακρίβεια και επαναληψιμότητα του βραχίονα. Η χαμηλή ακρίβεια οφείλεται, κυρίως, στους σερβοκινητήρες των αρθρώσεων, οι οποίοι είναι χαμηλού κόστους με χαμηλής ποιότητας encoders, αλλά και στα μηχανικά σφάλματα που έχουν γίνει κατά την κατασκευή. Η έλλειψη ακρίβειας στις κοπές και στις διατρήσεις, οι οποίες έγιναν με εργαλεία χειρός, επηρεάζει άμεσα την ακρίβεια στην κίνηση του βραχίονα, κατά την προσέγγιση των καθορισμένων σημείων. Στο κώδικα του βασικού προγράμματος της εργασίας, ο οποίος βρίσκεται στο παράρτημα, φαίνονται αρκετές διορθώσεις που έχουν γίνει σε σταθερές οι οποίες αφορούν σε μηχανικά μέρη, κατά την πειραματική διαδικασία.

Συνεπώς οι προτάσεις για τη βελτίωση της διάταξης και κυρίως του βραχίονα, βασίζονται στα μέσα κατεργασίας, τα οποία θα πρέπει να είναι CNC, καθώς επίσης και στην επιλογή ειδικών ρομποτικών σερβοκινητήρων, για τις αρθρώσεις.

Το κόστος κατασκευής, εκτός των εργαλείων, παρουσιάζεται αναλυτικά στον πίνακα 4.1.

Υλικό	**Τιμή**
Arduino Uno Starter kit	75€
2x Servo MG99R	22€
2x Servo CYS 8201	70€
Ξύλα και κοπές (μπάλσα και μελαμίνη)	40€
Κοχλίες, περικόχλια, γωνίες	5€
Τροφοδοτικό AC/DC	5€
Κόλλα	5€
Σύνολο	**222€**

Πίνακας 4.1: Πίνακας κόστους υλικών κατασκευής

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Εμίρης Δ.Μ., Κουλουριώτης Δ.Ε., Ρομποτική, (2006), Αθήνα, ΤεΚΔΟΤΙΚΗ

[2] Παναγιώτης Μ. Παπάζογλου, Σπύρος-Πολυχρόνης Λιώνης , Ανάπτυξη εφαρμογών με το Arduino, (2014), Αθήνα, Τζιόλα

[3] Ζωή Δουλγέρη, Ρομποτική κινηματική, δυναμική και έλεγχος αρθρωτών βραχιόνων, (2007), Αθήνα, Κριτική

ΠΑΡΑΡΤΗΜΑ

Δομές και λειτουργίες για συγγραφή προγράμματος (sketch) για το Arduino

Δομές ελέγχου ροής

- if (δομή ελέγχου μίας συνθήκης)
- if else (δομή ελέγχου πολλαπλών συνθηκών)
- for (δομή επαναληπτικού ελέγχου συνθήκης)
- while (δομή επαναληπτικού ελέγχου συνθήκης)
- do while (δομή επαναληπτικού ελέγχου συνθήκης)
- switch case (δομή ελέγχου περιπτώσεων)
- break (εντολή διακοπής μιας επαναληπτικής δομής)
- continue (εντολή παράλειψης της τρέχουσας επανάληψης)
- return (εντολή επιστροφής από μία συνάρτηση)
- goto (εντολή μετάβασης σε κάποιο σημείο του κώδικα)

Αριθμητικοί τελεστές

- = (τελεστής εκχώρησης)
- + (τελεστής πρόσθεσης)
- - (τελεστής αφαίρεσης)
- * (τελεστής πολλαπλασιασμού)
- / (τελεστής διαίρεσης)
- % (τελεστής υπόλοιπου ακεραίας διαίρεσης)

Λογικοί τελεστές

- && (λογική σύζευξη)
- || (λογική διάζευξη)
- ! (λογική άρνηση)

Δυαδικοί τελεστές

- & (δυαδική σύζευξη)
- | (δυαδική διάζευξη)
- ^ (δυαδική αποκλειστική διάζευξη)
- ~ (δυαδική άρνηση)
- << (δυαδική αριστερή ολίσθηση)
- >> (δυαδική δεξιά ολίσθηση)

Τελεστές αύξησης και μείωσης

- ++ (αύξηση κατά μία ακέραιη μονάδα)
- -- (μείωση κατά μία ακέραιη μονάδα)

Σύνθετοι τελεστές

- +=, -=, *=, /=, %= (σύνθετοι αριθμητικοί τελεστές)
- &=, |=, ^=, ~=, <<=, >>= (σύνθετοι δυαδικοί τελεστές)

Τελεστές σύγκρισης

- == (ισότητα)
- != (ανισότητα)
- < (μικρότερο)
- > (μεγαλύτερο)
- <= (μικρότερο ή ίσο)
- >= (μεγαλύτερο ή ίσο)

Τελεστές δεικτών

- * (τελεστής απόκτησης περιεχομένου)
- & (τελεστής απόκτησης διεύθυνσης)

Σταθερές
- HIGH (τιμή υψηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- LOW (τιμή χαμηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- false (λογικό επίπεδο ψεύδους σε μία συνθήκη)
- true (λογικό επίπεδο αλήθειας σε μία συνθήκη)
- INPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως είσοδο)
- OUTPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως έξοδο)
- A0, ..., A5 (συμβολοσταθερές για τις αναλογικές επαφές εισόδου)

Τύποι δεδομένων
- boolean (λογική δυαδική τιμή)
- char (προσημασμένος χαρακτήρας 8 ψηφίων)
- unsigned char (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- byte (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- int (προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- unsigned int (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- word (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- long (προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- unsigned long (μη προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- float, double (αριθμός κινητής υποδιαστολής απλής ακρίβειας)
- string (αντικείμενο αλφαριθμητικού με χρήσιμες μεθόδους)

Συναρτήσεις μετατροπής τύπων
- char(), byte()
- int(), word(), long()
- float(), double()

Συναρτήσεις εισόδου και εξόδου
- pinMode() (ορίζει μια επαφή ως είσοδο ή έξοδο)

Συναρτήσεις ψηφιακής εισόδου και εξόδου
- digitalWrite() (γράφει σε μία ψηφιακή επαφή εξόδου)
- digitalRead() (διαβάζει από μία ψηφιακή επαφή εισόδου)

Συναρτήσεις αναλογικής εισόδου και εξόδου
- analogReference() (ορίζει την τάση αναλογικής αναφοράς)
- analogWrite() (γράφει PWM σήματα σε μία επαφή εξόδου)
- analogRead() (διαβάζει από μία αναλογική επαφή εισόδου)

Προηγμένες συναρτήσεις εισόδου και εξόδου
- tone() (παράγει ένα τετραγωνικό σήμα ορισμένης συχνότητας)
- noTone() (διακόπτει την παραγωγή τετραγωνικών σημάτων)
- shiftOut() (ολισθαίνει τα ψηφία μιας τιμής σε μία επαφή εξόδου)
- pulseIn() (επιστρέφει την διάρκεια σε μs ενός παλμού HIGH ή LOW)

Συναρτήσεις χρόνου
- millis() (διάρκεια εκτέλεσης του προγράμματος σε ms)
- micros() (διάρκεια εκτέλεσης του προγράμματος σε μs)
- delay() (παύση προγράμματος - η διάρκεια δίδεται σε ms)
- delayMicroseconds() (παύση προγράμματος - η διάρκεια δίδεται σε μs)

Μαθηματικές και Τριγωνομετρικές συναρτήσεις
- max() (βρίσκει τον μεγαλύτερο ανάμεσα σε δύο αριθμούς)
- min() (βρίσκει τον μικρότερο ανάμεσα σε δύο αριθμούς)
- abs() (επιστρέφει την απόλυτη τιμή ενός αριθμού)
- constrain() (ελέγχει για υπερχείλιση ή υποχείλιση ορίων)
- map() (πραγματοποιεί γραμμικό μετασχηματισμό ορίων)

- pow() (επιστρέφει το αποτέλεσμα μίας δύναμης)
- sqrt() (επιστρέφει την ρίζα ενός αριθμού)
- sin() (υπολογίζει το ημίτονο ενός αριθμού)
- cos() (υπολογίζει το συνημίτονο ενός αριθμού)
- tan() (υπολογίζει την εφαπτομένη ενός αριθμού)

Συναρτήσεις γεννήτριας τυχαίων αριθμών
- random() (δίδεται ένας νέος αριθμός από την γεννήτρια)
- randomSeed() (θέτει τον σπόρο της γεννήτριας παραγωγής)

Συναρτήσεις επεξεργασίας δυαδικών αριθμών
- lowByte() (επιστρέφει το δεξιότερο byte μίας μεταβλητής)
- highByte() (επιστρέφει το αριστερότερο byte μίας μεταβλητής)
- bitRead() (διαβάζει ένα συγκεκριμένο ψηφίο μίας μεταβλητής)
- bitWrite() (γράφει σε ένα συγκεκριμένο ψηφίο μιας μεταβλητής)
- bitSet() (γράφει την τιμή 1 σε κάποιο ψηφίο μίας μεταβλητής)
- bitClear() (γράφει την τιμή 0 σε κάποιο ψηφίο μιας μεταβλητής)
- bit() (υπολογίζει μία συγκεκριμένη δύναμη με βάση το 2)

Συναρτήσεις χρήσης ρουτινών εξυπηρέτησης διακοπών
- attachInterrupt() (ενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)
- detachInterrupt() (απενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)

Συναρτήσεις ενεργοποίησης και απενεργοποίησης διακοπών
- interrupts() (ενεργοποιεί τα σήματα διακοπής)
- noInterrupts() (απενεργοποιεί τα σήματα διακοπής)

Υποστήριξη σειριακής επικοινωνίας
- Serial (αντικείμενο σειριακής επικοινωνίας με χρήσιμες μεθόδους)

Το πρόγραμμα της εργασίας

```
#include <VarSpeedServo.h>


VarSpeedServo Servo1; //dihwsh tou servo1
VarSpeedServo Servo2;
VarSpeedServo Servo3;
VarSpeedServo Servo4;

char leksi[6]; //leksi



float Servo1_angle=96; // gwnia kinisis servo 1
float Servo2_angle=18; // gwnia kinisis servo 2


const float a = 6.2;
const float b = 12.4;

float xstart=1;
float x;
float y;
float pi=3.14;
```

```
void startpoint(int i)  //arxiko shmeio gia ka8e gramma
{

                if (i==0) {
                xstart=xstart;

                  }
                  else if (i==1) {
                  xstart=xstart+2;

                  }
                  else if (i==2) {
                  xstart=xstart+2;

                  }
                  else if (i==3) {
                  xstart=xstart+2;

                  }
                  else if (i==4) {
                  xstart=xstart+2;

                  }
                  else if (i==5) {
                  xstart=xstart+2;

                  }
            else if (i==6) {
                  xstart=xstart+2;

                  }

}

void safepos() //asfalhs 8esh
{
  Penup();
  delay(1000);
  G00(14,3);
  delay(1000);
}


void setup()
{
  Servo1.attach(9,600,2400);
  Servo2.attach(10,600,2400);
  Servo3.attach(11,771,1798);
  Serial.begin(9600);
  Penup();
  G01(18.4,0,0,0,0,20,20,1);

}

void loop()
{
  safepos();
```

```
  delay(1000);
  Clearscreen();
  delay(1000);
  Serial.println("Parakalw eisagete mia leksi ews 7 grammata:\n"); //zhtaei apo to user na
valei leksi
  while (Serial.available()==0) {
                        }               //perimenei to user

  int byteAvailable = Serial.available();

          for(int i=0; i<7; i++){
        startpoint(i);
        delay(3000);
        leksi[i] = Serial.read();
        switch (leksi[i])
        {
          case 'A':
          Serial.print("Parakalw perimenete na zwgrafisw to A\n");
                        delay(1000);
                        Adraw();
                delay(5000);
                        break;
          case 'B':
          Serial.print("Parakalw perimenete na zwgrafisw to B\n");
                delay(1000);
                        Bdraw();
                delay(5000);
                        break;
          case 'G':
          Serial.print("Parakalw perimenete na zwgrafisw to G\n");
          delay(1000);
                        Gdraw();
                delay(5000);
                        break;

          case 'D':
          Serial.print("Parakalw perimenete na zwgrafisw to D\n");
                delay(1000);
                        Ddraw();
                delay(5000);
                        break;

          case 'E':
          Serial.print("Parakalw perimenete na zwgrafisw to E\n");
                delay(1000);
                        Edraw();
                delay(5000);
                        break;
          case 'Z':
          Serial.print("Parakalw perimenete na zwgrafisw to Z\n");
                delay(1000);
                        Edraw();
                delay(5000);
                        break;
          case 'H':
          Serial.print("Parakalw perimenete na zwgrafisw to H\n");
                delay(1000);
```

```
                Hdraw();
        delay(5000);
                break;
case 'U':
Serial.print("Parakalw perimenete na zwgrafisw to U\n");
        delay(1000);
                Udraw();
        delay(5000);
                break;
case 'I':
Serial.print("Parakalw perimenete na zwgrafisw to I\n");
        delay(1000);
                Idraw();
        delay(5000);
                break;
case 'K':
Serial.print("Parakalw perimenete na zwgrafisw to K\n");
        delay(1000);
                Kdraw();
        delay(5000);
                break;
case 'L':
Serial.print("Parakalw perimenete na zwgrafisw to L\n");
        delay(1000);
                Ldraw();
        delay(5000);
                break;
case 'M':
Serial.print("Parakalw perimenete na zwgrafisw to M\n");
        delay(1000);
                Mdraw();
        delay(5000);
                break;
case 'N':
Serial.print("Parakalw perimenete na zwgrafisw to N\n");
        delay(1000);
                Ndraw();
        delay(5000);
                break;
case 'J':
Serial.print("Parakalw perimenete na zwgrafisw to J\n");
        delay(1000);
                Jdraw();
        delay(5000);
                break;
case 'O':
Serial.print("Parakalw perimenete na zwgrafisw to O\n");
        delay(1000);
                Odraw();
        delay(5000);
                break;
case 'P':
Serial.print("Parakalw perimenete na zwgrafisw to P\n");
        delay(1000);
                Pdraw();
        delay(5000);
                break;
```

```
            case 'R':
            Serial.print("Parakalw perimenete na zwgrafisw to R\n");
                  delay(1000);
                        Rdraw();
                  delay(5000);
                        break;
            case 'S':
            Serial.print("Parakalw perimenete na zwgrafisw to S\n");
                  delay(1000);
                        Sdraw();
                  delay(5000);
                        break;
            case 'T':
            Serial.print("Parakalw perimenete na zwgrafisw to T\n");
                  delay(1000);
                        Tdraw();
                  delay(5000);
                        break;
            case 'Y':
            Serial.print("Parakalw perimenete na zwgrafisw to Y\n");
                  delay(1000);
                        Ydraw();
                  delay(5000);
                        break;
            case 'F':
            Serial.print("Parakalw perimenete na zwgrafisw to F\n");
                  delay(1000);
                        Fdraw();
                  delay(5000);
                        break;
            case 'X':
            Serial.print("Parakalw perimenete na zwgrafisw to X\n");
                  delay(1000);
                        Xdraw();
                  delay(5000);
                        break;
            case 'C':
            Serial.print("Parakalw perimenete na zwgrafisw to C\n");
                  delay(1000);
                        Cdraw();
                  delay(5000);
                        break;
            case 'W':
            Serial.print("Parakalw perimenete na zwgrafisw to W\n");
                  delay(1000);
                        Wdraw();
                  delay(5000);
                        break;
          }
          delay(1000);
        }
  safepos();
  delay(5000);

}

//KA8ARISMOS O8ONHS
```

```
void Clearscreen()
{
  Servo4.attach(12,771,1798);
  Servo4.write(0,25);
  delay(5000);
  Servo4.write(130,25);
  delay(5000);
  Servo4.write(0,40);
  delay(5000);
  Servo4.detach();
}

//ENTOLES PENAS

void Penup()
{
 Servo3.write(130);
 delay(500);
}

void Pendown()
{
  Servo3.write(110);
  delay(500);
}

// ENTOLES KINHSEIS G


void G00(float x_in, float y_in)
{
        Penup();
        delay(3000);
        FixCoordinates(x_in,y_in);
           InverseKin();
           Servo1.write(Servo1_angle,20,true);
        Servo2.write(Servo2_angle,20,true);


 }


void G01(float x_in ,float y_in ,float steps,float e,float u,int speed1,int speed2,float vima)
{
      for (float dx=0; dx<=steps; dx=dx+vima)
           {
           FixCoordinates(x_in +(e*dx),y_in +(u*dx));
           InverseKin();
           Servo1.write(Servo1_angle,speed1,true);
        Servo2.write(Servo2_angle,speed2,false);

        }

 }
```

```
void G02(float CircleCenterX,float CircleCenterY, float Radius_x, float Radius_y, float Cfrom, float Cto) //deksiostrofh kinhsh
 {
     for (float z = Cfrom; z>-Cto; z=z-0.2){

      x = Radius_x * cos(z *(pi/180)) + CircleCenterX;
      y = Radius_y * sin(z *(pi/180)) + CircleCenterY;
      FixCoordinates(x,y);
      InverseKin();
      Servo1.write(Servo1_angle,80,true);
      Servo2.write(Servo2_angle,70,true);


  }
 }


//ROUTINES GRAMMATWN

void Adraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0.166666,1,40,40,0.01);
  delay(3000);
  G01(xstart+1,10,3,0.166666,-1,40,40,0.01);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.75,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.75,8.5,1.5,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Bdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,60,60,0.02);
  delay(3000);
  G02(xstart+0.5,9.25,1,0.75,90,100);
  delay(3000);
  G02(xstart+0.5,7.75,1,0.75,100,100);
  delay(3000);
  Penup();
  delay(2000);
```

```
}
void Gdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  G01(xstart+0.5,10,1.25,1,0,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
}
void Ddraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0.166666,1,40,40,0.01);
  delay(3000);
  G01(xstart+1,10,3,0.166666,-1,40,40,0.01);
  delay(3000);
  G01(xstart+1.5,7,1,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}
void Edraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.5,7,1,-1,0,40,40,0.02);
  delay(3000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  G01(xstart+0.5,10,1,1,0,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  delay(2000);
  G01(xstart+1.5,8.5,1,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}
```

```
void Zdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.5,7,1,-1,0,40,40,0.02);
  delay(3000);
  G01(xstart+0.5,7,3,0.333333,1,40,40,0.01);
  delay(3000);
  G01(xstart+1.5,10,1,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Hdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1.5,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,8.5,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1.5,8.5,1,-1,0,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Udraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.75,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.75,8.5,1.25,-1,0,40,40,0.02);
  delay(3000);
  G02(xstart+1,8.5,0.75,1.5,360,0);
```

```
  delay(3000);
  Penup();
  delay(2000);
}

void Idraw()
{
  Penup();
  delay(2000);
  G01(xstart+1,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Kdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+0.5,8.5,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,8.5,1.5,0.6666666,1,30,30,0.01);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+0.5,8.5,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,8.5,1.5,0.6666666,-1,30,30,0.01);
  delay(3000);
  Penup();
  delay(2000);
}

void Ldraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0.166666,1,40,40,0.01);
```

```
  delay(3000);
  G01(xstart+1,10,3,0.166666,-1,40,40,0.01);
  delay(3000);
  Penup();
  delay(2000);
}

void Mdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0,1,40,40,0.02);
  delay(3000);
  G01(xstart+0.5,10,1.5,0.666666,-1,40,40,0.01);
  delay(3000);
  G01(xstart+0.5,8.5,1.5,0.666666,1,40,40,0.01);
  delay(3000);
  G01(xstart+1.5,10,3,0,-1,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Ndraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+0.5,7,3,0,1,40,40,0.02);
  delay(3000);
  G01(xstart+0.5,10,3,0.33333333,-1,40,40,0.01);
  delay(3000);
  G01(xstart+1.5,7,3,0,1,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Jdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.5,7,1,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,8.5,0,0,0,20,20,0.02);
  delay(3000);
  G01(xstart+1.5,8.5,1,-1,0,30,30,0.02);
```

```
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,10,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  delay(2000);
  G01(xstart+1.5,10,1,-1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Odraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.75,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  delay(2000);
  G02(xstart+1,8.5,0.75,1.5,360,0);
  delay(3000);
  Penup();
  delay(2000);
}

void Pdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  G01(xstart+0.5,10,1,1,0,30,30,0.02);
  delay(3000);
  G01(xstart+1.5,10,3,0,-1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Rdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0,1,30,30,0.02);
  delay(3000);
  G02(xstart+0.5,9.25,1,0.5,90,100);
  delay(3000);
```

```
  Penup();
  delay(2000);
}

void Sdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  G01(xstart+1.5,7,1,-1,0,40,40,0.02);
  delay(3000);
  G01(xstart+0.5,7,1.5,0.666666,1,30,30,0.01);
  delay(3000);
  G01(xstart+1.5,8.5,1.5,-0.666666,1,30,30,0.01);
  delay(3000);
  G01(xstart+0.5,10,1,1,0,40,40,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Tdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,10,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1.5,10,1,-1,0,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
}

void Ydraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,10,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,10,1.5,0.333333,-1,30,30,0.01);
  delay(3000);
  G01(xstart+1,8.5,1.5,0.333333,1,30,30,0.01);
  delay(3000);
```

```
  Penup();
  delay(2000);
  G01(xstart+1,8.5,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1,8.5,1.5,0,-1,20,20,0.01);
  delay(2000);
  Penup();
  delay(2000);
}

void Fdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1,7,3,0,1,30,30,0.02);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,8.5,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G02(xstart+1,8.5,0.5,1,360,0);
  delay(3000);
  Penup();
  delay(2000);
}

void Xdraw()
{
  Penup();
  delay(2000);
  G01(xstart+0.5,10,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,10,3,0.333333333,-1,30,30,0.01);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+0.5,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+0.5,7,3,0.333333333,1,30,30,0.01);
  delay(3000);
  Penup();
  delay(2000);
}

void Cdraw()
```

```
{
  Penup();
  delay(2000);
  G01(xstart,10,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart,10,1.5,0.666666,-1,30,30,0.01);
  delay(3000);
  G01(xstart+1,8.5,1.5,0.666666,1,30,30,0.01);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1,7,0,0,0,20,20,0.01);
  delay(2000);
  Pendown();
  delay(1000);
  G01(xstart+1,7,3,0,1,40,40,0.02);
  delay(2000);
  Penup();
  delay(2000);
}

void Wdraw()
{
  Penup();
  delay(2000);
  G01(xstart+1.75,8.5,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  delay(2000);
  G02(xstart+1,8.5,0.75,1.5,360,0);
  delay(3000);
  Penup();
  delay(2000);
  G01(xstart+1.5,7,0,0,0,20,20,0.02);
  delay(2000);
  Pendown();
  delay(2000);
  G01(xstart+1.5,7,1,-1,0,20,20,0.02);
  delay(2000);
  Penup();
  delay(2000);
}



//ROUTINES SYNTETAGMENWN

void FixCoordinates(float x_input, float y_input)
{
 x = x_input;
 y = y_input;
}
```

```
void InverseKin() //antistrofo kinimatiko
{
  float c;
  float s;
  float v;
  float w;
  float th_1;
  float th_2;
  c=(sqrt(sq(x)+sq(y)));
  s= atan2(y,x);
  v= acos((sq(x)+sq(y)+sq(a)-sq(b))/(2*a*c));
  w= acos((sq(x)+sq(y)-sq(a)-sq(b))/(2*a*b));

  th_2=(180/pi)* w;
  th_1=(180/pi)*(s-v);

  Servo1_angle = (th_1+96);
  Servo2_angle = (th_2+18);

}
```