

Ηλεκτρονικών Μηχανικών

Πτυχιακή Εργασία

Σχεδιασμός και Υλοποίηση Ρομποτικής Αράχνης

Design and Implementation of a Robotic Spider

Συγγραφέας:
Αλέξιος Ζαβός

Εξεταστική Επιτροπή:

1. Αναπληρωτής Καθηγητής …….
2. Καθηγητής ……..
3. Καθηγητής ……

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Αντωνιδάκη για την πολύτιμη συνεισφορά του καθώς και για το ενδιαφέρον του κατά την διάρκεια της εκπόνησης της πτυχιακής μου εργασίας, προκειμένου να την ολοκληρώσω.

Επιπλέον, θα ήθελα πω ένα μεγάλο ευχαριστώ στην οικογένειά μου για την αμέριστη συμπαράσταση που μου πρόσφεραν όλα αυτά τα χρόνια της φοίτησής μου στο συγκεκριμένο προπτυχιακό πρόγραμμα.

Αφιερώσεις

Αφιερώνω την εργασία μου

στην οικογένειά μου

Περίληψη

Πλέον, οι φυσικές και ανθρωπογενείς καταστροφές απειλούν κυρίως την ασφάλεια των πολιτών. Για αυτό παρατηρείται αυξημένη ζήτηση για ρομπότ διάσωσης αντί για ανθρώπινους εργάτες διάσωσης. Επίσης, η δυνατότητα των ρομπότ διάσωσης να λειτουργούν αυτόνομα τους επιτρέπει να είναι απαραίτητα τα συγκεκριμένα μηχανήματα, μιας και χρησιμοποιούνται στη θέση του ανθρώπου, προκειμένου να εκτελέσουν επικίνδυνες αποστολές διάσωσης και έρευνας. Ουσιαστικά, τα ρομπότ όπως έχει αποδειχθεί έχουν ταχύτερη και μεγαλύτερη διάρκεια ζωής καθώς και υψηλότερη ικανότητα εργασίας. Υπάρχουν διάφορων ειδών ρομπότ, όπου το καθένα έχει τα δικά του μοναδικά χαρακτηριστικά και ως σύνολο, τα ρομπότ διαφέρουν πολύ σε μέγεθος, σχήμα και δυνατότητες.

Σκοπός της παρούσας εργασίας είναι η μελέτη κατασκευής ενός προσαρμοστικού ευφυούς ρομπότ αράχνης. Με την ενσωμάτωση αισθητήρων, το ρομπότ αράχνη μπορεί να παρακολουθεί ασύρματα το περιβάλλον. Η συγκεκριμένη μελέτη προτείνει ένα αυτοπροσαρμόσιμο έξυπνο ρομπότ αράχνης που λειτουργεί χωρίς ανθρώπινη διασύνδεση, το οποίο έχει δυνατότητες κίνησης εκτός του βασικού περπατήματος και δυνατότητα πλοήγησης σε δύσκολα εδάφη και αναρρίχηση σε ανώμαλες επιφάνειες.

Λέξεις – Κλειδιά : ρομπότ αράχνη, αισθητήρας, ρομποτικό σύστημα, ρομπότ διάσωσης, σχεδιασμός ρομποτικής αράχνης, ανάλυση ρομποτικού συστήματος

Abstract

Now, natural and man-made disasters mainly threaten the safety of citizens. This is why there is an increased demand for rescue robots instead of human rescue workers. Also, the ability of rescue robots to operate autonomously allows them to be essential machines, as they are used in place of humans, in order to perform dangerous rescue and search missions. Essentially, robots have been proven to have faster and longer lifespans as well as higher work capacity. There are several kinds of robots, each with their own unique characteristics, and as a whole, robots vary greatly in size, shape, and capabilities.

The purpose of this work is to study the construction of an adaptive intelligent robot spider. By incorporating sensors, the robot spider can wirelessly monitor the environment. This study proposes a self-adaptive intelligent spider robot that operates without human interaction, which has movement capabilities beyond basic walking and the ability to navigate difficult terrain and climb uneven surfaces.

Keywords : robot spider, sensor, robotic system, rescue robot, robotic spider design, robotic system analysis

Πίνακας Περιεχομένων

Περίληψη 4
Abstract 5
Πίνακας Περιεχομένων 6
Εισαγωγή……9
Κεφάλαιο 1ο : Θεωρητικό Μέρος : Ρομποτική Αράχνη…….10
1.1 Η έννοια του όρου «ρομπότ»……....10
1.2 Η έννοια της ρομποτικής……10
1.3 Ρομποτικό Σύστημα : Η χρήση της Ρομποτικής Αράχνης και τα οφέλη της……10
Κεφάλαιο 2ο : Δομή Ρομποτικής Αράχνης……..12
2.1 Γενική Δομή Ρομποτικής Αράχνης…….12
2.2 Μορφές ρομποτικών αραχνών ……..……19
2.3 Κίνηση ρομποτικών αραχνών……24
2.3.1 Ανάλυση κινηματικής…….25
2.3.2 Ανάλυση κινηματικής προς τα εμπρός……26
2.3.3 Αντίστροφη κινηματική ανάλυση……27
Κεφάλαιο 3ο : Υλοποίηση Ρομποτικής Αράχνης…….28
3.1 Κατασκευή και Σχεδίαση Ρομποτικής Αράχνης……28
3.2 Βηματισμός Ρομποτικής Αράχνης……..…29
Κεφάλαιο 4ο : Ηλεκτρονικά μέρη ρομποτικής αράχνης…….31
4.1 Ηλεκτρονικά Κομμάτια……..…31
4.2 Συνδέσεις……..35
4.3 Ανάλυση Κώδικα……..37
Επίλογος……..…88
Προτάσεις για μελλοντική μελέτη……..…88

Βιβλιογραφικές αναφορές……36

Κατάλογος Εικόνων

Εικόνα 1 : Σκελετός βιολογικής αράχνης…….13

Εικόνα 2: Το πόδι μιας ρομποτικής αράχνης……..…14

Εικόνα 3: Ισχίο ρομποτικής αράχνης……15

Εικόνα 4 : Μηριαίο οστό ρομποτικής αράχνης……..…15

Εικόνα 5 : Κνήμη ρομποτικής αράχνης……..…16

Εικόνα 6 : Τετράποδη αράχνη……19

Εικόνα 7 : Εξάποδη αράχνη……20

Εικόνα 8 : Μοτίβο βάδισης οχτάποδης αράχνης……21

Εικόνα 9 : Robugtix T8X model…….21

Εικόνα 10 : Robot Spderino……24

Εικόνα 11: Κινηματική ανάλυση αραχνοειδών ρομπότ……..25

Εικόνα 12 : Βαση ρομποτικής αράχνης……..…28

Εικόνα 13 : Πόδι ρομποτικής αράχνης…….29

Εικόνα 14 : Απεικόνιση της κίνησης ενός τετράποδου ρομπότ με τη χρήση των μεθόδων wave gait και ripple gait …….29

Εικόνα 15 : SG90 Serno Motor…….31

Εικόνα 16 : Arduino Nano Board……..32

Εικόνα 17 : Nano IO Expansion Shield …….32

Εικόνα 18 : Τροφοδοσία ρομποτικής αράχνης…..…….33

Εικόνα 19 : Μετατροπέας DC-DC Buck LM2596…….34

Εικόνα 20 : Διάγραμμα κυκλώματος ρομποτικής αράχνης…….35

Εικόνα 21 : Σειρά σύνδεσης των Servo …….36

Εισαγωγή

Σήμερα, υπάρχουν πολλές καταστάσεις όπου τα ανθρώπινα όντα δεν μπορούν να ολοκληρώσουν ένα συγκεκριμένο έργο στην πραγματική ζωή, όπως η εύρεση αγνοούμενου στο δάσος για περισσότερες από 24 ώρες και η εξερεύνηση μιας σπηλιάς με έλλειψη οξυγόνου. Για να επιτύχει αυτά τα δύσκολα καθήκοντα, το ανθρώπινο ον θα πρέπει να βασιστεί σε κινητά ρομπότ [5-9]. Ως εκ τούτου, η χρήση ρομπότ στις μέρες μας είναι πολύ συνηθισμένη και οι άνθρωποι τα χρησιμοποιούν κυρίως σε ζώνες καταστροφών, όπως μετά από σεισμούς, ατυχήματα, χιονοστιβάδες ή εκρήξεις. Ουσιαστικά, τα ρομπότ μπορούν να πάρουν τη θέση των ανθρώπινων συμπαικτών τους, περιορίζοντας τον κίνδυνο για την ανθρώπινη ζωή και συμβάλλοντας στην ενίσχυση των πιθανοτήτων διάσωσης των θυμάτων. Ειδικότερα, στις μέρες μας υπάρχουν πολλές και διάφορες εφαρμογές ρομποτικών συστημάτων, τα οποία συγκροτούνται από δύο συστήματα, τα οποία είναι :

- Ηλεκτρονικό (επαναπρογραμματιζόμενη μνήμη)
- Μηχανικό (σύστημα κίνησης)

Σκοπός της παρούσας πτυχιακής εργασίας είναι η κατασκευή μιας ρομποτικής αράχνης. Συγκεκριμένα, η χρήση μιας ρομποτικής αράχνης είναι κυρίως η εξερεύνηση ανώμαλων περιοχών που καθίσταται δύσκολες από τον άνθρωπο, εφόσον έχει την ικανότητα να περπατά αλλά και να περιστρέφεται ακόμη και σε μη ομαλό έδαφος ή επιφάνειες. Πιο αναλυτικά, η διάρθρωση της παρούσας εργασίας ακολουθεί την παρακάτω οργανωτική δομή:

- Στο ***πρώτο κεφάλαιο*** γίνεται εννοιολογική προσέγγιση του περιεχομένου «ρομπότ» και ειδικότερα στην κατασκευή ενός ρομποτικού συστήματος.
- Στο ***δεύτερο κεφάλαιο*** γίνεται μια αναφορά στη δομή μιας ρομποτικής αράχνης.
- Στο ***τρίτο κεφάλαιο*** αναλύεται το μηχανικό μέρος της ρομποτικής αράχνης.
- Στο ***τέταρτο κεφάλαιο*** αναλύεται το ηλεκτρονικό μέρος της ρομποτικής αράχνης
- Τέλος στο ***πέμπτο κεφάλαιο*** της παρούσας εργασίας έχοντας ολοκληρωθεί η πτυχιακή εργασία, αποτυπώνονται τα βασικά συμπεράσματα της εν λόγω μελέτης, καθώς και οι μελλοντικές προτάσεις της παρούσας μελέτης.

Κεφάλαιο 1ο : Θεωρητικό Μέρος : Ρομποτική Αράχνη

1.1 Η έννοια του όρου «ρομπότ»

Μια πρώτη ερμηνεία σχετικά με την έννοια «ρομπότ» δόθηκε από το Ινστιτούτο Ρομπότ των ΗΠΑ αναφέροντας πως πρόκειται για μία επαναπρογραμματιζόμενη πολύ-λειτουργική χειριστική διάταξη, κατάλληλη σχεδιασμένη για τη μετακίνηση διάφορων αντικειμένων όπως εργαλεία, εξαρτήματα, διάφορα αντικείμενα, όπου εκτελεί σειρά προγραμματισμένων κινήσεων μέσα στο χώρο (Δελή, 2011).

Από την άλλη μεριά, σύμφωνα με την Wikipedia, ορίζει την έννοια «ρομπότ» ως μια μηχανική συσκευή (μηχανή που έχει ανθρωπόμορφη συμπεριφορά) που διενεργεί διάφορες εργασίες, οι οποίες είτε διενεργούνται υπό την εποπτεία ενός ανθρώπου είτε δρα μέσω ενός προγραμματισμένου υπολογιστή. Με λίγα λόγια, η χρήση των ρομπότ είναι σημαντική, μιας και πραγματοποιούν εργασίες που θα ήταν ιδιαίτερα δύσκολες να γίνουν από το άνθρωπο (εργασίες ταχύτερα ή φθηνότερα απ' ότι αν εκτελούσε ο άνθρωπος). Ακόμη, αξίζει να επισημανθεί πως μια ρομποτική οντότητα είναι κατασκευασμένη με τέτοιο τρόπο, ώστε να λαμβάνει τις απαραίτητες πληροφορίες, προκειμένου να επεμβαίνει με το πιο αποτελεσματικό τρόπο στο εξωτερικό περιβάλλον.

1.2 Η έννοια της Ρομποτικής

Μία από τις ενδιαφέρουσες τεχνολογίες του 21ου αιώνα είναι η ρομποτική, η οποία συνιστά έναν κλάδο της επιστήμης που ασχολείται με τις μηχανές, οι οποίες έχουν ως σκοπό την αντικατάσταση του ανθρώπου για τη διενέργεια ενός έργου, συνδυάζοντας τη διαδικασία λήψης αποφάσεων με τη φυσική δραστηριότητα (Μυλωνά, 2014).

1.3 Ρομποτικό Σύστημα : Η χρήση της Ρομποτικής Αράχνης και τα οφέλη της

Αρχικά, η ρομποτική αράχνη κατασκευάστηκε από τον Matt Denton και την εταιρεία του Micromagic Systems, ο οποίος την ανέπτυξε ως μέσο βελτίωσης, εφόσον εκτελεί λειτουργίες που δεν μπορούν άλλα ρομποτικά συστήματα να κάνουν όπως είναι η ανίχνευση, η αναρρίχηση και το άλμα. Συγκεκριμένα, η ρομποτική αράχνη είναι μία συσκευή που χρησιμοποιείται, με απώτερο σκοπό κυρίως την ασύρματη παρακολούθηση του περιβάλλοντος. Η ρομποτική αράχνη λειτουργεί χωρίς διασύνδεση και μάλιστα μπορεί να προσαρμοστεί εύκολα ακόμη και σε ανώμαλο

έδαφος λόγω των ποδιών κίνησης που διαθέτει (η μηχανική μιας ρομποτικής αράχνης στη κίνηση οφείλεται στην ανατομία των αρθρώσεων και των ποδιών της). Ένα ρομποτικό σύστημα αράχνης συμβάλλει στην παρακολούθηση επικίνδυνων, δυσπρόσιτων ή μη ομαλών περιοχών, στην αναρρίχηση ανώμαλων επιφανειών καθώς και στον εντοπισμό αντικειμένων, όπου η ανθρώπινη ύπαρξη αδυνατεί να έχει πρόσβαση (δυσκολίες και εμπόδια ως προς την παρακολούθηση του περιβάλλοντος). Όλα τα παραπάνω επιτυγχάνονται λόγω της κίνησης που διαθέτει η αράχνη και που ο ανθρώπινος οργανισμός λόγω των αρθρώσεων του αδυνατεί να έχει πρόσβαση στις εν λόγω περιοχές ή επιφάνειες. Ως εκ τούτου, με βάση τα ανωτέρω οι λόγοι κατασκευής μιας ρομποτικής αράχνης έχουν να κάνουν με τα κάτωθι :

- Είναι κατάλληλη λόγω των κινήσεων της να εξερευνηθούν ακανόνιστες περιοχές ή επιφάνειες που είναι άγνωστες, δηλαδή έχουν τη δυνατότητα να εξερευνούν επικίνδυνες και/ή ανώμαλες περιοχές καθώς και εμπόλεμες ζώνες για τον άνθρωπο.
- Είναι κατάλληλη για την επιθεώρηση ασταθών κτιρίων μετά από φυσικές καταστροφές όπως σεισμό καθώς και για την εξουδετέρωση βομβών όπως νάρκες ξηράς.
- Ο μηχανισμός της ρομποτικής αράχνης είναι πιο αποτελεσματικός σε σχέση με άλλη είδη ρομποτικών συστημάτων λόγω των ποδιών και των αρθρώσεων της αράχνης.

Με βάση τα ανωτέρω γίνεται ευκρινές, πως η χρήση μιας ρομποτικής αράχνης παρόλο που βρίσκεται ακόμα σε εξέλιξη (έργο που εξακολουθεί να αναπτύσσεται) είναι πολύ χρήσιμη για την εξερεύνηση διάφορων μερών του περιβάλλοντος ή περιβαλλοντικών συνθηκών, προκειμένου να προσδιοριστεί η κατάσταση του περιβάλλοντος.

Κεφάλαιο 2ο : Δομή Ρομποτικής Αράχνης

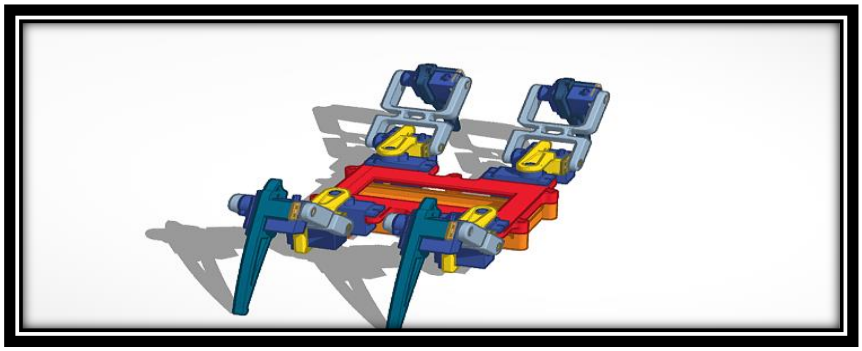
Σε αυτό το κεφάλαιο γίνεται αναφορά στην γενική δομή μιας ρομποτικής αράχνης. Ειδικότερα, θα δοθεί μια σύντομη περιγραφή άλλων ποικιλιών ρομποτικών αραχνών, όπως είναι οι εξάποδες, οι τετράποδες και οι οχτάποδες αράχνες μαζί με μια εξήγηση του τρόπου με τον οποίο κινούνται. Επιπροσθέτως, σε αυτή την υποενότητα θα γίνει αναφορά και στις διάφορες τεχνικές μετακίνησης που χρησιμοποιούν τα εν λόγω ρομπότ καθώς και στον τρόπο με τον οποίο πλοηγούνται σε διαφορετικά περιβάλλοντα, προκειμένου να εκτελέσουν απαιτητικές εργασίες.

2.1 Γενική Δομή Ρομποτικής Αράχνης

Αρχικά, τα τρία κύρια υποσυστήματα που συνθέτουν το ρομποτικό σύστημα αράχνης είναι η δομή του σώματος, οι αισθητήρες και το σύστημα ελέγχου. Η κατασκευή του σώματος μπορεί να κατασκευασμένη από διάφορα υλικά, κυρίως όμως χρησιμοποιείται από αλουμίνιο λόγω της εξαιρετικής αντοχής του[6]. Με τους τρισδιάστατους εκτυπωτές να γίνονται ευρύτερα διαθέσιμοι, η πρακτική της κατασκευής του σκελετού από πλαστικό PLA έχει κερδίσει δημοτικότητα τα τελευταία χρόνια. Εν συνεχεία, τα πόδια της δομής του σώματος έχουν δύο ή τρεις αρθρώσεις που ενώνονται μεταξύ τους με servos, ενώ το σύστημα ελέγχου και οι αισθητήρες στεγάζονται στο κεντρικό σώμα. Το σύστημα ελέγχου που χρησιμοποιείται γενικώς είναι η Arduino nano ή raspberry Pi καθώς είναι ευρέως διαθέσιμα. Παρακάτω θα γίνει ανάλυση των τριών κυρίων υποσυστημάτων, τα οποία είναι τα εξής:

2.1.2 Σκελετός

Ο σκελετός της ρομποτικής αράχνης μιμείται σε ένα μεγάλο βαθμό το σκελετό της βιολογικής αράχνης και γενικά αποτελείται από δυο κομμάτια την βάση και τα ποδιά.



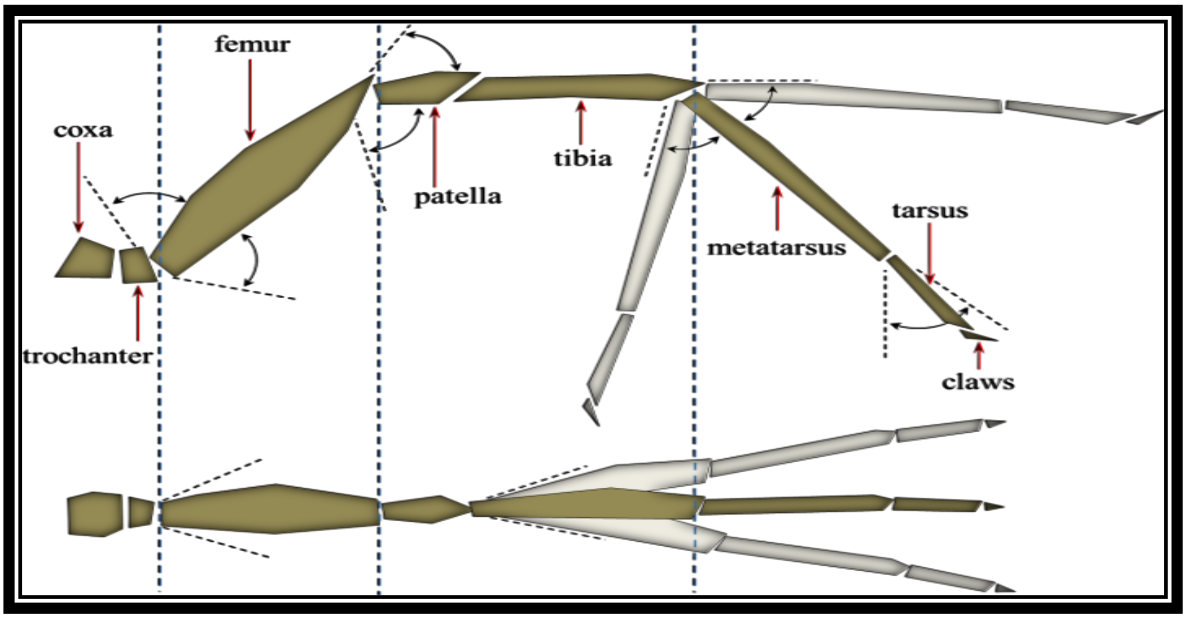
Εικόνα 1 : Σκελετός βιολογικής αράχνης, Πηγή : https://tuhfatalomran.com.efc-eg.com/?k=quadruped-spider-robot-arduino-4-vv-Zy6yjeCa/

2.1.3 Βάση ρομποτικής αράχνης

Η βάση ή το κεντρικό σώμα, κατασκευάζεται συνήθως από ανθεκτικά αλλά και ελαφριά υλικά όπως αλουμίνιο ή πλαστικό υψηλής αντοχής (για παράδειγμα PLA), παρέχοντας μια στιβαρή βάση για το ρομπότ. Το κεντρικό σασί στεγάζει τα κύρια συστήματα ελέγχου, συμπεριλαμβανομένου του μικροελεγκτή (όπως μια Arduino Nano ή Rasberry Pi), της μονάδας τροφοδοσίας και των διαφόρων αισθητήρων. Η βάση είναι σχεδιασμένη με πολλαπλά σημεία στερέωσης για την ασφαλή σύνδεση των αρθρωτών ποδιών, καθώς επιπλέον, η βάση ενσωματώνει βάσεις για μπαταρίες. Ο σχεδιασμός περιλαμβάνει επίσης διατάξεις για άλλες μονάδες, όπως Bluetooth card για τηλεχειρισμό. Ως εκ τούτου, η βάση μιας ρομποτικής αράχνης σχεδιάζεται σχολαστικά για να εξισορροπεί το βάρος, να παρέχει δομική ακεραιότητα καθώς και για να διευκολύνει την ενσωμάτωση μηχανικών και ηλεκτρονικών εξαρτημάτων, εξασφαλίζοντας με αυτό τον τρόπο τη βέλτιστη απόδοση του ρομπότ σε διάφορες εφαρμογές.

2.1.4 Ποδιά της ρομποτικής αράχνης

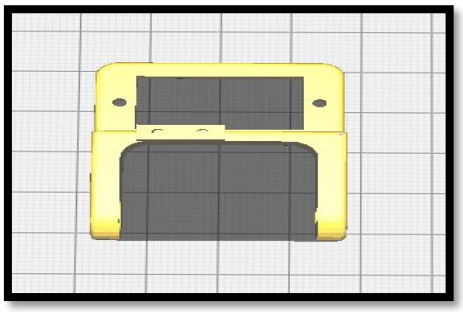
Τα πόδια μιας ρομποτικής αράχνης είναι κατασκευασμένα για να μιμούνται εκείνα μιας πραγματικής αράχνης. Το κάθε πόδι μιας ρομποτικής αράχνης περιέχει τρία τμήματα (ισχίο, μηριαίο οστό και κνήμη) και τρεις αρθρώσεις (βασική άρθρωση, άρθρωση ισχίου και άρθρωση γόνατος)[1], παρέχοντας μεγάλη ευελιξία και εύρος κίνησης. Παρακάτω απεικονίζεται συνοπτικά το κάθε τμήμα ξεχωριστά.



Εικόνα 2 : Το πόδι μιας ρομποτικής αράχνης – Πηγή : [4]

2.1.4.1 Ισχίο

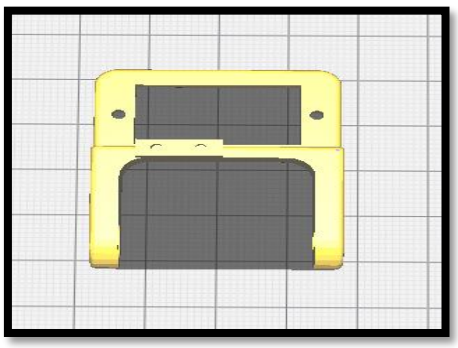
Το ισχίο μιας ρομποτικής αράχνης περιλαμβάνει μια βασική άρθρωση και ένα οριζόντιο στροφικό ή ευθύ ζεύγος που τη συνδέει με το σώμα (εικόνα 3). Στο άλλο άκρο περιλαμβάνει ένα κατακόρυφο περιστροφικό ζεύγος, γνωστό ως άρθρωση ισχίου, όπου συνδέει το ισχίο με το μηριαίο οστό [1].



Εικόνα 3 : Ισχίο ρομποτικής αράχνης.

2.1.4.2 Μηριαίο οστό

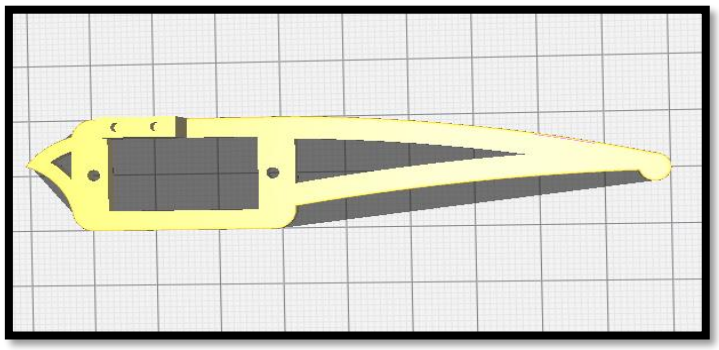
Το μηριαίο οστό βρίσκεται μεταξύ του ισχίου και της κνήμης και χρησιμεύει ως σημείο σύνδεσης και περιστροφής. Το ένα άκρο του μηριαίου οστού συνδέεται με το ισχίο, σχηματίζοντας ένα ευθύ περιστροφικό ζεύγος γνωστό ως άρθρωση ισχίου (εικόνα 4). Το άλλο άκρο συνδέεται με την κνήμη, δημιουργώντας ένα άλλο κατακόρυφο ή ευθύ περιστρεφόμενο ζεύγος, που αναφέρεται ως άρθρωση του γόνατος [1].



Εικόνα 4: Μηριαίο οστό ρομποτικής αράχνης.

2.1.4.3 Κνήμη

Η κνήμη είναι η τελευταία άρθρωση του σχεδιασμού του ποδιού. Το άκρο της κνήμης συνδέεται με το μηριαίο οστό για να σχηματίσει το περιστροφικό ζεύγος κατά την κατακόρυφη διεύθυνση, που ονομάζεται άρθρωση του γόνατος. Το άλλο άκρο της κνήμης έρχεται σε επαφή με το έδαφος, σχηματίζοντας ένα ζεύγος τριβής για να υποβοηθά την κίνηση, τη στροφή και άλλες κινήσεις του βιονικού ρομπότ αράχνης. Το τρισδιάστατο μοντέλο μιας κνήμης φαίνεται στην παρακάτω εικόνα[1].



Εικόνα 5 : Κνήμη ρομποτικής αράχνης.

2.1.5 Αισθητήρες

Οι αισθητήρες που χρησιμοποιούνται για κάθε αράχνη εξαρτώνται ανάλογα με την εργασία που θα εκτελέσει, παρόλα αυτά μπορεί να διαθέτει και όλους. Παρακάτω, γίνεται αναφορά εν συντομία για τον κάθε αισθητήρα ξεχωριστά και ειδικότερα για την χρήση τους στην ρομποτική αράχνη.

2.1.5.1 Αισθητήρες μέτρησης αποστάσεων και αποφυγής

Η ρομποτική αράχνη χρησιμοποιεί αισθητήρες υπέρυθρων, υπερήχων και απόστασης για να βελτιώσει τη λειτουργικότητα και την αυτονομία, ενώ παράλληλα παρέχει πλήρη επίγνωση του περιβάλλοντος και ακριβή αναγνώριση εμποδίων. Το μπροστινό μέρος και τα πόδια της αράχνης είναι εξοπλισμένα με αισθητήρες υπέρυθρων (IR), οι οποίοι εκπέμπουν υπέρυθρο φως και μέσω των αντανακλάσεων μπορούν να ανιχνεύουν αποστάσεις, κινήσεις και εμπόδια. Οι πιο προηγμένοι αισθητήρες IR είναι σε θέση να ανιχνεύουν ακόμη και τη θερμότητα, επιτρέποντάς της με αυτό τον τρόπο να ανταποκρίνεται σε θερμικές ενδείξεις και να λειτουργεί καλύτερα σε συνθήκες

χαμηλού φωτισμού. Οι αισθητήρες υπερήχων, οι οποίοι παράγουν ηχητικά κύματα υψηλής συχνότητας και η ρομποτική αράχνη μπορεί να υπολογίζει αποστάσεις μετρώντας το χρόνο που χρειάζεται για να επιστρέψουν οι ηχώ όταν ανακλώνται σε ένα αντικείμενο, χρησιμοποιούνται επίσης είτε σε συνδυασμό είτε χωριστά με τους αισθητήρες IR. Για να διασφαλιστεί η πλήρης κάλυψη και η αποφυγή συγκρούσεων, οι αισθητήρες αυτοί τοποθετούνται συνήθως στο μπροστινό μέρος για τον εντοπισμό εμποδίων, στα πλάγια για την αποφυγή πλευρικών συγκρούσεων και στο πίσω μέρος για την οπισθοπορεία. Μερικές φορές χρησιμοποιούνται επίσης αισθητήρες προσέγγισης, οι οποίοι χρησιμοποιούν χωρητικές, επαγωγικές είτε υπέρυθρες τεχνικές ανίχνευσης για να ανιχνεύουν την παρουσία αντικειμένων κοντά χωρίς να έρχονται σε άμεση επαφή με αυτά. Επίσης, η αράχνη μπορεί να πλοηγείται σε ποικίλα εδάφη με τη χρήση αισθητήρων εγγύτητας που τοποθετούνται στο μπροστινό μέρος, στο πλάι αλλά και στα πόδια της. Οι μπροστινοί αισθητήρες εντοπίζουν εμπόδια, ενώ οι πλευρικοί αισθητήρες βελτιώνουν την πλευρική επίγνωση και οι αισθητήρες ποδιών παρακολουθούν τις αλλαγές στην επιφάνεια και την εγγύτητα του εδάφους. Όταν συνδυάζονται τα ανωτέρω, επιτρέπει στη ρομποτική αράχνη να συλλέγει και να αξιολογεί περιβαλλοντικά δεδομένα σε πραγματικό χρόνο, καθώς και να λαμβάνει τεκμηριωμένες αποφάσεις σχετικά με το πού να πλοηγηθεί. Λόγω της ολοκληρωμένης μεθόδου αισθητήρων, η ρομποτική αράχνη είναι σε θέση να χειρίζεται πιο περίπλοκες ρυθμίσεις με ευκολία, ασφάλεια και αποτελεσματικότητα.

2.1.5.2 Γυροσκόπιο

Το γυροσκόπιο είναι ένας αισθητήρας που μετρώντας την γωνιακή ταχύτητα επιτρέπει τον έλεγχο της θέσης και της κίνησης της ρομποτικής αράχνης. Το γυροσκόπιο παρακολουθεί τις μεταβολές της γωνιακής ταχύτητας καθώς η αράχνη κινείται, μεταδίδοντας τις πληροφορίες στον ελεγκτή της αράχνης. Η ρομποτική αράχνη χρησιμοποιεί αυτά τα δεδομένα ώστε να μετατοπίσει την στάση και τροχιά της με αποτέλεσμα να έχει καλύτερη ισορροπία και να είναι πιο σταθερό σε διάφορες επιφάνειες.

2.1.5.3 Μικρόφωνο

Τοποθετημένα συνήθως στο κεντρικό σώμα ή στο κεφάλι της ρομποτικής αράχνης, τα μικρόφωνα συλλαμβάνουν ηχητικά κύματα από το περιβάλλον. Αυτά τα ηχητικά κύματα μετατρέπονται στη συνέχεια σε ηλεκτρικά σήματα που μπορούν να επεξεργαστούν από το σύστημα ελέγχου του ρομπότ. Αυτά τα ακουστικά δεδομένα επιτρέπουν στη ρομποτική αράχνη να εκτελεί εργασίες όπως ο εντοπισμός ηχητικών πηγών, ο οποίος είναι απαραίτητος για την πλοήγηση σε συνθήκες χαμηλής ορατότητας, και η αλληλεπίδραση με ανθρώπους ή άλλα ρομπότ. Για παράδειγμα, σε επιχειρήσεις έρευνας και διάσωσης, μια ρομποτική αράχνη εξοπλισμένη με μικρόφωνα μπορεί να ανιχνεύσει και να εντοπίσει κλήσεις για βοήθεια ή άλλους σχετικούς ήχους.

2.1.5.4 Κάμερα

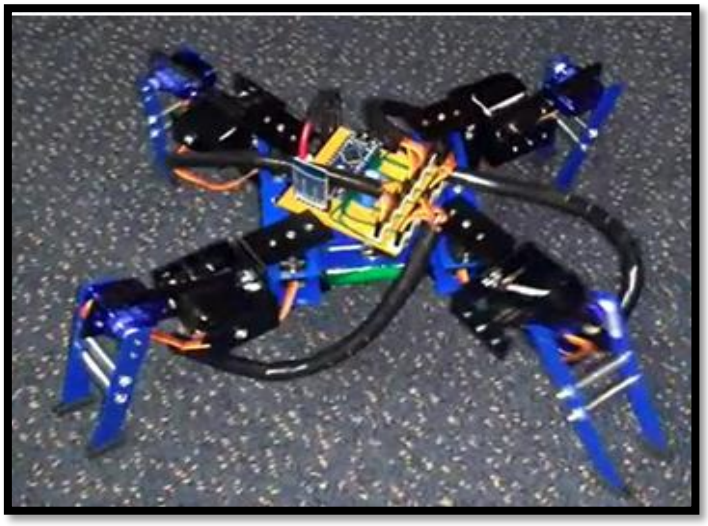
Συνήθως στο κεφάλι της ρομποτικής αράχνης είναι τοποθετημένες οι κάμερες, οι οποίες καταγράφουν εικόνες και βίντεο, τα οποία στη συνέχεια επεξεργάζονται από τον ενσωματωμένο υπολογιστή του ρομπότ. Αυτά τα οπτικά δεδομένα επιτρέπουν στο ρομπότ να εκτελούν εργασίες όπως είναι η ανίχνευση εμποδίων, ο σχεδιασμός διαδρομής και η αναγνώριση αντικειμένων. Επιπροσθέτως, σε εφαρμογές όπως είναι η έρευνα και η διάσωση, μια ρομποτική αράχνη εξοπλισμένη με κάμερα μπορεί να εντοπίσει οπτικά θύματα ή επικίνδυνες περιοχές, ενώ σε βιομηχανικές εγκαταστάσεις μπορεί να επιθεωρήσει εξοπλισμό και να εκτελέσει λεπτομερείς εργασίες παρακολούθησης. Η ενσωμάτωση καμερών επιτρέπει επίσης προηγμένα χαρακτηριστικά όπως είναι η αναγνώριση προσώπου και ο έλεγχος χειρονομιών για την αλληλεπίδραση ανθρώπου-ρομπότ.

2.2 Μορφές Ρομποτικών αραχνών

Υπάρχουν πολλοί διαφορετικοί σχεδιασμοί ρομποτικών αραχνών η κάθε μια χρησιμοποιείται για διαφορετική χρήση. Παρακάτω θα μελετηθούν μερικές από αυτές.

2.2.1 Τετράποδη Αράχνη

Μια τετράποδη ρομποτική αράχνη μιμείται τη βασική ανατομία και τη κίνηση μιας φυσικής αράχνης, αλλά με λιγότερα πόδια για λόγους . Αυτό το ρομπότ αποτελείται συνήθως από ένα κεντρικό σώμα που στεγάζει τους αισθητήρες και το σύστημα ελέγχου, στο οποίο συνδέονται τέσσερα πόδια. Προκειμένου να διατηρηθεί η ισορροπία και η πρόωση, η κίνηση του τετράποδου αποτελείται από συντονισμένα βήματα, όπου τα δύο πόδια στη μία πλευρά του σώματος κινούνται ταυτόχρονα με τα δύο πόδια στην αντίθετη πλευρά.



Εικόνα 6 : Τετράποδη αράχνη, Πηγή: [4]

2.2.2 Εξάποδη αράχνη

Τα εξαποδικά ρομπότ, ή ρομποτικές αράχνες με έξι πόδια, είναι εξαιρετικά ευέλικτα και σταθερά ρομπότ. Συνήθως, κάθε πόδι περιλαμβάνει 3 αρθρώσεις που κινούνται από servos για να επιτρέπουν ένα ευρύ φάσμα κινήσεων. Η κύρια κατασκευή αποτελείται από ένα κεντρικό σώμα με τρία πόδια σε κάθε πλευρά που στεγάζει την παροχή ρεύματος, τους αισθητήρες και τα συστήματα ελέγχου. Τα εξάποδα ρομπότ συχνά περπατούν με τρίποδο βάδισμα, δηλαδή τρία πόδια πατούν σταθερά στο έδαφος και τα υπόλοιπα τρία να κινούνται προς τα εμπρός για να διατηρούν τη σταθερότητα.



Εικόνα 7 : Εξάποδη αράχνη, Πηγή: [5]

2.2.3 Οχτάποδη αράχνη

Μια ρομποτική αράχνη με οκτώ πόδια, έχει σχεδιαστεί για να μιμείται στενά τη σύνθετη δομή και την κίνηση των πραγματικών αραχνών, παρέχοντας εξαιρετική σταθερότητα και ευελιξία. Το κεντρικό σώμα του ρομπότ στεγάζει το κύριο σύστημα ελέγχου, τους αισθητήρες και την παροχή ρεύματος. Στο θέμα της κίνησης η οχτάποδη ρομποτική αράχνη αντιγράφει πλήρως την κίνηση των βιολογικών αραχνών, 4 ποδιά χρησιμοποιούνται για σταθεροποίηση και τα αλλά 4 για κίνηση. Ένα παράδειγμα μοτίβου κίνησης βρίσκεται στην παρακάτω εικόνα οπού στην μια τα L1-L3-R2-R4 πόδια (L= αριστερά , R= δεξιά) χρησιμοποιούνται σαν πόδια σταθεροποίησης και στην άλλη τα L2-L4-R1-R3 ποδιά. Αυτό τα καθιστά ιδιαίτερα κατάλληλα για εργασίες που απαιτούν πλοήγηση σε πολύπλοκα περιβάλλοντα [3].



Fig. 2 Walking of Neoscona nautica

Table 1 Walking step patterns

Leg No	Left				Right			
	L1	L2	L3	L4	R1	R2	R3	R4
Step 1	●	○	●	○	○	●	○	●
Step 2	○	●	○	●	●	○	●	○

(●Support ○Return)

Εικόνα 8 : Μοτίβο βάδισης οχτάποδης αράχνης, Πηγή : [3]



Εικόνα 9 : Robugtix T8X model, Πηγή : [3]

2.2.4 Micro και Nano

Οι μικρό- και νανο ρομποτικές αράχνες, σχεδιασμένες σε απίστευτα μικρές κλίμακες χρησιμοποιούνται για εφαρμογές που απαιτούν εξαιρετική ακρίβεια και ευελιξία. Αυτά τα μικροσκοπικά ρομπότ κατασκευάζονται με τη χρήση προηγμένων υλικών και τεχνικών, ενσωματώνοντας συχνά μικρο-ηλεκτρομηχανικά συστήματα (MEMS) για τις περίπλοκες δομές τους. Αυτές οι αυτόνομες μικροσκοπικές μηχανές επιτρέπουν ελεγχόμενες λειτουργίες σε στενούς χώρους ή δυσπρόσιτα σημεία για την εκτέλεση περίπλοκων εργασιών. Οι εξελίξεις σε αυτό τον τομέα έχουν προσδώσει στα νανορομπότ ανώτερες επιδόσεις σε βιοϊατρικούς τομείς, όπως η βιοανίχνευση, η αποτοξίνωση, η χορήγηση φαρμάκων [7], καθώς και εκτέλεση λεπτών χειρουργικών επεμβάσεων.

2.2.5 Underwater spider

Μια υποβρύχια ρομποτική αράχνη, σχεδιασμένη να λειτουργεί σε υδάτινο περιβάλλον, διαθέτει στιβαρή και αδιάβροχη δομή για να αντέχει στις προκλήσεις της υποβρύχιας εξερεύνησης. Το σώμα της είναι συνήθως κατασκευασμένο από υλικά ανθεκτικά στη διάβρωση, με σφραγισμένες αρθρώσεις και ηλεκτρονικά συστήματα για την αποφυγή εισόδου νερού. Αυτά τα ρομπότ χρησιμοποιούν έναν συνδυασμό κινήσεων περπατήματος και κολύμβησης για να περιηγηθούν στο νερό, παρέχοντας εξαιρετική ευελιξία στον πυθμένα της θάλασσας και σε ανοιχτά νερά. Οι υποβρύχιες ρομποτικές αράχνες χρησιμοποιούνται σε διάφορες εφαρμογές όπως είναι έρευνα της θαλάσσιας βιολογίας, υποβρύχια αρχαιολογία και περιβαλλοντική παρακολούθηση. Επιπροσθέτως, οι υποβρύχιες ρομποτικές αράχνες είναι ιδιαίτερα χρήσιμες για την επιθεώρηση υποβρύχιων δομών, τη χαρτογράφηση του ωκεάνιου πυθμένα καθώς και για τη μελέτη της θαλάσσιας ζωής στα φυσικά τους ενδιαιτήματα.

2.2.6 Soft robotics αράχνη

Μια μαλακή ρομποτική αράχνη, χρησιμοποιεί υλικά όπως σιλικόνη και άλλα ελαστικά υλικά για να μιμηθεί την εύκαμπτη και ελαστική φύση των πραγματικών αραχνών. Τα βιοεμπνευσμένα ρομπότ με εγγενή συμμόρφωση μπορούν να διακριθούν σε δύο κύριους τύπους. Την "αληθινή" μαλακή ρομποτική, που σημαίνει ότι το ρομπότ αποτελείται αποκλειστικά από παραμορφώσιμο υλικό, και τις σκληρές ρομποτικές κατασκευές, οι οποίες ενεργοποιούνται μαλακά ή έχουν μαλακές δυνατότητες [8]. Η δομή της περιλαμβάνει συνήθως ένα κεντρικό σώμα που στεγάζει τα συστήματα ελέγχου και τους αισθητήρες, με μαλακά, αρθρωτά πόδια που κινούνται από πνευματικούς ή υδραυλικούς ενεργοποιητές. Αυτός ο σχεδιασμός επιτρέπει στο ρομπότ να περιηγείται σε πολύπλοκα και περιορισμένα περιβάλλοντα παραμορφώνοντας το σώμα και τα πόδια του ώστε να στριμώχνεται σε στενούς χώρους και να σκαρφαλώνει πάνω από εμπόδια.

2.2.7 Αυτόματες αράχνες

Έξυπνα συστήματα ελέγχου και αυτοματισμού για τις ρομποτικές αράχνες έχουν αναπτυχθεί για να εκτελούν ορισμένες εργασίες που ο άνθρωπος δυσκολεύεται να εκτελέσει [6]. Η κίνηση της ρομποτικής αράχνης είναι ιδιαίτερα προσαρμοστική, χρησιμοποιώντας αλγόριθμους για τον σχεδιασμό διαδρομής και την αποφυγή εμποδίων, επιτρέποντάς της να περιηγείται αυτόνομα σε πολύπλοκα εδάφη και περιβάλλοντα.

2.2.8 Ρομποτικές αράχνες σμήνους

Εμπνευσμένα από τα ζώα της φύσης, όπως τα μυρμήγκια και οι μέλισσες, τα ρομπότ σμήνους προσφέρουν νέες προσεγγίσεις, όπως η αυτοοργάνωση, η αυτομάθηση και η αυτοσυναρμολόγηση [10]. Αυτά τα ρομπότ συνεργάζονται για να επιτύχουν εργασίες που θα ήταν δύσκολο ή αδύνατο να φέρει εις πέρας ένα μεμονωμένο ρομπότ μόνο του. Κάθε ρομπότ αράχνη λειτουργεί αυτόνομα, αλλά ακολουθεί ένα σύνολο απλών κανόνων και επικοινωνεί με άλλα στο σμήνος για να συντονίζει τις ενέργειές του. Αυτό τους δίνει τη δυνατότητα να καλύπτουν αποτελεσματικά μεγάλες εκτάσεις, να πλοηγούνται σε πολύπλοκα περιβάλλοντα και να εκτελούν εργασίες όπως έρευνα και διάσωση, περιβαλλοντική παρακολούθηση και γεωργία ακριβείας. Η κατανεμημένη φύση του σμήνους προσφέρει ευρωστία και ευελιξία, καθώς η αποτυχία ενός ρομπότ

δεν εμποδίζει τη συνολική αποστολή, καθιστώντας τα συστήματα αυτά ιδιαίτερα χρήσιμα σε δυναμικά και απρόβλεπτα περιβάλλοντα.



Εικόνα 10 : Robot spderino, Πηγή: [10]

2.3 Κίνηση ρομποτικών αραχνών

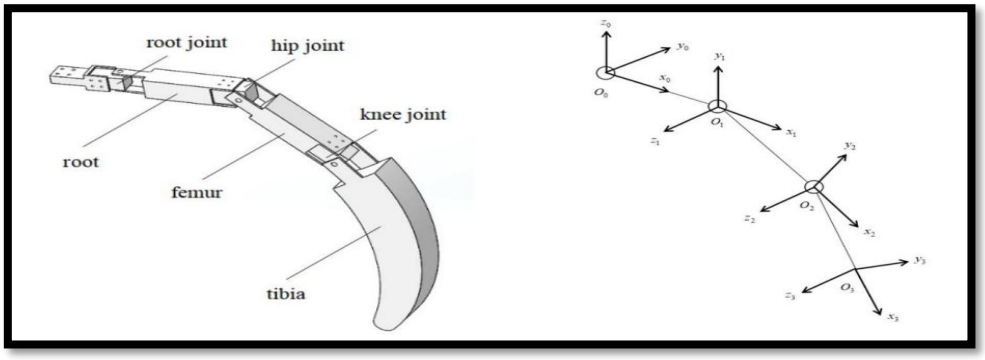
Η κίνηση των ρομποτικών αραχνών, εμπνευσμένη από τις αντίστοιχες βιολογικές αράχνες περιλαμβάνει ένα περίπλοκο συντονισμό πολλαπλών ποδιών για την επίτευξη σταθερής και προσαρμοστικής μετακίνησης. Συνήθως, αυτά τα ρομπότ χρησιμοποιούν μια μορφή μετακίνησης γνωστή ως εναλλασσόμενο τρίποδο βάδισμα, όπου τρία πόδια παραμένουν σε επαφή με το έδαφος, ενώ τα άλλα τρία κινούνται προς τα εμπρός, παρέχοντας έναν ισορροπημένο και αποτελεσματικό τρόπο κίνησης. Αυτή η βάδιση εξασφαλίζει συνεχή σταθερότητα και επιτρέπει στο ρομπότ να περιηγείται αποτελεσματικά σε ανώμαλα εδάφη. Στα τετράποδα ρομπότ, η κίνηση συχνά περιλαμβάνει δύο πόδια σε αντίθετες πλευρές που κινούνται σε αρμονία, ενώ τα εξάποδα ρομπότ χρησιμοποιούν μια πιο κατανεμημένη προσέγγιση για τη διατήρηση της ισορροπίας.

2.3.1. Ανάλυση κινηματικής

Η κινηματική ανάλυση αποτελεί σημαντική βάση για τον σχεδιασμό της βάδισης και την ανάλυση της σταθερότητας των αραχνοειδών ρομπότ. Η κινηματική ανάλυση του αραχνοειδούς ρομπότ έχει ως βασικό στόχο να βρεθεί η σχέση μεταξύ της χωρικής θέσης του άκρου του κάθε ποδιού και των τριών γωνιών των αρθρώσεων. Οι εξισώσεις κινηματικής των ποδιών είναι πανομοιότυπες στη μορφή και στη δομή, ενώ οι παράμετροι είναι διαφορετικές σε διαφορετικές χωρικές θέσεις και καταστάσεις κίνησης. Ως εκ τούτου, μόνο ένα από τα πόδια χρειάζεται να αναλυθεί κινηματικά.

Η κινηματική ανάλυση διακρίνεται σε εμπρόσθια κινηματική ανάλυση και σε αντίστροφη κινηματική ανάλυση. Προκειμένου να διευκολυνθεί η εμπρόσθια και η αντίστροφη κινηματική ανάλυση του μηχανικού ποδιού, θα γίνει ανάλυση σε ένα D-H σύστημα συντεταγμένων όπως φαίνεται στην παρακάτω εικόνα. Συγκεκριμένα : $\$O_{\{0\}} - x_{\{0\}}y_{\{0\}}z_{\{0\}\$}$ είναι στερεωμένο στον κορμό, $O_1 - x_1y_1z_1$ στερεώνεται στη ρίζα, $O_2 - x_2y_2z_2$ στερεώνεται στο μηριαίο οστό και $O_3 - x_3y_3z_3$ στερεώνεται στην κνήμη. Οι αφετηρίες των συστημάτων συντεταγμένων βρίσκονται στην άρθρωση της φτέρνας, του ισχίου, του γόνατος και του άκρου του ποδιού, αντίστοιχα. Ο άξονας X δείχνει την κατεύθυνση έκτασης της ράβδου, ο άξονας Z δείχνει την κατεύθυνση περιστροφής του

άρθρωσης και ο άξονας Y είναι κάθετος στους αντίστοιχους άξονες X και Z, οι οποίοι αποτελούν το δεξιό σύστημα συντεταγμένων. Επιπροσθέτως, οι Θ_1、θ_2、θ_3 είναι γωνίες περιστροφής της ριζικής άρθρωσης, της άρθρωσης ισχίου και γονάτου αντίστοιχα. Οι l_1、l_2、l_3 είναι τα μήκη της ρίζας του μηριαίου οστού και της κνήμης, αντίστοιχα[2].



Εικόνα 11 : Κινηματική ανάλυση αραχνοειδών ρομπότ, Πηγή : [2]

2.3.2 Ανάλυση κινηματικής προς τα εμπρός

Η εμπρόσθια κινηματική ανάλυση αναφέρεται στην επίλυση της θέσης στο χώρο του άκρου του ποδιού του μηχανικού ποδιού και στη λήψη των εξισώσεων της εμπρόσθιας κινηματικής όταν είναι γνωστές οι γωνίες περιστροφής των τριών αρθρώσεων. Παραδείγματος χάριν, τα συστήματα συντεταγμένων D-H της οχτάποδης ρομποτικής αράχνης με 3 βαθμούς ελευθέριας σε κάθε πόδι. Μάλιστα, η ιδιά ανάλυση ισχύει για οποιοδήποτε ρομποτική αράχνη που έχει τους ιδίους βαθμούς ελευθερίας. Παρακάτω απεικονίζονται οι ακόλουθοι πίνακες ομογενών μετασχηματισμών, οι οποίοι είναι οι εξής:

$$R_{01} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & l_1\cos\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 & l_1\sin\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{12} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{23} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & l_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{03} = R_{01} \cdot R_{12} \cdot R_{23}$$

$$R_{03} = \begin{bmatrix} \cos\theta_1\cos(\theta_2+\theta_3) & -\cos\theta_1\sin(\theta_2+\theta_3) & \sin\theta_1 & (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\cos\theta_1 \\ \sin\theta_1\cos(\theta_2+\theta_3) & -\sin\theta_1\sin(\theta_2+\theta_3) & -\cos\theta_1 & (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\sin\theta_1 \\ \sin(\theta_2+\theta_3) & \cos(\theta_2+\theta_3) & 0 & l_2\sin\theta_2 + l_3\sin(\theta_2+\theta_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Οι συντεταγμένες του άκρου του ποδιού στο σύστημα συντεταγμένων $\{O\}_{\{3\}} - x_{\{3\}}y_{\{3\}}z_{\{3\}}$ είναι $(0,0,0)$. Συνεπώς:

$$\begin{bmatrix} \cos\theta_1\cos(\theta_2+\theta_3) & -\cos\theta_1\sin(\theta_2+\theta_3) & \sin\theta_1 & (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\cos\theta_1 \\ \sin\theta_1\cos(\theta_2+\theta_3) & -\sin\theta_1\sin(\theta_2+\theta_3) & -\cos\theta_1 & (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\sin\theta_1 \\ \sin(\theta_2+\theta_3) & \cos(\theta_2+\theta_3) & 0 & l_2\sin\theta_2 + l_3\sin(\theta_2+\theta_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

πηγή:[2]

Οι λύσεις είναι:

$$x_0 = (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\cos\theta_1 \quad (1)$$
$$y_0 = (l_1 + l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3))\sin\theta_1 \quad (2)$$
$$z_0 = l_2\sin\theta_2 + l_3\sin(\theta_2+\theta_3) \quad (3)$$

Οι συντεταγμένες του άκρου του ποδιού είναι $x_{\{0\}}$، $y_{\{0\}}$، $z_{\{0\}}$ στο σύστημα συντεταγμένων $O_{\{0\}} - x_{\{0\}}y_{\{0\}}z_{\{0\}}$

πηγή:[2]

2.3.3 Αντίστροφη κινηματική ανάλυση

Η αντίστροφη κινηματική ανάλυση αναφέρεται στην επίλυση των γωνιών περιστροφής τριών αρθρώσεων καθώς και στην απόκτηση αντίστροφης εξισώσεις κινηματικής όταν είναι γνωστή η χωρική θέση του άκρου του ποδιού του μηχανικού ποδιού. Η ρίζα, το μηριαίο οστό και η κνήμη βρίσκονται στο ίδιο επίπεδο, οπότε:

$$\theta_1 = \arctan\left(\frac{y_0}{x_0}\right) \quad (4)$$
$$x_0\cos\theta_1 - l_1 - l_2\cos\theta_2 = l_3\cos(\theta_2+\theta_3) \quad (5)$$
$$z_0 - l_2\sin\theta_2 = l_3\sin(\theta_2+\theta_3) \quad (6)$$

Προσθέτουμε τον τύπο (1), τον τύπο (2) και τον τύπο (3) στο τετράγωνο και αντικαθιστούμε τον τύπο (4), τον τύπο (5) και τον τύπο (6). Υπάρχει:

$$\left(2x_0l_2\left(\cos\left(\arctan\left(\frac{y_0}{x_0}\right)\right)\right)^{-1} - 2l_1l_2\right)\cos\theta_2 + 2z_0l_2\sin\theta_2 = x_0^2 + y_0^2 + z_0^2 - l_1^2 + l_2^2 - l_3^2 - 2x_0l_1(\cos(\arctan(y_0/x_0)))^{-1} \quad (7)$$

Η λύση είναι:

$$\theta_2 = (x_0^2 + y_0^2 + z_0^2 - l_1^2 + l_2^2 - l_3^2)((2x_0l_2(\cos(\arctan(y_0/x_0)))^{-1} - 2l_1l_2)^2 + (2z_0l_2)^2)^{-\frac{1}{2}} - 2x_0l_1(\cos(\arctan(y_0/x_0)))^{-1}((2x_0l_2(\cos(\arctan(y_0/x_0)))^{-1} - 2l_1l_2)^2 + (2z_0l_2)^2)^{-\frac{1}{2}} - \arctan(x_0l_2z_0\ {}^{-1}l_2^{-1}(\cos(\arctan(y_0/x_0)))^{-1} - l_1l_2z_0\ {}^{-1}l_2^{-1})$$

Ο τύπος (6) παραμόρφωσης, έχει:

$$\theta_3 = \arcsin(z_0l_3^{-1} - l_2l_3^{-1}\sin\theta_2) - \theta_2$$

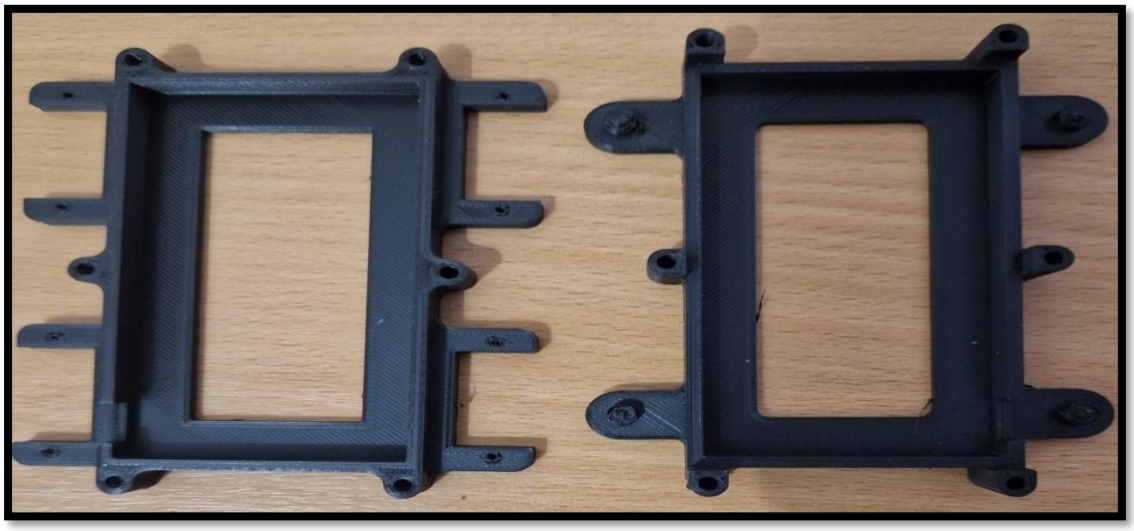
πηγή:[2]

Κεφάλαιο 3ο : Υλοποίηση Ρομποτικής Αράχνης

Στο κεφάλαιο αυτό θα γίνει αναφορά στην τετράποδη ρομποτική αράχνη που κατασκευάστηκε στην παρούσα εργασία και μάλιστα θα παρουσιαστεί η κατασκευή της και τα μηχανικά της μέρη, καθώς και ο τρόπος βηματισμού της. Θα εξεταστεί επίσης και η κατασκευαστική διαδικασία και τα υλικά που χρησιμοποιήθηκαν.

3.1 Κατασκευή και Σχεδίαση Ρομποτικής Αράχνης

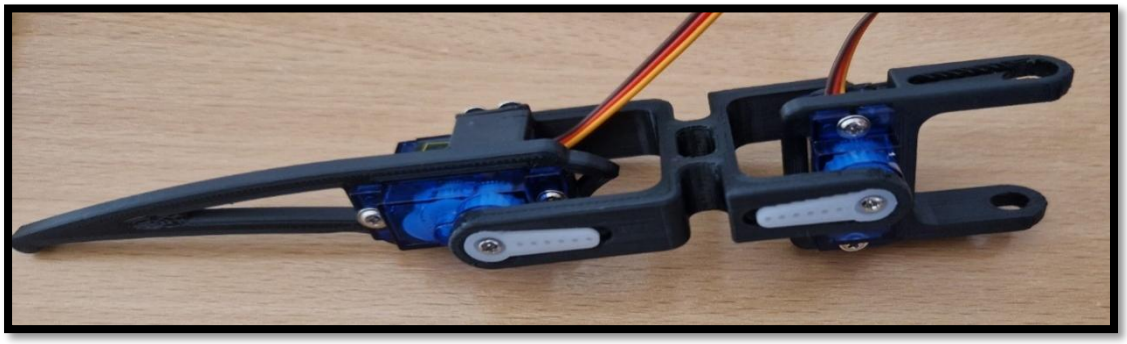
Η κατασκευή της ρομποτικής αράχνης περιλαμβάνει έναν συνδυασμό μηχανικών και ηλεκτρονικών στοιχείων σε αυτό το κεφάλαιο όμως θα αναφερθούμε μόνο στα μηχανικά κομμάτια. Ο σκελετός που χρησιμοποιήθηκε είναι ένας γενικός σκελετός τετράποδης ρομποτικής αράχνης. Για την δημιουργία του σκελετού χρησιμοποιήθηκε πλαστικό PLA (Πολυγαλακτικό οξύ) με την βοήθεια 3D εκτυπωτή, ώστε να εξασφαλιστεί η σταθερότητα και η ελαφρότητα της κατασκευής. Η βάση της ρομποτικής αράχνης αποτελείται από 2 κομμάτια το πάνω και το κάτω μέρος και τα δυο με μήκος 10,5cm και πλάτος 5cm.



Εικόνα 12 : Κατασκευή ρομποτικής αράχνης.

Το κάθε πόδι της ρομποτικής αράχνης αποτελείται από τρία κομμάτια το ισχίο , το μηριαίο οστό και την κνήμη. Το ισχίο (coxa) έχει μήκος 2,5cm και πλάτος 3.4cm, το μηριαίο οστό (femur) είναι σε μήκος 6.5cm και πλάτος 3.6cm ενώ η κνήμη (tibia) έχει μήκος 9,9cm και πλάτος 2cm (εικόνες και των τριων από κάτω). Στην κατασκευή

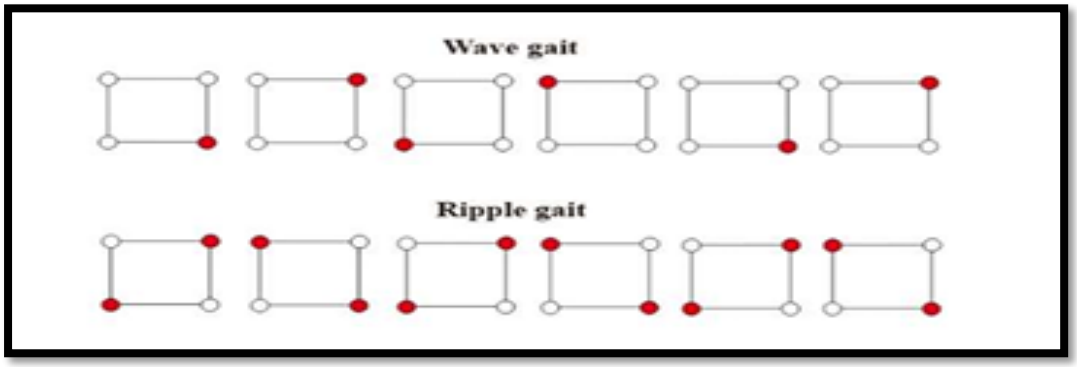
υπάρχουν 12 servos SG90 που βοηθούν την κίνηση της ρομποτικής αράχνης . Για κάθε πόδι υπάρχει ένα servo που ενώνει την βάση με το ισχίο, ένα που ενώνει το ισχίο με το μηριαίο οστό και ένα που ενώνει το μηριαίο οστό με την κνήμη.



Εικόνα 13 : Πόδι ρομποτικής αράχνης.

3.2 Βηματισμός Ρομποτικής Αράχνης

Η κυματιστή βάδιση (Ripple gait) είναι το είδος βηματισμού που χρησιμοποιείται για κίνηση της ρομποτικής μας αράχνης. Είναι μια μέθοδος ελέγχου της κίνησης με την κίνηση δύο ποδιών ταυτόχρονα, τα κινούμενα πόδια βρίσκονται σε διαφορετικές πλευρές. Για παράδειγμα, αν το μπροστινό δεξί πόδι κινείται, κινείται και το πίσω αριστερό πόδι, διαφορετικά όταν το μπροστινό αριστερό πόδι κινείται, κινείται και το πίσω δεξί πόδι. Η χρήση της μεθόδου ripple gait κάνει το ρομπότ να κινείται ταχύτερα από τη μέθοδο, wave gait, αλλά η σταθερότητα του ρομπότ είναι πολύ δύσκολη [9].



Εικόνα 14 : Απεικόνιση της κίνησης ενός τετράποδου ρομπότ με τη χρήση των μεθόδων wave gait και ripple gait**,** Πηγή : [9]

3.3 Υλικά που χρησιμοποιήθηκαν

Τα υλικά που χρησιμοποιήθηκαν είναι τα παρακάτω

1. Τρισδιάστατα εκτυπωμένα μέρη ρομπότ αράχνης (σασί, πόδια, σύνδεσμοι)
2. 12 X SG90 σερβοκινητήρας (180 μοίρες)
3. Πλακέτα Arduino Nano
4. Ασπίδα επέκτασης Nano IO
5. Θήκη μπαταρίας 18650 - 2 κυψέλες
6. Δύο μπαταρίες ιόντων λιθίου 18650 3,7v
7. Διακόπτης
8. LM2596 DC-DC Buck μετατροπέας Step-Down Power Module
9. Αισθητήρας υπερήχων HC-SR04 4Pin
10. Μονάδα Bluetooth HC-05 ή HC-06
11. Θηλυκό προς θηλυκό καλώδιο βραχυκυκλωτήρα
12. 4 X M3 30mm Βίδες και παξιμάδια
13. Πιστόλι κόλλας
14. Σίδερο συγκόλλησης

Κεφάλαιο 4ο: Ηλεκτρονικά μέρη ρομποτικής αράχνης

4.1 Ηλεκτρονικά Κομματια

Σε αυτό το κεφάλαιο θα γίνει αναφορά στα ηλεκτρονικά εξαρτήματα της ρομποτικής αράχνης, δηλαδή εν συντομία θα γίνει αναφορά για το κάθε εξάρτημα, πως συνδέονται όλα μεταξύ τους καθώς και μια ανάλυση του κώδικα που χρησιμοποιήθηκε.

4.1.1 SG90 Servo Motor

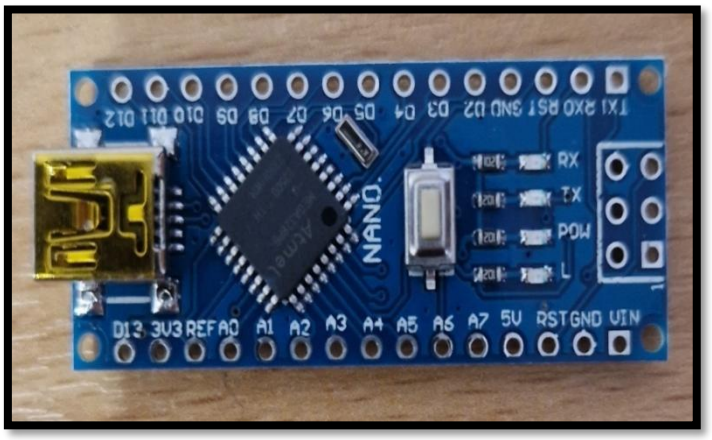
Ο σερβοκινητήρας SG90 είναι ένας συμπαγής και ευρέως χρησιμοποιούμενος μικρο-σερβοκινητήρας, ιδανικός για μικρά έργα ρομποτικής. Λειτουργεί σε ένα εύρος τάσης 4,8 έως 6,0 βολτ, παρέχει περίπου 1,8 kg-cm ροπής και έχει εύρος περιστροφής 180 μοιρών, επιτρέποντας ακριβή έλεγχο. Ο ελαφρύς σχεδιασμός του καθιστά εύκολη την ενσωμάτωσή του με μικροελεγκτές όπως ο Arduino.



Εικόνα 15 : SG90 Servo Motor.

4.1.2 Arduino Nano board

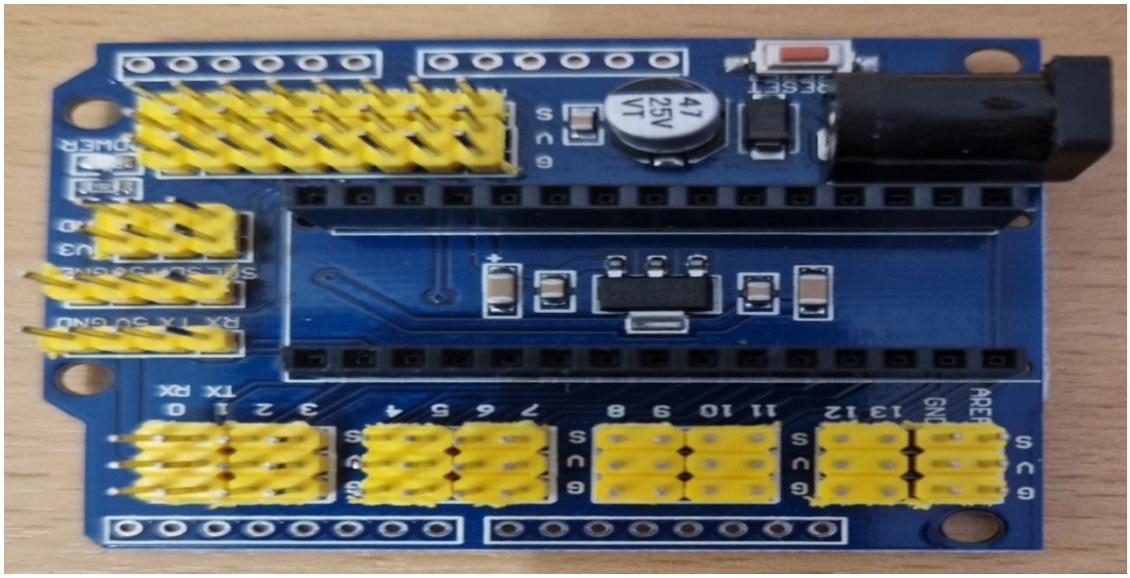
Το Arduino Nano είναι μια συμπαγής πλακέτα μικροελεγκτή βασισμένη στον ATmega328P, σχεδιασμένη για μικρά ηλεκτρονικά έργα. Με διαστάσεις 45x18 mm, ταιριάζει καλά σε στενούς χώρους. Η πλακέτα διαθέτει 14 ψηφιακές ακίδες εισόδου/εξόδου, 8 αναλογικές εισόδους και λειτουργεί στα 16 MHz. Διαθέτει θύρα mini-USB για προγραμματισμό και τροφοδοσία. Συμβατή με το Arduino IDE, είναι φιλική προς το χρήστη και ευέλικτη. Το Arduino Nano είναι προσιτό και υποστηρίζεται ευρέως από μια ισχυρή κοινότητα, καθιστώντας το κατάλληλο τόσο για αρχάριους όσο και για προχωρημένους χρήστες σε διάφορα έργα όπως η ρομποτική και ο οικιακός αυτοματισμός.



Εικόνα 16 : Arduino Nano board.

4.1.3 Nano IO Expansion Shield

Το Arduino Nano I/O Expansion Shield ενισχύει το Arduino Nano παρέχοντας εύκολη πρόσβαση στους ακροδέκτες I/O του. Διαχωρίζει τις ακίδες σε σαφώς επισημασμένες κεφαλίδες, καθιστώντας απλούστερη τη σύνδεση αισθητήρων και ενεργοποιητών. Η ασπίδα περιλαμβάνει βιδωτούς ακροδέκτες για ασφαλείς συνδέσεις, πρόσθετες επιλογές τροφοδοσίας και ένα κουμπί επαναφοράς. Η συμβατότητά της με διάφορες μονάδες και εξαρτήματα βοηθά στη γρήγορη ανάπτυξη και δοκιμή. Αυτή η ασπίδα εξορθολογίζει την καλωδίωση και επεκτείνει τη συνδεσιμότητα, καθιστώντας την πολύτιμο εργαλείο για χομπίστες και προγραμματιστές που εργάζονται σε έργα με το Arduino Nano.



Εικόνα 17: Nano IO Expansion Shield

4.1.4 Τροφοδοσία

Η πηγή ενέργειας που χρησιμοποιήσαμε για τη ρομποτική αράχνη είναι μια θήκη μπαταριών 2 κελιών 18650, η οποία χωράει δύο μπαταρίες ιόντων λιθίου 3,7V 18650. Αυτή η διάταξη παρέχει συνολική τάση 7,4V, κατάλληλη για την τροφοδοσία των κινητήρων και των κυκλωμάτων της αράχνης. Η θήκη μπαταριών εξασφαλίζει ασφαλείς συνδέσεις και εύκολη αντικατάσταση των μπαταριών. Οι μπαταρίες 18650 προσφέρουν υψηλή ενεργειακή πυκνότητα και μεγάλη διάρκεια ζωή. Η επαναφορτιζόμενη φύση τους τις καθιστά επίσης μια οικονομικά αποδοτική και φιλική προς το περιβάλλον επιλογή για την τροφοδοσία της ρομποτικής αράχνης.

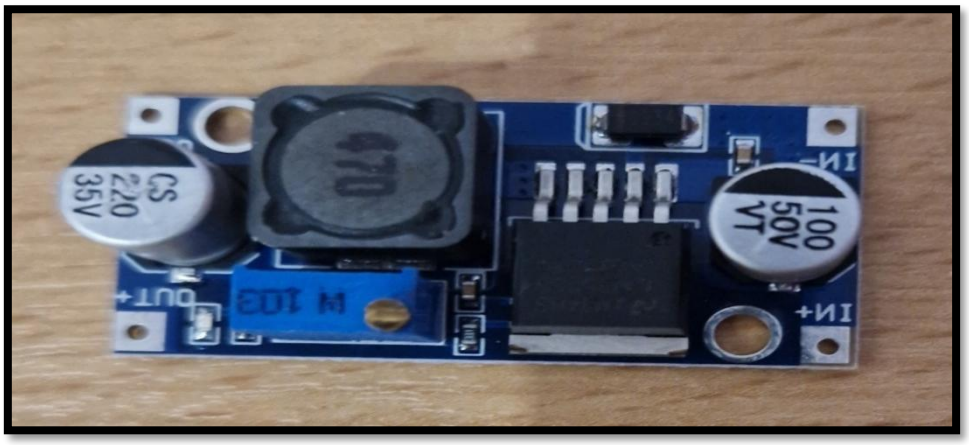


Εικόνα 18 : Τροφοδοσία ρομποτικής αράχνης.

4.1.5 LM2596 DC-DC Buck Converter Step-Down Power Module

Ο μετατροπέας DC-DC Buck LM2596 είναι ένας αξιόπιστος ρυθμιστής τάσης που μειώνει τις υψηλότερες τάσεις εισόδου σε μια χαμηλότερη, σταθερή τάση εξόδου. Δέχεται τάσεις εισόδου από 4,5V έως 40V και μπορεί να ρυθμίσει τάσεις εξόδου από 1,23V έως 37V. Με απόδοση έως και 92%, ελαχιστοποιεί τη θερμότητα και την απώλεια ισχύος. Ο μετατροπέας μπορεί να παρέχει ρεύμα έως 3A, καθιστώντας τον κατάλληλο για την τροφοδοσία διαφόρων ηλεκτρονικών συσκευών. Περιλαμβάνει προστασία θερμικής απενεργοποίησης και ορίου ρεύματος για ασφαλή λειτουργία. Το συμπαγές μέγεθος και η ευελιξία του τον καθιστούν πρακτική επιλογή για έργα που χρειάζονται αποτελεσματική ρύθμιση τάσης. Για τ ην δικιά μας χρήση τον θέτουμε στα

7V ώστε να παρέχει αρκετά Volt στα Servos ώστε να μπορούν να σηκώσουν το σώμα της αράχνης.



Εικόνα 19 : Μετατροπέας DC-DC Buck LM2596

4.1.6 HC-SR04 4Pin Ultrasonic Sensor

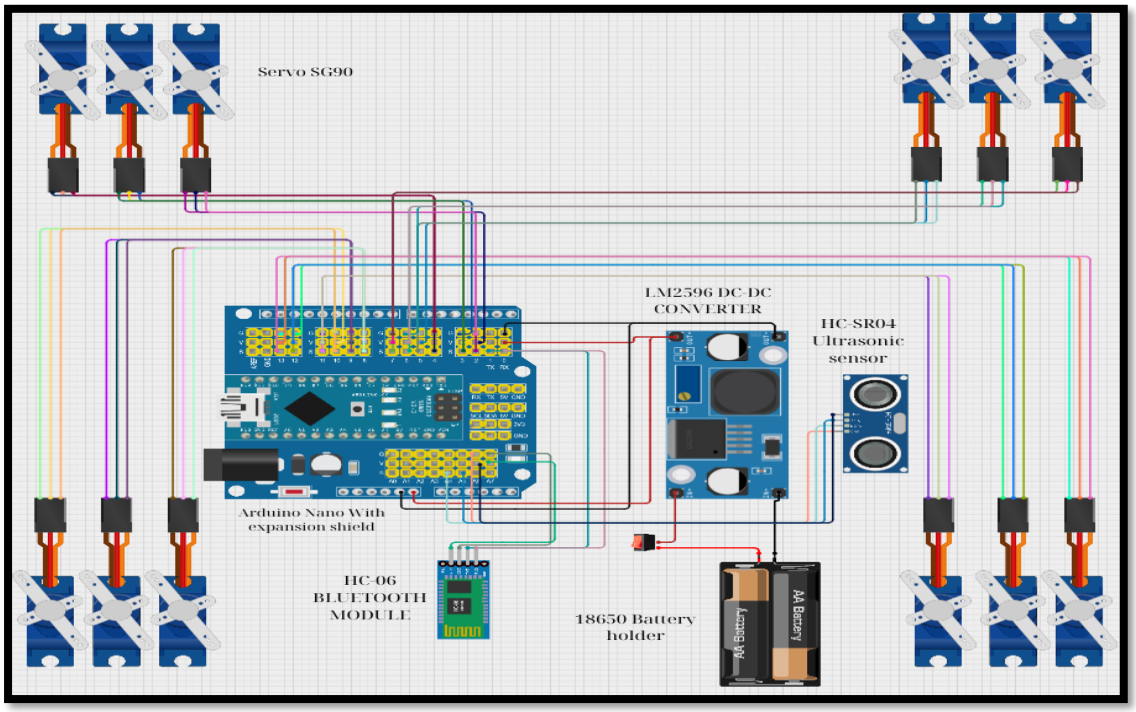
Ο αισθητήρας υπερήχων HC-SR04 4 ακίδων είναι μια μονάδα μέτρησης απόστασης. Λειτουργεί εκπέμποντας έναν υπερηχητικό παλμό και μετρώντας το χρόνο που χρειάζεται η ηχώ για να επιστρέψει, προσδιορίζοντας την απόσταση από ένα αντικείμενο. Λειτουργεί σε τάση 5V και μετρά αποστάσεις από 2 cm έως 400 cm με ακρίβεια περίπου 3 mm. Ο αισθητήρας διαθέτει τέσσερις ακροδέκτες: VCC, Trig, Echo και GND, καθιστώντας εύκολη τη σύνδεσή του με μικροελεγκτές όπως ο Arduino. Ο ακροδέκτης Trig εκκινεί τον παλμό και ο ακροδέκτης Echo εξάγει τον χρόνο επιστροφής.

4.1.7 HC-06 Bluetooth Module

Η μονάδα Bluetooth HC-06 χρησιμοποιείτε για την ασύρματη σύνδεση συσκευών ή σαν τρόπος σύνδεσης με ασύρματο τηλεχειριστήριο. Λειτουργεί στα 3,3V έως 6V και επιτρέπει τη σειριακή επικοινωνία μέσω Bluetooth. Με τέσσερις ακροδέκτες-VCC, GND, TXD (μετάδοση) και RXD (λήψη), είναι εύκολο να συνδεθεί σε μικροελεγκτές όπως το Arduino. Η μονάδα υποστηρίζει Bluetooth 2.0+EDR και έχει εμβέλεια έως και 10 μέτρα. Σχεδιασμένη για λειτουργία slave, μπορεί να συνδεθεί με άλλες συσκευές Bluetooth, αλλά δεν μπορεί να ξεκινήσει συνδέσεις.

4.2 Συνδέσεις

Η παρακάτω εικόνα παρουσιάζει το τρόπο σύνδεσης μεταξύ κάθε εξαρτήματος που χρησιμοποιήθηκε για την δημιουργία της αράχνης

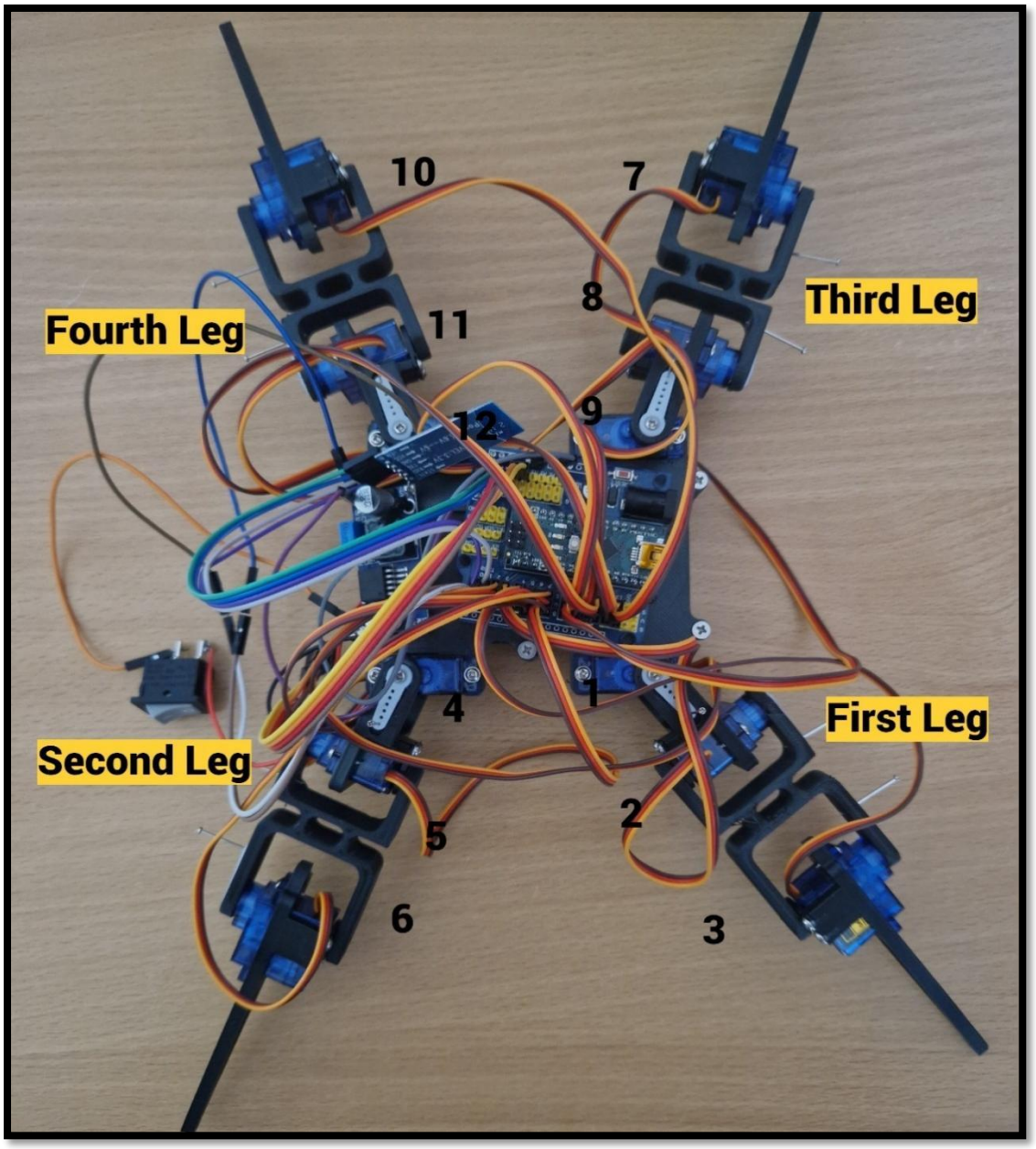


Εικόνα 20 : Διάγραμμα κυκλώματος ρομποτικής αράχνης

Πρώτα συνδέεται ο διακόπτης με την μπαταριά και τον LM2596. Ο LM2596 ρυθμίζεται στα 7V και συνδέεται με την nano io expansion shield παρέχοντας ρεύμα και στις δυο μεριές θηκών, από το OUT- ένα καλώδιο συνδέεται με TX G (Transmit) ενώ το άλλο συνδέεται με το GROUND στην άλλη μεριά. Το OUT+ συνδέεται με το

RX V (receive) και το Vin. Το κάθε servo συνδέεται με μια από τις ομάδες θηκών VGS(VCC, GROUND, SIGNAL). Το πορτοκαλί συνδέεται με το S , το κόκκινο με το V και το καφέ στο G.Στην παρακάτω εικόνα παρατηρουμε την σειρα που συνδεονται τα servo από το 2 εως το 13 Για το αισθητήρα υπέρυθρων συνδέονται το Vcc και GND στα A6 V και A6 G αντίστοιχα, ενώ τα ECHO και Trigger στο A4 S και A5 S. Τέλος για το Bluetooth module συνδέεται το TX και RX στα αντίστοιχα πάνω στην IO

expansion shield TX 0 και RX 1.



Εικονα:21 Σειρά σύνδεσης των Servo

4.3 Ανάλυση Κώδικα

Παρακάτω θα αναλυθούν οι κώδικες που χρησιμοποιήθηκαν για την ρομποτική αράχνη .

Κώδικας ρύθμισης ποδιών:

```
#include <Servo.h>

Servo servo[4][3];

//define servos' ports
const int servo_pin[4][3] = { {3, 4, 2}, {6, 7, 5}, {9, 8, 10},{12, 11, 13} };

void setup()
{
 //initialize all servos
 for (int i = 0; i < 4; i++)
 {
  for (int j = 0; j < 3; j++)
  {
   servo[i][j].attach(servo_pin[i][j]);
   delay(20);
  }
 }
}

void loop(void)
{
 for (int i = 0; i < 4; i++)
```

```
  {
   for (int j = 0; j < 3; j++)
   {
    servo[i][j].write(90);
    delay(20);
   }
  }
}
```

Ο παραπάνω Κώδικας χρησιμοποιείται για την ρύθμιση των αρχικών θέσεων τον 12 servo της ρομποτικής αράχνης . Η βιβλιοθήκη Servo.h μας επιτρέπει την δυνατότητα να ελέγχουμε servo μέσω Arduino board. Θέτετε δισδιάστατος πίνακας που κρατάει τα 12 servos και με το servo_pin αποθηκεύουμε τους αριθμούς ακίδων που αντιστοιχούν σε κάθε servo. Με το servo[][].attach loop σύστοιχή τις ακίδες με το κάθε servo. Με την συνάρτηση loop όλα τα servo θέτονται στις 90 μοίρες η οποία είναι η ουδέτερη θέση .

Κώδικας απλών κινήσεων:

```
/* Includes ------------------------------------------------------------------*/
#include <Servo.h>    //to define and control servos
#include <FlexiTimer2.h>//to set a timer to manage all servos
/* Servos --------------------------------------------------------------------*/
//define 12 servos for 4 legs
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = { {3, 4, 2}, {6, 7, 5}, {9, 8, 10},{12, 11, 13} };
/* Size of the robot ---------------------------------------------------------*/
const float length_a = 55;
```

```
const float length_b = 77.5;
const float length_c = 27.5;
const float length_side = 71;
const float z_absolute = -28;
/* Constants for movement ------------------------------------------------------*/
const float z_default = -50, z_up = -30, z_boot = z_absolute;
const float x_default = 62, x_offset = 0;
const float y_start = 0, y_step = 40;
const float y_default = x_default;
/* variables for movement ------------------------------------------------------*/
volatile float site_now[4][3];    //real-time coordinates of the end of each leg
volatile float site_expect[4][3]; //expected coordinates of the end of each leg
float temp_speed[4][3];   //each axis' speed, needs to be recalculated before each movement
float move_speed;     //movement speed
float speed_multiple = 1; //movement speed multiple
const float spot_turn_speed = 4;
const float leg_move_speed = 8;
const float body_move_speed = 3;
const float stand_seat_speed = 1;
volatile int rest_counter;      //+1/0.02s, for automatic rest
//functions' parameter
const float KEEP = 255;
//define PI for calculation
const float pi = 3.1415926;
/* Constants for turn ---------------------------------------------------------*/
//temp length
const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step, 2));
const float temp_b = 2 * (y_start + y_step) + length_side;
const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 * y_start + y_step + length_side,
2));
```

```
const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) - pow(temp_c, 2)) / 2 / temp_a / temp_b);
//site for turn
const float turn_x1 = (temp_a - length_side) / 2;
const float turn_y1 = y_start + y_step / 2;
const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;
/* ---------------------------------------------------------------------------*/


/*
  - setup function
   ----------------------------------------------------------------------------*/
void setup()
{
  //start serial for debug
  Serial.begin(115200);
  Serial.println("Robot starts initialization");
  //initialize default parameter
  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
  //start servo service
```

```
  FlexiTimer2::set(20, servo_service);
  FlexiTimer2::start();
  Serial.println("Servo service started");
  //initialize servos
  servo_attach();
  Serial.println("Servos initialized");
  Serial.println("Robot initialization Complete");
}



void servo_attach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].attach(servo_pin[i][j]);
      delay(100);
    }
  }
}

void servo_detach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].detach();
```

```
      delay(100);
    }
  }
}
/*
  - loop function
   ------------------------------------------------------------------------*/
void loop()
{
  Serial.println("Stand");
  stand();
  delay(2000);
  Serial.println("Step forward");
  step_forward(5);
  delay(2000);
  Serial.println("Step back");
  step_back(5);
  delay(2000);
  Serial.println("Turn left");
  turn_left(5);
  delay(2000);
  Serial.println("Turn right");
  turn_right(5);
  delay(2000);
  Serial.println("Hand wave");
  hand_wave(3);
  delay(2000);
  Serial.println("Hand wave");
  hand_shake(3);
```

```
  delay(2000);
  Serial.println("Body dance");
  body_dance(10);
  delay(2000);
  Serial.println("Sit");
  sit();
  delay(5000);
}

/*
  - sit
  - blocking function
   ------------------------------------------------------------------------*/
void sit(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_boot);
  }
  wait_all_reach();
}

/*
  - stand
  - blocking function
   ------------------------------------------------------------------------*/
void stand(void)
{
```

```
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_default);
  }
  wait_all_reach();
}



/*
  - spot turn to left
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------*/
void turn_left(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[3][1] == y_start)
    {
      //leg 3&1 move
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x1 - x_offset, turn_y1, z_default);
      set_site(1, turn_x0 - x_offset, turn_y0, z_default);
      set_site(2, turn_x1 + x_offset, turn_y1, z_default);
      set_site(3, turn_x0 + x_offset, turn_y0, z_up);
```

```
  wait_all_reach();

  set_site(3, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x1 + x_offset, turn_y1, z_default);
  set_site(1, turn_x0 + x_offset, turn_y0, z_default);
  set_site(2, turn_x1 - x_offset, turn_y1, z_default);
  set_site(3, turn_x0 - x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(1, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_default);
  set_site(1, x_default + x_offset, y_start, z_up);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 0&2 move
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();
```

```
      set_site(0, turn_x0 + x_offset, turn_y0, z_up);
      set_site(1, turn_x1 + x_offset, turn_y1, z_default);
      set_site(2, turn_x0 - x_offset, turn_y0, z_default);
      set_site(3, turn_x1 - x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x0 - x_offset, turn_y0, z_default);
      set_site(1, turn_x1 - x_offset, turn_y1, z_default);
      set_site(2, turn_x0 + x_offset, turn_y0, z_default);
      set_site(3, turn_x1 + x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(2, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_up);
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();

      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}
```

```
/*
  - spot turn to right
  - blocking function
  - parameter step steps wanted to turn
   ----------------------------------------------------------------------------*/
void turn_right(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[2][1] == y_start)
    {
      //leg 2&0 move
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x0 - x_offset, turn_y0, z_default);
      set_site(1, turn_x1 - x_offset, turn_y1, z_default);
      set_site(2, turn_x0 + x_offset, turn_y0, z_up);
      set_site(3, turn_x1 + x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(2, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_default);
      set_site(1, turn_x1 + x_offset, turn_y1, z_default);
      set_site(2, turn_x0 - x_offset, turn_y0, z_default);
```

```
  set_site(3, turn_x1 - x_offset, turn_y1, z_default);
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_up);
  set_site(1, x_default + x_offset, y_start, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 1&3 move
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x1 + x_offset, turn_y1, z_default);
  set_site(1, turn_x0 + x_offset, turn_y0, z_up);
  set_site(2, turn_x1 - x_offset, turn_y1, z_default);
  set_site(3, turn_x0 - x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(1, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();
```

```
      set_site(0, turn_x1 - x_offset, turn_y1, z_default);
      set_site(1, turn_x0 - x_offset, turn_y0, z_default);
      set_site(2, turn_x1 + x_offset, turn_y1, z_default);
      set_site(3, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(3, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_default);
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - go forward
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------*/
void step_forward(unsigned int step)
{
```

```
move_speed = leg_move_speed;
while (step-- > 0)
{
 if (site_now[2][1] == y_start)
 {
  //leg 2&1 move
  set_site(2, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();

  move_speed = body_move_speed;

  set_site(0, x_default + x_offset, y_start, z_default);
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;

  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
```

```
  }
  else
  {
    //leg 0&3 move
    set_site(0, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = body_move_speed;

    set_site(0, x_default - x_offset, y_start + y_step, z_default);
    set_site(1, x_default - x_offset, y_start + y_step, z_default);
    set_site(2, x_default + x_offset, y_start, z_default);
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = leg_move_speed;

    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_default);
    wait_all_reach();
  }
}
```

```
}

/*
  - go back
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------*/
void step_back(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
  {
    if (site_now[3][1] == y_start)
    {
      //leg 3&0 move
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = body_move_speed;

      set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
      set_site(1, x_default + x_offset, y_start, z_default);
      set_site(2, x_default - x_offset, y_start + y_step, z_default);
      set_site(3, x_default - x_offset, y_start + y_step, z_default);
      wait_all_reach();
```

```
  move_speed = leg_move_speed;

  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start, z_default);
  wait_all_reach();
 }
 else
 {
  //leg 1&2 move
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();

  move_speed = body_move_speed;

  set_site(0, x_default - x_offset, y_start + y_step, z_default);
  set_site(1, x_default - x_offset, y_start + y_step, z_default);
  set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
  set_site(3, x_default + x_offset, y_start, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;
```

```
      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

// add by RegisHsu

void body_left(int i)
{
  set_site(0, site_now[0][0] + i, KEEP, KEEP);
  set_site(1, site_now[1][0] + i, KEEP, KEEP);
  set_site(2, site_now[2][0] - i, KEEP, KEEP);
  set_site(3, site_now[3][0] - i, KEEP, KEEP);
  wait_all_reach();
}

void body_right(int i)
{
  set_site(0, site_now[0][0] - i, KEEP, KEEP);
  set_site(1, site_now[1][0] - i, KEEP, KEEP);
  set_site(2, site_now[2][0] + i, KEEP, KEEP);
  set_site(3, site_now[3][0] + i, KEEP, KEEP);
  wait_all_reach();
```

```
}

void hand_wave(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(2, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
```

```
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(0, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}

void hand_shake(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
```

```
  y_tmp = site_now[2][1];
  z_tmp = site_now[2][2];
  move_speed = body_move_speed;
  for (int j = 0; j < i; j++)
  {
    set_site(2, x_default - 30, y_start + 2 * y_step, 55);
    wait_all_reach();
    set_site(2, x_default - 30, y_start + 2 * y_step, 10);
    wait_all_reach();
  }
  set_site(2, x_tmp, y_tmp, z_tmp);
  wait_all_reach();
  move_speed = 1;
  body_left(15);
}
else
{
  body_left(15);
  x_tmp = site_now[0][0];
  y_tmp = site_now[0][1];
  z_tmp = site_now[0][2];
  move_speed = body_move_speed;
  for (int j = 0; j < i; j++)
  {
    set_site(0, x_default - 30, y_start + 2 * y_step, 55);
    wait_all_reach();
    set_site(0, x_default - 30, y_start + 2 * y_step, 10);
    wait_all_reach();
  }
```

```
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}

void head_up(int i)
{
  set_site(0, KEEP, KEEP, site_now[0][2] - i);
  set_site(1, KEEP, KEEP, site_now[1][2] + i);
  set_site(2, KEEP, KEEP, site_now[2][2] - i);
  set_site(3, KEEP, KEEP, site_now[3][2] + i);
  wait_all_reach();
}

void head_down(int i)
{
  set_site(0, KEEP, KEEP, site_now[0][2] + i);
  set_site(1, KEEP, KEEP, site_now[1][2] - i);
  set_site(2, KEEP, KEEP, site_now[2][2] + i);
  set_site(3, KEEP, KEEP, site_now[3][2] - i);
  wait_all_reach();
}

void body_dance(int i)
{
  float x_tmp;
  float y_tmp;
```

```
float z_tmp;
float body_dance_speed = 2;
sit();
move_speed = 1;
set_site(0, x_default, y_default, KEEP);
set_site(1, x_default, y_default, KEEP);
set_site(2, x_default, y_default, KEEP);
set_site(3, x_default, y_default, KEEP);
wait_all_reach();
//stand();
set_site(0, x_default, y_default, z_default - 20);
set_site(1, x_default, y_default, z_default - 20);
set_site(2, x_default, y_default, z_default - 20);
set_site(3, x_default, y_default, z_default - 20);
wait_all_reach();
move_speed = body_dance_speed;
head_up(30);
for (int j = 0; j < i; j++)
{
 if (j > i / 4)
   move_speed = body_dance_speed * 2;
 if (j > i / 2)
   move_speed = body_dance_speed * 3;
 set_site(0, KEEP, y_default - 20, KEEP);
 set_site(1, KEEP, y_default + 20, KEEP);
 set_site(2, KEEP, y_default - 20, KEEP);
 set_site(3, KEEP, y_default + 20, KEEP);
 wait_all_reach();
 set_site(0, KEEP, y_default + 20, KEEP);
```

```
    set_site(1, KEEP, y_default - 20, KEEP);
    set_site(2, KEEP, y_default + 20, KEEP);
    set_site(3, KEEP, y_default - 20, KEEP);
    wait_all_reach();
  }
  move_speed = body_dance_speed;
  head_down(30);
}



/*
  - microservos service /timer interrupt function/50Hz
  - when set site expected,this function move the end point to it in a straight line
  - temp_speed[4][3] should be set before set expect site,it make sure the end point
   move in a straight line,and decide move speed.
   ------------------------------------------------------------------------------*/
void servo_service(void)
{
  sei();
  static float alpha, beta, gamma;

  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      if (abs(site_now[i][j] - site_expect[i][j]) >= abs(temp_speed[i][j]))
        site_now[i][j] += temp_speed[i][j];
      else
        site_now[i][j] = site_expect[i][j];
```

```
    }

    cartesian_to_polar(alpha, beta, gamma, site_now[i][0], site_now[i][1], site_now[i][2]);
    polar_to_servo(i, alpha, beta, gamma);
  }

  rest_counter++;
}

/*
  - set one of end points' expect site
  - this founction will set temp_speed[4][3] at same time
  - non - blocking function
   ------------------------------------------------------------------------*/
void set_site(int leg, float x, float y, float z)
{
  float length_x = 0, length_y = 0, length_z = 0;

  if (x != KEEP)
    length_x = x - site_now[leg][0];
  if (y != KEEP)
    length_y = y - site_now[leg][1];
  if (z != KEEP)
    length_z = z - site_now[leg][2];

  float length = sqrt(pow(length_x, 2) + pow(length_y, 2) + pow(length_z, 2));

  temp_speed[leg][0] = length_x / length * move_speed * speed_multiple;
  temp_speed[leg][1] = length_y / length * move_speed * speed_multiple;
```

```
  temp_speed[leg][2] = length_z / length * move_speed * speed_multiple;


  if (x != KEEP)
    site_expect[leg][0] = x;
  if (y != KEEP)
    site_expect[leg][1] = y;
  if (z != KEEP)
    site_expect[leg][2] = z;
}


/*
  - wait one of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_reach(int leg)
{
  while (1)
    if (site_now[leg][0] == site_expect[leg][0])
      if (site_now[leg][1] == site_expect[leg][1])
        if (site_now[leg][2] == site_expect[leg][2])
          break;
}


/*
  - wait all of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_all_reach(void)
{
```

```
  for (int i = 0; i < 4; i++)
    wait_reach(i);
}

/*
  - trans site from cartesian to polar
  - mathematical model 2/2
  ----------------------------------------------------------------------------*/
void cartesian_to_polar(volatile float &alpha, volatile float &beta, volatile float &gamma, volatile float x, volatile float y, volatile float z)
{
  //calculate w-z degree
  float v, w;
  w = (x >= 0 ? 1 : -1) * (sqrt(pow(x, 2) + pow(y, 2)));
  v = w - length_c;
  alpha = atan2(z, v) + acos((pow(length_a, 2) - pow(length_b, 2) + pow(v, 2) + pow(z, 2)) / 2 / length_a / sqrt(pow(v, 2) + pow(z, 2)));
  beta = acos((pow(length_a, 2) + pow(length_b, 2) - pow(v, 2) - pow(z, 2)) / 2 / length_a / length_b);
  //calculate x-y-z degree
  gamma = (w >= 0) ? atan2(y, x) : atan2(-y, -x);
  //trans degree pi->180
  alpha = alpha / pi * 180;
  beta = beta / pi * 180;
  gamma = gamma / pi * 180;
}

/*
  - trans site from polar to microservos
  - mathematical model map to fact
  - the errors saved in eeprom will be add
```

```
  ----------------------------------------------------------------------------*/
void polar_to_servo(int leg, float alpha, float beta, float gamma)
{
 if (leg == 0)
 {
  alpha = 90 - alpha;
  beta = beta;
  gamma += 90;
 }
 else if (leg == 1)
 {
  alpha += 90;
  beta = 180 - beta;
  gamma = 90 - gamma;
 }
 else if (leg == 2)
 {
  alpha += 90;
  beta = 180 - beta;
  gamma = 90 - gamma;
 }
 else if (leg == 3)
 {
  alpha = 90 - alpha;
  beta = beta;
  gamma += 90;
 }

 servo[leg][0].write(alpha);
```

```
  servo[leg][1].write(beta);
  servo[leg][2].write(gamma);
}
```

Σε αυτό τον κώδικα χρησιμοποιείται μια ακόμη βιβλιοθήκη η Flexitimer2.h για την ενημέρωση των θέσεων του σερβομηχανισμού κάθε 20 χιλιοστά του δευτερολέπτου. Ο κύρια συνάρτηση του προγράμματος εκτελεί μια ακολουθία προκαθορισμένων ρομποτικών ενεργειών, όπως η ορθοστασία, το βηματισμό και πιο δυναμικές κινήσεις όπως το κούνημα των χεριών και ο χορός. Κάθε ενέργεια χρησιμοποιεί συγκεκριμένες βοηθητικές λειτουργίες που συντονίζουν τις κινήσεις της αράχνης μέσω στοχευμένης τοποθέτησης των ποδιών και προσαρμογής της ταχύτητας. Αλλάζοντας τα spot_turn_speed ,leg_move_speed , body_move_speed και stand_seat_speed μας δίνει την δυνατότητα να αλλάξουμε την ταχύτητα κίνησης της αράχνης. Η κίνηση της αράχνης εξαρτάται από την μετατροπή των καρτεσιανών συντεταγμένων σε πολικές συντεταγμένες και στη συνέχεια τη μετατροπή τους σε σερβογωνίες για τον ακριβή έλεγχο της κίνησης κάθε ποδιού. Αυτό επιτυγχάνεται μέσω λειτουργιών όπως cartesian_to_polar και polar_to_servo, εξασφαλίζοντας ακριβή τοποθέτηση των ποδιών. Η συνάρτηση servo_service, που ενεργοποιείται από ένα χρονοδιακόπτη, ρυθμίζει σταδιακά τις θέσεις των σερβομηχανισμών, διατηρώντας σταθερές ενημερώσεις κίνησης ανεξάρτητα από τη ροή του κύριου προγράμματος. Οι λειτουργίες wait_reach και wait_all_reach, διασφαλίζουν ότι όλες οι κινήσεις ολοκληρώνονται πριν προχωρήσουν σε επόμενες ενέργειες.

Κώδικας για αποφυγή εμποδίων:

```
#include <Servo.h>    //to define and control servos
#include <FlexiTimer2.h>//to set a timer to manage all servos
#include <NewPing.h>   // ultrasonic distance sensor library
/* Servos --------------------------------------------------------------------*/
//define 12 servos for 4 legs
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = { {3, 4, 2}, {6, 7, 5}, {9, 8, 10},{12, 11, 13} };
/* Size of the robot ---------------------------------------------------------*/
```

```
const float length_a = 55;
const float length_b = 77.5;
const float length_c = 27.5;
const float length_side = 71;
const float z_absolute = -28;
/* Constants for movement ------------------------------------------------*/
const float z_default = -50, z_up = -30, z_boot = z_absolute;
const float x_default = 62, x_offset = 0;
const float y_start = 0, y_step = 40;
/* variables for movement ------------------------------------------------*/
volatile float site_now[4][3];    //real-time coordinates of the end of each leg
volatile float site_expect[4][3]; //expected coordinates of the end of each leg
float temp_speed[4][3];   //each axis' speed, needs to be recalculated before each movement
float move_speed;     //movement speed
float speed_multiple = 1.2; //movement speed multiple
const float spot_turn_speed = 4;
const float leg_move_speed = 8;
const float body_move_speed = 3;
const float stand_seat_speed = 1;
volatile int rest_counter;      //+1/0.02s, for automatic rest
//functions' parameter
const float KEEP = 255;
//define PI for calculation
const float pi = 3.1415926;
/* Constants for turn ----------------------------------------------------*/
//temp length
const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step, 2));
const float temp_b = 2 * (y_start + y_step) + length_side;
const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 * y_start + y_step + length_side, 2));
const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) - pow(temp_c, 2)) / 2 / temp_a / temp_b);
//site for turn
const float turn_x1 = (temp_a - length_side) / 2;
const float turn_y1 = y_start + y_step / 2;
const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
```

```
const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;

//for ultrasonic sensor
#define TRIGGER_PIN  19
#define ECHO_PIN     18
#define MAX_DISTANCE 200
int thresh = 20;
long randNumber;

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
/* ---------------------------------------------------------------------------*/

/*
  - setup function
   ---------------------------------------------------------------------------*/
void setup()
{
  //start serial for debug
  Serial.begin(115200);
  Serial.println("Robot starts initialization");
   delay(5000);
  //initialize default parameter
  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
  //start servo service
  FlexiTimer2::set(20, servo_service);
```

```
  FlexiTimer2::start();
  Serial.println("Servo service started");
  //initialize servos
  servo_attach();
  Serial.println("Servos initialized");
  Serial.println("Robot initialization Complete");

  randomSeed(analogRead(0));

      Serial.println("Stand");
      stand();
      delay(500);

      hand_wave(3);
      delay(1000);
}


void servo_attach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      delay(50);
      servo[i][j].attach(servo_pin[i][j]);
      delay(50);
    }
  }
}

void servo_detach(void)
{
  for (int i = 0; i < 4; i++)
  {
```

```
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].detach();
      delay(100);
    }
  }
}
/*
  - loop function
   --------------------------------------------------------------------------*/
void loop()
{
  int uS = sonar.ping_cm();
//  Serial.print("Ping: ");
//  Serial.print(uS);
//  Serial.println("cm");
     if (uS < thresh && uS > 1){
        step_back(3);
        delay(500);

  randNumber = random(1,3);
  Serial.println(randNumber);

        if(randNumber == 1){
          turn_right(5);
          delay(500);
        }
        else if(randNumber == 2){
          turn_left(5);
          delay(500);
        }
     }

     else {
       step_forward(1);
```

```
    }
}

/*
  - sit
  - blocking function
  ---------------------------------------------------------------------------*/
void sit(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_boot);
  }
  wait_all_reach();
}

/*
  - stand
  - blocking function
  ---------------------------------------------------------------------------*/
void stand(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_default);
  }
  wait_all_reach();
}


/*
  - spot turn to left
  - blocking function
```

```
  - parameter step steps wanted to turn
   ---------------------------------------------------------------------------*/
void turn_left(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[3][1] == y_start)
    {
      //leg 3&1 move
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x1 - x_offset, turn_y1, z_default);
      set_site(1, turn_x0 - x_offset, turn_y0, z_default);
      set_site(2, turn_x1 + x_offset, turn_y1, z_default);
      set_site(3, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(3, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x1 + x_offset, turn_y1, z_default);
      set_site(1, turn_x0 + x_offset, turn_y0, z_default);
      set_site(2, turn_x1 - x_offset, turn_y1, z_default);
      set_site(3, turn_x0 - x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(1, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default + x_offset, y_start, z_default);
      set_site(1, x_default + x_offset, y_start, z_up);
      set_site(2, x_default - x_offset, y_start + y_step, z_default);
      set_site(3, x_default - x_offset, y_start + y_step, z_default);
```

```
  wait_all_reach();

  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 0&2 move
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_up);
  set_site(1, turn_x1 + x_offset, turn_y1, z_default);
  set_site(2, turn_x0 - x_offset, turn_y0, z_default);
  set_site(3, turn_x1 - x_offset, turn_y1, z_default);
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x0 - x_offset, turn_y0, z_default);
  set_site(1, turn_x1 - x_offset, turn_y1, z_default);
  set_site(2, turn_x0 + x_offset, turn_y0, z_default);
  set_site(3, turn_x1 + x_offset, turn_y1, z_default);
  wait_all_reach();

  set_site(2, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default - x_offset, y_start + y_step, z_default);
  set_site(1, x_default - x_offset, y_start + y_step, z_default);
  set_site(2, x_default + x_offset, y_start, z_up);
  set_site(3, x_default + x_offset, y_start, z_default);
  wait_all_reach();
```

```
      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - spot turn to right
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------*/
void turn_right(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[2][1] == y_start)
    {
      //leg 2&0 move
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x0 - x_offset, turn_y0, z_default);
      set_site(1, turn_x1 - x_offset, turn_y1, z_default);
      set_site(2, turn_x0 + x_offset, turn_y0, z_up);
      set_site(3, turn_x1 + x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(2, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_default);
      set_site(1, turn_x1 + x_offset, turn_y1, z_default);
      set_site(2, turn_x0 - x_offset, turn_y0, z_default);
      set_site(3, turn_x1 - x_offset, turn_y1, z_default);
```

```
  wait_all_reach();

  set_site(0, turn_x0 + x_offset, turn_y0, z_up);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_up);
  set_site(1, x_default + x_offset, y_start, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  set_site(0, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 1&3 move
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();

  set_site(0, turn_x1 + x_offset, turn_y1, z_default);
  set_site(1, turn_x0 + x_offset, turn_y0, z_up);
  set_site(2, turn_x1 - x_offset, turn_y1, z_default);
  set_site(3, turn_x0 - x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(1, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();

  set_site(0, turn_x1 - x_offset, turn_y1, z_default);
  set_site(1, turn_x0 - x_offset, turn_y0, z_default);
  set_site(2, turn_x1 + x_offset, turn_y1, z_default);
  set_site(3, turn_x0 + x_offset, turn_y0, z_default);
  wait_all_reach();
```

```
    set_site(3, turn_x0 + x_offset, turn_y0, z_up);
    wait_all_reach();

    set_site(0, x_default - x_offset, y_start + y_step, z_default);
    set_site(1, x_default - x_offset, y_start + y_step, z_default);
    set_site(2, x_default + x_offset, y_start, z_default);
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();

    set_site(3, x_default + x_offset, y_start, z_default);
    wait_all_reach();
   }
  }
}

/*
  - go forward
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------*/
void step_forward(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
  {
   if (site_now[2][1] == y_start)
   {
    //leg 2&1 move
    set_site(2, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();
```

```
  move_speed = body_move_speed;

  set_site(0, x_default + x_offset, y_start, z_default);
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;

  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 0&3 move
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();

  move_speed = body_move_speed;

  set_site(0, x_default - x_offset, y_start + y_step, z_default);
  set_site(1, x_default - x_offset, y_start + y_step, z_default);
  set_site(2, x_default + x_offset, y_start, z_default);
  set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();
```

```
    move_speed = leg_move_speed;

    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_default);
    wait_all_reach();
   }
  }
}

/*
  - go back
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------*/
void step_back(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
  {
   if (site_now[3][1] == y_start)
   {
    //leg 3&0 move
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = body_move_speed;

    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
```

```
  set_site(1, x_default + x_offset, y_start, z_default);
  set_site(2, x_default - x_offset, y_start + y_step, z_default);
  set_site(3, x_default - x_offset, y_start + y_step, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;

  set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(0, x_default + x_offset, y_start, z_default);
  wait_all_reach();
}
else
{
  //leg 1&2 move
  set_site(1, x_default + x_offset, y_start, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
  wait_all_reach();
  set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
  wait_all_reach();

  move_speed = body_move_speed;

  set_site(0, x_default - x_offset, y_start + y_step, z_default);
  set_site(1, x_default - x_offset, y_start + y_step, z_default);
  set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
  set_site(3, x_default + x_offset, y_start, z_default);
  wait_all_reach();

  move_speed = leg_move_speed;

  set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
```

```
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

// add by RegisHsu

void body_left(int i)
{
  set_site(0, site_now[0][0] + i, KEEP, KEEP);
  set_site(1, site_now[1][0] + i, KEEP, KEEP);
  set_site(2, site_now[2][0] - i, KEEP, KEEP);
  set_site(3, site_now[3][0] - i, KEEP, KEEP);
  wait_all_reach();
}

void body_right(int i)
{
  set_site(0, site_now[0][0] - i, KEEP, KEEP);
  set_site(1, site_now[1][0] - i, KEEP, KEEP);
  set_site(2, site_now[2][0] + i, KEEP, KEEP);
  set_site(3, site_now[3][0] + i, KEEP, KEEP);
  wait_all_reach();
}

void hand_wave(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
```

```
if (site_now[3][1] == y_start)
{
  body_right(15);
  x_tmp = site_now[2][0];
  y_tmp = site_now[2][1];
  z_tmp = site_now[2][2];
  move_speed = body_move_speed;
  for (int j = 0; j < i; j++)
  {
    set_site(2, turn_x1, turn_y1, 50);
    wait_all_reach();
    set_site(2, turn_x0, turn_y0, 50);
    wait_all_reach();
  }
  set_site(2, x_tmp, y_tmp, z_tmp);
  wait_all_reach();
  move_speed = 1;
  body_left(15);
}
else
{
  body_left(15);
  x_tmp = site_now[0][0];
  y_tmp = site_now[0][1];
  z_tmp = site_now[0][2];
  move_speed = body_move_speed;
  for (int j = 0; j < i; j++)
  {
    set_site(0, turn_x1, turn_y1, 50);
    wait_all_reach();
    set_site(0, turn_x0, turn_y0, 50);
    wait_all_reach();
  }
  set_site(0, x_tmp, y_tmp, z_tmp);
  wait_all_reach();
```

```
    move_speed = 1;
    body_right(15);
  }
}

void hand_shake(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(2, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
```

```
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(0, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}



/*
  - microservos service /timer interrupt function/50Hz
  - when set site expected,this function move the end point to it in a straight line
  - temp_speed[4][3] should be set before set expect site,it make sure the end point
   move in a straight line,and decide move speed.
   ----------------------------------------------------------------------------*/
void servo_service(void)
{
  sei();
  static float alpha, beta, gamma;

  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      if (abs(site_now[i][j] - site_expect[i][j]) >= abs(temp_speed[i][j]))
        site_now[i][j] += temp_speed[i][j];
```

```
      else
        site_now[i][j] = site_expect[i][j];
    }

    cartesian_to_polar(alpha, beta, gamma, site_now[i][0], site_now[i][1], site_now[i][2]);
    polar_to_servo(i, alpha, beta, gamma);
  }

  rest_counter++;
}

/*
  - set one of end points' expect site
  - this founction will set temp_speed[4][3] at same time
  - non - blocking function
   ---------------------------------------------------------------------------*/
void set_site(int leg, float x, float y, float z)
{
  float length_x = 0, length_y = 0, length_z = 0;

  if (x != KEEP)
    length_x = x - site_now[leg][0];
  if (y != KEEP)
    length_y = y - site_now[leg][1];
  if (z != KEEP)
    length_z = z - site_now[leg][2];

  float length = sqrt(pow(length_x, 2) + pow(length_y, 2) + pow(length_z, 2));

  temp_speed[leg][0] = length_x / length * move_speed * speed_multiple;
  temp_speed[leg][1] = length_y / length * move_speed * speed_multiple;
  temp_speed[leg][2] = length_z / length * move_speed * speed_multiple;

  if (x != KEEP)
    site_expect[leg][0] = x;
```

```
  if (y != KEEP)
    site_expect[leg][1] = y;
  if (z != KEEP)
    site_expect[leg][2] = z;
}

/*
  - wait one of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_reach(int leg)
{
  while (1)
    if (site_now[leg][0] == site_expect[leg][0])
      if (site_now[leg][1] == site_expect[leg][1])
        if (site_now[leg][2] == site_expect[leg][2])
          break;
}

/*
  - wait all of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_all_reach(void)
{
  for (int i = 0; i < 4; i++)
    wait_reach(i);
}

/*
  - trans site from cartesian to polar
  - mathematical model 2/2
   ----------------------------------------------------------------------------*/
void cartesian_to_polar(volatile float &alpha, volatile float &beta, volatile float &gamma, volatile
float x, volatile float y, volatile float z)
{
```

```
  //calculate w-z degree
  float v, w;
  w = (x >= 0 ? 1 : -1) * (sqrt(pow(x, 2) + pow(y, 2)));
  v = w - length_c;
  alpha = atan2(z, v) + acos((pow(length_a, 2) - pow(length_b, 2) + pow(v, 2) + pow(z, 2)) / 2 / length_a / sqrt(pow(v, 2) + pow(z, 2)));
  beta = acos((pow(length_a, 2) + pow(length_b, 2) - pow(v, 2) - pow(z, 2)) / 2 / length_a / length_b);
  //calculate x-y-z degree
  gamma = (w >= 0) ? atan2(y, x) : atan2(-y, -x);
  //trans degree pi->180
  alpha = alpha / pi * 180;
  beta = beta / pi * 180;
  gamma = gamma / pi * 180;
}

/*
  - trans site from polar to microservos
  - mathematical model map to fact
  - the errors saved in eeprom will be add
   ----------------------------------------------------------------------------*/
void polar_to_servo(int leg, float alpha, float beta, float gamma)
{
  if (leg == 0)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  else if (leg == 1)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 2)
  {
```

```
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 3)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }

  servo[leg][0].write(alpha);
  servo[leg][1].write(beta);
  servo[leg][2].write(gamma);
}
```

Η μονή αλλαγή σε αυτό τον κώδικα είναι η χρήση του αισθητήρα υπέρηχων και η προσθήκη της βιβλιοθήκης newping. Κατά την εκκίνηση, η ρομποτική αράχνη καθορίζει τις προεπιλεγμένες θέσεις για τα πόδια του και ενεργοποιεί έναν σερβοχρονοδιακόπτη που ενημερώνει τις θέσεις με συχνότητα 50 Hz,. Το κύριο πρόγραμμα χρησιμοποιεί τον αισθητήρα υπερήχων για την ανίχνευση εμποδίων, εκτελώντας ελιγμούς, όπως η οπισθοχώρηση ή η στροφή, προσθέτοντας ένα επίπεδο ανταπόκρισης στο περιβάλλον.

Υπήρχε και ένας τέταρτος κώδικας με την χρήση του Bluetooth module για τηλεκατευθυνόμενή κίνηση της ρομποτικής αράχνης αλλά δυστυχώς λόγο ότι εφαρμογή έχει σταματήσει να λειτουργεί σωστά δεν είναι δυνατή η τηλεκατευθυνόμενη κίνηση της ρομποτικής αράχνης για τώρα.

Όλοι οι κώδικες και STL αρχεία που χρησιμοποιήθηκαν μπορούν βρεθούν σε αυτή την σελίδα:https://robotlk.com/how-to-make-3-in-1-spider-robot-quadruped-arduino-nano/

Επίσης παρακάτω είναι τα βίντεο που δείχνουν την αράχνη να εκτελεί τους κώδικες:

Βασικές κινήσεις:
https://youtu.be/J1PfwMD365Q

Αποφυγή εμποδίων :

https://youtu.be/Oh4MgxsNLqU

Αράχνη χορεύει ζεϊμπέκικο:

https://drive.google.com/file/d/1Hv0sQFcqed6sXLGVBw1jBjvFNO44wJZJ/view

Επίλογος

Η παρούσα εργασία ανέδειξε τον τρόπο κατασκευής και ανάλυσης μιας ρομποτικής αράχνης, Με την εξέλιξη της τεχνολογίας, η συμβολή της ρομποτικής αράχνης είναι σημαντική για την παρακολούθηση του περιβάλλοντος, δηλαδή με λίγα λόγια παρέχει πληροφορίες για την κατάσταση ενός περιβάλλοντος που διαφορετικά δεν θα ήταν διαθέσιμο για την ανθρωπότητα.

Συγκεκριμένα, μια ρομποτική αράχνη με κατάλληλο μηχανισμό ελέγχου έχει την ικανότητα να κυκλοφορεί σε οποιαδήποτε επιφάνεια λόγω της ευέλικτης κίνησης που διαθέτει. Η παρούσα εργασία, επίσης διερευνά την απόδοση μιας ρομποτικής αράχνης, η οποία εξυπηρετεί διάφορους πρακτικούς σκοπούς. Πιο αναλυτικά, η ρομποτική αράχνη έχει την ικανότητα να έρπει σε μικρότερους χώρους όπως σε κτίρια που έχουν καταρρεύσει και να παρέχει πληροφορίες τοποθεσίας σχετικά με τα παγιδευμένα θύματα, καθώς και εκτιμήσεις ζημιών του χώρου, δηλαδή χρησιμοποιείται για απαιτητικές και επικίνδυνες αποστολές που είναι επικίνδυνες για τους ανθρώπους, όπως η εύρεση επιζώντων σε ασταθή ερείπια και η εξερεύνηση του διαστήματος, των ορυχείων και των ναυαγίων.

Προτάσεις για μελλοντική μελέτη

Η παρούσα μελέτη διερεύνησε τη συμβολή της ρομποτικής αράχνης, η οποία λειτουργεί χωρίς ανθρώπινη διασύνδεση και μάλιστα έχει δυνατότητες κίνησης εκτός του βασικού περπατήματος, δυνατότητα πλοήγησης σε δύσκολα εδάφη αλλά και αναρρίχησης σε ανώμαλες επιφάνειες. Ωστόσο, η παρούσα εργασία αφήνει προοπτικές για περαιτέρω μελέτη και έρευνα στο μέλλον.

Γενικά, η ρομποτική αράχνη έχει αρκετές ικανότητες εξέλιξης, όμως για να υλοποιηθούν χρειάζονται κυρίως οικονομικοί πόροι για την περαιτέρω βελτίωση της και την διόρθωση σφαλμάτων που προέκυψαν όπως για παράδειγμα η αδυναμία χειρισμού της αράχνης με τηλεχειριστήριο . Αρχικά, μία πρωταρχική σκέψη για επέκταση της εφαρμογής μας αποτελεί η δυνατότητα να δέχεται η ρομποτική αράχνη

φωνητικές εντολές από τον χρήστη, επίσης ενσωμάτωση κάμερας για τον καλύτερο εντοπισμό αντικειμένων ή εμποδίων . Κλείνοντας, την παρούσα εργασία, μία ακόμα πιθανή πρόταση αφορά την αναγνώριση αντικειμένων, Πρόκειται βέβαια για μια πιο σύνθετη υλοποίηση, η οποία απαιτεί αλγορίθμους και τεχνικές αναγνώρισης εικόνων και ανάλυσης δεδομένων.

Βιβλιογραφικές αναφορές

Ξενόγλωσση

1. Chao Sun et al 2019 J. Phys.: Conf. Ser. 1168 022094 . Design and Simulation Analysis of Hexapod Bionic Spider Robot
2. Zihao Yang et al 2019 J. Phys.: Conf. Ser. 1314 012109 Mechanism Design and Kinematics Analysis of Spider-like Octopod Robot
3. Takeo OHNISHI and Toshiyuki ASAKURA.: Spider-Robot 8-Leg Cooperative Walking Velocity Control Strategy Based on Environmental Information
4. Santiago Noriega Álvarez, María Camila Rojas, Hernando Leon-Rodriguez Universidad El Bosque – Bogotá – Colombia Design and development of a quadruped spider robot
5. S.Yu. Misyurina , G.V. Kreinin , N.Yu. Nosova ,A.P. Nelubin Six-Legged Walking Robot (Hexabot), Kinematics, Dynamics and Motion Optimization
6. Rozita Teymourzadeh, Rahim Nakhli Nakhli Mahal, Ng Keng Shen, Kok Wai Chan. Adaptive Intelligent Spider Robot. IEEE 8th International colloquium on signal Processing Applications conference (CSPA 2012). pp: 99-104. DOI: 10.1109/CSPA.2012.6194699, 2012, Kuala Lumpur Malaysia. hal 01800750
7. Autonomous Nanorobots as Miniaturized Surgeons for Intracellular Applications by Daitian Tang Xiqi Peng Song Wu andSongsong Tang
8. Biomimetic Spider Leg Joints: A Review from Biomechanical Research to Compliant Robotic Actuators. Stefan Landkammer ,Florian Winter ,Daniel Schneider ,Rüdiger Hornfeck

9. Richa watiasih, galuh widia rohman, prihatsono, Ahmadi gait control applications on four-legged robot
10. Spiderino- a low-cost robot for swarm research and educational purposes Midhat Jdeed1, Sergii Zhevzhyk2. Florian Steinkellner3 and Wilfried Elmenreich4 Alpen-Adria-University¨ at Klagenfurt, Austria

Ελληνόγλωσση

1. Δελή, Γ. (2011). Διπλωματική εργασία : Εκπαιδευτική αξιοποίηση ρομποτικών κατασκευών στη διδασκαλία μαθηματικών εννοιών και πληροφορικής. Πανεπιστήμιο Πατρών, Σχολή Θετικών Επιστημών, Τμήμα Μαθηματικών.
2. Μυλωνά, Α. (2014). Διπλωματική Εργασία : Αξιοποίηση της Ρομποτικής Κοινωνικής Υποστήριξης στο χώρο του Δημοτικού Σχολείου. Πανεπιστήμιο Δυτικής Μακεδονίας- Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Διατμηματικό μεταπτυχιακό τμήμα Νηπιαγωγών – Φυσικής Αγωγής και Αθλητισμού.