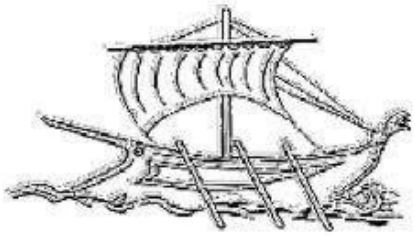
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ



Τ.Ε.Ι ΠΕΙΡΑΙΑ

ΟΠΤΙΚΗ ΟΔΟΜΕΤΡΙΑ ΣΕ ΑΥΤΟΚΙΝΟΥΜΕΝΗ ΡΟΜΠΟΤΙΚΗ ΔΙΑΤΑΞΗ

ΗΛΙΑΣ ΑΔΑΜΙΔΗΣ

ΓΡΗΓΟΡΗΣ ΝΙΚΟΛΑΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΙΟΥΝΙΟΣ 2017 (28/6/2017)

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ Π. Ραλλη & Θηβών 250,
12244 Αιγάλαιω , Αθήνα – Ελλάδα Τηλ. 210-5381488

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο / ~~Η~~ κάτωθι υπογεγραμμένος / η ΗΛΙΑΣ ΔΑΜΙΔΗΣ,

του ΒΑΛΕΡΙΙ, με αριθμό μητρώου 38220 φοιτητής / τρια του Τμήματος **Μηχανικών Αυτοματισμού Τ.Ε.** του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

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

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

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

Ο Δηλών

Ημερομηνία
30/6/2017

Περιεχόμενα

Πρόλογος

Στην εποχή μας η ανάπτυξη της ρομποτικής και πληροφορικής επιστήμης καθώς και της τεχνητής νοημοσύνης μας ωθεί στην αναζήτηση τρόπων βελτίωσης και εκτίμησης ευφυίας των ρομποτικών συστημάτων. Η επίλυση προβλημάτων ανθρώπινης φύσης καθώς και λογικά παιγνίδια, όπως είναι το σκάκι, το κινέζικης προέλευσης «Go!», ο κύβος του «Rubik» καθώς και η ανεύρεση εξόδου σε έναν λαβύρινθο είναι μερικά παραδείγματα δοκιμασίων που καλούνται να περάσουν.

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η αξιοποίηση της μηχατρονικής επιστήμης για την κατασκευή ενός τροχοφόρου ρομπότ που θα επιλύει λαβυρίνθους. Για τον λόγο αυτό στήνεται ένα δίτροχο όχημα με αισθητήρες ανάκλασης υπέρυθρης ακτινοβολίας (IR analogue reflectance sensors). Θα κινείται από δύο DC κινητήρες **6-12V** οι οποίοι θα οδηγούνται από κύκλωμα **H – Bridge**. Ελεγκτής επιλέγεται ο «**Arduino Uno**» λόγω της ευκολίας διαχείρισης ψηφιακών και αναλογικών σημάτων και ευχρηστίας στον προγραμματισμό. Ο αλγόριθμος επίλυσης του λαβύρινθου είναι ο «**Κανόνας του αριστερού χεριού**». Η τροφοδοσία του ρομπότ θα γίνεται από μπαταρία **9V**.

Το όχημα θα κινείται σε λευκή επιφάνεια με μαύρες γραμμές. Οι μαύρες γραμμές είναι οι διάδρομοι στους οποίους θα επιτρέπεται να κινείται το όχημα και θα σχηματίζουν έναν λαβύρινθο με ένα τέλος και πολλές διακλαδώσεις. Η αποστολή του «μήχατρον» είναι η ανεύρεση της εξόδου του λαβύρινθου, από ένα σημείο του που θα το θεωρήσουμε ως αρχή, στον τερματισμό του λαβύρινθου. Κατόπιν αυτού να σταματάει και να αναμένει την επαναφορά του στην αρχή. Τέλος να επαναλαμβάνει την διαδρομή με τον βέλτιστο τρόπο.

Με αυτόν τον τρόπο πετυχαίνουμε να έχουμε ρομπότ επίλυσης προβλήματος ανθρώπινης φύσης με την χρήση αλγόριθμων.

Κεφάλαιο 1

Μηχατρονική

Μηχατρονική: = **Μηχα**νολογία + Ηλε**κτρον**ική + Πληροφορ**ική** Παρόμοιος όρος για την Μηχατρονική είναι η Τεχνική Κυβερνητική - Technical Cybernetics.

Σπανίως χρησιμοποιείται και ο όρος Μηχανοτρονική ή Ηλεκρομηχανολογικοί Αυτοματισμοί.

[1]Η Μηχατρονική θεωρείται εμπλουτισμός των κατά βάση μηχανολογικών συστημάτων με ηλεκτρονικά εξαρτήματα που αρκετά συχνά εμπεριέχουν λογισμικό, δηλαδή:

Μηχατρονική είναι η συνεργία τριών επιστημών

- Μηχανολογία
- Ηλεκτρονική/Ηλεκτρολογία
- Πληροφορική

με σκοπό τη δημιουργία συστημάτων που να απλοποιούν την παραγωγή.

1.1 Φιλοσοφία της μηχατρονικής

Στην φιλοσοφία της μηχατρονικής, ο ενσωματωμένος υπολογιστής ελέγχου είναι το κεντρικό στοιχείο, και ο πυρήνας της τεχνολογίας η οποία την καθιστά την Μηχατρονική ένα μοναδικό τομέα. Ψηφιακά και αναλογικά κυκλώματα, μαζί με επενεργητές και επιστημονικά όργανα περιβάλλουν άμεσα τον υπολογιστή ελέγχου και λειτουργούν προσαρμοστικά μεταξύ του υπολογιστή και του ελεγχόμενου φυσικού συστήματος. Τα χαρακτηριστικά που διαφοροποιούν το κάθε σύγχρονο μηχανικό σύστημα, καθορίζονται σε μεγάλο βαθμό από την εφευρετικότητα και αποτελεσματικότητα του ενσωματωμένου σε αυτό λογισμικό. Τα παρεμβαλλόμενα στοιχεία υποστηρίζουν το λογισμικό αυτό παρέχοντας του τις τρέχουσες πληροφορίες από το ελεγχόμενο σύστημα και μεταφράζοντας τις εντολές του σε ενεργή παροχή διαμορφωμένης ισχύος.

[1] 'Μηχατρονική', *Βικιπαίδεια*, 9 Φεβρουάριος 2016, https://el.wikipedia.org/w/index.php?title=%CE%9C%CE%B7%CF%87%CE%B1%CF%84%CF%81%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE&oldid=5676110.

1.2 Ιστορία

Η Μηχατρονική επικεντρώνεται στη μηχανική, την ηλεκτρονική, την μηχανολογία συστημάτων ελέγχου, των ηλεκτρονικών υπολογιστών, τη μοριακή μηχανική (από νανοχημεία και Βιολογία), η οποία και σε συνδυασμό μεταξύ των, να καταστήσει την παραγωγή:

- απλούστερη,
- πιο οικονομική,
- αξιόπιστη και ευέλικτη.

Ο όρος "Μηχανοτρονική" επινοήθηκε για πρώτη φορά από τον Tetsuro Mori, ανώτερο μηχανικό της ιαπωνικής εταιρείας Yaskawa, το 1969. Η Μηχατρονική εναλλακτικά, μπορεί να αναφέρεται και ως η Επιστήμη των "Ηλεκτρομηχανολογικών συστημάτων" ή λιγότερο συχνά ως η Επιστήμη "ελέγχου και του αυτοματισμού της μηχανικής".

Το 1982 επιτρέπεται από την εταιρεία η ελεύθερη χρήση του όρου.

Η Μηχατρονική αποτελεί το άμεσο εκείνο υπόβαθρο για την έρευνα στο τεχνικό τομέα της Κυβερνητικής. Σημαντικές φυσιογνωμίες και χρονολογίες σταθμοί στην Κυβερνητική και κατ' επέκταση στην Μηχατρονική υπήρξαν το 1936 από τον Άλαν Τούρινκ το 1948 από τον Νόρμπερτ Βίνερ και Μόρθυ (Morthy), με τις μηχανές ψηφιακού ελέγχου, που αρχικά αναπτύχθηκαν το 1946 ο Τηλεχειρισμός το 1951 από τον Γκερτζ (Goertz) καθώς και η ανώνυμη εταιρεία Bedford Associates που αναπτύχθηκε το 1968.

1.3 Ορισμοί Μηχατρονικής

Επί του παρόντος υπάρχουν διάφοροι ορισμοί της Μηχατρονικής, ανάλογα με την περιοχή ενδιαφέροντος. Ειδικότερα, η UNESCO ορίζει για την Μηχατρονική ότι είναι:

"Η συνεργιακή ολοκλήρωση της μηχανολογίας με την ηλεκτρονική και τον ευφυή υπολογιστή ελέγχου στον σχεδιασμό και την κατασκευή των προϊόντων και διαδικασιών."

Ωστόσο ένας πιο ενδιαφέρων ορισμός είναι ότι Μηχατρονική είναι: **"Η Μελέτη και κατασκευή των ευφυών μηχανικών συστημάτων."**

Κάτω από αυτή τη θεώρηση, η Μηχατρονική μπορεί να ερμηνευθεί ως **"Η εφαρμογή πολύπλοκης διαδικασίας λήψης αποφάσεων κατά τη λειτουργία φυσικών συστημάτων."**

1.4 Αντικείμενο

Η Μηχατρονική όπως προαναφέρθηκε, πρόκειται να συγχωνεύσει τις πιο πάνω επιστήμες και να περιγράψει αντί διάφορων προτύπων ένα γενικό ολιστικό Μηχατρονικό σύστημα.

Τα συστήματα της Μηχατρονικής έχουν το στόχο να μετατρέψουν με την τεχνολογία που τα διέπει:

- Επεξεργαστές
- Ενεργοποιητές
- Αισθητήρες κτλ

την μορφή της ενέργειας αλλά και των υλικών, την μεταφορά τους και την περαιτέρω επεξεργασία τους καθώς και τη μεταφορά ή/και αποθήκευση των πληροφοριών.

1.4.1 Μηχατρονικό Σύστημα

Ένα Μηχατρονικό σύστημα αποτελείται κυρίως από μηχανισμούς

- Κίνησης,
- Ελέγχου και
- Αισθητήρες.

Η παραδοσιακή Μηχανική αποτελείται μόνο από μηχανισμούς και ενεργοποιητές, και προαιρετικά μπορεί να ενσωματωθεί ο έλεγχος. Η Μηχατρονική ενσωματώνει όλες τις απαιτούμενες προϋποθέσεις για έλεγχο κλειστού βρόχου και ως εκ τούτου και τους ανάλογους αισθητήρες

Ένα Μηχατρονικό σύστημα είναι ένα σύστημα το οποίο ενσωματώνει την ψηφιακή επεξεργασία σήματος και την έκδοση του σήματος αυτού σε ένα τελικό σημείο δράσης μέσω ενός ενεργοποιητή, δημιουργώντας κινήσεις ή ενέργειες σχετικά με το σύστημα. Είναι ένα ολοκληρωμένο σύστημα με αισθητήρες, μικροεπεξεργαστές και ελεγκτές.

Τα συστήματα Μηχατρονικής μπορούν να διαιρεθούν έτσι σε ομάδες λειτουργίας, να διαμορφωθούν σε εκείνους τους βρόχους αυτόματου ελέγχου και να αποτελέσουν μέρος των ενοτήτων με τα μηχανικά - ηλεκτρικά - μαγνητικά - θερμικά - οπτικά στοιχεία τους και την τεχνολογία αισθητήρων, με σκοπό τη συλλογή των μετρημένων μεταβλητών της επιβλέπουσας κατάστασης, την ενεργοποίηση την κανονικοποίηση και τον έλεγχο καθώς επίσης και επεξεργασία και την πληροφορική στην επεξεργασία δεδομένων.

1.4.2 Παραδείγματα Μηχατρονικών Συστημάτων

- Χειρισμός/συστήματα ρομπότ
- Ενότητες εργαλειομηχανών
- Ψηφιακές φωτογραφικές μηχανές
- Κίνηση και έλεγχος φορέα CD/DVD Player
- Ανεμογεννήτριες
- Αντιολισθητικά συστήματα
- Ηλεκτρονικά προγράμματα σταθερότητας οχημάτων

Ειδικά για τα δύο τελευταία παραδείγματα, τα ηλεκτρονικά και το λογισμικό αντικαθιστούν τα μικρότερης ακρίβειας, πιο ευπαθή και πιο ακριβά αναλογικά μηχανικά συστήματα με ψηφιακό ηλεκτρονικό έλεγχο, όπως τα συστήματα αντιεμπλοκής πέδησης

(ABS) τα συστήματα ελέγχου μείγματος, προπορείας, σπινθιρισμού (ECU) και τα συστήματα ελέγχου ολίσθησης (ASP/ESP) στα αυτοκίνητα. Παγκοσμίως η μηχατρονική είναι αντικείμενο ειδίκευσης μηχανολόγων ή μηχανικών παραγωγής.

1.5 Εφαρμογές

Η απλούστερη εφαρμογή αφορά στην δυναμική ανάλυση ενός μηχανικού συστήματος και τον (ενεργό, ημί - ενεργό ή παθητικό) έλεγχό του.

Οι πιο σημαντικές εφαρμογές της Μηχατρονικής είναι η ρομποτική, τα συστήματα μεταφορών, συστήματα παραγωγής, μηχανές CNC, και οι βιομηχατρονικές νανομηχανές. Η τελειότερη όμως εφαρμογή της Μηχατρονικής είναι το Ρομπότ.

Η Ρομποτική είναι κλάδος της Μηχατρονικής.

Ρομποτική είναι η τέχνη του σχεδιασμού και της κατασκευής επαναπρογραμματιζομένων στοιχείων - συσκευών ευέλικτων και ικανών να εκτελούν διάφορες λειτουργίες. Το επίπεδο του αυτοματισμού είναι πολύ πιο ευέλικτο και δείχνει τις μελλοντικές τάσεις στην υπόλοιπη μηχατρονική.

Η εφαρμογή των μηχατρονική στη μεταφορά λαμβάνει χώρα κατά το σχεδιασμό των ενεργητικών μηχανισμούς (π.χ. ενεργός αναστολή), τους κραδασμούς ελέγχου, μηχανισμούς σταθεροποίησης και αυτόνομη πλοήγηση.

Στην κατασκευή, η Μηχατρονική έχει χρησιμοποιηθεί για μοντέλα διακριτών κατά περίπτωση συστημάτων και έχει υποβάλει αίτηση για το βέλτιστο σχεδιασμό των γραμμών παραγωγής, καθώς και τη βελτιστοποίηση των υφιστάμενων διαδικασιών. Επίσης, έχει συμβάλει στην αυτοματοποίηση των γραμμών παραγωγής και τη δημιουργία της έννοιας της ευέλικτης κατασκευής.

Μηχατρονική είναι η ιστορία του ψηφιακού ελέγχου μηχανών. Σε αυτό το θέμα τις τελευταίες εξελίξεις είναι οι εξής: της ανάλυσης, ανίχνευσης και ελέγχου των κραδασμών και της θερμοκρασίας στην εργαλεία κοπής, των μεθόδων διάγνωσης και εργαλεία κοπής ταχεία προτυποποίηση, EDM λέιζερ και σύνθεση.

Στο πεδίο αυτό γίνεται σύντομη εισαγωγή στην προσομοίωση δυναμικών μηχανικών συστημάτων, στον έλεγχο κατασκευών και ιδιαίτερα στην χρήση μεθόδων Ανάλυσης και Μοντελοποίησης:

- ευφυούς ελέγχου και συγκεκριμένα, σε μεθόδους οι οποίες στηρίζονται σε
- ασαφή λογική,
- νευρωνικά δίκτυα
- συναφείς υβριδικές τεχνικές.
- γενετικούς αλγορίθμους

Η βασική γνώση της τεχνικής δυναμικής καθώς και η δυνατότητα τουλάχιστον χρήσης ηλεκτρονικού υπολογιστή θεωρούνται αναγκαία .

Καθώς τα συστήματα αυτά σπάνια πληρούν τις προϋποθέσεις μιας μελέτης , το μαθηματικό μοντέλο που χρησιμοποιείται είναι πολύπλοκο (μη-γραμμικό), έχει ατέλειες κτλ. Για αυτό και χρησιμοποιούμε τα προαναφερθέντα ευφυή συστήματα ελέγχου.

1.5.1 Γενικότερες εφαρμογές

- Αυτοματισμού, και στον τομέα της ρομποτικής
- Σερβοϋδραυλική μηχανική
- Αισθητήρες και συστήματα ελέγχου
- Αυτοκίνητο Βιομηχανίες, στη σχεδίαση των υποσυστημάτων, όπως η αντι-εμπλοκή κατά την πέδηση
- Μηχανικών Ηλεκτρονικών Υπολογιστών, του σχεδιασμού μηχανισμών, όπως οι οδηγοί δίσκων

1.5.2 Παραλλαγές Μηχατρονικής

Μια παραλλαγή του αναδυόμενου αυτού τομέα είναι η biomechatronics - Βιομηχατρονική, σκοπός της οποίας είναι η ενσωμάτωση μηχανικών μέρών με ένα ανθρώπινο ον, συνήθως με τη μορφή των αποσπώμενων συσκευών όπως exoskeleton. Αυτή είναι η "πραγματική ζωή" έκδοση του cyberware.

Η Βιομηχατρονική είναι η εφαρμογή της μηχατρονικής για την επίλυση των προβλημάτων των βιολογικών συστημάτων, και ιδίως την ανάπτυξη νέων τύπων προθέσεων, χειρουργικών προσομοιωτών, τον έλεγχο της θέσης των ιατρικών πράξεων (π.χ. καθετήρες), αναπηρικές πολυθρόνες και χειρουργικές τηλεχειρισμός

Επίσης η νανομηχανική είναι ένας τομέας που έχει επωφεληθεί από τις εξελίξεις στη μηχατρονική.

Κεφάλαιο 2

Οδομετρία

[2]Οδομετρία είναι η χρήση των δεδομένων από τα αισθητήρια κίνησης για την εκτίμηση της μετατόπισης. Χρησιμοποιείται στην ρομποτική για την εκτίμηση της θέσης του ρομπότ σε σχέση με το σημείο εκκίνησης. Η μέθοδος αυτή είναι ευαίσθητη σε σφάλματα λόγω της ενσωμάτωσης των μετρήσεων ταχύτητας συναρτήσει του χρόνου για να δώσει εκτίμηση της θέσης. Γρήγορη και ακριβής συλλογή δεδομένων, βαθμονόμηση αισθητηρίων, καθώς και επεξεργασία χρειάζεται στις περισσότερες περιπτώσεις για να χρησιμοποιηθεί αποτελεσματικά η οδομετρία.

Μερικές από τις μεθόδους οδομετρίας είναι οι εξής:

- Περιστροφική κωδικοποίηση
- Αδρανειακή οδομετρία
- Αστρονομική πλοήγηση
- Δορυφορική πλοήγηση (GPS)
- Οπτική οδομετρία

Η καθεμία από τις παραπάνω μεθόδους έχει χρήσεις σε πολλούς τομείς και το καθένα έχει τις ιδιαιτερότητές της και χρησιμοποιούνται αντίστοιχα. Πιο αναλυτικά:

[2] 'Odometry', *Wikipedia*, 9 Μάρτιος 2017, https://en.wikipedia.org/w/index.php?title=Odometry&oldid=769409564.

2.1 Περιστροφική κωδικοποίηση

[3]Ο αισθητήρας γωνίας περιστροφής είναι μια ηλεκτρομηχανική συσκευή που μετατρέπει την γωνιακή θέση ή την κίνηση του άξονα περιστροφής του σε αναλογικό η ψηφιακό σήμα. Υπάρχουν δύο κυρίως είδη αισθητήρων περιστροφής: οι σχετικοί και οι απόλυτοι. Η έξοδος του απόλυτου αισθητήρα είναι η τρέχουσα θέση του άξονα περιστροφής, κάνοντας τον γωνιακό μορφοτροπέα. Η έξοδος του σχετικού αισθητήρα περιστροφής παρέχει πληροφορίες για την κίνηση του άξονα, η οποία συνήθως μετατρέπεται σε πληροφορίες όπως: ταχύτητα, απόσταση και θέση.

2.1.1 Τεχνολογίες αισθητήρων

- **Αγώγιμοι**: Αποτελούνται από σειρές περιφερειακών χάλκινών διαδρόμων χαραγμένων επάνω σε ένα PCB. Αγώγιμες βούρτσες αντιλαμβάνονται τους διαδρόμους. Οι αισθητήρες αυτοί χρησιμοποιούνται σπάνια λόγω μηχανικής φθοράς.
- **Οπτικοί**: Στους αισθητήρες αυτούς χρησιμοποιείται φως που πέφτει σε μια φωτοδίοδο μέσα από σχισμές σε μεταλλικό ή γυάλινο δίσκο. Υπάρχουν επίσης και εκδοχές που λειτουργούν με την ανάκλαση του φωτός από δίσκο με λευκές και μαύρες περιοχές. Ο τύπος αυτός είναι ο πιο διαδεδομένος. Οι οπτικοί αισθητήρες είναι ιδιαίτερα ευαίσθητοι στην σκόνη.
- **Μαγνητικοί επί του άξονα**: Σε αυτούς τους αισθητήρες χρησιμοποιείτε ένας διπολικός μαγνητισμένος δίσκος νεοδυμίου του ίδιου μεγέθους με τον άξονα. Η ακρίβεια είναι κακή και δεν επιτρέπει πολλές επιλογές ανάλυσης. Λόγω του διπολικού μαγνήτη υπάρχει τρέμουλο στην έξοδο λόγο εσωτερικής παρεμβολής.
- **Μαγνητικοί εκτός άξονα**: Αυτή η τεχνολογία χρησιμοποιεί μαγνήτες από φερρίτη στερεωμένους με καουτσούκ πάνω σε άξονα. Αυτό προσφέρει μεγάλη ευελιξία στον σχεδιασμό. Τα μικροτσίπ της κωδικοποίησης μπορούν να προγραμματιστούν για οποιονδήποτε αριθμό πόλων καθώς και να τοποθετηθούν σε οποιαδήποτε θέση χρειάζεται για την εφαρμογή.

[3] 'Rotary Encoder', *Wikipedia*, 14 Μάρτιος 2017, https://en.wikipedia.org/w/index.php?title=Rotary_encoder&oldid=770341895.

2.1.2 Εφαρμογές περιστροφικής κωδικοποίησης

Οι περιστροφικοί αισθητήρες χρησιμοποιούνται σε πολλές εφαρμογές που απαιτούν ατέρμονη περιστροφή του άξονα και ακριβείς μετρήσεις όπως:

- Βιομηχανικά συστήματα ελέγχου
- Ρομποτική
- Φωτογραφικούς φακούς
- Συσκευές εισαγωγής Η/Υ (Ποντίκια)
- Περιστρεφόμενες πλατφόρμες Radar

Μια από τις σημαντικότερες εφαρμογές τις περιστροφικής κωδικοποίησης είναι στα όργανα των αυτοκινήτων όπου χρησιμοποιείται για την μέτρηση της ταχύτητας, οδομέτρηση και άλλους πιο σύνθετους υπολογισμούς.

Σε δίτροχα ρομπότ καθώς και σε ερπυστριοφόρα οχήματα η τεχνολογία αυτή μπορεί να χρησιμοποιηθεί και σε πιο σύνθετους υπολογισμούς. Αν τοποθετηθούν στον κάθε ένα από τους δύο τροχούς από έναν περιστροφικό κωδικοποιητή μπορούμε να υπολογίσουμε την διαφορά της ταχύτητας και με τον τρόπο αυτόν την γωνία του οχήματος, την κατεύθυνση καθώς και τις συντεταγμένες σε σχέση με την αρχική θέση. Η σημαντική αδυναμία αυτής της μεθόδου είναι η ολίσθηση των τροχών (ερπυστριών) κατά τις στροφές του οχήματος.

2.2 Αδρανειακή οδομετρία

[4]Το αδρανειακό σύστημα προσανατολισμού χρησιμοποιεί υπολογιστή, αισθητήρες κίνησης (αξελερόμετρα) και αισθητήρες περιστροφής (γυροσκόπια) για τον διαρκή υπολογισμό της θέσης, του προσανατολισμού και της ταχύτητας ενός κινούμενου αντικειμένου χωρίς την χρήση εξωτερικών σημείων αναφοράς. Χρησιμοποιείτε σε οχήματα όπως: πλοία, αεροπλάνα, καθοδηγούμενους πυραύλους και διαστημόπλοια.

[4] 'Inertial Navigation System', *Wikipedia*, 12 Απρίλιος 2017, https://en.wikipedia.org/w/index.php?title=Inertial_navigation_system&oldid=775031862.

2.2.1 Επισκόπηση

Ο αδρανειακός προσανατολισμός είναι μια αυτάρκης τεχνική με την οποία χρησιμοποιούνται μετρήσεις από τα γυροσκόπια και αξελερόμετρα για την παρακολούθηση της θέσης και του προσανατολισμού ενός αντικειμένου σε σχέση με ένα σημείο εκκίνησης. Οι αδρανειακές μονάδες μέτρησης συνήθως περιέχουν τρία γυροσκόπια και τρία αξελερόμετρα τα οποία μετρούν την γωνιακή ταχύτητα και την ευθύγραμμη επιτάχυνση αντίστοιχα. Με την επεξεργασία των μετρήσεων από τις συσκευές αυτές μπορούμε να παρακολουθήσουμε την θέση και τον προσανατολισμό της συσκευής.
Ο αδρανειακός προσανατολισμός χρησιμοποιείται σε ευρύ φάσμα εφαρμογών όπως: πλοήγηση αεροσκαφών, τακτικούς και στρατηγικούς πυραύλους, διαστημόπλοια, υποβρύχια και πλοία. Η πρόσφατη πρόοδος στην κατασκευή των μικροελεκτρομηχανικών συστημάτων (MEMS) έκανε δυνατό να κατασκευαστούν μικρές και ελαφριές μονάδες που μπορούν να χρησιμοποιηθούν σε ποίκιλες εφαρμογές όπως για παράδειγμα η καταγραφή της κίνησης ανθρώπων ή ζώων.
Ένα σύστημα αδρανειακού προσανατολισμού περιέχει τουλάχιστον έναν υπολογιστή και μια πλατφόρμα που εμπεριέχει αξελερόμετρα, γυροσκόπια ή άλλους αισθητήρες κίνησης. Στο σύστημα αρχικά παρέχεται η θέση, η ταχύτητα και ο προσανατολισμός του από μια διαφορετική πηγή (χειροκίνητα από άνθρωπο, από σύστημα GPS κλπ.) και ύστερα υπολογίζει την ανανεωμένη θέση και ταχύτητά του με την ενσωμάτωση των μετρήσεων από τα αισθητήρια κίνησης. Το πλεονέκτημα του συστήματος είναι ότι δεν χρειάζεται εξωτερικές αναφορές για να καθορίσει την θέση, τον προσανατολισμό και την ταχύτητά του από την στιγμή που θα αρχικοποιηθεί.
Τα συστήματα αδρανειακού προσανατολισμού μπορούν να αντιληφθούν την αλλαγή της γεωγραφικής τους θέσης (όπως μια κίνηση προς τον βορά ή την ανατολή πχ.), αλλαγή στην ταχύτητα τους και την αλλαγή του προσανατολισμού τους (περιστροφή γύρο από άξονα). Αυτό γίνεται με την μέτρηση της γραμμικής επιτάχυνσης και της γωνιακής ταχύτητάς τους. Αφού δεν χρειάζεται εξωτερικά σημεία αναφοράς (μετά την αρχικοποίηση) είναι προστατευμένα από θόρυβο και εξωτερικές παρεμβολές.

2.2.2 Σφάλμα

Όλα τα συστήματα αδρανειακού προσανατολισμού πάσχουν από το σωρευτικό σφάλμα: μικρά σφάλματα στην μέτρηση της επιτάχυνσης και γωνιακής ταχύτητας συσσωρεύονται σταδιακά σε μεγαλύτερα σφάλματα στην ταχύτητα, τα οποία μετατρέπονται σε ακόμα μεγαλύτερα σφάλματα στην θέση. Αφού η νέα θέση υπολογίζεται από την προηγούμενη υπολογισμένη θέση και την μετρημένη επιτάχυνση και γωνιακή ταχύτητα, τα συστήματα αυτά τα σφάλματα συσσωρεύονται περίπου αναλογικά με τον χρόνο που πέρασε από την αρχικοποίηση της θέσης. Για την αποτελεσματική χρήση των συστημάτων αδρανειακού προσανατολισμού χρειάζονται αισθητήρες εξαιρετικής ακρίβειας και κόστους βαθμονομημένοι με σχολαστικές διαδικασίες ώστε το σωρευτικό σφάλμα να μην υπερβαίνει το επιθυμητό ή να χρησιμοποιούνται σε συνδυασμό με άλλα συστήματα προσανατολισμού όπως GPS, αστρονομική πλοήγηση, οπτική οδομετρία κλπ.

2.3 Αστρονομική πλοήγηση

[5]Η αστρονομική πλοήγηση είναι μια αρχαία επιστήμη που επιτρέπει στον πλοηγό να κινείται μέσα στον χώρο χωρίς να βασίζεται σε εκτιμημένους υπολογισμούς, dead reckoning, για να γνωρίζει (το γεωγραφικό πλάτος) την θέση του. Η αστρονομική πλοήγηση χρησιμοποιεί σημεία αναφοράς ή γωνιακέ μετρήσεις που έχουν παρθεί μεταξύ ουράνιων σωμάτων (τον ήλιο, την σελήνη, πλανήτη ή άστρο) και του ορατού ορίζοντα. Συχνότερα από όλα χρησιμοποιείται ο ήλιος, αλλά μπορούν ακόμα να χρησιμοποιηθούν η σελήνη, κάποιος πλανήτης, ο βόρειος πολικός αστέρας ή άλλοι 57 αστέρες των οποίων οι συντεταγμένες να είναι καταγεγραμμένοι στους πίνακες της αστρονομικής εφημερίδας.

Η αστρονομική πλοήγηση χρησιμοποιεί γωνιακές μετρήσεις μεταξύ των ουράνιων σωμάτων και του ορατού ορίζοντα για να εντοπίσουν την θέση κάποιου επάνω στην γη ή στην θάλασσα. Σε δεδομένη στιγμή, κάθε ουράνιο σώμα βρίσκεται πάνω από ένα σημείο στην επιφάνεια της γης. Το γεωγραφικό μήκος του σημείου αυτού είναι γνωστό ως η γεωγραφική θέση του ουράνιου σώματος (γεωγραφική θέση), η θέση του οποίου μπορεί να προσδιοριστεί

[5] 'Celestial Navigation', *Wikipedia*, 21 Απρίλιος 2017, https://en.wikipedia.org/w/index.php?title=Celestial_navigation&oldid=776561117.

από τους (ετήσια παρατήρηση των αστεριών σε εκείνη την τοποθεσία) πίνακες της αστρονομικής εφημερίδας για το συγκεκριμένο έτος.

Η μετρημένη γωνία μεταξύ του ουράνιου σώματος και του ορατού ορίζοντα σχετίζεται άμεσα με την απόσταση μεταξύ της γεωγραφικής θέσης του ουράνιου σώματος και της θέσης του παρατηρητή. Μετά από ορισμένους υπολογισμούς αναφερόμενους ως sight reduction αυτή η μέτρηση χρησιμοποιείται για να σχεδιάσει μια γραμμή θέσης πάνω σε ναυτικό χάρτη ή την σχεδίαση διαγράμματος πλοήγησης, ο παρατηρητής βρίσκεται κάπου επάνω σε αυτήν τη γραμμή (Η γραμμή της θέσης είναι ένας μικρός τομέας ενός πολύ μεγάλου κύκλου επάνω στην γη ο οποίος κυκλώνει την γεωγραφική θέση του παρατηρούμενου ουράνιου σώματος. Ένας παρατηρητής που βρίσκεται οπουδήποτε πάνω στην περιφέρεια του κύκλου επάνω στην γη, μετρώντας την γωνία του ίδιου ουράνιου σώματος με τον ορίζοντα την ίδια ακριβώς στιγμή θα παρατηρήσει αυτό το σώμα να σχηματίζει την ίδια γωνία με τον ορίζοντα.) Η μέτρηση της γωνίας δυο ουράνιων σωμάτων μας δίνει δύο τέτοιες γραμμές στον χάρτη που τέμνονται πάνω στην θέση του παρατηρητή (στην πραγματικότητα, οι δυο κύκλοι μας δίνουν δύο σημεία διασταύρωσης, αλλά η μία είναι πολύ εύκολο να απορριφθεί αφού βρίσκεται πολύ μακριά από το εκτιμητέο σημείο.) Οι περισσότεροι πλοηγοί χρησιμοποιούν από τρία μέχρι και πέντε άστρα, αν είναι διαθέσιμα, σε αυτήν τη περίπτωση η διασταύρωση των κύκλων θα είναι μόνο μια και μειώνεται η πιθανότητα σφάλματος.

2.3.1 Σύγχρονη αστρονομική πλοήγηση

Ενώ η αστρονομική πλοήγηση γίνεται ολοένα και πιο περιττή λόγω της εμφάνισης φθηνών και υψηλής ακρίβειας δορυφορικών δεκτών πλοήγησης (GPS), χρησιμοποιήθηκε εκτενώς στην αεροπορία μέχρι τη δεκαετία του 1960 και στη ναυτιλιακή πλοήγηση μέχρι πρόσφατα. Αλλά δεδομένου ότι ένας συνετός ναυτικός δεν βασίζεται σε κανένα μόνο μέσο για τον καθορισμό της θέσης του, πολλές εθνικές ναυτιλιακές αρχές εξακολουθούν να απαιτούν από τους αξιωματικούς του καταστρώματος να επιδεικνύουν γνώση της αστρονομικής ναυσιπλοΐας στις εξετάσεις, πρωτίστως ως εφεδρική μέθοδο πλοήγησης. Μία από τις συνηθέστερες τρέχουσες χρήσεις της ουράνιας πλοήγησης στα μεγάλα εμπορικά πλοία είναι η βαθμονόμηση της πυξίδας και ο έλεγχος σφαλμάτων στη θάλασσα όταν δεν υπάρχουν διαθέσιμες επίγειες αναφορές.

Ήδη από τα μέσα της δεκαετίας του 1960, τα προηγμένα ηλεκτρονικά συστήματα και τα υπολογιστικά συστήματα εξελίχθηκαν επιτρέποντας στους πλοηγούς να αποκτήσουν αυτοματοποιημένη αστρονομική πλοήγηση. Αυτά τα συστήματα χρησιμοποιήθηκαν τόσο σε πλοία όσο και σε αεροσκάφη των ΗΠΑ, και ήταν εξαιρετικά ακριβή, ικανά να κλειδώνουν έως και 11 αστέρια (ακόμη και τη διάρκεια της ημέρας) και να υπολογίζουν τη θέση του σκάφους με ακρίβεια 91 μέτρων. Το αεροσκάφος αναγνώρισης υψηλής ταχύτητας SR-71 ήταν ένα παράδειγμα ενός αεροσκάφους που χρησιμοποίησε ένα συνδυασμό αυτοματοποιημένης ουράνιας και αδρανειακής πλοήγησης. Ωστόσο, αυτά τα σπάνια συστήματα ήταν ακριβά και οι λίγοι που εξακολουθούν να χρησιμοποιούνται σήμερα θεωρούνται ως εφεδρικά για πιο αξιόπιστα συστήματα εντοπισμού δορυφόρων.

Οι διηπειρωτικοί βαλλιστικοί πύραυλοι χρησιμοποιούν την αστρονομική πλοήγηση για να ελέγξουν και να διορθώσουν την πορεία τους (αρχικά με τη χρήση εσωτερικών γυροσκοπίων) ενώ βρίσκονται έξω από την ατμόσφαιρα της Γης. Η ανοσία στα σήματα παρεμβολής είναι ο κύριος μοχλός πίσω από αυτή την προφανώς αρχαϊκή τεχνική.

2.4 Δορυφορική πλοήγηση

[6]Το GPS (Global Positioning System), Παγκόσμιο Σύστημα Στιγματοθέτησης, ή Θεσιθεσίας είναι παγκόσμιο σύστημα εντοπισμού γεωγραφικής θέσης, (στίγματος), ακίνητου ή κινούμενου χρήστη, το οποίο βασίζεται σε ένα "πλέγμα" εικοσιτεσσάρων δορυφόρων της Γης, εφοδιασμένων με ειδικές συσκευές εντοπισμού, οι οποίες ονομάζονται "πομποδέκτες GPS". Οι πομποδέκτες αυτοί παρέχουν ακριβείς πληροφορίες για τη θέση ενός σημείου, το υψόμετρό του, την ταχύτητα και την κατεύθυνση της κίνησης του. Επίσης, σε συνδυασμό με ειδικό λογισμικό χαρτογράφησης μπορούν να απεικονίσουν γραφικά τις πληροφορίες αυτές.

Το σύστημα ξεκίνησε από το Υπουργείο Άμυνας των ΗΠΑ και ονομάστηκε NAVSTAR GPS (Navigation Signal Timing and Ranging Global Positioning System). Το δορυφορικό αυτό σύστημα ρυθμίζεται καθημερινά από τη Βάση Πολεμικής Αεροπορίας Σρίβερ (Schriever) με κόστος 400 εκατομμύρια δολάρια το χρόνο.

[6] 'Global Positioning System', *Βικιπαίδεια*, 30 Νοέμβριος 2016, https://el.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=6133101.

2.4.1 Λειτουργικά τμήματα

Το σύστημα εντοπισμού θέσης GPS σχηματίζει ένα παγκόσμιο δίκτυο, με εμβέλεια που καλύπτει ξηρά, θάλασσα και αέρα. Εξαιτίας αυτής της έκτασής του, είναι απαραίτητος ο διαχωρισμός του σε επιμέρους τμήματα όπου πραγματοποιούνται όλες οι λειτουργίες του αλλά και ο συντονισμός του. Αναλυτικά, τα τμήματα αυτά είναι:

1. Διαστημικό τμήμα: Αποτελείται από το δίκτυο των 24 - 32 δορυφόρων που ήδη αναφέραμε. Οι δορυφόροι αυτοί «σκεπάζουν» ομοιόμορφα με το σήμα τους ολόκληρο τον πλανήτη, γεγονός που αποδεικνύει τη φιλοσοφία που κρύβεται πίσω από τη λειτουργία του συστήματος GPS, δηλαδή τη διαθεσιμότητά του σε κάθε σημείο της Γης, ώστε να μην υπάρχει κίνδυνος να αποπροσανατολιστεί κανείς ποτέ και πουθενά.

Όλοι οι δορυφόροι βρίσκονται σε ύψος 12.552 μιλίων (20.200 χιλιομέτρων) πάνω από την επιφάνεια της θάλασσας και εκτελούν δύο περιστροφές γύρω από τη Γη κάθε 24ωρο. Η κατασκευάστρια εταιρεία είναι η Rockwell International, η εκτόξευσή τους πραγματοποιήθηκε από το ακρωτήριο Canaveral, ενώ η τροφοδοσία τους με ηλεκτρική ενέργεια πραγματοποιείται μέσω των φωτοβολταϊκών συστημάτων που διαθέτουν.

2. Επίγειο τμήμα ελέγχου: Οι δορυφόροι, όπως είναι αναμενόμενο, είναι πολύ πιθανό να αντιμετωπίσουν ανά πάσα στιγμή προβλήματα στη σωστή λειτουργία τους. Οι έλεγχοι που πραγματοποιούνται σε αυτούς αφορούν στη σωστή τους ταχύτητα και υψόμετρο και στην κατάσταση της επάρκειάς τους σε ηλεκτρική ενέργεια. Παράλληλα, εφαρμόζονται όλες οι διορθωτικές ενέργειες που αφορούν στο σύστημα χρονομέτρησης των δορυφόρων, ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος. Το τμήμα επίγειου ελέγχου αποτελείται από ένα επανδρωμένο και τέσσερα μη επανδρωμένα κέντρα, εγκατεστημένα σε ισάριθμες περιοχές του πλανήτη.

Οι περιοχές αυτές είναι οι εξής: α) Κολοράντο (ΗΠΑ) β) Χαβάη (Ανατολικός Ειρηνικός Ωκεανός) γ) Ascension Island (Ατλαντικός Ωκεανός) δ) Diego Garcia (Ινδικός Ωκεανός) ε) Kwajalein (Δυτικός Ειρηνικός Ωκεανός)

Ο κυριότερος σταθμός βάσης είναι αυτός του Κολοράντο, ο οποίος είναι μάλιστα και ο μοναδικός που βρίσκεται στην ξηρά. Αναλαμβάνει τον έλεγχο της σωστής λειτουργίας των εναπομεινάντων τεσσάρων σταθμών, καθώς και τον συντονισμό τους. Σημειώνοντας τη θέση των σταθμών αυτών πάνω σε έναν παγκόσμιο χάρτη, παρατηρεί κανείς ότι η διάταξή τους δεν είναι τυχαία, αλλά ακολουθούν μια γραμμή παράλληλη με τα γεωγραφικά μήκη της Γης.

3. Το τμήμα τελικού χρήστη: Απαρτίζεται από τους χιλιάδες χρήστες δεκτών GPS ανά την υφήλιο. Οι δέκτες αυτοί μπορούν να χρησιμοποιηθούν τόσο κατά τη διάρκεια μιας απλής πεζοπορίας, όσο και σε οχήματα ή θαλάσσια σκάφη και κατά κανόνα διαθέτουν αρκετά μικρές διαστάσεις. Για να προσφέρουν όσο το δυνατόν περισσότερες πληροφορίες, οι δέκτες συνδυάζονται με ειδικό λογισμικό, που προβάλλει ένα χάρτη στην οθόνη της συσκευής GPS. Πρόκειται, δηλαδή, για λογισμικό που λαμβάνει από τους δορυφόρους τις πληροφορίες για το στίγμα του σημείου στο οποίο βρίσκεται ο δέκτης και τις μετατρέπει σε κατανοητή «ανθρώπινη» μορφή, πληροφορώντας το χρήστη για την ακριβή γεωγραφική του θέση.

2.5 Οπτική οδομετρία

[7]«Στη ρομποτική και την όραση του υπολογιστή, η οπτική οδομετρία είναι η διαδικασία προσδιορισμού της θέσης και του προσανατολισμού ενός ρομπότ με την ανάλυση των σχετικών εικόνων κάμερας. Έχει χρησιμοποιηθεί σε μια μεγάλη ποικιλία ρομποτικών εφαρμογών, όπως στο Mars Exploration Rovers.»

2.5.1 Αλγόριθμος

Οι περισσότερες υπάρχουσες προσεγγίσεις στην οπτική οδομετρία βασίζονται στα ακόλουθα στάδια.

1. Απόκτηση εικόνας εισόδου: χρησιμοποιώντας μεμονωμένη κάμερα, στερεοφωνικές κάμερες ή ευρυγώνια κάμερα.

[7] Mark Maimone, Yang Cheng, και Larry Matthies, ‘Two years of Visual Odometry on the Mars Exploration Rovers’, 2007.

2. Διόρθωση εικόνας: εφαρμογή τεχνικής επεξεργασίας εικόνας για απομάκρυνση παραμορφώσεων φακού κ.λπ.
3. Ανίχνευση σημείων παρακολούθησης: ορισμός στοιχείων ενδιαφέροντος και αντιστοιχία σημείων παρακολούθησης ανάμεσα στα καρέ για την κατασκευή του πεδίου οπτικής κύλισης.

    1. Χρήση συσχέτισης για να δημιουργηθεί αντιστοιχία δυο εικόνων αντί για μακροπρόθεσμη παρακολούθηση χαρακτηριστικών
    2. Εξαγωγή σημείων παρακολούθησης και συσχέτισης.
    3. Κατασκευή οπτικού πεδίου κύλισης (μέθοδος Lucas – Kanade).

4. Έλεγχος διανύσματος πεδίου ροής για πιθανά σφάλματα παρακολούθησης και αφαίρεσή τους.

5. Εκτίμηση της κίνησης της κάμερας από την οπτική ροή

    1. Επιλογή 1: Φίλτρο Kalman για συντήρηση κατανομής εκτιμημένης κατάστασης.
    2. Επιλογή 2: Εύρεση γεωμετρικών και τρισδιάστατων ιδιοτήτων των σημείων παρακολούθησης για τη ελαχιστοποίηση του σφάλματος της επαναπροβολής μεταξύ δύο γειτονικών καρέ. Αυτό μπορεί να γίνει με μαθηματική ελαχιστοποίηση ή με τυχαία δειγματοληψία.

6. Περιοδική επαναληπτική λήψη σημείων παρακολούθησης για τη διατήρηση της κάλυψης σε όλη την εικόνα.

Μια εναλλακτική λύση στις μεθόδους βασισμένες σε σημεία παρακολούθησης είναι η «άμεση μέθοδος» που ελαχιστοποιεί το σφάλμα στον χώρο των αισθητηρίων και στη συνέχεια αποφεύγει την αντιστοίχηση και την εξαγωγή των εσφαλμένων σημείων παρακολούθησης.

2.6 Οπτική οδομετρία σε περιβάλλον Mat Lab

[8]Στο παράδειγμα αυτό, έχοντας μια βιντεοκάμερα (ή μια διάταξη από κάμερες) στερεωμένες σε ένα κινούμενο αντικείμενο (όπως ένα αυτοκίνητο ή ένα ρομπότ) να κατασκευαστεί τροχιά κίνησης 6 βαθμών ελευθερίας χρησιμοποιώντας το βίντεο της κάμερας. Σε περίπτωση χρήσης μιας κάμερας αναφερόμαστε σε μονόφθαλμη οπτική οδομετρία. Ενώ όταν χρησιμοποιούμε 2 ή περισσότερες κάμερες μιλάμε για στέρεο οπτική οδομετρία.

2.6.1 Στέρεο και μονόφθαλμη οδομετρία

Υπάρχουν πλεονεκτήματα και μειονεκτήματα σε κάθε μια από τις δυο μεθόδους. Το πλεονέκτημα της στέρεο οδομετρίας είναι ότι μπορεί να εκτιμηθεί η ακριβής τροχιά, ενώ στην μονόφθαλμη μόνο με μια υποκειμενική κλίμακα. Δηλαδή στην μονόφθαλμη μπορούμε να πούμε ότι η μετατόπιση έγινε κατά μια μονάδα στον άξονα **x** και δύο μονάδες στον άξονα **y**, κλπ. Ενώ στην στέρεο οδομετρία ότι έγινε μετατόπιση 1 μέτρου στον άξονα **x** και δύο μέτρα στον άξονα **y**. Επίσης η στέρεο οδομετρία είναι πιο ακριβής (λόγω δεδομένων από δυο κάμερες). Στην περίπτωση όμως που η απόσταση των αντικειμένων που βλέπει η κάμερα είναι πολύ μεγάλη σε σύγκριση με την απόσταση ανάμεσα στις δυο κάμερες ή στέρεο οπτική οδομετρία εκφυλίζεται σε μονόφθαλμη. Έτσι σε ένα πολύ μικρού μεγέθους ρομπότ η μονόφθαλμη είναι η καλύτερη λύση.

2.6.2 Διατύπωση του προβλήματος

Είσοδος

Έχουμε ροή εικόνων που έρχονται από το ζευγάρι των καμερών. Ορίζουμε τις εικόνες που προέρχονται από την αριστερή και την δεξιά και την αριστερή κάμερα σε χρόνο t και t+1:

$I_l^t, I_r^t, I_l^{t+1}, I_r^{t+1}$.

Έχουμε προηγούμενες γνώσεις για όλες τις εγγενείς και εξωγενείς παραμέτρους βαθμονόμησης του στερεοφωνικού εξοπλισμού, που επιτυγχάνονται μέσω ενός από τους πολυάριθμους διαθέσιμους αλγόριθμους στερεοφωνικής βαθμονόμησης.

[8] Avi Singh, 'Visual Odmetry from scratch - A tutorial for beginners', *Avi Singh's blog*, ημερομηνία πρόσβασης 6 Μάιος 2017, https://avisingh599.github.io/vision/visual-odometry-full/.

Έξοδος

Για κάθε ζεύγος στερεοφωνικών εικόνων, πρέπει να βρούμε τον πίνακα περιστροφής **R** και τον φορέα μετάφρασης **t**, ο οποίος περιγράφει την κίνηση του οχήματος μεταξύ των δύο καρέ.

Ο αλγόριθμος

1. Λήψη εικόνων I_l^t , I_r^t, I_l^{t+1}, I_r^{t+1}
2. Καθαρισμός και διόρθωση των εικόνων.
3. Υπολογισμός του χάρτη ανομοιότητας D^t από I_l^t,I_r^t και του χάρτη D^{t+1} από , I_l^{t+1}, I_r^{t+1}
4. Χρήση του αλγόριθμου FAST για τον εντοπισμό σημείων παρακολούθησης σε I_l^t, I_l^{t+1} και ταίριασμά τους.
5. Χρήση των χαρτών ανομοιοτήτων D^t και D^{t+1} για τον υπολογισμό των τρισδιάστατων θέσεων των σημείων παρακολούθησης που εντοπίστηκαν στο προηγούμενο βήμα. Για να πάρουμε σύννεφο δυο σημείων W^t, W^{t+1}
6. Επιλέγουμε ένα υποσύνολο σημείων από το παραπάνω σύννεφο σημείων έτσι ώστε όλες οι αντιστοιχίσεις να είναι αμοιβαία συμβατές.
7. Εκτίμηση του **R, t** από τις τιμές που εντοπίστηκαν στα προηγούμενα βήματα.

2.6.3 Καθαρισμός και διόρθωση

Καθαρισμός: Αυτό το βήμα αντισταθμίζει την παραμόρφωση του φακού. Εκτελείται με τη βοήθεια παραμέτρων παραμόρφωσης που έχουν ληφθεί κατά τη βαθμονόμηση.

Διόρθωση: Αυτό το βήμα εκτελείται για να διευκολύνει το πρόβλημα του υπολογισμού του χάρτη ανισότητας. Μετά από αυτό το βήμα, όλες οι επιπολικές γραμμές γίνονται παράλληλες προς την οριζόντια και το βήμα υπολογισμού της ανισότητας πρέπει να εκτελέσει την αναζήτησή της για αντιστοίχιση μπλοκ μόνο σε μία κατεύθυνση.

Εικόνα 2.1: Στερεοφωνικές εικόνες που έχουν επικαλυφθεί από την βιβλιοθήκη KITTI, παρατηρήστε ότι οι αντιστοιχίσεις σημείων παρακολούθησης είναι κατά μήκος παράλληλων (οριζόντιων) γραμμών.[9]

Οι δύο παραπάνω λειτουργίες είναι ενσωματωμένες στο MATLAB στην βιβλιοθήκη KITTI που είναι μέρος του: Computer Vision Toolbox

2.6.4 Υπολογισμός του χάρτη ανομοιότητας

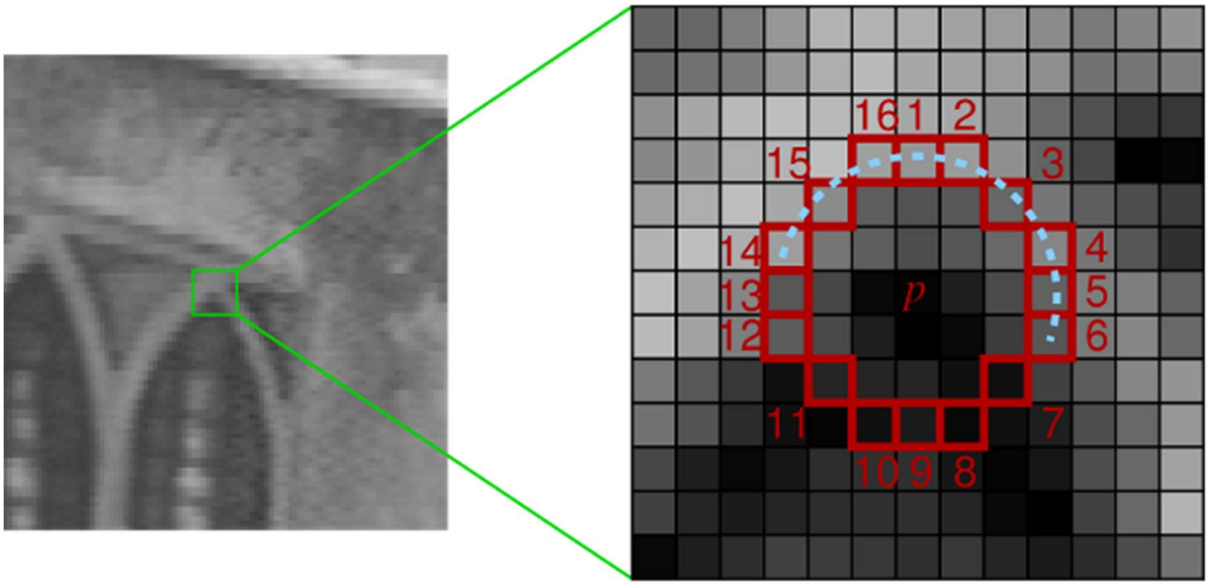
Δεδομένου ενός ζεύγους εικόνων από μια στερεοφωνική κάμερα, μπορούμε να υπολογίσουμε έναν χάρτη ανομοιότητας. Υποθέτουμε ότι ένα συγκεκριμένο σημείο στον τρισδιάστατο χώρο του φυσικού κόσμου F βρίσκεται στη θέση (x, y) στην αριστερή εικόνα και το ίδιο χαρακτηριστικό βρίσκεται στο (x + d, y) στη δεύτερη εικόνα, τότε η θέση (x, y) στο χάρτη ανισότητας έχει την τιμή d. Σημειώνουμε ότι οι συντεταγμένες y είναι οι ίδιες αφού οι εικόνες έχουν διορθωθεί. Έτσι, μπορούμε να ορίσουμε την ανισότητα σε κάθε σημείο του επιπέδου της εικόνας ως: d = xl-xr

Εικόνα 2.2: Ο χάρτης ανομοιότητας στα καρέ από την βιβλιοθήκη KITTI

[9] Στο ίδιο.

2.6.4 Ανεύρεση σημείων παρακολούθησης

Η ανεύρεση σημείων παρακολούθησης πραγματοποιείται με την μέθοδο αναγνώρισης γωνιών FAST. Υποθέτουμε ότι έχουμε ένα σημείο **P** το οποίο ελέγχεται αν είναι γωνία ή όχι. Διαγράφεται κύκλος 16 pixel γύρο του όπως στην εικόνα 2.3. Για κάθε pixel που βρίσκεται στην περιφέρεια αυτού του κύκλου, εξετάζεται εάν υπάρχει ένα συνεχές σύνολο pixels των οποίων η ένταση υπερβαίνει την ένταση του αρχικού pixel κατά έναν ορισμένο παράγοντα **I** και για ένα άλλο σύνολο συνεχόμενων pixels εάν η ένταση είναι μικρότερη τουλάχιστον κατά τον ίδιο παράγοντα **I**. Αν ναι, τότε σημειώνεται αυτό το σημείο ως γωνία. Χρησιμοποιείτε ευρετικός αλγόριθμος για την απόρριψη της συντριπτικής πλειοψηφίας των μη γωνιών, στο οποίο εξετάζεται πρώτα το pixel στο 1, 9, 5, 13 και τουλάχιστον τρεις από αυτές πρέπει να έχουν μια υψηλότερη ένταση τουλάχιστον ίση με **I** ή πρέπει να έχουν μικρότερη ένταση κατά το ίδιο ποσό **I** για να είναι γωνία το σημείο αυτό.



Εικόνα 2.3: Εικόνα από το επίσημο εγχειρίδιο του αλγόριθμου FAST[10]

Ένα άλλο πράγμα που χρειάζεται σε αυτήν την προσέγγιση είναι κάτι που ονομάζεται "bucketing". Εάν τρέξει απλώς ένας ανιχνευτής χαρακτηριστικών σε μια ολόκληρη εικόνα, υπάρχει μια πολύ καλή πιθανότητα ότι τα περισσότερα από τα σημεία παρακολούθησης θα συγκεντρωθούν σε ορισμένες πλούσιες περιοχές της εικόνας, ενώ ορισμένες άλλες περιοχές δεν θα είχαν καμία εκπροσώπηση. Αυτό δεν είναι καλό για τον αλγόριθμο, αφού βασίζεται

[10] Στο ίδιο.

στην παραδοχή μιας στατικής σκηνής και για να βρει την "αληθινή" στατική σκηνή, πρέπει να εξεταστεί όλη η εικόνα, αντί για ορισμένες περιοχές της. Προκειμένου να αντιμετωπιστεί αυτό το ζήτημα, διαιρούμε τις εικόνες σε τμήματα (περίπου 100x100 pixels) και εξάγουμε το πολύ 20 σημεία παρακολούθησης από κάθε ένα από αυτά τα τμήματα, διατηρώντας έτσι μια πιο ομοιόμορφη κατανομή των σημείων παρακολούθησης.

Στον κώδικα αυτό γίνεται με την εξής γραμμή:

```
points1_l = bucketFeatures(I1_l, h, b, h_break, b_break, numCorners);
```

Η οποία καλεί την εξής συνάρτηση:

```
function points = bucketFeatures(I, h, b, h_break, b_break, numCorners)
% input image I should be grayscale


y = floor(linspace(1, h - h/h_break, h_break));
x = floor(linspace(1, b - b/b_break, b_break));


final_points = [];
for i=1:length(y)
    for j=1:length(x)
    roi =   [x(j),y(i),floor(b/b_break),floor(h/h_break)];
    corners  =  detectFASTFeatures(I,  'MinQuality',  0.00,  'MinContrast',  0.1,
'ROI',roi );
    corners = corners.selectStrongest(numCorners);
    final_points = vertcat(final_points, corners.Location);
    end
end
points = cornerPoints(final_points);
```

Όπως φαίνεται παραπάνω η εικόνα χωρίζεται σε τμήματα και οι «δυνατότερες» γωνίες κάθε τμήματος επιλέγονται για τα μετέπειτα βήματα.

2.6.5 Περιγραφή και ταίριασμα σημείων παρακολούθησης

Οι γωνίες του αλγόριθμου FAST που εντοπίστηκαν στο προηγούμενο βήμα τροφοδοτούνται στο επόμενο βήμα, το οποίο χρησιμοποιεί έναν KLT tracker. Ο KLT tracker κοιτάζει ουσιαστικά γύρω από κάθε γωνιά που παρακολουθεί και χρησιμοποιεί αυτές τις τοπικές πληροφορίες για να βρει τη γωνία στην επόμενη εικόνα. Οι γωνίες που βρέθηκαν στην I_l^t ανιχνεύονται στην I_l^{t+1}. Θα ορίσουμε τις γωνίες που βρέθηκαν στην I_l^t ως: F^t, και τις αντίστοιχες στην I_l^{t+1} ως: F^{t+1}.

Στο MATLAB αυτό γίνεται εύκολα με την αρχικοποίηση του tracker και τρέξιμό του μια φορά.

```
tracker = vision.PointTracker('MaxBidirectionalError', 1);
initialize(tracker, points1_l.Location, I1_l);
[points2_l, validity] = step(tracker, I2_l);
```

Τριγωνισμός του τρισδιάστατου σύννεφου σημείων

Οι πραγματικές 3D συντεταγμένες του κάθε σημείου F^t και F^{t+1} υπολογίζονται σε σχέση με την αριστερή κάμερα χρησιμοποιώντας την τιμή ανισότητας που αντιστοιχεί σε αυτά τα σημεία παρακολούθησης από τον χάρτη ανισότητας και τους γνωστούς πίνακες προβολής των δύο καμερών **P**1 και **P**2. Αρχικά διαμορφώνουμε τον πίνακα επαναλήψεων **Q**, χρησιμοποιώντας δεδομένα από τα **P**1 και **P**2:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & -f \\ 0 & 0 & -1/T_x & 0 \end{bmatrix}$$

C_x : συντεταγμένη **x** του οπτικού κέντρου της αριστερής κάμερας (σε pixel)

C_y : συντεταγμένη **y** του οπτικού κέντρου της αριστερής κάμερας (σε pixel)

f : Εστιακή απόσταση της πρώτης κάμερας

T_x : συντεταγμένη **x** της δεξιάς κάμερας σε σχέση με την πρώτη κάμερα (σε μέτρα)

Για την ανεύρεση των 3D συντεταγμένων του κάθε σημείου παρακολούθησης F^t και F^{t+1} χρησιμοποιούμε την παρακάτω σχέση:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Q \times \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}$$

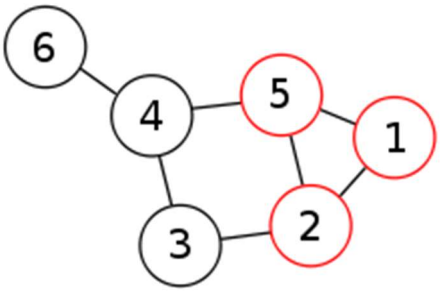
Τα σύννεφα σημείων που υπολογίζονται ορίζονται ως W^t και W^{t+1}.

2.6.6 Αλγόριθμος εντοπισμού κινούμενων σημείων παρακολούθησης

Στο τρέχων παράδειγμα υποθέτουμε ότι το σκηνικό που καταγράφει η κάμερα είναι άκαμπτο που σημαίνει ότι η απόσταση δυο σημείων παρακολούθησης δεν αλλάζει εν συναρτήσει με τον χρόνο **t** και **t + 1**. Ως αποτέλεσμα η απόσταση δυο σημείων στο σύννεφο σημείων W^t πρέπει να είναι ίδια με αυτήν του W^{t+1}. Αν αυτή δεν είναι ίδια τότε ή συνέβη σφάλμα κατά τον 3D τριγωνισμό ή το ότι έγινε τριγωνισμός κινούμενου αντικειμένου το οποίο δεν μπορεί να χρησιμοποιηθεί στο επόμενο βήμα. Για να ταιριάζουν όσο το δυνατόν περισσότερα σημεία μεταξύ τους δημιουργούμε έναν πίνακα **M:**

$$\boldsymbol{M_{i,j}} = \begin{cases} \mathbf{1}, & \boldsymbol{\text{αν η απόσταση μεταξύ των i και j είναι ίδια και στα δυο σύνεφα σημείων}} \\ \mathbf{0}, & \boldsymbol{\text{διαφορετικά}} \end{cases}$$

Από τα αρχικά σύννεφα σημείων θέλουμε τώρα να διαλέξουμε το μεγαλύτερο υποσύνολο του οποίου τα σημεία να ταιριάζουν μεταξύ τους (όλα τα σημεία του πίνακα **M** να είναι 1). Αυτό το πρόβλημα είναι ισοδύναμο με το πρόβλημα της μέγιστης κλίκας, με **M** ως πίνακα γειτνίασης. Οι κλίκες είναι υποσύνολο γραφικής παράστασης που περιέχουν μόνο κόμβους που είναι αλληλοσυνδεμένοι. Ένα εύκολο παράδειγμα είναι η απεικόνιση ενός κοινωνικού δικτύου και το πρόβλημα ανεύρεσης της μεγαλύτερης ομάδας που γνωρίζονται μεταξύ τους.



Έτσι λειτουργεί το πρόβλημα τις μέγιστης κλίκας:

Γνωρίζουμε ότι το πρόβλημα αυτό δεν έχει απόλυτη λύση, αλλά μπορεί να βρεθεί μια ικανοποιητική με την βοήθεια ευρητικού αλγόριθμου.

1. Εύρεση του κόμβου με τις περισσότερες συνδέσεις και ορισμός της κλίκας που περιέχει αυτόν τον κόμβο.
2. Από την υπάρχουσα κλίκα προσδιορισμός του υποσυνόλου των κόμβων **v** που είναι συνδεμένοι με όλους τους κόμβους που υπάρχουν στην κλίκα.
3. Από το σύνολο **v** επιλέγεται ο κόμβος που είναι συνδεμένος με τον μέγιστο αριθμό άλλων κόμβων στο **v.** Επαναλαμβάνεται το βήμα 2 έως ότου δεν μπορούν να συνδεθούν άλλοι κόμβοι στην κλίκα.

Ο κώδικας του παραπάνω αλγόριθμου είναι:

```
function cl = updateClique(potentialNodes, clique, M)



maxNumMatches = 0;
curr_max = 0;
for i = 1:length(potentialNodes)
    if(potentialNodes(i)==1)
        numMatches = 0;
        for j = 1:length(potentialNodes)
            if (potentialNodes(j) & M(i,j))
                numMatches = numMatches + 1;
            end
```

```
        end
        if (numMatches>=maxNumMatches)
            curr_max = i;
            maxNumMatches = numMatches;
        end
    end
end

if (maxNumMatches~=0)
    clique(length(clique)+1) = curr_max;
end

cl = clique;



function newSet = findPotentialNodes(clique, M)

newSet = M(:,clique(1));
if (size(clique)>1)
    for i=2:length(clique)
        newSet = newSet & M(:,clique(i));
    end
end

for i=1:length(clique)
    newSet(clique(i)) = 0;
end
```

2.6.7 Υπολογισμός του R και t

Για τον υπολογισμό του πίνακα περιστροφής **R** και του διανύσματος **t** χρησιμοποιούμε την μη γραμμική μέθοδο ελαχιστοποίησης ελαχίστων τετραγώνων των: Levenberg – Marquardt:

$$\epsilon = \sum_{F^t, F^{t+1}} (j_t - PTw_{t+1})^2 + (j_{t+1} - PT^{-1}w_t)^2$$

$\boldsymbol{F^t}$, $\boldsymbol{F^{t+1}}$: Σημεία παρακολούθησης στην αριστερή εικόνα την χρονική στιγμή **t** και **t+1**

$\mathbf{j_t}$, $\mathbf{j_{t+1}}$: Δισδιάστατες ομογενείς συντεταγμένες των σημείων παρακολούθησης $\boldsymbol{F^t}$, $\boldsymbol{F^{t+1}}$

$\boldsymbol{w^t}$, $\boldsymbol{w^{t+1}}$: Τρισδιάστατες ομογενείς συντεταγμένες των σημείων παρακολούθησης $\boldsymbol{F^t}$, $\boldsymbol{F^{t+1}}$

P: Πίνακας προβολής της αριστερής κάμερας 3 x 4

T: Ομογενής πίνακας μετατροπής

Το Optimization Toolbox στο MATLAB χρησιμοποιεί απευθείας τον αλγόριθμο Levenberg-Marquardt στην συνάρτηση lsqnonlin, που πρέπει να εφοδιαστεί με συνάρτηση διανύσματος που πρέπει να ελαχιστοποιηθεί και ένα σύνολο παραμέτρων που μπορούν να μεταβληθούν.

Αυτό το μέρος του αλγόριθμου χρειάζεται την μεγαλύτερη υπολογιστική ισχύ:

```
function F = minimize(PAR, F1, F2, W1, W2, P1)
r = PAR(1:3);
t = PAR(4:6);
%F1, F2 -> 2d coordinates of features in I1_l, I2_l
%W1, W2 -> 3d coordinates of the features that have been triangulated
%P1, P2 -> Projection matrices for the two cameras
%r, t -> 3x1 vectors, need to be varied for the minimization
F = zeros(2*size(F1,1), 3);
reproj1 = zeros(size(F1,1), 3);
```

```
reproj2 = zeros(size(F1,1), 3);


dcm = angle2dcm( r(1), r(2), r(3), 'ZXZ' );
tran = [ horzcat(dcm, t); [0 0 0 1]];


for k = 1:size(F1,1)
    f1 = F1(k, :)';
    f1(3) = 1;
    w2 = W2(k, :)';
    w2(4) = 1;


    f2 = F2(k, :)';
    f2(3) = 1;
    w1 = W1(k, :)';
    w1(4) = 1;


    f1_repr = P1*(tran)*w2;
    f1_repr = f1_repr/f1_repr(3);
    f2_repr = P1*pinv(tran)*w1;
    f2_repr = f2_repr/f2_repr(3);


    reproj1(k, :) = (f1 - f1_repr);
    reproj2(k, :) = (f2 - f2_repr);
end


F = [reproj1; reproj2];
```

Έλεγχος αποτελέσματος

Για να είναι έγκυρο το ζευγάρι των **R** και **t** θα πρέπει να πληρούνται οι εξής προϋποθέσεις:

1. Ο αριθμός των σημείων παρακολούθησης σε μια κλίκα πρέπει να είναι τουλάχιστον 8.
2. Το σφάλμα επαναπροβολής ∈ να είναι λιγότερο από μια ορισμένη τιμή.

Οι παραπάνω προϋποθέσεις βοηθούν σε περίπτωση δεδομένων με θόρυβο.

2.7 Οπτικό ποντίκι Ηλεκτρονικού υπολογιστή

[11]Η τεχνολογία στην οποία βασίζεται το σύγχρονο οπτικό ποντίκι είναι γνωστή ως ψηφιακή συσχέτιση εικόνας, μια τεχνολογία που πρωτοστάτησε η αμυντική βιομηχανία για την παρακολούθηση στρατιωτικών στόχων. Μια απλή έκδοση δυαδικών εικόνων της συσχέτισης ψηφιακής εικόνας χρησιμοποιήθηκε στο οπτικό ποντίκι Lyon 1980. Τα οπτικά ποντίκια χρησιμοποιούν αισθητήρες εικόνας για να απεικονίσουν τη φυσική υφή σε υλικά όπως το ξύλο, το ύφασμα, τα μαξιλάρια ποντικιού και το Formica. Αυτές οι επιφάνειες, όταν φωτίζονται υπό μια ορισμένη γωνία από μια φωτοδίοδο, ρίχνουν σκιές που μοιάζουν με ένα λοφώδες έδαφος φωτισμένο στο ηλιοβασίλεμα. Οι εικόνες αυτών των επιφανειών συλλαμβάνονται σε συνεχή διαδοχή και συγκρίνονται μεταξύ τους για να καθορίσουν πόσο μακριά έχει μετακινηθεί το ποντίκι.

Τα οπτικά ποντίκια συλλαμβάνουν χίλιες διαδοχικές εικόνες ή περισσότερες ανά δευτερόλεπτο. Ανάλογα με το πόσο γρήγορα κινείται το ποντίκι, κάθε εικόνα θα απέχει από την προηγούμενη από ένα κλάσμα του pixel μέχρι μερικά pixel. Τα οπτικά ποντίκια επεξεργάζονται μαθηματικά αυτές τις εικόνες χρησιμοποιώντας διασταυρούμενη συσχέτιση για να υπολογίσουν πόσο κάθε διαδοχική εικόνα απέχει από την προηγούμενη.

2.8 Οπτική οδομετρία με χρήση οπτικού ποντικιού Η/Υ

Αφού το οπτικό ποντίκι έχει την δυνατότητα να μεταφράσει την κίνηση του σε κίνηση του κέρσορα του Η/Υ μπορεί να μετατραπεί σε εξωγενή μονάδες μέτρησης με την βοήθεια της βαθμονόμησης.

[11] 'Optical Mouse', *Wikipedia*, 18 Μάρτιος 2017, https://en.wikipedia.org/w/index.php?title=Optical_mouse&oldid=770892163.

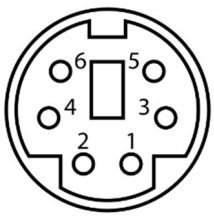
Ο αισθητήρας του οπτικού ποντικιού έχει δυο βαθμούς ελευθερίας δηλαδή την κίνησή του στον κατακόρυφο άξονα y' y και την κίνηση του στον οριζόντιο άξονα x' x με αυτόν τον τρόπο παρακολουθεί την κίνησή του στον δισδιάστατο χώρο. Η χρήση των δεδομένων της μετατόπισης στον άξονα y' y είναι χρήσιμη σε ένα τροχοφόρο ρομπότ (μήχατρον) αφού με αυτό μπορεί να υπολογιστεί η απόσταση που έχει διανύσει καθώς και τα παράγωγα μεγέθη όπως η ταχύτητα και η επιτάχυνση.

2.9 Πείραμα οδομετρίας με χρήση οπτικού αισθητήρα

Στο πείραμα αυτό θα γίνει προσέγγιση της διανυόμενης απόστασης του μήχατρον με την καταγραφή μετρήσεων του οπτικού αισθητήρα από ποντίκι Η/Υ προσαρμοσμένου στο μήχατρον.

2.9.1 Σύνδεση του κυκλώματος ποντικιού σε μικροελεγκτή Arduino

[12]Τα περισσότερα σύγχρονα ποντίκια Η/Υ συνδέονται με σύνδεση USB με την χρήση του πρωτόκολλου HID (Human Interface Device). Οι μικροελεγκτές τύπου Arduino παρόλο που έχουν δυνατότητα σύνδεσης με USB επιτρέπουν την χρήση του μόνο για την μετατροπή του από USB σε σειριακή επικοινωνία και η δίαυλος αυτή χρησιμοποιείται για τον προγραμματισμό του μικροελεγκτή και την επικοινωνία του με τον Η/Υ που είναι συνδεδεμένος. Για το λόγο αυτό για το πείραμα χρησιμοποιήθηκε ποντίκι παλαιότερης τεχνολογίας με σύνδεση τύπου PS/2 που χρησιμοποιεί σειριακό πρωτόκολλο επικοινωνίας το οποίο εύκολα μπορεί να χρησιμοποιηθεί για την επικοινωνία των δύο συσκευών.



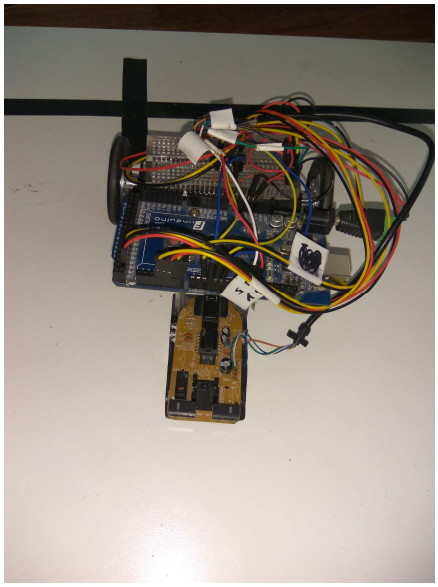
Εικόνα 2.4: Θηλυκή υποδοχή τύπου PS/2. Pin1: DATA, Pin3: GND, Pin4:Vcc 5V, Pin5:CLK

[12] 'PS/2 Port', *Wikipedia*, 19 Απρίλιος 2017, https://en.wikipedia.org/w/index.php?title=PS/2_port&oldid=776208483.

Τα pin4 και pin3 συνδέονται στην 5V τροφοδοσία και γείωση του μικροελεγκτή αντίστοιχα. Τα pin1 και pin5 συνδέονται στα pin9 και pin 10 του μικροελεγκτή και αποτελούν την σειριακή επικοινωνία δηλαδή τα DATA και CLOCK αντίστοιχα.

2.9.2 Επικοινωνία αισθητήρα - μικροελεγκτή

Για την επικοινωνία των συσκευών χρησιμοποιήθηκε η βιβλιοθήκη «Kristopher / PS2 – Mouse – Arduino» του «Kristopher Chambers»[13] η οποία δίνει την δυνατότητα εξαγωγής δεδομένων κίνησης από το ποντίκι.



Εικόνα 2.4: Μήχατρον με το κύκλωμα ποντικιού

Για τον χειρισμό του αισθητήρα με την χρήση της βιβλιοθήκης χρειάζεται δήλωση κλάσης τυπού «mouse» με τα pin που θα χρησιμοποιηθούν για την επικοινωνία, αρχικοποίηση της επικοινωνίας και ανάγνωση των μετρήσεων στόν βρόχο λειτουργείας του μικροελεγκτή.

[13] Kris Chambers, *PS2-Mouse-Arduino: Arduino/Wiring Library for interfacing with a PS2 mouse*, C++, (2009; repr., 2017), https://github.com/kristopher/PS2-Mouse-Arduino.

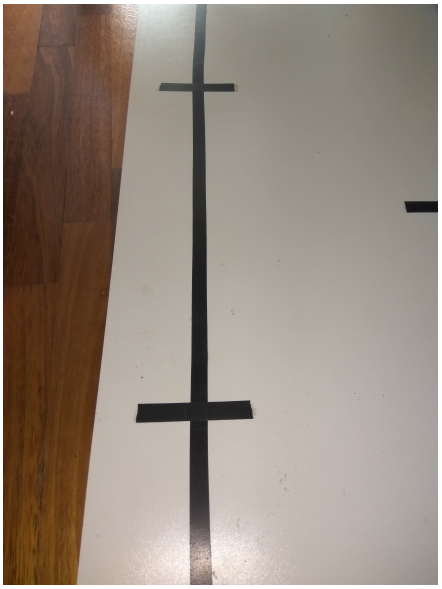
```
#include <PS2Mouse.h>
#define MOUSE_DATA 9
#define MOUSE_CLOCK 10

PS2Mouse mouse(MOUSE_CLOCK, MOUSE_DATA, STREAM);

void setup()
{
mouse.initialize();
}
void loop()
{
int data[2];
mouse.report(data);

}
```

Κώδικας επικοινωνίας μικροελεγκτή – ποντικιού

Ο πίνακας «data[2]» επιστρέφει την απόσταση που έχει διανύσει το ποντίκι στον άξονα x'x και y'y από την τελευταία φορά που είχε γίνει η ανάγνωση σε εσωτερικές μονάδες του ποντικιού. Οι μονάδες της καταγραφής δεν έχουν άμεση αντιστοίχιση σε πραγματικές μονάδες μέτρησης του μήκους καθώς για την ίδια διανυόμενη απόσταση μπορούν να εμφανίζονται διαφορετικές τιμές όταν το αισθητήριο κινείται σε επιφάνειες με διαφορετική υφή.

2.9.3 Σχεδιασμός πειράματος

Για το πείραμα χρησιμοποιείται λευκή επίπεδη επιφάνεια με μια κάθετη μαύρη γραμμή και δύο οριζόντιες γραμμές που την τέμνουν. Οι γραμμές είναι μαύρη μονωτική ταινία. Το μήχατρον είναι εξοπλισμένο με αισθητήρα υπέρυθρης ακτινοβολίας «Pololu QTR8 – RC». Είναι μια διάταξη από 8 αισθητήρια που έχουν την δυνατότητα να αναγνωρίζουν μαύρες γραμμές σε λευκή επιφάνεια (περισσότερες πληροφορίες για το αισθητήριο στο κεφάλαιο 4 (σελ. 54)). Στο παρόν πείραμα τα 6 εσωτερικά αισθητήρια της διάταξής χρησιμοποιούνται για να μπορεί το μήχατον να ακολουθεί την γραμμή και να διορθώνει την πορεία του σε περίπτωση παρεκτροπής και υλοποιείται με την χρήση αλγορίθμου ελέγχου PID (περισσότερες πληροφορίες για τον PID έλεγχο στο κεφάλαιο 5 (σελ. 59)). Τα δύο αισθητήρια στις άκρες της διάταξης χρησιμοποιούνται για την αναγνώριση των οριζόντιων γραμμών. Οι οριζόντιες γραμμές απέχουν μεταξύ τους ακριβώς 500mm.



Εικόνα 2.5: Πίστα οδομετρικού πειράματος

Κατά την διάρκεια του πειράματος το μήχατρον ξεκινάει να κινείτε εκτός των οριζόντιων γραμμών προς αυτές χωρίς να καταγράφει την απόσταση. Μόλις εντοπιστεί η πρώτη οριζόντια γραμμή επιβραδύνει μέχρι την ελάχιστη ταχύτητά του για το διάστημα που το αισθητήριο ανάκλασης βρίσκεται πάνω από την οριζόντια γραμμή. Η καταγραφή της απόστασης ξεκινάει από την στιγμή που το αισθητήριο θα φύγει πάνω από την οριζόντια γραμμή και σταματάει όταν το αισθητήριο προσπεράσει την δεύτερη γραμμή. Επιβράδυνση πραγματοποιείται και πάνω από την δεύτερη γραμμή και γίνεται για μεγαλύτερη ακρίβεια των σημείων της έναρξης και τέλους της καταμέτρησης.

2.9.4 Αποτελέσματα μετρήσεων

Μετά από 10 επαναλήψεις του παραπάνω πειράματος οι τιμές των μετρήσεων, οι αναλογίες μονάδων μέτρησης του ποντικιού – mm, ο μέσος όρος των μετρήσεων καθώς και η τυπική απόκλιση είναι οι εξής:

Πραγματική απόσταση: 500mm

α/α	**Μετρηθείσα απόσταση (μονάδες ποντικιού)**	**Αναλογία (μονάδες ποντικιού / mm)**	$(\overline{x} - x_i)^2$
1	25099	50.20	2520.04
2	25269	50.54	14352.04
3	25616	51.23	217902.24
4	25122	50.24	739.84
5	25164	50.33	219.04
6	25316	50.63	27822.24
7	24963	49.93	34670.44
8	25057	50.11	8500.84
9	25034	50.07	13271.04
10	24852	49.70	88327.84
Μέσος όρος ->	25149	**S (τυπική απόκλιση) ->**	639.00

Πίνακας 3.1: Αποτελέσματα πειράματος

Όπως βλέπουμε από τα παραπάνω αποτελέσματα, ένω η ανάλυση των μετρήσεων είναι πολύ μεγάλη και είναι κοντά στις 50 μονάδες ποντικιού ανά mm, η τυπική απόκλιση είναι κι αυτή μεγάλη: 639 μονάδες ποντικίου δηλαδή κοντά στα 12.7mm για 500mm διανυμένης απόστασης.

2.9.5 Συμπεράσματα

Όπως και σε άλλες εφαρμογές του dead reckoning έτσι και στην παρούσα κατασκευή το μεγαλύτερο πρόβλημα στην ακρίβεια των μετρήσεων είναι το σωρευτικό σφάλμα. Μικρά, τυχαία σφάλματα, αμελητέα σε μικρές κινήσεις συσσωρεύονται και δημιουργούν μεγάλες αποκλίσεις από τα πραγματικά δεδομένα. Το δεύτερο πρόβλημα του αισθητηρίου είναι η διαφορά των μετρούμενων τιμών σε σχέση με την επιφάνεια κίνησης του μήχατρον που

επέβαλαν σχολαστική βαθμονόμηση σε κάθε διαφορετική επιφάνεια κίνησης. Έτσι το αισθητήριο του οπτικού ποντικιού δεν απέδωσε την απαιτούμενη αξιοπιστία και ακρίβεια. Παρ' όλα αυτά
με προσαρμογές της λειτουργίας του οπτικού αισθητηρίου όπως διαφορετικός φακός, καλύτερος φωτισμός και μεγαλύτερης ακρίβειας αισθητήριο καθώς και εφαρμογές όπου δεν χρειάζεται συσσώρευση μετρήσεων (ταχύτητα, επιτάχυνση) θα μπορούσαν να το καταστήσουν πρακτικό και χρήσιμο.

2.9.6 Κώδικας προγραμματισμού πειράματος

```
#include <QTRSensors.h>
#include <PS2Mouse.h>
#define MOUSE_DATA 9
#define MOUSE_CLOCK 10
#define NUM_SENSORS   8
#define TIMEOUT       2500
#define EMITTER_PIN   QTR_NO_EMITTER_PIN

PS2Mouse mouse(MOUSE_CLOCK, MOUSE_DATA, STREAM);

QTRSensorsRC qtrrc((unsigned char[]) {14, 15, 16, 17, 18, 22, 24, 26},
NUM_SENSORS, TIMEOUT, EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS];
const int calSpeed = 180;
const int minSpeed = 40;
const int maxSpeed = 255;
const int baseSpeed = 180;
float Kp = 0.5;
float Ki = 0;
float Kd = 1;
float I = 1;
float error = 0;
unsigned int line_position;
int x = 0;
boolean bFlag = false;
int yacc = 0;
boolean rgt = false;
int sLeft;
int sRight;
int sLine;
int threshold = 300;


void Stop(){
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}
```

Δήλωση μεταβλητών και βιβλιοθηκών του προγράμματος.

```
void Forward(int rightSpeed, int leftSpeed){
  analogWrite(2, rightSpeed);
  digitalWrite(3, LOW);
  analogWrite(4, leftSpeed);
  digitalWrite(5, LOW);
}

void Left(int spd){
  analogWrite(2, spd);
  digitalWrite(3, LOW);
  analogWrite(5, spd);
  digitalWrite(4, LOW);
}
void Right(int spd){
  analogWrite(3, spd);
  digitalWrite(2, LOW);
  analogWrite(4, spd);
  digitalWrite(5, LOW);
}
```

Υπορουτίνες κίνησης

```
void sensorRead(){
  line_position = qtrrc.readLine(sensorValues);
  sLeft = 0;
  sRight = 0;
  sLine = 0;
  if (sensorValues[7] > threshold) sLeft = 1;
  if (sensorValues[0] > threshold) sRight = 1;
  if (sensorValues[1] > threshold || sensorValues[2] > threshold ||
sensorValues[3] > threshold || sensorValues[4] > threshold ||
sensorValues[5] > threshold || sensorValues[6] > threshold) sLine = 1;
}
```

Ανάγνωση αισθητήρων ανάκλασης

```
void setup()
{
  Serial.begin(38400);
  mouse.initialize();
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);


  delay(500);
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  for (int i = 0; i < 125; i++)
  {
    if (i==0 || i==60)
    {
       Right(baseSpeed);
    }
    else if (i==20 || i==100)
    {

       Left(baseSpeed);
    }

    qtrrc.calibrate();
  }
  Stop();
```

Βαθμονόμηση αισθητήρων ανάκλασης

```
  digitalWrite(13, LOW);


  for (int i = 0; i < NUM_SENSORS; i++)
  {
    Serial.print(qtrrc.calibratedMinimumOn[i]);
    Serial.print(' ');
  }
  Serial.println();


  for (int i = 0; i < NUM_SENSORS; i++)
  {
    Serial.print(qtrrc.calibratedMaximumOn[i]);
    Serial.print(' ');
  }
  Serial.println();
  Serial.println();

  delay(1000);
}
```

```
float previousError = 0;

int followLine (){

  sensorRead();

  if (bFlag == false){
    if (sLeft == 0 && sLine == 1 && sRight == 0){


        error = (float)line_position - 3500;
        float P = error;
        I = I + error;
        float D = error - previousError;
        float calculatePID = (Kp * P) + (Ki * I) + (Kd * D);
        previousError = error;
        int rightMotorSpeed = baseSpeed + calculatePID;
        int leftMotorSpeed = baseSpeed - calculatePID;


        if (rightMotorSpeed > maxSpeed) rightMotorSpeed = maxSpeed;
        if (leftMotorSpeed > maxSpeed) leftMotorSpeed = maxSpeed;
        if (rightMotorSpeed < 0) rightMotorSpeed = 0;
        if (leftMotorSpeed < 0) leftMotorSpeed = 0;
        Forward(rightMotorSpeed, leftMotorSpeed);
        return 100;
    }
    else {
        bFlag = true;
        return 100;
    }
  }
  else{
    if (sLeft == 1 || sLine == 0 || sRight == 1){
```

PID έλεγχος κίνησης

```
        Forward(minSpeed, minSpeed);
        return 100;
    }
    else{
        bFlag = false;
        return 0;
    }
  }


}

void odo (){
   int f;
  int data[2];

  if (x == 0){
    f = followLine();
    if (f != 100) x++;
  }
  else if (x == 1){
    f = followLine();
    mouse.report(data);
    yacc += data[2];


    if (f != 100){

      float ymm = (float)yacc/500.0;
       Serial.print(ymm);
       Serial.print(",");
       Serial.print(yacc);
       Serial.println();
       x++;
    }
  }
  else if (x > 1){
    Stop();
  }

}

void loop()
{
  odo();
}
```

Έναρξη και τέλος της καταγραφής απόστασης

Εκτύπωση αποτελέσματος

Κεφάλαιο 3

Λαβύρινθοι

[14]Λαβύρινθος είναι ο φυσικός χώρος ή το οικοδόμημα που έχει πολύπλοκους διαδρόμους κι είναι δύσκολος ο προσανατολισμός σε αυτό ή η έξοδος από αυτό ή κάθε παρόμοια διάταξη δρόμων, στοών κ.λπ. οπού είναι δύσκολο να προσανατολιστεί κανείς

3.1 Ιστορία και προέλευση

[15]Η λέξη «λαβύρινθος» έχει προ – Ελληνική (Μινωική) προέλευση που χρησιμοποιούταν για το ανάκτορο της Κνωσσού στην Κρήτη και πιθανών προέρχεται από την Λυδική λέξη «labyris» που σημαίνει: «διπλός πέλεκυς». Ήταν το σήμα της βασιλικής εξουσίας, γεγονός που υποδηλώνει ότι ο λαβύρινθος ήταν αρχικά το βασιλικό ανάκτορο της Μινωικής Κρήτης και σημαίνει «παλάτι διπλού πέλεκυ». Ο χαρακτηρισμός αυτός δεν μπορεί να περιοριστεί στο παλάτι της Κνωσού, επειδή τα ίδια σύμβολα ανακαλύφθηκαν σε άλλα ανάκτορα της Κρήτης.

Στην Ελληνική μυθολογία λαβύρινθος ήταν ένα περίτεχνο οικοδόμημα κατασκευασμένο από τον μυθικό εφευρέτη Δαίδαλο για τον βασιλιά της Κρήτης: Μίνωα. Ο σκοπός του ήταν να κρατάει φυλακισμένο τον Μινώταυρο που αργότερα σκοτώθηκε από τον Αθηναίο ήρωα Θησέα. Ο Δαίδαλος κατασκεύασε τον λαβύρινθο τόσο καλά που και ο ίδιος δυσκολεύτηκε να βρει την έξοδο αφού ολοκλήρωσε το έργο.

---

[14] ‘λαβύρινθος - Βικιλεξικό’, ημερομηνία πρόσβασης 14 Δεκέμβριος 2016, https://el.wiktionary.org/wiki/%CE%BB%CE%B1%CE%B2%CF%8D%CF%81%CE%B9%CE%BD%CE%B8%CE%BF%CF%82.

[15] ‘Maze’, *Wikipedia*, 15 Δεκέμβριος 2016, https://en.wikipedia.org/w/index.php?title=Maze&oldid=754927315.

Αν και στα πρώτα Κρητικά νομίσματα εμφανίζονταν μοτίβα με διακλαδώσεις (πολύδρομα), ο κλασσικός μονόδρομός σχεδιασμός επτά διαδρόμων χωρίς διακλαδώσεις και αδιέξοδα επικράτησε στα νομίσματα και σε άλλες αναπαραστάσεις λαβύρινθων από το 430π.Χ. παρόλο που οι λογικές και λογοτεχνικές περιγραφές καθιστούν σαφές το γεγονός ότι ο Μινώταυρος ήταν παγιδευμένος σε ένα πολύπλοκο λαβύρινθο με διακλαδώσεις. Με την πάροδο του χρόνου οι αναπαραστάσεις έγιναν πολυπλοκότερες το μονόδρομο μοτίβο επικρατούσε μέχρι την αναγέννηση. Οι πολύδρομοι λαβύρινθοι επανεμφανίστηκαν μόνο όταν οι λαβύρινθοι – θάμνοι έγιναν δημοφιλής στην αναγέννηση.

3.2 Ταξινόμηση λαβύρινθων

[16]Οι λαβύρινθοι γενικότερα (καθώς και οι αλγόριθμοι δημιουργίας τους) μπορούν να ταξινομηθούν με την βοήθεια των εξής κατηγοριών:

- Διαστάσεις
- Υπερδιαστάσεις
- Τοπολογία
- Σχήμα
- Διακλαδώσεις
- Υφή
- Προσανατολισμό

Ο κάθε λαβύρινθος είναι μια συλλογή των παρακάτω χαρακτηριστικών. Παρακάτω θα αναλύθουν οι σημαντικότερες από αυτές.

3.2.1 Ως προς τις διαστάσεις

[16] ‘Think Labyrinth: Maze Algorithms’, ημερομηνία πρόσβασης 16 Δεκέμβριος 2016, http://www.astrolog.org/labyrnth/algrithm.htm#solve.

Η κατηγορία των διαστάσεων υποδηλώνει σε πόσες διαστάσεις μπορεί να κινηθεί κανείς μέσα στον λαβύρινθο. Οι τύποι είναι:

- Δισδιάστατοι

Οι περισσότεροι Λαβύρινθοι, στο χαρτί ή σε πραγματικές διαστάσεις. Είναι οι λαβύρινθοι εκείνοι που μπορούμε να σχεδιάσουμε σε φύλλο χαρτί χωρίς οι διάδρομοι να υπερπηδούν ο ένας τον άλλον.

- Τρισδιάστατοι

Τρισδιάστατοι λέγονται οι λαβύρινθοι με πολλαπλά επίπεδα οι διάδρομοι μπορούν να πηγαίνουν στα τέσσερα σημεία του ορίζοντα καθώς και πάνω και κάτω. Οι τρισδιάστατοι λαβύρινθοι συνηθώς απεικονίζονται ως μια σειρά δισδιάστατων με σκάλες να συμβολίζουν τις πορείες προς τα πάνω και κάτω.

Εικόνα 3.1: Ρωσικό παιγνίδι τρισδιάστατου λαβύρινθου 1988[17]

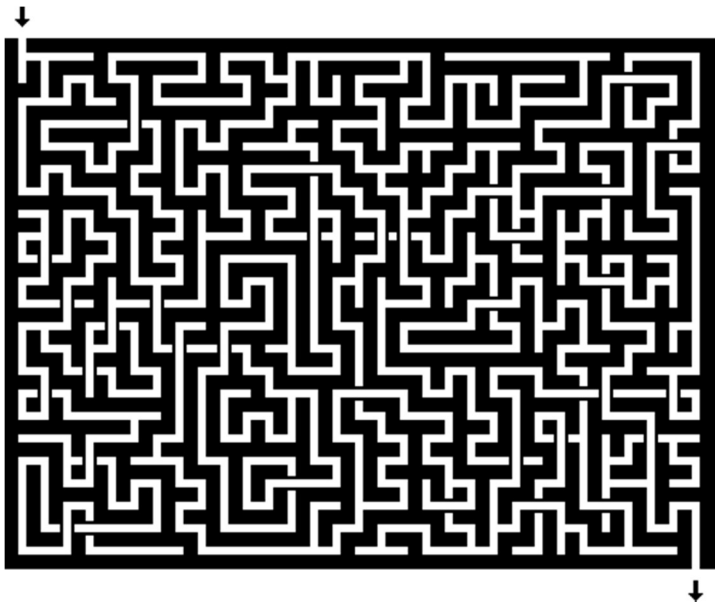
- Περισσότερων διαστάσεων

Είναι δυνατών να έχουμε τετραδιάστατους λαβύρινθους. Αυτοί είναι συνήθως τρισδιάστατοι λαβύρινθοι, με ειδικές «πύλες» που μας βοηθούν να κινούμαστε στην τέταρτη διάσταση.

- Υφαντοί

Οι υφαντοί λαβύρινθοι καμιά φορά αναφέρονται και ως δυόμιση διαστάσεων. Είναι οι περιπτώσεις εκείνες που δεν υπάρχει η διάσταση του ύψους αλλά οι διάδρομοι μπορούν να υπερπηδούν ο ένας τον άλλον.

[17] 'Rob's Puzzle Page - Mazes and Other Route Finding Puzzles', ημερομηνία πρόσβασης 16 Δεκέμβριος 2016, http://www.robspuzzlepage.com/routefind.htm.



Εικόνα 3.2: Στο σχήμα φαίνεται καθαρά που υπάρχουν τα αδιέξοδα και που οι υπερπηδήσεις

3.2.2 Υπερλαβύρινθοι

Η υπερδιάστατη ταξινόμηση αναφέρονται σε λαβύρινθους όπου το αντικείμενο που κινείται μέσα του είναι περισσότερο από μιας διάστασης (σημείο). Οι κατηγορίες είναι:

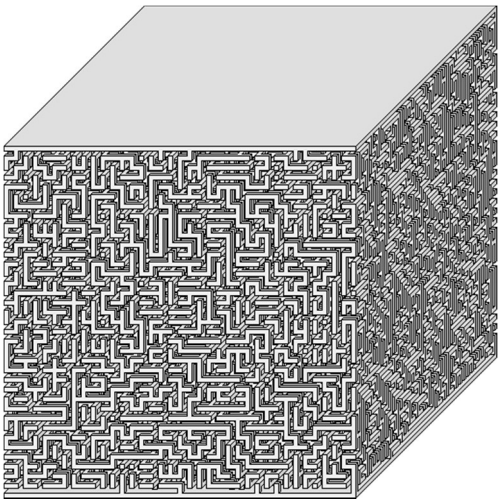
- Μη – υπερλαβύρινθος:

Οι περισσότεροι λαβύρινθοι που υπάρχουν, ακόμα κι αυτοί με περισσότερες διαστάσεις και με ειδικούς κανόνες είναι κανονικοί (μη υπερλαβύρινθοι). Σε αυτούς ο λύτης είναι ένα μικρό σημείο, όπως ο εαυτός μας ή ένας βόλος. Αυτός κινείται από σημείο σε σημείο και από διάδρομο σε διάδρομο και σε κάθε διασταύρωση υπάρχει μετρήσιμος αριθμός επιλογών πορείας.

- Υπερλαβύρινθος

Ο υπερλαβύρινθος είναι λαβύρινθος στον οποίον το αντικείμενο που κινείται είναι περισσότερο από ένα σημείο. Ο κανονικός υπερλαβύρινθος (ή λαβύρινθος πρώτης τάξης) αποτελείται από γραμμή που όπως κινείται και στρίβει στους διαδρόμους σχηματίζει επιφάνεια. Ο υπερλαβύρινθος μπορεί να υφίσταστε σε τρείς και πάνω διαστάσεις όπου η είσοδος του αντί για σημείο είναι γραμμή. Ο υπερλαβύρινθος είναι ριζικά διαφορετικός

αφού καλείται κανείς να εργαστεί με πολλαπλά μέρη του και ο αριθμός τον επιλογών πορείας τείνει στο άπειρο.



Εικόνα 3.3

3.2.3 Ως προς την γεωμετρία

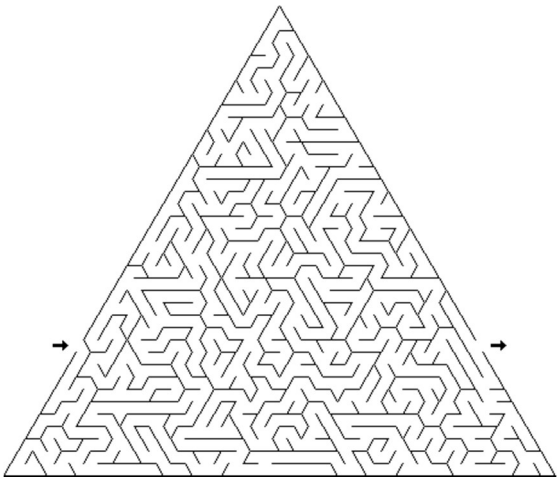
Μια κατηγορία ταξινόμησης είναι η γεωμετρία των ξεχωριστών τμημάτων που αποτελούν τον λαβύρινθο. Μερικά από αυτά είναι τα εξής:

- Ορθογώνιοι

Είναι ο τυπικός λαβύρινθος όπου οι διάδρομοί του τέμνουν ο ένας τον άλλον υπό ορθή γωνία. Στο πλαίσιο της γεωμετρίας των λαβύρινθων μπορεί να ονομαστεί και τύπου «Γάμμα».

- Τύπου «δέλτα»

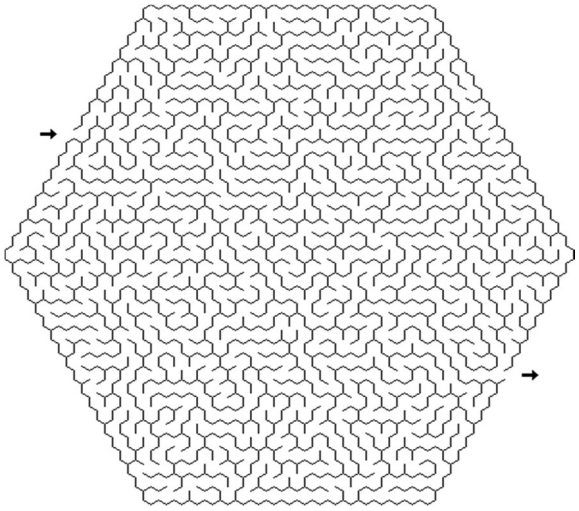
Οι λαβύρινθοι τύπου «δέλτα» αποτελούνται από σύμπλεξη τριγώνων όπου η κάθε διασταύρωση μπορεί να έχει μέχρι τρείς επιλογές πορείας.



Εικόνα 3.4:

- Τύπου «σίγμα»

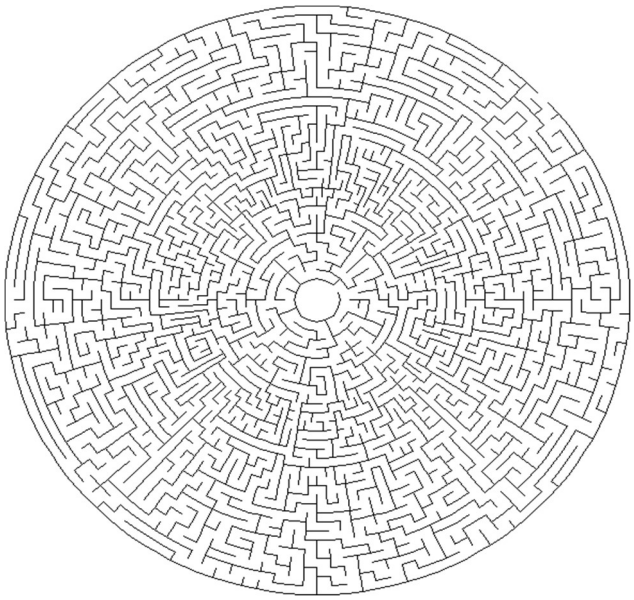
Οι λαβύρινθοι τυπου «σίγμα» αποτελούνται από αλληλοσυνδεδεμένα εξάγωνα, οπού η κάθε διασταύρωσή του μπορεί να έχει μέχρι και έξη επιλογές πορείας.



Εικόνα 3.5:

- Τύπου «θήτα»

Οι λαβύρινθοι τύπου «θήτα» αποτελούνται από ομόκεντρους κύκλους και ακτίνες με περάσματα αναμεσά τους όπου η αρχή ή το τέλος είναι το κέντρο του και η άλλη άκρη εκτός των κύκλων. Οι διασταυρώσεις έχουν συνήθως μέχρι τέσσερεις επιλογές πορείας αλλά μπορούν να έχουν και περισσότερες λόγω μεγάλου αριθμού κελιών στους εξωτερικούς κύκλους.



Εικόνα 3.6:

- Τύπου «ζήτα»

Οι λαβύρινθοι τύπου «ζήτα» είναι σαν τους ορθογώνιους με την διαφορά ότι επιτρέπονται διάδρομοι με 45 ˚ κλίση μαζί με τους οριζόντιους και τους κάθετους.

- Τύπου «ρωγμών»

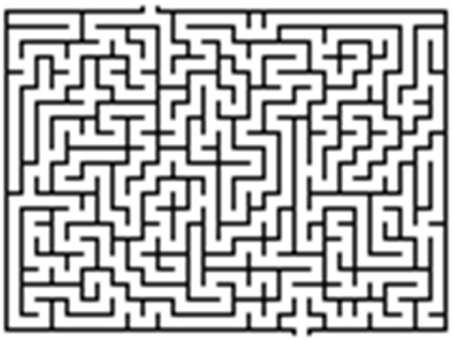
Οι λαβύρινθοι τύπου ρωγμών είναι άμορφοι χωρίς κάποια συγκεκριμένη γεωμετρία. Είναι απλά τοίχοι και διάδρομοι ύπο τυχαία γωνία.

3.2.4 Ως προς την δρομολόγηση

Η δρομολόγηση του λαβύρινθου είναι ίσως η ποιο ενδιαφέρουσα κατηγορία, διότι παίζει τον σημαντικότερο ρόλο στην επιλογή αλγορίθμου επίλυσής του. Αναφέρεται στον τύπο τον διαδρόμων στην εκάστοτε γεωμετρία του λαβύρινθου. Υπάρχουν οι εξής κατηγορίες:

- Τέλειος λαβύρινθος

Τέλειος λέγεται ο λαβύρινθος όταν δεν περιέχει βρόγχους, κλειστούς κύκλους ή μη προσβάσιμες περιοχές. Από οποιοδήποτε σημείο σε οποιοδήποτε άλλο υπάρχει μόνο μια σωστή διαδρομή. Ο λαβύρινθος αυτός έχει μόνο μια λύση. Ένας τέτοιος λαβύρινθος μπορεί να περιγραφεί ως δέντρο που αποτελείται από διακλαδώσεις και αδειέξοδα.



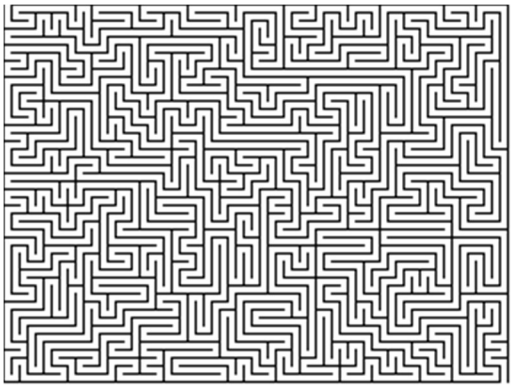
Εικόνα 3.7:

- Πλεκτός λαβύρινθος

Πλεκτοί λέγονται οι λαβύρινθοι που δεν έχουν καθόλου αδιέξοδα. Οι λαβύρινθοι αυτοί αποτελούνται από διαδρόμους που ενώνονται μεταξύ τους σχηματίζοντας βρόγχους και κάνουν τον λύτη να κάνει κύκλους αντί να βρίσκει αδιέξοδα. Ένας καλοσχεδιασμένος πλεκτός λαβύρινθος είναι πολύ πιο δύσκολος στην λύση από έναν τέλειο ίδιου μεγέθους.

- Μονόδρομος λαβύρινθος

Μονόδρομος λέγεται ο λαβύρινθος που δεν έχει καθόλου διασταυρώσεις. Είναι ένας μακρύς διάδρομος που μοιάζει με φιδάκι που σχηματίζει κύκλους στον λαβύρινθο. Δεν είναι δύσκολο να λυθεί εκτός αν μπερδευτεί κανείς στις στροφές και φτάσει πίσω στην αφετηρία.



Εικόνα 3.8:

- Μερικώς πλεκτοί

Μερικώς πλεκτοί λέγονται οι λαβύρινθοι που εμπεριέχουν και κλειστούς βρόγχους, και αδιέξοδα. Η λέξη «πλεκτοί» μπορεί να χρησιμοποιηθεί ποσοτικά. Έτσι ένας ελαφρά πλεκτός

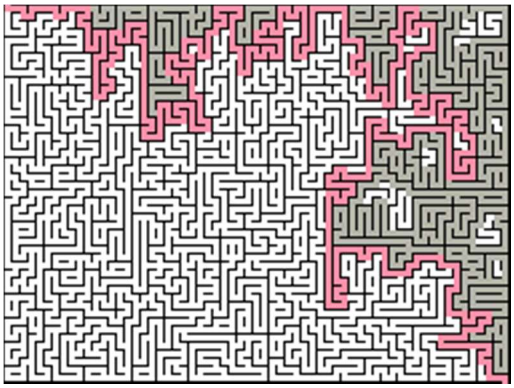
λαβύρινθος περιέχει λίγους κλειστούς βρόγχους ενώ ένας βαριά πλεκτός λαβύρινθος περιέχει πολλούς.

3.3 Αλγόριθμοι επίλυσης λαβύρινθων

Υπάρχει ένας αριθμός τρόπων με τον οποίον μπορεί κανείς να λύσει λαβύρινθους και ο καθένας από αυτούς έχει διαφορετικά χαρακτηριστικά από τους άλλους. Μερικούς από αυτούς θα αναλύσουμε παρακάτω.

3.3.1 Ακολουθητής τοίχου

[18][19]Είναι ο πιο απλός τρόπος επιλύσης λαβύρινθου. Είναι συγκεντρωμένος στον λύτη και δεν χρειάζεται μνήμη. Ξεκινάει ακολουθώντας τον διάδρομο, όταν φτάσει σε διασταύρωση πάντοτε στρίβει αριστερά (ή δεξιά). Είναι αντίστοιχο με άνθρωπο που λύνει λαβύρινθο ακουμπώντας το αριστερό (ή δεξί) του χέρι μόνιμα στον τοίχο καθώς κινείται. Αυτή η μέθοδος δεν βρίσκει υποχρεωτικά την συντομότερη διαδρομή και δεν λειτουργεί καθόλου σε λαβύρινθο με κλειστούς βρόγχους με την έξοδο να βρίσκεται στο κέντρο του. Εγγυημένη λύση ο ακολουθητής τοίχου μπορεί να δώσει μόνο σε τέλειο λαβύρινθο.

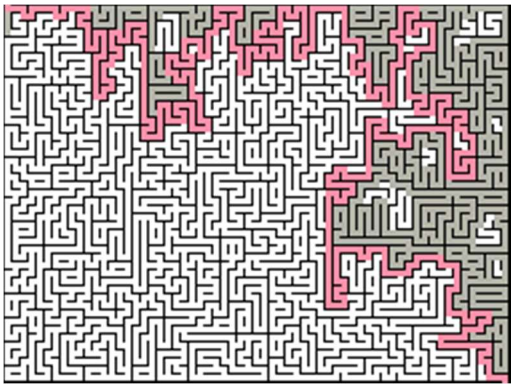


Εικόνα 3.9: Η ροζ γραμμή απεικονίζει την συντομότερη διαδρομή, ενώ η γκρίζα την υπόλοιπη διαδρομή. Η επίλυση έγινε με τον κανόνα αριστερού χεριού.

3.3.2 Αλγόριθμος Pledge

[18] 'Maze Solving Algorithm', *Wikipedia*, 3 Ιανουάριος 2017, https://en.wikipedia.org/w/index.php?title=Maze_solving_algorithm&oldid=758112000.
[19] 'Think Labyrinth: Maze Algorithms'.

Ο αλγόριθμος του Pledge είναι μια τροποποίηση του ακολουθητή τοίχου που μπορεί να πηδάει ανάμεσα σε νησίδες, για να μπορεί να λύνει λαβύρινθους που δεν μπορεί να λύσει ο προηγούμενος. Ο αλγόριθμος αυτός μπορεί, εγγυημένα να λύσει οποιονδήποτε λαβύρινθο όταν ο τερματισμός του είναι εξωτερικά του λαβύρινθου και η αφετηρία είναι εσωτερικά, δεν μπορεί να κάνει όμως το αντίθετο: π.χ. να βρει έξοδο εσωτερικά του λαβύρινθου. Είναι ιδιαίτερα καλή μέθοδος για ρομπότ επίλυσης λαβύρινθου αφού δεν χρειάζεται να μαρκάρεται ή να αποθηκεύεται η πορεία του.
Ο λύτης ξεκινάει επιλέγοντας μια κατεύθυνση και πάντα κινείται σε αυτήν όποτε αυτό είναι δυνατό, όταν βρει μπροστά του τοίχο τον ακολουθεί μέχρι να μπορέσει να ξαναπάρει την επιλεγμένη του πορεία. Όταν ο λύτης ακολουθεί τον τοίχο μετράει τον αριθμό στροφών που έχει κάνει π.χ. η στροφή προς τα αριστερά -1 και η στροφή προς τα δεξιά 1. Σταματάει να ακολουθεί τον τοίχο και ξαναπαίρνει την επιλεγμένη πορεία μόνο όταν ο αριθμός των στροφών ξαναγίνει 0. Αν ο αριθμός των στροφών ξεπεράσει τις 360 μοίρες θα ακολουθεί τον τοίχο μέχρι να φτάσει πάλι στο 0. Το μέτρημα των στροφών εγγυάται ότι κάποια στιγμή ο λύτης θα φτάσει στον εξωτερικό τοίχο τον οποίον θα ακολουθήσει και θα βρει την έξοδο. Σημαντικό είναι το γεγονός ότι με αυτόν τον αλγόριθμο ο λύτης μπορεί να περάσει από το ίδιο σημείο πάνω από μια φορά αλλά με διαφορετικό δείκτη στροφών την κάθε φορά.

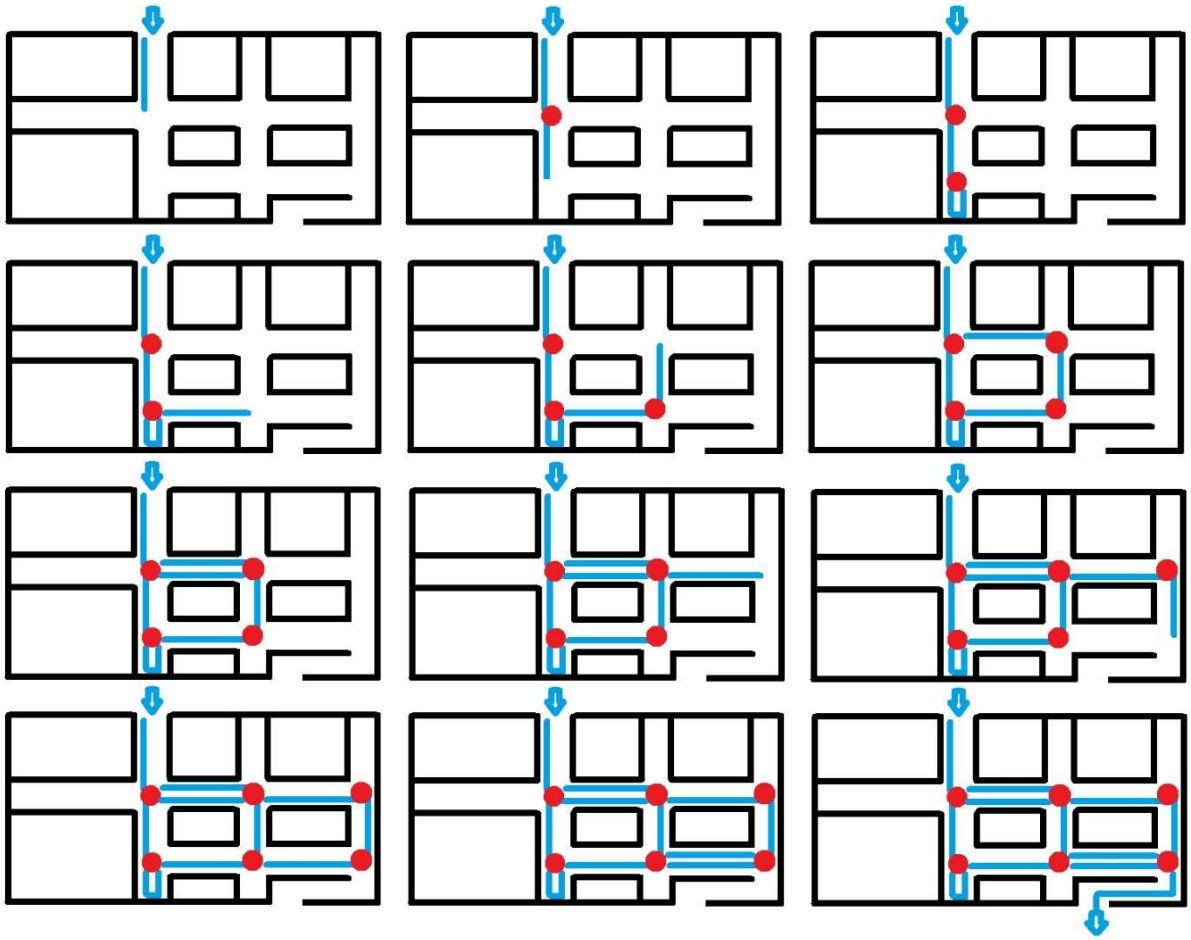


Εικόνα 3.10:

3.3.3 Αλγόριθμος Trémaux's

Ο αλγόριθμος αυτός έχει σχεδιαστεί έτσι ώστε να μπορεί να χρησιμοποιηθεί από άνθρωπο μέσα σε λαβύρινθο. Με την μέθοδο αυτή μπορούμε να λύσουμε όλους τους δισδιάστατους λαβύρινθους. Ο λύτης καθώς κινείται σε διάδρομο σχηματίζει μια γραμμή κάτω. Όταν βρεί

αδιέξοδο κάνει αναστροφή και σχηματίζει δεύτερη γραμμή δίπλα στην πρώτη. Όταν φτάσει σε διασταύρωση που δεν έχει γραμμές διαλέγει μια πορεία στην τύχη. Όταν βρίσκεται σε νέο διάδρομο και συναντήσει διασταύρωση που έχει ξαναεπισκευθεί κάνει ότι κάνει σε αδιέξοδο δηλαδή ξαναέρχεται από εκεί που ήρθε (έτσι αποκλείει την δυνατότητα να εγκλωβιστεί σε βρόγχο). Όταν κινείται σε διάδρομο που έχει ξαναπεράσει (με μια γραμμή) και συναντήσει διασταύρωση που έχει επισκεφτεί διαλέγει μια κατεύθυνση που δεν έχει ξαναπάρει και αν δεν υπάρχει, μια κατεύθυνση που έχει επισκεφτεί μια φορά (με μια γραμμή). Αυτό σημαίνει ότι όλοι οι διάδρομοι είναι: είται καμία φορά άρα πρέπει να την επισκεφτούμε, είτε μια φορά που σημαίνει ότι μπορούμε να ξαναπεράσουμε αν δεν υπάρχει άλλη λύση, είτε δυο που σημαίνει ότι είναι πορεία που καταλήγει σε αδιέξοδο ή σε βρόγχο άρα δεν την επισκεπτόμαστε ξανά. Όταν τελικά καταλήγουμε στην έξοδο οι λάθος διαδρομές θα είναι σημειωμένες δυο φορές, ενώ η σωστή μια φορά. Αν ο λαβύρινθος δεν έχει καθόλου λύσεις, ο λύτης θα βρεθεί στην αφετηρία με όλους τους διαδρόμους σημειωμένες δυο φορές.



Εικόνα 3.11: Εδώ βλέπουμε βήμα βήμα την επίλυση λαβύρινθου με τον αλγόριθμο Trémaux'

Κεφάλαιο 4

Κατασκευή

Η κατασκευή είναι το σημαντικότερο κομμάτι τις παρούσας εργασίας. Στο κεφάλαιο αυτό θα παρουσιαστούν και θα περιγραφούν: η λειτουργία της συσκευής, τα δομικά υλικά της καθώς και η διαδικασία που χρειάστηκε για να συναρμολογηθεί.

4.1 Λειτουργία

Η παρούσα κατασκευή αποτελεί ένα μικρό δίτροχο και δικινητήριο αμαξίδιο. Το αμαξίδιο κινήται σε έναν τέλειο λαβύρινθο ο οποίος αποτελείται από λευκή επιφάνεια και από μαύρες γραμμές. Όπου οι γραμμές αποτελούν τους διαδρόμους και τις διασταυρώσεις του λαβύρινθου ενώ η λευκή επιφάνεια τα μη διαθέσιμα σημεία. Το αμαξίδιο θα πρέπει να βρει την έξοδο του λαβύρινθου και να βρει την βέλτιστη διαδρομή από την είσοδο στην έξοδο.

4.2 Εξαρτήματα

4.2.1 Μικροελεγκτής

[20]Ο Arduino Uno είναι ένας μικροελεγτής για πειραματικές κατασκευές που άρχισε να κατασκευάζεται από την ομώνυμη Ιταλική εταιρία το 2005. Είναι κατασκευασμένος με βάση τον 8-bit μικροεπεξεργαστή AVR328P της εταιρίας Atmel. Προγραμματίζεται με την χρήση της ανοιχτής (open source) πλατφόρμας: Arduino IDE που είναι πλατφόρμα προγραμματισμού βασισμένη στις διαδεδομένες γλώσσες: C, C++.

[20] ‘Arduino - ArduinoBoardUno’, ημερομηνία πρόσβασης 1 Φεβρουάριος 2017, https://www.arduino.cc/en/Main/ArduinoBoardUno.

Εικόνα 4.1: Μικροελεγκτής Arduino Uno

Οι λόγοι που επιλέχθηκε ο συγκεκριμένος μικροελεγκτής είναι:

- Ευκολία διαχείρισης ψηφιακών και αναλογικών εξόδων.
- Ικανή επεξεργαστική ισχύει για την υποστήριξη αλγορίθμων επίλυσης λαβύρινθων.
- Ευκολία στον προγραμματισμό λόγω γλώσσας υψηλού επιπέδου.
- Μεγάλη ωριμότητα στην αγορά, που προσφέρει καλή υποστήριξη.
- Προσιτό κόστος αγοράς.

Τεχνικά Χαρακτηριστικά	
Ελεγκτής	ATmega328P
Τάση λειτουργίας	5V
Τάση τροφοδοσίας	6 – 20V
Ψηφιακοί είσοδοι/έξοδοι	14
Δυνατότητα PWM εξόδου σε	6
Αναλογικοί είσοδοι	6

Ρεύμα εισόδου/εξόδου	20 mA
Μνήμη Flash	32 KB (ATmega328P) Εκ των οποίων 0,5 KB χρησιμοποιούνται από το bootloader
SRAM	8 KB
EEPROM	4 KB
Συχνότητα ρολογιού	16 MHz

Πίνακας 4.1

4.2.2 Κινητήρες DC

Στην κατασκευή χρησιμοποιούνται δύο ηλεκτροκινητήρες συνεχούς τάσης για να πραγματοποιούν την κίνηση του αμαξιδίου στον χώρο.
[21]Οι DC κινητήρες αποτελούνται από τον μόνιμο μαγνήτη στον στάτη, τον δρομέα που περιέχει τυλίγματα χαλκού και ένα μηχανικό σύστημα εναλλαγής ρεύματος.
Για την εφαρμογή επιλέχθηκε μικροκινητήρας 3 – 12V με μεταλλικό γρανάζωμα μειωτήρα για μεγαλύτερη ευστάθεια.

Εικόνα 4.2: Ηλεκτροκινητήρας κατασκευής

[21] 'DC Motor', *Wikipedia*, 31 Δεκέμβριος 2016, https://en.wikipedia.org/w/index.php?title=DC_motor&oldid=757583132.

Τεχνικά χαρακτηριστικά	
Τάση λειτουργίας	3 – 12 V
Ταχύτητα	100 rpm στα 12V
Ρεύμα χωρίς φορτίο	50 mA
Ρεύμα εν στάση	600 mA
Ροπή εν στάση	1 Kg.cm
Διάμετρος άξονα	3 mm
Διαστάσεις	12 x 10 mm
Βάρος	10 gr

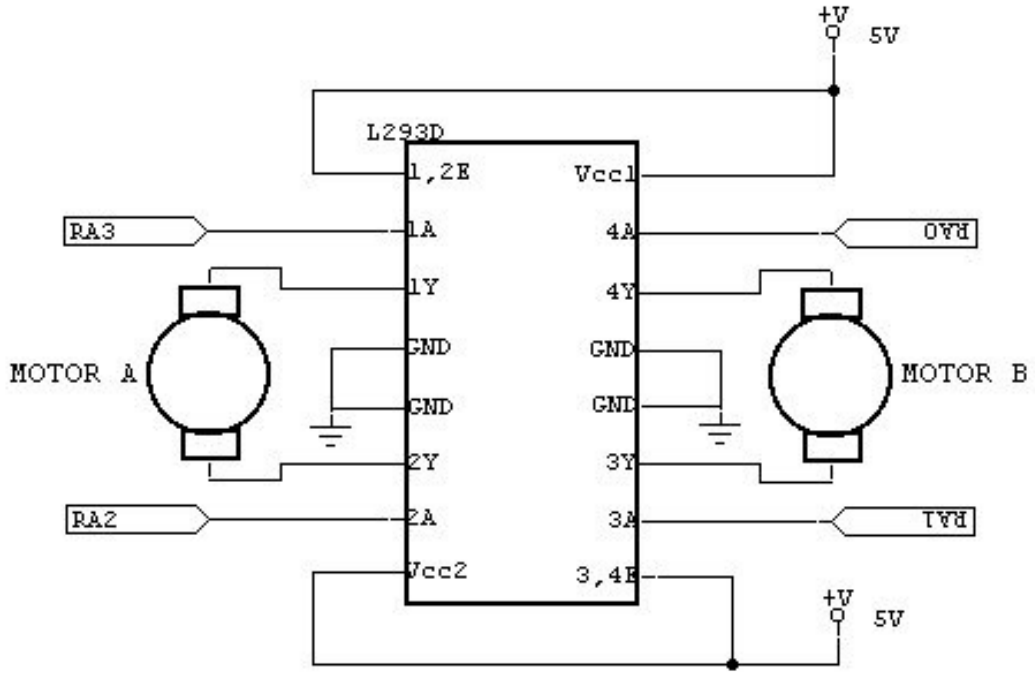
Πίνακας 4.1 Χαρακτηριστικά DC κινητήρα[22]

4.2.3 Οδηγός ηλεκτροκινητήρων

Επειδή το ρεύμα που χρειάζονται οι κινητήρες για να λειτουργήσουν είναι μεγαλύτερο από το μεγαλύτερο που μπορεί να δώσει η έξοδος του μικροελεγκτή είναι 20 mA. Για τον λόγο αυτό συνδέεται κύκλωμα οδήγησης με την μορφή ολοκληρωμένου κυκλώματος **L293D**.

Το ολοκληρωμένο κύκλωμα **L293D** είναι ένας οδηγός υψηλής τάσης και ρεύματος τεσσάρων καναλιών. Λέγεται επίσης και **H – Bridge** δηλαδή κύκλωμα που μπορεί να οδηγήσει τάση στο φορτίο και στις δύο κατευθύνσεις.

[22] 'Micro Metal Gearmotor 12mm - 100RPM 12V', ημερομηνία πρόσβασης 7 Ιανουάριος 2017, http://grobotronics.com/micro-metal-gearmotor-100-1.html.



Εικόνα 4.3: Συνδεσμολογία **L293D**[23]

[24]Τεχνικά χαρακτηριστικά:

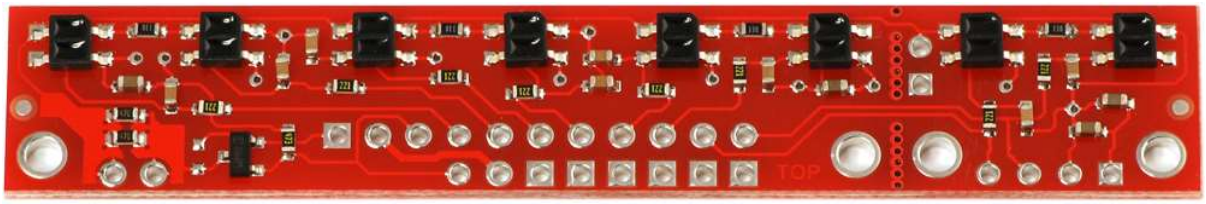
- Μεγάλο περιθώριο τάσης τροφοδοσίας 4.5 – 36 V
- Ξεχωριστή τροφοδοσία κυκλώματος ελέγχου / κυκλώματος ισχύος
- Προστασία από ηλεκτροστατική εκκένωση
- Ρεύμα εξόδου 600 mA
- Στιγμιαίο ρεύμα εξόδου έως: 1.2 A

4.2.4 Αναλογικοί αισθητήρες ανάκλασης

[25]Για την αναγνώριση των διαδρόμων του λαβύρινθου από το αμαξίδιο χρησιμοποιείται μια διάταξη από 5 αισθητήρες ανάκλασης.
Οι αισθητήρες ανάκλασης μετρούν την ποσότητα φωτός που ανακλάται από μια επιφάνεια.

[23] 'electroons.com - DC Motor direction control using l293d', ημερομηνία πρόσβασης 17 Μάρτιος 2017, http://www.electroons.com/8051/electroons/dc_motor_control.html.
[24] 'L293D H-Bridge 1A', 293, ημερομηνία πρόσβασης 1 Φεβρουάριος 2017, https://grobotronics.com/l293d-h-bridge-1a.html?sl=el.
[25] 'Pololu - QTR-8RC Reflectance Sensor Array', ημερομηνία πρόσβασης 17 Μάρτιος 2017, https://www.pololu.com/product/961.

Με αυτόν τον τρόπο μπορούν να εντοπιστούν σημεία με διαφορετικό χρώμα από την υπόλοιπη επιφάνεια (συνήθως μαύρες γραμμές σε λεύκη επιφάνεια). Λειτουργούν με την βοήθεια μιας φωτεινής πηγής που μπορεί να είναι υπέρυθρης ή ορατής ακτινοβολίας και ενός φωτοευαίσθητου ηλεκτρονικού στοιχείου (όπως: φωτοαντίσταση ή φωτοτρανζίστορ) τοποθετημένα το ένα δίπλα στο άλλο με την ίδια κατεύθυνση.



Εικόνα 4.4: Υπέρυθρος αισθητήρας ανάκλασης

Για τη κατασκευή χρησιμοποιήθηκε διάταξη από οκτώ αισθητήρες ανάκλασης που λειτουργεί με υπέρυθρη φωτοδίοδο και φωτοτρανζίστορ.

Τεχνικά χαρακτηριστικά	
Τάση τροφοδοσίας	3.3 V - 5 V
Ρεύμα τροφοδοσίας	100 mA
Θερμοκρασία λειτουργίας	-25 C / +85 C
Ιδανική απόσταση λειτουργείας	3 mm
Βάρος	3.06 g

Πίνακας 4.2: Τεχνικά χαρακτηριστικά αισθητήρων

Η διάταξη είναι στερεωμένη κάθετα στην κατεύθυνση πορείας του αμαξιδίου. Οι εσωτερικοί έξι αισθητήρες που βρίσκονται στην μέση της διάταξής χρησιμοποιούνται για να διατηρούν το αμαξίδιο επάνω στην γραμμή του διαδρόμου. Οι αισθητήρες που βρίσκονται στις άκρες χρησιμοποιούνται για την ανίχνευση των διασταυρώσεων.

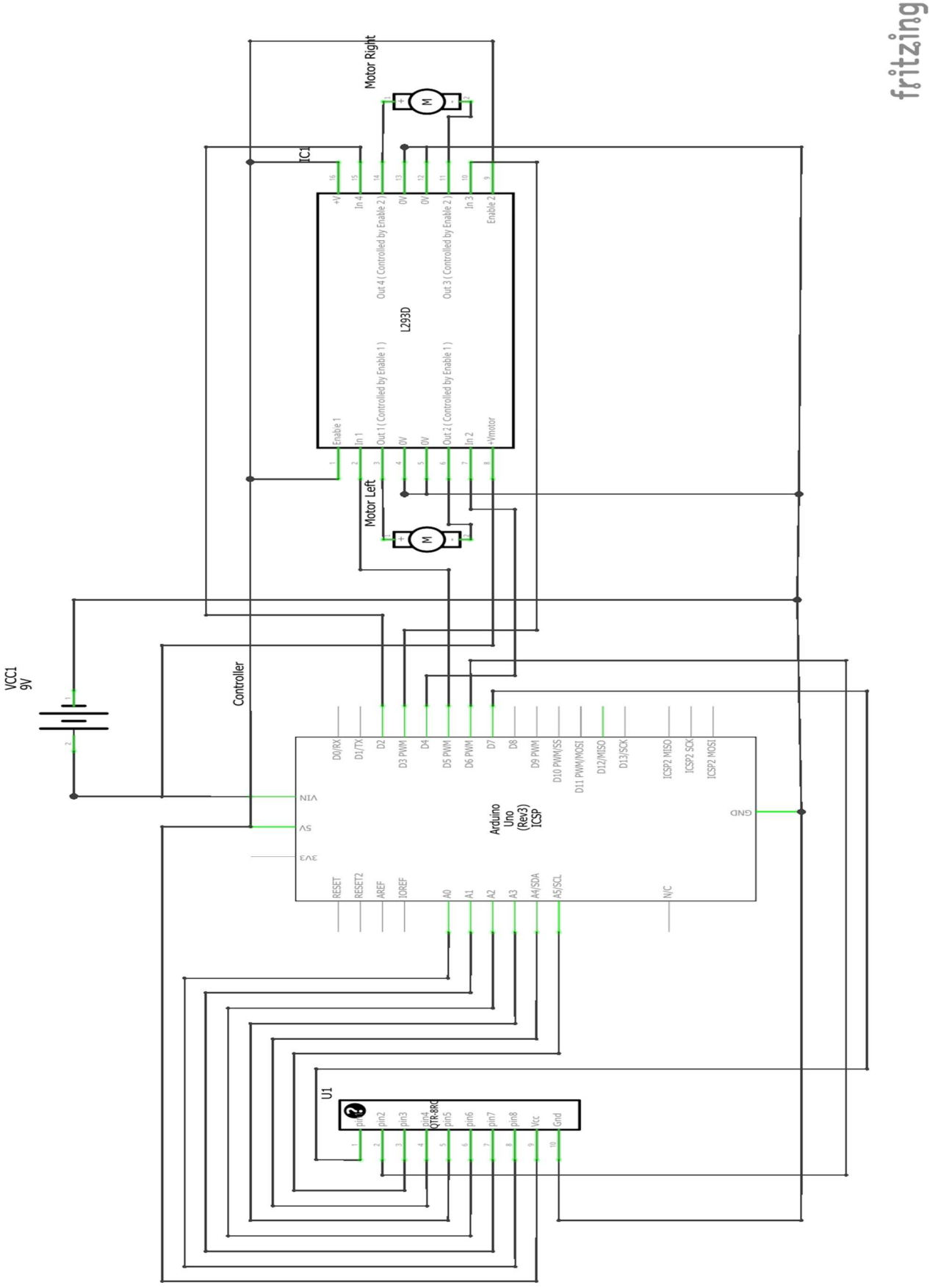
4.2.5 Τροφοδοσία

Η τροφοδοσία όλης της κατασκευής επιλέχθηκε να πραγματοποιείται από μπαταρία 9V.

Στην κατασκευή επιλέχθηκε να χρησιμοποιηθεί ενιαία τροφοδοσία τόσο για τα μοτέρ όσο και για τον μικροελεγκτή και τα αισθητήρια. Με τον τρόπο αυτό:

- Τροφοδοτούμε επαρκώς τον μικροελεγκτή
- Εκμεταλλευόμαστε την έξοδο των 5 V του ελεγκτή για την τροφοδοσία του L293D και των αισθητηρίων
- Επαρκή τροφοδοσία των κινητήρων
- Μικρό βάρος και ευκολία στην αντικατάσταση

4.3 Ηλεκτρολογίκο σχέδιο κατασκευής



Εικόνα 3.8: Ηλεκτρλογικό σχέδιο κατασκευής

Κεφάλαιο 5

Αλγόριθμος λειτουργίας

Ο αλγόριθμος λειτουργίας της κατασκευής έχει ως σκοπό την ανεύρεση της εξόδου ενός τέλειου λαβύρινθου καθώς και την απλοποίηση της πορείας του αξιοποιώντας τον ηλεκτρονικό εξοπλισμό του. Ο αλγόριθμος είναι χωρισμένος στα εξής μέρη:

- Υπορουτίνες κίνησης
- Ακολουθητής γραμμής
- Αναγνώριση διασταύρωσης
- Επιλογή πορείας
- Απλοποίηση διαδρομής
- Δεύτερο πέρασμα με απλοποιημένη διαδρομή

5.1 Υπορουτίνες κίνησης

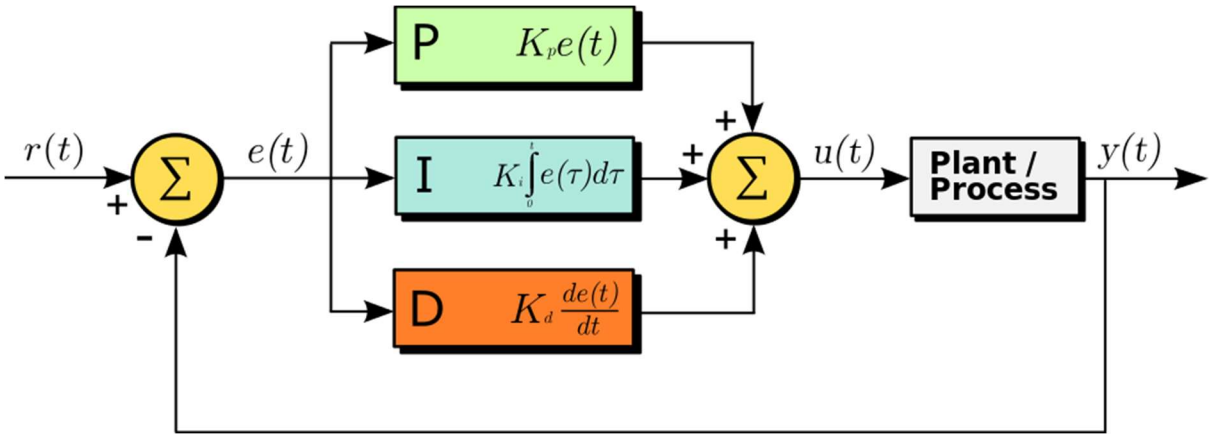
Οι υπορουτίνες κίνησης είναι οι βασικοί δομικοί λίθοι του αλγόριθμου. Χρησιμοποιούνται για:

- Την κίνηση εμπρός
- Τις στροφές αριστερά και δεξία
- Τις διορθώσεις πορείας σε περίπτωση παρεκτροπής
- Την αναστροφή
- Την στάση

5.2 Ακολουθητής γραμμής

Ο ακολουθητής γραμμής είναι ένας αλγόριθμος που φροντίζει ώστε το αμαξίδιο να παραμένει επάνω στην μαύρη γραμμή της πορείας του και να προβαίνει σε διορθωτικές ενέργειες σε περίπτωση που παρεκτραπεί.
Για την καλύτερη απόκριση και την αποτελεσματικότερη κίνηση του μήχατρον έχει εφαρμοστεί έλεγχος PID.

5.2.1 PID έλεγχος



Εικόνα 5.1: Σχηματική αναπαράσταση PID ελέγχου

[26]PID από τα αναλογικός (Proportional) – ολοκληρωτικός (Integral) – διαφορικός (Derivative) ελεγκτής είναι ένας μηχανισμός κλειστού βρόχου ανατροφοδότησης που χρησιμοποιείτε σε βιομηχανικά συστήματα ελέγχου. Ο PID υπολογίζει διαρκώς την τιμή του «σφάλματος» που είναι η διαφορά μεταξύ ενός επιθυμητού σημείου (setpoint) και της μετρούμενης μεταβλητής της εξόδου και εφαρμόζει διόρθωση βάση του αναλογικού, ολοκληρωτικού και διαφορικού όρου από τους οποίους παίρνει και το όνομά του. Ο ελεγκτής προσπαθεί να ελαχιστοποιήσει το σφάλμα τροποποιόντας την μεταβλητή ελέγχου u(t), όπως

[26] 'PID Controller', *Wikipedia*, 8 Μάρτιος 2017, https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=769341738.

την πχ την τάση τροφοδοσίας κινητήρα, η τιμή αύτη υπολογίζεται από το σταθμισμένο άθροισμα:

$$u(t) = K_P e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}(t)$$

Όπου K_p, K_i και K_d όλοι θετικοί συντελεστές για τον αναλογικό, ολοκληρωτικό και διαφορικό όρους αντίστοιχα. Σε αυτό το μοντέλο:

- P αντιπροσωπεύει την παρούσα τιμή του σφάλματος. Για παράδειγμα: αν το σφάλμα είναι θετικό και μεγάλο, η μεταβλητή θα είναι κι αυτή θετική και μεγάλη.
- I αντιπροσωπεύει τις προηγούμενες τιμές του σφάλματος. Για παράδειγμα: αν η παρούσα έξοδος δεν είναι αρκετά δυνατή, το σφάλμα θα συσσωρευτεί και ο ελεγκτής θα δώσει δυνατότερη έξοδο.
- D αντιπροσωπεύει τις πιθανές τάσεις του σφάλματος στο μέλλων, βασιζόμενος στον παρόντα ρυθμό μεταβολής του σφάλματος.

5.2.2 Έλεγχος θέσης οχήματος με PID ελεγκτή

Το μήχατρον διαθέτει την διάταξη αισθητήρων: QTR-8RC η οποία διαθέτει 8 αισθητήρες ανάκλασης σε σειρά κάθετα στην πορεία του. Μια λειτουργία της διάταξης είναι να αναγνωρίζει την θέση της μαύρης γραμμής, όταν βρίσκεται από κάτω της, και να την κωδικοποιεί σε έναν αριθμό από το 0 μέχρι το 7000.
Όταν το αισθητήριο στην δεξιά άκρη (με την φορά του οχήματος) «βλέπει» την γραμμή, επιστρέφει την τιμή 0. Όταν η γραμμή «φαίνεται» από το αισθητήριο στην δεξιά άκρη και το διπλανό αισθητήριο ταυτόχρονα επιστρέφει την τιμή 500. Όταν η γραμμή «φαίνεται» μόνο από το δεύτερο από δεξιά αισθητήριο η τιμή γίνεται: 1000 και με την ίδια λογική μέχρι την τιμή 7000.
Έτσι λοιπών η επιθυμητή τιμή είναι 3500 που σημαίνει ότι τα δυο μεσαία αισθητήρια «βλέπουν» την γραμμή. Άρα το σφάλμα (error) ισούται με την τιμή που διαβάζει το αισθητήριο μείον 3500. Αν το σφάλμα έχει θετική τιμή η γραμμή βρίσκεται αριστερότερα από το επιθυμητό και το αντίστροφο. Για να διορθωθεί το θετικό σφάλμα θα πρέπει η

μεταβλητή ελέγχου u(t), υπολογισμένη από την εξίσωση του PID, να προστεθεί στην ταχύτητα του δεξιού μοτέρ και να αφαιρεθεί από την ταχύτητα του αριστερού, με αποτέλεσμα το μήχατρον να στρίψει δεξιά. Στην περίπτωση αρνητικού σφάλματος η μεταβλητή ελέγχου γίνεται αρνητική και το μήχατρον θα στρίψει αριστερά.
Επειδή σε ορισμένες περιπτώσεις η μεταβλητή ελέγχου μπορεί να ξεπεράσει την μέγιστη δυνατή ταχύτητα των μοτέρ εφαρμόζονται φίλτρα κορεσμού που απαγορεύουν την ταχύτητα να ξεπεράσουν την μέγιστη τιμή ή να πέσουν κάτω από το μηδέν.

5.3 Αναγνώριση διασταύρωσης

Για καλύτερη κατανόηση του αντικειμένου θα δοθεί ο ορισμός της διασταύρωσης στο συγκεκριμένο αντικείμενο:
«Διασταύρωση λέγεται ένα σημείο του λαβύρινθου όπου το μήχατρον έχει την δυνατότητα να αλλάξει την πορεία του.»
Υπάρχουν 8 διαφορετικοί τύποι διασταύρωσης:

- Μόνο αριστερά
- Μόνο δεξιά
- Τύπου «Τ» αριστερά ή δεξιά
- Διασταύρωση τεσσάρων κατευθύνσεων
- Ευθεία ή αριστερά
- Ευθεία ή δεξιά
- Αδιέξοδο
- Τέλος του λαβύρινθου

Η αναγνώριση της διασταύρωσης από το μήχατρον γίνεται σε δυο φάσεις:

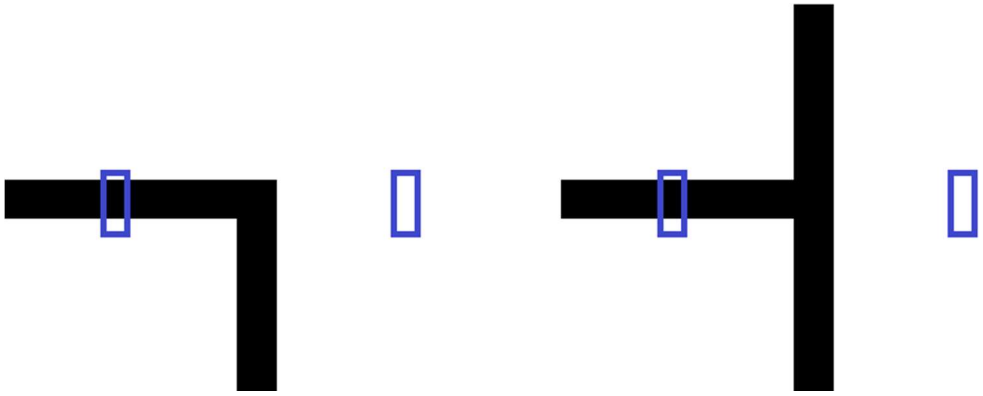
- Αναγνώριση αριστερής ή δεξιάς στροφής
- Αναγνώριση ευθείας πορείας

5.3.1 Αναγνώριση στροφής

Όταν κάποιο από τα εξωτερικά αισθητήρια εντοπίσει την παρουσία της μαύρης γραμμής το μήχατρον αντιλαμβάνεται ότι έφτασε σε διασταύρωση και αν αυτή έχει αριστερή, δεξιά ή και τις δύο κατευθύνσεις.

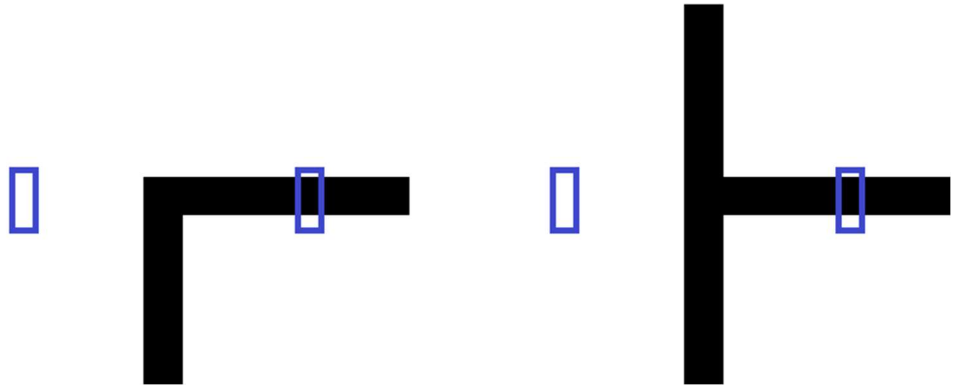
Έτσι στην πρώτη αναγνώριση ξεχωρίζουμε τέσσερεις διαφορετικές περιπτώσεις:

- Για αναγνώριση από αριστερό αισθητήριο μπορεί να είναι: Μόνο αριστερά ή ευθεία και αριστερά



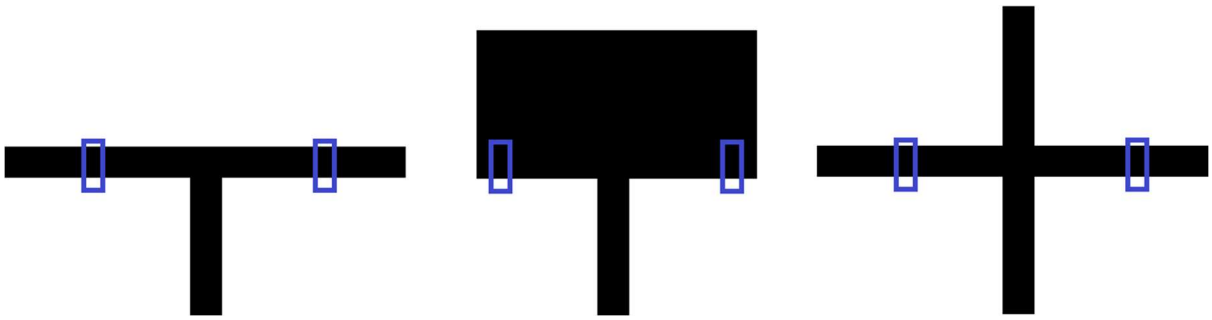
Εικόνα 5.11

- Για αναγνώριση από δεξί αισθητήριο μπορεί να είναι: Μόνο δεξιά ή ευθεία και δεξιά



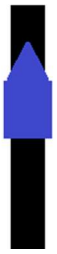
Εικόνα 5.12

- Για αναγνώριση και από τα δύο αισθητήρια μπορεί να είναι: Διασταύρωση τύπου «Τ», διασταύρωση τεσσάρων κατευθύνσεων ή το τέλος του λαβύρινθου



Εικόνα 5.13

- Για την περίπτωση που όλα τα αισθητήρια σταματήσουν να αντιλαμβάνονται την γραμμή σημαίνει ότι βρέθηκε αδιέξοδο.

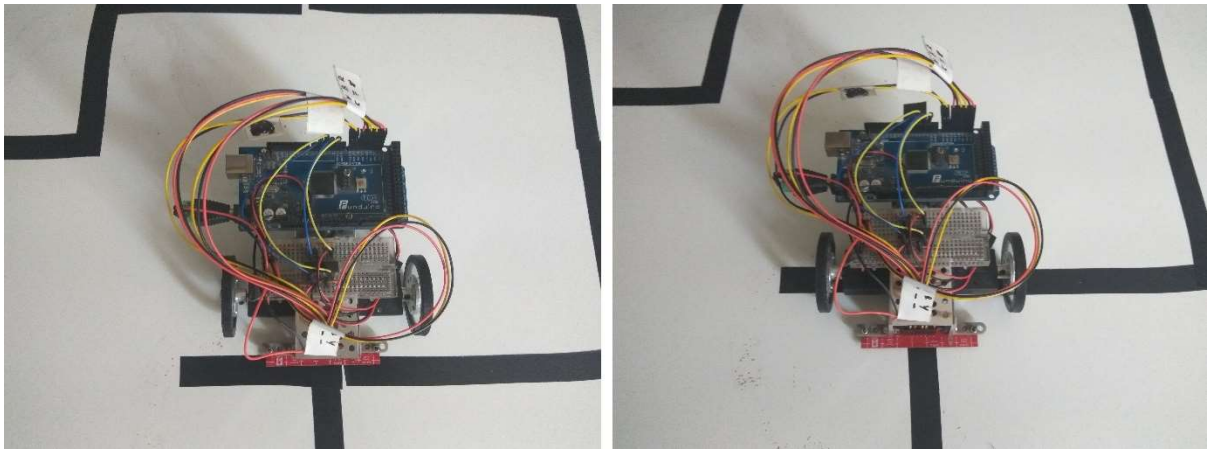


Εικόνα 5.14

Στην περίπτωση αύτη δεν χρειάζεται δεύτερη φάση αναγνώρισης, παρόλα αυτά η κίνηση πραγματοποιείται.

5.3.2 Αναγνώριση ευθείας πορείας

Κατόπιν της πρώτης αναγνώρισης το μήχατρον συνεχίζει και κινείται ευθεία για να αναγνωρίσει αν υπάρχει συνέχεια της πορείας εμπρός. Επίσης αυτό βοηθάει να ευθυγραμμιστούν οι τροχοί με την διασταύρωση για να διευκολύνει την στροφή.

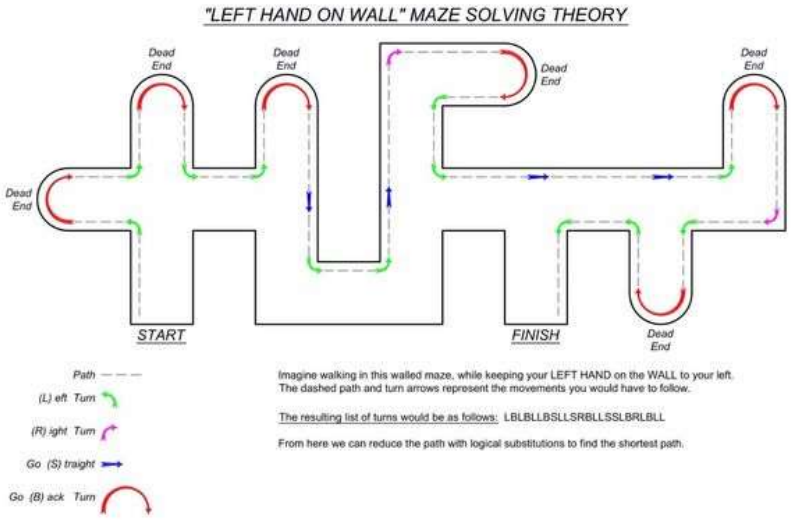


Εικόνα 5.15: Στις φωτογραφίες φαίνεται το μήχατρον στην πρώτη (αριστερά) και την δεύτερη φάση της αναγνώρισης

Η συνθήκη για την ύπαρξη ευθείας πορείας είναι: να αναγνωρίζεται μαύρη γραμμή έστω από ένα από τα έξι μεσαία αισθητήρια την στιγμή που ευθυγραμμίζονται οι τροχοί με την διασταύρωση.

5.4 Επιλογή πορείας

Ο αλγόριθμος επίλυσης λαβύρινθων που χρησιμοποιείται στην παρούσα κατασκευή είναι ο ακολουθητής τοίχου και τυχαία επιλέχτηκε να χρησιμοποιεί τον κανόνα του αριστερού χεριού.



Εικόνα 5.16: Σχεδιάγραμμα του κανόνα αριστερού χεριού[27]

[27] 'Robot Maze Solver', *Instructables.com*, ημερομηνία πρόσβασης 1 Φεβρουάριος 2017, http://www.instructables.com/id/Robot-Maze-Solver/.

Ο αλγόριθμος αποτελείται από τρείς απλές συνθήκες:

- Όταν υπάρχει δυνατότητα να στρίψει αριστερά, στρίβει αριστερά
- Όταν δεν μπορεί να στρίψει αριστερά αλλά μπορεί να συνεχίσει ευθεία, συνεχίζει ευθεία
- Όταν δεν έχει άλλη επιλογή στρίβει δεξιά

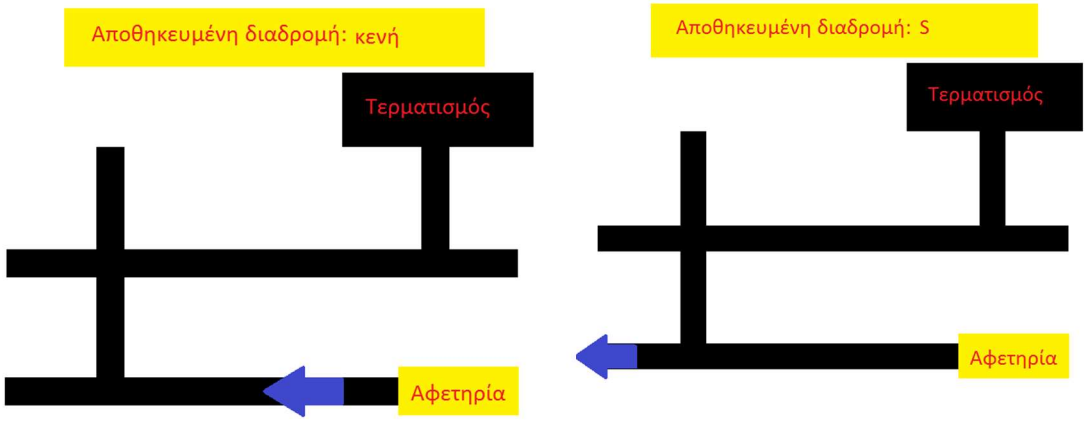
5.5 Αλγόριθμος απλοποίησης διαδρομής

[28]Το μήχατρον εκτός από το να βρίσκει την έξοδο του λαβύρινθου υπολογίζει και την συντομότερη διαδρομή από την αφετηρία μέχρι τον τερματισμό της διαδρομής.
Ο λαβύρινθος που λύνεται είναι «τέλειος» πράγμα που σημαίνει ότι δεν υπάρχουν κλειστοί βρόχοι μέσα του. Η συντομότερη διαδρομή είναι αυτή που δεν περιλαμβάνει αδιέξοδα.
Ο αλγόριθμος απλοποίησης αποθηκεύει τις αποφάσεις που παίρνει το μήχατρον σε κάθε διασταύρωση με την μορφή χαρακτήρων. Έτσι έχουμε «**L**» για αριστερά, «**R**» για δεξιά, «**S**» για ευθεία και «**U**» για αναστροφή (αδιέξοδο). Κάθε φορά που εμφανίζεται αδιέξοδο στην πορεία αντικαθίσταται μια απόφαση πριν, η αναστροφή και μια απόφαση μετά με την σωστή βάση ενός πίνακα:

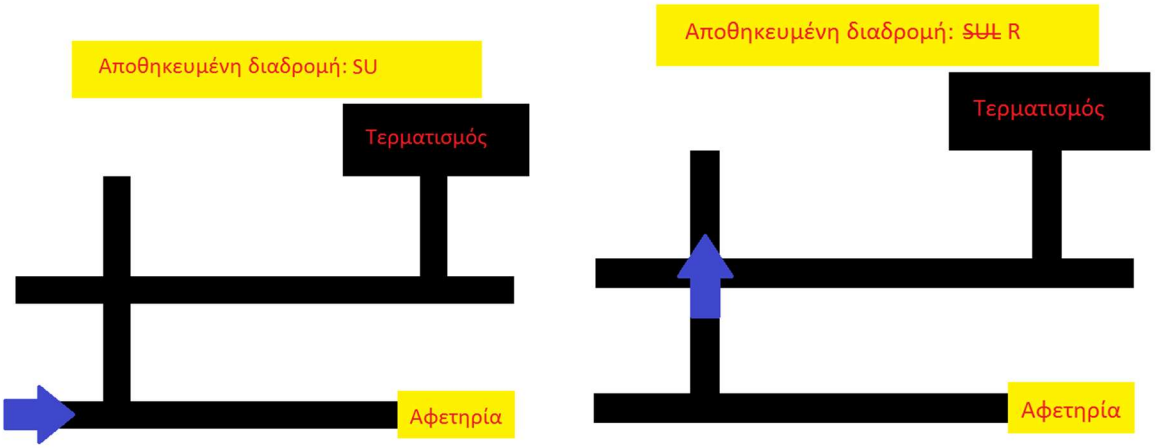
Εξάλειψη αδιέξοδων	
LUR	U
LUS	R
RUL	U
SUL	R
SUS	U
LUL	S

Πίνακας 5.1 Απλοποίηση λαβύρινθων

[28] 'line-maze-algorithm.pdf', ημερομηνία πρόσβασης 1 Φεβρουάριος 2017, https://www.pololu.com/file/0J195/line-maze-algorithm.pdf.

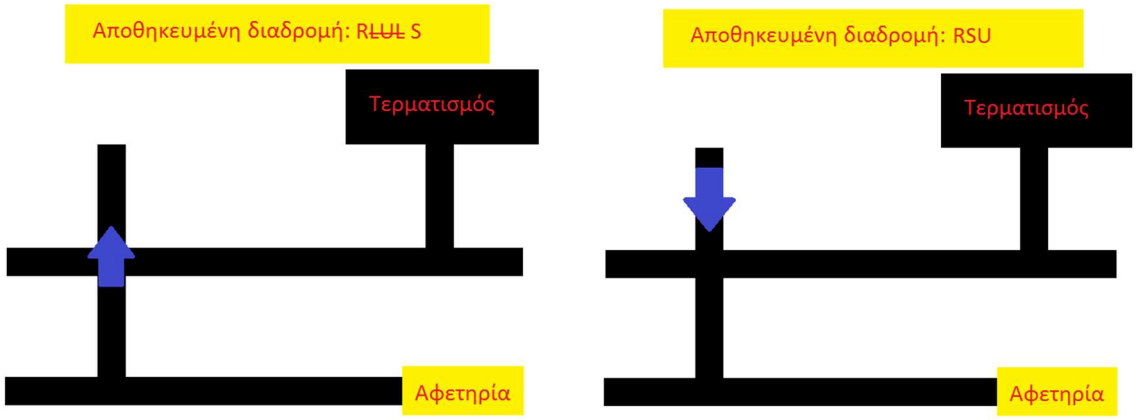


Εικόνα 5.17

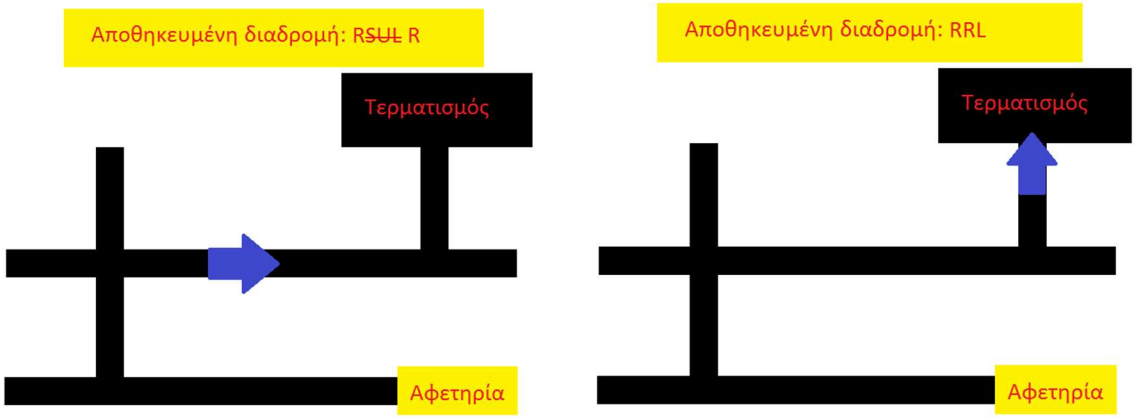


Εικόνα 5.18

Όπως φαίνεται παραπάνω η διαδρομή: “SUL” απλοποιείται σε “R” βάση του πίνακα.



Εικόνα 5.19



Εικόνα 5.20

Έτσι βλέπουμε την αρχική διαδρομή που είναι: “SULLULULL” να απλοποιείται στην βέλτιστη που είναι η “RRL”.

5.6 Τελικό πέρασμα

Κατόπιν του πρώτου περάσματος και της απλοποίησης της πορείας το μήχατρον καλείται να ξαναπεράσει την διαδρομή με τον βέλτιστο τρόπο. Για τον λόγο αυτό αναμένει την επανατοποθέτηση του στην αρχή του λαβύρινθου. Όταν τοποθετηθεί ξανά αναμένει για 2

δευτερόλεπτα για να διευκολύνει τον χρήστη. Αφού τοποθετηθεί περνάει τον λαβύρινθο με τον βέλτιστο τρόπο και σταματάει στον τερματισμό.

Κεφάλαιο 6

Κώδικας

Ο κώδικας προγραμματισμού της παρούσας κατασκευής αποτελεί το σημαντικότερο κομμάτι αφού αυτός θέτει σε κίνηση όλα τα εξαρτήματά της και αποτελεί το «έξυπνο» κομμάτι της.

Όλος ο κώδικας είναι γραμμένος σε Arduino IDE που είναι μια γλώσσα πολύ κοντά στην γνωστή C++.

[29]#include <QTRSensors.h>

Δήλωση βιβλιοθήκης λειτουργίας αισθητηρίων και βοηθητικών μεταβλητών.

```
#define NUM_SENSORS   8
#define TIMEOUT       2500
#define EMITTER_PIN   QTR_NO_EMITTER_PIN

QTRSensorsRC qtrrc((unsigned char[]) {14, 15, 16, 17, 18, 19, 20, 21},
  NUM_SENSORS, TIMEOUT, EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS];
```

```
const int calSpeed = 180;
const int maxSpeed = 255;
const int baseSpeed = 200;
```

Ταχύτητα κίνησης ηλεκτροκινητήρων και σε κατάσταση πορείας, βαθμονόμησης και διόρθωσης πορείας

```
float Kp = 0.5;
float Ki = 0.0001;
float Kd = 5;
float I = 0;
float error = 0;
unsigned int line_position;
```

Σταθερές και μεταβλητές λειτουργείας του PID ελέγχου πορείας.

```
int sLeft;
int sRight;
```

Μεταβλητές αισθητήρων ανάκλασης.

[29] 'Pololu - 3.k. Pololu QTR Sensor Functions', ημερομηνία πρόσβασης 17 Μάρτιος 2017, https://www.pololu.com/docs/0J20/3.k.

```
int sLine;


int x = 0;
int inchLoops = 150;
```

Επιλογή λειτουργίας και διάρκεια ευθυγράμμισης τροχών – διασταύρωσης.

```
const int pathSize = 50;
char path[pathSize];
int pathLength = 0;
int pathPosition = 1;
int threshold = 300;
```

Συμβολοσειρά αποθήκευσης πορείας, μήκος πορείας και δείκτης πορείας.

```
void sensorRead(){
  line_position = qtrrc.readLine(sensorValues);
  sLeft = 0;
  sRight = 0;
  sLine = 0;
  if (sensorValues[7] > threshold) sLeft = 1;
  if (sensorValues[0] > threshold) sRight = 1;
  if (sensorValues[1] > threshold || sensorValues[2] > threshold || sensorValues[3] > threshold
|| sensorValues[4] > threshold || sensorValues[5] > threshold || sensorValues[6] > threshold)
sLine = 1;

}

void Stop(){
  digitalWrite(2, LOW);
  digitalWrite(3 LOW);
```

Ρουτίνα ανάγνωσης αισθητηρίων ανάκλασης.

Ρουτίνα διακοπής κίνησης κινητήρων.

```
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}


void Forward(int rightSpeed, int leftSpeed){
  analogWrite(2, rightSpeed);
  digitalWrite(3, LOW);
  analogWrite(4, leftSpeed);
  digitalWrite(5, LOW);
}
```

Ρουτίνα ευθείας κίνησης όπου οι τιμές των τυπικών παραμέτρων καθορίζουν την ταχύτητα του κάθε κινητήρα ξεχωριστά με σκοπό την διόρθωση πορείας σε περίπτωση παρεκτροπής

```
void Left(int spd){
  analogWrite(2, spd);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  analogWrite(5, spd);
}

void Right(int spd){
  digitalWrite(2, LOW);
  analogWrite(3, spd);
  analogWrite(4, spd);
  digitalWrite(5, LOW);
}
```

Ρουτίνες αριστερής και δεξιάς κίνησης με είσοδο την ταχύτητα.

```
  void turnLeft(){
 while (sLeft == 0){
  sensorRead();
  Left(baseSpeed);
 }
 while (sensorValues[4] < threshold){
  sensorRead();
  Left(baseSpeed);
 }
}
void turnRight(){
 while (sRight == 0){
  sensorRead();
  Right(baseSpeed);
 }
 while (sensorValues[2] < threshold){
  sensorRead();
  Right(baseSpeed);
 }
}
```

Ρουτίνες που εκτελούν την αριστερή την δεξιά στροφή και την αναστροφή. Η στροφή ολοκληρώνεται όταν το εξωτερικό και το κεντρικό αισθητήριο εντοπίσουν την μαύρη γραμμή διαδοχικά.

```
 void sensorPrint (){
 sensorRead();
 Serial.print(sLeft);
 Serial.print('\t');
 Serial.print(sLine);
 Serial.print('\t');
 Serial.print(sRight);
 Serial.print('\t');
 Serial.println();
}
```

Ρουτίνα που εκτυπώνει δυναμικά στην οθόνη του υπολογιστή την κατάσταση του κάθε αισθητηρίου απόστασης. Χρησιμοποιείται για τον έλεγχο λειτουργίας

```
int inch(){
  int lft = 0;
  int rgt = 0;
  for (int i = 0; i < inchLoops; i++){
    sensorRead();
    if (sLeft == 1) lft = 1;
    if (sRight == 1) rgt = 1;
    Forward(baseSpeed, baseSpeed);
    }
    sensorRead();
  if (lft == 1 && rgt == 1){
    if (sLeft == 1 && sLine == 1 && sRight == 1)  return 0;
    else if (sLine == 1) return 2;
    else if (sLine == 0) return 3;
  }
  else if (lft == 1){
    if (sLine == 1) return 4;
    else return 6;
  }
  else if (rgt == 1){
    if (sLine == 1) return 5;
    else return 7;
  }
  else return 1;

}
```

Μια από τις σημαντικότερες συναρτήσεις του προγράμματος. Τίθεται σε λειτουργία όταν εντοπιστεί διασταύρωση. Κινεί το όχημα εμπρός μέχρι την ευθυγράμμιση των τροχών με την διασταύρωση αναγνωρίζοντας παράλληλα το είδος της διασταύρωσης. Κατόπιν επιστρέφει στην ρουτίνα από την οποία κλήθηκε αριθμό που αντιστοιχεί στο είδος της διασταύρωσης

Η συνάρτηση «selectTurn» αποτελεί τον πίνακα απλοποιήσεων της διαδρομής. Ως είσοδο έχει το πρώτο και τελευταίο χαρακτήρα της τριάδας προς απλοποίηση και επιστρέφει το αποτέλεσμα.

```
char selectTurn(char first, char last){
  if (first == 'L' && last == 'L')
    return 'S';
  if ((first == 'S' && last == 'L') || (first == 'L' && last == 'S'))
    return 'R';
  if ((first == 'L' && last == 'R') || (first == 'R' && last == 'L') || (first == 'S' && last == 'S'))
    return 'U';
}
```

Η ρουτίνα αυτή απλοποιεί την διαδρομή κάθε φορά που συναντάει τριάδα διασταυρώσεων με αδιέξοδο στην μέση. Καλεί την συνάρτηση «selectTurn» για την αντιστοίχηση

```
void simplify(){
  if (pathLength >= 3){
    if (path[pathLength - 1] == 'U'){
      path[pathLength - 2] = selectTurn(path[pathLength - 2], path[pathLength]);
      pathLength-=2;
    }
  }
}
```

```
int waitForRound2(){
  Stop();
  sensorRead();
  if (farLeft == 0 && (nearLeft == 1 || Center == 1 || nearRight == 1) && farRight == 0){
    delay(2000);
    return 2;
  }
  else{
    return 1;
  }
}
```

Η συνάρτηση αυτή, αναμένει να επανατοποθετηθεί το μήχατρον στην αρχή της διαδρομής για το δεύτερο πέρασμα. Καλείται όταν φτάσει στο τέρμα του λαβύρινθου και επιστρέφει «2» σε περίπτωση που ένα από τα τρία εσωτερικά αισθητήρια εντοπίσει την μαύρη γραμμή.

```
void setup(){
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);


  delay(500);
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
```

Στο σημείο αυτό είναι η δήλωση των ακροδεκτών εξόδου που χρησιμοποιούνται για τους κινητήρες και εισόδου που χρησιμοποιούνται για τους αισθητήρες ανάκλασης.

```
for (int i = 0; i < 125; i++)
  {
   if (i==0 || i==60)
   {
     Right(baseSpeed);
   }
   else if (i==20 || i==100)
   {

     Left(baseSpeed);
   }

   qtrrc.calibrate();
  }
  Stop();
  digitalWrite(13, LOW);
 delay(1000);
}

float previousError = 0;

int followLine (){
  sensorRead();


  if (sLeft == 0 && sLine == 1 && sRight == 0){
    error = (float)line_position - 3500;
    float P = error;
```

Σε αυτό το μέρος του κώδικα εκτελείτε η βαθμονόμηση των αισθητήρων ανάκλασης, στρίβοντας διαδοχικά αριστερά και δεξιά πάνω από την μαύρη γραμμή ώστε το κάθε αισθητήριο να καταγράψει την τιμή της γραμμής και της επιφάνειας

Αλγόριθμος ελέγχου PID το σφάλμα έρχεται από το αισθητήριο, ο ολοκληρωτικός παράγοντας από την συσσώρευση σε μια μεταβλητή και ο διαφορικός παράγοντας είναι το υπόλοιπο του σφάλματος μείον το σφάλμα του προηγουμενου κυκλου εργασίας

[30] 'PID Based Line Follower | Let's Make Robots! | RobotShop', ημερομηνία πρόσβασης 17 Μάρτιος 2017, http://www.robotshop.com/letsmakerobots/pid-based-line-follower.

```
  I = I + error;
  float D = error - previousError;
  float calculatePID = (Kp * P) + (Ki * I) + (Kd * D);
  previousError = error;
  int rightMotorSpeed = baseSpeed + calculatePID;
  int leftMotorSpeed = baseSpeed - calculatePID;
```

```
  if (rightMotorSpeed > maxSpeed) rightMotorSpeed = maxSpeed;
  if (leftMotorSpeed > maxSpeed) leftMotorSpeed = maxSpeed;
  if (rightMotorSpeed < 0) rightMotorSpeed = 0;
  if (leftMotorSpeed < 0) leftMotorSpeed = 0;
  Forward(rightMotorSpeed, leftMotorSpeed);
  return 100;
 }
 else{
  return inch();
 }
}
```

Το φίλτρο κορεσμού αποτρέπει την υπέρβαση της μέγιστης και ελάχιστης ταχύτητας. Όταν το μήχατρον συναντήσει διασταύρωση καλεί την συνάρτηση «inch» για αναγνώριση.

```
void leftHandSolve(){

 int f;
 int g;
 if (x == 0){
  f = followLine();
```

Το πρώτο στάδιο λειτουργίας καλεί την συνάρτηση «followLine», όταν αυτή επιστρέψει κωδικό διασταύρωσης αλλάζει πορεία σύμφωνα με τον αλγόριθμο, αποθηκεύει τον αντίστοιχο χαρακτήρα στην συμβολοσειρά της πορείας και καλεί την ρουτίνα απλοποίησης. Όταν βρεθεί το τέλος του λαβύρινθου ενεργοποιεί το δεύτερο στάδιο λειτουργείας.

```
switch (f){

case 0:
  x = 1;
  for(int i = 0; i <= pathLength; i++){
    Serial.print(path[i]);
  }
break;



case 1:
  turnLeft();
  pathLength++;
  path[pathLength] = 'U';
  simplify();
break;



case 2:
  turnLeft();
  pathLength++;
  path[pathLength] = 'L';
  simplify();
break;



case 3:
  turnLeft();
  pathLength++;
  path[pathLength] = 'L';
  simplify();
```

Η ρουτίνα «leftHandSolve» είναι η κυρίως ρουτίνα του προγράμματος. Αποτελείται από τρία στάδια λειτουργίας:

1. Την ανεύρεση εξόδου από τον λαβύρινθο.
2. Την αναμονή για επανατοποθέτηση στην αφετηρία
3. Το δεύτερο πέρασμα με τον βέλτιστο τρόπο

Η μεταβλητή «x» καθορίζει το τρέχων στάδιο

```
break;



case 4:
  turnLeft();
  pathLength++;
  path[pathLength] = 'L';
  simplify();
break;



case 5:
  pathLength++;
  path[pathLength] = 'S';
  simplify();
break;



case 6:
  turnLeft();
  pathLength++;
  path[pathLength] = 'L';
  simplify();
break;



case 7:
  turnRight();
  pathLength++;
  path[pathLength] = 'R';
```

```
      simplify();
    break;
  }

  }
  else if (x == 1){
    x = waitForRound2();
  }


  else if (x == 2){
    g = followLine();
    if (g == 0){
      pathPosition = 1;
      x = 1;
    }
    else if (g != 100 && g != 0){

      Serial.print(pathPosition);
      Serial.println();
      if (path[pathPosition] == 'L'){
        turnLeft();
      }
      else if (path[pathPosition] == 'R'){
        turnRight();
      }
    pathPosition++;
    }
  }
}

void loop() {
  leftHandSolve();
}
```

Στο δεύτερο στάδιο λειτουργίας το μήχατρον αναμένει την επανατοποθέτησή του με την βοήθεια της συνάρτησης: «waitForRound2».

Στο τρίτο στάδιο λειτουργίας χρησιμοποιείται ξανά η συνάρτηση «followLine» όταν αυτή επιστρέψει κάποια διασταύρωση το μήχατρον πορεύεται βάση της αποθηκευμένης απλοποιημένης διαδρομής.

Βιβλιογραφία

‘λαβύρινθος - Βικιλεξικό’. Ημερομηνία πρόσβασης 14 Δεκέμβριος 2016. https://el.wiktionary.org/wiki/%CE%BB%CE%B1%CE%B2%CF%8D%CF%81%CE%B9%CE%BD%CE%B8%CE%BF%CF%82.

‘Μηχατρονική’. *Βικιπαίδεια*, 9 Φεβρουάριος 2016. https://el.wikipedia.org/w/index.php?title=%CE%9C%CE%B7%CF%87%CE%B1%CF%84%CF%81%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE&oldid=5676110.

‘Arduino - ArduinoBoardUno’. Ημερομηνία πρόσβασης 1 Φεβρουάριος 2017. https://www.arduino.cc/en/Main/ArduinoBoardUno.

‘Celestial Navigation’. *Wikipedia*, 21 Απρίλιος 2017. https://en.wikipedia.org/w/index.php?title=Celestial_navigation&oldid=776561117.

Chambers, Kris. *PS2-Mouse-Arduino: Arduino/Wiring Library for interfacing with a PS2 mouse*. C++, 2017. https://github.com/kristopher/PS2-Mouse-Arduino.

‘DC Motor’. *Wikipedia*, 31 Δεκέμβριος 2016. https://en.wikipedia.org/w/index.php?title=DC_motor&oldid=757583132.

‘electroons.com - DC Motor direction control using l293d’. Ημερομηνία πρόσβασης 17 Μάρτιος 2017. http://www.electroons.com/8051/electroons/dc_motor_control.html.

‘Global Positioning System’. *Βικιπαίδεια*, 30 Νοέμβριος 2016. https://el.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=6133101.

‘Inertial Navigation System’. *Wikipedia*, 12 Απρίλιος 2017. https://en.wikipedia.org/w/index.php?title=Inertial_navigation_system&oldid=775031862.

‘L293D H-Bridge 1A’. Ημερομηνία πρόσβασης 1 Φεβρουάριος 2017. https://grobotronics.com/l293d-h-bridge-1a.html?sl=el.

‘line-maze-algorithm.pdf’. Ημερομηνία πρόσβασης 1 Φεβρουάριος 2017. https://www.pololu.com/file/0J195/line-maze-algorithm.pdf.

Maimone, Mark, Yang Cheng, και Larry Matthies. ‘Two years of Visual Odometry on the Mars Exploration Rovers’, 2007.

‘Maze’. *Wikipedia*, 15 Δεκέμβριος 2016. https://en.wikipedia.org/w/index.php?title=Maze&oldid=754927315.

‘Maze Solving Algorithm’. *Wikipedia*, 3 Ιανουάριος 2017. https://en.wikipedia.org/w/index.php?title=Maze_solving_algorithm&oldid=758112000.

‘Micro Metal Gearmotor 12mm - 100RPM 12V’. Ημερομηνία πρόσβασης 7 Ιανουάριος 2017. http://grobotronics.com/micro-metal-gearmotor-100-1.html.

‘Odometry’. *Wikipedia*, 9 Μάρτιος 2017. https://en.wikipedia.org/w/index.php?title=Odometry&oldid=769409564.

‘Optical Mouse’. *Wikipedia*, 18 Μάρτιος 2017. https://en.wikipedia.org/w/index.php?title=Optical_mouse&oldid=770892163.

‘PID Based Line Follower | Let’s Make Robots! | RobotShop’. Ημερομηνία πρόσβασης 17 Μάρτιος 2017. http://www.robotshop.com/letsmakerobots/pid-based-line-follower.

‘PID Controller’. *Wikipedia*, 8 Μάρτιος 2017. https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=769341738.

‘Pololu - 3.k. Pololu QTR Sensor Functions’. Ημερομηνία πρόσβασης 17 Μάρτιος 2017. https://www.pololu.com/docs/0J20/3.k.

‘Pololu - QTR-8RC Reflectance Sensor Array’. Ημερομηνία πρόσβασης 17 Μάρτιος 2017. https://www.pololu.com/product/961.

‘PS/2 Port’. *Wikipedia*, 19 Απρίλιος 2017. https://en.wikipedia.org/w/index.php?title=PS/2_port&oldid=776208483.

‘Robot Maze Solver’. *Instructables.com*. Ημερομηνία πρόσβασης 1 Φεβρουάριος 2017. http://www.instructables.com/id/Robot-Maze-Solver/.
‘Rob’s Puzzle Page - Mazes and Other Route Finding Puzzles’. Ημερομηνία πρόσβασης 16 Δεκέμβριος 2016. http://www.robspuzzlepage.com/routefind.htm.
‘Rotary Encoder’. *Wikipedia*, 14 Μάρτιος 2017. https://en.wikipedia.org/w/index.php?title=Rotary_encoder&oldid=770341895.
Singh, Avi. ‘Visual Odmetry from scratch - A tutorial for beginners’. *Avi Singh’s blog*. Ημερομηνία πρόσβασης 6 Μάιος 2017. https://avisingh599.github.io/vision/visual-odometry-full/.
‘Think Labyrinth: Maze Algorithms’. Ημερομηνία πρόσβασης 16 Δεκέμβριος 2016. http://www.astrolog.org/labyrnth/algrithm.htm#solve.