Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Μηχανολογων Μηχανικων

Εργαστηριο Αυτοματου Ελεγχου και Ρυθμισεως Μηχανων

Διπλωματικη Εργασια

Ανάπτυξη Συστήματος Ελέγχου Κίνησης Τροχοφόρων Ρομπότ

Πούλιας Απόστολος

Επιβλέπων:

Κυριακόπουλος Κ. Κωνσταντίνος

Καθηγητής ΕΜΠ

Φεβρουάριος 2015

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η ανακατασκευή των mobile robots της εταιρίας Adept. Τα νέα συστήματα αυτά λειτουργούν υπό το Robot Operating System (ROS), το οποίο συνεπάγεται πληθώρα ευκολιών στην χρήση και περαιτέρω ανάπτυξη και εμπλουτισμό των συστημάτων με νέα χαρακτηριστικά. Παράλληλα, πέραν των ρομπότ αναπτύσσεται λογισμικό υπολογισμού των χαρακτηριστικών της οδομετρίας τους.

Στο κείμενο που ακολουθεί αρχικά περιγράφονται οι εγκαταστάσεις των ρομπότ που έχει στην κατοχή του το εργαστήριο και ανακατασκευάστηκαν. Στην συνέχεια, αναπτύσσεται η διαδικασία που ακολουθήθηκε, τα προβλήματα που συναντήθηκαν και οι τρόποι αντιμετώπισής τους. Μετά την επίλυση παρουσιάζεται η αξιολόγηση των τελικών συστημάτων με την διεξαγωγή πειραμάτων. Τέλος παρατίθενται τρία παραρτήματα.

Το πρώτο αφορά την περιγραφή της δομής του λογισμικού που αναπτύχθηκε. Στο δεύτερο περιγράφονται τα αναπτυχθέντα κυκλώματα και οι συνδεσμολογίες των συστημάτων. Τέλος, το τρίτο είναι το εγχειρίδιο εγκατάστασης των λογισμικών και χρήσης των ρομπότ.

Στη μητέρα, στον πατέρα και στην αδερφή μου.

Ευχαριστίες

Αρχικά θέλω να ευχαριστήσω τον κ. Κ. Κυριακόπουλο για την ευκαιρία που μου έδωσε να εργαστώ στο εργαστήριο του υπό την επίβλεψή του και με πολύ καλή συνεργασία. Επίσης τους Μπάμπη Μπεχλιούλη, Παναγιώτη Βλαντή, Γιώργο Καρά, Γιώργο Ζωγόπουλο και Πάνο Μαράντο για την βοήθεια και τις συμβουλές που μου παρείχαν όποτε τους χρειάστηκα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η παρούσα διπλωματική εργασία αναλύει την αρχιτεκτονική και λειτουργία των mobile robots της εταιρίας Adept που έχει στην κατοχή του το εργαστήριο Αυτομάτου Ελέγχου και Ρυθμίσεως Μηχανών της Σχολής Μηχανολόγων Μηχανικών ΕΜΠ. Τα ρομπότ αυτά εξοπλίστηκαν με νέους μικροϋπολογιστές και μικρο-ελεγκτές (Arduino) και προγραμματίστηκαν εκ νέου χρησιμοποιώντας το ταχέως αναπτυσσόμενο Robot Operating System (ROS). Τα τελικά ρομποτικά συστήματα μπορούν να λειτουργήσουν αυτόνομα προγραμματιζόμενα εντός του ROS αλλά και χειροκίνητα μέσω χειριστηρίου ή άλλου τρόπου αποστολής εντολών κίνησης. Παρακάτω περιγράφονται τα mobile robots, το ROS και το Arduino βασικά εργαλεία για την υλοποίηση της εργασίας.

1.1 Mobile robot

Το mobile robot είναι μια αυτόματη μηχανή ικανή να κινηθεί στον περιβάλλοντα χώρο. Μπορούν να είναι αυτόνομα, να μην χρειάζονται δηλαδή εξωτερική καθοδήγηση, ή εναλλακτικά η κίνηση τους και οι ενέργειες που εκτελούν να καθορίζονται ανεξάρτητα από αυτά. Χρησιμοποιούνται ευρέως από την βιομηχανία και την εκπαίδευση έως τις στρατιωτικές εφαρμογές. Μπορούν να διακριθούν βάσει της μεθόδου κίνησής τους, δηλαδή μέσω ποδών, τροχών και τροχιών (ράγες).

Τα μη αυτόνομα ρομπότ πλοηγούνται ακολουθώντας πιστά εξωτερικές εντολές ενώ πολλές φορές μπορεί να ενσωματωθεί σε αυτό έλεγχος αποφυγής εμποδίων. Η χρήση τους είναι τις περισσότερες φορές απλή καθώς το δύσκολο έργο της πλοήγησης αναλαμβάνεται από εξωτερικό παράγοντα.

Από την άλλη τα αυτόνομα ρομπότ είναι προγραμματισμένα να εκτελούν συγκεκριμένες εργασίες και πορείες δίχως εξωτερική παρέμβαση. Για τον σκοπό αυτόν απαιτούν γνώση της θέσης τους στον χώρο (localization) καθώς και τρόπους προσέγγισης των στόχων τους που είναι προγραμματισμένα να πετύχουν-προσεγγίσουν. Τις περισσότερες φορές τα αυτόνομα ρομπότ

προσφέρουν επιλογές χειροκίνητης οδήγησης.

Το εργαστήριο έχει στην κατοχή του δίτροχα και τετράτροχα ρομπότ της εταιρίας Adept, τις εκδόσεις Pioneer 3DX και 2AT αντίστοιχα. Τα ρομπότ αυτά χρησιμοποιούνται για ερευνητικούς σκοπούς όπως αυτόνομη πλοήγηση, χαρτογράφηση και όραση. Είναι ικανά να φέρουν σημαντικό φορτίο και να λειτουργήσουν σε πολλών ειδών εδάφη.

Σχήμα 1.1: Pioneer 3DX

Σχήμα 1.2: Pioneer 2AT

1.2 Robot Operating System

Το Robot Operating System[1] (ROS) αποτελεί μία ευέλικτη πλατφόρμα ανάπτυξης εφαρμογών ρομποτικής, της οποίας σκοπός είναι η απλοποίηση της δημιουργίας πολύπλοκων ρομποτικών εφαρμογών. Αυτό επιτυγχάνεται μέσω πληθώρας βιβλιοθηκών και εργαλείων τα οποία απευθύνονται σε ευρύ φάσμα συστημάτων. Ταυτόχρονα κάνει εφικτή την επικοινωνία όλων των μελών ενός συστήματος ρομπότ χωρίς να είναι απαραίτητη η γνώση του τρόπου λειτουργίας τους. Η ανταλλαγή πληροφοριών είναι ανεξάρτητη της αρχιτεκτονικής των συστημάτων και υλοποιείται εύκολα. Γενική αρχή είναι η λειτουργία ενός ή περισσοτέρων κόμβων (nodes) που αναλαμβάνουν την εκτέλεση των διαφόρων λειτουργιών και υπολογισμών κάθε επιμέρους συστήματος. Οι κόμβοι μπορούν να είναι μεταξύ τους είτε εξαρτημένοι είτε όχι.

Για παράδειγμα ένας κόμβος μπορεί να υπολογίζει την τροχιά που πρέπει να ακολουθήσει το ρομπότ λαμβάνοντας δεδομένα από έναν ανεξάρτητο κόμβο παροχής ανάδρασης των ταχυτήτων του ρομπότ ενώ ταυτόχρονα ένας άλλος να αναλαμβάνει την δημοσίευση των μετρήσεων των αισθητήρων. Υπάρχει δηλαδή εύκολη επικοινωνία μεταξύ των κόμβων οι οποίοι μοιράζονται με γνωστή και ενιαία μορφή τις απαραίτητες πληροφορίες.

Το ROS παρέχει απλή επικοινωνία συστημάτων μέσω δικτύου, δίχως να απαιτείται ανάπτυξη αλγορίθμων υλοποίησης διαφόρων πρωτοκόλλων επικοινωνίας μεταξύ ηλεκτρονικών υπολογιστών. Αυτό σημαίνει ότι οι κόμβοι μπορούν να εκτελούνται είτε στον ενσωματωμένο υπολογιστή του ρομπότ είτε απομακρυσμένα σε άλλον υπολογιστή.

Η ανταλλαγή πληροφοριών στο ROS επιτυγχάνεται μέσω των θεμάτων (topics) στα οποία κάθε κόμβος μπορεί να δημοσιεύει ή να διαβάζει μηνύματα. Η πρόσβαση ενός κόμβου σε ένα topic είναι απλή και μόνη απαίτηση είναι η ύπαρξη συμβατού λογισμικού ROS και ύπαρξη δικτύου αν ο κόμβος λειτουργεί σε διαφορετικό σύστημα.

Το ROS προσφέρει πληθώρα πακέτων για βασικές λειτουργίες των ρομπότ αριθμώντας σήμερα πάνω από 3000 πακέτα ανοικτού κώδικα σε ένα ευρύ φάσμα εφαρμογών. Επιπρόσθετα, η υπάρχουσα κοινότητα προγραμματιστών και χρηστών συμβάλει στην αντιμετώπιση των περισσοτέρων προβλημάτων.

1.3 Arduino

Το Arduino[2] UNO είναι μια ανοικτού κώδικα πλακέτα που χρησιμοποιείται σε εφαρμογές όπως ανάπτυξης αυτοματισμών και ελέγχου. Ως προϊόν αποτελείται από δύο μέρη, το hardware (ηλεκτρονικά) και το software

[1] www.ros.com

[2] www.arduino.com

(IDE ανάπτυξης εφαρμογών, βιβλιοθήκη συναρτήσεων). Υπερέχει έναντι άλλων πλατφορμών ανάπτυξης ηλεκτρονικών εφαρμογών καθώς είναι εύκολα προγραμματίσιμο σε γλώσσα Wiring (παραλλαγή της C++) και παρέχει πληθώρα συναρτήσεων. Οι συναρτήσεις αυτές αφορούν από την υλοποίηση πρωτοκόλλων επικοινωνίας με άλλες ηλεκτρονικές συσκευές (I^2C, SPI) έως την οδήγηση stepper motors. Η πλακέτα περιλαμβάνει μικρο-ελεγκτή της εταιρίας AVR και στο Arduino UNO, που χρησιμοποιείται στην παρούσα εργασία, τον ελεγκτή Atmel328 χρονισμένο στα 16MHz. Επίσης, φέρει θύρα τροφοδοσίας 5V και θύρα USB για επικοινωνία με τον Η/Υ. Σημαντικό πλεονέκτημα αποτελεί το πλήθος των έτοιμων κυκλωμάτων (shields) τα οποία συνεργάζονται εύκολα με τον ελεγκτή και εφαρμόζουν γεωμετρικά στην πλακέτα.

Κεφάλαιο 2

Τεχνική Διατύπωση Προβλήματος

2.1 Περιγραφή Εγκαταστάσεων

Όπως αναφέρθηκε το εργαστήριο έχει στην κατοχή του δύο κατηγοριών mobile robots, τα δίτροχα (Pioneer 3DX) και τα τετράτροχα (Pioneer 2AT). Στο παρόν κεφάλαιο αναλύονται τα χαρακτηριστικά των οχημάτων όπως στην κατάσταση πριν την ανακατασκευή. Έπειτα, περιγράφονται τα νέα στοιχεία που χρησιμοποιήθηκαν και τέλος αναπτύσσεται σύντομα ο τρόπος λειτουργίας και οι κύριες απαιτήσεις κάθε στοιχείου που απαρτίζουν τα τελικά συστήματα.

2.1.1 Pioneer 3DX

Το Pioneer 3DX χρησιμοποιείται κυρίως για εσωτερικές εργασίες όπως χωρομέτρηση, χαρτογράφηση, αποφυγή εμποδίων και υπολογισμό πορείας. Έχει δύο κινητήριους τροχούς και έναν βοηθητικό ο οποίος μπορεί να περιστρέφεται γύρω από τον κατακόρυφο άξονα. Φέρει ενσωματωμένο ηλεκτρονικό υπολογιστή με λειτουργικό σύστημα Linux, δικτύωση μέσω Ethernet και Wifi και οκτώ sonar sensors. Μπορεί να αναπτύξει μέγιστη ταχύτητα ίση με 1,4m/s και να φέρει μέγιστο ωφέλιμο φορτίο ίσο με 23kg. Το όχημα φέρει έναν κινητήρα 9236-E349 της εταιρίας Pittmann σε κάθε τροχό. Στο σώμα του κινητήρα είναι ενσωματωμένος μειωτήρας στροφών με λόγο μείωσης 38,3. Επίσης, συμπεριλαμβάνει quadrature encoder με ανάλυση 500 παλμών ανά περιστροφή. Κάθε ρομπότ τροφοδοτείται από τρεις μπαταρίες 12V, 9Ah. Ο ενσωματωμένος ελεγκτής πραγματοποιεί αυτόματα έλεγχο ταχύτητας και παρέχει ανάδραση της εκτίμησης της θέσης (x,y,θ), της κατάστασης της μπαταρίας και των τιμών των αισθητήρων.

2.1.2 Pioneer 2AT

Το Pioneer 2AT είναι μια τετράτροχη πλατφόρμα που μπορεί να κινηθεί σε πολλών ειδών εδάφη, κυρίως όμως, χρησιμοποιείται σε εξωτερικούς χώρους για παρακολούθηση, όραση και χαρτογράφηση. Όπως και το 3DX έχει ενσωματωμένο Η/Υ παλαιότερης τεχνολογίας με δικτυακές επικοινωνίες Ethernet και Wifi. Μπορεί να αναπτύξει μέγιστη ταχύτητα ίση με 0.8m/s και μέγιστο ωφέλιμο φορτίο ίσο με 30kg. Έχει τέσσερις κινητήρες οι οποίοι είναι συζευγμένοι ανά δύο σε κάθε πλευρά του οχήματος. Έτσι, οι δύο τροχοί κάθε πλευράς έχουν κάθε στιγμή την ίδια γωνιακή ταχύτητα. Η μεταφορά της κίνησης γίνεται με ιμάντα. Οι κινητήρες είναι 9236-E204 και έχουν ενσωματωμένο μειωτήρα στροφών με λόγο μείωσης 65.5. Ένας από κάθε ζεύγος κινητήρων φέρει quadrature encoder με ανάλυση 100 παλμών ανά περιστροφή. Τροφοδοτείται από τρεις μπαταρίες 12V, 9Ah. Όπως και στο Pioneer 3DX, ο ενσωματωμένος ελεγκτής κάνει έλεγχο ταχύτητας και παρέχει ανάδραση.

2.1.3 Νέα συστήματα

Στα νέα συστήματα ο υπολογιστής αντικαθίσταται από τον μικρο-υπολογιστή Odroid U2. Ο ενσωματωμένος ελεγκτής αντικαθίσταται από τον μικροελεγκτή Arduino Uno του οποίου ρόλος είναι ο αυτόματος έλεγχος της ταχύτητας και παροχή της απολύτως απαραίτητης ανάδρασης. Για την οδήγηση των κινητήρων χρησιμοποιείται ένας DC motor driver που ελέγχεται μέσω παλμών PWM και μπορεί να παρέχει μέγιστο ρεύμα ανά κινητήρα ίσο με 15A. Από τα παλιά ηλεκτρονικά χρησιμοποιήθηκαν μόνο αυτά που ελέγχουν την τάση των μπαταριών καθώς και την φόρτισή τους. Έτσι, η τελική εγκατάσταση προκύπτει σημαντικά πιο απλή στην σύνθεση και έχει μικρότερο βάρος.

Παρακάτω αναλύονται τα τμήματα που χρησιμοποιούνται στα τελικά συστήματα.

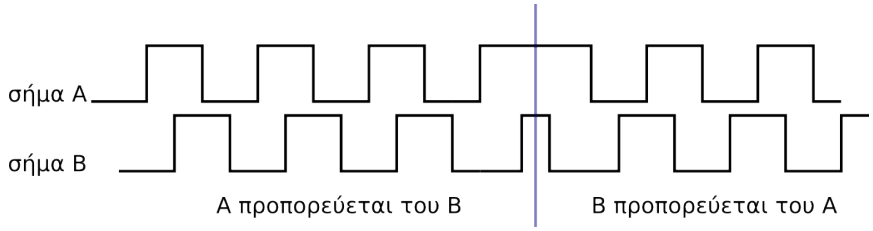
2.1.4 Περιγραφή εξαρτημάτων

Quadrature Encoder

Ο περιστροφικός encoder είναι μια ηλεκτρο-μηχανολογική συσκευή που μετατρέπει την γωνιακή θέση ή κίνηση μιας ατράκτου σε αναλογικό ή ψηφιακό σήμα. Οι οπτικοί quadrature encoders που φέρουν οι κινητήρες χρησιμοποιούνται για τον υπολογισμό της γωνίας περιστροφής της ατράκτου του κινητήρα. Μέσω αυτής, γνωρίζοντας τον χρόνο που έχει μεσολαβήσει υπολογίζεται η ταχύτητα περιστροφής. Παράγουν δύο τετραγωνικούς παλμούς οι οποίοι έχουν σταθερή διαφορά φάσης $90°$. Η διαδοχή των δύο σημάτων για δεδομένη φορά περιστροφής φαίνεται παρακάτω.

Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Η διαφορά φάσης των παλμών είναι σταθερή και ίση με $90/textdegree$. Στο σχήμα παρακάτω φαίνονται οι δύο παλμοί πριν και μετά την αλλαγή της κίνησης.

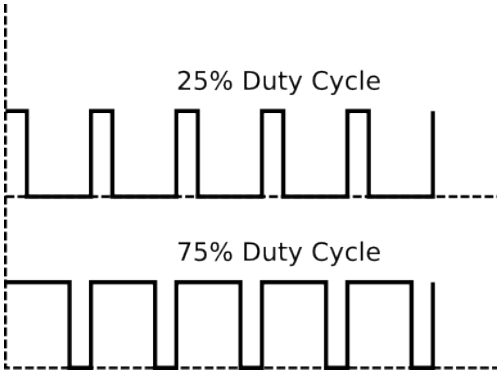


Έτσι, συγκρίνοντας τα δύο σήματα προσδιορίζεται η φορά περιστροφής της ατράκτου. Συνεπώς, για την μέτρηση της γωνίας στροφής αρκεί η καταμέτρηση των παλμών του ενός σήματος και για τη φορά περιστροφής απλά ο έλεγχος του δεύτερου ως προς το πρώτο.

Κάθε encoder έχει 4 καλώδια, τα δύο είναι για την τροφοδοσία με ηλεκτρική ενέργεια (5V) και τα άλλα δύο φέρουν τους παραγόμενους παλμούς.

DC Motor Driver

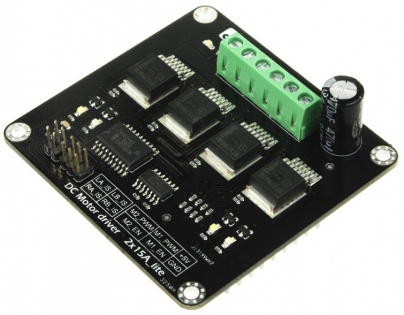
Ο DC motor driver χρησιμοποιείται για την παροχή της απαραίτητης ισχύος στους κινητήρες, καθώς ο μικρο-ελεγκτής αδυνατεί να την παρέχει. Δέχεται ως είσοδο δύο σήματα για κάθε κινητήρα, ένα που προσδιορίζει την φορά περιστροφής και είναι είτε 5V είτε 0V, και ένα άλλο που είναι παλμός σταθερής συχνότητας διαμορφωμένος κατά πλάτος (PWM), στον οποίο έχει διαμορφωθεί το duty cycle, δηλαδή το διάστημα που είναι διαφορετικός από 0. Στο σχήμα παρακάτω φαίνονται δύο σήματα PWM με ίδια συχνότητα αλλά διαφορετικό duty cycle.



Η οδήγηση κάθε κινητήρα γίνεται από δύο H-Bridge κυκλώματα (BTS7960).

H-Bridge. Είναι ένα κύκλωμα που επιτρέπει την εφαρμογή τάσης σε ένα φορτίο και στις δύο κατευθύνσεις. Χρησιμοποιείται σε εφαρμογές ρομποτικής για την οδήγηση κινητήρων συνεχούς τάσης αποτελούμενο από τέσσερις διακόπτες (solidstate ή μηχανικοί). Στον οδηγό που χρησιμοποιείται στην παρούσα εργασία οι διακόπτες αυτοί είναι solidstate. Με συνδυασμούς διακοπτών ανά ζεύγη επιτυγχάνεται περιστροφή του κινητήρα και στις δύο κατευθύνσεις, πέδηση του κινητήρα και ελεύθερη περιστροφή του. Παρατηρώντας την λειτουργία του κυκλώματος συμπεραίνεται ότι το σήμα εξόδου στους κινητήρες έχει την ίδια μορφή (συχνότητα) με το σήμα εισόδου (παλμός PWM). Τα δύο σήματα έχουν την ίδια συχνότητα, διαφορετική όμως μέγιστη τιμή, με της εξόδου να είναι ίση με την τάση τροφοδοσίας του κυκλώματος.

Τέλος, ο driver απαιτεί και την παροχή ισχύος από 4.8V έως 35V. Στη συγκεκριμένη εφαρμογή τροφοδοτείται με την τάση των μπαταριών, δηλαδή 12V. Ο παλμός PWM που δέχεται μπορεί να έχει μέγιστη συχνότητα ίση με 25KHz.



Odroid U2

Το Odroid U2 είναι ένας ολοκληρωμένος υπολογιστής σε πολύ μικρό μέγεθος και λειτουργεί με Λ/Σ Linux ή Android. Στη συγκεκριμένη εφαρμογή λειτουργεί με Λ/Σ Linux Ubuntu 12.04. Έχει μεγάλη υπολογιστική ισχύ αφού φέρει επεξεργαστή τεσσάρων πυρήνων χρονισμένο στα 1.7GHz και 2GB RAM. Επίσης φέρει θύρες USB για επικοινωνία με περιφερειακές συσκευές και θύρα Ethernet για σύνδεση σε δίκτυο. Είναι ικανό να αναλάβει την επικοινωνία με τον μικρο-ελεγκτή και ταυτόχρονα επικοινωνία με άλλο υπολογιστή στο δίκτυο. Τέλος, υπάρχει θύρα HDMI για σύνδεση σε οθόνη.

2.2 Απαιτήσεις

Αρχικά, τα συστήματα πρέπει να έχουν τις ίδιες ή καλύτερες δυνατότητες με την παλαιά μορφή τους. Οι δυνατότητες αυτές αφορούν κυρίως τη μέγιστη ταχύτητα, την απόκριση αλλά και τον θόρυβο. Επίσης, πρέπει να υπάρχει συνδεσιμότητα με το δίκτυο (wifi) έτσι ώστε να είναι δυνατός ο απομακρυσμένος έλεγχός τους και η αυτόνομη λειτουργία τους.

Ο ενσωματωμένος ελεγκτής πρέπει να ελέγχει αυτόματα και ανεξάρτητα από τον υπολογιστή την ταχύτητα του οχήματος. Ταυτόχρονα θα πρέπει να είναι εύκολη η επιπλέον προσθήκη αισθητήρων στο σύστημα και η αποστολή των τιμών τους μέσω του ROS. Τα τελικά συστήματα πρέπει να είναι φιλικά προς τον χρήστη και ο χειρισμός τους να μην απαιτεί γνώση της αρχιτεκτονικής τους.

Αυτό προϋποθέτει ότι με απλή αποστολή μηνύματος μέσω του συστήματος ROS το ρομπότ θα εκτελεί την επιθυμητή κίνηση. Το μήνυμα αυτό μπορεί να περιλαμβάνει τριών ειδών εντολές. Η πρώτη κατηγορία αφορά την γραμμική και γωνιακή ταχύτητα του κέντρου μάζας του οχήματος. Η δεύτερη, τις περιστροφικές ταχύτητες κάθε τροχού. Τέλος, η τρίτη αφορά την απευθείας αποστολή εντολών στον driver των κινητήρων. Οι εντολές αυτές αντιστοιχούν στην ροπή του κάθε κινητήρα. Όπως οι πληροφορίες εισόδου στο ρομπότ έτσι και οι πληροφορίες εξόδου από αυτό, δηλαδή η ανάδραση, πρέπει είναι απλές και εύκολα αναγνώσιμες. Η ανάδραση πρέπει να παρέχει τις απαραίτητες πληροφορίες έτσι ώστε να υπολογίζεται η οδομετρία του οχήματος και οι ταχύτητες του κέντρου μάζας.

Επίσης, είναι επιθυμητή η εύκολη μεταβολή των κερδών του ελέγχου μέσω μηνύματος του ROS, έτσι ώστε να μην απαιτείται τροποποίηση του κώδικα του ελεγκτή.

Όλα τα παραπάνω κάνουν τα καινούρια συστήματα συνεργάσιμα τόσο μεταξύ τους όσο και με άλλα ρομπότ αγνώστου αρχιτεκτονικής.

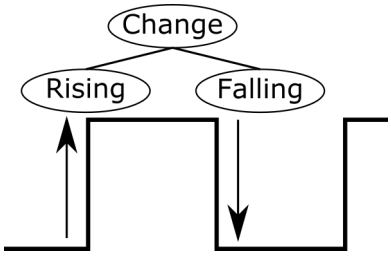
Κεφάλαιο 3

Επίλυση: Ανάπτυξη Συστήματος Pioneer 3DX

Αρχικά αφαιρέθηκαν τα παλιά ηλεκτρονικά και διαχωρίστηκαν τα καλώδια των κινητήρων και των encoders. Έτσι, απέμειναν μόνο οι κινητήρες και η μονάδα παροχής ηλεκτρικής ισχύος.

3.1 Διαχείριση Encoders

Πρώτη εργασία ήταν η ανάπτυξη του αλγορίθμου μέτρησης των encoders. Δεδομένου ότι ο ελεγκτής πρέπει να είναι σε θέση να καταμετρά σωστά τους παλμούς των encoders και να προσδιορίζει την φορά περιστροφής επιλέχθηκε η αποδοτική τεχνική της σχεδίασης μίας λειτουργίας interrupt στον ελεγκτή που θα καταμετρά τους παλμούς. Έτσι, επιτυγχάνεται η ελάχιστη απώλεια παλμών, αφού ο ελεγκτής διακόπτει την εργασία που εκτελεί εκείνη την στιγμή που το σήμα εισόδου μεταβάλλεται σύμφωνα με τον προκαθορισμένο τρόπο (triggering). Δίνεται δηλαδή μέγιστη προτεραιότητα στον κώδικα υπολογισμού των παλμών του encoder. Μετά την ολοκλήρωση των εργασιών που γίνονται στο interrupt ο μικρο-ελεγκτής συνεχίζει τη ροή του προγράμματος από το σημείο που είχε μείνει. Οι δυνατές μεταβολές του σήματος για ενεργοποίηση του interrupt (triggering) είναι αλλαγή (Change), αναρρίχηση (Rising) και πτώση (Falling) του σήματος.



Όπως είναι προφανές η μέθοδος triggering με απλή αλλαγή του σήματος προσφέρει την καλύτερη ανάλυση αφού αντιλαμβάνεται αμέσως την αλλαγή

φοράς περιστροφής. Αυτό συμβαίνει γιατί την στιγμή αυτή της αλλαγής φοράς το σήμα που προπορευόταν, υστερεί του άλλου και ο μικρο-ελεγκτής αντιλαμβάνεται αμέσως την μεταβολή αυτή. Επειδή όμως ο encoder έχει μεγάλη ανάλυση χρησιμοποιήθηκε σαν triggering η αναρρίχηση του ενός παλμού. Από την στιγμή που ενεργοποιηθεί η interrupt υπορουτίνα προσδιορίζεται η φορά περιστροφής και αναλόγως αυξάνεται ή μειώνεται κατά μία μονάδα ο μετρητής των παλμών.

Σημαντικό πρόβλημα που παρατηρήθηκε ήταν η αδυναμία σε επίπεδο λογισμικού να καταγραφούν σωστά οι παλμοί στις πολύ γρήγορες εναλλαγές της φοράς περιστροφής. Αιτία του προβλήματος είναι η χρήση έτοιμων συναρτήσεων που προσφέρει το περιβάλλον του Arduino (digitalRead()) οι οποίες απαιτούν περισσότερο χρόνο από την απευθείας ανάγνωση της κατάστασης της εισόδου. Έτσι, η ανάγνωση των εισόδων που γίνεται στις συναρτήσεις των interrupts έγιναν με απευθείας ανάγνωση (direct port manipulation) μέσω εντολής σε γλώσσα AVR C++. Αυτή η αλλαγή βελτίωσε σημαντικά την αποδοτικότητα του κώδικα.

Παρόλο που το παραπάνω πρόβλημα επιλύθηκε σε επίπεδο λογισμικού στη συνέχεια παρατηρήθηκε αδυναμία καταμέτρησης όλων των παλμών όταν και οι δύο τροχοί περιστρέφονται με ταχύτητα κοντά στην μέγιστη ανεξάρτητα από την φορά περιστροφής. Το πρόβλημα εξακριβώθηκε απενεργοποιώντας το interrupt του ενός τροχού διατηρώντας σταθερή γωνιακή ταχύτητα και στους δύο τροχούς, με αποτέλεσμα η μετρούμενη ταχύτητα περιστροφής να αυξάνεται κατά ένα σημαντικό ποσοστό. Δεδομένου ότι η μέγιστη ταχύτητα περιστροφής του τροχού είναι

$$\omega_{max} = \frac{v_{max}}{r} = \frac{1.4[\frac{m}{s}]}{0.095[m]} = 14.74[\frac{rad}{s}] = 2.35[\frac{rev}{s}],$$

η μέγιστη συχνότητα παλμών είναι

$$f_{max} = \omega_{max} enc_{res} n = 2.35[\frac{rev}{s}]500[\frac{1}{rev}]38.3 = 44938Hz.$$

Με enc_{res} την ανάλυση του encoder ανά περιστροφή του άξονα του κινητήρα και n τον λόγο μετάδοσης της κίνησης. Παρατηρείται ότι η συχνότητα των παλμών είναι σχετικά μεγάλη και τα περιθώρια που έχει ο ελεγκτής να λάβει σωστές τιμές, δηλαδή το χρονικό διάστημα που μεσολαβεί μέχρι να αλλάξει η τιμή, είναι στενά. Η αδυναμία του ελεγκτή οφείλεται σε δύο παράγοντες.

Πρώτον, είναι πιθανό δύο παλμοί να έρχονται ταυτόχρονα στον ελεγκτή και αυτός μέχρι να τους χειριστεί να χάνει πληροφορίες. Αυτή η περίπτωση δεν επηρεάζει τόσο συχνά καθώς ο ελεγκτής έχει εσωτερική προτεραιότητα όταν ορίζονται περισσότερες από μία interrupts.

Δεύτερη, και πιο συχνή περίπτωση είναι το διάστημα που απαιτείται για να εισέλθει η ροή του προγράμματος στην συνάρτηση υπολογισμού (Interrupt ISR) να είναι τόσο ώστε η τιμή του δεύτερου σήματος του encoder να έχει αλλάξει. Έτσι, μια ψευδής μέτρηση, ενώ το ένα σήμα μπορεί να προπορεύεται του άλλου, θα δείξει το αντίθετο, δηλαδή ότι υστερεί. Αυτό οφείλεται

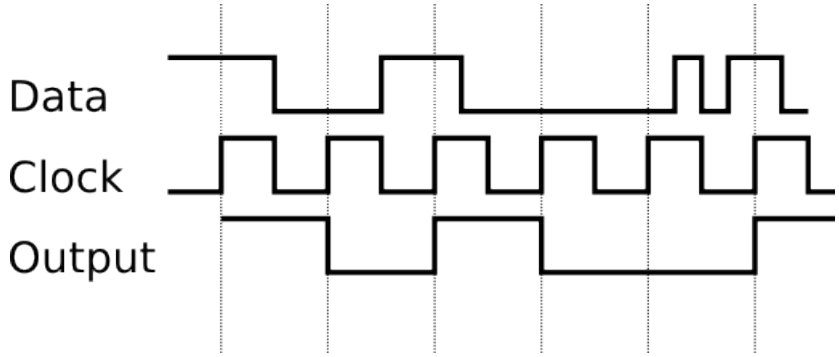
στο γεγονός ότι έχει οριστεί ως κανόνας triggering η αναρρίχηση (Rising) του σήματος. Θεωρείται δηλαδή δεδομένο ότι τη στιγμή που εισέρχεται στη interrupt ISR το σήμα είναι λογικό HIGH. Αν καθυστερήσει η είσοδος στη συνάρτηση αυτή και το δεύτερο σήμα έχει μεταβληθεί, οι πράξεις που εκτελούνται ελέγχουν το δεύτερο ως προς το πρώτο σήμα με την πιθανώς ψευδή θεώρηση ότι το κύριο σήμα είναι HIGH.

Υπάρχουν έτοιμα κυκλώματα που αναλαμβάνουν την διαχείριση των encoders αλλά το κόστος τους είναι μεγάλο και πολλές φορές ίσως είναι δυσκολότερο να προσαρμοστούν στο σύστημα από την ανάπτυξη ενός καινούριου συστήματος.

Μία δυνατή λύση θα ήταν η αλλαγή του μικρο-ελεγκτή με έναν ισχυρότερο, που θα απαιτούσε δηλαδή λιγότερο χρόνο για χειρισμό της interrupt ISR. Αυτή δεν προτιμήθηκε κυρίως για λόγους συμβατότητας με το σύστημα ROS και συγκεκριμένα με το πακέτο rosserial-arduino που χρησιμοποιείται αφού ο πλέον αξιόπιστος ελεγκτής (για χρήση του rosserial-arduino) είναι το Arduino UNO, αλλά και για λόγους κόστους.

Άλλη πιθανή λύση θα ήταν η χρησιμοποίηση ως μέθοδο triggering την αλλαγή του σήματος και σύγκριση των δύο σημάτων (δηλαδή δύο αναγνώσεις θυρών) κάθε φορά που εισέρχεται στην υπορουτίνα. Αυτή η λύση δεν θα ωφελούσε καθώς διπλασιάζεται η συχνότητα εισόδου στην συνάρτηση interrupt αλλά και γίνονται διπλές αναγνώσεις θυρών. Δηλαδή, το πιο πιθανό είναι να επιβάρυνε την λειτουργία του ελεγκτή και να μεγένθυνε το πρόβλημα.

Το πρόβλημα των μικρών χρονικών περιθωρίων για τον έλεγχο της φοράς περιστροφής επιλύθηκε με την χρήση κυκλώματος D-flip-flop με ρολόι χρονισμού. Το ολοκληρωμένο αυτό κύκλωμα έχει δύο εισόδους, μία για τα δεδομένα (data) και μία για το ρολόι (clock). Λειτουργία του είναι να παρακολουθεί την μία είσοδο και να αποθηκεύει την τιμή της κάθε φορά που το σήμα στο clock μεταβαίνει από 0 σε 1. Έτσι, στην είσοδο clock συνδέεται το ίδιο σήμα που λειτουργεί ως trigger στον ελεγκτή, και στο data το σήμα προσδιορισμού της φοράς περιστροφής, αυτού δηλαδή που αποθηκεύεται η κατάσταση του. Η λύση αυτή με σταθερή φορά περιστροφής εγγυάται τον ορθό προσδιορισμό της αφού το σήμα που την καθορίζει είναι σταθερό. Στην περίπτωση αλλαγής της φοράς περιστροφής ο ελεγκτής λαμβάνει ψευδείς μετρήσεις για χρονικό διάστημα ίσο με την περίοδο του σήματος που λειτουργεί ως trigger, το οποίο ισοδυναμεί με έναν παλμό. Το σχήμα απεικονίζει ένα παράδειγμα με τη λειτουργία του D-flip-flop.



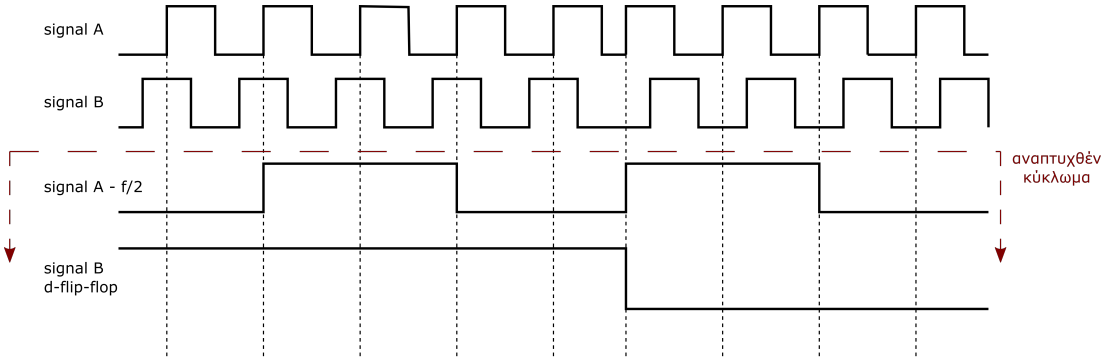
Παράλληλα αναπτύχθηκε κύκλωμα υποδιαίρεσης της συχνότητας των παλμών (ισοδύναμα της ανάλυσης) του encoder. Αυτό είναι αποδεκτό αφού όπως προαναφέρθηκε η ανάλυση είναι σχετικά μεγάλη. Για την ακρίβεια κάθε παλμός του encoder αντιστοιχεί σε $\theta = \frac{360}{500 \cdot 38,3} = 0.01879$ μοίρες στροφής της ρόδας. Για τον σκοπό αυτό χρησιμοποιήθηκε ένας binary counter. Ο binary counter στην απλή υλοποίησή του έχει μία είσοδο και τέσσερις εξόδους, οι οποίες αντιστοιχούν στα bits του μετρητή. Κάθε φορά που το σήμα εισόδου μεταβαίνει από 0 σε 1 ο δυαδικός αριθμός που αντιστοιχεί στα 4 bits αυξάνεται και στο δεκαδικό σύστημα η αύξηση αυτή ισούται με μία μονάδα. Επιστρέφει δηλαδή σε δυαδικό αριθμό τους παλμούς που έχει λάβει. Παρόλα αυτά δεν χρησιμοποιήθηκε η λειτουργία καταμέτρησης των παλμών, αλλά η ιδιότητα του να υποδιαιρεί την συχνότητα του σήματος εισόδου. Είναι εύκολα παρατηρήσιμη η διαδοχή των άσσων και μηδενικών στις εξόδους του μετρητή.

bit_3	bit_2	bit_1	bit_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1
0	0	0	0
0	0	0	1

Από την παρατήρηση αυτή προκύπτει ότι το πρώτο bit (bit_0) διαιρεί την συχνότητα εισόδου με το 2, το δεύτερο με το 4, το τρίτο με το 8 και το τέταρτο με το 16. Στην περίπτωση αυτή είναι αρκετός ο υποδιπλασιασμός

της συχνότητας αφού και ο ελεγκτής ελαφρύνεται και η ανάλυση του encoder παραμένει σε ικανοποιητικά επίπεδα.

Σε αυτό το σημείο έχουν χρησιμοποιηθεί δύο ολοκληρωμένα κυκλώματα. Το αποτέλεσμα της λειτουργίας τους είναι δύο σήματα. Το ένα είναι ο κύριος παλμός του encoder με διαιρεμένη συχνότητα (έξοδος binary coutner) που τροφοδοτεί την θύρα interrupt του ελεγκτή και το άλλο είναι το σήμα που χρησιμοποιείται για τον προσδιορισμό της φοράς περιστροφής (έξοδος d-flip-flop), το οποίο όταν δεν μεταβάλλεται η φορά είναι σταθερό. Όταν αλλάξει η φορά αλλάζει απλά είτε από 0 σε 1 είτε αντίστροφα. Στο παρακάτω σχήμα φαίνεται η σχέση των σημάτων αυτών.



3.2 Επικοινωνία με Motor Driver

Στη συνέχεια αναπτύχθηκε η επικοινωνία του μικρο-ελεγκτή με τον motor driver. Όπως προαναφέρθηκε, ο τελευταίος απαιτεί δύο σήματα για κάθε κινητήρα. Το ένα προσδιορίζει τη φορά περιστροφής και περιορίζεται στις λογικές τιμές 0 και 1. Τα χαρακτηριστικά του δεύτερου που αφορούν την τάση που στέλνει στον κινητήρα είναι πιο δύσκολα να προσδιοριστούν. Αυτό οφείλεται στο μεγάλο εύρος συχνοτήτων των παλμών (PWM) που μπορεί να παράξει ο μικρο-ελεγκτής και κυμαίνονται από 50Hz έως 32500Hz. Η συχνότητα επιδρά στην ικανότητα παραγωγής ροπής από τον κινητήρα αλλά και στον παραγόμενο θόρυβο. Πρέπει επομένως να υπολογιστεί το βέλτιστο εύρος συχνοτήτων των παλμών. Ο μικρο-ελεγκτής παρέχει αυτή την δυνατότητα μεταβάλλοντας την συχνότητα του timer που ελέγχει τις εξόδους του. Έχει έξι εξόδους που μπορούν να παράγουν PWM παλμούς, τις 3,5,6,9,10 και 11. Για κάθε ζεύγος αντιστοιχεί και άλλος timer. Επειδή ο timer0 που χειρίζεται τις δύο πρώτες ελέγχει επίσης και τις συναρτήσεις υπολογισμού και χειρισμού του χρόνου, μια μεταβολή της προκαθορισμένης συχνότητας έχει αρνητικές επιδράσεις. Έτσι, χρησιμοποιήθηκαν οι έξοδοι που λειτουργούν με τον timer1, δηλαδή οι 9,10.

Αρχικά δοκιμάστηκε η προκαθορισμένη συχνότητα του ελεγκτή που είναι 490Hz. Η επιλογή αυτή δεν ήταν αποδεκτή παρόλο που η απόκριση του κινητήρα όσον αφορά την ροπή ήταν ικανοποιητική, αφού ο κινητήρας παρήγαγε πολύ θόρυβο. Έτσι, σταδιακά αυξανόταν η συχνότητα και παρόλο

που παρατηρήθηκε ότι ο θόρυβος μειωνόταν, ο driver απαιτούσε παλμούς με όλο και μεγαλύτερο duty cycle έτσι ώστε να παραχθεί αρκετή ροπή για να ξεκινήσει να περιστρέφεται.

Το πρόβλημα αυτό επιλύθηκε με την χρήση ενός MKT πυκνωτή με χωρητικότητα 1μF παράλληλα με τον κινητήρα. Αυτό που επετεύχθη ήταν η φόρτιση του πυκνωτή με αποτέλεσμα την εξομάλυνση της τάσης που δέχεται ο κινητήρας που με την σειρά του οδήγησε στην ανάπτυξη ροπής από αυτόν ακόμα και με παλμούς μικρού duty cycle. Επελέχθη αυτός ο τύπος πυκνωτή καθώς δεν έχει συγκεκριμένη πολικότητα πράγμα απαραίτητο για την αλλαγή της φοράς περιστροφής του κινητήρα. Παράλληλα με αυτή τη λύση για την μεγαλύτερη μείωση του θορύβου επιλέχθηκε συχνότητα κοντά στα όρια της ακουστικής δηλαδή τις τάξης των 20000Hz.

3.3 Επικοινωνία με Η/Υ

Σε αυτό το σημείο είναι πλήρως αξιοποιήσιμοι οι encoders και κάθε στιγμή καταγράφονται αξιόπιστα οι παλμοί που παράγονται. Επόμενο βήμα είναι η ανάπτυξη του κώδικα στον ελεγκτή για την επικοινωνία με τον υπολογιστή (αποστολή και παραλαβή μηνυμάτων). Όπως προαναφέρθηκε σημαντικό πλεονέκτημα των ρομπότ είναι η επικοινωνία τους με το σύστημα ROS. Υπάρχει ένα πακέτο συναρτήσεων που βοηθούν στην εύκολη επικοινωνία μεταξύ Arduino και Η/Υ και ονομάζεται rosserial. Το πακέτο αυτό παρέχει συναρτήσεις που επιτρέπουν την δημιουργία topics μέσω του Arduino και δημοσίευση πληροφοριών σε αυτά. Τα topics που απαιτούνται είναι τρία.

Το ένα αφορά την είσοδο εντολών κίνησης προς τον ελεγκτή, το άλλο την παροχή πληροφοριών προς τον Η/Υ και το τελευταίο την είσοδο εντολών αλλαγής των κερδών του ελέγχου. Λόγω περιορισμένης εσωτερικής μνήμης του Arduino τα μηνύματα αυτά πρέπει να έχουν το ελάχιστο μέγεθος. Αυτό σημαίνει ότι πρέπει να παρέχονται οι απολύτως απαραίτητες πληροφορίες, απ'τις οποίες όμως προκύπτουν τα υπόλοιπα απαραίτητα μεγέθη.

3.3.1 Μήνυμα Εντολής Κίνησης

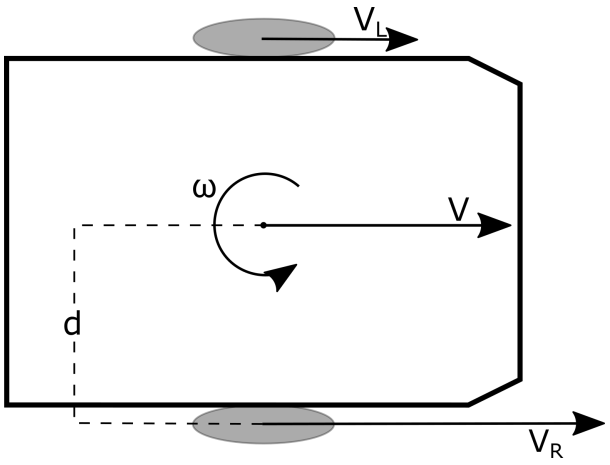
Το μήνυμα αυτό παρέχει τις εντολές κίνησης του ρομπότ. Η πρώτη μεταβλητή του προσδιορίζει το mode λειτουργίας του ρομπότ, δηλαδή τον έλεγχο των ταχυτήτων του κέντρου μάζας (mode=1), τον έλεγχο περιστροφικής ταχύτητας κάθε τροχού (mode=2) και τέλος τον απευθείας προσδιορισμό της τάσης των κινητήρων (mode=3). Οι άλλες δύο μεταβλητές παρέχουν την πληροφορία ανάλογα με το mode που έχει επιλεγεί. Ο ελεγκτής αρχικά διαβάζει το mode που περιλαμβάνεται στο μήνυμα και αναλόγως προσδιορίζει την ταχύτητα-στόχο για τον έλεγχο (στην περίπτωση 1 και 2) ή δίνει απευθείας εντολή στον motor driver (περίπτωση 3) μέσω των τιμών των δύο άλλων μεταβλητών. Στην πρώτη περίπτωση δίνονται η επιθυμητή γραμμική και

περιστροφική ταχύτητα του κέντρου μάζας. Οι ταχύτητες αυτές συνδέονται με τις γραμμικές ταχύτητες των τροχών μέσω των παρακάτω σχέσεων.

$$v = \frac{v_R + v_L}{2}$$

$$\omega = \frac{v_R - v_L}{2d}$$

Όπου d η απόσταση κάθε τροχού από το κέντρο μάζας, όπως φαίνεται στο σχήμα.



Έτσι, οι ταχύτητες κάθε τροχού είναι

$$v_R = v + \omega d$$

$$v_L = v - \omega d$$

Οι υπολογισμοί αυτοί είναι απαραίτητοι καθώς όπως θα αναλυθεί στην συνέχεια ο έλεγχος αφορά την ταχύτητα κάθε τροχού. Στις δύο άλλες περιπτώσεις δεν απαιτείται κάποιος υπολογισμός για τον προσδιορισμό των μεγεθών ελέγχου του ρομπότ αφού όλες οι αναγκαίες πληροφορίες για τον έλεγχο περιλαμβάνονται στο μήνυμα.

Για την επικοινωνία Arduino - Η/Υ αναπτύχθηκε κόμβος (στον Η/Υ) για την δημοσίευση του παραπάνω μηνύματος στο αντίστοιχο topic. Στα πλαίσια της διπλωματικής εργασίας αναπτύχθηκαν κόμβοι που αναλαμβάνουν αυτή την εργασία δεχόμενοι πληροφορίες από χειριστήριο ή από αλγόριθμο πλοήγησης σε συγκεκριμένο στόχο.

3.3.2 Μήνυμα Ανάδρασης

Μέσω αυτού του μηνύματος παρέχονται από τον ελεγκτή οι παλμοί κάθε encoder, τα σήματα PWM προς τον motor driver, το χρονικό διάστημα μεταξύ κάθε κύκλου υπολογισμών (looptime) και οι τιμές ανάγνωσης των αναλογικών εισόδων των μετρητικών οργάνων. Από τα παραπάνω μπορούν να υπολογιστούν άλλα χρήσιμα δεδομένα όπως η ταχύτητα του κέντρου μάζας, η οδομετρία κ.α.

3.3.3 Μήνυμα αλλαγής κερδών ελέγχου

Με το μήνυμα αυτό παρέχεται η δυνατότητα αλλαγής των κερδών του PID ελέγχου ενώ το ρομπότ βρίσκεται σε λειτουργία. Περιλαμβάνει έξι μεταβλητές, τρεις για κάθε κινητήρα (Kp, Ki και Kd). Η αλλαγή των κερδών δεν είναι μόνιμη αφού τα κέρδη επαναφέρονται στην προεπιλεγμένη τους τιμή κάθε φορά που απενεργοποιείται. Για την μόνιμη αλλαγή των κερδών απαιτείται αλλαγή στον κώδικα του ελεγκτή.

3.4 Ελεγκτής Ταχύτητας

Για τον έλεγχο της ταχύτητας αναπτύχθηκε ελεγκτής PI που ελέγχει την ταχύτητα περιστροφής των κινητήρων και χρησιμοποιείται για τα δύο πρώτα modes και πραγματοποιείται σε σταθερά χρονικά διαστήματα. Στην αρχή κάθε κύκλου υπολογισμών υπολογίζονται η ταχύτητα περιστροφής κάθε τροχού και με βάση την ταχύτητα στόχο που έχει τεθεί προκύπτει η κατάλληλη τιμή του duty cycle του παλμού προς τον οδηγό των κινητήρων. Η εξίσωση του PI ελεγκτή σε συνεχή μορφή είναι:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau$$

με $e(t)$ το σφάλμα.
Και σε διακριτή μορφή:

$$u[k] = k_p e[k] + k_i I_e[k]$$

με

$$I_e[k] = I_e[k-1] + e[k] \cdot dt$$

Δεδομένου ότι η έξοδος του ελεγκτή είναι φραγμένη στο διάστημα $[-255, 255]$ προστέθηκε κριτήριο κορεσμού του αθροίσματος του ολοκληρωτή για αποφυγή ανεξέλεγκτης αύξησής του όταν το όχημα δεν μπορεί να αναπτύξει την ζητούμενη ταχύτητα. Τα άκρα του διαστήματος αντιστοιχούν στην μέγιστη τιμή που μπορεί να πάρει το duty cycle του παλμού. Έτσι, όταν η τιμή του $u[k]$ βρίσκεται εκτός ορίων περιορίζεται η τιμή του ολοκληρώματος ώστε το $u[k]$ να παραμένει ίσο με την τιμή στο άκρο του διαστήματος.

Παράλληλα λόγω της μη γραμμικής λειτουργίας του motor driver (με συνέπεια την παραγωγή ροπής από τους κινητήρες) ο έλεγχος PI γίνεται έλεγχος P όταν η ταχύτητα περιστροφής είναι μικρή (κοντά στην ακινησία). Αυτό για να μην παρατηρούνται απότομες αυξήσεις της εξόδου του ελέγχου.

3.5 Σύνοψη λειτουργίας μικρο-ελεγκτή

Ο ελεγκτής έχει προγραμματιστεί έτσι ώστε οι υπολογισμοί να εκτελούνται σε σταθερό χρονικό διάστημα (loop time). Αν ο χρόνος που πέρασε από την

προηγούμενη είσοδο στον βρόχο ελέγχου είναι ίσος με το loop time, τότε εισέρχεται πάλι.

Ακολουθεί ο υπολογισμός της ταχύτητας περιστροφής κάθε τροχού με χρήση του αριθμού των παλμών των encoders. Η ταχύτητα αυτή ισούται με

$$\omega = \frac{enc_{count}}{enc_{res} \cdot \Delta t} \cdot 2\pi[\frac{rad}{s}]$$

με enc_{count} το πλήθος παλμών που καταμετρήθηκαν από την προηγούμενη είσοδο στον βρόχο ελέγχου και enc_{res} την ανάλυση του encoder, δηλαδή τους παλμούς ανά περιστροφή. Αυτό το μέγεθος μπορεί να είναι είτε ανά περιστροφή της ατράκτου του κινητήρα είτε ανά περιστροφή του τροχού. Η τελευταία ισούται με την ανάλυση ανά περιστροφή της ατράκτου του κινητήρα πολλαπλασιασμένη με τον λόγο μείωσης των τροχών. Τέλος, το Δt είναι ίσο με το loop time.

Στη συνέχεια η ταχύτητα μετατρέπεται σε γραμμική ταχύτητα μέσω της σχέσης

$$u = \omega \cdot r[\frac{m}{s}]$$

με r την ακτίνα του τροχού.

Επόμενο βήμα είναι ο υπολογισμός του σήματος ελέγχου των κινητήρων δηλαδή το duty cycle του παλμού προς τον οδηγό των κινητήρων μέσω του PI ελέγχου. Ανάλογα με το πρόσημο της εξόδου καθορίζεται η φορά περιστροφής του κινητήρα, δηλαδή το σήμα καθορισμού φοράς που θα σταλεί στον οδηγό και φράσσεται το duty cycle εντός του διαστήματος $[0, 255]$.

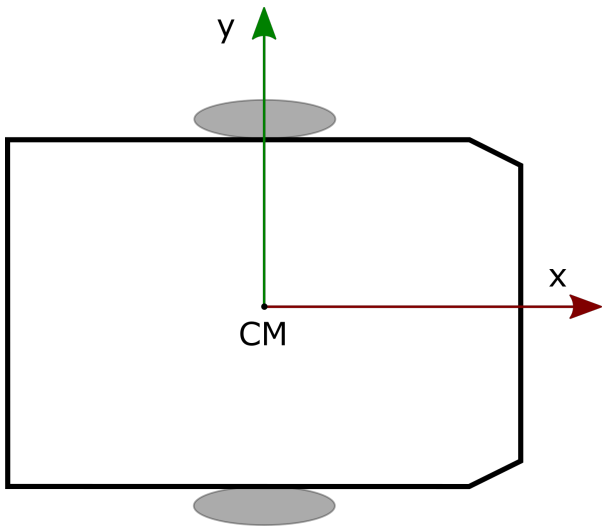
Τέλος ακολουθεί η ανάγνωση των αναλογικών τιμών των αισθητήρων και η αποστολή του μηνύματος ανάδρασης στον Η/Υ το οποίο αναλύθηκε παραπάνω. Απομένει ο ορισμός των μεταβλητών όπου αποθηκεύονται ο αριθμός των καταμετρημένων παλμών των encoders ίσες με μηδέν.

Καθ'όλη την παραπάνω διαδικασία η λειτουργία των interrupts του ελεγκτή είναι ενεργοποιημένη, με αποτέλεσμα κάθε φορά που έρχεται νέος παλμός να διακόπτει την λειτουργία του και να καταμετρά τον αριθμό τους.

Η χρήση των πυκνωτών παράλληλα των κινητήρων για τα οφέλη που αναφέρθηκαν αλλά και η υψηλή συχνότητα των παλμών PWM επηρεάζουν τον ομαλό έλεγχο της ταχύτητας των κινητήρων. Παρατηρήθηκε ότι αυξάνοντας το χρονικό διάστημα ελέγχου η δυνατότητα ελέγχου στις χαμηλές ταχύτητες περιστροφής γίνεται πιο δύσκολος. Από την άλλη όμως ο μικρο-ελεγκτής είναι σε θέση να καταμετρά πιο ικανοποιητικά τους παλμούς σε κάθε κύκλο. Ως βέλτιστη λύση βρέθηκε ότι το χρονικό διάστημα ελέγχου πρέπει να είναι μικρότερο από 20ms και η συχνότητα των παλμών από 16000Hz έως 20000Hz. Το μετρούμενο σήμα παρουσιάζεται στο κεφάλαιο της αξιολόγησης παρακάτω.

3.6 Οδομετρία

Ο υπολογισμός της οδομετρίας του οχήματος γίνεται στον Η/Υ με χρήση των δεδομένων του μηνύματος ανάδρασης του ελεγκτή. Ως ολικό σύστημα συντεταγμένων (ΣΣ) θεωρείται το σωματόδετο σύστημα του οχήματος την χρονική στιγμή που ξεκινάει την κίνηση του. Οι άξονες φαίνονται στο σχήμα παρακάτω.



Ο μετασχηματισμός από ένα ΣΣ (A) σε ένα άλλο (B) γίνεται μέσω του ομογενούς μετασχηματισμού

$$\begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}_{B}R & & & {}^{A}P_{B_{ORG}} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{{}^{A}T_{B}} \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}$$

Όπου ${}^{A}P$ είναι ένα σημείο P εκφρασμένο ως προς το ΣΣ-A και ${}^{B}P$ το ίδιο σημείο ως προς το ΣΣ-B. ${}^{A}_{B}R$ είναι ο πίνακας στροφής του ΣΣ-B ως προς το ΣΣ-A. Αν θεωρήσουμε ότι η περιστροφή πραγματοποιείται γύρω από τον άξονα z ισούται με

$${}^{A}_{B}R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

και ${}^{A}P_{B_{ORG}}$ η θέση της αρχής του ΣΣ-B ως προς το ΣΣ-A.

Όπως όλα το unicycle οχήματα, το ρομπότ θεωρείται ότι κινείται ως προς τον x άξονα του σωματόδετου ΣΣ. Προκύπτει έτσι ότι

$${}^{A}P_{B_{BORB}} = \begin{bmatrix} \delta x \\ 0 \\ 0 \end{bmatrix}$$

με δx την μετατόπιση του οχήματος που προκύπτει από δύο διαδοχικά μηνύματα ανάδρασης του ελεγκτή. Ο πίνακας στροφής ισούται με

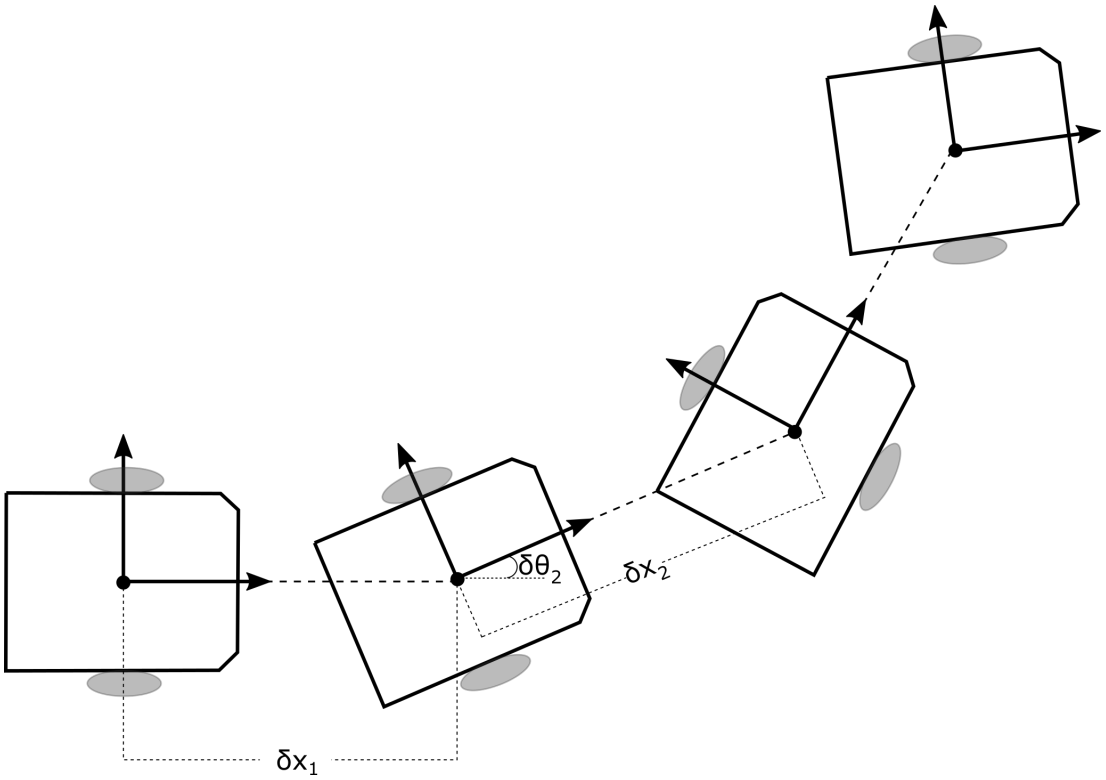
$$ {}^{A}_{B}R = \begin{bmatrix} \cos\delta\theta & -\sin\delta\theta & 0 \\ \sin\delta\theta & \cos\delta\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

με αντίστοιχα $\delta\theta$ την στροφή του οχήματος.

Για τον υπολογισμό της θέσης του κέντρου μάζας ως προς το αρχικό ΣΣ αρκεί ο υπολογισμός όλων των πινάκων T αφού ισχύει

$$ {}^{1}T_{N} = {}^{1}T_{2}\,{}^{2}T_{3}\,{}^{3}T_{4} \cdots {}^{N-2}T_{N+1}\,{}^{N-1}T_{N}. $$

Παρακάτω φαίνεται ένα παράδειγμα τυχαίας κίνησης του ρομπότ.



Έτσι, κάθε φορά που λαμβάνεται νέο μήνυμα ανάδρασης αρχικά υπολογίζονται τα μεγέθη δx και $\delta\theta$ μέσω των σχέσεων

$$ \delta x = \frac{enc_R + enc_L}{2 \cdot enc_{res}} \cdot 2\pi \cdot r $$

$$ \delta\theta = \frac{enc_R - enc_L}{2 \cdot enc_{res} \cdot d} \cdot 2\pi \cdot r $$

όπου enc_R και enc_L οι τιμές του καταμετρητή κάθε encoder. Στην συνέχεια υπολογίζεται ο νέος T χρησιμοποιώντας όμως τα δx και $\delta\theta$ της *προηγούμενης* επανάληψης και πολλαπλασιάζεται με τον παλαιό, που είναι το γινόμενο όλων των προηγουμένων.

Κεφάλαιο 4

Επίλυση: Ανάπτυξη Συστήματος Pioneer 2AT

4.1 Ομοιότητες με Pioneer 3DX

Η λειτουργία του τετράτροχου ρομπότ παρουσιάζει πολλές ομοιότητες με το δίτροχο, αφού η μοναδική ουσιαστική διαφορά είναι η διαφορετική κινηματική τους. Αυτό οφείλεται στο γεγονός ότι για να πραγματοποιήσει σύνθετη κίνηση (περιστροφική και γραμμική ταυτόχρονα) απαιτείται η ολίσθηση των τροχών, που σημαίνει ότι η γραμμική ταχύτητα των τροχών δεν είναι παράλληλη με την κίνηση του οχήματος όπως συμβαίνει στο δίτροχο ρομπότ. Αυτό το θέμα είναι υψηλής πολυπλοκότητας και ο ακριβής προσδιορισμός της σχέσης των ταχυτήτων των τροχών με τις ταχύτητες του κέντρου μάζας επιτυγχάνεται πειραματικά. Έτσι, σε αυτή την διπλωματική χρησιμοποιήθηκε η σχέση των ταχυτήτων που ισχύει και στο δίτροχο όχημα και αναλύθηκε παραπάνω.

4.2 Διαχείριση Encoders

Συγκρίνοντας τα χαρακτηριστικά των encoders που φέρουν τα ρομπότ παρατηρείται ότι το τετράτροχο έχει σημαντικά χαμηλότερη ανάλυση ($100[\frac{clicks}{rev}]$). Πρέπει επομένως να υπολογιστεί η μέγιστη συχνότητα παλμών και να συγκριθεί με αυτή του δίτροχου. Από τα τεχνικά χαρακτηριστικά του ρομπότ η μέγιστη γραμμική ταχύτητα του οχήματος είναι $v_{max} = 0.8[\frac{m}{s}]$ και ισούται με την μέγιστη γραμμική ταχύτητα των τροχών. Εκτός από τον ενσωματωμένο μειωτήρα στροφών, ο ιμάντας μειώνει περαιτέρω τις στροφές αφού οι δύο τροχαλίες που λειτουργούν στον μηχανισμό δεν έχουν ίσο αριθμό οδόντων - ίση διάμετρο. Ο λόγος μείωσης ισούται με το πηλίκο των αριθμών οδόντων τους, με αριθμητή την μεγαλύτερη τροχαλία και ισούται με $n_{pulley} = \frac{25}{20} = 1.25$. Έτσι, η μέγιστη ταχύτητα περιστροφής του κινητήρα

είναι

$$\omega_{motor} = \frac{v_{max}}{r} \cdot n \cdot n_{pulley} = 590.09[\frac{rad}{s}].$$

Η μέγιστη συχνότητα των παλμών προκύπτει

$$f_{max} = \frac{\omega_{motor}}{2\pi} \cdot enc_{res} = \frac{590.09}{2\pi} \cdot 100 = 9391.85[Hz].$$

Είναι φανερό ότι η συχνότητα αυτή είναι εύκολα διαχειρίσιμη από τον ελεγκτή αφού είναι μικρότερη και από την διαιρεμένη συχνότητα του δίτροχου ρομπότ επομένως δεν απαιτείται η χρήση αντίστοιχου κυκλώματος.

4.3 Σύνοψη

Επομένως, η λύση που αναπτύχθηκε σε επίπεδο software είναι όμοια με αυτήν του δίτροχου ρομπότ, με μόνες διαφορές τις τεχνικές και κατασκευαστικές παραμέτρους, όπως την ανάλυση των encoders, την ακτίνα των τροχών κ.α. Δεν αναπτύχθηκε κόμβος υπολογισμού της οδομετρίας του οχήματος για τους λόγους που αναφέρθηκαν παραπάνω.

Κεφάλαιο 5

Αποτελέσματα-Αξιολόγηση-Μελλοντικές Δράσεις

5.1 Αποτελέσματα

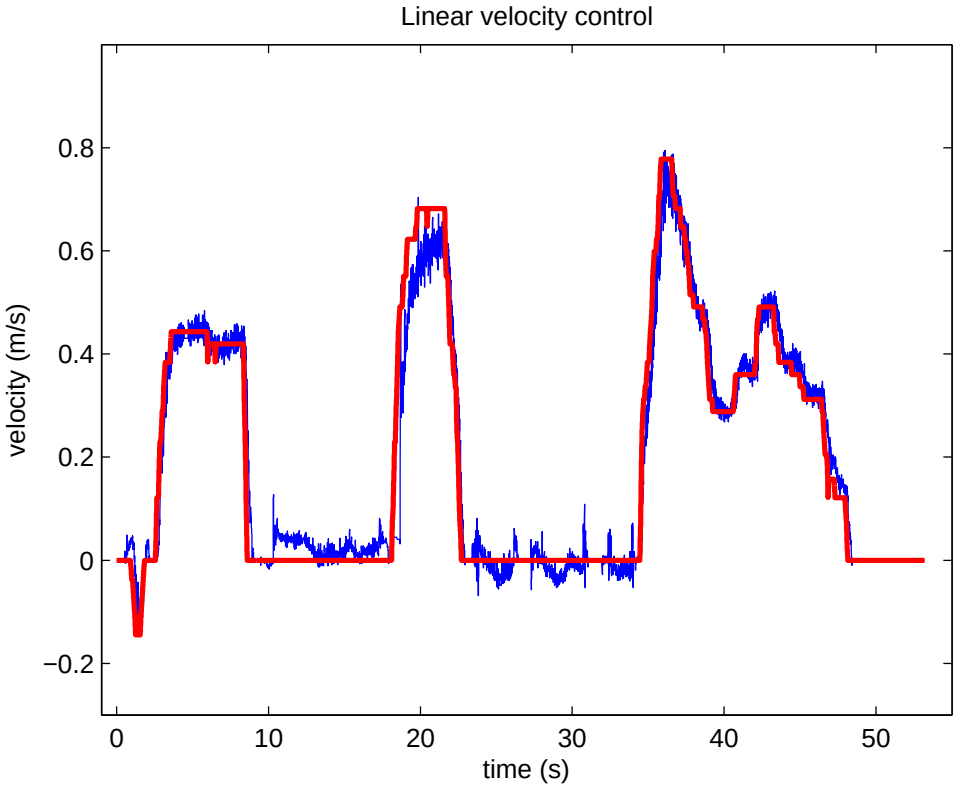
Μετά το πέρας των εργασιών έχουν ανακατασκευαστεί δύο ρομπότ. Τα συστήματα αυτά λειτουργούν κάτω από το λειτουργικό σύστημα ROS και μπορούν να επικοινωνούν με τον κεντρικό υπολογιστή που δίνει τις εντολές κίνησης μέσω Wifi. Η αποστολή εντολών στα ρομπότ και η λήψη πληροφοριών ανάρδασης από αυτά στον κεντρικό υπολογιστή είναι απλή διαδικασία, χάρη στο σύστημα ROS. Η χρήση ROS συνεπάγεται εύκολη ανάπτυξη λογισμικού που θα διαχειρίζεται τις παραπάνω πληροφορίες και θα προσθέτει δυνατότητες στα συστήματα (κάμερα, αισθητήρες κ.α.). Παράλληλα, ο -ανοιχτού κώδικα- ελεγκτής Arduino επιτρέπει την παρέμβαση στον τρόπο λειτουργίας του έτσι ώστε να μπορούν να μεταβληθούν οι πληροφορίες που στέλνονται από το ρομπότ στον κεντρικό υπολογιστή.

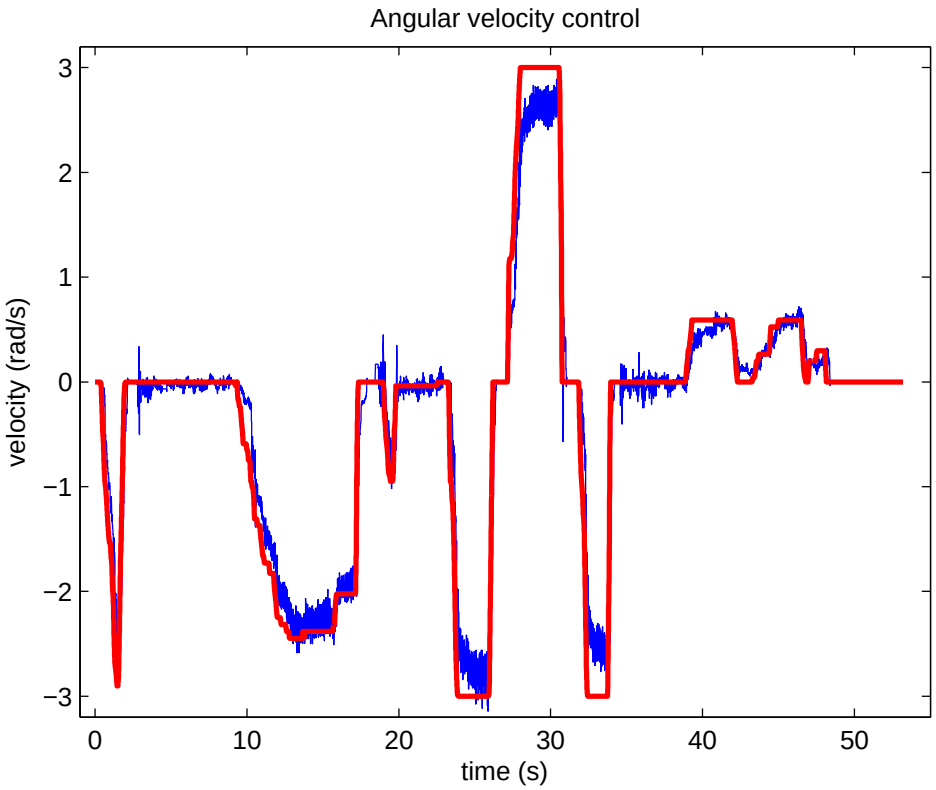
5.2 Αξιολόγηση

Αρχικά πραγματοποιήθηκε πείραμα για την εξακρίβωση της σωστής μέτρησης των encoders από τον μικρο-ελεγκτή και της σωστής λειτουργίας του κυκλώματος υποδιαίρεσης της συχνότητας των παλμών. Με ψηφιακό παλμογράφο μετράται η συχνότητα των παραγόμενων παλμών και μέσω αυτής με την γνωστή σχέση (βλ. 3.1) προκύπτει η ταχύτητα περιστροφής του τροχού. Παρατηρήθηκε ότι το κύκλωμα υποδιαίρεσης της συχνότητας των παλμών συμβάλει θετικά και στην ποιότητα του παραγόμενου σήματος αφού το πρωτότυπο σήμα δεν είναι ακριβώς τετραγωνικό και δεν κυμαίνεται εντός του ορίου των 0 και 5V, αλλά παίρνει και μικρές αρνητικές τιμές.

Το επεξεργασμένο σήμα όμως έχει τετραγωνική μορφή και βρίσκεται εντός του παραπάνω ορίου.

Ο ορθός έλεγχος της ταχύτητας αξιολογείται με τυχαία κίνηση του ρομπότ δεχόμενο εντολές από τηλεχειριστήριο και φαίνεται στα παρακάτω διαγράμματα, όπου με κόκκινη γραμμή είναι η επιθυμητή ταχύτητα.

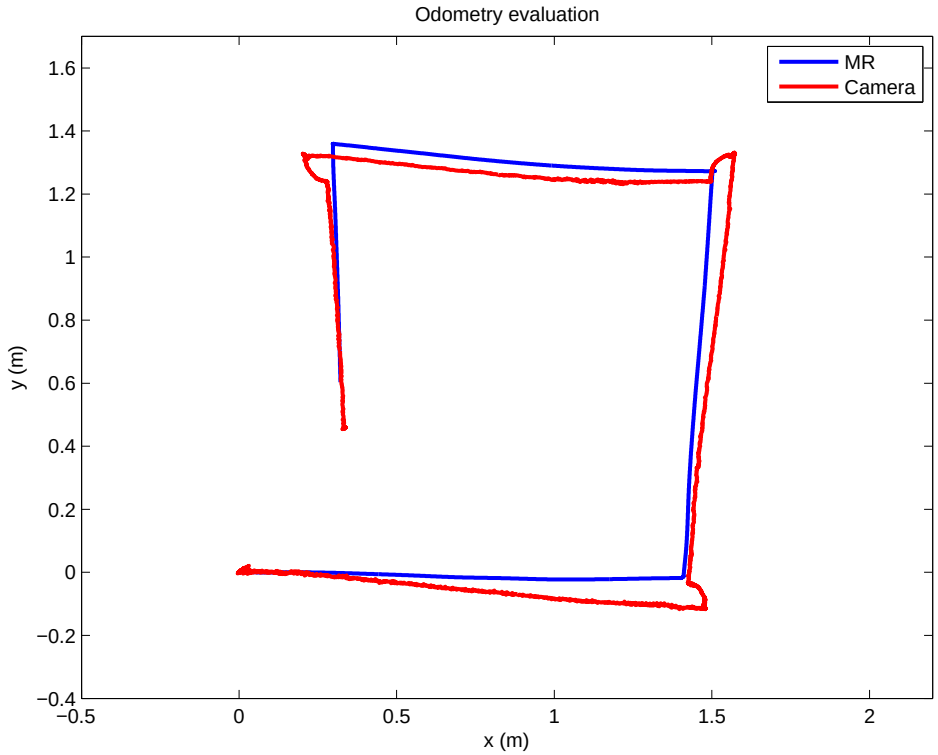


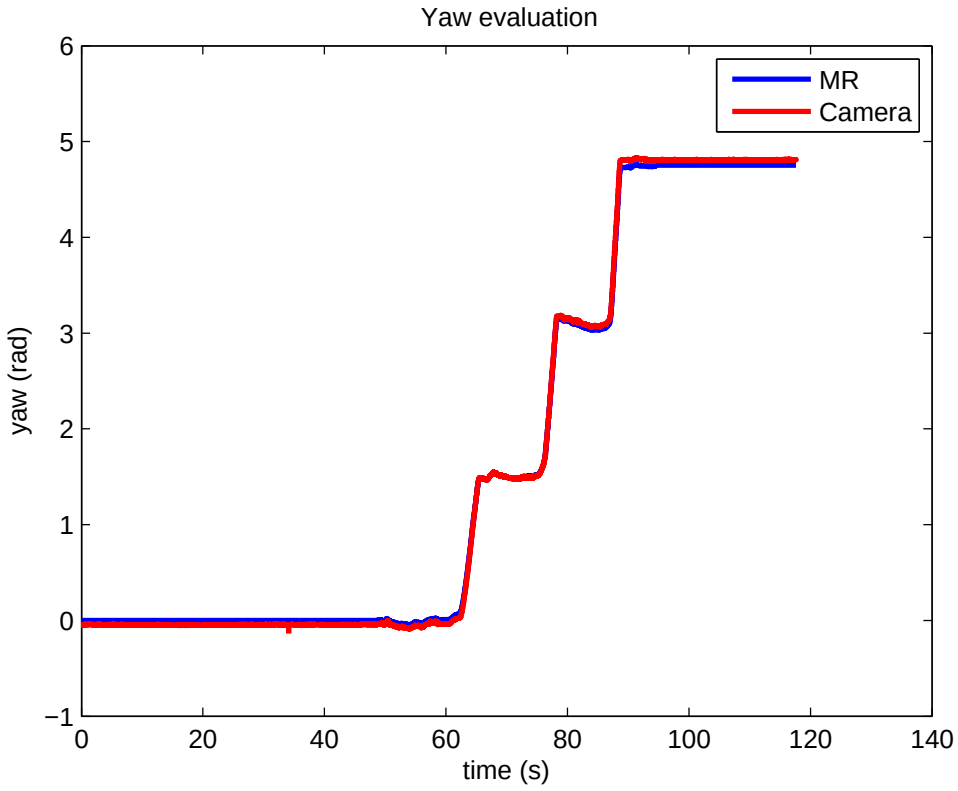


Η μορφή του σήματος παρουσιάζει ακμές λόγω του μικρού διαστήματος του κύκλου υπολογισμών (15ms). Ορίστηκε αυτή η τάξη μεγέθους καθώς η υψηλή συχνότητα του PWM για την οδήγηση των κινητήρων και οι πυκνωτές που τοποθετήθηκαν παράλληλα στους κινητήρες δυσκολεύουν τον έλεγχο όταν το χρονικό διάστημα αυξάνεται. Έτσι, προτιμήθηκε αυτή η απώλεια αντί του αυξημένου θορύβου (ηχητικού) των κινητήρων ή του ευαίσθητου ελέγχου. Το συγκεκριμένο looptime επιτρέπει την επιλογή κερδών από ένα ευρύ πεδίο δυνατών τιμών.

Σε κάποια χρονικά διαστήματα παρατηρούνται διακυμάνσεις της γραμμικής ταχύτητας γύρω από το μηδέν. Αυτό οφείλεται στην περιστροφική κίνηση του οχήματος που προκαλεί και μικρή μεταφορική ταχύτητα. Αντίστοιχη συμπεριφορά παρατηρείται και στο σήμα της γωνιακής ταχύτητας.

Στην συνέχεια για την αξιολόγηση του τελικού αποτελέσματος πραγματοποιήθηκε πείραμα κατά το οποίο το ρομπότ εκτελούσε τυχαία και όσο το δυνατό λιγότερο ολισθηρή κίνηση, ενώ ταυτόχρονα κάμερα κατέγραφε την θέση του στον χώρο. Η επεξεργασία των δεδομένων της κάμερας και ο προσδιορισμός των χαρακτηριστικών της θέσης του οχήματος στον χώρο (x,y,z,θ) έγινε μέσω αλγορίθμου του εργαστηρίου στο σύστημα ROS. Παρακάτω παρατίθενται οι γραφικές παραστάσεις της πορείας του οχήματος στον χώρο (x,y) και η γωνία (yaw) γύρω από τον άξονα z, όπως αυτά μετρήθηκαν από το ρομπότ (MR) και από την κάμερα.



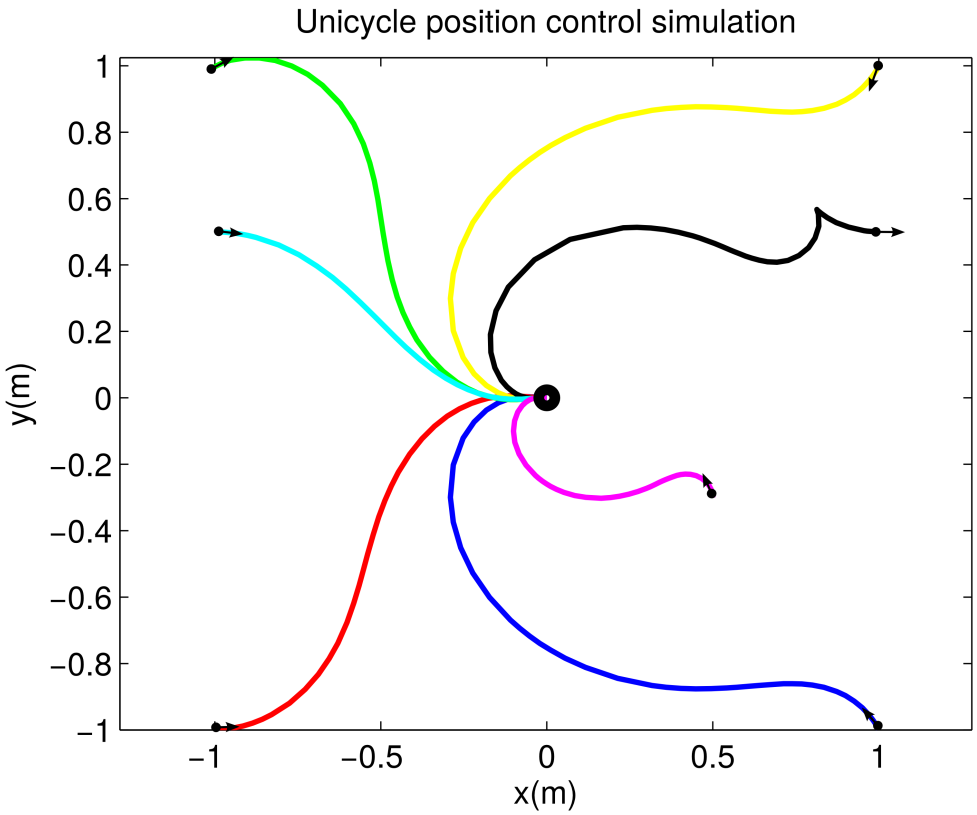


Τα δεδομένα αυτά συγκρίνονται με τα δεδομένα της οδομετρίας του ρομπότ και τα αποτελέσματα του πειράματος προκύπτουν πολύ ικανοποιητικά λαμ-

βάνοντας υπόψη τα σφάλματα της κάμερας και την ολίσθηση των τροχών. Από τα δεδομένα της κάμερας φαίνεται ότι η περιστροφή του ρομπότ δεν γίνεται γύρω από το κέντρο περιστροφής γιατί ο marker που χρησιμοποιείται για την όραση έχει τοποθετηθεί ελάχιστα έκκεντρα.

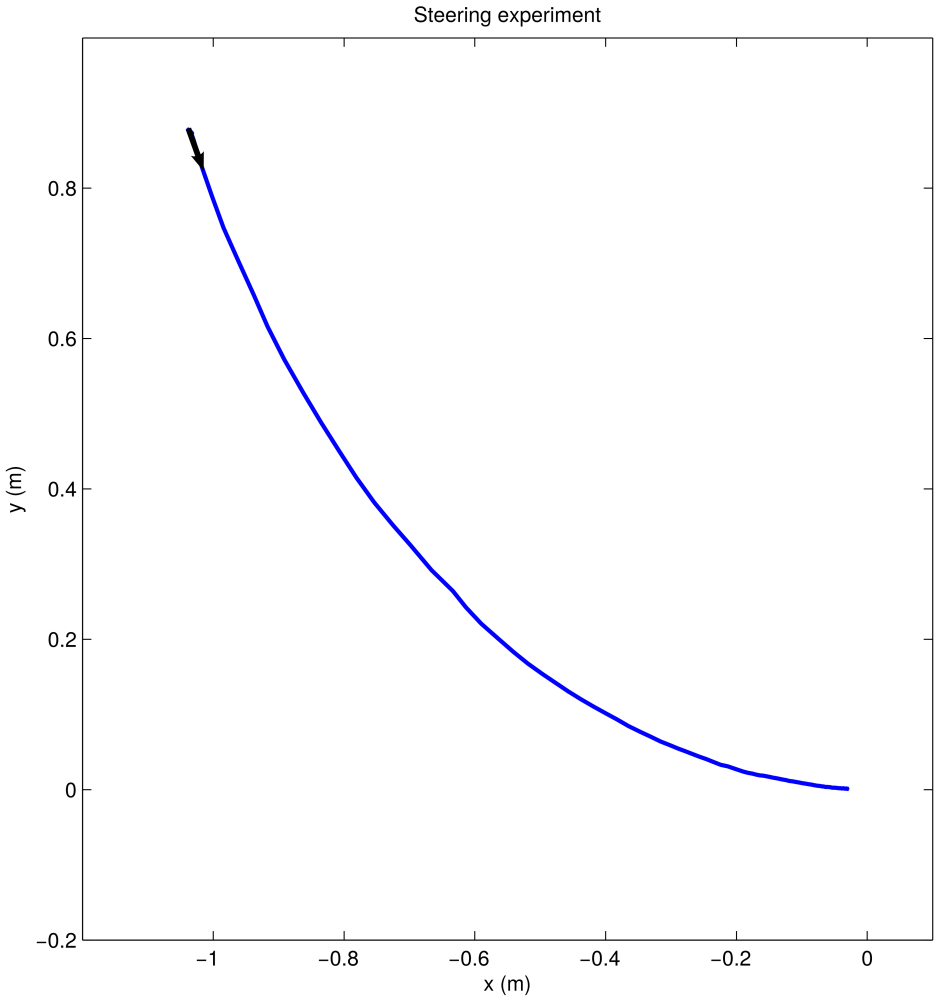
Επίσης, η απόκλιση των δύο τροχιών πιθανόν οφείλεται και στο γεγονός ότι οι άξονες των δύο ΣΣ δεν ήταν απολύτως ευθυγραμμισμένοι κατά την εκκίνηση του πειράματος.

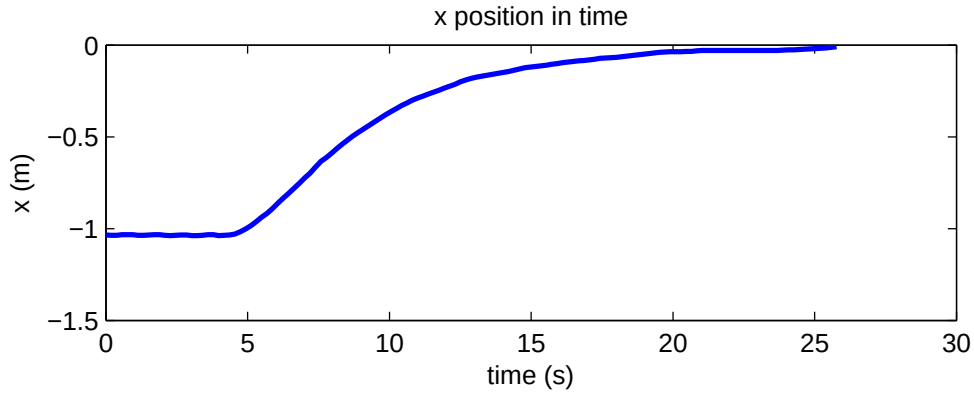
Τέλος, υλοποιήθηκε αλγόριθμος ελέγχου της θέσης του ρομπότ που περιγράφεται στην ανάλυση[1] των M.Aicardio, G.Casalino, A.Balestrino, A.Bicchi. Μέσω κάμερας υπολογίζεται η θέση και ο προσανατολισμός του οχήματος στον χώρο και από αυτά προκύπτουν η κατάλληλη γραμμική και περιστροφική ταχύτητα του κέντρου μάζας του οχήματος έτσι ώστε να φτάσει στον στόχο προσανατολισμένο παράλληλα στο σύστημα συντεταγμένων της κάμερας. Αρχικά, έγινε προσομοίωση για την εύρεση της σωστής τάξης μεγέθους των τριών κερδών του ελεγκτή. Έτσι, για διάφορες αρχικές συνθήκες (θέση και προσανατολισμό) και με τελικό στόχο την αρχή των αξόνων προέκυψαν κατάλληλα κέρδη που παρουσιάζουν την παρακάτω συμπεριφορά.

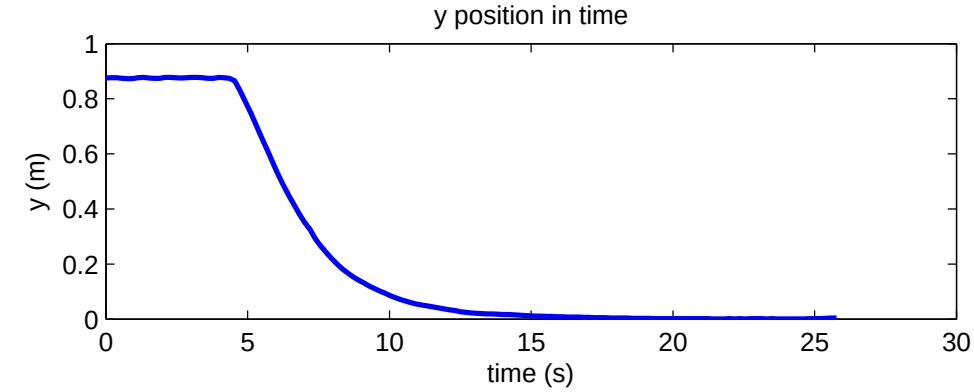


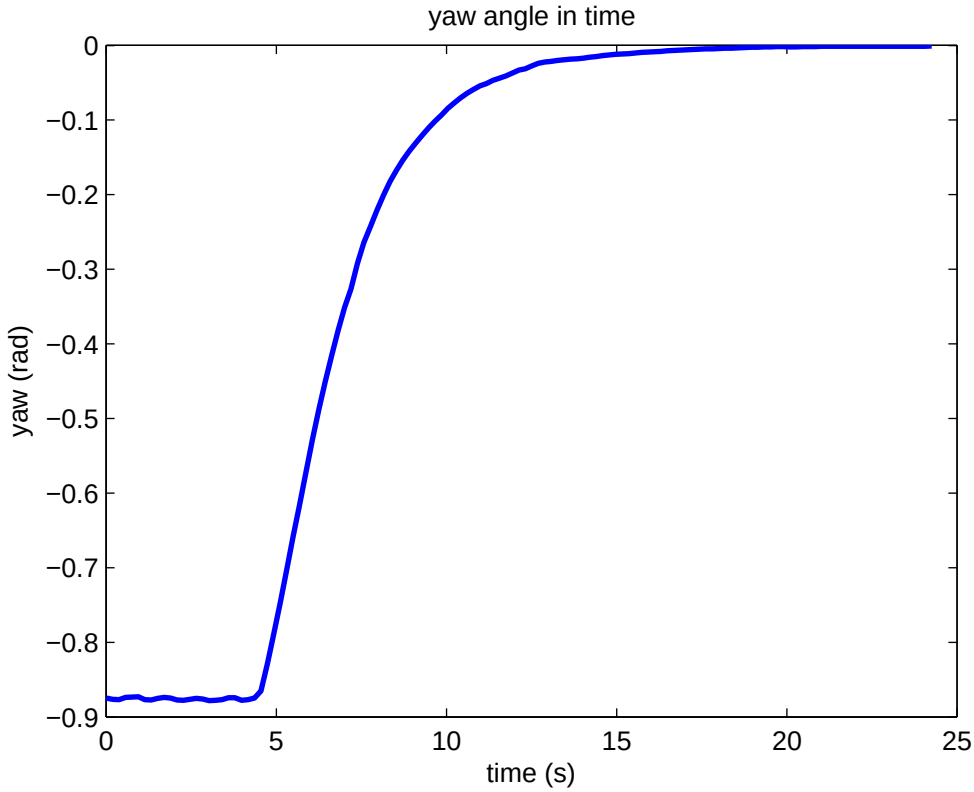
Στην συνέχεια παρατίθενται τα δεδομένα της κάμερας για την οδομετρία του οχήματος καθώς υλοποιείται ο αλγόριθμος.

[1]Closed loop smooth steering of unicyle-like vehicles


Steering experiment
0.8
0.6
0.4
0.2
0
−0.2
y (m)
−1
−0.8
−0.6
−0.4
−0.2
0
x (m)







Παρατηρείται ότι το ρομπότ φτάνει πολύ κοντά στον στόχο (x, y, θ). Οι μικρές αποκλίσεις οφείλονται στις μικρές απαιτούμενες ταχύτητες (της τάξης

$0.05\frac{m}{s}$) που είναι αδύνατο να κινήσουν το ρομπότ.

5.3 Μελλοντικές Δράσεις

Όπως ειπώθηκε δεν έχει ληφθεί υπόψη η κινηματική του τετράτροχου ρομπότ στην ανάπτυξη των αλγορίθμων. Αυτό σημαίνει ότι για την σύνδεση μεταξύ των ταχυτήτων των τροχών με τις ταχύτητες του κέντρου μάζας, ελήφθησαν υπόψη οι σχέσεις του δίτροχου ρομπότ. Οι σχέσεις αυτές όμως ισχύουν όταν το ρομπότ εκτελεί ευθύγραμμη κίνηση και παρουσιάζουν σφάλμα όταν εκτελεί και περιστροφική ταυτόχρονα. Ο υπολογισμός των σχέσεων της κινηματικής του απαιτεί σειρά πειραμάτων για τον προσδιορισμό της επίδρασης της ολίσθησης στις ταχύτητες.

Έτσι, μελλοντικά μπορούν να πραγματοποιηθούν τα πειράματα αυτά και να αξιοποιηθούν σε δύο σημεία. Πρώτον στις εντολές κίνησης του ρομπότ, έτσι ώστε από τις επιθυμητές ταχύτητες του κέντρου μάζας να υπολογίζονται οι ταχύτητες των τροχών. Δεύτερον στην ανάπτυξη κόμβου υπολογισμού της οδομετρίας έτσι ώστε να προσδιορίζεται η θέση του οχήματος σε σχέση με αυτήν που είχε στην αρχή της κίνησής του.

Επιπροσθέτως, και τα δύο συστήματα μπορούν να ενισχυθούν λειτουργικά ενσωματώνοντας κάμερα για όραση και ενός ρομποτικού βραχίονα για απλές εργασίες στον χώρο. Οι προσθήκες αυτές είναι ανεξάρτητες από το αναπτυχθέν σύνολο κόμβων και μπορούν να τρέχουν ανεξάρτητα στο ROS.

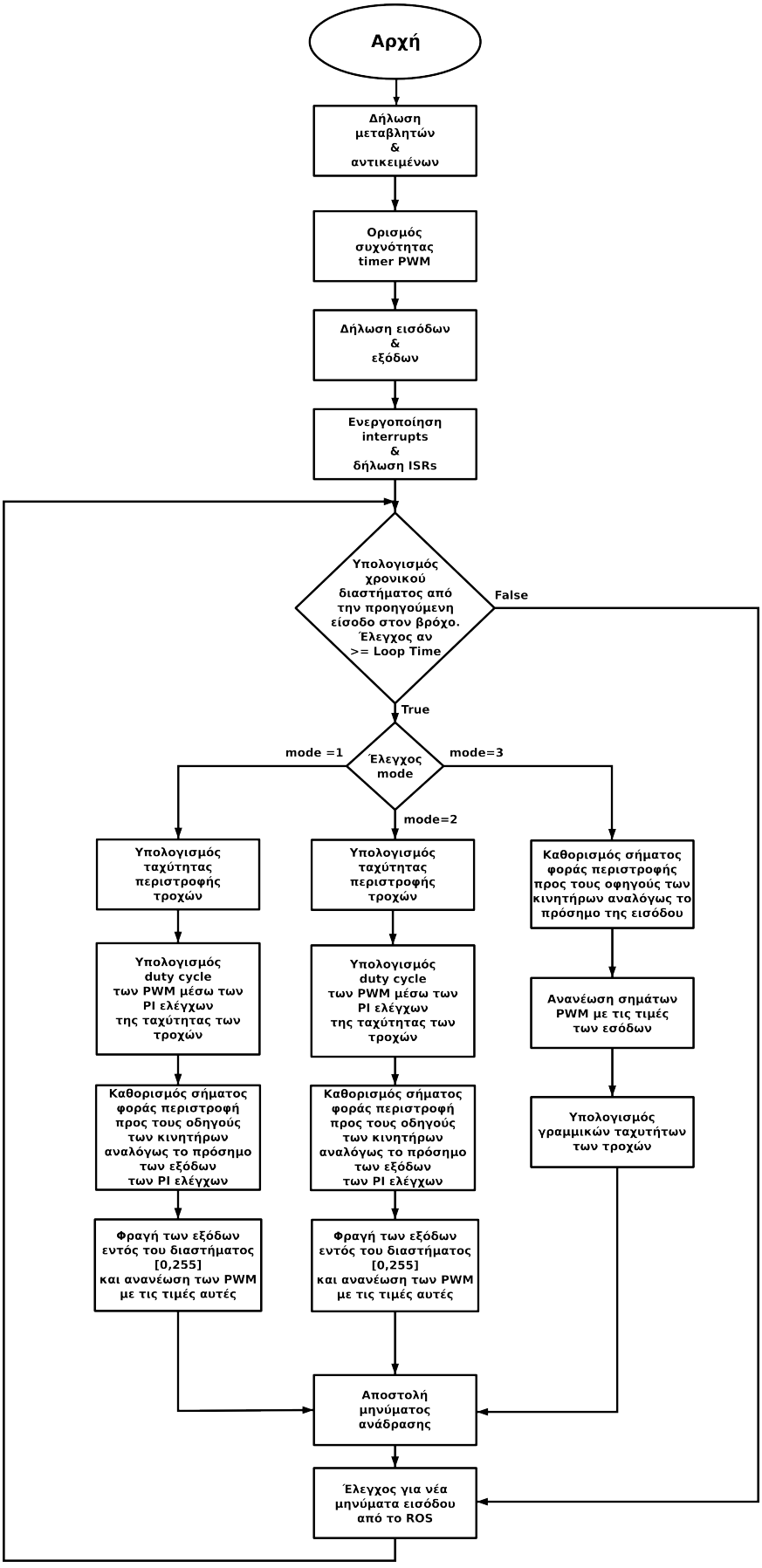
Παράρτημα Α΄

Περιγραφή Software

Α΄.1 Arduino

Α΄.1.1 Δομικό Διάγραμμα Arduino

Παρατίθεται το δομικό διάγραμμα του αλγορίθμου του μικρο-ελεγκτή. Η λειτουργία των interrupts δεν μπορεί να παρασταθεί στο δομικό διάγραμμα. Όπως αναφέρθηκε στην παραπάνω ανάλυση η εκτέλεση του παρακάτω αλγορίθμου διακόπτεται όταν ενεργοποιείται μία εκ των δύο συναρτήσεων interrupt (ISR) και μετά το πέρας των υπολογισμών της συνεχίζει από το προηγούμενο σημείο.


Αρχή
Δήλωση μεταβλητών & αντικειμένων
Ορισμός συχνότητας timer PWM
Δήλωση εισόδων & εξόδων
Ενεργοποίηση interrupts & δήλωση ISRs
Υπολογισμός χρονικού διαστήματος από την προηγούμενη είσοδο στον βρόχο. Έλεγχος αν >= Loop Time
False
True
Έλεγχος mode
mode =1
mode=2
mode=3
Υπολογισμός ταχύτητας περιστροφής τροχών
Υπολογισμός duty cycle των PWM μέσω των PI ελέγχων της ταχύτητας των τροχών
Καθορισμός σήματος φοράς περιστροφή προς τους οδηγούς των κινητήρων αναλόγως το πρόσημο των εξόδων των PI ελέγχων
Φραγή των εξόδων εντός του διαστήματος [0,255] και ανανέωση των PWM με τις τιμές αυτές
Υπολογισμός ταχύτητας περιστροφής τροχών
Υπολογισμός duty cycle των PWM μέσω των PI ελέγχων της ταχύτητας των τροχών
Καθορισμός σήματος φοράς περιστροφή προς τους οδηγούς των κινητήρων αναλόγως το πρόσημο των εξόδων των PI ελέγχων
Φραγή των εξόδων εντός του διαστήματος [0,255] και ανανέωση των PWM με τις τιμές αυτές
Καθορισμός σήματος φοράς περιστροφής προς τους οφηγούς των κινητήρων αναλόγως το πρόσημο της εισόδου
Ανανέωση σημάτων PWM με τις τιμές των εσόδων
Υπολογισμός γραμμικών ταχυτήτων των τροχών
Αποστολή μηνύματος ανάδρασης
Έλεγχος για νέα μηνύματα εισόδου από το ROS

Α΄.1.2 Ανάλυση Συναρτήσεων

Κατά την έναρξη του κώδικα του Arduino εκτελούνται μία φορά οι συναρτήσεις ορισμού της συχνότητας των παλμών PWM.

```
InitTimersSafe();
SetPinFrequencySafe(pin, freq);
```

Η πρώτη αρχικοποιεί όλους τους timers πλην του timer0 που χρησιμοποιείται στις συναρτήσεις ελέγχου του χρόνου. Η άλλη συνάρτηση δέχεται ως ορίσματα το pin που θα παράγει τον παλμό και την επιθυμητή συχνότητα σε Hz.

Αντικείμενο PID_motor

Για κάθε κινητήρα ορίζεται ένα αντικείμενο το οποίο εμπεριέχει τις συναρτήσεις υπολογισμού της ταχύτητας περιστροφής (get_speed) του κινητήρα, τον υπολογισμό του PID (compute_PID) και την συνάρτηση μεταβολής των κερδών του ελέγχου.

Ο ορισμός των αντικειμένων απαιτεί ως είσοδο την περίοδο δειγματοληψίας (loop time), τα κέρδη του ελεγκτή (Kp, Ki και Kd) και την ανάλυση του encoder ανά περιστροφή. Η ανάλυση αυτή αφορά την ανάλυση ανά περιστροφή του τροχού.

```
motor_PID(unsigned int Looptime, int Kp, int Ki, int Kd, float
    enc_res);
```

PID_motor::get_speed( ) Η συνάρτηση δέχεται την τιμή της μεταβλητής του encoder και επιστρέφει την ταχύτητα περιστροφής με μονάδες $[\frac{rev}{s}]$. Η τιμή του encoder ουσιαστικά αποτελεί την μεταβολή της μεταβλητής από την προηγούμενη είσοδο στον βρόχο ελέγχου.

```
float motor_PID::get_speed(int encoder_counter);
```

PID_motor::compute_PID( ) Δέχεται την τιμή της μεταβλητής ελέγχου, στην προκειμένη περίπτωση την ταχύτητα, είτε γραμμική είτε περιστροφική και την αντίστοιχη τιμή-στόχο προς επίτευξη. Επιστρέφει το αποτέλεσμα του ελέγχου PID όπως αναλύθηκε στο παραπάνω κεφάλαιο.

```
float motor_PID::compute_PID(float current_speed, float
    speed_setpoint);
```

Α΄.1.3 Publishing & Subscribing

Στην ενότητα αυτή αναλύονται οι ενέργειες που απαιτούνται για την επικοινωνία με το σύστημα ROS και την ανταλλαγή μηνυμάτων.

Αρχικά είναι απαραίτητη η δήλωση του κόμβου. Για να γίνει αυτό πρέπει να οριστεί το αντικείμενο χειρισμού του κόμβου

```
ros::NodeHandle nh;
```

Ακολουθεί η δήλωση του κόμβου στην συνάρτηση αρχικοποίησης του κώδικα του Arduino (setup()).

```
nh.initNode();
```

Publishing. Αρχικά ορίζεται το αντικείμενο που αντιστοιχεί στο μήνυμα του ROS και παράγεται από το αντίστοιχο πακέτο του ROS.

```
arduino_mr::feedback_int feed;
```

Ακολουθεί η δήλωση του αντικειμένου που χειρίζεται το Publishing

```
ros::Publisher to_ros_feedback("feedback", &feed);
```

Που δέχεται ως ορίσματα το όνομα του topic και τον δείκτη (pointer) του μηνύματος που δημοσιεύεται.

Στην συνέχεια δηλώνεται το αντικείμενο στον κόμβο ως Publisher εντός της setup()

```
nh.advertise(to_ros_feedback);
```

το οποίο το αντικείμενο που χειρίζεται τις ενέργειες του Publishing.

Τέλος, η δημοσίευση ενός μηνύματος σε ένα topic γίνεται μέσω της συνάρτησης (εντός της loop())

```
to_ros_feedback.publish(&feed);
```

Έτσι, εκτελώντας την συνάρτηση

```
publish_feedback();
```

Ανανεώνονται οι τιμές του μηνύματος (feed) και καλείται η συνάρτηση δημοσίευσης του μηνύματος (to_ros_feedback.publish()) που αναφέρθηκε παραπάνω.

Subscribing. Σε αυτή την περίπτωση αρχικά ορίζονται οι callback συναρτήσεις που εκτελούνται όταν υπάρχει νέο μήνυμα. Οι συναρτήσεις αυτές δεν επιστρέφουν κάποια τιμή και ως όρισμα απαιτούν τον δείκτη του αντικειμένου του μηνύματος. Στον κώδικα υπάρχουν δύο Subscribers και για απλότητα χρησιμοποιείται ως παράδειγμα αυτός που χειρίζεται το μήνυμα των εντολών κίνησης.

```
void set_arduino_input(const arduino_mr::arduino_input& input);
```

Το όρισμα της συνάρτησης είναι ο δείκτης του μηνύματος εισόδου. Ταυτόχρονα δηλώνεται το αντικείμενο του μηνύματος κατ'αντιστοιχία με το μήνυμα του Publisher, με την διαφορά ότι η δήλωση γίνεται εντός του ορίσματος της συνάρτησης.

Ακολουθεί η δήλωση του αντικειμένου που χειρίζεται το Subscribing στο topic

```
ros::Subscriber<arduino_mr::arduino_input> sub("arduino_input", &
    set_arduino_input,1);
```

Το αντικείμενο απαιτεί σαν είσοδο τον τύπο του μηνύματος εντός των < >. Επίσης, το όνομα του topic όπου θα γίνει η εγγραφή, τον δείκτη στην callback συνάρτηση και τέλος το μήκος της ουράς που αποθηκεύονται τα ανεπεξέργαστα μηνύματα.

Επόμενο βήμα είναι η δήλωση του αντικειμένου στον κόμβο ως Subscriber (στην setup())

```
nh.subscribe(sub);
```

Όπου δέχεται το αντικείμενο που χειρίζεται το Subscribing.

Τέλος, ο έλεγχος για νέα μηνύματα γίνεται μέσω της συνάρτησης (στην loop())

```
nh.SpinOnce();
```

Α΄.1.4 Headers για την χρήση των συναρτήσεων

Για την χρήση των αντικειμένων επικοινωνίας με το σύστημα ROS απαιτείται το header

```
#include ros.h
```

Για την αλλαγή της συχνότητας των PWM

```
#include PWM.h
```

Για τα αντικείμενο ελέγχου των κινητήρων

```
#include PID_motor.h
```

Για τα μηνύματα του ROS χρησιμοποιείται και ο αντίστοιχος header. Για απλότητα αναφέρεται μόνο για το ένα μήνυμα.

```
#include <arduino_mr/feedback_int.h>
```

Όπου πρώτο σε σειρά είναι το πακέτο που ανήκει το μήνυμα και στην συνέχεια ακολουθεί αυτό.

Α΄.2 ROS

Α΄.2.1 ROS messages

feedback

Το μήνυμα arduino_mr/feedback_int περιλαμβάνει της πληροφορίες ανάδρασης του Arduino. Αποτελείται από

int8	mode
int16	encoderR
int16	encoderL
int16	pwm_to_motorR
int16	pwm_to_motorL
uint8	looptime
int16	analogData0
int16	analogData1

Mode είναι το mode που λειτουργει το ρομπότ, encoder είναι η τιμή κάθε encoder ανά κύκλο λειτουργίας (loop time), pwm_to_motor είναι το duty cycle που στέλνεται στον οδηγό των κινητήρων και analogData η τιμή της αναλογικής εισόδου του Arduino.

arduino_input

Το μήνυμα arduino_mr/arduino_input περιέχει την πληροφορία της εντολής κίνησης. Έχει την παρακάτω δομή

int8	mode
float32	data1
float32	data2

Για το mode ελέγχου των ταχυτήτων του κέντρου μάζας (mode=1) η αντιστοίχηση των μεταβλητών data είναι

data1	γραμμική ταχύτητα
data2	περιστροφική ταχύτητα

Για το mode ελέγχου της περιστροφικής ταχύτητας των τροχών (mode=2) η αντιστοίχηση είναι

data1	ταχύτητα δεξιού τροχού
data2	ταχύτητα αριστερού τροχού

Τέλος, για το mode απευθείας αποστολής duty cycle στους κινητήρες (mode=3) είναι

data1	PWM δεξιού κινητήρα
data2	PWM αριστερού κινητήρα

Μονάδες. Οι μονάδες των ταχυτήτων είναι στο SI και το εύρος των τιμών για το PWM είναι $[-255, 255]$ με τα θετικά να ισοδυναμούν με πρόσθια κίνηση.

PID_motors_gains

Το μήνυμα έχει την παρακάτω δομή

arduino_mr/PID_gains[2]	gains

Και το μήνυμα PID_gains αποτελείται από

uint16	Kp
uint16	Ki
uint16	Kd

Εμπεριέχει δηλαδή την πληροφορία για τρία κέρδη ελέγχου για κάθε κινητήρα.

Παράρτημα Β΄

Περιγραφή Hardware

Β΄.1 Pioneer 3DX

Β΄.1.1 Χρησιμοποιηθέντα εξαρτήματα

Στο δίτροχο ρομπότ χρησιμοποιήθηκαν τα παρακάτω εξαρτήματα

1x	Arduino UNO R3
1x	DC Motor Driver
2x	1μF MKT πυκνωτής τάσης 25V
1x	prototyping board
1x	4520 IC
1x	4013 IC

Β΄.1.2 Πλακέτα

Αρχικά αναλύεται το κύκλωμα που αναπτύχθηκε στην επίλυση του Pioneer 3DX για την διαίρεση της συχνότητας των παλμών των encoders. Χρησιμοποιήθηκε ειδική πλακέτα prototyping η οποία ταιριάζει στο Arduino UNO αφού έχει διαφορετική απόσταση μεταξύ των pins σε σχέση με την τυποποιημένη τιμή.

Επειδή για την ευθύγραμμη κίνηση του οχήματος οι κινητήρες πρέπει να περιστρέφονται αντίρροπα, για κάθε encoder θεωρείται διαφορετικό ως κύριο σήμα μετρούμενων παλμών (signal A). Αναλυτικότερα για τον encoder R είναι

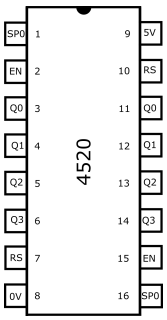
κόκκινο	τάση
μαύρο	γείωση
κίτρινο	signal A
μπλε	signal B

και για τον encoder L

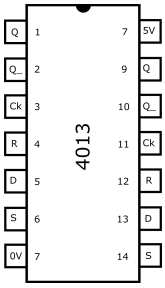
κόκκινο	τάση
μαύρο	γείωση
μπλε	signal A
κίτρινο	signal B

Στην συνέχεια στην πλακέτα κολλήθηκαν κλέμες για την εύκολη σύνδεση των καλωδίων των encoders. Η σειρά σύνδεσης των καλωδίων από αριστερά προς δεξιά *για κάθε encoder* είναι κόκκινο, μαύρο, κίτρινο και μπλε καλώδιο.

Για binary counter χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα HEF4520 που φαίνεται στο σχήμα.



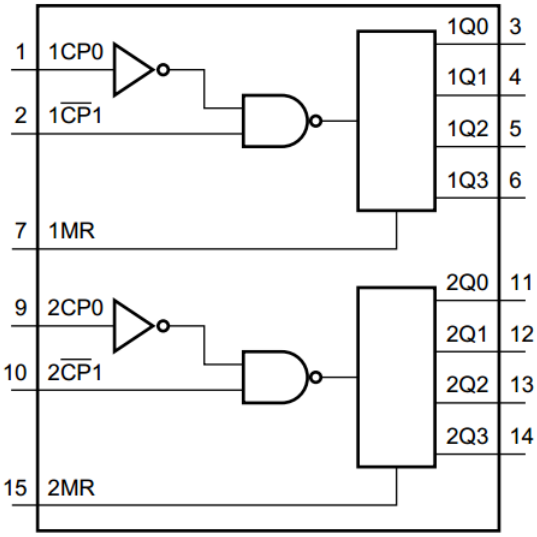
Για d-flipflop χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα 4013 που φαίνεται στο σχήμα.



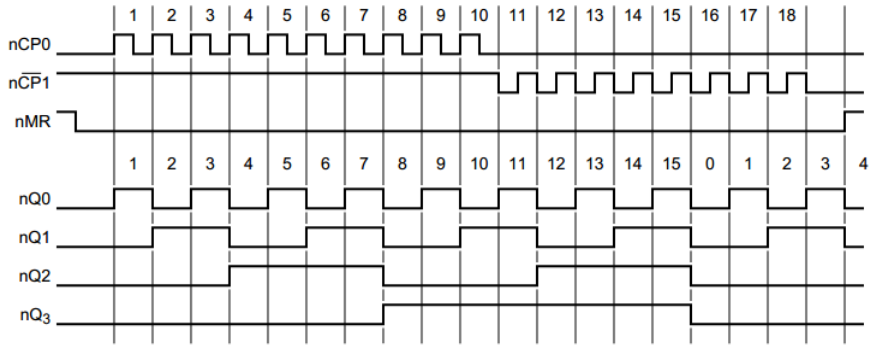
Κάθε ένα από τα παραπάνω ολοκληρωμένα κυκλώματα περιλαμβάνει δύο ανεξάρτητα κυκλώματα. Έτσι, για τον χειρισμό των δύο encoders απαιτείται ένα από κάθε ολοκληρωμένο κύκλωμα.

4520

Το κύκλωμα αυτό έχει τρία pins εισόδου, το σήμα που καταμετράται (Clock-CP0), το pin ενεργοποίησης της καταμέτρησης (Enable-CP1) και το pin μηδενισμού της μεταβλητής καταμέτρησης (Reset-MR). Όταν το Enable είναι HIGH το κύκλωμα καταμετρά τις μεταβάσεις του Clock από 0 σε 1. Το Reset μηδενίζει τις εξόδους του κυκλώματος όταν αυτό παίρνει την τιμή HIGH.



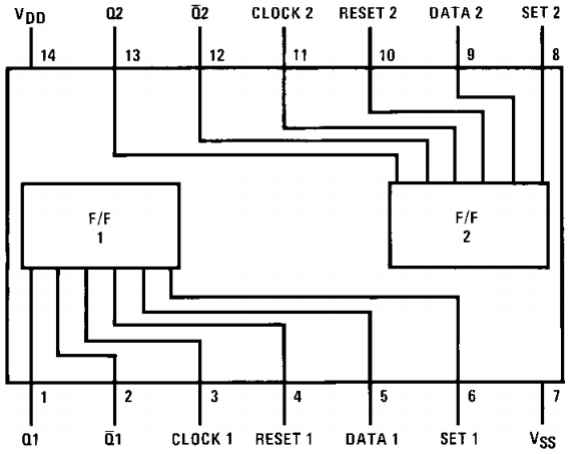
Τέλος, έχει τέσσερις εξόδους τις Q0, Q1, Q2 και Q3. Αυτές αντιστοιχούν στα bits του μετρητή της εισόδου Clock, με το Q0 να είναι το κατώτερο bit (least significand bit - lsb). Η λειτουργία του κυκλώματος κατανοείται καλύτερα μέσω του διαγράμματος χρονισμού.



Όπως αναλύθηκε στην ανάπτυξη της επίλυσης το κύκλωμα χρησιμοποιείται για την υποδιαίρεση της συχνότητας. Έτσι, η έξοδος που χρησιμοποιείται είναι η Q0 που παράγει συχνότητα ίση με την μισή του σήματος εισόδου. Τέλος, παρατηρείται ότι για την επιθυμητή λειτουργία το Reset-MR πρέπει να είναι LOW και το Enable-CP1 HIGH.

4013

Έχει τέσσερα pins εισόδου, το σήμα που αποθηκεύεται (Data), το ρολόι χρονισμού (Clock), το Reset και το Set. Οι έξοδοι του κυκλώματος είναι δύο, οι Q και $\overline{Q}$. Παρακάτω φαίνεται το διάγραμμα του κυκλώματος.



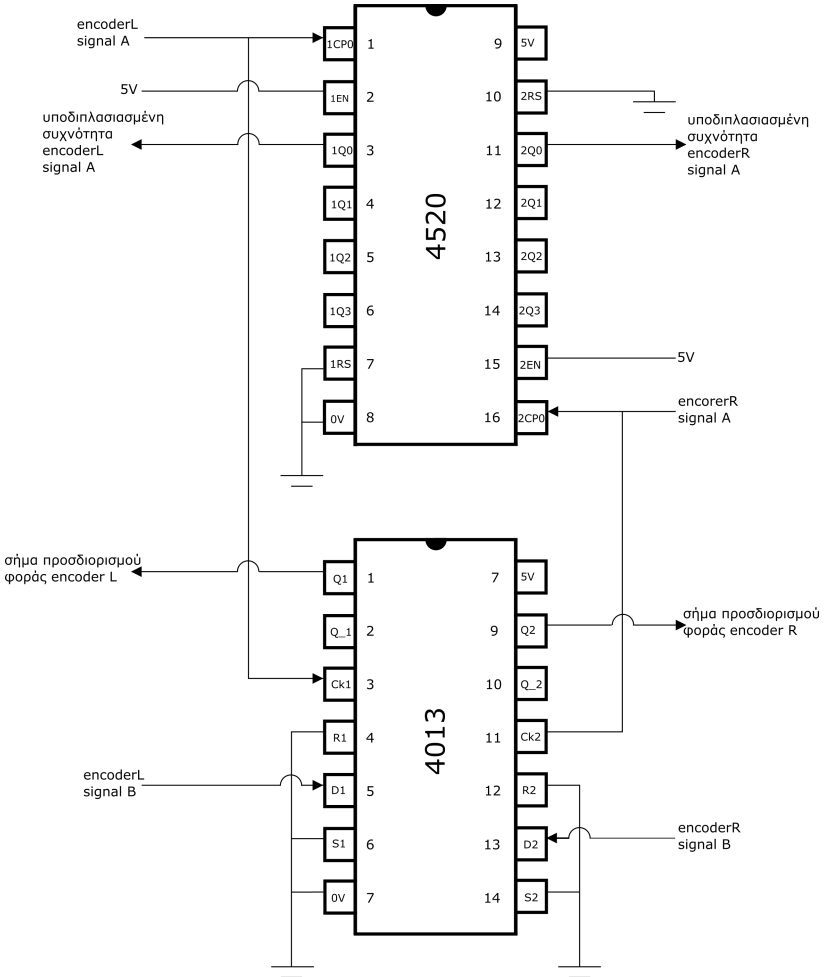
Το κύκλωμα αποθηκεύει την τιμή του Data στην έξοδο Q κάθε φορά που το Clock μεταβαίνει από 0 σε 1. Η λειτουργία του κυκλώματος γίνεται καλύτερα κατανοητή μέσω του πίνακα αληθείας.

Ck	D	R	S	Q	$\overline{Q}$
$0 \rightarrow 1$	0	0	0	0	1
$0 \rightarrow 1$	1	0	0	1	0
$1 \rightarrow 0$	x	0	0	Q	$\overline{Q}$
x	x	1	0	0	1
x	x	0	0	1	0
x	x	1	1	1	1

Έτσι, η επιθυμητή λειτουργία επιτυγχάνεται κρατώντας τα pins Set και Reset LOW, και χρησιμοποιώντας ως έξοδο το pin Q.

Συνδεσμολογία κυκλωμάτων

Αρχικά συνδέονται τα κατάλληλα pins με την γείωση και την τάση ανάλογα. Ακολουθούν τα pins των σημάτων των encoders. Το πρώτο σήμα του encoder (signal A) συνδέεται με το CP0 (4520) και το Clock (4013). Αυτό είναι το σήμα του οποίου διαιρείται η συχνότητα και ταυτόχρονα λειτουργεί σαν ρολόι στον binary counter. Το δεύτερο σήμα του encoder (signal B) που χρησιμοποιείται στον προσδιορισμό της φοράς περιστροφής συνδέεται με το pin Data (4013).



Έτσι τον encoderR είναι

κίτρινο	2CP0
μπλε	D2

και για τον encoderL είναι

κίτρινο	D1
μπλε	1CP0

Ακολουθεί η σύνδεση των εξόδων των κυκλωμάτων με τα pins εισόδου του Arduino UNO. Το σήμα A του encoderR με υποδιπλασιασμένη συχνότητα συνδέεται στο pin 2. Το σήμα προσδιορισμού της φοράς περιστροφής του encoderR συνδέεται στο pin 4. Αντίστοιχα το σήμα A του encoderL στο pin 3 και το άλλο στο pin 5.

1Q0	pin 2
Q1	pin 4
2Q0	pin 3
Q2	pin 5

Β΄.1.3 Πυκνωτές

Οι πυκνωτές τοποθετούνται όπως αναλύθηκε για την εξομάλυνση της τάσης του οδηγού του κινητήρα που συμβάλει στην καλύτερη και αθόρυβη λειτουργία των κινητήρων. Έτσι, τοποθετείται ένας πυκνωτής ΜΚΤ με χωρητικότητα 1μF και τάσης 25V *παράλληλα* στον κινητήρα.

Β΄.1.4 Σύνδεση τμημάτων

Arduino-motor driver

Για την σύνδεση του Arduino με τον motor driver ακολουθείται η παρακάτω αντιστοιχία των pins

Arduino Pin	Driver Pin
9	M1_PWM
10	M2_PWM
11	M1_EN
12	M2_EN

Οδηγός κινητήρων-κινητήρες

Η σύνδεση των κινητήρων με τον οδηγό γίνεται με την ακόλουθη αντιστοιχία: για τον δεξιό κινητήρα

καλώδιο κινητήρα	motor driver
κόκκινο	M1+
μαύρο	M1-

και για τον αριστερό

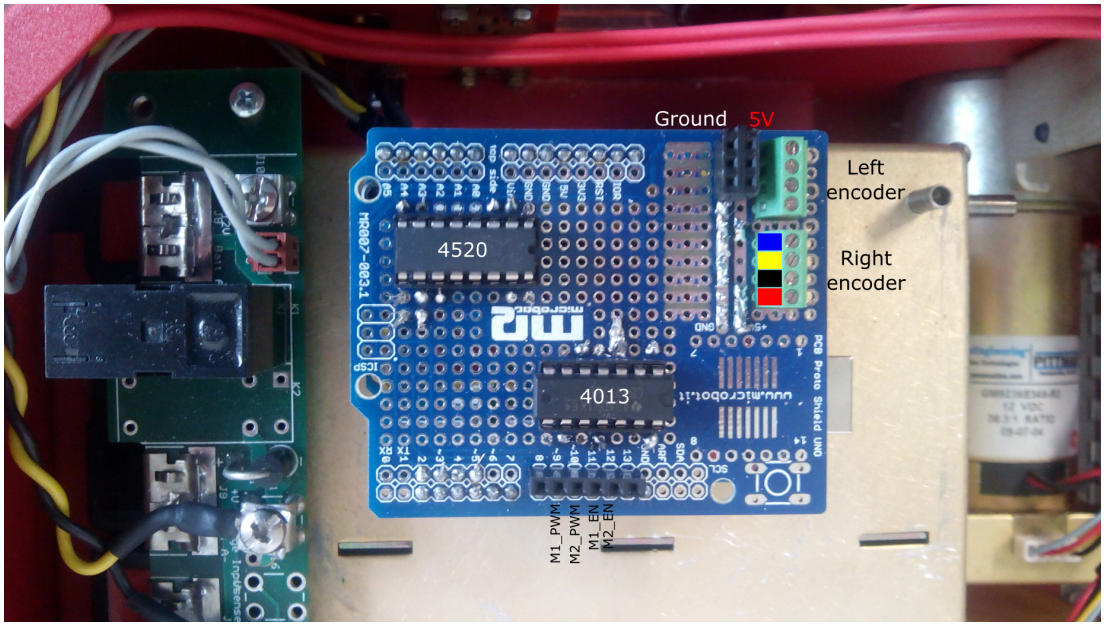
καλώδιο κινητήρα	motor driver
κόκκινο	M2-
μαύρο	M2+

Arduino-encoders

Σύμφωνα με την σήμανση στην πλακέτα (L & R) συνδέονται οι encoders με την παρακάτω σειρά *για κάθε* encoder, κόκκινο, μαύρο, κίτρινο και μπλε καλώδιο από αριστερά προς δεξιά.

Αναλογικές είσοδοι

Οι αναλογικές είσοδοι συνδέονται στα pins A0 και A1 της πλακέτας του Arduino.



Β΄.2 Pioneer 2AT

Β΄.2.1 Χρησιμοποιηθέντα εξαρτήματα

Επειδή για κάθε ζεύγος τροχών χρησιμοποιούνται δύο κινητήρες υπάρχει μεγαλύτερη απαίτηση ρεύματος σε σχέση με το δίτροχο ρομπότ. Έτσι, χρησιμοποιήθηκε ένας οδηγός για κάθε ζεύγος κινητήρων. Η λίστα των εξαρτημάτων είναι

1x	Arduino UNO R3
2x	DC Motor Driver
2x	1μF MKT πυκνωτής τάσης 25V
1x	prototyping board

Β΄.2.2 Πλακέτα

Στο τετράτροχο ρομπότ η αντιστοιχία των καλωδίων των encoders είναι αντίθετη με αυτή του δίτροχου. Έτσι, η αντιστοίχηση είναι για τον encoder L

κόκκινο	τάση
μαύρο	γείωση
κίτρινο	signal A
μπλε	signal B

και για τον encoder R

κόκκινο	τάση
μαύρο	γείωση
μπλε	signal A
κίτρινο	signal B

Η σύνδεση των καλωδίων των encoders με τα pins του Arduino είναι για την αριστερή πλευρά

κίτρινο	pin 2
μπλε	pin 4

και για την δεξιά πλευρά

μπλε	pin 3
κίτρινο	pin 5

Η διαφοροποίηση αυτή συνέβη καθώς θεωρήθηκε αρχικά λάθος θετική φορά κίνησης στο ρομπότ ενώ είχε ήδη αναπτυχθεί η πλακέτα.

Β΄.2.3 Πυκνωτές

Όπως και στο Pioneer3DX προστίθεται ένας πυκνωτής MKT χωρητικότητας 1μF και τάσης 25V *παράλληλα* σε κάθε κινητήρα.

Β΄.2.4 Σύνδεση τμημάτων

Arduino-motor driver

Όπως αναφέρθηκε χρησιμοποιήθηκε ένας οδηγός για κάθε ζεύγος κινητήρων, με αποτέλεσμα κάθε οδηγός λαμβάνει την ίδια εντολή και για τους δύο κινητήρες. Για την σύνδεση του Arduino με τον δεξιό motor driver ακολουθείται η παρακάτω αντιστοιχία των pins

Arduino Pin	Right Driver Pin
9	M1_PWM & M2_PWM
11	M1_EN & M2_EN

Για την σύνδεση με τον αριστερό motor driver είναι

Arduino Pin	Left Driver Pin
10	M1_PWM & M2_PWM
12	M1_EN & M2_EN

Οδηγοί κινητήρων-κινητήρες

Προσοχή. Επειδή η κίνηση κάθε ζεύγους τροχών απαιτεί οι δύο κινητήρες που συμμετέχουν να περιστρέφονται **αντίροπα** η σύνδεση των πόλων των κινητήρων για το δεξιό τμήμα είναι:
για τον πάνω κινητήρα

καλώδιο κινητήρα	motor driver
κόκκινο	M1+
μαύρο	M1-

για τον κάτω κινητήρα

καλώδιο κινητήρα	motor driver
κόκκινο	M1-
μαύρο	M1+

Αντίστοιχα για το αριστερό τμήμα:
για τον πάνω κινητήρα

καλώδιο κινητήρα	motor driver
κόκκινο	M2-
μαύρο	M2+

και για τον κάτω κινητήρα

καλώδιο κινητήρα	motor driver
κόκκινο	M1+
μαύρο	M1-

Arduino-encoders

Κάθε καλώδιο συνδέεται στο αντίστοιχο σύμφωνα με την σήμανση στην πλακέτα (L & R).

Αναλογικές είσοδοι

Οι αναλογικές είσοδοι συνδέονται στα pins A0 και A1 της πλακέτας του Arduino.

Παράρτημα Γ΄

Manual

Γ΄.1 Εγκατάσταση Λογισμικού

Γ΄.1.1 Arduino IDE

Η εγκατάσταση του IDE γίνεται με τις εντολές

```
$ sudo apt-get update
$ sudo apt-get install arduino arduino-core
```

Γ΄.1.2 ROS

Για την εγκατάσταση του ROS γίνεται παραπομπή στο site[1] του ROS

Γ΄.1.3 ROS packages

rosserial

Αρχικά γίνεται η εγκατάσταση του πακέτου rosserial εκτελώντας [2]

```
$ sudo apt-get update
$ sudo apt-get install rosserial
$ sudo apt-get install rosserial-arduino.
```

ARM computer. Σε περίπτωση που το Arduino επικοινωνεί με Η/Υ που έχει επεξεργαστή ARM (πχ. Odroid, Raspberry Pi κ.α) το πακέτο rosserial-arduino πιθανώς να μην λειτουργήσει σωστά. Σε αυτήν την περίπτωση εγκαθίσταται το πακέτο

[1] http://wiki.ros.org/ROS/Installation
[2] http://wiki.ros.org/rosserial_arduino/Tutorials

```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/chuck-h/rosserial.git
$ cd ..
$ catkin_make
$ catkin_make install
$ source catkin_ws/install/setup.bash
```

Το πακέτο rosserial επιτρέπει την σειριακή επικοινωνία του ROS με άλλες συσκευές. Το πακέτο rosserial_arduino εμπεριέχει κώδικες για την μετατροπή των πακέτων του ROS σε συμβατές βιβλιοθήκες για το Arduino.

arduino_mr

Για την αποστολή μηνυμάτων στο ρομπότ απαιτείται το πακέτο arduino_mr που περιέχει τα μηνύματα. Η εγκατάσταση πραγματοποιείται με απλή αντιγραφή του φακέλου στο `~/catkin_ws/src/` και στην συνέχεια τρέχοντας την εντολή

```
$ catkin_make
```

ενώ το σύστημα βρίσκεται στο directory `~/catkin_ws/`.

odometry

Αντίστοιχα, αντιγράφεται ο φάκελος του πακέτου της οδομετρίας (odometry) στη διεύθυνση `~/catkin_ws/src/` και εκτελείται η εντολή

```
$ catkin_make
```

joy_to_arduino

Το πακέτο αυτό χρησιμεύει στον τηλεχειρισμό του ρομπότ μέσω χειριστηρίου. Αρχικά, απαιτείται η εγκατάσταση του πακέτου joy

```
$ rosdep install joy
$ rosmake joy
```

Αφού συνδεθεί το χειριστήριο στον υπολογιστή, μπορεί να βρεθεί στη διεύθυνση

```
$ ls /dev/input/
```

και έχει την μορφή jsX. Έλεγχος του χειριστηρίου γίνεται μέσω της εντολής

```
$ sudo jstest /dev/input/jsX
```

Αν οι τιμές που εμφανίζονται αλλάζουν κατά την κίνηση του χειριστηρίου τότε λειτουργεί σωστά. Στην συνέχεια ορίζονται τα κατάλληλα δικαιώματα για την χρήση του χειριστηρίου

```
$ sudo chmod a+rw /dev/input/jsX
```

Το χειριστήριο τώρα είναι έτοιμο προς χρήση. Τελικό βήμα είναι η αντιγραφή του φακέλου joy_to_arduino στην διεύθυνση `~/catkin_ws/src` και εκτέλεση της εντολής

```
$ catkin_make
```

Για ευκολία ορίζεται η παράμετρος του ROS που περιγράφει την διεύθυνση αυτή

```
$ rosparam set joy_node/dev "/dev/input/jsX"
```

Γ΄.1.4 Φόρτωση κώδικα Arduino

Η παρακάτω διαδικασία χρειάζεται στην περίπτωση που ζητείται φόρτωση νέου κώδικα στο Arduino. Αν το Arduino φέρει ήδη τον κώδικα δεν απαιτείται η δημιουργία των παρακάτω βιβλιοθηκών καθώς χρησιμεύουν μόνο στην διαδικασία του compiling και building του κώδικα. Οι βιβλιοθήκες του Arduino αποθηκεύονται στην διεύθυνση `~/sketchbook/libraries`.

Δημιουργία βιβλιοθηκών από το ROS

Για να χρησιμοποιηθεί το πακέτο arduino_mr στο Arduino, δηλαδή τα μηνύματα επικοινωνίας με το ROS απαιτούνται δύο βήματα.

Πρώτον πρέπει να διαγραφεί ο φάκελος

`~/sketchbook/libraries/ros_lib`

εκτελώντας

```
$ rm -rf ~/sketchbook/libraries/ros_lib
```

αν τυχόν υπάρχει στο σύστημα. Τέλος, πρέπει να εκτελεστεί η εντολή

```
$ rosrun rosserial_arduino make_libraries.py ~/sketchbook/libraries/
```

δημιουργώντας τις συμβατές βιβλιοθήκες του ROS στο Arduino.

Βιβλιοθήκες Arduino

Δύο επιπλέον βιβλιοθήκες είναι απαραίτητες για την φόρτωση του κώδικα στο Arduino, η `PWM` και η `PID_motor`. Έτσι, αντιγράφονται οι δύο αντίστοιχοι φάκελοι στην διεύθυνση `~/sketchbook/libraries/`.

Φόρτωση κώδικα

Για την αποθήκευση του κώδικα στο Arduino ανοίγεται απλά το αρχείο (*.ino) με το Arduino IDE και στην συνέχεια επιλέγεται το Upload.

Γ΄.2 Χρήση ρομπότ

Για την λειτουργία του ρομπότ απαιτείται η εκτέλεση του κόμβου rosserial_python/ serial_node.py στον μικρο-υπολογιστή του ρομπότ, ο οποίος αναλαμβάνει την επικοινωνία Arduino - μικρο-ελεγκτή (είσοδος-έξοδος μηνυμάτων).

Τα topics που χρησιμοποιούνται είναι τα **arduino_input_<robot>** για τις εντολές **κίνησης**, **feedback_<robot>** για την **ανάδραση** (encoders, αναλογικές είσοδοι κ.α.) και το **gains_input_<robot>** για την **αλλαγή των κερδών ελέγχου**.

Η θέση <robot> αντιστοιχεί στο εκάστοτε ρομπότ που χρησιμοποιείται. Έτσι για το Pioneer 3DX είναι <robot>=3dx και για το Pioneer 2AT <robot>=2at.

Σημείωση. Κατά τον τερματισμό του κόμβου rosserial_python/serial_node.py αν δεν έχει σταλεί πρωτύτερα εντολή μηδενικής ταχύτητας, το ρομπότ θα συνεχίσει να κινείται. Για τον σκοπό αυτόν έχει αναπτυχθεί κόμβος που αναλαμβάνει την εκτέλεση του κόμβου της σειριακής επικοινωνίας rosserial_python/ serial_node.py και διαχειρίζεται τον τερματισμό του κόμβου αυτού για την αποφυγή παραμένουσας εντολής κίνησης μετά τον τερματισμό. Η λειτουργία δεν απαιτεί από τον χρήστη να στείλει εντολή μηδενικής ταχύτητας πριν τον τερματισμό ώστε να αποφευχθεί συνέχιση της κίνησης ενώ ο κόμβος serial_node.py δεν λειτουργεί, απλώς να σταματήσει οποιονδήποτε κόμβο δημοσιεύει στο topic της κίνησης. Ο κόμβος περιγράφεται στο κεφάλαιο Γ΄.2.4.

Στην συνέχεια παρουσιάζεται η μέθοδος λειτουργίας του ρομπότ **δίχως** την εκτέλεση του κόμβου αυτού. Η διαδικασία δεν διαφοροποιείται πέραν της εκτέλεσης διαφορετικού κόμβου στον μικρο-ελεγκτή όπως περιγράφεται και αυτή στην συνέχεια.

Από την στιγμή που εκτελεστεί ο κόμβος σειριακής επικοινωνίας, μόνη απαίτηση για την κίνηση του ρομπότ είναι η δημοσίευση στο topic arduino_input_<robot> του επιθυμητού μηνύματος κίνησης. Η εντολή κίνησης μπορεί να δημοσιευτεί μέσω εντολής αλλά και μέσω κόμβου γραμμένο σε C++ ή Python. Για περισσότερες πληροφορίες ο χρήστης παραπέμπεται στο site του ROS [3].

Τα συστήματα στο ROS συνεργάζονται μεταξύ τους και ανταλλάσσουν πληροφορίες με δύο μεθόδους. Σύμφωνα με την πρώτη ένας υπολογιστής λειτουργεί ως master του συνόλου και οι υπόλοιποι εξαρτώνται από αυτόν[4]. Η άλλη μέθοδος αξιοποιεί ένα πακέτο του ROS (Multimaster)[5] που επιτρέπει τον συγχρονισμό των διαφορετικών topics που λειτουργούν σε ένα σύνολο συστημάτων, κάθε ένα από τα οποία είναι master.

[3] http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(python)
[4] http://wiki.ros.org/ROS/Tutorials/MultipleMachines
[5] http://wiki.ros.org/multimaster

Στις παρακάτω οδηγίες χρησιμοποιείται η πρώτη μέθοδος καθώς έχει μικρότερες απαιτήσεις από τον μικρο-υπολογιστή. Για την επικοινωνία μεταξύ των συστημάτων ορίζεται ο κύριος υπολογιστής (αυτός που δίνει τα μηνύματα κίνησης) ως master και ο άλλος ως slave (ρομπότ). Για την δήλωση του master στον slave χρησιμοποιείται η διεύθυνση IP του πρώτου. Απαιτείται η έναρξη ενός ros core στον master ώστε να επικοινωνεί με αυτόν ο slave.

Master

Στον master υπολογιστή δημιουργείται σε νέο terminal window (`Ctrl+Alt+T`) ο ros core

```
$ roscore
```

Ο ros core πρέπει να τρέχει σε όλη την διάρκεια που λειτουργεί το ρομπότ. Έτσι, όποια άλλη λειτουργία πρέπει να γίνεται σε καινούριο terminal.

Για την επικοινωνία των δύο υπολογιστών απαιτείται η γνώση της διεύθυνσης IP του master. Αυτή βρίσκεται εύκολα με την εντολή

```
$ ifconfig
```

Η IP φαίνεται στην θέση inet addr:<>. Αν η σύνδεση είναι μέσω Wifi η θέση αυτή εντοπίζεται στην καρτέλα wlan0 και αν είναι μέσω Ethernet στην καρτέλα eth0.

Slave

Ο Η/Υ του ρομπότ πρέπει να έχει εκκινήσει πλήρως και να έχει πρόσβαση στο δίκτυο (1 λεπτό περίπου). Για απομακρυσμένη πρόσβαση στον υπολογιστή του ρομπότ (slave) γίνεται σύνδεση μέσω ssh

```
$ ssh <user>@<IP address>
```

Χρήσιμη αλλά όχι απαραίτητη είναι η δυνατότητα της χρήσης πολλών εικονικών κονσολών (πχ για την εκτέλεση δύο κόμβων) απομακρυσμένα. Για τον σκοπό αυτόν μπορεί να χρησιμοποιηθεί το πρόγραμμα screen[6]

```
$ screen
```

Χρήση screen. Για την δημιουργία νέας κονσόλας χρησιμοποιείται ο συνδυασμός `Ctrl+A` και `C`. Η μετακίνηση στην επόμενη κονσόλα γίνεται με `Ctrl+A` και `N` και στην προηγούμενη με `Ctrl+A` και `P`.

Αρχικά είναι απαραίτητη η εντολή

```
$ export LC_ALL=C
```

όπου εξαναγκάζει το απομακρυσμένο σύστημα να χρησιμοποιήσει κοινές ρυθμίσεις με το κύριο. Στην συνέχεια δηλώνεται η διεύθυνση του master

[6] `https://help.ubuntu.com/community/Screen`

όπου τρέχει ο ros core συνοδευόμενη από την port της σύνδεσης (προεπιλεγμένη η 11311) έπειτα από <:>

```
$ export ROS_MASTER_URI=http://<Master IP>:11311
```

Τώρα μπορεί να εκτελεστεί ο κόμβος σειριακής επικοινωνίας με το Arduino στον slave Η/Υ

```
$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0
```

Ο κόμβος serial_node.py δέχεται ως όρισμα την διεύθυνση της σειριακής σύνδεσης με το Arduino. Στο Λ/Σ Ubuntu είναι της μορφής ttyACMx. Συνήθως όταν δεν είναι συνδεδεμένη άλλη συσκευή στον υπολογιστή, η διεύθυνση είναι ttyACM0. Οι διευθύνσεις που έχουν δοθεί σε συσκευές από το σύστημα φαίνονται στον φάκελο `/dev/`. Ένας εναλλακτικός τρόπος να βρεθεί η διεύθυνση σε περίπτωση που δεν είναι η παραπάνω είναι μέσω του Arduino IDE που την εμφανίζει κάτω δεξιά της οθόνης.

Όπως προαναφέρθηκε ο κόμβος αυτός μπορεί να αντικατασταθεί με τον κόμβο arduino_mr/pioneer_<robot>.py που αναλαμβάνει τον τερματισμό της σειριακής επικοινωνίας και το σταμάτημα της κίνησης. Έτσι, σε αυτή την περίπτωση εκτελείται

```
$ rosrun arduino_mr pioneer_<robot>.py
```

Συνίσταται στον χρήστη να συμβουλευτεί το κεφάλαιο Γ΄.2.4 για τις απαραίτητες ρυθμίσεις για χρήση σε νέο σύστημα.

Σημείωση. Για την εκτέλεση ενός κόμβου στο slave σύστημα δεν εκτελείται ξεχωριστός ros core στο σύστημα αυτό.

Όταν εκτελείται ο κόμβος και είναι συνδεδεμένος με το Arduino φαίνονται τα topics που είναι εγγεγραμμένος τόσο στον slave όσο και στον master. Αυτά είναι τα feedback_<robot>, arduino_input_<robot> και gains_input_<robot>. Μπορεί να γίνει έλεγχος μέσω της εντολής στον master

```
$ rostopic list
```

Για έλεγχο της λειτουργίας του δικτύου και της δυνατότητας μετάδοσης των πληροφοριών εκτελείται η εντολή

```
$ rostopic echo feedback_<robot>
```

Αν τα μηνύματα ανανεώνονται σημαίνει ότι δεν υπάρχει κάποιο πρόβλημα.

Πιθανά Προβλήματα

Υπάρχει περίπτωση τα topics να είναι ορατά, αλλά να μην εμφανίζονται τα δεδομένα. Για την επίλυση προστίθενται στο αρχείο `/etc/hosts` οι IP των συνεργαζόμενων συστημάτων. Η παρέμβαση αυτή πρέπει να γίνει και στον master όπου προστίθεται η διεύθυνση του/των slave, αλλά και στον/στους slave όπου προστίθεται του master.

Σε αυτόν τον τρόπο επικοινωνίας η αποστολή του μηνύματος κίνησης στο ρομπότ γίνεται μέσω του Wifi. Έτσι, αν υπάρχει καθυστέρηση ή άσχημο σήμα είναι πιθανό να υπάρξει μία καθυστέρηση των δεδομένων. Έπειτα από δοκιμές παρατηρήθηκε ότι αυτό συμβαίνει σπάνια.

Αποστολή μηνύματος κίνησης

Για την αποστολή μηνύματος κίνησης από τον master δημοσιεύεται μήνυμα τύπου arduino_mr/ arduino_input στο topic arduino_input_<robot>. Η δημοσίευση αυτή μπορεί να γίνει μέσω εντολής σε νέο terminal

`$ rostopic pub arduino_input_<robot> arduino_mr/arduino_input -- <mode> <data1> <data2>`.

Το περιεχόμενο των data και mode αναλύεται στο παράρτημα Α΄.2.1 περιγραφής του software.

Έτσι, αν για παράδειγμα το ρομπότ πρέπει να γίνει έλεγχος των ταχυτήτων του κέντρου μάζας με γραμμική κίνηση ίση με $0.5[\frac{m}{s}]$ και περιστροφική ταχύτητα ίση με $0.7[\frac{rad}{s}]$ η εντολή είναι

`$ rostopic pub arduino_input_3dx arduino_mr/arduino_input -- 1 0.5 0.7`.

Αποστολή μηνύματος αλλαγής κερδών ελέγχου PID

Για την αλλαγή των κερδών του ελεγκτή εκτελείται σε νέο terminal

`$ rostopic pub gains_input_<robot> arduino_mr/PID_motors_gains -- '[[<Kp>, <Ki>, <Kd>], [<Kp>, <Ki>, <Kd>]]'`

με πρώτα τα κέρδη του δεξιού κινητήρα. Η περιγραφή του περιεχόμενου του μηνύματος γίνεται στο παράρτημα Α΄.2.1. Για να γίνει επιτυχής αλλαγή των κερδών πρέπει ο κόμβος rosserial_python/serial_node.py που είναι συνδεδεμένος στο Arduino και περιγράφεται στο παραπάνω κεφάλαιο, να τρέχει ήδη.

Για παράδειγμα αν τα νέα κέρδη για τον δεξιό κινητήρα είναι

Kp	400
Ki	50
Kd	0

και για τον αριστερό

Kp	500
Ki	80
Kd	0

Η εντολή είναι

`$ rostopic pub gains_input_3dx arduino_mr/PID_motors_gains -- '[[400,50,0], [500,80,0]]'`

Τα κέρδη αυτά θα παραμείνουν αποθηκευμένα στον ελεγκτή μέχρι να αποσυνδεθεί από την τροφοδοσία. Για μόνιμη αλλαγή των κερδών χρειάζεται να μεταβληθεί ο κώδικας του Arduino όπως περιγράφεται στην ενότητα Γ΄.3.

Γ΄.2.1 Οδομετρία

Έχει αναπτυχθεί κόμβος οδομετρίας μόνο για το δίτροχο ρομπότ για τους λόγους που αναλύθηκαν. Ο κόμβος αυτός μπορεί να εκτελεστεί τόσο στον master όσο και στον slave υπολογιστή. Βέβαια, είναι προτιμότερο να εκτελείται στον υπολογιστή που επικοινωνεί απευθείας με το Arduino, δηλαδή τον slave, για να μην επηρεάζονται οι υπολογισμοί από καθυστερήσεις του δικτύου.

```
$ rosrun odometry odometry_node_3dx
```

Αν ο έλεγχος του slave γίνεται μέσω ssh τότε πρέπει να έχει χρησιμοποιηθεί η εντολή screen πριν την εκτέλεση οποιουδήποτε κόμβου. Αν λειτουργεί, με τον συνδυασμό `Ctrl+A` και `C` δημιουργείται νέα κονσόλα. Στην συνέχεια εκτελείται η εντολή έναρξης του κόμβου της οδομετρίας.

Τα δεδομένα δημοσιεύονται στο topic odometry, του οποίου το μήνυμα περιγράφεται στο παράρτημα Α΄.2.1 messages.

Γ΄.2.2 Παρακολούθηση μηνυμάτων

Για την πρόσβαση σε οποιοδήποτε topic απαιτείται η εντολή

```
$ rostopic echo <topic name>
```

Η λίστα με τα διαθέσιμα topics φαίνεται με την εντολή

```
$ rostopic list
```

Τα περιεχόμενα των μηνυμάτων κάθε topic αναλύονται στο παράρτημα Α΄.2.1 αλλά μπορούν να ευρεθούν με την εντολή

```
$ rostopic info <topic>
```

για την εύρεση του μηνύματος κάθε topic και

```
$ rosmsg show <topic message>
```

για το περιεχόμενο του μηνύματος.

Γ΄.2.3 Τερματισμός λειτουργίας

Για τον τερματισμό της λειτουργίας πρέπει πρώτα να σταλεί εντολή στον μικρο-ελεγκτή να σταματήσει την κίνηση. Αυτό ισοδυναμεί με mode=1 ή 2 ή 3, data1=0, data2=0. Η ορισμός του **mode=3** επιφέρει πιο ομαλό σταμάτημα της κίνησης αφού δεν παρεμβάλλεται ο PI έλεγχος.

Αν έχει εκτελεστεί ο κόμβος arduino_mr/pioneer_<robot>.py όπως αναφέρθηκε παραπάνω και αναλύεται στο κεφάλαιο Γ΄.2.4 δεν απαιτείται κάποια ενέργεια για τον τερματισμό του ρομπότ παρά μόνο ο τερματισμός όποιας άλλης δημοσίευσης στο topic arduino_input_<robot>.

Γ΄.2.4 Κόμβος ασφαλούς τερματισμού σειριακής επικοινωνίας

Ο κόμβος arduino_mr/pioneer_<robot>.py κατά την εκτέλεση του καλεί τον κόμβο σειριακής επικοινωνίας rosserial_python/serial_node.py χρησιμοποιώντας το πακέτο roslaunch. Αναλαμβάνει την διαχείριση του σήματος τερματισμού του κόμβου (Ctrl + C) έτσι ώστε το όχημα να μην συνεχίσει την κίνηση του μετά τον τερματισμό.

Αυτό επιτυγχάνεται δημοσιεύοντας μήνυμα μηδενικής τάσης στους κινητήρες μόλις ληφθεί η εντολή τερματισμού. Στην συνέχεια αναλαμβάνει τον τερματισμό του κόμβου σειριακής επικοινωνίας.

Στον κώδικα του κόμβου ορίζεται η διεύθυνση της σειριακής επικοινωνίας υπό την μεταβλητή args. Προεπιλεγμένη είναι η `/dev/ttyACM0`, αλλά είναι πιθανό να χρειάζεται τροποποίηση με την σωστή διεύθυνση. Η διεύθυνση του Arduino βρίσκεται με τον τρόπο που αναφέρθηκε παραπάνω.

Γ΄.3 Μόνιμη αλλαγή κερδών ελεγκτή

Για την μόνιμη αλλαγή των κερδών του ελεγκτή απαιτείται μεταβολή του κώδικα του Arduino. Στο αντίστοιχο αρχείο (*.ino) ανάλογα με το ρομπότ ορίζονται οι επιθυμητές τιμές των κερδών όπου αντιστοιχούν στις τιμές Kp, Ki και Kd στην αρχή του κώδικα μετά την δήλωση των σταθερών (#define).

```
#include <ros.h> // ros library
#include <arduino_mr/feedback_int.h> // feedback message
#include <arduino_mr/arduino_input.h> // input message
#include <arduino_mr/PID_motors_gains.h> //gains message
#include <PWM.h> // library for setting desired pwm frequency
#include <PID_motor.h> // library for pid

#define DirR 11 // connected to pin M1_EN in motor driver
#define DirL 12 // connected to pin M2_EN in motor driver
#define PwmR 9 // connected to pin M1_PWM in motor driver
#define PwmL 10 // connected to pin M2_PWM in motor driver
#define EncR_PinA 2 // Right encoder PinA
// EncR (right) pinB is on pin 4
#define EncL_PinA 3 // Left encoder PinA
// EncL (left) pinB is on pin 5

// PID gains
int Kp = 850; // proportional gain
int Ki = 200; // integral gain
int Kd = 0;  // derivative gain
```

Στην συνέχεια συνδέεται το Arduino στον υπολογιστή και επιλέγεται το Upload στο Arduino IDE.

Γ΄.4 Χρήση κόμβου τηλεχειριστηρίου

Ο κόμβος joy_to_arduino/ teleop_MR_<robot> δημοσιεύει στο topic arduino_input_ <robot> εντολές για την κίνηση του ρομπότ σύμφωνα με τις τιμές του τηλεχειριστηρίου. Δεδομένου ότι έχουν εγκατασταθεί όλα τα απαραίτητα πακέτα (βλ. παράρτημα Γ΄.1.3) εκτελούνται οι κόμβοι

```
$ rosrun joy joy_node
$ rosrun joy_to_arduino teleop_MR_<robot>
```

Καλό είναι να γίνει δοκιμή των εντολών του χειριστηρίου δίχως να τροφοδοτούνται οι κινητήρες. Αυτό γιατί η αντιστοιχία των εξόδων του χειριστηρίου μπορεί να διαφέρουν από αυτού για το οποίο σχεδιάστηκε ο κόμβος. Έλεγχος γίνεται παρατηρώντας το topic arduino_input_<robot> που δημοσιεύει ο teleop_MR_<robot>, ενώ ταυτόχρονα χρησιμοποιείται το χειριστήριο. Αν χρειάζεται μεταβολή των ρυθμίσεων γίνεται παραπομπή στο παράρτημα Γ΄.5.

Γ΄.5 Αλλαγή ρυθμίσεων τηλεχειριστηρίου

Για να αλλάξει η αντιστοίχηση των πλήκτρων στον τηλεχειρισμό του ρομπότ χρειάζεται να μεταβληθεί ο κώδικας teleop_MR_<robot>.cpp που βρίσκεται στον φάκελο

```
~/catkin_ws/src/joy_to_arduino/
```

Ο κόμβος joy εξάγει ένα διάνυσμα buttons[] το οποίο παρέχει τα δεδομένα των πλήκτρων του χειριστηρίου και ένα διάνυσμα axes[] που περιέχει τα δεδομένα των μοχλών. Η αντιστοίχιση κάθε πλήκτρου ή μοχλού του χειριστηρίου με το διάνυσμα buttons[] ή axes[] ελέγχεται εύκολα με την παρακολούθηση του topic joy (`$ rostopic echo joy`) κινώντας και σημειώνοντας την θέση στο εκάστοτε διάνυσμα. Η αντιστοίχηση των τιμών γίνεται στην συνάρτηση joyCallback και παρακάτω φαίνονται τμήματα του κώδικα για απλότητα και καλύτερη κατανόηση.

Αρχικά ορίζεται η αντιστοίχιση για τον ορισμό του mode λειτουργίας του ρομπότ

```
// map button to mode
if (joy->buttons[0]==1){ message.mode=1; current_mode=1; }
```

Όπου το πρώτο στοιχείο του buttons[] (buttons[0]) αντιστοιχεί στο mode λειτουργίας 1. Η αντιστοίχηση των εντολών ελέγχου γίνεται αντίστοιχα

```
if (current_mode == 1) //mode 1 mapping
  {
    message.data1 = l_scale * joy->axes[3];
    message.data2 = a_scale * joy->axes[2];
  }
```

Επειδή η τιμές των στοιχείων του axes[] βρίσκονται στο διάστημα $[0, 1]$ η αντίστοιχη τιμή πολλαπλασιάζεται με μία μεταβλητή (l_scale ή a_scale) αντιστοιχίζοντας τη έτσι στο διάστημα $[0, l\_scale]$ ή $[0, a\_scale]$. Το περιεχόμενο του μηνύματος arduino_input (αντικειμένου message στην περίπτωση της συνάρτησης αυτής) αναλύεται στο παράρτημα Α΄.2.1. Αρκεί λοιπόν η αλλαγή του στοιχείου του button[] ή του axes[] για την μεταβολή της αντιστοίχησης.

Τέλος ενώ το terminal βρίσκεται στην διεύθυνση

`~/catkin_ws/`

εκτελείται

`$ catkin_make`

Γ΄.6 Τροποποίηση μηνύματος ανάδρασης

Η μεταβολή του μηνύματος feedback γίνεται στο αρχείο

`~/catkin_ws/arduino_mr/msg/feedback_int.msg`

Εκεί μπορεί να προστεθεί επιπλέον μεταβλητή αν είναι απαραίτητο.

Μετά την αλλαγή εκτελείται η εντολή

`~/catkin_ws$ catkin_make`

Η διαδικασία αυτή πρέπει να εκτελεστεί σε όλους τους υπολογιστές που συνεργάζονται και χειρίζονται το νέο μήνυμα.

Επίσης, στον κώδικα του Arduino προστίθενται στο αντικείμενο feed εντός της συνάρτησης publish_feedback() οι νέες πληροφορίες.

Στην συνέχεια πρέπει να δημιουργηθούν εκ νέου οι αντίστοιχες βιβλιοθήκες του ROS για το Arduino (βλ. Γ΄.1.4) και να πραγματοποιηθεί η φόρτωση του στον ελεγκτή.

Περιορισμοί. Λόγω των μικρών δυνατοτήτων του ελεγκτή προτείνεται να μην αυξηθεί πολύ το μέγεθος των μηνυμάτων γιατί είναι πιθανό να μην επαρκεί η εσωτερική του μνήμη. Αν παρόλα αυτά είναι απαραίτητη η αύξηση του μηνύματος και παρατηρηθεί αδυναμία συγχρονισμού με τον κόμβο serial_node.py τότε προτείνεται η απενεργοποίηση της δυνατότητας αλλαγής των κερδών του ελέγχου μέσω ROS (βλ. παράρτημα Γ΄.7).

Γ΄.7 Ενεργοποίηση - Απενεργοποίηση δυνατότητας αλλαγής κερδών

Για την απενεργοποίηση της δυνατότητας αυτής γίνονται τροποποιήσεις στον κώδικα του Arduino. Συγκεκριμένα τοποθετούνται εντός σχολίων (σύμβολο //) τρεις γραμμές όπως φαίνονται παρακάτω.

```
//#include <arduino_mr/PID_motors_gains.h> //gains message
//ros::Subscriber<arduino_mr::PID_motors_gains> gains_sub("
    gains_input", &set_gains,1);
//nh.subscribe(gains_sub);
```

Τελικό βήμα είναι η φόρτωση του κώδικα.

Βιβλιογραφία

[1] John J. Craig, *Εισαγωγή στη Ρομποτική: Μηχανική και Αυτόματος Έλεγχος*.

[2] Jason M. O'Kane, *A Gentle Introduction to ROS*

[3] Paul J. Deitel, *C How to Program (5th Edition)*