ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ Μ.Κ. & Α.Ε.
Εργαστήριο Αυτομάτου Ελέγχου

Διπλωματική Εργασία

ΟΡΓΑΝΩΣΗ ΛΟΓΙΣΜΙΚΟΥ ΡΟΜΠΟΤΙΚΟΥ ΨΑΡΙΟΥ ΣΤΟ ROS ΚΑΙ ΑΝΑΠΤΥΞΗ 3^{ΟΥ} ΑΞΟΝΑ ΓΙΑ ΤΟ ΦΟΡΕΙΟ ΡΥΜΟΥΛΚΗΣΗΣ ΤΟΥ

Σπύρος Χατζηζαχαρίας

Επιβλέπων Καθηγητής: Ε. Γ. Παπαδόπουλος

ΑΘΗΝΑ 2019

Περίληψη

Τα ρομποτικά συστήματα αποτελούνται από πολλά επιμέρους υποσυστήματα. Για την πιο αποδοτική όμως λειτουργία των ρομπότ αυτών, είναι χρήσιμη η ύπαρξη μίας κεντρικής μονάδας ελέγχου, η οποία θα ενσωματώνει όλα τα επιμέρους υποσυστήματα του ρομπότ σε ένα ενιαίο σύστημα (κεντρικός ηλεκτρονικός υπολογιστής). Η υλοποίηση της κεντρικής μονάδας ελέγχου αυτής, καθιστά ευκολότερο τον συγχρονισμό των λειτουργιών του ρομπότ και βελτιώνει την ολοκληρωμένη επίβλεψη σε πραγματικό χρόνο από τον χειριστή. Επιπρόσθετα, επιταχύνει τη διαδικασία διάγνωσης σφαλμάτων μέσω της χρήσης αισθητήρων ανατροφοδότησης (feedback). Η διπλωματική αυτή εργασία πραγματεύεται αρχικά την ενσωμάτωση του συστήματος ελέγχου θέσης ανοικτού βρόχου μικρού μεγέθους βιομιμητικού ρομποτικού ψαριού, σε έναν κεντρικό ηλεκτρονικό υπολογιστή με εγκατεστημένο λειτουργικό σύστημα ROS (Robot Operating System). Ο ηλεκτρονικός υπολογιστής αυτός, είναι υπεύθυνος για την αποστολή της απαραίτητης πληροφορίας στον μικροεπεξεργαστή του ρομποτικού ψαριού, ώστε αυτό να ακολουθεί επιθυμητή επίπεδη τροχιά. Το σύστημα προϋποθέτει δυνατότητα λήψης δεδομένων από περιφερειακή σειριακή συσκευή, που θα διευκολύνει την μελλοντική υλοποίηση ελέγχου κλειστού βρόχου με ανατροφοδότηση από κάμερα. Η υλοποίηση του συστήματος αυτού πραγματοποιήθηκε με τη χρήση ανοικτού λογισμικού που αναπτύχθηκε στο Εργαστήριο Αυτομάτου Ελέγχου. Η επικοινωνία του κεντρικού συστήματος ελέγχου με τα περιφερειακά συστήματα λήψης και αποστολής δεδομένων πραγματοποιήθηκε με το σειριακό πρωτόκολλο επικοινωνίας RS-232 και με τη χρήση χαμηλού κόστους ενσωματωμένου υπολογιστικού συστήματος. Ο δεύτερος στόχος της εργασίας είναι ο σχεδιασμός και η κατασκευή μίας οριζόντιας πλατφόρμας τριών βαθμών ελευθερίας, η οποία προσαρμόζεται πάνω στο φορείο ρυμούλκησης της δεξαμενή υποβρυχίων πειραματικών δοκιμών του Εργαστηρίου Αυτομάτου Ελέγχου. Η πλατφόρμα αυτή παρέχει τη δυνατότητα στήριξης ηλεκτρικού κινητήρα συνεχούς ρεύματος, που είναι υπεύθυνος για την περιστροφική κίνηση κατακόρυφης ατράκτου συνδεδεμένης με το ρομποτικό ψάρι. Οι προδιαγραφές που τέθηκαν αφορούσαν μεταξύ άλλων την εύκολη συναρμολόγηση και αποσυναρμολόγηση της πλατφόρμας, την ενσωμάτωση της στο υπάρχον σύστημα κίνησης (φορείο ρυμούλκησης) καθώς και την αυτοτέλεια του μηχανισμού.

Λέξεις κλειδιά: Μηχανοτρονική, Συστήματα Αυτομάτου Ελέγχου, Έλεγχος ανοικτού βρόχου, Σειριακή Επικοινωνία, Λειτουργικό Σύστημα ROS, Μηχανολογικός Σχεδιασμός.

Abstract

Robotic systems are complex systems, consisting of several different subsystems. Robotic fishes are not the exception. However, in order to increase the efficiency of these robots, a central control unit is needed, which will integrate all the subsystems of the robot in one single system (central computer). The implementation of this central control unit enables easy synchronization of robot's operations and improves its real-time monitoring. Another major advantage that this control unit provides is the ability to accelerate the error diagnosis process through the use of feedback sensors. Initially, this Diploma Thesis deals with the integration of the open-loop position-control system of a small-scale biomimetic robotic fish, in a central computer running ROS (Robot Operating System). This computer is responsible for sending information to the microprocessor of the robotic fish so that it follows a desired flat trajectory. The system is capable of receiving data from peripheral serial device, which will facilitate the future implementation of closed-loop control with the use of feedback from a digital camera. The implementation of this system was accomplished with the use of open-source software, which was developed in the Control Systems Laboratory. The communication between the central control unit and the peripheral reception and transmission systems was accomplished with the use of the serial communication protocol RS-232 and low-cost embedded computing system. The second goal of this Diploma Thesis is the design and the construction of a horizontal platform with three degrees of freedom, which is attached to the towing carriage of the Control's System Lab experiment tank. This platform is capable of supporting an electric DC motor, which provides the rotary motion of a vertical shaft attached to the robotic fish. Some of the specifications set are related with the easy assembly and disassembly of the platform, the integration of the platform to the existing motion system (towing carriage) and the autonomy of the whole construction.

Key-words: Mechatronics, Automatic Control Systems, Open-loop Control, Serial Communication, Robot Operating System (ROS), Mechanical Design.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω όλους όσους με στήριξαν με οποιοδήποτε τρόπο κατά το χρονικό διάστημα εκπόνησης της διπλωματικής μου εργασίας. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Ε. Παπαδόπουλο για τις πολύτιμες συμβουλές του και την απαραίτητη καθοδήγησή του. Χάρη στην ευκαιρία που μου έδωσε να εκπονήσω την διπλωματική αυτή εργασία και με την υποστήριξη του απέκτησα σημαντικές γνώσεις και εμπειρία στον τομέα της μηχανολογίας και της μηχανοτρονικής.

Ευχαριστώ ιδιαίτερα τον υποψήφιο Διδάκτορα Μ. Μακροδημήτρη για τις πολύτιμες γνώσεις του που μοιράστηκε μαζί μου και γενικά για την βοήθεια, στις συμβουλές του και τις μακροσκελείς συζητήσεις μας.

Θερμές ευχαριστίες στους Ε. Σωτηρόπουλο, Γ. Βάλβη, Γ. Μπολονάκη, Θ. Καμπύλη και στους υποψήφιους Διδάκτορες Κ. Κουτσούκη, Κ. Μαχαιρά και Θ. Μαστρογεωργίου που πρόσφεραν καθοριστική βοήθεια σε θέματα της διπλωματικής μου, τόσο με τις θεωρητικές γνώσεις τους όσο και σε τεχνικά ζητήματα, αλλά και για την υποστήριξη τους.

Επίσης ευχαριστώ όλα τα παιδιά του Εργαστηρίου Αυτομάτου Ελέγχου για την προθυμία τους και την βοήθεια τους, αλλά και για το ευχάριστο περιβάλλον που δημιούργησαν στον χώρο εργασίας μου. Ιδιαίτερη μνεία στα παιδιά της ομάδας του τετραπόδου με τα οποία μοιραζόμασταν την ίδια αίθουσα.

Τέλος, θέλω να ευχαριστήσω τους γονείς μου, την αδερφή μου και τους φίλους μου για όλη την στήριξη τους, σε κάθε επίπεδο, όλα αυτά τα χρόνια.

Αφιερωμένο στους γονείς μου,
την αδερφή μου
και τον παππού μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Σκοπός Εργασίας

Τα ρομποτικά συστήματα είναι σύνθετα και πολύπλοκα συστήματα, αποτελούμενα από πολλά επιμέρους υποσυστήματα. Τα ρομποτικά ψάρια δεν αποτελούν εξαίρεση. Για την πιο αποδοτική όμως λειτουργία των ρομπότ αυτών, είναι χρήσιμη η ύπαρξη μίας κεντρικής μονάδας ελέγχου, η οποία θα ενσωματώνει όλα τα επιμέρους υποσυστήματα του ρομπότ κάτω από ένα ενιαίο σύστημα (κεντρικός ηλεκτρονικός υπολογιστής). Οι λόγοι είναι πολλοί. Από τον συγχρονισμό των λειτουργιών του ρομπότ, την ολοκληρωμένη επίβλεψη σε πραγματικό χρόνο από τον χειριστή, μέχρι και την ευκολότερη διάγνωση σφαλμάτων μέσω αισθητήρων ανατροφοδότησης (feedback).

Ο σκοπός της εργασίας επομένως είναι η ενσωμάτωση των υποσυστημάτων του ρομποτικού ψαριού του Εργαστηρίου Αυτομάτου Ελέγχου σε έναν κεντρικό ηλεκτρονικό υπολογιστή. Στο [1] προσομοιώνεται το σύστημα ελέγχου ανοικτού βρόχου του ρομποτικού ψαριού στο περιβάλλον Matlab-Simulink. Στην παρούσα διπλωματική εργασία έγινε προσπάθεια ενσωμάτωσης του συστήματος ελέγχου αυτού σε έναν κεντρικό ηλεκτρονικό υπολογιστή με λειτουργικό σύστημα (ROS). Επιπρόσθετα, υλοποιήθηκε η σύνδεση του μικροελεγκτή που βρίσκεται μέσα στο ρομποτικό ψάρι με την κεντρική μονάδα, για αποστολή πληροφορίας με σκοπό την κίνηση του ψαριού. Το κεντρικό σύστημα παρέχει επίσης την δυνατότητα λήψης δεδομένων από περιφερειακή συσκευή, με σκοπό την ολοκλήρωση συστήματος ελέγχου κλειστού βρόχου στο μέλλον, με ανατροφοδότηση θέσης του ρομποτικού ψαριού μέσω κάμερας (Raspberry Pi).

Η ενσωμάτωση των υποσυστημάτων του ρομποτικού ψαριού σε μία κεντρική μονάδα ελέγχου είναι μία σύνθετη διεργασία η οποία προκειμένου να πραγματοποιηθεί, στηρίζεται σε βασικές αρχές της Ρομποτικής, της Μηχανοτρονικής, του Αυτομάτου Ελέγχου, των Λειτουργικών Συστημάτων και των Επικοινωνιών, που είναι ιδιαίτερα χρήσιμες για την εκπαίδευση του μηχανικού. Μέσω αυτών των αρχών κατέστη δυνατός ο έλεγχος κίνησης του ουραίου πτερυγίου του ρομποτικού μέσω κεντρικού υπολογιστή, αλλά και της δημιουργίας ενός κεντρικού συστήματος, για ενσωμάτωση περαιτέρω υποσυστημάτων του ρομπότ στο μέλλον.

Ο δεύτερος στόχος της εργασίας αυτής, είναι ο σχεδιασμός και η κατασκευή μίας οριζόντιας πλατφόρμας τριών βαθμών ελευθερίας, η οποία προσαρμόζεται στον μηχανισμό φορείου ρυμούλκησης της δεξαμενής πειραματικών δοκιμών του Εργαστηρίου Αυτομάτου Ελέγχου. Η πλατφόρμα έχει την δυνατότητα στήριξης ηλεκτρικού κινητήρα, που είναι υπεύθυνος για την περιστροφική κίνηση της ατράκτου που συνδέει το ρομποτικό ψάρι, κατά τον κατακόρυφο άξονα z. Οι σχεδιαστικές προδιαγραφές της πλατφόρμας, καθορίστηκαν τόσο από συνεργασία της με τον υπάρχοντα μηχανισμό ρυμούλκησης όσο και από το περιβάλλον λειτουργίας της (π.χ. υγρό περιβάλλον της δεξαμενής δοκιμών). Έτσι, με την χρήση κατάλληλου σχεδιαστικού λογισμικού και με στόχο το μικρό βάρος και την κατασκευή όσον τον δυνατό περισσότερων τμημάτων της στο

μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου, σχεδιάστηκε, κατασκευάστηκε και συναρμολογήθηκε η πλατφόρμα τριών βαθμών ελευθερίας.

1.2 Βιβλιογραφική Ανασκόπηση

Στην παρούσα παράγραφο γίνεται μία προσπάθεια συνοπτικής καταγραφής της βιβλιογραφίας που χρησιμοποιήθηκε για την συγγραφή της διπλωματικής αυτής εργασίας.

Μία εισαγωγή στην ιστορία των λειτουργικών συστημάτων για ρομπότ και σε εφαρμογές τους παρουσιάζεται στα [2], [3], [4] και [5].

Μία σύντομη αναφορά στα διαθέσιμα λειτουργικά συστήματα για ρομπότ μέχρι και σήμερα γίνεται στα [6], [7] και [8].

Τα βασικά χαρακτηριστικά του λειτουργικού συστήματος ROS παρουσιάζονται στο βιβλίο [8], στη δημοσίευση [9], στις διπλωματικές [10] και [11], και στις ιστοσελίδες [5] και [12]. Αναφορά στις πρώτες ρομποτικές εφαρμογές που χρησιμοποίησαν το λειτουργικό σύστημα ROS παρουσιάζεται στο [13].

Όσον αφορά το πεδίο των υποβρυχίων επικοινωνιών, μία σύντομη περιγραφή των μεθόδων που χρησιμοποιούνται γενικά, γίνεται στις δημοσιεύσεις [14] και [15] και στην ιστοσελίδα [16]. Πιο συγκεκριμένα, εφαρμογή των υποβρύχιων μεθόδων επικοινωνίας σε ρομποτικά ψάρια περιγράφεται στη διπλωματική [17] και στη δημοσίευση [18].

Τέλος, για τον σχεδιασμό και την κατασκευή της βάσης $3^{ου}$ άξονα, χρησιμοποιήθηκαν πληροφορίες για τον κινητήρα από την ιστοσελίδα [15] και για τις ιδιότητες των υλικών κατασκευής από την ιστοσελίδα [19]. Η έρευνα για τα έδρανα κύλισης και τους ελαστικούς συνδέσμους που υπάρχουν διαθέσιμοι στην αγορά πραγματοποιήθηκε μέσα από τις ιστοσελίδες [20] και [21]. Η μεθοδολογία επιλογής κατάλληλων συναρμογών για έδρανα κύλισης περιγράφεται στο [22].

1.3 Δομή εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από έξι κεφάλαια τα οποία και αναλύονται περιληπτικά στην συνέχεια.

Στο δεύτερο κεφάλαιο αναλύεται η θεωρία για τα λειτουργικά συστήματα που χρησιμοποιούν τα ρομπότ και τις μεθόδους επικοινωνίας των ρομποτικών ψαριών. Μετά από μία σύντομη αναφορά στην ιστορία των λειτουργικών συστημάτων των ρομπότ, το ενδιαφέρον εστιάζεται στο λειτουργικό σύστημα ROS και στα βασικά δομικά στοιχεία του, το οποίο πρόκειται να χρησιμοποιηθεί στην εργασία αυτή. Παρατίθενται επίσης οι βασικές αρχές λειτουργίας της σειριακής επικοινωνίας και του υποσυστήματος UART.

Το τρίτο κεφάλαιο εμπεριέχει την αναλυτική περιγραφή της διαδικασίας σχεδιασμού και υλοποίησης της κεντρικής μονάδας ελέγχου λειτουργιών του ρομποτικού ψαριού, συμπεριλαμβανομένων του υλικού (hardware) που επιλέχθηκε και του λογισμικού (software) που χρησιμοποιήθηκε και επιλέχθηκε. Ειδικότερα, αναφέρονται και επεξηγούνται κάποια βασικά σημεία της λογικής επικοινωνίας μεταξύ των επιμέρους συστημάτων και περιγράφεται πως ο έλεγχος θέσης ανοικτού βρόχου του ρομποτικού ψαριού ενσωματώθηκε στο συνολικό σύστημα. Τα παραπάνω συνοψίζονται και σε μορφή διαγράμματος επικοινωνίας συστήματος ROS.

Στο τέταρτο κεφάλαιο περιγράφεται ο σχεδιασμός και η υλοποίηση της βάσης 3^{ου} άξονα για το φορείο ρυμούλκησης του ρομποτικού ψαριού. Πιο αναλυτικά, παρουσιάζονται οι ελάχιστες προδιαγραφές μηχανικής υλοποίησης της βάσης και αναφέρονται οι παράγοντες που οδήγησαν στις επιλογές αυτές. Περιγράφονται συνοπτικά οι σχεδιαστικές λύσεις που εξετάστηκαν μέχρι την επιλογή της τελικής λύσης προς κατασκευή. Στη συνέχεια περιγράφονται εκτενώς οι μηχανικές συνιστώσες που αποτελούν την τελική διαμόρφωση και οι μεθοδολογίες που ακολουθήθηκαν για την επιλογή τους. Επιλέγονται τα εξαρτήματα που πρόκειται να κατασκευαστούν στο Εργαστήριο Αυτομάτου Ελέγχου και περιγράφεται η διαδικασία κατασκευής τους συμπεριλαμβανομένου υλικού (hardware) και λογισμικού (software) που χρησιμοποιήθηκε. Στο τέλος του κεφαλαίου αυτού παρουσιάζεται η συναρμολόγηση της τελικής κατασκευής και η ενσωμάτωση της στο φορείο ρυμούλκησης.

Στο πέμπτο κεφάλαιο περιγράφεται η πειραματική διαδικασία που ακολουθήθηκε για τη δοκιμή του κεντρικού συστήματος ελέγχου του ρομποτικού ψαριού.

Τέλος, στο έκτο κεφάλαιο περιλαμβάνονται τα συμπεράσματα που προέκυψαν από την υλοποίηση της διπλωματικής εργασίας τα οποία είναι χρήσιμα για οποιαδήποτε μελλοντική χρήση.

2 Θεωρία λειτουργικών συστημάτων για ρομπότ και υποβρύχιων μεθόδων επικοινωνίας

2.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται το θεωρητικό υπόβαθρο και αναλύεται ο τρόπος λειτουργίας του λειτουργικού συστήματος ROS, προκειμένου να μπορέσουν να ενσωματωθούν σε αυτό τα επιμέρους υποσυστήματα του ρομποτικού ψαριού. Επιπρόσθετα, αναφέρονται οι υπάρχουσες μέθοδοι επικοινωνίας που χρησιμοποιούνται στα ρομποτικά ψάρια και ειδικότερα στις βασικές αρχές λειτουργίας της σειριακής επικοινωνίας UART.

2.2 Λειτουργικά συστήματα για ρομπότ και τρόποι επικοινωνίας ρομποτικών ψαριών

Μία μεγάλη πρόκληση στην ανάπτυξη μία κεντρικής μονάδας ελέγχου ρομποτικών εφαρμογών είναι το γεγονός ότι απαιτείται μία διεπιστημονική προσέγγιση. Δεδομένου ότι τα συστήματα αυτά ενσωματώνουν αρκετές διαφορετικές, αλλά διασυνδεδεμένες πτυχές, καθίσταται όλο και πιο σημαντική η απόκτηση επαρκούς εξοικείωσης με τους σχετικούς τομείς. Η επανεξέταση της υπάρχουσας τεχνολογίας, καθώς κι η αναγνώριση των πλεονεκτημάτων αλλά και των ελλείψεων που την χαρακτηρίζουν, διευκολύνει την αποφυγή λαθών του παρελθόντος και διευρύνει την ανάπτυξη νέας τεχνολογίας.

Σε πρώτο στάδιο, γίνεται μία ανασκόπηση στα υπάρχουσα ρομποτικά λειτουργικά συστήματα και σε ρομποτικές εφαρμογές που υλοποιήθηκαν με την χρήση του λειτουργικού συστήματος ROS. Στη συνέχεια, ακολουθεί μία αναφορά στις υπάρχουσες μεθόδους επικοινωνίας που χρησιμοποιούνται σε υποβρύχια ρομποτικά συστήματα.

2.2.1 Ρομποτικά Λειτουργικά Συστήματα

Καθώς τα ρομποτικά συστήματα γίνονται όλο και πιο πολύπλοκα, δημιουργείται η ανάγκη εξειδικευμένου λογισμικού που βοηθά στη διαχείρισή τους. Υλοποιήθηκαν πολλές εφαρμογές, τόσο ιδιωτικής όσο και ανοικτής προέλευσης, που μπορούν να παρέχουν την απαιτούμενη ευελιξία για την ενσωμάτωση διαφόρων εξαρτημάτων όπως αισθητήρες, επενεργητές κ.λπ. για την κατασκευή ολοκληρωμένων ρομποτικών συστημάτων.

Διάφορες προσπάθειες στο πανεπιστήμιο του Στάνφορντ στα μέσα της δεκαετίας του 2000 που αφορούσαν ενσωματωμένο σύστημα τεχνητής νοημοσύνης όπως το STanford AI Robot (STAIR) (βλ. Σχήμα 2-1) [2], και το πρόγραμμα Personal Robots (PR) (βλ. Σχήμα 2-2) [3], [4], δημιούργησαν εσωτερικά πρωτότυπα ευέλικτων, δυναμικών συστημάτων λογισμικού που προορίζονται για ρομποτικές εφαρμογές. Το 2007, η εταιρία Willow Garage παρείχε σημαντικούς πόρους για την περαιτέρω επέκταση αυτών των εννοιών και για την δημιουργία δοκιμασμένων και ελεγμένων υλοποιήσεων [5]. Σήμερα, μερικά από τα πλέον γνωστά συστήματα που ανταποκρίνονται στις σύγχρονες απαιτήσεις είναι το ROCK [6], το [23], το Orocos [7], το GenoM [24] και το ROS [5], [9]. Ωστόσο, τα συστήματα αυτά είναι σχεδιασμένα για να λειτουργούν μέσα σε υπάρχοντα λειτουργικά

συστήματα, όπως μία από της πολλές εκδοχές του συστήματος Linux. Στο παρόν στάδιο, δεν υπάρχουν ρομποτικά λειτουργικά συστήματα σχεδιασμένα ειδικά για ενσωματωμένα συστήματα και να μην εξαρτώνται από υπάρχοντα λειτουργικά συστήματα [25].

Σχήμα 2-1. Το ρομπότ STAIR 1 του πανεπιστημίου του Stanford [2].

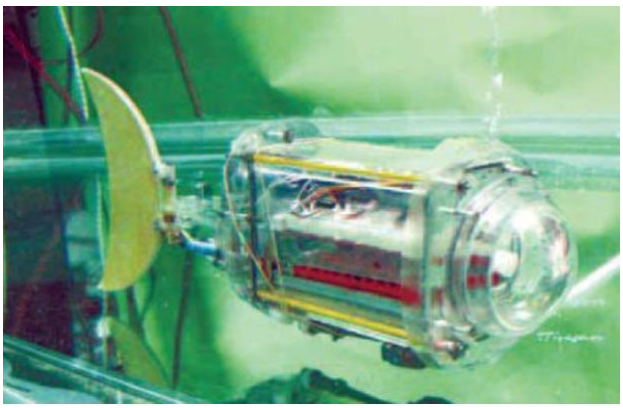
Σχήμα 2-2. Το πρωτότυπο Personal Robot PR-1 του πανεπιστημίου του Stanford [26].

Από το ξεκίνημά του, το λειτουργικό σύστημα ROS αναπτύχθηκε σε πολλαπλά ιδρύματα για πληθώρα ρομποτικών εφαρμογών, συμπεριλαμβανομένων και των ιδρυμάτων που παρέλαβαν το ρομπότ PR2 (βλ. Σχήμα 2-3) [11] από την Willow Garage [13]. Παρόλο που θα ήταν πολύ πιο απλό για όλους τους συνεισφέροντες να μοιράζονταν το λογισμικό τους σε κοινούς διακομιστές, με το

πέρασμα του χρόνου, το “ομοσπονδιακό” μοντέλο εξελίχθηκε σε ένα από τα κυριότερα πλεονεκτήματα του λειτουργικού συστήματος ROS. Για παράδειγμα, οποιαδήποτε ομάδα προγραμματιστών μπορεί να ξεκινήσει το δικό της λογισμικό πακέτο, στους δικούς της διακομιστές, διατηρώντας πλήρη δικαιώματα και τον έλεγχο του πακέτου αυτού. Το σύστημα παρέχει επίσης την δυνατότητα δημοσίευσης των λογισμικών πακέτων από τους ιδιοκτήτες για ελεύθερη χρήση. Αυτό έχει ως αποτέλεσμα την επιβράβευση και αναγνώριση των δημιουργών για τα επιτεύγματά τους, αλλά και την βελτίωση του λογισμικού μέσω ανατροφοδότησης πληροφοριών και σφαλμάτων από τους χρήστες.

Σχήμα 2-3. Το Personal Robot PR-2 της Willow Garage [13].

Το δίκτυο του λειτουργικού συστήματος ROS αποτελείται από δεκάδες χιλιάδες χρήστες ανά το παγκόσμιο και εφαρμόζεται σε τομείς που κυμαίνονται από χόμπι μέχρι μεγάλα βιομηχανικά συστήματα αυτοματισμών [5]. Ο τρόπος λειτουργίας του, σε συνεργασία με την ευρεία διαδικτυακή κοινότητα που διαθέτει, το καθιστούν ιδανική επιλογή για την εφαρμογή του στο ρομποτικό ψάρι που αναπτύσσεται στο Εργαστήριο Αυτομάτου Ελέγχου (βλ. Σχήμα 2-4).



Σχήμα 2-4. Το ρομποτικό ψάρι του Εργαστηρίου Αυτομάτου Ελέγχου.

2.2.2 Υποβρύχιες μέθοδοι επικοινωνίας

Η επικοινωνία μεταξύ υποβρύχιων ρομποτικών συστημάτων αποτελεί ιδιαίτερη πρόκληση για τις ερευνητικές ομάδες του κλάδου αυτού. Η φύση του μέσου μετάδοσης, δηλαδή του νερού, αποκλείει

αρκετές συμβατικές μεθόδους, που χρησιμοποιούνται συνήθως στα επίγεια ή και εναέρια ρομπότ. Για παράδειγμα, τα ραδιοκύματα μεταδίδονται σε πολύ χαμηλές συχνότητες στο νερό και απαιτούν μεγάλες κεραίες και υψηλή μετάδοση ισχύος. Γενικά, οι μέθοδοι επικοινωνίας υποβρύχιων ρομποτικών συστημάτων κατηγοριοποιούνται σε δύο βασικές ομάδες:

- Ενσύρματες μέθοδοι επικοινωνίας
- Ασύρματες μέθοδοι επικοινωνίας

Ενσύρματες μέθοδοι επικοινωνίας

Αποτελούν τις πλέον διαδεδομένες και αξιόπιστες μεθόδους επικοινωνίας στις ρομποτικές εφαρμογές. Ακολουθώντας πρωτόκολλα σειριακής επικοινωνίας, όπως το UART (Universal Asynchronous Receiver Transmitter) εξασφαλίζεται η μεταφορά δεδομένων από και προς την μονάδα ελέγχου χωρίς απώλειας πληροφορίας. Ωστόσο, το εύρος ελευθερίας κινήσεων του ρομπότ, εξαρτάται σε μεγάλο βαθμό από το μήκος του καλωδίου μεταφοράς, γεγονός που αποτελεί εμπόδιο στην πλήρη αυτονόμηση του ρομποτικού συστήματος.

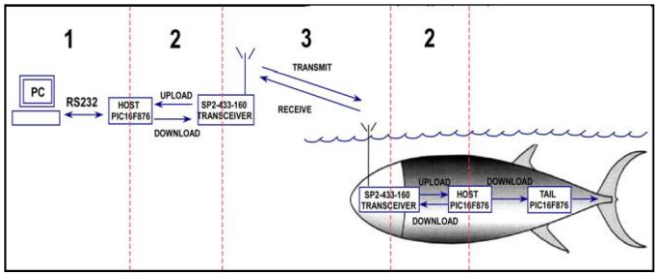
Η πλήρης ενσύρματη επικοινωνία, αποτελείται ουσιαστικά από την απευθείας σύνδεση της κεντρικής μονάδας ελέγχου (κεντρικός ηλεκτρονικός υπολογιστής) με τον εκάστοτε μικροεπεξεργαστή που είναι υπεύθυνος για την επιθυμητή λειτουργία, μέσω φυσικού καλωδίου (RS-232).

Ασύρματες μέθοδοι επικοινωνίας

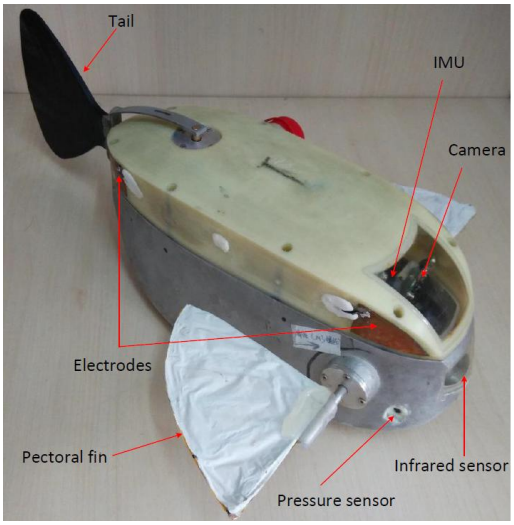
Οι ασύρματες μέθοδοι επικοινωνίας παρουσιάζουν μεγαλύτερη ποικιλία, ανάλογα με το μέσο μετάδοσης που χρησιμοποιούν (αέρας ή νερό). Στην περίπτωση που χρησιμοποιείται το νερό, λαμβάνονται υπόψη οι παράγοντες που επηρεάζουν την υποβρύχια ασύρματη επικοινωνία [14], όπως είναι:

- Η διάδοση σε πολλές κατευθύνσεις (multi-path propagation)
- Το μικρό εύρος συχνοτήτων που είναι επιτρεπτές (Bandwidth)
- Η μεγάλη απόσβεση σήματος (Strong Signal Attenuation)

Ένας συνδυασμός ασύρματης και ενσύρματης επικοινωνίας σε ρομποτικό ψάρι παρουσιάζεται στο [17], όπου χρησιμοποιήθηκαν πομποδέκτες (κεραίες) για επικοινωνία από την επιφάνεια του νερού στην κεντρική μονάδα ελέγχου, με μέσο μετάδοσης τον αέρα (βλ. Σχήμα 2-5). Μία εφαρμογή υποβρύχιας επικοινωνίας σε ρομποτικό ψάρι , με την χρήση ηλεκτρικού ρεύματος και ηλεκτροδίων ως πομποδέκτες, υλοποιήθηκε στο [18], όπου η πληροφορία διαδίδεται μέσω ηλεκτρικών πεδίων στο νερό (βλ. Σχήμα 2-6).

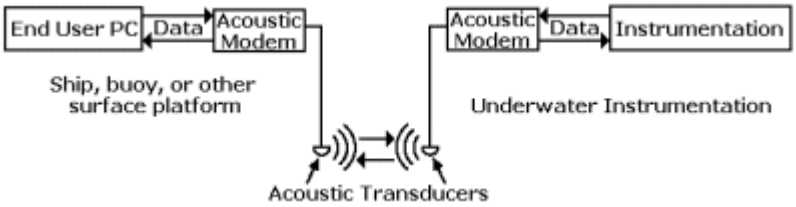


Σχήμα 2-5. Εφαρμογή ασύρματης επικοινωνίας [17].

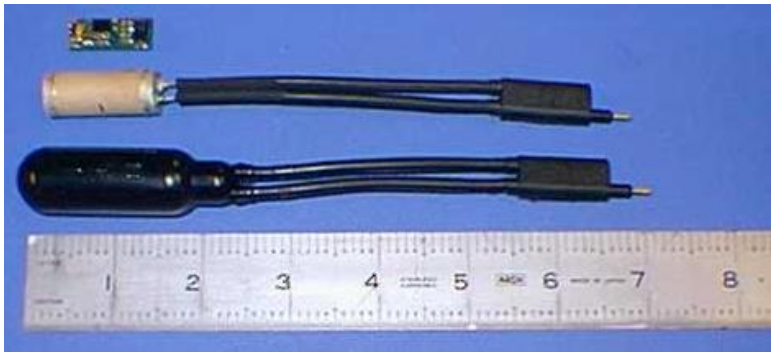


Σχήμα 2-6. Ρομποτικό ψάρι που επικοινωνεί με ηλεκτρόδια [18].

Στον τομέα των υποβρύχιων επικοινωνιών χρησιμοποιούνται ολοένα και περισσότερο τα ακουστικά κύματα. Σε αντίθεση με τα ραδιοκύματα, που απαιτούν μεγάλες κεραίες και υψηλή μετάδοση ισχύος, και τα οπτικά κύματα, που επηρεάζονται από διασκόρπιση (scattering), τα ακουστικά κύματα παρουσιάζουν αξιοσημείωτα χαρακτηριστικά στις υποβρύχιες επικοινωνίες. Μεταδίδονται με την ταχύτητα του ήχου (στο νερό: 1493 m/s) [27], γεγονός που εξασφαλίζει άμεση μετάδοση πληροφορίας. Ο πιο συνήθης τρόπος μετάδοσης είναι με την χρήση υδροφώνων (hydrophones) (βλ. Σχήμα 2-8), που βασίζονται σε έναν πιεζοηλεκτρικό μετατροπέα (transducer), που παράγει ηλεκτρισμό όταν υποβάλλεται σε μεταβολές πίεσης. Το βασικό μοντέλο ακουστικής επικοινωνίας (βλ. Σχήμα 2-7) αποτελείται από ακουστικά μόντεμ, για μετατροπή της ψηφιακής πληροφορίας σε ηχητικά σήματα και αντίστροφα, και από υδρόφωνα (acoustic transducers), για αποστολή και λήψη των ηχητικώς σημάτων αυτών. Μία περιγραφή της κατασκευής ενός ακουστικού μόντεμ γίνεται στο [15].



Σχήμα 2-7. Βασικό μοντέλο ακουστικής επικοινωνίας [16].



Σχήμα 2-8. Τυπικό υδρόφωνο (hydrophone) [16].

Η χρήση ακουστικών κυμάτων ως μέσο ασύρματης υποβρύχιας επικοινωνίας αποτελεί νέο πεδίο έρευνας, με πολλές προοπτικές ανάπτυξης στο μέλλον. Για λόγους απλότητας της κατασκευής, αλλά και επειδή δεν αποτελεί αυτοσκοπό η ασύρματη υποβρύχια επικοινωνία, στην παρούσα διπλωματική εργασία επιλέχθηκε η ενσύρματη σειριακή επικοινωνία (UART – RS-232), που θα περιγραφεί στην συνέχεια.

2.3 Λειτουργικό Σύστημα ROS

Το ROS (Robot Operating System), είναι μία πλατφόρμα ανάπτυξης λογισμικού για ρομπότ. Είναι μία συλλογή από εργαλεία, βιβλιοθήκες και συμβάσεις, με στόχο την απλοποίηση της δημιουργίας σύνθετων και εύρωστων ρομποτικών εφαρμογών [8]. Περιλαμβάνει πολλά χαρακτηριστικά, ειδικά σχεδιασμένα ώστε να διευκολύνουν τη μεγάλης κλίμακας συνεργασία μεταξύ των προγραμματιστικών ομάδων παγκοσμίως. Το σύστημα ROS, δίνει την δυνατότητα σε οποιανδήποτε ομάδα προγραμματιστών να υλοποιήσει το δικό της πακέτο λογισμικού ROS και ταυτόχρονα να διατηρεί τα πλήρη δικαιώματα και τον έλεγχο του. Η πλατφόρμα του ROS μπορεί να ονομαστεί και ως “μετα-λειτουργικό σύστημα” (“meta operating system”), εφόσον πραγματοποιεί πολλές λειτουργίες ενός λειτουργικού συστήματος, αλλά απαιτεί ένα τυπικό λειτουργικό σύστημα υπολογιστών, όπως το Linux [28]. Το ROS ακολουθεί την φιλοσοφία των Unix όσον αφορά την ανάπτυξη λογισμικού σε διάφορες βασικές πτυχές [9], [8], [10]:

- Peer to Peer Δίκτυο

  Τα συστήματα ROS, συνήθως αποτελούνται από πολλά μικρά προγράμματα, τα οποία εκτελούνται σε πολλούς διαφορετικούς κεντρικούς υπολογιστές (hosts), ενώ ανταλλάσουν παράλληλα μηνύματα (messages) σε μία peer to peer φιλοσοφία. Τα μηνύματα αυτά μεταδίδονται κατευθείαν από το ένα πρόγραμμα στο άλλο, χωρίς να μεσολαβεί κεντρική μονάδα ελέγχου (central routing service). Παρόλο που αυτό κάνει την δομή του συστήματος πιο πολύπλοκη, οδηγεί σε ένα σύστημα που ανταποκρίνεται καλύτερα όσο ο όγκος δεδομένων αυξάνεται.
- Αρχιτεκτονική

  Το ROS έχει αναπτυχθεί με έναν microkernel σχεδιασμό, στον οποίο λειτουργίες όπως η περιήγηση στον κώδικα, η γραφική παρουσίαση δεδομένων και η οπτικοποίηση των διασυνδέσεων του συστήματος εκτελούνται από μικρότερα επιμέρους στοιχεία του ROS κι όχι από ένα ολιστικό και ενσωματωμένο περιβάλλον εκτέλεσης. Ο σχεδιασμός αυτός ενθαρρύνει την συνεχή δημιουργία νέων και βελτιωμένων εφαρμογών.
- Συμβατότητα με πολλές γλώσσες προγραμματισμού:

  Ανάλογα με την φύση του κάθε προγραμματιστικού προβλήματος, διαφορετική γλώσσα προγραμματισμού μπορεί να κριθεί ιδανική κάθε φορά. Για τον λόγο αυτό, το ROS έχει σχεδιαστεί ώστε να είναι συμβατό με διάφορες γλώσσες προγραμματισμού. Το γεγονός αυτό, επιτρέπει στα διάφορα στοιχεία (modules) του ROS να αναπτυχθούν σε οποιαδήποτε γλώσσα έχει γραφτεί η εκάστοτε βιβλιοθήκη που θα χρησιμοποιήσουν. Οι πιο συχνά χρησιμοποιούμενες γλώσσες προγραμματισμού στο ROS είναι η C++ και η Python. Στην

παρούσα διπλωματική εργασία η ανάπτυξη λογισμικού πραγματοποιήθηκε με την χρήση της Python.

- Ευελιξία προγραμματισμού

  Το ROS ενθαρρύνει τους προγραμματιστές να δημιουργούν αυτόνομες βιβλιοθήκες, ανεξάρτητες από το ROS. Αυτό απλοποιεί την εξαγωγή του κώδικα και την επαναχρησιμοποίηση βιβλιοθηκών για διαφορετικού είδους ρομπότ. Επιπρόσθετα, οι μικρές αυτόνομες βιβλιοθήκες διευκολύνουν την διαδικασία δοκιμής της εφαρμογής.

- Δωρεάν και ανοικτό λογισμικό

  Ο πλήρης πηγαίος κώδικας του ROS είναι διαθέσιμος δημόσια και κυκλοφόρησε με άδεια BSD, η οποία επιτρέπει εμπορική και μη χρήση. Πολλοί σχεδιαστές λογισμικού επιλέγουν να δημοσιοποιήσουν το έργο τους, γεγονός που επιτρέπει σε άλλες ομάδες προγραμματιστών να χρησιμοποιήσουν και να βελτιώσουν τα εν λόγο πακέτα λογισμικού.

2.3.1 Βασικά Δομικά Στοιχεία Λειτουργικού Συστήματος ROS

Στην ενότητα αυτή περιγράφονται συνοπτικά τα βασικά δομικά στοιχεία ενός συστήματος ROS, συμπεριλαμβανομένων των κόμβων (nodes), των μηνυμάτων (messages), των θεμάτων (topics), των υπηρεσιών (services) και των ενεργειών (actions). Η δομή ενός πακέτου ROS περιγράφεται στην ενότητα 2.3.2.

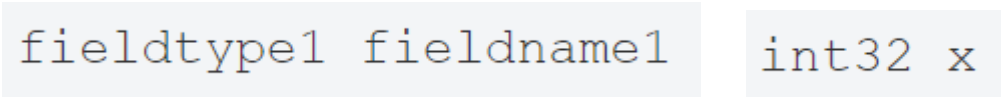
Κόμβοι (Nodes)

Ένας κόμβος ROS είναι ένα μικρό πρόγραμμα, το οποίο εκτελείται μέσα σε ένα δίκτυο ROS. Ο λεγόμενος “ROS-Master” κόμβος (ROS-Master node) επιτρέπει την επικοινωνία μεταξύ όλων των κόμβων ενός δικτύου. Κάθε κόμβος μπορεί να εκτελεί μία διαφορετική λειτουργία. Για παράδειγμα, ένας κόμβος μπορεί να διαβάζει έναν αισθητήρα δύναμης, ενώ παράλληλα ένας άλλος μπορεί να είναι υπεύθυνος για τον έλεγχο ενός κινητήρα. Μερικά πλεονεκτήματα που προκύπτουν από τον καταμερισμό εργασίας μεταξύ κόμβων, είναι η αισθητή μείωση της πολυπλοκότητας του κώδικα και ότι τα σφάλματα περιορίζονται σε μεμονωμένους και ανεξάρτητους κόμβους, γεγονός που διευκολύνει τον εντοπισμό τους. Το ROS χρησιμοποιεί διάφορες μεθόδους επικοινωνίας μεταξύ των κόμβων, όπως τα “θέματα” (topics), τις υπηρεσίες (services) και τις ενέργειες (actions). Πολλές βιβλιοθήκες ROS, όπως είναι η “roscpp” και η “rospy”, επιτρέπουν την δημιουργία προσαρμοσμένων κόμβων σε διάφορες γλώσσες προγραμματισμού, όπως την C++ και την Python αντίστοιχα [29].

Μηνύματα (Messages)

Οι κόμβοι επικοινωνούν μεταξύ τους χρησιμοποιώντας συγκεκριμένα μηνύματα, τα οποία είναι γραμμένα στην ουδέτερη γλώσσα IDL (Interface Definition Language) [9]. Τα μηνύματα ROS μπορεί να είναι απλά, όταν αυτά μεταφέρουν απλά έναν βασικό τύπο δεδομένων (βλ. Σχήμα 2-9-ii), αλλά μπορεί να έχουν πολύπλοκη δομή όταν αποτελούνται από πολλαπλά ένθετα μηνύματα. Η μορφή ενός μηνύματος (message format) είναι μία λίστα από πεδία δεδομένων σε διαφορετικές γραμμές. Κάθε πεδίο αποτελείται από τον τύπο δεδομένων και ένα όνομα, τα οποία διαχωρίζονται μεταξύ τους με ένα κενό (βλ. Σχήμα 2-9-i) [12]. Το μήνυμα “int32” για παράδειγμα, χαρακτηρίζεται από ένα μόνο

πεδίο δεδομένων τύπου “int32” που ονομάζεται “x”. Κοινοί τύποι μηνυμάτων είναι οι integer, floating point, boolean κλπ.



(i) **(ii)**

Σχήμα 2-9. Τύπος μηνύματος ROS: i. Γενική μορφή ενός μηνύματος ROS, ii. Παράδειγμα ενός μηνύματος ROS [12].

Θέματα (Topics)

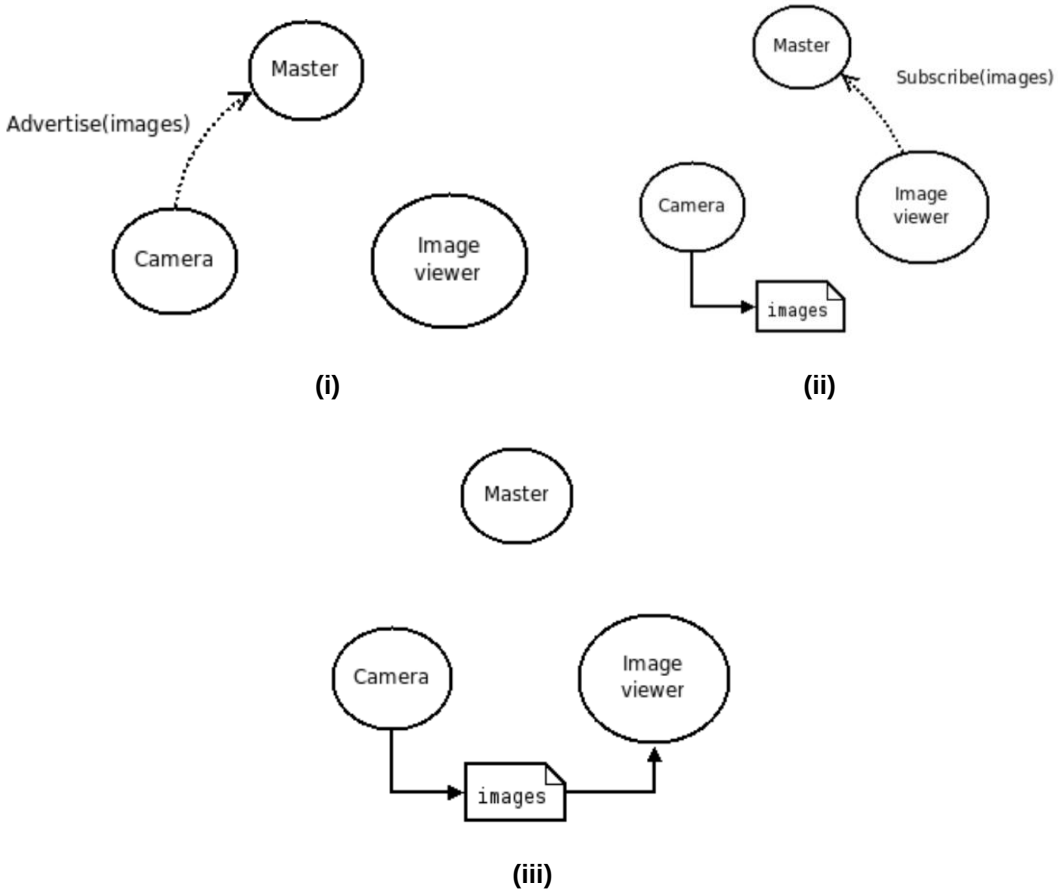
Όπως αναφέρθηκε πιο πάνω, οι κόμβοι μπορούν να στείλουν και να παραλάβουν μηνύματα. Τα λεγόμενα θέματα ROS (ROS topics) είναι ένας τρόπος με τον οποίο οι κόμβοι μπορούν να ανταλλάξουν μηνύματα με δεδομένα μέσα σε ένα δίκτυο ROS. Τα διάφορα θέματα σε ένα δίκτυο, προσδιορίζονται από το μοναδικό τους όνομα και από τον τύπο του μηνύματος το οποίο μεταδίδεται. Ένας κόμβος που παράγει δεδομένα μπορεί να τα δημοσιεύει σε ένα θέμα (publishing), ενώ ταυτόχρονα, άλλοι κόμβοι μπορούν να διαβάσουν τα δεδομένα αυτά, κάνοντας “εγγραφή” (subscribing) στο ίδιο θέμα. Η ανεξαρτησία μεταξύ των κόμβων που παράγουν δεδομένα και των κόμβων που τα λαμβάνουν, επιτρέπει στους κόμβους να είναι εγγεγραμμένοι σε ένα θέμα, ακόμα και αν δεν δημοσιεύονται δεδομένα σε αυτό την παρούσα στιγμή [8]. Σε ένα θέμα πολλοί κόμβοι μπορούν να δημοσιεύουν (publishers), ή να διαβάζουν (subscribers) δεδομένα από αυτό. Ένα πλεονέκτημα της αρχιτεκτονικής αυτής, είναι ότι οι κόμβοι δεν χρειάζεται να γνωρίζουν σε ποιον κόμβο θα στείλουν ή από ποιον κόμβο θα λάβουν δεδομένα. Το μόνο που χρειάζεται, είναι να γνωρίζουν σε ποιο θέμα πρέπει να δημοσιεύσουν, ή από ποιο θέμα πρέπει να διαβάσουν δεδομένα. Ο τύπος ενός θέματος καθορίζεται από τον τύπο του μηνύματος το οποίο δημοσιεύθηκε σε αυτό [5].

Master Κόμβος

Ο κόμβος Master διευθύνει την οργανωμένη λειτουργία ενός δικτύου ROS. Είναι υπεύθυνος για την ονομασία και την εγγραφή υπηρεσιών (services) σε όλους τους άλλους κόμβους του συστήματος και κατευθύνει τους κόμβους που δημοσιεύουν ή διαβάζουν δεδομένα στα θέματα (topics) που χρειάζεται. Ο ρόλος του Master είναι να επιτρέπει σε μεμονωμένους κόμβους ROS να εντοπίζουν ο ένας τον άλλο [5].

Ένα απλοποιημένο παράδειγμα της λειτουργίας του κόμβου Master παρουσιάζεται στο Σχήμα 2-10. Το σύστημα του παραδείγματος αποτελείται από δύο κόμβους, έναν υπεύθυνο για την λήψη μίας φωτογραφίας από μία κάμερα (camera) και έναν που θέλει να διαβάσει την φωτογραφία αυτή (Image viewer) (π.χ. για πιθανή επεξεργασία εικόνας). Μία τυπική αλληλουχία βημάτων θα άρχιζε με τον κόμβο της κάμερας να ειδοποιεί τον κόμβο Master ότι θέλει να δημοσιεύσει την φωτογραφία στο θέμα με το όνομα “images” (βλ. Σχήμα 2-10-i). Μετά την έγκριση του Master, ο κόμβος Camera δημοσιεύει την φωτογραφία στο θέμα “images”. Μέχρι στιγμής κανένας κόμβος δεν έχει κάνει εγγραφή στο θέμα “images”, οπότε κανένα δεδομένο δεν έχει μεταφερθεί μεταξύ των δύο κόμβων. Στη συνέχεια, ο κόμβος Image_viewer ειδοποιεί τον Master ότι θέλει να κάνει εγγραφή στο θέμα “images”, για να δει εάν τυχόν έχει δημοσιευτεί κάποια φωτογραφία (βλ. Σχήμα 2-10-ii). Τώρα που πλέον το θέμα

“images” έχει έναν κόμβο που δημοσιεύει δεδομένα (publisher), αλλά και έναν που λαμβάνει δεδομένα (subscriber), ο Master ενημερώνει τον κάθε κόμβο για την ύπαρξη του άλλου, για να μπορέσει τελικά να ξεκινήσει η μεταφορά της φωτογραφίας μεταξύ τους (βλ. Σχήμα 2-10-iii).



Σχήμα 2-10. Παράδειγμα λειτουργίας κόμβου Master [5]: (i) Δημοσίευση δεδομένων σε ένα θέμα (publishing), (ii) Εγγραφή κόμβου σε ένα θέμα (subscribing), (iii) Μεταφορά δεδομένων μεταξύ των δύο κόμβων.

Υπηρεσίες (Services)

Οι υπηρεσίες αποτελούν μία από τις θεμελιώδης έννοιες της υλοποίησης ενός συστήματος ROS. Σε αντίθεση με τα θέματα ROS, οι υπηρεσίες δεν ακολουθούν την many-to-many φιλοσοφία. Την ώρα που ένας κόμβος (server-node) προσφέρει μία συγκεκριμένη υπηρεσία, ένας άλλος κόμβος (client-node) μπορεί να καλέσει τον πρώτο να εκτελέσει την υπηρεσία ρουτίνας του [9]. Οι κόμβοι client και server, επικοινωνούν μεταξύ τους μέσω συγκεκριμένων μηνυμάτων υπηρεσίας (service messages). Σε αντίθεση με τα συνηθισμένα μηνύματα ROS, τα μηνύματα υπηρεσίας αποτελούνται από δύο μέρη, request και response, τα οποία χωρίζονται μεταξύ τους με μία διακεκομμένη γραμμή (βλ. Σχήμα 2-11). Στο παράδειγμα αυτό, ο κόμβος client ζητά από τον κόμβο server να επιστρέψει το άθροισμα δύο ακέραιων αριθμών. Τα πεδία “int64 A” και “int64 B” αποτελούν το αίτημα (request), ενώ το πεδίο “int64 Sum” αποτελεί την απάντηση (response).

```
int64 A
int64 B
---
int64 Sum
```

Σχήμα 2-11. Παράδειγμα ROS Service [5].

2.3.2 Δομή Πακέτου ROS (ROS Package)

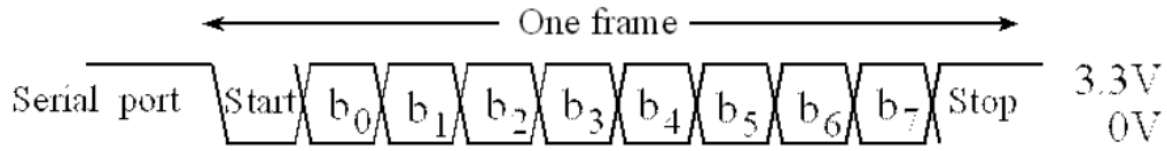
Το λογισμικό του ROS είναι οργανωμένο σε πακέτα (packages), τα οποία τείνουν να ακολουθούν μία συγκεκριμένη δομή. Ένα πακέτο ROS μπορεί να περιέχει κόμβους, μία βιβλιοθήκη, ένα σύνολο δεδομένων, αρχεία διαμόρφωσης (configuration files), ή και λογισμικό το οποίο κατασκευάστηκε εκτός του ROS. Σκοπός των πακέτων ROS είναι να παρέχουν μία χρήσιμη λειτουργία η οποία θα τους επιτρέπει να επαναχρησιμοποιούνται εύκολα. Θα πρέπει να είναι λειτουργικά και ταυτόχρονα ευέλικτα ώστε να μπορούν να χρησιμοποιηθούν και από άλλο λογισμικό. Το πακέτο ROS δημιουργείται μέσα σε έναν ευρύτερο χώρο εργασίας (ROS workspace), ο οποίος αποτελεί την ευρύτερη δομική μονάδα ενός συστήματος ROS. Σε ένα χώρο εργασίας ROS μπορούν να δημιουργηθούν ένα, ή και περισσότερα πακέτα ROS.

2.4 Universal Asynchronous Receiver / Transmitter – UART

Όπως αναφέρθηκε σε προηγούμενη ενότητα, για την υλοποίηση της επικοινωνίας μεταξύ των υποσυστημάτων του ρομποτικού ψαριού χρησιμοποιείται ενσύρματη επικοινωνία. Ένας πολύ εύκολος και αποδοτικός τρόπος αποστολής και λήψης δεδομένων από έναν υπολογιστή σε μία περιφερειακή συσκευή ενσύρματα είναι μέσω σειριακής επικοινωνίας. Από τα διάφορα είδη σειριακής επικοινωνίας (UART, SPI, I2C κλπ), επιλέχθηκε το σύστημα UART (Universal Asynchronous Receiver / Transmitter) αφού είναι ιδιαίτερα απλό και εξυπηρετικό στην χρήση. Το UART είναι ένα υποσύστημα που συναντάται συνήθως στους ηλεκτρονικούς υπολογιστές, σε μικροϋπολογιστές και στις περιφερειακές συσκευές τους συσκευές. Το υποσύστημα αυτό καθιστά δυνατή τη σειριακή ασύγχρονη επικοινωνία, αφού παραλαμβάνει παράλληλα πακέτα δεδομένων και τα αποστέλλει σειριακά σε έναν ακροδέκτη αποστολής (Transmitter) ή, αντίστοιχα, τα παραλαμβάνει σε έναν ακροδέκτη λήψης (Receiver), χωρίς να απαιτείται η ύπαρξη κάποιου κοινού για τις δύο συσκευές ρολογιού συγχρονισμού. Η πλειοψηφία των ηλεκτρονικών υπολογιστών διαθέτει τουλάχιστον μία σειριακή πύλη αυτού του τύπου, ενώ υπάρχει επίσης η δυνατότητα μετατροπής μίας θύρας USB σε σειριακή πύλη μέσω κατάλληλου ηλεκτρονικού μετασχηματιστή. Συστήματα μικροϋπολογιστών, όπως το Raspberry Pi, μπορούν να συνδεθούν με μία άλλη συσκευή σειριακά μέσω υποσυστήματος (module) UART με την χρήση ακροδεκτών (Reciever-Transmitter). Το UART συνήθως χρησιμοποιείται σε συνδυασμό με τα πρωτόκολλα επικοινωνίας RS-232, RS-422, RS-485.

2.4.1 Βασικές Αρχές Λειτουργίας της Σειριακής Επικοινωνίας και του Υποσυστήματος UART

Η σειριακή επικοινωνία περιλαμβάνει την αποστολή δεδομένων (bits), ένα τη φορά, ανά μονάδα χρόνου. Το σύνολο των δεδομένων (bits) που αποστέλλονται ανά μονάδα χρόνου ονομάζεται baud rate. Αν ο αριθμός baud rate είναι έστω x bits/s, τότε ο χρόνος που απαιτείται για να σταλεί ένα bit θα είναι x^{-1} (s). Μία ομάδα από τον ελάχιστο αριθμό δεδομένων (bits) που μπορεί να αποτελέσει μονάδα σειριακής μετάδοσης δεδομένων, λέγεται frame (βλ. Σχήμα 2-12). Το πρώτο bit του frame υποδηλώνει την αρχή των δεδομένων (είναι 0). Στη συνέχεια, ακολουθούν οχτώ bits που αποτελούν το μήνυμα προς αποστολή (b_0 – b_7). Το τελευταίο bit του frame (σε κάποιες περιπτώσεις είναι δύο) υποδηλώνει το τέλος του πακέτου πληροφορίας (είναι 1). Σε παλαιότερες εκδοχές του πρωτοκόλλου αυτού ήταν δυνατή η αποστολή ενός ακόμη bit ασφαλείας (parity bit).



Σχήμα 2-12. Ένα frame μετάδοσης δεδομένων.

Σε προγραμματιστικό επίπεδο, οι παράμετροι αυτοί ορίζονται από τον χρήστη αμέσως μετά το άνοιγμα της σειριακής θύρας. Βασική προϋπόθεση για την αποτελεσματική και ασφαλή μετάδοση σειριακής πληροφορίας μεταξύ δύο συσκευών είναι οι παράμετροι αυτοί να έχουν οριστεί με τον ίδιο τρόπο στις δύο συσκευές κατά το άνοιγμα της σειριακής επικοινωνίας (ίδιο baud rate, ίσος αριθμός stop bits κλπ).

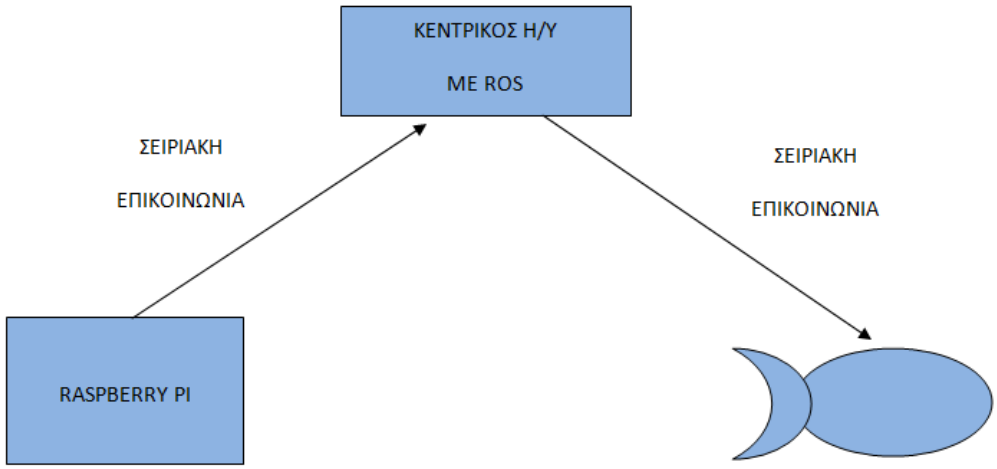
3 Ενσωμάτωση υποσυστημάτων ρομποτικού ψαριού στην κεντρική μονάδα ελέγχου μέσω ROS

3.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται το hardware που χρησιμοποιήθηκε για την ενσωμάτωση των υποσυστημάτων του ρομποτικού ψαριού στην κεντρική μονάδα ελέγχου μέσω του ROS. Περιγράφεται ο τρόπος με τον οποίο προγραμματίστηκαν τα απαραίτητα υποσυστήματα για την επίτευξη της επιθυμητής λειτουργίας της κεντρικής μονάδας ελέγχου. Επίσης παρουσιάζεται η συνδεσμολογία των ηλεκτρονικών στοιχείων με τον κεντρικό υπολογιστή και η τελική διάταξη της κεντρικής μονάδας ελέγχου.

Πιο συγκεκριμένα, στην παρούσα εργασία περιγράφεται η ενσωμάτωση δύο υποσυστημάτων του ρομποτικού ψαριού. Το πρώτο υποσύστημα υλοποιεί τον έλεγχο ανοικτού βρόχου του ρομποτικού ψαριού, που δέχεται ως είσοδο την επιθυμητή τροχιά για το κέντρο μάζας του και επιστρέφει τις παραμέτρους κίνησης του ουραίου πτερυγίου ώστε να επιτευχθεί η τροχιά αυτή. Οι παράμετροι αυτοί αποστέλλονται μέσω σειριακής επικοινωνίας στον μικροελεγκτή του ρομποτικού ψαριού.

Το δεύτερο υποσύστημα υλοποιεί τη λήψη δεδομένων από μία περιφερειακή συσκευή στην κεντρική μονάδα ελέγχου, μέσω σειριακής επικοινωνίας. Ως περιφερειακή συσκευή χρησιμοποιείται ο μικροϋπολογιστής Raspberry Pi, του οποίου η λειτουργία περιγράφεται στη συνέχεια. Η ενσωμάτωση των δύο υποσυστημάτων παρουσιάζεται σε απλοποιημένη μορφή στο Σχήμα 3-1.



Σχήμα 3-1. Διάγραμμα υποσυστημάτων κεντρικής μονάδας ελέγχου.

3.2 Σύντομη περιγραφή υποσυστημάτων τεχνολογικού εξοπλισμού

Στην υποενότητα αυτή παρουσιάζεται όλος ο τεχνολογικός εξοπλισμός που ήταν αναγκαίος για την διεξαγωγή της πειραματικής δοκιμής στο ουραίο πτερύγιο του ρομποτικού ψαριού και γίνεται

αναφορά στις βασικές αρχές λειτουργίας των επί μέρους συστημάτων. Στη συνέχεια παρουσιάζεται η τελική συνδεσμολογία των ηλεκτρονικών στοιχείων με το υπολογιστή της κεντρικής μονάδας ελέγχου.

3.2.1 Μικροελεγκτής PIC-18F4431

Η κίνηση του ουραίου πτερυγίου του ρομποτικού ψαριού υλοποιείται με τη χρήση του μικροελεγκτή PIC-18F4433 8-bit της εταιρίας MICROCHIP. Ο μικροελεγκτής αυτός έχει χαμηλό κόστος και υψηλή υπολογιστική ισχύ. Μπορεί εύκολα να προγραμματιστεί στη γλώσσα C με τη χρήση του ενσωματωμένου περιβάλλοντος ανάπτυξης CCS [30]. Χρησιμοποιεί ταλαντευόμενους κρυστάλλους χαλαζία συχνότητας 40 Mhz [31]. Τα βασικά χαρακτηριστικά του μικροελεγκτή παρουσιάζονται στον Πίνακας 3-1.

Πίνακας 3-1. Χαρακτηριστικά μικροελεγκτή PIC-18F4431.

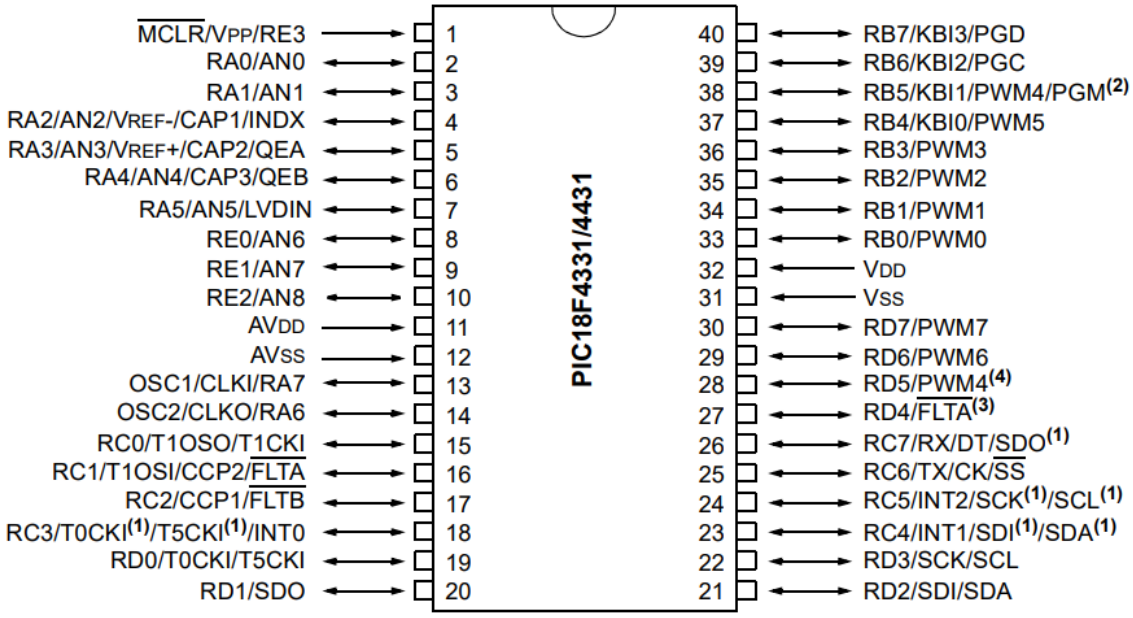
Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	16
CPU Speed (MIPS/DMIPS)	10
SRAM Bytes	768
Data EEPROM/HEF (bytes)	256
Digital Communication Peripherals	1-UART, 1-SPI, 1-I2C, 1-SSP(SPI/I2C)
Capture/Compare/PWM Peripherals	2 CCP
Timers	1 x 8-bit, 3 x 16-bit
ADC Input	9 ch, 10-bit
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	2 to 5.5
Pin Count	40

Ο μικροελεγκτής αυτός, διαθέτει ενσωματωμένο υποσύστημα σειριακής επικοινωνίας EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter). Οι τρεις βασικές λειτουργίες του υποσυστήματος είναι:

- Ασύγχρονη επικοινωνία (Asynchronous Communication): Ενεργοποιείται αυτόματα στη λήψη του πρώτου χαρακτήρα δεδομένων και διαθέτει αυτόματη ρύθμιση του ρυθμού μετάδοσης δεδομένων (baudrate).
- Σύγχρονη επικοινωνία - Master μονής κατεύθυνσης (Synchronous – Master, Half-duplex): Διαθέτει επιλέξιμη πολικότητα ρολογιού (clock polarity).
- Σύγχρονη επικοινωνία – Slave μονής κατεύθυνσης (Synchronous – Slave, Half-duplex): Διαθέτει επιλέξιμη πολικότητα ρολογιού (clock polarity)

και μπορούν να καθοριστούν με την κατάλληλη ρύθμιση τριών καταχωρητών (TXSTA, RCSTA, BAUDCON) [32].

Στην παρούσα εργασία, για την λήψη δεδομένων από την κεντρική μονάδα ελέγχου, χρησιμοποιείται η λειτουργία ασύγχρονης επικοινωνίας. Η σύνδεση με τον κεντρικό υπολογιστή υλοποιείται μέσω των ακροδεκτών 25, 26 και 32 του μικροελεγκτή (βλ. Σχήμα 3-2) και περιγράφεται αναλυτικά σε ακόλουθη ενότητα.



Σχήμα 3-2. Διάγραμμα ακροδεκτών μικροελεγκτή PIC-18F4431.

3.2.2 Μετατροπέας USB – RS-232 (USB to RS-232 Converter)

Η αποστολή και λήψη δεδομένων από και προς την κεντρική μονάδα ελέγχου γίνεται με τη χρήση του σειριακού πρωτοκόλλου επικοινωνίας RS-232. Λόγω περιορισμένου αριθμού σειριακών θυρών στον υπολογιστή της κεντρικής μονάδας ελέγχου, είναι απαραίτητη η αξιοποίηση κάποιων θυρών USB (Universal Serial Bus). Το πρωτόκολλο επικοινωνίας USB, παρόλο που είναι και αυτό σειριακό, παρουσιάζει αρκετές διαφορές σε σχέση με το πρωτόκολλο RS-232 (βλ. Πίνακας 3-2). Οι διαφορές αυτές καθιστούν αναγκαία την χρήση μετατροπέα USB – RS-232 (βλ. Σχήμα 3-3-i).

Πίνακας 3-2. Πρωτόκολλα επικοινωνίας USB και RS-232.

Χαρακτηριστικά	USB	RS-232
Εύρος	Μεγάλο	Μικρό (μερικά μέτρα)
Ταχύτητα	12 Mbps (αρχικά) – 5 Gbsps (USB3)	9600 bps – 100 kbps
Ακροδέκτες	4	9
Τάση	± 5 V	± 12 V
Είδος επικοινωνίας	Διπλής κατεύθυνσης 1 : 1	Μονής κατεύθυνσης 1 : 1 Ασύγχρονη
Έλεγχος δεδομένων	Με ενσωματωμένο λογισμικό	Με σήματα ελέγχου

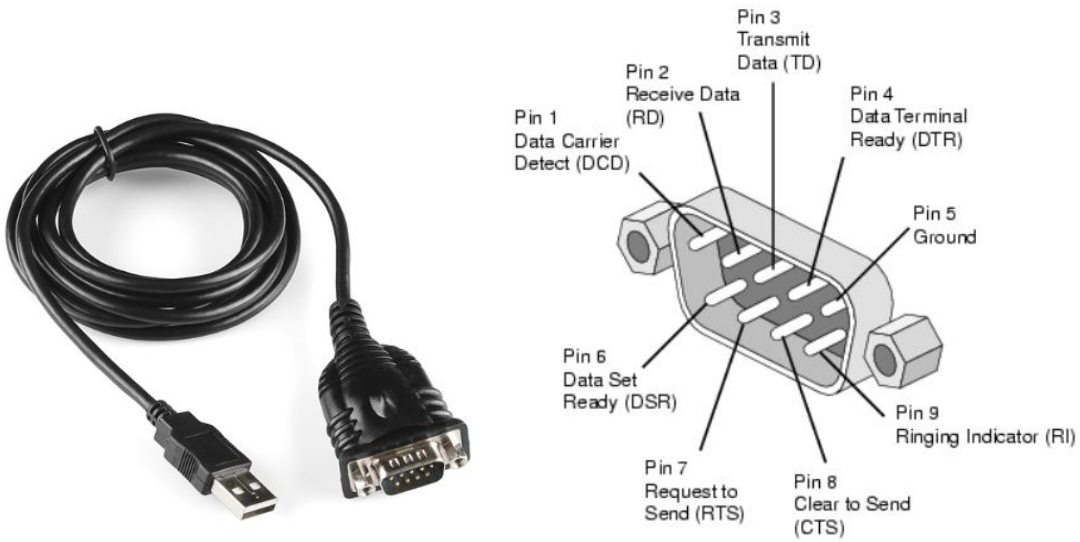
Ο μετατροπέας αυτός ανήκει στην κατηγορία μετατροπέων πρωτοκόλλου και είναι υπεύθυνος για την μετατροπή σημάτων δεδομένων USB σε κοινά σήματα σειριακής θύρας πρωτοκόλλου RS-232 και αντίστροφα. Τέτοιου είδους μετατροπείς χρησιμοποιούνται συνήθως σε προσωπικές, εμπορικές και βιομηχανικές εφαρμογές για να παρέχουν σε ηλεκτρονικούς υπολογιστές με θύρες USB τη δυνατότητα πρόσβασης και επικοινωνίας με σειριακές συσκευές.

Τρόπος λειτουργίας

Ένας τυπικός μετατροπέας USB – RS-232 αποτελείται από ένα τσιπ επεξεργαστή, το οποίο επεξεργάζεται σήματα δεδομένων USB. Ο επεξεργαστής αυτός στέλνει τα επεξεργασμένα σήματα USB σε ένα σειριακό τσιπ το οποίο εφαρμόζει την απαραίτητη τάση και στέλνει τα σήματα αυτά σε μία σειριακή θύρα εξόδου.

Για να μπορεί ο ηλεκτρονικός υπολογιστής να εντοπίσει και να επεξεργαστεί τα σήματα που δέχεται από την θύρα USB, είναι απαραίτητη πρώτα η εγκατάσταση των προγραμμάτων οδήγησης του μετατροπέα, όπως αυτή περιγράφεται σε ακόλουθη ενότητα.

Όταν ο μετατροπέας συνδέεται στον ηλεκτρονικό υπολογιστή μέσω της θύρας USB, το πρόγραμμα οδήγησης δημιουργεί μία εικονική θύρα COM (Communication port), η οποία μπορεί να προσπελαστεί και να χρησιμοποιηθεί σαν μία ενσωματωμένη σειριακή θύρα COM (βλ Σχήμα 3-3-ii). Ωστόσο, τα χαρακτηριστικά της εικονικής θύρας δεν είναι ακριβώς τα ίδια με αυτά της ενσωματωμένης, κυρίως λόγω λανθάνουσας κατανομής δεδομένων (data latency).Το γεγονός αυτό καθιστά τον μετατροπέα USB – RS-232 αναξιόπιστη και μη επιθυμητή λύση, όταν απαιτείται ευαίσθητη και ακριβής μεταφορά δεδομένων. Λόγω της σχετικά μικρής ταχύτητας επικοινωνίας που απαιτείται μεταξύ ρομποτικού ψαριού και κεντρικής μονάδας ελέγχου, η χρήση του μετατροπέα αυτού κρίνεται επαρκής για την εφαρμογή αυτή.



Σχήμα 3-3. (i) Μετατροπέας USB – RS-232, (ii) Διάγραμμα ακροδεκτών RS-232 [33]**.**

Διαδικασια εγκατάστασης προγράμματος οδήγησης

Πριν την χρήση του μετατροπέα στην διασύνδεση της κεντρικής μονάδας ελέγχου με το ρομποτικό ψάρι, είναι απαραίτητη η εγκατάσταση του προγράμματος οδήγησης του (driver) στον κεντρικό υπολογιστή.

Σε πρώτο στάδιο, λόγω της μεγάλης ποικιλίας μετατροπέων που είναι διαθέσιμοι στην αγορά, είναι προτιμότερο το κατέβασμα του προγράμματος οδήγησης του αντίστοιχου μετατροπέα που είναι συμβατό με το λογισμικό Linux, εφόσον αυτό υπάρχει, από την ιστοσελίδα του αντίστοιχου κατασκευαστή. Στη συνέχεια, ο μετατροπέας συνδέεται σε μία από τις θύρες USB του κεντρικού υπολογιστή. Μετά την πάροδο ενός λεπτού, σε ένα τερματικό (terminal) εκτελείται η εντολή:

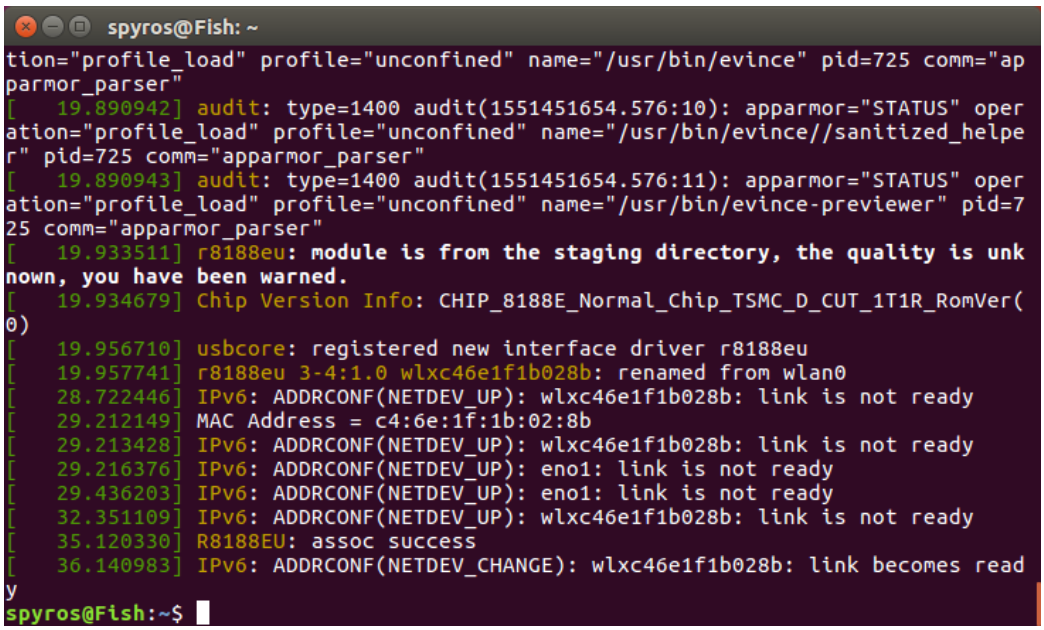
```
dmesg
```

η οποία εμφανίζει μία λίστα μηνυμάτων. Το αποτέλεσμα της εκτέλεσης έχει την μορφή που παρουσιάζεται στο Σχήμα 3-4. Ακολούθως ο μετατροπέας αποσυνδέεται από τον κεντρικό υπολογιστή και στο τερματικό εκτελείται η εντολή:

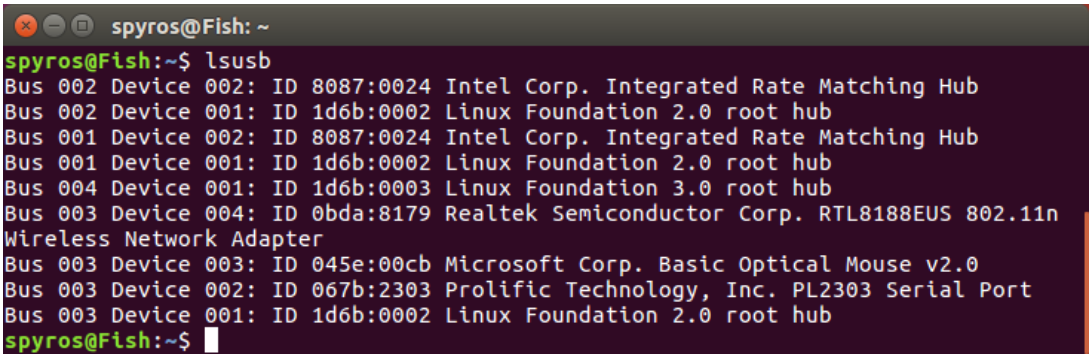
```
lsusb
```

η οποία εμφανίζει μία λίστα με τις συνδεδεμένες θύρες USB που υπάρχουν στον υπολογιστή, όπως αυτή που παρουσιάζεται στο Σχήμα 3-5. Ο μετατροπέας ξανασυνδέεται στον κεντρικό υπολογιστή και επαναλαμβάνεται η εκτέλεση της εντολής “lsusb”. Στο τερματικό εμφανίζεται μία λίστα παρόμοια με την προηγούμενη, έχοντας μία επιπρόσθετη γραμμή (βλ. Σχήμα 3-6). Στην περίπτωση του μετατροπέα που χρησιμοποιήθηκε στην παρούσα εφαρμογή, αυτή είναι:

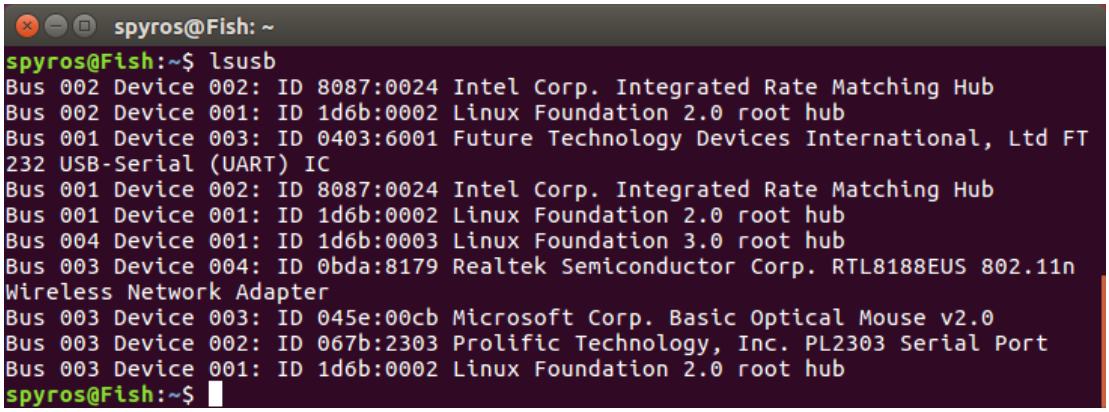
Bus 001 Device 003: ID 0403:6001 Future Technology Device International, Ltd FT232 USB-Serial (UART) IC



Σχήμα 3-4. Αποτέλεσμα εκτέλεσης εντολής dmesg.



Σχήμα 3-5. Αποτέλεσμα εκτέλεσης εντολής lsusb.



Σχήμα 3-6. Αποτέλεσμα εκτέλεσης εντολής lsusb, μετά τη σύνδεση του μετατροπέα.

Από την νέα γραμμή στην λίστα γίνονται γνωστοί οι παράμετροι “Ventor ID” και “Product ID” του μετατροπέα, που είναι ίσοι με τις τιμές 0403 και 6001 αντίστοιχα. Οι παράμετροι αυτοί εισάγονται στο σύστημα του κεντρικού υπολογιστή με την εκτέλεση της εντολής:

```
sudo modprobe usbserial ventor=0x0403 product=0x6001
```

Στη συνέχεια, εκτελείται ξανά η εντολή “dmesg”, αλλά αυτήν τη φορά εμφανίζεται στην λίστα η θύρα στην οποία έχει συνδεθεί ο μετατροπέας (βλ. Σχήμα 3-7).

Σχήμα 3-7. Αποτέλεσμα εντολής dmesg, μετά την εγκατάσταση του μετατροπέα.

Με την εντολή αυτή επιβεβαιώνεται η εγκατάσταση του μετατροπέα στον υπολογιστή και γίνεται γνωστή η τοποθεσία της σειριακής θύρας, στην οποία έχει αναγνωρίσει ο υπολογιστής τον μετατροπέα. Η τοποθεσία αυτή θα χρησιμοποιηθεί αργότερα, για την επίτευξη της σειριακής επικοινωνίας μεταξύ κεντρικού υπολογιστή και ρομποτικού ψαριού, κατά την εκτέλεση της πειραματικής διαδικασίας. Στην συγκεκριμένη περίπτωση, η τοποθεσία αυτή είναι:

```
/dev/ttyUSB1
```

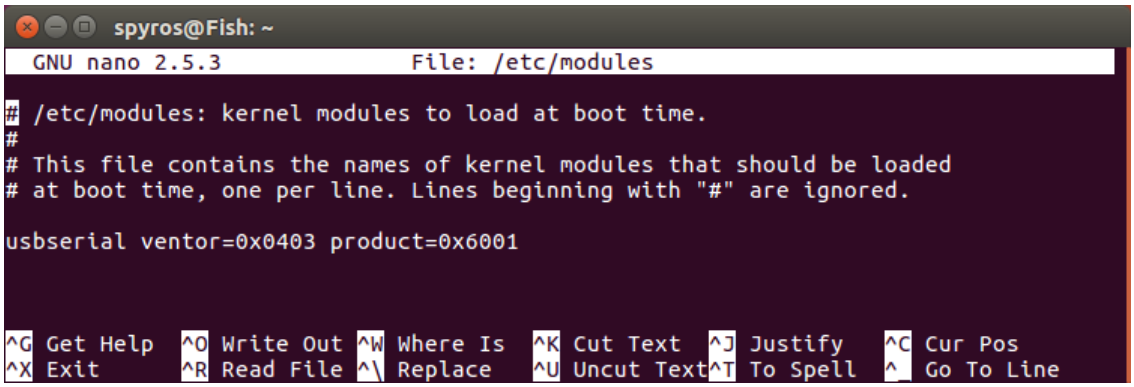
Για να αποφευχθεί η επανάληψη της πιο πάνω διαδικασίας κάθε φορά που συνδέεται ο μετατροπέας στον υπολογιστή σε διαφορετική θύρα USB, οι παράμετροι “Ventor ID” και “Product ID” πρέπει να καταχωρηθούν στο αρχείο “modules” του συστήματος. Το αρχείο ανοίγει σε επεξεργάσιμη μορφή με τη χρήση της εντολής:

```
sudo nano -w /etc/modules
```

Στο τέλος του αρχείου προστίθενται οι παράμετροι με την μορφή της ακόλουθης γραμμής κειμένου (βλ. Σχήμα 3-8):

```
usbserial ventor=0x0403 product=0x6001
```

Το αρχείο αποθηκεύεται με το αρχικό του όνομα και η εγκατάσταση του προγράμματος οδήγησης του μετατροπέα έχει πλέον ολοκληρωθεί.



Σχήμα 3-8. Επεξεργασία αρχείου “modules”.

3.2.3 Raspberry Pi

Σήμερα, οι μικρές αναπτυξιακές πλατφόρμες μικροϋπολογιστών γίνονται ολοένα και περισσότερο διαδεδομένες στο ευρύτερο κοινό. Η υψηλή επεξεργαστική τους ισχύς, το ιδιαίτερα μικρό μέγεθος τους, το χαμηλό κόστος τους, η χαμηλή κατανάλωση ισχύος, η υψηλή προσαρμοστικότητα τους και η δυνατότητα διεπαφής με εξωτερικές συσκευές, αισθητήρες και περιφερειακά τις καθιστά ιδιαίτερα χρήσιμες σε τομείς όπως η ρομποτική, η μηχανοτρονική, το λεγόμενο Internet of Things κ.α. Τέτοιες πλατφόρμες μικροϋπολογιστών είναι για παράδειγμα τα Arduino, Mega2560, Beaglebone και το Raspberry Pi.

Οι πλατφόρμες αυτές, που συχνά διαθέτουν υποστήριξη για λειτουργικό σύστημα (συνήθως κάποια διανομή των Linux), προσφέρουν τα πλεονεκτήματα ενός υπολογιστή και δίνουν στον χρήστη τη δυνατότητα πρόσβασης και χρήσης πληθώρας περιφερειακών συστημάτων, που επιτρέπουν την διεπαφή με συσκευές και αισθητήρες. Μερικά παραδείγματα περιφερειακών υποσυστημάτων, με τα οποία είναι εφοδιασμένοι οι μικροϋπολογιστές, είναι τα:

- **General Purpose Input/Output (GPIO):** Οι γενικής χρήσης ακροδέκτες εισόδου/εξόδου παρέχουν τη δυνατότητα επικοινωνίας με εξωτερικές συσκευές μέσα από κανάλια μεταφοράς δεδομένων ανά bit. Αποτελεί τον πιο εύκολο και άμεσο τρόπο επικοινωνίας και ελέγχου εξωτερικών συσκευών.
- **Universal Asynchronous Receiver/Transmitter (UART):** Το υποσύστημα αυτό παρέχει τη δυνατότητα ασύγχρονης σειριακής επικοινωνίας. Η ταχύτητα και η μορφή των δεδομένων μπορούν να καθοριστούν από τον χρήστη. Η χρήση μόνο τριών καλωδίων αμφίδρομης επικοινωνίας το καθιστά τον πιο απλό, αξιόπιστο και εύκολο τρόπο επικοινωνίας.
- **Serial Peripheral Interface (SPI):** Το υποσύστημα αυτό παρέχει τη δυνατότητα σύγχρονης σειριακής επικοινωνίας. Εφαρμόζεται κυρίως σε επικοινωνίες μεταξύ συσκευών σε μικρή απόσταση.
- **Inter-Integrated Circuit (I2C):** Χρησιμοποιείται επίσης ως μέθοδος σύγχρονης σειριακής επικοινωνίας. Προτιμάται σε εφαρμογές που απαιτείται ευκολία χρήσης και χαμηλό κόστος, αλλά χαμηλή ταχύτητα επικοινωνίας. Επιπρόσθετα, παρέχει δυνατότητα για έλεγχο της επικοινωνίας πολλαπλών συσκευών σε μορφή δικτύου με μόνο 2 ακροδέκτες γενικής χρήσης, σε αντίθεση με το SPI.
- **Pulse Width Modulation (PWM):** Το υποσύστημα αυτό παρέχει τη δυνατότητα παραγωγής ελεγχόμενου σήματος στη μορφή παλμών μεταβλητού πλάτους. Με τον τρόπο αυτό παράγεται ψηφιακό σήμα που μπορεί να χρησιμοποιηθεί ως ψευδοαναλογικό σε πολλές εφαρμογές.

Στην παρούσα διπλωματική εργασία, ως περιφερειακή συσκευή του δεύτερου υποσυστήματος, επιλέχθηκε το Raspberry Pi 2 Model B (βλ Σχήμα 3-9), της εταιρίας Raspberry Pi Foundation. Ο μικροϋπολογιστής αυτός χρησιμοποιείται ως περιφερειακή συσκευή της κεντρικής μονάδας ελέγχου του ρομποτικού ψαριού, η οποία μπορεί να παρέχει δεδομένα από επιπλέον περιφερειακές συσκευές που είναι συνδεδεμένες σε αυτή (όπως ψηφιακή κάμερα). Το Raspberry Pi 2 Model B βασίζεται σε ένα ισχυρό System-On-a-Chip (SoC) και συγκεκριμένα το BCM2837 που ενσωματώνει τον επεξεργαστή Cortex-A7 της εταιρίας ARM.

Σχήμα 3-9. Ο μικροϋπολογιστής Raspberry Pi 2 Model B [34]**.**

Μερικά από τα βασικά χαρακτηριστικά του, συνοψίζονται στον Πίνακας 3-3 και καταδεικνύουν γιατί είναι ιδιαίτερα χρήσιμο σε ρομποτικές εφαρμογές. Στην παρούσα εργασία το υποσύστημα που χρησιμοποιήθηκε ήταν το:

- **UART:** Με το περιφερειακό αυτό επιτυγχάνεται η επικοινωνία με εξωτερικές συσκευές μέσω του σειριακού πρωτοκόλλου επικοινωνίας RS-232 ή RS-485.

Πίνακας 3-3. Συνοπτικός πίνακας των χαρακτηριστικών του Raspberry Pi.

Processing	Broadcom BCM2837 Arm7 Quad Core Processor powered Single Board Computer running at 900 MHz
Memory	1 GB RAM
	Micro SD port
Input-Output	40pin extended GPIO
	4 x USB 2 ports
	4 pole Stereo output and Composite video port
	Full size HDMI
	CSI camera port for connecting the Raspberry Pi camera
	DSI display port for connecting the Raspberry Pi touch screen display
	Combined 3.5mm audio jack
	UART, SPI, I2C, Ethernet communication
Power supply	Micro USB power source

3.2.4 Σειριακός Μετασχηματιστής RS-232 / TTL (3 – 5.5 V)

Η μεταφορά δεδομένων από το Raspberry Pi στον υπολογιστή της κεντρικής μονάδας ελέγχου γίνεται με τη χρήση του σειριακού πρωτοκόλλου RS-232. Πιο συγκεκριμένα, το ενσωματωμένο υποσύστημα UART του Raspberry Pi συνδέεται με το καλώδιο σειριακής θύρας RS-232, το οποίο είναι συνδεδεμένο στον κεντρικό υπολογιστή. Οι δύο αυτές πλευρές (UART και RS-232 port), παρόλο που είναι συμβατές μεταξύ τους ως προς το λογισμικό, δεν μπορούν να συνδεθούν κατευθείαν με φυσικό καλώδιο, διότι οι διασυνδέσεις υλικού (hardware) δεν είναι συμβατές.

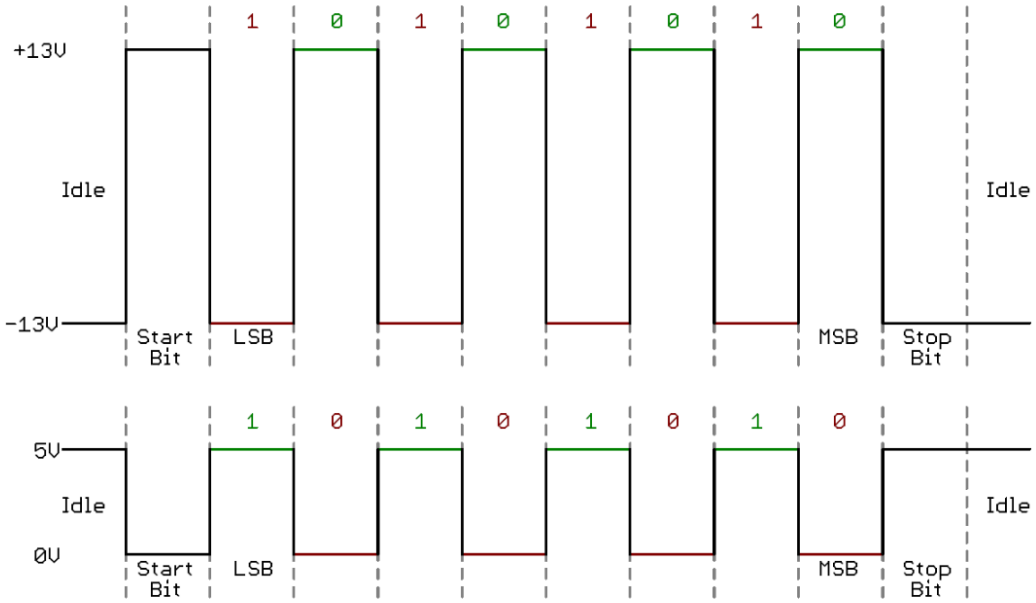
Τα ενσωματωμένα υποσυστήματα UART (που χρησιμοποιούνται στους περισσότερους μικροεπεξεργαστές), παραλαμβάνουν και στέλνουν δεδομένα σειριακά, μεταφέροντας ένα bit δεδομένων τη φορά, σε συγκεκριμένο ρυθμό μετάδοσης (π.χ. 9600 bps, 115200 bps, κλπ.). Αυτή η σειριακή μέθοδος επικοινωνίας, συχνά αναφέρεται ως TTL serial (Transistor-Transistor Logic). Η επικοινωνία σε TTL επίπεδο, κυμαίνεται μεταξύ τάσης 0 V και Vcc, η οποία συνήθως είναι 5 ή 3,3 V. Το λογικό “high” (“1”) αντιπροσωπεύεται από την τάση Vcc, ενώ το λογικό “low” (“0”) από την τάση των 0 V (βλ Σχήμα 3-10-κάτω). Στην περίπτωση του Raspberry Pi, η τάση Vcc είναι ίση με 3,3 V.

Σε αντίθεση με την επικοινωνία TTL, κατά το πρωτόκολλο RS-232, το λογικό “high” (“1”) αντιπροσωπεύεται από αρνητική τάση (οπουδήποτε μεταξύ -3 και -25 V), ενώ το λογικό “low” (“0”) μεταδίδει θετική τάση (οπουδήποτε μεταξύ +3 και +25 V). Στην πλειοψηφία των υπολογιστών αυτά τα σήματα κυμαίνονται μεταξύ -13 και +13 V (βλ Σχήμα 3-10-πάνω).

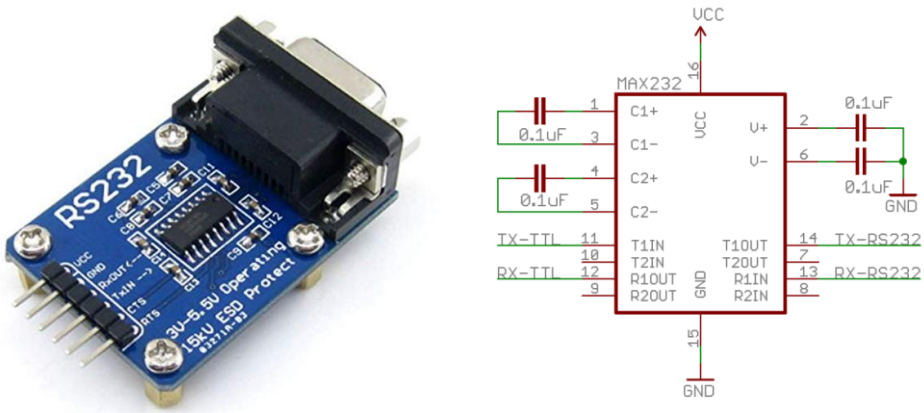
Το μεγαλύτερο εύρος τάσεων ενός σήματος RS-232, το καθιστά λιγότερο ευαίσθητο σε θόρυβο, παρεμβολές και εξασθένηση. Αυτό σημαίνει ότι ένα σήμα RS-232 μπορεί να ταξιδέψει σε μεγαλύτερες

φυσικές αποστάσεις από ότι το αντίστοιχο σήμα TTL, ενώ εξακολουθεί να παρέχει αξιόπιστη μεταφορά δεδομένων.

Λαμβάνοντας υπόψη τα πιο πάνω, γίνεται κατανοητό ότι για την σύνδεση των δύο αυτών θυρών (TTL και RS-232) είναι απαραίτητη όχι μόνο η αντιστροφή των σημάτων (ώστε να συμφωνούν τα λογικά “1” και ”0”), αλλά και η ρύθμιση του εύρους τάσης του σήματος RS-232, ώστε να φτάσει σε επίπεδα μη βλαβερά για τους ακροδέκτες του Raspberry Pi. Η πιο συνήθης και εύκολη λύση για το πρόβλημα αυτό είναι η σύνδεση ενός μικροτσίπ MAX-232 (βλ. Σχήμα 3-11-ii) μεταξύ των δύο θυρών, το οποίο είναι υπεύθυνο για την αντιστροφή των σημάτων και την προσαρμογή του εύρους τάσης. Στην παρούσα εργασία χρησιμοποιείται ο μετατροπέας RS-232 – TTL (βλ. Σχήμα 3-11-i), ο οποίος έχει ενσωματωμένο το μικροτσίπ MAX-232



Σχήμα 3-10. Σύγκριση σήματος TTL (κάτω) και RS-232 (πάνω) [35]**.**



(i) **(ii)**

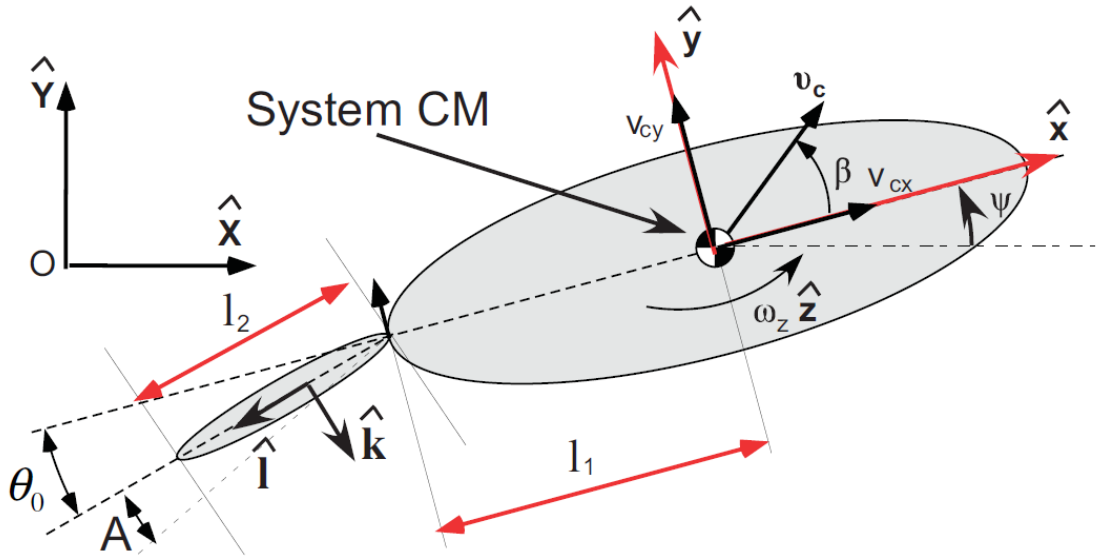
Σχήμα 3-11. (i) Μετατροπέας RS-232 – TTL [36], (ii) Διάγραμμα ακροδεκτών μικροτσίπ MAX-232 [35].

3.3 Ανάπτυξη Λογισμικού Κεντρικής Μονάδας Ελέγχου με ROS

Η κεντρική μονάδα ελέγχου, όπως αναφέρθηκε και προηγουμένως, συγκεντρώνει και συντονίζει όλα τα επιμέρους υποσυστήματα του ρομποτικού ψαριού σε έναν κεντρικό υπολογιστή με τη χρήση του λειτουργικού συστήματος ROS. Πιο συγκεκριμένα, στην παρούσα διπλωματική εργασία ενσωματώνονται δύο κύρια υποσυστήματα, τα οποία περιγράφονται αναλυτικά στη συνέχεια.

3.3.1 Υποσύστημα ελέγχου ανοικτού βρόχου

Το πρώτο υποσύστημα περιλαμβάνει την ενσωμάτωση του ελέγχου ανοικτού βρόχου του ρομποτικού ψαριού. Υλοποιεί την επίλυση του αντίστροφου δυναμικού προβλήματος, όπως αυτό παρουσιάζεται στο [1] και περιγράφεται από το σύστημα εξισώσεων 3-1 – 3-11. Το σύστημα αυτό, δέχεται ως είσοδο την επιθυμητή τροχιά του κέντρου μάζας του ρομποτικού ψαριού (βλ. Σχήμα 3-12). Κατά τη δυναμική ανάλυση του συστήματος, γίνεται η παραδοχή ότι το συνολικό κέντρο μάζας του συστήματος είναι το κέντρο μάζας του κυρίως σώματος του ρομποτικού ψαριού (αγνοείται η μάζα του ουραίου πτερυγίου) [1]. Η παραδοχή αυτή σε συνεργασία με το γεγονός ότι το σύστημα είναι μη ολόνομο, περιορίζουν το πλήθος των τροχιών που μπορούν να επιτευχθούν από το ρομποτικό ψάρι. Οι περιορισμοί αυτοί, συνοψίζονται στην εξίσωση 3-11. Εάν η επιθυμητή τροχιά ικανοποιεί την συνθήκη αυτή, τότε το σύστημα δίνει ως έξοδο τον τρόπο με τον οποίο πρέπει να κινηθεί το ουραίο πτερύγιο, ούτως ώστε το κέντρο μάζας του ρομποτικού ψαριού να εκτελέσει την τροχιά αυτή.



Σχήμα 3-12. Διάγραμμα δυναμικής ανάλυσης ρομποτικού ψαριού [1].

Η επιθυμητή τροχιά εισόδου, που αποτελεί την είσοδο του συστήματος, εκφράζεται από την ταχύτητα ($\dot{X}, \dot{Y}$) και την επιτάχυνση ($\ddot{X}, \ddot{Y}$) του κέντρου μάζας του συστήματος στο αδρανειακό σύστημα συντεταγμένων $O(\hat{X}, \hat{Y})$. Η κίνηση του ουραίου πτερυγίου εκφράζεται από την γωνία που σχηματίζει το πτερύγιο (θ) ως προς το σωματόδετο σύστημα συντεταγμένων ($\hat{x}, \hat{y}$) (Εξίσωση 3-10). Οι τιμές των σταθερών που χρησιμοποιούνται στις εξισώσεις του συστήματος παρουσιάζονται αναλυτικά στο [1].

$$v_{cx} = \dot{X}\cos(\psi) + \dot{Y}\sin(\psi) \tag{3-1}$$

$$v_{cy} = \dot{Y}\cos(\psi) - \dot{X}\sin(\psi) \tag{3-2}$$

$$\dot{\psi} = \omega_z \tag{3-3}$$

$$\dot{v}_{cx} = \ddot{X}\cos(\psi) - \dot{X}\sin(\psi)\dot{\psi} + \ddot{Y}\sin(\psi) + \dot{Y}\cos(\psi)\dot{\psi} \tag{3-4}$$

$$\dot{v}_{cy} = \ddot{Y}\cos(\psi) - \dot{Y}\sin(\psi)\dot{\psi} - \ddot{X}\sin(\psi) - \dot{X}\cos(\psi)\dot{\psi} \tag{3-5}$$

$$g_1 = -\frac{m_1}{c_3}\dot{v}_{cx} + \frac{m_2}{c_3}v_{cy}\omega_z - \frac{c_1}{c_3}v_{cx}\|\upsilon_c\| + \frac{c_2}{c_3}v_{cy}\|\upsilon_c\|\beta \tag{3-6}$$

$$g_2 = \frac{m_2}{c_3}\dot{v}_{cy} + \frac{m_1}{c_3}v_{cx}\omega_z + \frac{c_1}{c_3}v_{cy}\|\upsilon_c\| + \frac{c_2}{c_3}v_{cx}\|\upsilon_c\|\beta \tag{3-7}$$

$$\mathrm{g}_3 = -\dot{\omega}_z + \frac{m_1 - m_2}{J}v_{cx}v_{cy} - c_4\omega_z|\omega_z| \tag{3-8}$$

$$\dot{\omega}_z = \frac{m_1 - m_2}{J}v_{cx}v_{cy} - c_4\omega_z|\omega_z| - c_5 g_2 - \mathrm{sgn}(\ddot{\theta})\,\mathrm{c}_6\sqrt{g_1^2 + g_2^2} \tag{3-9}$$

$$\theta = \arctan\left(\frac{g_1}{g_2}\right) \tag{3-10}$$

$$\ddot{\theta} = \frac{g_3 - g_2 c_5}{c_6} = \pm\sqrt{g_1^2 + g_2^2} \tag{3-11}$$

όπου:

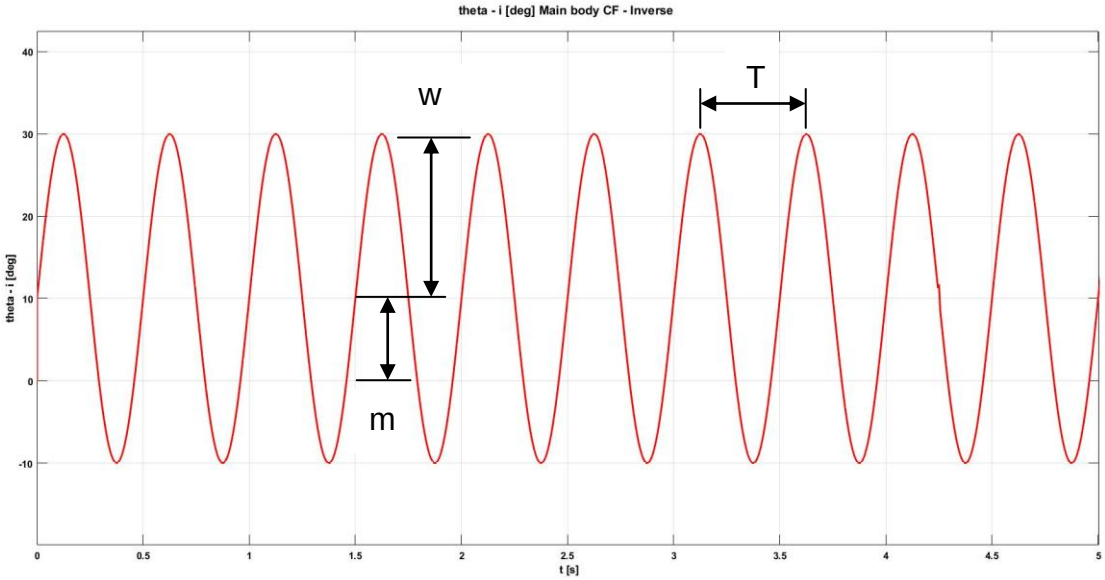
- v_{cx}, v_{cy}, και $\dot{v}_{cx}$, $\dot{v}_{cy}$: οι ταχύτητες και οι επιταχύνσεις του κέντρου μάζας του ρομποτικού ψαριού αντίστοιχα, εκφρασμένες στο σωματόδετο σύστημα συντεταγμένων
- υ_c: το διανυσματικό άθροισμα των ταχυτήτων v_{cx}, v_{cy}
- β: η γωνία που σχηματίζεται από το διάνυσμα υ_c και τον άξονα $\hat{x}$
- ψ: η γωνία στροφής του σωματόδετου συστήματος συντεταγμένων σε σχέση με το αδρανειακό
- ω_z, $\dot{\omega}_z$: η γωνιακή ταχύτητα του ρομποτικού ψαριού ως προς τον άξονα $\hat{z}$ και η αντίστοιχη γωνιακή επιτάχυνση
- m_1, m_2, c_1, c_2, c_3, c_4, c_5, c_6: σταθερές που προκύπτουν όπως περιγράφεται στο [1]
- J: η ροπή αδράνειας του ρομποτικού ψαριού
- g_1, g_2, g_3: μεταβλητές που προκύπτουν από αντικατάσταση των πιο πάνω
- $\ddot{\theta}$: η γωνιακή επιτάχυνση του ουραίου πτερυγίου

Για λόγους απλοποίησης της διαδικασίας και διευκόλυνσης της πειραματικής δοκιμής της κεντρικής μονάδας ελέγχου, ως είσοδος του συστήματος επιλέγεται η κυκλική τροχιά που προκύπτει από την επίλυση του ευθέως δυναμικού προβλήματος [1]. Η τροχιά αυτή, εφόσον προκύπτει από το ίδιο δυναμικό πρόβλημα στην αντίστροφη μορφή του, είναι δεδομένο πως ικανοποιεί την συνθήκη περιορισμού (3-11).

Από την επίλυση του συστήματος, προκύπτει το διάγραμμα της γωνίας του ουραίου πτερυγίου συναρτήσει του χρόνου. Στην περίπτωση της κυκλικής τροχιάς εισόδου, αυτό θα έχει την μορφή που

παρουσιάζεται στο (Σχήμα 3-13). Η κεντρική μονάδα ελέγχου, αναλαμβάνει να μετατρέψει την μεταβολή της γωνίας αυτής σε τρεις μεταβλητές, που περιγράφουν πλήρως τον τρόπο με τον οποίο πρέπει να κινηθεί το ουραίο πτερύγιο, ούτως ώστε να επιτευχθεί η επιθυμητή τροχιά. Οι μεταβλητές αυτές είναι:

- Η μέση γωνία ταλάντωσης του πτερυγίου (m)
- Το πλάτος ταλάντωσης του πτερυγίου (w)
- Η συχνότητα ταλάντωσης του πτερυγίου (f)



Σχήμα 3-13. Μορφή διαγράμματος γωνίας ουραίου πτερυγίου.

Η συχνότητα ταλάντωσης του πτερυγίου δίνεται ως είσοδος στο σύστημα και θεωρείται σταθερή. Για τις ανάγκες της εργασίας αυτής έχει οριστεί να είναι ίση με 2 Hz. Εναλλακτικά, η συχνότητα μπορεί να υπολογιστεί από την περίοδο ταλάντωσης του πτερυγίου (T), με τη χρήση της ακόλουθης σχέσης:

$$f = \frac{1}{T} \tag{3-12}$$

όπου η περίοδος T είναι ίση με τον χρόνο που μεσολαβεί μεταξύ δύο διαδοχικών μεγίστων στο διάγραμμα γωνίας ουραίου πτερυγίου.

Στη συνέχεια, η κεντρική μονάδα ελέγχου αναλαμβάνει την αποστολή των μεταβλητών αυτών στον μικροεπεξεργαστή του ρομποτικού ψαριού, μέσω σειριακής θύρας του κεντρικού υπολογιστή. Τα αποστελλόμενα δεδομένα, αρχικά μετατρέπονται σε μορφή χαρακτήρων ASCII και αποστέλλονται ως γραμμές κειμένου (string buffer), με την ακόλουθη μορφή:

```
mawbfc
```

όπου οι χαρακτήρες m, w και f αποτελούν δείκτες για την μέση γωνία, το πλάτος και τη συχνότητα ταλάντωσης αντίστοιχα, ενώ οι χαρακτήρες a b και c αντιπροσωπεύουν τις υπολογιζόμενες τιμές για τις μεταβλητές αυτές.

3.3.2 Υποσύστημα λήψης δεδομένων από περιφερειακή συσκευή

Το δεύτερο υποσύστημα που ενσωματώνεται στην κεντρική μονάδα ελέγχου παρέχει τη δυνατότητα σύνδεσης και λήψης δεδομένων από περιφερειακή συσκευή. Στην παρούσα εργασία, ως περιφερειακή συσκευή λαμβάνεται ο μικροϋπολογιστής Raspberry Pi.

Μία πιθανή περιφερειακή συσκευή που μπορεί να συνδεθεί μελλοντικά στον μικροϋπολογιστή είναι η ψηφιακή κάμερα, που είναι προσαρμοσμένη πάνω από τη δεξαμενή πειραματικών δοκιμών του Εργαστηρίου Αυτομάτου Ελέγχου. Η κάμερα αυτή χρησιμοποιείται για καταγραφή της κίνησης του ρομποτικού ψαριού και αποστολή των συντεταγμένων της πραγματικής του θέσης και των ταχυτήτων του στην κεντρική μονάδα ελέγχου, μετά από ψηφιακή επεξεργασία της λαμβανόμενης εικόνας. Η λήψη των μεταβλητών αυτών θεωρείται αναγκαία για την μελλοντική υλοποίηση του συστήματος ελέγχου κλειστού βρόχου, αφού αποτελεί την ανατροφοδότηση (feedback) του συστήματος. Η ανάπτυξη του λογισμικού επεξεργασίας εικόνας από τη ψηφιακή κάμερα αποτελεί αντικείμενο Μεταπτυχιακής Εργασίας, οπότε δεν κρίνεται απαραίτητη εκτενέστερη περιγραφή. Λαμβάνοντας υπόψη τα πιο πάνω, στον μικροϋπολογιστή Raspberry Pi αναπτύχθηκε λογισμικό, το οποίο προσομοιώνει τα δεδομένα που θα προέκυπταν από την ψηφιακή ανάλυση της εικόνας και στη συνέχεια τα στέλνει στην κεντρική μονάδα ελέγχου μέσω σειριακής θύρας.

Πιο συγκεκριμένα, το λογισμικό παράγει έξι τυχαίους αριθμούς με δύο δεκαδικά ψηφία, οι οποίοι αντιπροσωπεύουν τις τιμές της θέσης (x,y,z), της κατεύθυνσης (thita) και της ταχύτητας (u_x, u_y) του ρομποτικού ψαριού. Στη συνέχεια οι μεταβλητές αυτές μετατρέπονται σε μορφή χαρακτήρων ASCII και αποστέλλονται στην κεντρική μονάδα ελέγχου σειριακά, ως γραμμές κειμένου (string buffer) με την ακόλουθη μορφή:

```
x,y,z,thita,u_x,u_y/
```

όπου ο χαρακτήρας “/” σηματοδοτεί το τέλος της γραμμής κειμένου και την έναρξη ανάγνωσης της επόμενης.

3.3.3 Δομή κεντρικής μονάδας ελέγχου στο ROS

Για την ενσωμάτωση των δύο υποσυστημάτων που περιγράφηκαν στις προηγούμενες ενότητες, χρησιμοποιήθηκαν βασικά δομικά στοιχεία ενός λειτουργικού συστήματος ROS. Η κεντρική μονάδα ελέγχου αποτελείται από δύο κόμβους (camera_node, serial_connection) και ένα θέμα (real_position) που παρουσιάζονται με μπλε, πράσινο και κόκκινο χρώμα αντίστοιχα στο Σχήμα 3-14.

Σχήμα 3-14. Διάγραμμα κόμβων-θεμάτων κεντρικής μονάδας ελέγχου.

Το υποσύστημα ελέγχου ανοικτού βρόχου υλοποιείται εξολοκλήρου από τον κόμβο serial_connection και λειτουργεί ανεξάρτητα από το υπόλοιπο σύστημα. Με εκτέλεση του κόμβου αυτού (αρχείο serial_connection.py), ξεκινά η αποστολή των τριών μεταβλητών που περιγράφουν την κίνηση του ουραίου πτερυγίου στον μικροεπεξεργαστή του ρομποτικού ψαριού.

Το υποσύστημα λήψης δεδομένων από περιφερειακή συσκευή υλοποιείται με τη συνεργασία και των τριών δομικών στοιχείων. Με εκτέλεση του κόμβου camera_node (αρχείο camera_node.py), το σύστημα ενεργοποιείται και η σειριακή θύρα ανοίγει, περιμένοντας εισερχόμενα δεδομένα από τον μικροϋπολογιστή Raspberry Pi. Μόλις ξεκινήσει η λήψη των δεδομένων, ο κόμβος δημοσιεύει αυτόματα τη γραμμή δεδομένων που λαμβάνει κάθε φορά, στο topic real_position. Πλέον, η γραμμή δεδομένων αυτή είναι διαθέσιμη σε όποιον κόμβο του συστήματος είναι subscriber στο συγκεκριμένο topic. Με την εκτέλεση του κόμβου serial_connection, ο οποίος είναι subscriber στο topic real_position, η γραμμή δεδομένων αποθηκεύεται προσωρινά στο σύστημα. Μετά από κατάλληλη επεξεργασία, οι έξι τιμές που περιγράφουν την πραγματική θέση και ταχύτητα του ρομποτικού ψαριού αποθηκεύονται σε έξι διαφορετικές μεταβλητές και είναι έτοιμες για να χρησιμοποιηθούν σε τυχόν υπολογισμούς.

Στο σημείο αυτό, αξίζει να σημειωθεί ότι τα δύο υποσυστήματα είναι ανεξάρτητα μεταξύ τους, καθώς η μεταβλητές πραγματικής θέσης και ταχύτητας δεν χρησιμοποιούνται στην υλοποίηση του συστήματος ελέγχου ανοικτού βρόχου. Ωστόσο, η ανατροφοδότηση των μεταβλητών αυτών αποτελεί βάση για την μελλοντική υλοποίηση του συστήματος ελέγχου κλειστού βρόχου. Για τον λόγο αυτό, είναι προτιμότερο κατά την λειτουργία του υποσυστήματος λήψης δεδομένων από περιφερειακή συσκευή, να απενεργοποιείται η λειτουργία του συστήματος ελέγχου ανοικτού βρόχου από τον κόμβο serial_connection, με τον τρόπο που περιγράφεται σε ακόλουθη ενότητα. Λεπτομερείς ανάλυση των εκτελέσιμων αρχείων των δύο κόμβων, όπως και του λογισμικού του μικροϋπολογιστή Raspberry Pi ακολουθεί στη συνέχεια.

3.3.4 Δημιουργία περιβάλλοντος εργασίας στο ROS

Σε πρώτο στάδιο, γίνεται η επιλογή της κατάλληλης διανομής λειτουργικού ROS, λαμβάνοντας υπόψη το υπάρχον λειτουργικό σύστημα και τις απαιτήσεις του συστήματος που πρόκειται να υλοποιηθεί. Στη συνέχεια γίνεται η εγκατάσταση της διανομής ROS που επιλέχθηκε, ακολουθώντας τις αναλυτικές οδηγίες που παρέχονται στο [5]. Για τις ανάγκες του συστήματος του ρομποτικού ψαριού, επιλέχθηκε η εγκατάσταση της διανομής του ROS Kinetic που είναι συμβατή με το λειτουργικό σύστημα Linux (Ubuntu 16.04 LTS) του κεντρικού υπολογιστή της κεντρικής μονάδας ελέγχου.

Δημιουργία Catkin Workspace

Μετά την ολοκλήρωση της εγκατάστασης του λειτουργικού συστήματος ROS, είναι απαραίτητη η δημιουργία του περιβάλλοντος εργασίας (Catkin Workspace), μέσα στο οποίο πρόκειται να δομηθεί όλο το σύστημα της κεντρικής μονάδας ελέγχου.

Βασική προϋπόθεση για κάθε διεργασία που εκτελείται στο ROS σε ένα νέο τερματικό (terminal), είναι η φόρτωση του περιβάλλοντος ROS με τη χρήση της εντολής:

```
source /opt/ros/kinetic/setup.bash
```

Στη συνέχεια, μέσα στο φάκελο home του συστήματος (συμβολίζεται με “~”), δημιουργείται ο φάκελος catkin_ws. Μέσα στον φάκελο αυτό, δημιουργείται ο φάκελος src, μέσα στον οποίο

δημιουργείται το απαραίτητο αρχείο διαμόρφωσης (CMakeLists.txt) για το περιβάλλον εργασίας, με τη χρήση της εντολής .

```
kdir –p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Ακολούθως, είναι απαραίτητη η φόρτωση του αρχείου setup.*sh μέσα από τον φάκελο catkin_ws, με την εκτέλεση της εντολής:

```
source devel/setup.bash
```

Για επαλήθευση της σωστής δημιουργίας του περιβάλλοντος εργασίας, πρέπει η διαδρομή τοποθεσίας (path) του αντικειμένου ROS_PACKAGE να περιλαμβάνει το όνομα του παρόντος φακέλου. Στην συγκεκριμένη περίπτωση αυτός είναι ο φάκελος catkin_ws. Εκτελώντας την εντολή:

```
echo $ROS_PACKAGE_PATH
```

η οποία επιστρέφει το path για το αντικείμενο που της ορίζεται, προκύπτει το αποτέλεσμα:

```
/home/my_user/catkin_ws/src:/opt/ros/kinetic/share
```

Το όνομα του φακέλου catkin_ws βρίσκεται μέσα στο path του αποτελέσματος, γεγονός που επαληθεύει την επιτυχημένη δημιουργία του περιβάλλοντος εργασίας.

Δημιουργία Catkin Package

Μετά τη δημιουργία του Catkin Workspace, ακολουθεί η δημιουργία ενός Catkin Package, μέσα στο οποίο θα υλοποιηθούν οι δύο κόμβοι και το topic. Αρχικά, γίνεται μετάβαση στον φάκελο src του Catkin Workspace.

```
cd ~/catkin_ws/src
```

Η δημιουργία του Catkin Package γίνεται με τη χρήση της εντολής catkin_create_pkg, η οποία δέχεται ως ορίσματα το όνομα του package που πρόκειται να δημιουργηθεί και τις βιβλιοθήκες από τις οποίες εξαρτάται (dependencies). Η εντολή catkin_create_pkg έχει την ακόλουθη μορφή.

```
catkin_create_pkg package_name depend1 depend2 depend3
```

Στο package της συγκεκριμένης εφαρμογής δίνεται το όνομα usb_rs232 και τρία βασικά dependencies:

- **std_msgs:** Αποτελεί τον πιο συνήθη τύπο dependency. Εφαρμόζεται στις περιπτώσεις που χρησιμοποιούνται βασικοί τύποι δεδομένων (πχ. Bool, Char, Float64, Int32, String κ.α.). Περισσότερες λεπτομέρειες για το std_msg dependency παρουσιάζονται στο [37]

- **rospy:** Χρησιμοποιείται όταν το package περιλαμβάνει λογισμικό σε γλώσσα προγραμματισμού Python και περιλαμβάνει όλες τις τυποποιημένες εντολές της. Στο συγκεκριμένο package χρησιμοποιείται διότι όλο το λογισμικό της κεντρικής μονάδας ελέγχου έχει αναπτυχθεί στην γλώσσα προγραμματισμού Python. Περισσότερες λεπτομέρειες για το rospy dependency παρουσιάζονται στο [38].
- **roscpp:** Χρησιμοποιείται όταν το package περιλαμβάνει λογισμικό σε γλώσσα προγραμματισμού C++. Στη συγκεκριμένη εφαρμογή χρησιμοποιείται έτσι ώστε τυχόν προσθήκη κόμβων C++ στο σύστημα, να μην απαιτεί επανακαθορισμό των dependencies του package.

οπότε η εντολή catkin_create_pkg εκτελείται στη μορφή:

```
catkin_create_pkg usb_rs232 std_msgs rospy roscpp
```

Ακολούθως, γίνεται μετάβαση στον φάκελο του Catkin Workspace, όπου ολοκληρώνεται η δημιουργία του package (build) με τη χρήση της εντολής catkin_make.

```
cd ~/catkin_ws
catkin_make
```

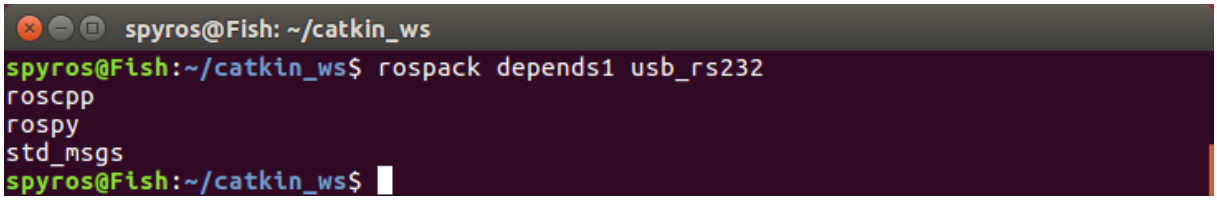
Για να είναι δυνατή η πρόσβαση στο package μέσα από το περιβάλλον του ROS, γίνεται φόρτωση του αρχείου setup.*sh (όπως και στην περίπτωση του Catkin Workspace), αυτή τη φορά με χρήση της εντολής:

```
. ~/catkin_ws/devel/setup.bash
```

Για επαλήθευση της επιτυχημένης δημιουργίας του package εκτελείται η εντολή:

```
rospack depends1 usb_rs232
```

η οποία επιστρέφει τη λίστα με τα dependencies πρώτου βαθμού που έχουν οριστεί στο package με όνομα usb_rs232. Το αποτέλεσμα της εντολής έχει τη μορφή που παρουσιάζεται στο Σχήμα 3-15.

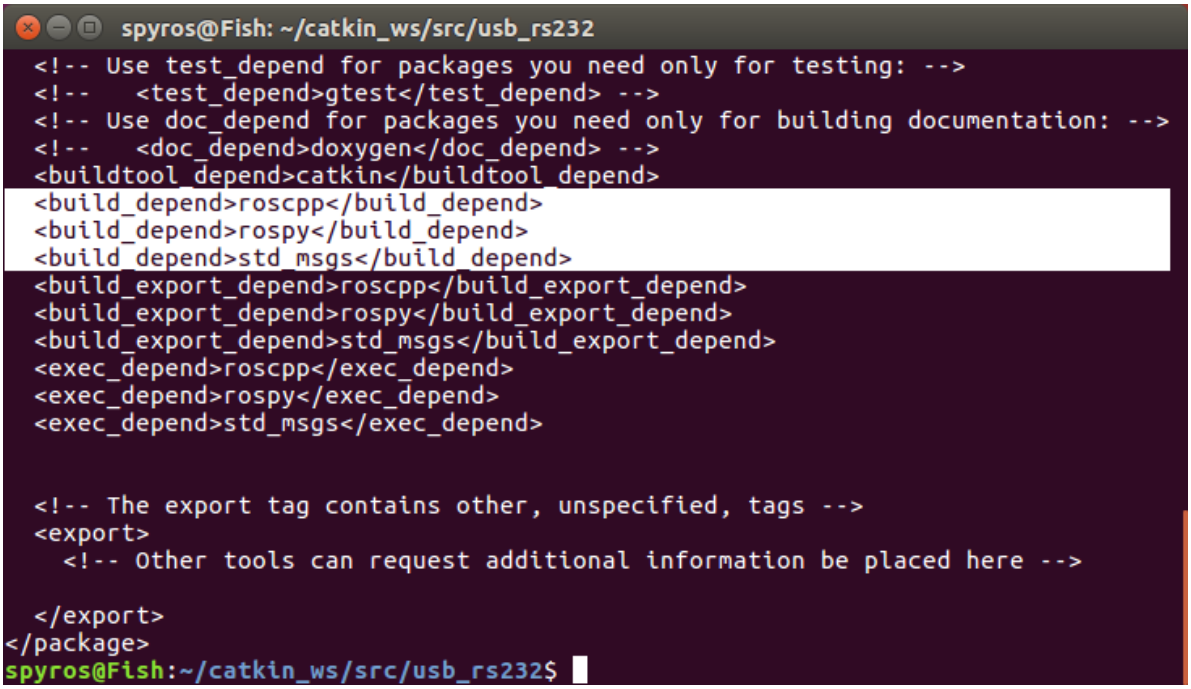


Σχήμα 3-15. Dependencies του package usb_rs232.

Εναλλακτικά, η επαλήθευση μπορεί να γίνει ελέγχοντας εάν τα dependencies που έχουν οριστεί εμφανίζονται στο τέλος του αρχείου package.xml. Μέσα από τον φάκελο του package usb_rs232, εκτελείται η εντολή cat για ανάγνωση του περιεχομένου του αρχείου.

```
roscd usb_rs232
cat package.xml
```

Στο τέλος του αρχείου package.xlm εμφανίζονται τα τρία dependencies που έχουν οριστεί, με την μορφή που παρουσιάζεται στο Σχήμα 3-16.



Σχήμα 3-16. Περιεχόμενο αρχείου package.xlm.

Τροποποίηση αρχείου package.xml

Επιπρόσθετα, για την ομαλή λειτουργία του package, είναι απαραίτητη η τροποποίηση του περιεχομένου του αρχείου package.xml. Η τροποποίηση μπορεί να γίνει με τη χρήση κάποιου εργαλείου τροποποίησης αρχείων. Στην παρούσα εργασία χρησιμοποιείται το εργαλείο nano, το οποίο ενεργοποιείται με την εκτέλεση της εντολής:

```
export EDOTOR='nano -w'
```

Στη συνέχεια, το αρχείο package.xml ανοίγει για επεξεργασία με εκτέλεση της εντολής:

```
nano package.xml
```

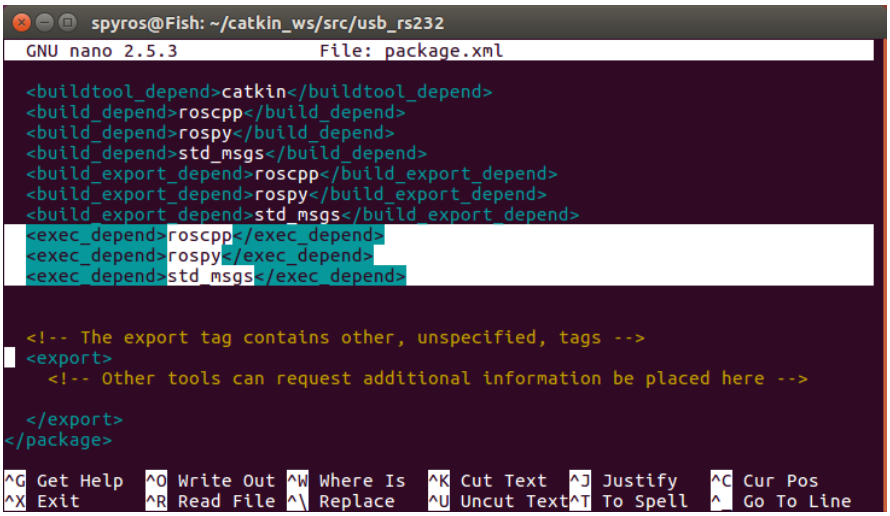
Στην ενότητα license του αρχείου, καθορίζεται η άδεια που πρόκειται να χρησιμοποιεί το package. Μερικές από τις άδειες που χρησιμοποιούνται συχνά είναι οι BSD, MIT, Boost Software Licence κ.α. Στη συγκεκριμένη εφαρμογή χρησιμοποιείται η άδεια BSD επειδή χρησιμοποιείται ήδη από την βιβλιοθήκη rospy που έχει οριστεί στο package της προηγούμενης ενότητας. Περισσότερες λεπτομέρειες για τις άδειες που μπορούν χρησιμοποιηθούν και τις εφαρμογές τους παρουσιάζονται στην ιστοσελίδα [39]. Για τον καθορισμό της άδειας, η γραμμή:

```
<license>BSD<license>
```

προστίθεται στο αρχείο package.xlm, όπως παρουσιάζεται στο σχήμα Σχήμα 3-17.

Σχήμα 3-17. Καθορισμός άδειας package.

Στην ενότητα dependencies, εμφανίζονται όλα τα dependencies που έχουν οριστεί στο package usb_rs232 και είναι τύπου “build_depend”. Τα dependencies αυτά, για να είναι διαθέσιμα κατά το χτίσιμο και την λειτουργία του package πρέπει να προστεθούν στο αρχείο ξανά, ως dependencies τύπου “exec_depend”. Για τον λόγο αυτό, τρεις νέες γραμμές προστίθενται στο αρχείο package.xlm, όπως παρουσιάζεται στο Σχήμα 3-18.



Σχήμα 3-18. Καθορισμός dependencies τύπου “exec_depend”.

Στο σημείο αυτό η τροποποίηση του αρχείου έχει ολοκληρωθεί. Οι αλλαγές αποθηκεύονται στο αρχείο, διατηρώντας το υπάρχον όνομα (package.xml).

Χτίσιμο package usb_rs232 (Building Package)

Σε περίπτωση που το τρέχον τερματικό έκλεισε, ή η φόρτωση του αρχείου setup.*sh δεν έγινε σε προηγούμενο στάδιο, η φόρτωση γίνεται ξανά με εκτέλεση της εντολής:

```
source /opt/ros/kinetic/setup.bash
```

Για αποθήκευση όλων των ρυθμίσεων και αλλαγών που έχουν υλοποιηθεί στο package usb_rs232, εκτελείται η εντολή catkin_make, μέσα από τον φάκελο του Catkin Workspace.

```
cd ~/catkin_ws
catkin_make
```

Μια γενική περιγραφή της διαδικασίας δημιουργίας ενός Catkin Package παρουσιάζεται στο tutorial [40]. Στο σημείο αυτό, οι απαραίτητες ρυθμίσεις στο package έχουν ολοκληρωθεί και μπορεί να ξεκινήσει η δημιουργία των κόμβων του συστήματος.

3.3.5 Δημιουργία κόμβου serial_connection

Το πρώτο στάδιο για τη δημιουργία ενός κόμβου Python, είναι η δημιουργία ενός φακέλου μέσα στο package, ο οποίος θα περιέχει το εκτελέσιμο αρχείο του κόμβου. Στον φάκελο αυτό αποθηκεύονται τα εκτελέσιμα αρχεία όλων των κόμβων Python που περιέχονται στο ίδιο package. Στην περίπτωση δημιουργίας ενός κόμβου C++ στο ίδιο package, ο φάκελος αυτός δεν χρησιμοποιείται. Η διαδικασία διαφέρει σημαντικά και η λεπτομερείς περιγραφή της παρουσιάζεται στο tutorial [41]. Στη παρούσα περίπτωση ο φάκελος αυτός ονομάζεται scripts και δημιουργείται με την εκτέλεση των πιο κάτω εντολών.

```
roscd usb_rs232
mkdir scripts
```

Στη συνέχεια, μέσα στο φάκελο scripts, δημιουργείται ένα κενό αρχείο με το όνομα serial_connection, το οποίο ανοίγει κατευθείαν για επεξεργασία.

```
cd scripts
cat > serial_connection.py
```

serial_connection.py

Για λόγους απλοποίησης της περιγραφής, ο κώδικας του κόμβου serial_connection παρουσιάζεται αυτούσιος στο Παράρτημα, που βρίσκεται στο τέλος της παρούσας διπλωματικής εργασίας. Στη ενότητα αυτή παρουσιάζονται τα απαραίτητα στοιχεία ενός κόμβου Python και περιγράφονται οι επιμέρους λειτουργίες που εκτελούνται στον κόμβο serial_connection.

Μία απλοποιημένη μορφή του κόμβου παρουσιάζεται στη συνέχεια:

```
1  #!/usr/bin/python
2
3  # Εισαγωγή βιβλιοθηκών
4  import rospy
```

```
5  from std_msgs.msg import String
6  .
7  .
8
9  # Ορισμός σταθερών
10 .
11 .
12
13 def txt_to_arrays():
14     .
15     .
16     .
17 def callback():
18     .
19     .
20 def inverse():
21     .
22     .
23 def solution():
24     .
25     .
26 def serial_connection():
27     .
28     .
29
30 if __name__ == '__main__':
31     try:
32         rospy.init_node('serial_connection', anonymous = True)
33         ser = serial.Serial('/dev/ttyUSB0', 115200)
34         print (ser.name)
35         time.sleep(2)
36         rospy.Subscriber('real_position', String, callback)
37         rate = rospy.Rate(200) #Hz
38         txt_to_arrays()
39         solution()
40         while not rospy.is_shutdown():
41             print("in the while loop")
42             serial_connection()
43             rate.sleep()
44         ser.close()
45     except rospy.ROSInterruptException:
46         pass
```

Κάθε κόμβος Python, ξεκινά με τη γραμμή:

```
#!/usr/bin/python
```

έτσι ώστε το αρχείο του κόμβου να εκτελείται κάθε φορά ως ένα αρχείο κώδικα Python. Στη συνέχεια, εισάγεται η βασική πακέτο rospy, το οποίο περιλαμβάνει όλες τις τυποποιημένες εντολές και βιβλιοθήκες της γλώσσας προγραμματισμού Python.

```
import rospy
```

Τα δεδομένα που λαμβάνονται και αποστέλλονται από τον κόμβο σειριακά, είναι τύπου String. Για τον λόγο αυτό, από το βασικό πακέτο std_msgs εισάγεται η βιβλιοήκη String.

```
from std_msgs.msg import String
```

Στο σημείο αυτό, εκτός από τις δύο βασικές βιβλιοθήκες που εισάγονται σε όλους τους κόμβους Python, εισάγονται και όλες οι βιβλιοθήκες που χρησιμοποιούνται από τα επιμέρους συστήματα του κόμβου, όπως περιγράφεται στη συνέχεια.

Με εκτέλεση του αρχείου serial_connection.py αποθηκεύονται στο σύστημα οι σταθερές που έχουν οριστεί και εξετάζεται η συνθήκη της γραμμής 30. Εάν ο κόμβος έχει εκτελεστεί χωρίς κάποιο σφάλμα, τότε ο κόμβος αρχικοποιείται με τη χρήση της εντολής:

```
rospy.init_node('serial_connection', anonymous = True)
```

Δίνεται στον κόμβο το όνομα serial_connection, ενώ θέτοντας το όρισμα anonymous σε True εξασφαλίζεται η μοναδικότητα του ονόματος του κόμβου. Στη συνέχεια, ενεργοποιείται η σειριακή θύρα, από την οποία αποστέλλονται τα δεδομένα στον μικροελεγκτή του ρομποτικού ψαριού, με την χρήση της εντολής:

```
ser = serial.Serial('/dev/ttyUSB0', 115200)
```

Το πρώτο όρισμα της εντολής δηλώνει το όνομα της θύρας που πρόκειται να ενεργοποιηθεί. Αυτό μπορεί να εντοπιστεί εύκολα, με την εκτέλεση της εντολής:

```
dmesg
```

σε ένα τερματικό (όπως στην περίπτωση εγκατάστασης του μετατροπέα USB – RS-232). Ως δεύτερο όρισμα της εντολής ορίζεται ο ρυθμός μεταφοράς δεδομένων (baud rate: 115200 bits/s). Για αποφυγή σφαλμάτων κατά την ενεργοποίηση της σειριακής θύρας, το πρόγραμμα περιμένει για δύο δευτερόλεπτα προτού εκτελέσει την επόμενη εντολή. Ακολούθως, η εντολή:

```
rospy.Subscriber('real_position', String, callback)
```

δηλώνει ότι ο κόμβος ου αρχικοποιήθηκε με την εντολή 32 είναι subscriber του topic real position, το οποίο είναι τύπου std_msgs.msgs.String. Κάθε φορά που μία νέα σειρά δεδομένων δημοσιεύεται στο topic real position, καλείται η συνάρτηση callback (γραμμή 17). Η συνάρτηση αυτή είναι υπεύθυνη για την λήψη των δεδομένων από το topic real position και το περιεχόμενο της αναλύεται στη συνέχεια. Με χρήση της εντολής:

```
rate = rospy.Rate(200)
```

ορίζεται η συχνότητα με την οποία λαμβάνονται τα δεδομένα από το topic real position. Η τιμή της ορίζεται σε Hz και πρέπει να είναι ίση με τη συχνότητα δημοσίευσης δεδομένων στο topic από τους κόμβους publishers. Στη συγκεκριμένη εφαρμογή είναι ίση με 200 Hz (200 times/s). Ακολούθως,

καλούνται διαδοχικά οι συναρτήσεις txt_to_arrays και solution. Η πρώτη είναι υπεύθυνη για την ανάγνωση των επιθυμητών ταχυτήτων και επιταχύνσεων του κέντρου μάζας του ρομποτικού ψαριού. Οι τιμές των μεταβλητών αυτών είναι αποθηκευμένες σε τέσσερα αρχεία τύπου txt. Αφού διαβαστούν, αποθηκεύονται εκ νέου σε τέσσερα διανύσματα, ως δεδομένα τύπου float και είναι έτοιμες για εισαγωγή τους στο σύστημα ελέγχου ανοικτού βρόχου. Η δεύτερη συνάρτηση, είναι υπεύθυνη για την επίλυση του συστήματος εξισώσεων που έχει παρουσιαστεί πιο πάνω και υπολογισμό της γωνίας θ του ουραίου πτερυγίου. Η συνάρτηση inverse καλείται μέσα από τη συνάρτηση solution και το περιεχόμενό της αναλύεται πιο κάτω.

Στη συνέχεια, το σύστημα ελέγχει την συνθήκη της γραμμής 40. Όσο ο κόμβος βρίσκεται σε λειτουργία, εκτελείται επαναλαμβανόμενα ο βρόχος των γραμμών 41-43. Η συνάρτηση serial_connection είναι υπεύθυνη για την αποστολή των δεδομένων στον μικροεπεξεργαστή του ρομποτικού ψαριού, μέσω της σειριακής θύρας. Η εντολή:

```
rate.sleep()
```

δημιουργεί την απαραίτητη καθυστέρηση, έτσι ώστε τα τηρείται η συχνότητα λήψης δεδομένων που ορίστηκε με την εντολή 37. Μόλις η εκτέλεση του προγράμματος διακοπεί με τον συνδυασμό των πλήκτρων:

```
CTRL + C
```

το σύστημα εξέρχεται του βρόχου, η σειριακή θύρα απενεργοποιείται με την εντολή 44 και ο κόμβος παύει να εκτελείται. Στο σημείο αυτό, περιγράφονται τα βασικότερα στοιχεία των συναρτήσεων που καλούνται κατά την εκτέλεση του κόμβου serial_connection.

txt_to_arrays()

```
1  import os
2  import numpy as np
3  from scipy.interpolate import interp1d
4
5  def txt_to_arrays():
6      #base_dir = rospy.get_param('~arg_name')
7      #with open(os.path.join(base_dir, 'xdot.txt'), 'r') as filehandle:
8
9      with
open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/xdot.txt'),
'r') as filehandle:
10         filecontents = filehandle.readlines()
11
12         for line in filecontents:
13             current_place = line[:-1]
14             xdot_list.append(current_place)
15
16     global xdot
17     xdot = np.array(xdot_list, dtype=np.float32)
18
19     global interp
20     interp = interp1d(t,(xdot,ydot,xdotdot,ydotdot))
```

Στο πιο πάνω παράδειγμα, παρουσιάζεται μία απλοποιημένη μορφή της συνάρτησης txt_to_arrays. Πιο συγκεκριμένα, περιγράφεται η διαδικασία ανάγνωσης των τιμών της επιθυμητής ταχύτητας $\dot{X}$ από ένα αρχείο txt και η αποθήκευση τους σε ένα διάνυσμα. Η διαδικασία επαναλαμβάνεται για την ταχύτητα $\dot{Y}$ και τις επιταχύνσεις $\ddot{X}$ και $\ddot{Y}$. Η ολοκληρωμένη μορφή της συνάρτησης παρουσιάζεται στον πλήρες κώδικα, στο Παράρτημα της εργασίας αυτής.

Σε πρώτο στάδιο, με χρήση της εντολής:

```
with open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/xdot.txt'),
'r') as filehandle:
```

το αρχείο txt ανοίγει και το σύστημα αποκτά πρόσβαση στο περιεχόμενο του. Ως όρισμα της εντολής δίνεται η απόλυτη τοποθεσία (absolute path) του αρχείου, όπως αυτό έχει αποθηκευτεί στον κεντρικό υπολογιστή. Η μέθοδος αυτή δεν συνιστάται στην περίπτωση που ο κόμβος εκτελεστεί από έναν άλλο ηλεκτρονικό υπολογιστή, καθώς η τοποθεσία των αρχείων txt μπορεί να διαφέρει από τον έναν υπολογιστή στον άλλο. Για αποφυγή της ανανέωσης του absolute path κάθε φορά που συμβαίνει αυτό, παρουσιάζεται μία εναλλακτική μέθοδος για το άνοιγμα του αρχείου, στις γραμμές 6 - 7. Η τοποθεσία του αρχείου εμφανίζεται ως μεταβλητή (arg_name), η οποία ορίζεται από τον χρήστη κατά την εκτέλεση του κόμβου από τη γραμμή εντολών. Στη συγκεκριμένη περίπτωση η εντολή αυτή είναι:

```
rosrun usb_rs232 serial_connection.py _arg_name:=$(rospack find
usb_rs232)/scripts
```

Για το άνοιγμα των αρχείων txt είναι απαραίτητη η εισαγωγή της βιβλιοθήκης os με τη χρήση της εντολής της γραμμής 1. Η βιβλιοθήκη αυτή παρέχει όλα τα εργαλεία για χρήση απόλυτων και σχετικών τοποθεσιών στο σύστημα (absolute and relative path).

Μετά το άνοιγμα του αρχείου, τα περιεχόμενα του αποθηκεύονται ανά γραμμή στην μεταβλητή filecontents, με χρήση της εντολής της γραμμής 10. Στη συνέχεια, ο βρόχος των γραμμών 12 - 14 διαβάζει κάθε γραμμή που αποθηκεύτηκε στην μεταβλητή αυτή (μία τιμή ανά γραμμή) και την προσθέτει στην λίστα xdot_list. Μετά την ολοκλήρωση της ανάγνωσης, η λίστα μετατρέπεται στο διάνυσμα xdot με την εντολή της γραμμής 17. Το διάνυσμα ορίζεται ως τύπου global, ώστε να είναι προσβάσιμο και από εντολές εκτός της συνάρτησης txt_to_arrays. Για τη δημιουργία και επεξεργασία διανυσμάτων είναι απαραίτητη η εισαγωγή της βιβλιοθήκης numpy, με την εντολή της γραμμής 2. Η βιβλιοθήκη αυτή παρέχει τα βασικά εργαλεία για μαθηματικούς υπολογισμούς, γραμμική άλγεβρα, μετασχηματισμό Fourier, και δημιουργία τυχαίων αριθμών.

Τέλος, αφού επαναληφθεί η διαδικασία και για τα υπόλοιπα αρχεία txt, πραγματοποιείται παρεμβολή μεταξύ των σημείων των διανυσμάτων, έτσι ώστε να προκύψουν οι εξισώσεις των ταχυτήτων και των επιταχύνσεων σε συνάρτηση με τον χρόνο (γραμμή 20). Οι εξισώσεις αποθηκεύονται στην μεταβλητή interp, που έχει οριστεί ως τύπου global και είναι πλέον διαθέσιμες για εισαγωγή στο σύστημα ελέγχου κλειστού βρόχου. Για τη πραγματοποίηση της παρεμβολής είναι απαραίτητη η εισαγωγή του εργαλείου interp1d της βιβλιοθήκης scipy, με την εντολή της γραμμής 3. Η

βιβλιοθήκη αυτή παρέχει χρήσιμα εργαλεία για υλοποίηση μεθόδων αριθμητικής ανάλυσης, όπως αριθμητική ολοκλήρωση, παρεμβολή μεταξύ σημείων και βελτιστοποίηση.

Οι βιβλιοθήκες numpy και scipy δεν είναι ενσωματωμένες στη διανομή της Python που χρησιμοποιείται στη παρούσα εργασία (Python 2.7.12). Για τον λόγο αυτό, πρέπει να εγκατασταθούν στον κεντρικό υπολογιστή πριν της εισαγωγή τους σε οποιοδήποτε κώδικα Python. Η εγκατάσταση τους γίνεται με τη χρήση του εργαλείου εγκατάστασης pip, το οποίο εγκαθίσταται στο σύστημα με την εκτέλεση των πιο κάτω εντολών σε ένα νέο τερματικό.

```
apt-get update
curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
python get-pip.py
```

Περισσότερες λεπτομέρειες για τον εργαλείο εγκατάστασης pip παρουσιάζονται αναλυτικά στην ιστοσελίδα [42]. Στη συνέχεια οι δύο βιβλιοθήκες εγκαθίστανται με εκτέλεση της εντολής:

```
python –m pip install --user numpy scipy
```

Εναλλακτικές μέθοδοι εγκατάστασης των βιβλιοθηκών, ανάλογα με το υπάρχον λειτουργικό σύστημα, παρουσιάζονται στην ιστοσελίδα [43]. Η διανομή της Python 2.7.12 είναι ενσωματωμένη στο λειτουργικό σύστημα Ubuntu 16.04 LTS, οπότε δεν χρειάζεται να εγκατασταθεί.

solution()

```
1  from scipy.integrate import solve_ivp
2
3  def solution():
4      vcx_i_0 = 0.1257
5      vcy_i_0 = 0
6      psi_i_0 = 0
7      wz_i_0 = 0
8      theta_i_0 = 0
9      theta_deg_i_0 = 0
10     k0 = [vcx_i_0, vcy_i_0, psi_i_0, wz_i_0, theta_i_0, theta_deg_i_0]
11
12     # Epilysi systimatos diaforikwn
13     steps = 1
14     method = 'RK23'
15     atol = 1e-12 # (default = 1e-6)
16     k = solve_ivp(inverse, (0, 100), k0, method=method, t_eval=t,
atol=atol, vectorized=True)
17
18     # apothikeysi apotelesmatwn
19     vcx_i = k.y[0,:]
20     vcy_i = k.y[1,:]
21     psi_i = k.y[2,:]
22     wz_i = k.y[3,:]
23     theta_i = k.y[4,:]
24     theta_deg_i = k.y[5,:]
25     k_new = np.transpose(k.y)
26     theta_i = [inverse(t_i, k_i)[4] for t_i, k_i in zip(t, k_new)]
27     theta_deg_i = [inverse(t_i, k_i)[5] for k_i, t_i in zip(k_new, t)]
28
```

```
29      # Ypologismos mesis gwnias theta kai platus talantwsis
30      global mesi_gwnia, platos
31      mesi_gwnia_full = sum(theta_deg_i)/len(theta_deg_i) # deg
32      theta_deg_i_trim = theta_deg_i[1000:7000]
33      platos_full = (max(theta_deg_i_trim)-min(theta_deg_i_trim))/2
34      mesi_gwnia = round(mesi_gwnia_full,1)
35      platos = round(platos_full,1)
```

Η συνάρτηση solution είναι υπεύθυνη για την επίλυση του συστήματος εξισώσεων (6x6), που υλοποιεί το σύστημα ελέγχου ανοικτού βρόχου. Με την κλήση της συνάρτησης αυτής προκύπτουν η μέση γωνία και το πλάτος ταλάντωσης του ουραίου πτερυγίου υπολογισμένα σε μοίρες.

Σε πρώτο στάδιο, ορίζονται οι αρχικές συνθήκες του συστήματος και αποθηκεύονται στην λίστα k0 (γραμμές 4 – 10). Για την επίλυση του συστήματος των τεσσάρων διαφορικών εξισώσεων ($\dot{v}_{cx}, \dot{v}_{cy}, \dot{\psi}, \dot{\omega}_z$) χρησιμοποιείται η συνάρτηση solve_ivp:

```
k = solve_ivp(inverse, (0, 100), k0, method=method, t_eval=t, atol=atol,
vectorized=True)
```

Η συνάρτηση αυτή, επιλύει το σύστημα διαφορικών εξισώσεων που ορίζεται στην συνάρτηση inverse, με τη χρήση της αριθμητικής μεθόδου Range-Kutta δεύτερης τάξης. Ως διάστημα ολοκλήρωσης ορίζεται το διάστημα tε[0,100], ενώ ως αρχικές συνθήκες λαμβάνονται οι τιμές της λίστας k0. Οι χρονικές στιγμές στις οποίες υπολογίζεται η λύση καθορίζονται από το διάνυσμα t, που ορίζεται στη συνάρτηση inverse. Το απόλυτο σφάλμα (atol) είναι ίσο με $1x10^{-12}$. Οι λύσεις του συστήματος εξάγονται σε μορφή διανυσμάτων, οπότε το όρισμα vectorized έχει τεθεί σε κατάσταση True. Τα αποτελέσματα του συστήματος αποθηκεύονται στο διάνυσμα k.y, που είναι διάστασης 6x8000. Για τη χρήση της συνάρτησης solve_ivp είναι απαραίτητη η εισαγωγή της βιβλιοθήκης solve_ivp από την βιβλιοθήκη scipy.integrate (γραμμή 1). Περισσότερες λεπτομέρειες για τις δυνατότητες που προσφέρει η συνάρτηση solve_ivp παρουσιάζονται στην ιστοσελίδα [44].

Στη συνέχεια, οι λύσεις του συστήματος διαφορικών εξισώσεων εξάγονται από το διάνυσμα k.y και αποθηκεύονται σε επιμέρους διανύσματα διάστασης 8000x1 (γραμμές 19 – 24). Με χρήση των εντολών των γραμμών 25 – 27, οι λύσεις του συστήματος διαφορικών εξισώσεων εισάγονται στις εξισώσεις υπολογισμού της γωνίας θ. Τα αποτελέσματα, γωνία θ[rad] και θ[deg], αποθηκεύονται επίσης σε διανύσματα διαστάσεων 8000x1.

Ακολούθως, υπολογίζεται η μέση γωνία ταλάντωσης από τον μέσο όρο των τιμών της γωνίας θ σε μοίρες (γραμμή 31). Υπολογίζοντας το εύρος ταλάντωσης από τη διαφορά μεταξύ μέγιστης και ελάχιστης τιμής θ και διαιρώντας το στη συνέχεια διά δύο, προκύπτει το πλάτος ταλάντωσης του ουραίου πτερυγίου (γραμμή 33). Τέλος, οι τιμές της μέσης γωνίας και του πλάτους στρογγυλοποιούνται ως προς το πρώτο δεκαδικό ψηφίο (γραμμές 34 – 35) και πλέον μπορούν να αποσταλούν μέσω της σειριακής θύρας.

inverse()

```
1  def inverse(t,k):
2      vcx_i = k[0]
3      vcy_i = k[1]
4      psi_i = k[2]
5      wz_i = k[3]
```

```
6       theta_i = k[4]
7       theta_deg_i = k[5]
8
9       xdot, ydot, xdotdot, ydotdot = interp(t)
10
11      vcx_i = xdot*np.cos(psi_i)-ydot*np.sin(psi_i)
12      vcy_i = ydot*np.cos(psi_i)+xdot*np.sin(psi_i)
13      psidot_i = wz_i
14      vcxdot_i = xdotdot*np.cos(psi_i)-xdot*np.sin(psi_i)*psidot_i-
ydotdot*np.sin(psi_i)-ydot*np.cos(psi_i)*psidot_i
15      vcydot_i = ydotdot*np.cos(psi_i)-
ydot*np.sin(psi_i)*psidot_i+xdotdot*np.sin(psi_i)+xdot*np.cos(psi_i)*psidot
_i
16
17      g1 = -(m1/c3)*vcxdot_i+(m2/c3)*vcy_i*wz_i-
(c1/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))+(c2/c3)*vcy_i*np.sqrt((vcx_i**
2)+(vcy_i**2))*np.arctan2(vcy_i,vcx_i)
18      g2 =
(m2/c3)*vcydot_i+(m1/c3)*vcx_i*wz_i+(c1/c3)*vcy_i*np.sqrt((vcx_i**2)+(vcy_i
**2))+(c2/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))*np.arctan2(vcy_i,vcx_i)

19      A = 12*np.sin(2*np.pi*f*t+np.pi) #tail frequency
20      if A>=0.1:
21          wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-
c5*g2-c6*np.sqrt((g1**2)+(g2**2))
22      elif A<-0.1:
23          wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-
c5*g2+c6*np.sqrt((g1**2)+(g2**2))
24      else:
25          wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-c5*g2
26
27
28      if g2>0:
29          theta_i = np.arctan2(g1,g2)
30      elif g2<0 and g1>=0:
31          theta_i = np.arctan2(g1,g2)-np.pi
32      elif g2<0 and g1<0:
33          theta_i = np.arctan2(g1,g2)+np.pi
34      elif g2==0 and g1>0:
35          theta_i = -np.pi/2
36      elif g2==0 and g1<0:
37          theta_i = np.pi/2
38      elif g1==0 and g2==0:
39          theta_i = 0
40
41      theta_deg_i = (theta_i*180)/np.pi
42      return [vcxdot_i, vcydot_i, psidot_i, wzdot_i, theta_i, theta_deg_i]
```

Η συνάρτηση inverse περιλαμβάνει τις εξισώσεις του συστήματος ελέγχου ανοικτού βρόχου και καλείτε από τη συνάρτηση solution κατά την επίλυση του συστήματος. Αρχικά, οι άγνωστοι του συστήματος ορίζονται ως διανύσματα γραμμής του διανύσματος k (γραμμές 2 – 7). Οι εξισώσεις των επιθυμητών ταχυτήτων και επιταχύνσεων που προέκυψαν από την παρεμβολή στη συνάρτηση txt_to_arrays αποθηκεύονται σε τέσσερεις μεταβλητές (γραμμή 9).

Στη συνέχεια, ορίζονται οι εξισώσεις του συστήματος, όπως αυτές παρουσιάζονται αναλυτικά στο [1]. Ακολουθείται η ίδια διαδικασία επίλυσης με αυτήν που χρησιμοποιείται στην προσομοίωση του συστήματος ελέγχου ανοικτού βρόχου στο περιβάλλον Simulink. Η συχνότητα ταλάντωσης του πτερυγίου εισάγεται στο σύστημα υπό τη μορφή της ημιτονοειδούς συνάρτησης της γραμμής 19. Η τιμή της συνάρτησης αυτής καθορίζει ποια από τις τρεις εξισώσεις της wzdot_i χρησιμοποιείται σε

κάθε επανάληψη. Ακολούθως, οι υπολογιζόμενες τιμές των συναρτήσεων g1 και g2 καθορίζουν την τιμή της γωνιάς του ουραίου πτερυγίου σε rad (γραμμές 28 – 39). Τέλος, η εξίσωση της γραμμής 41 μετατρέπει την τιμή της γωνίας θ σε μοίρες.

serial_connection()

```
1  import serial
2  import time
3
4  def serial_connection():
5      if not rospy.is_shutdown():
6          if ser.isOpen():
7              print ("Port Open")
8              for i in range(1):
9                  ser.write('m'+bytes(mesi_gwnia))
10                 time.sleep(0.001)
11                 ser.write('w'+bytes(platos))
12                 time.sleep(0.001)
13                 ser.write('f'+bytes(f))
14                 time.sleep(0.001)
```

Η συνάρτηση serial_connection είναι υπεύθυνη για την αποστολή των παραμέτρων ταλάντωσης του ουραίου πτερυγίου στον μικροεπεξεργαστή του ρομποτικού ψαριού μέσω σειριακής θύρας. Εάν ο κόμβος συνεχίζει να εκτελείται χωρίς κάποιο σφάλμα και η σειριακή θύρα είναι ενεργοποιημένη, τότε η μέση γωνία, το πλάτος και η συχνότητα ταλάντωσης αποστέλλονται σειριακά. Τα δεδομένα αυτά, ακολουθούν τη μορφή της γραμμής κειμένου που περιγράφηκε σε προηγούμενη ενότητα. Οι τιμές των μεταβλητών μετατρέπονται σε χαρακτήρες ASCII με την εντολή bytes() και αφού προστεθούν στον αντίστοιχο δείκτη (m, w, f), αποστέλλονται διαδοχικά με διαφορά 1 ms μεταξύ τους. Για τη χρήση της σειριακής θύρας και τη διατήρηση των χρονικών διαστημάτων είναι απαραίτητη η εισαγωγή των βιβλιοθηκών serial και time (γραμμές 1 – 2).

callback()

```
1  def callback(data):
2      global camera
3      camera = data.data
4      str_x_c, str_y_c, str_z_c, str_theta_c, str_ux_c, str_uy_c =
camera.strip().rstrip("/").split(",")[:6]
5      x_c = float(str_x_c)
6      y_c = float(str_y_c)
7      z_c = float(str_z_c)
8      theta_c = float(str_theta_c)
9      ux_c = float(str_ux_c)
10     uy_c = float(str_uy_c)
11
12     print x_c, y_c, z_c, theta_c, ux_c, uy_c
```

Η συνάρτηση callback καλείται από τον κόμβο κάθε φορά που μία νέα σειρά δεδομένων δημοσιεύεται στον topic real_position και είναι υπεύθυνη για την λήψη των δεδομένων. Οι έξι μεταβλητές που δημοσιεύονται στο topic έχουν την μορφή γραμμής κειμένου, όπως περιγράφεται σε

προηγούμενη ενότητα. Η γραμμή αυτή, αποθηκεύεται προσωρινά στην μεταβλητή camera (γραμμή 3). Στη συνέχεια, οι έξι τιμές εξάγονται από την γραμμή κειμένου ως δεδομένα τύπου string και αποθηκεύονται σε έξι επιμέρους μεταβλητές (γραμμή 4). Τέλος, οι μεταβλητές μετατρέπονται σε δεδομένα τύπου float και είναι διαθέσιμες για περαιτέρω επεξεργασία από το σύστημα.

3.3.6 Δημιουργία κόμβου camera_node

camera_node.py

```
1  #!/usr/bin/python
2
3  import rospy
4  import serial
5  import time
6  from std_msgs.msg import String
7
8  ser = serial.Serial('/dev/ttyS0', 115200)
9  print (ser.name)
10 time.sleep(2)
11
12 def camera_node():
13     pub = rospy.Publisher('real_position', String, queue_size=10)
14     rospy.init_node('camera_node', anonymous=True)
15     rate = rospy.Rate(200) #Hz
16
17     if ser.isOpen():
18         camera_str = ser.readline()
19         rospy.loginfo(camera_str)
20         pub.publish(camera_str)
21         rate.sleep()
22
23 if __name__ == '__main__':
24     try:
25         while not rospy.is_shutdown():
26           camera_node()
27         ser.close()
28     except rospy.ROSInterruptException:
29         pass
```

Ο κόμβος camera_node είναι υπεύθυνος για την λήψη των δεδομένων από τη σειριακή θύρα (αυτήν που συνδέεται με το Raspberry Pi) και την δημοσίευση τους στο topic real position. Αρχικά, όπως σε κάθε κόμβο Python, χρησιμοποιείται η εντολή:

```
#!/usr/bin/python
```

ώστε το αρχείο να διαβάζεται από το σύστημα ως έναν αρχείο Python. Επίσης, εισάγονται οι βιβλιοθήκες rospy και String για τους ίδιους λόγους που περιγράφονται στην ανάλυση του κόμβου serial_connection. Με εκτέλεση του κόμβου camera_node, η σειριακή θύρα ενεργοποιείται με χρήση της εντολής:

```
ser = serial.Serial('/dev/ttyS0', 115200)
```

Στη συγκεκριμένη περίπτωση, το όνομα της σειριακής θύρας είναι ttyS0 και εντοπίζεται με χρήση της εντολής dmesg, όπως περιγράφεται στην ενότητα του κόμβου serial_connection. Ο ρυθμός μετάδοσης δεδομένων ορίζεται στα 115200 bits/s και είναι ίσος με αυτόν που ορίζεται στο λογισμικό του Raspberry Pi. Ακολουθεί παύση της εκτέλεσης, διάρκειας 2 s, για αποφυγή σφαλμάτων κατά την ενεργοποίηση της σειριακής θύρας.

Στη συνέχεια, ελέγχεται η συνθήκη της γραμμής 25 και όσο ο κόμβος εκτελείται χωρίς να προκύπτει κάποιο σφάλμα, καλείται η συνάρτηση camera_code. Η εντολή:

```
pub = rospy.Publisher('real_position', String, queue_size=10)
```

δηλώνει ότι ο κόμβος αυτός δημοσιεύει δεδομένα τύπου String στο topic real_position. Το όρισμα queue_size ορίζει τον μέγιστο αριθμό μηνυμάτων που μπορεί να αποθηκεύσει το topic προσωρινά πριν να ξεκινήσει να διαγράφει τα προηγούμενα. Για παράδειγμα, εάν ο ρυθμός δημοσίευσης μηνυμάτων είναι μεγαλύτερος από αυτόν που μπορεί να διαχειριστεί το topic, τότε μπορούν να αποθηκευτούν μέχρι και 10 μηνύματα προσωρινά, πριν να ξεκινήσει η διαγραφή των προηγούμενων. Ακολούθως, η εντολή:

```
rospy.init_node('camera_node', anonymous=True)
```

αρχικοποιεί τον κόμβο, δίνοντας του το όνομα camera_node. Το όρισμα anonymous στην κατάσταση True, εξασφαλίζει την μοναδικότητα του ονόματος του κόμβου στο σύστημα. Η εντολή:

```
rate = rospy.Rate(200)
```

ορίζει η συχνότητα με την οποία δημοσιεύονται τα δεδομένα στο topic real position. Η τιμή της ορίζεται σε Hz και πρέπει να είναι ίση με τη συχνότητα λήψης δεδομένων από τους κόμβους subscribers. Στη συγκεκριμένη εφαρμογή είναι ίση με 200 Hz (200 times/s). Στη συνέχεια, εξετάζεται η συνθήκη της γραμμής 17 και αν η σειριακή θύρα είναι ενεργοποιημένη, τότε η γραμμή δεδομένων που αποστέλλεται από το Raspberry Pi αποθηκεύεται προσωρινά στην μεταβλητή camera_str. Η εντολή:

```
rospy.loginfo(camera_str)
```

εμφανίζει στην οθόνη του υπολογιστή το περιεχόμενο της μεταβλητής αυτής και το αποθηκεύει στα αρχεία καταγραφής (log files) του κόμβου. Η δημοσίευση των δεδομένων στο topic real_position γίνεται με την εντολή:

```
pub.publish(camera_str)
```

Τέλος, η εντολή:

```
rate.sleep()
```

δημιουργεί την απαραίτητη καθυστέρηση, έτσι ώστε τα τηρείται η συχνότητα δημοσίευσης δεδομένων που ορίστηκε με την εντολή 15. Με συνδυασμό των πλήκτρων:

```
CTRL + C
```

το σύστημα εξέρχεται του βρόχου 25 – 26, η σειριακή θύρα απενεργοποιείται και η εκτέλεση του κόμβου σταματά.

3.3.7 Εκτέλεση κόμβων

Μετά τη δημιουργία των αρχείων serial_connection.py και camera_node.py, είναι απαραίτητη η μετατροπή του σε εκτελέσιμα αρχεία, ώστε να μπορούν να αναγνωριστούν από το σύστημα κατά την εκτέλεση των κόμβων. Η μετατροπή υλοποιείται με εκτέλεση της εντολής:

```
chmod +x [node_name].py
```

μέσα από τον φάκελο που είναι αποθηκευμένα τα αρχεία, χρησιμοποιώντας το όνομα του κάθε αρχείου αντίστοιχα.

Ακολούθως, για αποθήκευση οποιονδήποτε αλλαγών και μεταγλώττιση (compiling) των εκτελέσιμων αρχείων, εκτελείται η εντολή catkin_make μέσα από το φάκελο του catkin workspace.

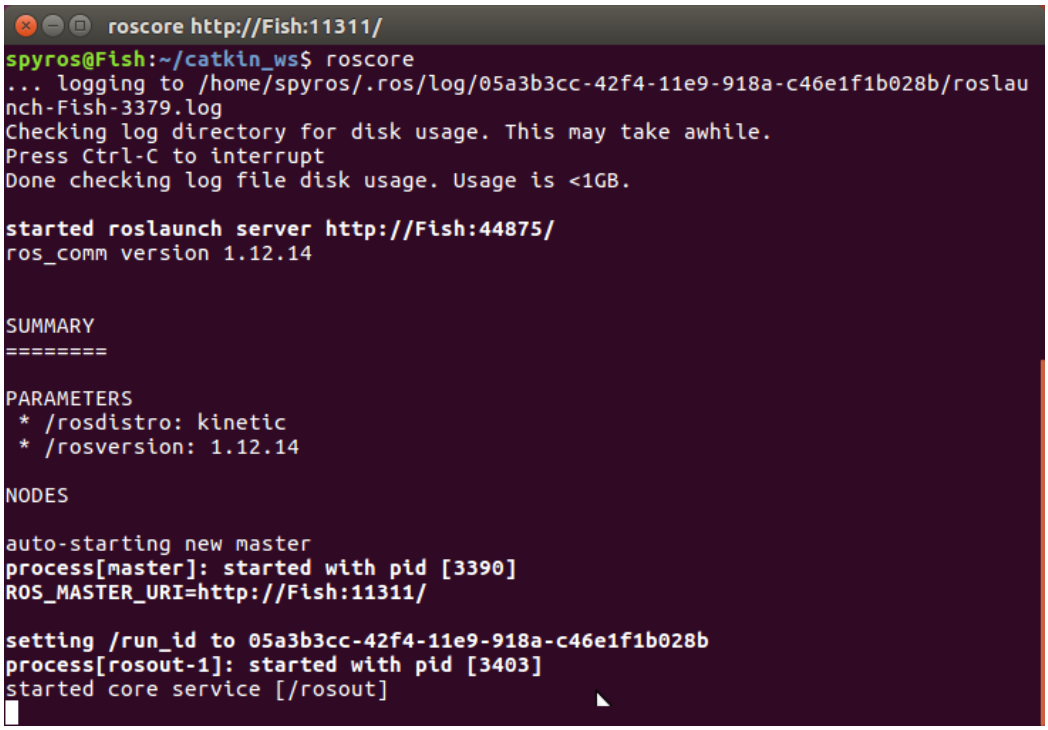
```
cd ~/catkin_ws
catkin_make
```

Πριν την εκτέλεση οποιουδήποτε κόμβου, είναι απαραίτητη η ενεργοποίηση του κόμβου master, ο οποίος επιτρέπει και συντονίζει την επικοινωνία μεταξύ των κόμβων του συστήματος ROS. Σε ένα νέο τερματικό, εκτελείται η εντολή roscore, αφού πρώτα γίνει η φόρτωση του περιβάλλοντος ROS. Το αποτέλεσμα της εντολής παρουσιάζεται στο Σχήμα 3-19.

```
source /opt/ros/kinetic/setup.bash
cd ~/catkin_ws
source devel/setup.bash
roscore
```

Ο κόμβος master ενεργοποιείται μία φορά, πριν από τη εκτέλεση του πρώτου κόμβου και παραμένει ενεργοποιημένος καθ' όλη τη διάρκεια λειτουργίας του συστήματος ROS. Για την εκτέλεση ενός κόμβου χρησιμοποιείται η εντολή rosrun στο αρχικό τερματικό (όχι αυτό που ενεργοποιεί τον ROS master).

```
rosrun usb_rs232 [node_name].py
```



Σχήμα 3-19. Αποτέλεσμα εντολής roscore.

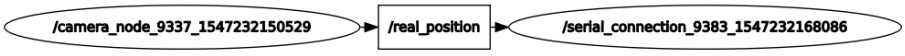
Στο πεδίο node_name τοποθετείται το όνομα του κόμβου που πρόκειται να εκτελεστεί κάθε φορά. Τα αποτελέσματα του κόμβου ξεκινούν να εμφανίζονται στο τερματικό εκτέλεσης. Η εκτέλεση του κόμβου σταματά με τον συνδυασμό των πλήκτρων:

CTRL + C

Για αποφυγή σφαλμάτων κατά τη λειτουργία του συστήματος, πρέπει να σημειωθεί ότι:

- Μετά από οποιαδήποτε αλλαγή στα εκτελέσιμα αρχεία, είναι απαραίτητη η επανάληψη της διαδικασίας μεταγλώττισης (compiling) πριν από την εκτέλεση του κόμβου. Σε κάθε άλλη περίπτωση οι αλλαγές δεν πρόκειται να εκτελεστούν από το σύστημα.
- Κάθε κόμβος εκτελείται σε ένα διαφορετικό τερματικό, αφού γίνει πρώτα φόρτωση του περιβάλλοντος ROS όπως περιγράφεται πιο πάνω.

Η επαλήθευση της σωστής λειτουργίας του συστήματος μπορεί να γίνει με τη χρήση των εντολών rqt_graph και rostopic (σε διαφορετικά τερματικά). Η πρώτη εμφανίζει το διάγραμμα κόμβων – topics για τους κόμβους που βρίσκονται σε λειτουργία εκείνη την στιγμή. Στην περίπτωση ταυτόχρονης λειτουργίας των δύο κόμβων του συστήματος, το διάγραμμα θα έχει την μορφή που παρουσιάζεται στο Σχήμα 3-20.



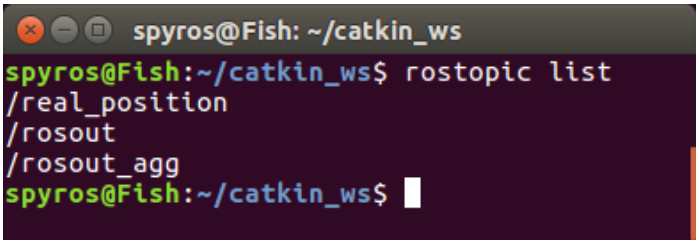
Σχήμα 3-20. Αποτέλεσμα εντολής rqt_graph.

Στη γενική μορφή του διαγράμματος, οι κόμβοι παρουσιάζονται με ελλειψοειδή σχήμα, ενώ τα topics με παραλληλόγραμμο σχήμα. Η πορεία που ακολουθούν τα δεδομένα εκφράζεται με τόξα που ξεκινούν από τους κόμβους publishers και καταλήγουν στους κόμβους subscribers.

Η εντολή rostopic παρέχει χρήσιμες πληροφορίες, ανάλογα με το όρισμα που μπορεί να δεχτεί. Για παράδειγμα, η εντολή:

```
rostopic list
```

εμφανίζει την λίστα με τα topics που βρίσκονται σε λειτουργία την δεδομένη στιγμή (έχουν τουλάχιστον έναν publisher ή έναν subscriber). Το αποτέλεσμα της εντολής αυτής παρουσιάζεται στο Σχήμα 3-21.

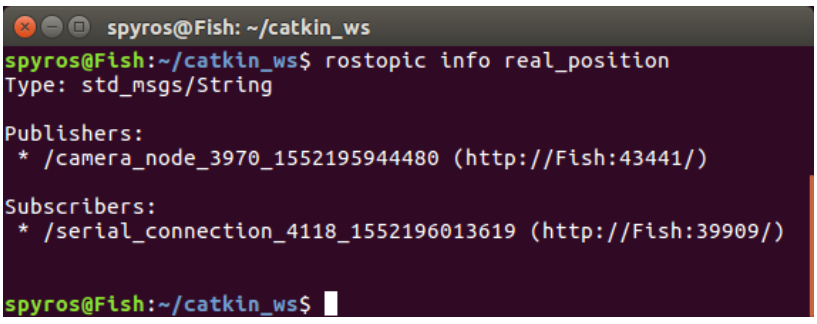


Σχήμα 3-21. Αποτέλεσμα εντολής rostopic list.

Επιπρόσθετα, η εντολή:

```
rostopic info [topic_name]
```

εμφανίζει πληροφορίες για το topic που της δίνεται ως όρισμα. Οι πληροφορίες αυτές, μεταξύ άλλων, είναι ο τύπος δεδομένων που δημοσιεύονται στο topic και τα ονόματα των κόμβων που είναι publishers ή subscribers. Στην περίπτωση του topic real_position, το αποτέλεσμα παρουσιάζεται στο Σχήμα 3-22.



Σχήμα 3-22. Αποτέλεσμα εντολής rostopic info.

Ακόμα, εάν κάποιο topic βρίσκεται σε λειτουργία, η εντολή:

```
rostopic echo [topic_name]
```

εμφανίζει στην οθόνη τα δεδομένα που λαμβάνει ή στέλνει το topic αυτό. Οι επιλογές αυτές παρέχουν την δυνατότητα ανίχνευσης σφαλμάτων ανά τμήματα του συστήματος και διευκολύνουν την επίβλεψη της λειτουργίας του. Περισσότερες λεπτομέρειες για τις δυνατότητες της εντολής rostopic παρουσιάζονται στην ιστοσελίδα [45].

3.4 Ανάπτυξη Λογισμικού Μικροϋπολογιστή Raspberry Pi

Όπως αναφέρθηκε και σε προηγούμενη ενότητα, ο μικροϋπολογιστής Raspberry Pi μπορεί να υποστηρίξει δικό του λειτουργικό σύστημα, που είναι συνήθως κάποια διαφοροποιημένη διανομή των Linux. Στον μικροϋπολογιστή Raspberry Pi 2 Model B της εφαρμογής αυτής, εγκαταστάθηκε το λειτουργικό σύστημα Raspbian, με τη βοήθεια του αναλυτικού οδηγού της ιστοσελίδας [46].

3.4.1 Ρύθμιση σειριακής θύρας

Εκ κατασκευής, η σειριακή θύρα του Raspberry Pi είναι ρυθμισμένη για να λειτουργεί ως μέθοδος σύνδεσης κονσόλας. Η ρύθμιση αυτή είναι χρήσιμη στην επίλυση προβλημάτων κατά την εκκίνηση του μικροϋπολογιστή. Ωστόσο, για να είναι δυνατή η χρήση της σειριακής θύρας για σύνδεση και επικοινωνία με άλλες συσκευές (π.χ. υπολογιστές, εκτυπωτές κ.α.), η ρύθμιση σύνδεσης κονσόλας πρέπει να απενεργοποιηθεί.

Για τον εντοπισμό των διαθέσιμων σειριακών θυρών, στο τερματικό του Raspberry Pi εκτελείται η εντολή:

```
dmesg | grep tty
```

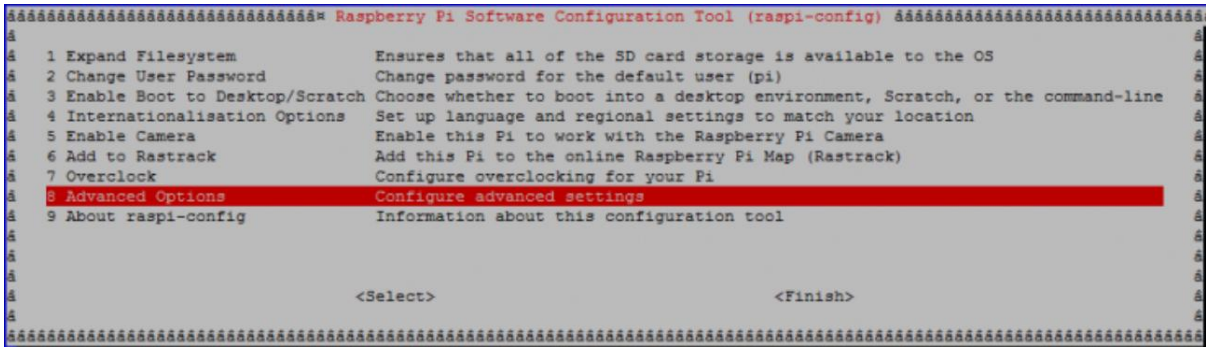
Το αποτέλεσμα της εντολής παρουσιάζεται στη συνέχεια.

```
pi@raspberrypi ~ $ dmesg | grep tty
  [    0.000000] Kernel command line: dma.dmachans=0x7f35
bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2709.boardrev=0xa01041
bcm2709.serial=0x93f9c7f9 smsc95xx.macaddr=B8:27:EB:F9:C7:F9
bcm2708_fb.fbswap=1 bcm2709.disk_led_gpio=47
bcm2709.disk_led_active_low=0 sdhci-bcm2708.emmc_clock_freq=250000000
vc_mem.mem_base=0x3dc00000 vc_mem.mem_size=0x3f000000
dwc_otg.lpm_enable=0 console=tty1 console=ttyAMA0,115200
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

  [    0.001774] console [tty1] enabled
  [    0.749509] dev:f1: ttyAMA0 at MMIO 0x3f201000 (irq = 83, base_baud
= 0) is a PL011 rev3
  [    1.268971] console [ttyAMA0] enabled
  pi@raspberrypi ~ $
```

Η τελευταία γραμμή, υποδηλώνει ότι η ρύθμιση σύνδεσης κονσόλας είναι ενεργοποιημένη στη σειριακή θύρα ttyAMA0. Για την απενεργοποίηση της, ακολουθείται η πιο κάτω διαδικασία. Πρώτα, εκτελείται η εντολή:

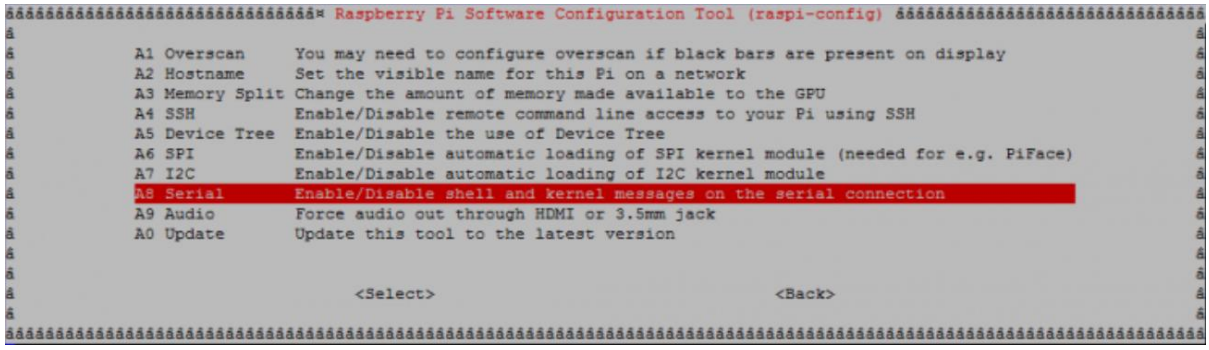
```
sudo raspi-config
```

η οποία εμφανίζει το μενού ρυθμίσεων του Raspberry Pi (βλ. Σχήμα 3-23).



Σχήμα 3-23. Μενού ρυθμίσεων Raspberry Pi.

Από το μενού αυτό, επιλέγεται το πεδίο Advanced Settings, το οποίο οδηγεί σε ένα νέο μενού επιλογών (βλ. Σχήμα 3-24).



Σχήμα 3-24. Advanced settings Raspberry Pi.

Από το νέο μενού, επιλέγεται το πεδίο serial. Στο ερώτημα που προκύπτει επιλέγεται να μην χρησιμοποιείται η σειριακή θύρα ως μέθοδος σύνδεσης κονσόλας. Ένα νέο παράθυρο επιβεβαιώνει την απενεργοποίηση της ρύθμισης. Στη συνέχεια, γίνεται επανεκκίνηση του μικροϋπολογιστή και εκτελείται ξανά στο τερματικό η εντολή:

```
dmesg | grep tty
```

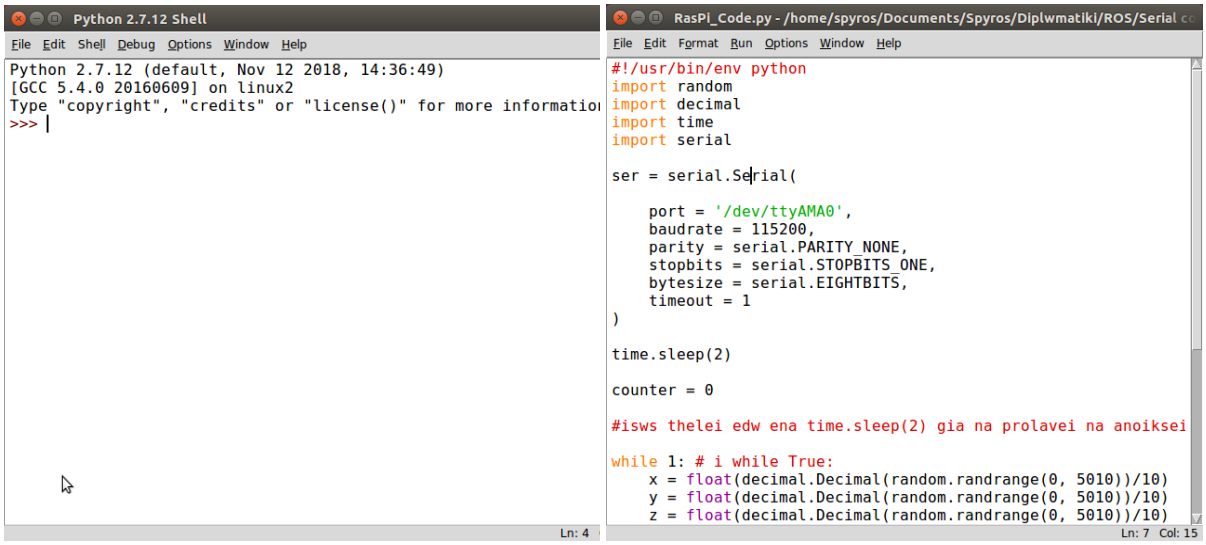
Αυτή τη φορά προκύπτει το αποτέλεσμα:

```
pi@raspberrypi ~ $ dmesg | grep tty
  [    0.000000] Kernel command line: dma.dmachans=0x7f35
bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2709.boardrev=0xa01041
bcm2709.serial=0x93f9c7f9 smsc95xx.macaddr=B8:27:EB:F9:C7:F9
bcm2708_fb.fbswap=1 bcm2709.disk_led_gpio=47
bcm2709.disk_led_active_low=0 sdhci-bcm2708.emmc_clock_freq=250000000
vc_mem.mem_base=0x3dc00000 vc_mem.mem_size=0x3f000000
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait
  [    0.001769] console [tty1] enabled
  [    0.749438] dev:f1: ttyAMA0 at MMIO 0x3f201000 (irq = 83, base_baud
= 0) is a PL011 rev3
  pi@raspberrypi ~ $
```

Όπως αποδεικνύεται από τις δύο τελευταίες γραμμές του αποτελέσματος, η ρύθμιση σύνδεσης κονσόλας έχει απενεργοποιηθεί. Πλέον, είναι δυνατή η χρήση της σειριακής θύρας ttyAMA0 για σύνδεση και επικοινωνία με περιφερειακή συσκευή. Ένα παράδειγμα σύνδεσης του Raspberry Pi με περιφερειακή συσκευή, μέσω σειριακής θύρας, παρουσιάζεται στην ιστοσελίδα [47].

3.4.2 Ενσωματωμένο περιβάλλον ανάπτυξης λογισμικού IDLE

Το λογισμικό που εκτελείται στον μικροϋπολογιστή Raspberry Pi, αναπτύχθηκε στο ενσωματωμένο περιβάλλον ανάπτυξης IDLE. Το περιβάλλον αυτό, αποτελεί έναν από τους πιο εύχρηστους και απλούς τρόπους ανάπτυξης λογισμικού στην γλώσσα προγραμματισμού Python. Αποτελείται από δύο κεντρικά παράθυρα, ένα της γραμμής εντολών και ένα για την επεξεργασία των εκτελέσιμων αρχείων (βλ. Σχήμα 3-25).



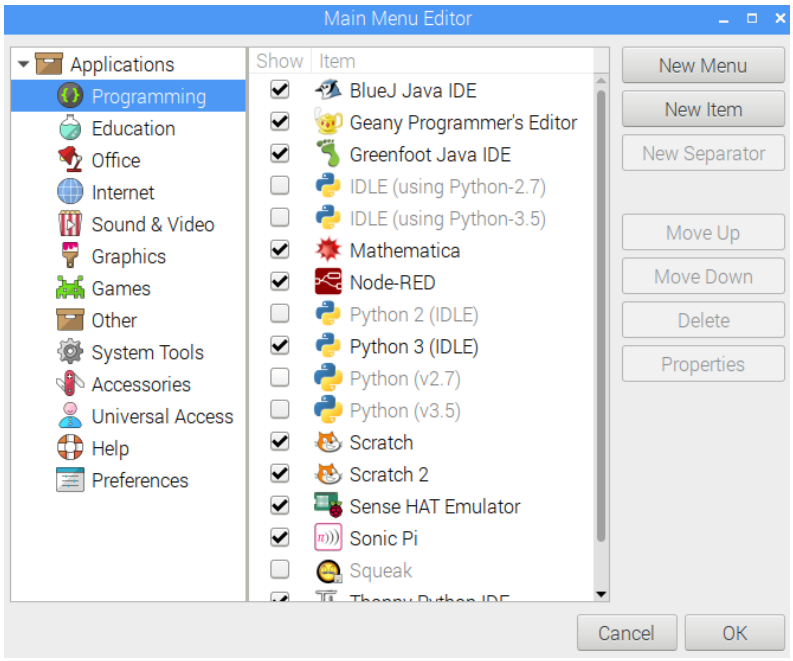
(i) **(ii)**

Σχήμα 3-25. Παράθυρα IDLE: (i) Γραμμής εντολών, (ii) Επεξεργασίας κειμένου.

Το λογισμικό αναπτύχθηκε στη γλώσσα προγραμματισμού Python 2.7.12 οπότε και χρησιμοποιείται η αντίστοιχη διανομή του περιβάλλοντος IDLE 2.7. Η διανομή αυτή, παρόλο που είναι ενσωματωμένη στο λειτουργικό σύστημα Raspbian, δεν εμφανίζεται στο μενού προγραμματισμού. Για την τοποθέτηση της στο κεντρικό μενού προγραμματισμού, ακολουθείται η πιο κάτω διαδικασία.

Από το κεντρικό μενού του Raspberry Pi, επιλέγεται η καρτέλα Preferences και στη συνέχεια η καρτέλα Main Menu Editor. Στο παράθυρο που εμφανίζεται επιλέγεται η καρτέλα Programming και ενεργοποιείται η επιλογή Python 2 (IDLE) (βλ. Σχήμα 3-26). Πλέον, η διανομή του περιβάλλοντος IDLE 2.7 είναι διαθέσιμη στο κεντρικό μενού προγραμματισμού.

Στην ενότητα που ακολουθεί, περιγράφεται η ανάπτυξη του λογισμικού, που είναι υπεύθυνο για την αποστολή δεδομένων από το Raspberry Pi στον κεντρικό υπολογιστή, μέσω σειριακής θύρας.



Σχήμα 3-26. Main menu editor Raspberry Pi.

3.4.3 Λογισμικό αποστολής δεδομένων μέσω σειριακής θύρας

Μέσα από το περιβάλλον ανάπτυξης IDLE δημιουργείται ένα νέο εκτελέσιμο αρχείο και αποθηκεύεται με το όνομα RasPi_Code.py. Το περιεχόμενο του αναλύεται λεπτομερώς στη συνέχεια.

```
1  #!/usr/bin/env python
2
3  import random
4  import decimal
5  import time
6  import serial
7
8  ser = serial.Serial(
9
10     port = '/dev/ttyAMA0',
11     baudrate = 115200,
12     parity = serial.PARITY_NONE,
13     stopbits = serial.STOPBITS_ONE,
14     bytesize = serial.EIGHTBITS,
15     timeout = 1
16 )
17
18 time.sleep(2)
19
20 counter = 0
21
22 while 1:
23     x = float(decimal.Decimal(random.randrange(0, 5010))/10)
24     y = float(decimal.Decimal(random.randrange(0, 5010))/10)
25     z = float(decimal.Decimal(random.randrange(0, 5010))/10)
26     thita = float(decimal.Decimal(random.randrange(0, 5010))/10)
27     u_x = float(decimal.Decimal(random.randrange(0, 5010))/10)
28     u_y = float(decimal.Decimal(random.randrange(0, 5010))/10)
29     string_buffer =
str(x)+","+str(y)+","+str(z)+','+str(thita)+','+str(u_x)+','+str(u_y)+'/'
+'\n'
```

```
30      print string_buffer
31      ser.write(string_buffer)
32      time.sleep(0,005)
33      counter += 1
34
35 ser.close()
```

Αρχικά, η εντολή:

```
#!/usr/bin/env python
```

χρησιμοποιείται για την περίπτωση που το αρχείο εκτελεστεί κατευθείαν από το τερματικό του Raspberry Pi. Εάν συμβεί αυτό, τότε το σύστημα θα αναγνωρίσει το αρχείο ως ένα αρχείο Python. Στη συνέχεια, ενεργοποιείται η σειριακή θύρα ttyAMA0 (γραμμές 8 – 16). Ο ρυθμός μετάδοσης δεδομένων ορίζεται 115200 bits/s και είναι ίσος με τον αντίστοιχο στον κόμβο camera_node, ο οποίος λαμβάνει τα δεδομένα από το Raspberry Pi. Στην εντολή αυτή δίνονται επίσης ως ορίσματα οι παράμετροι του byte δεδομένων που αποστέλλονται σειριακά, όπως το parity bit, το stopbit και το μέγεθος byte. Μετά την εντολή ενεργοποίησης, η εκτέλεση σταματά για 2 s για αποφυγή σφαλμάτων κατά την ενεργοποίηση της σειριακής θύρας. Για την ενεργοποίηση της σειριακής θύρας και την παύση του προγράμματος, είναι απαραίτητη η εισαγωγή των βιβλιοθηκών serial και time (γραμμές 5 – 6).

Ακολούθως, το σύστημα εξετάζει την συνθήκη της γραμμής 22. Αν η εκτέλεση του αρχείου δεν έχει διακοπεί, τότε το σύστημα εισέρχεται στο βρόχο (γραμμές 22 – 33). Πιο συγκεκριμένα, για την αναπαράσταση της πραγματικής θέσης και ταχύτητας του ρομποτικού ψαριού, παράγονται 6 τυχαίοι αριθμοί μεταξύ 0 και 501 με ένα δεκαδικό ψηφίο, και αποθηκεύονται σε έξι επιμέρους μεταβλητές (γραμμές 23 – 28). Για την παραγωγή των τυχαίων δεκαδικών αριθμών είναι απαραίτητη η εισαγωγή των βιβλιοθηκών random και decimal (γραμμές 3 – 4). Οι μεταβλητές αυτές, αφού μετατραπούν από δεδομένα τύπου float σε χαρακτήρες ASCII, ενοποιούνται μεταξύ τους, σχηματίζοντας την γραμμή κειμένου της μορφής που περιγράφηκε σε προηγούμενη ενότητα. Η γραμμή αυτή αποθηκεύεται στην μεταβλητή string_buffer (γραμμη 29) και αποστέλλεται μέσω της σειριακής θύρας (γραμμή 31). Μεταξύ κάθε αποστολής μεσολαβεί διάστημα 5 ms (γραμμή 32).

Με το συνδυασμό των πλήκτρων:

```
CTRL + C
```

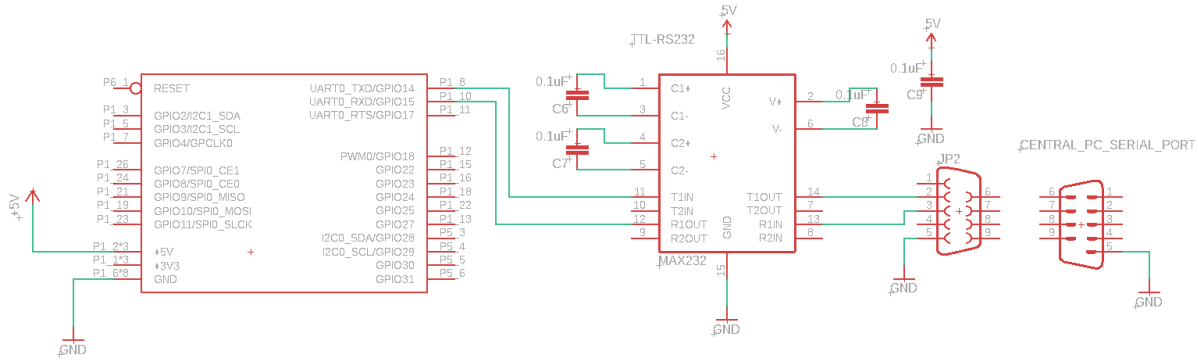
το σύστημα εξέρχεται του βρόχου, και η σειριακή θύρα απενεργοποιείται.

3.5 Συνδεσμολογία Κεντρικής Μονάδας Ελέγχου

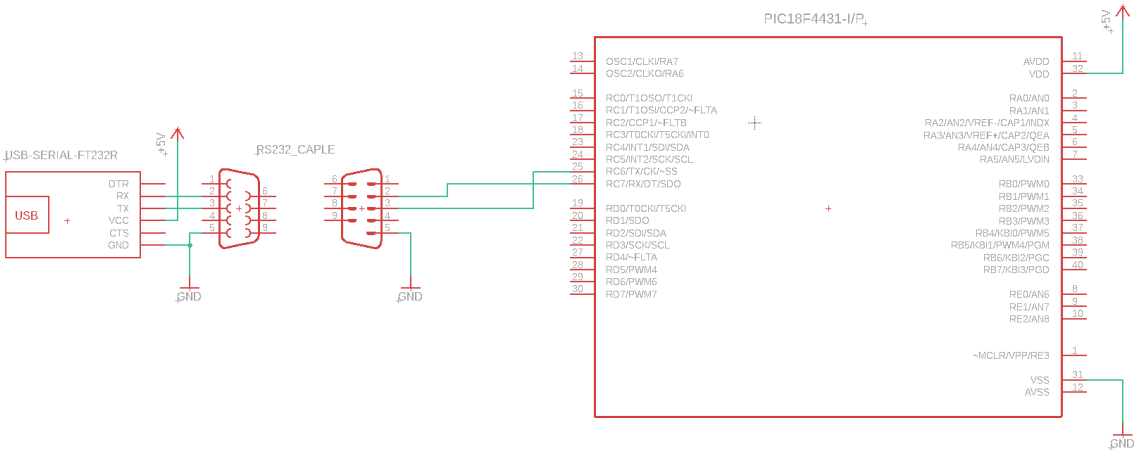
Η υλοποίηση της κεντρικής μονάδας ελέγχου ολοκληρώνεται με τη σύνδεση των επιμέρους υποσυστημάτων τεχνολογικού εξοπλισμού. Στο Σχήμα 3-27-i, παρουσιάζεται η σύνδεση του μικροϋπολογιστή Raspberry Pi με τη σειριακή θύρα του κεντρικού υπολογιστή. Πιο συγκεκριμένα, οι ακροδέκτες του υποσυστήματος UART του μικροϋπολογιστή (ακροδέκτες γενικής χρήσης 14 και 15) συνδέονται με τους αντίστοιχους ακροδέκτες του μετασχηματιστή RS-232 – TTL (ακροδέκτες 10 και

12). Στην αντίθετη πλευρά, οι ακροδέκτες τύπου DB9 (9 ακροδέκτες του πρωτοκόλλου RS-232) του μετασχηματιστή συνδέονται με το καλώδιο RS-232, που καταλήγει στην σειριακή θύρα του κεντρικού υπολογιστή.

Στο Σχήμα 3-27-ii, παρουσιάζεται η σύνδεση του κεντρικού υπολογιστή με τον μικροελεγκτή PIC18F4431 του ρομποτικού ψαριού. Σε πρώτο στάδιο, ο μετατροπέας USB – RS-232 συνδέεται σε μία από τις θύρες USB του κεντρικού υπολογιστή. Στη συνέχεια, οι ακροδέκτες τύπου DB9 του μετατροπέα συνδέονται στο ένα άκρο ενός καλωδίου RS-232. Τέλος, οι ακροδέκτες του υποσυστήματος EUSART του μικροελεγκτή (ακροδέκτες 25 και 26) συνδέονται στο άλλο άκρο του καλωδίου RS-232.



(i)



(ii)

Σχήμα 3-27. Σύνδεση υποσυστημάτων τεχνολογικού εξοπλισμού: (i) Raspberry Pi – Κεντρικός υπολογιστής, (ii) Κεντρικός υπολογιστής – Μικροεπεξεργαστής ρομποτικού ψαριού.

Αφού ολοκληρωθεί η συνδεσμολογία των δύο υποσυστημάτων με την κεντρική μονάδα ελέγχου, το συνολικό σύστημα είναι έτοιμο για δοκιμή, στην πειραματική διάταξη που παρουσιάζεται σε ξεχωριστή ενότητα στη συνέχεια.

4 Σχεδιασμός και Κατασκευή Βάσης 3ου Άξονα

4.1 Εισαγωγή

Ο δεύτερος στόχος της διπλωματικής εργασίας αυτής, είναι ο σχεδιασμός και η κατασκευή μίας οριζόντιας πλατφόρμας τριών βαθμών ελευθερίας, η οποία προσαρμόζεται στον μηχανισμό φορείου ρυμούλκησης της δεξαμενής πειραματικών δοκιμών του Εργαστηρίου Αυτομάτου Ελέγχου. Η πλατφόρμα έχει την δυνατότητα στήριξης ηλεκτρικού κινητήρα, που είναι υπεύθυνος για την περιστροφική κίνηση της ατράκτου που συνδέει το ρομποτικό ψάρι, κατά τον κατακόρυφο άξονα z (άξονα κάθετο στην επιφάνεια του νερού).

Η αρχική πλατφόρμα αποτελείτο από μία πρόχειρη κατασκευή από plexiglass και εξαρτήματα αλουμινίου, όπως παρουσιάζεται στο Σχήμα 4-2. Η άτρακτος του ρομποτικού ψαριού ήταν πακτωμένη στην πλατφόρμα από plexiglass, οπότε κάθε μεταβολή στην κατεύθυνση του ρομποτικού ψαριού απαιτούσε αποσύνδεση και επανασύνδεση του ρομπότ στην άτρακτο. Η διαδικασία αυτή ήταν χρονοβόρα και η διεξαγωγή πειραμάτων με μεταβλητή γωνία του ρομποτικού ψαριού δεν ήταν εφικτή. Λαμβάνοντας αυτά υπόψη, κρίθηκε απαραίτητη η αντικατάσταση της υπάρχουσας πλατφόρμας με μία νέα, η οποία πληροί τα κριτήρια σχεδιασμού που περιγράφονται πιο κάτω.

Στο παρόν κεφάλαιο, περιγράφεται η διαδικασία σχεδιασμού και κατασκευής της βάσης. Αρχικά, παρουσιάζονται οι προδιαγραφές σχεδιασμού της βάσης αλλά και οι παράγοντες που οδήγησαν σε αυτές. Στη συνέχεια, περιγράφονται συνοπτικά οι υποψήφιες σχεδιαστικές λύσεις που εξετάστηκαν και οι λόγοι που οδήγησαν στην απόρριψη τους, μέχρι την επιλογή της τελικής διαμόρφωσης προς κατασκευή. Ο σχεδιασμός των κατασκευαστικών σχεδίων υλοποιήθηκε στο λογισμικό σχεδίασης SolidWORKS. Ακολούθως, περιγράφεται η επιλογή των μηχανικών συνιστωσών της βάσης. Τέλος, παρουσιάζεται η διαδικασία κατασκευής των επιμέρους συνιστωσών, με παράλληλη αναφορά στον μηχανολογικό εξοπλισμό και στο λογισμικό που χρησιμοποιήθηκε κατά την διάρκειά της, μέχρι την συναρμολόγηση της τελικής κατασκευής.

4.2 Καθορισμός ελάχιστων προδιαγραφών μηχανικής υλοποίησης βάσης

Λαμβάνοντας υπόψη τις απαιτήσεις της συγκεκριμένης εφαρμογής, το περιβάλλον λειτουργίας της δεξαμενής πειραματικών δοκιμών του εργαστηρίου αυτομάτου ελέγχου καθώς και την υπάρχουσα μηχανολογική εγκατάσταση, καθορίστηκε το πλαίσιο προδιαγραφών της προς σχεδιασμό βάσης στήριξης.

Λειτουργικές απαιτήσεις

- **Βαθμοί ελευθερίας (DoF):** Δύο βαθμοί ελευθερίας είναι απαραίτητοι για την μεταφορική κίνηση της βάσης στο οριζόντιο επίπεδο, κατά τους άξονες x και y. Ένας τρίτος βαθμός ελευθερίας θα επιτρέπει την περιστροφική κίνηση του άξονα του κινητήρα που θα μεταφέρεται από τη βάση, γύρω από τον κατακόρυφο άξονα z (yaw). Η επιλογή αυτή

χαρακτηρίζεται από σχετικά μειωμένες κατασκευαστικές απαιτήσεις, ενώ η ελάχιστη χρήση επενεργητών (μόνο για τον τρίτο βαθμό ελευθερίας) οδηγεί σε υλοποιήσεις μειωμένου βάρους.

- **Αυτοτέλεια μηχανισμού (modular design):** Η υλοποίηση της βάσης θα πρέπει να είναι μία αυτοτελή δομική μονάδα που μπορεί να αφαιρείται εύκολα και να δοκιμάζεται ανεξάρτητα σε οποιαδήποτε άλλη εφαρμογή που απαιτεί επιπλέον άξονα περιστροφής.
- **Εύρος κίνησης** (εσωτερικές διαστάσεις δεξαμενής πειραματικών δοκιμών): Οι εξωτερικές διαστάσεις της βάσης θα πρέπει να τις επιτρέπουν να προσαρμόζεται με ευκολία στο υπάρχον σύστημα κίνησης (μηχανισμός φορείου ρυμούλκησης) και παράλληλα, το σημείο του κατακόρυφου άξονα περιστροφής να καλύπτει τη μεγαλύτερη δυνατή επιφάνεια του εσωτερικού τις δεξαμενής.
- **Ενσωμάτωση στο υπάρχον σύστημα κίνησης** (μηχανισμός φορείου ρυμούλκησης): Η κατασκευή της βάσης θα πρέπει να αξιοποιεί το υπάρχον σύστημα μετάδοσης κίνησης, για την μεταφορική κίνηση κατά το οριζόντιο επίπεδο xy (παράλληλο στην επιφάνεια του νερού), που χρησιμοποιεί το φορείο ρυμούλκησης της δεξαμενής (γραμμικά έδρανα ολίσθησης, μετάδοση κίνησης με συνεργασία πλαστικής αλυσίδας – οδοντωτών τροχών).
- **Προσαρμογή ηλεκτρικού κινητήρα:** η βάση πρέπει να στηρίζει τον ηλεκτρικό κινητήρα Maxon RE 25 10 Watt συνδεδεμένο έναν Maxon Encoder MR 225780, με σκοπό την παροχή περιστροφικής κίνησης κατά το κατακόρυφο άξονα z.
- **Συναρμολόγηση:** Η βάση θα πρέπει να συναρμολογείται και να αποσυναρμολογείται εύκολα, χωρίς χρήση εξειδικευμένων εργαλείων, απαίτηση που θα διευκολύνει τη συντήρηση και τη πιθανή αλλαγή εξαρτημάτων.

Κατασκευαστικές απαιτήσεις

- **Μειωμένο βάρος:** Η ελαφρότητα του μηχανισμού είναι σημείο κλειδί για τη μείωση των αδρανειακών δυνάμεων που ασκούνται στο συνολικό σύστημα και επηρεάζουν την ακρίβεια θέσης και κίνησης. Ο στόχος αυτός μπορεί να επιτευχθεί ποικιλοτρόπως με:
    - Χρήση ελαφρών υλικών υψηλής αντοχής για την κατασκευή των εξαρτημάτων. Χαρακτηριστικές περιπτώσεις αποτελούν το αλουμίνιο 2006 και ακρυλικά υλικά όπως plexiglass.
    - Σχεδίαση δομών με ενισχυμένη γεωμετρική ακαμψία και αντοχή στις διευθύνσεις με τις μεγαλύτερες φορτίσεις. Αυτό εντάσσεται στην γενικότερη προσπάθεια για καλύτερο βαθμό εκμεταλλεύσεως του υλικού με αποδοτικότερη κατανομή των φορτίσεων και αποφυγή συγκέντρωσης τάσεων.
    - Ελαχιστοποίηση του πλήθους των εξαρτημάτων και των μεταξύ τους κοχλιοσυνδέσεων. Αυτό βέβαια αντιβαίνει το τροποποιήσιμο της διάταξης.
- **Προστασία από διάβρωση:** Τα εξαρτήματα που θα χρησιμοποιηθούν για την κατασκευή της βάσης πρέπει να αποτελούνται από ανθεκτικά στο διαβρωτικό υγρό περιβάλλον της δεξαμενής υλικά.
- **Μείωση των ανοχών στα κινούμενα μέρη:** αυτό αποτελεί κριτήριο μιας μηχανικά ποιοτικής ελεγχόμενης διάταξης καθώς οι τζόγοι στις αρθρώσεις αυξάνουν το σφάλμα μετατόπισης που

τελικώς δυσχεραίνει την επίτευξη στιβαρότητας και υψηλής ακρίβειας θέσης. Η αντιμετώπιση αυτού του σφάλματος γίνεται π.χ. με χρήση εμπορικά διαθέσιμων εξαρτημάτων πιστοποιημένων ανοχών (π.χ. ρουλεμάν) ή με σφικτές συναρμογές που επιτυγχάνονται κατά την κατασκευή των εκάστοτε εξαρτημάτων του μηχανουργείου του Εργαστηρίου Αυτομάτου Ελέγχου.

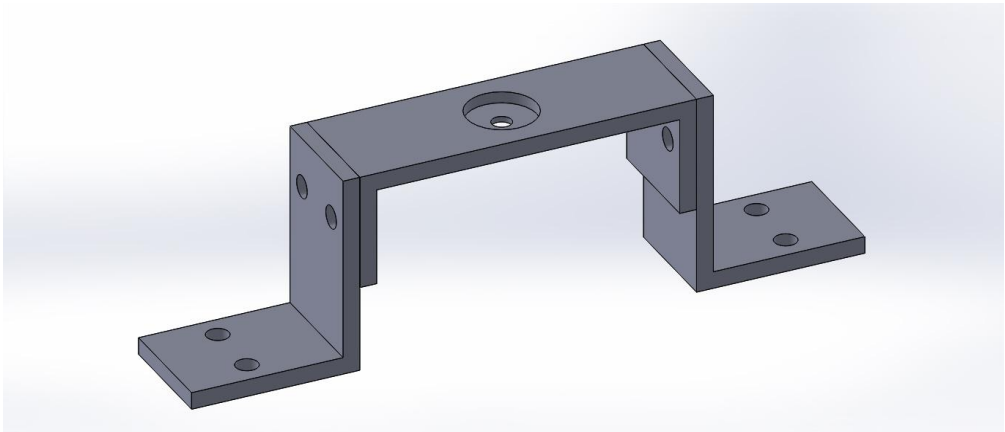
- **Αντοχή σε καμπτικά φορτία:** η κατασκευή της βάσης πρέπει να αντέχει τα καμπτικά φορτία που προέρχονται από την υδροδυναμική αντίσταση κατά την κίνηση του ψαριού (καμπτική ροπή κατά την διάρκεια της κίνησης με μέγιστη ταχύτητα: 3,24 Nm [25]

Στις γενικές προδιαγραφές σχεδιασμού περιλαμβάνεται επίσης το χαμηλό κόστος κατασκευής, η δυνατότητα υλοποίησης της σε μεγάλο βαθμό από το μηχανουργείο του Εργαστήριο Αυτομάτου Ελέγχου και η αποφυγή ατυχημάτων στο ανθρώπινο δυναμικό.

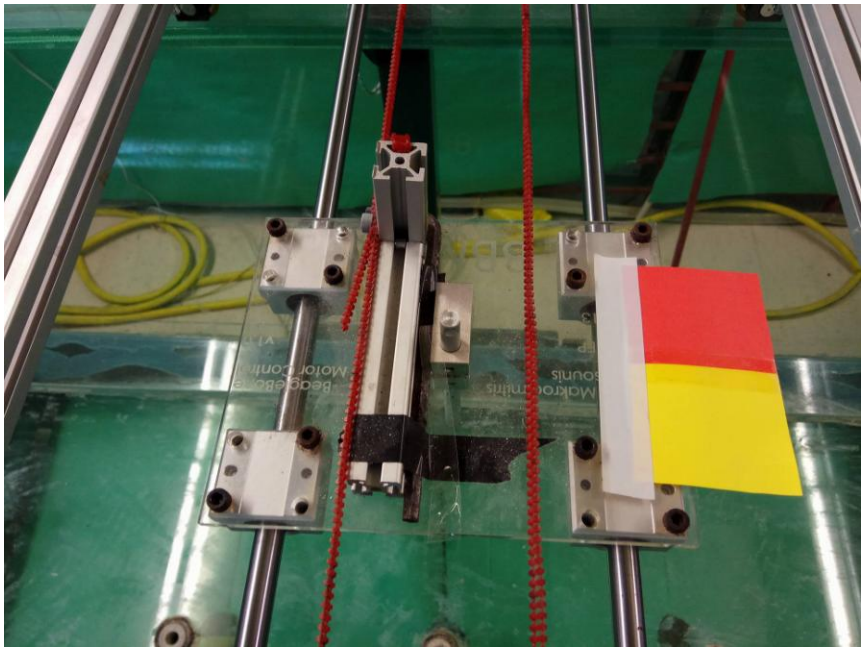
4.3 Σύγκριση κατασκευαστικών υλοποιήσεων

Αρχικά, εξετάστηκαν διάφορες πιθανές λύσεις συναρμολογημάτων, από τις οποίες επιλέχθηκε η καταλληλότερη, απορρίπτοντας αυτές που δεν πληρούσαν τις απαιτήσεις σχεδιασμού. Ακολούθως, παρουσιάζονται οι κυριότερες από αυτές, καθώς επίσης και οι λόγοι που οδήγησαν στην απόρριψη τους, καταλήγοντας στην τελική διαμόρφωση προς κατασκευή.

Η πρώτη σχεδιαστική λύση (βλ. Σχήμα 4-1) περιελάμβανε συναρμολόγημα αποτελούμενο από τρία μεταλλικά τεμάχια (δύο σχήματος Γ και ένα σχήματος Π) . Τα τεμάχια αυτά συνδέονταν μεταξύ τους με κοχλιοσύνδεση, σχηματίζοντας βάση σχήματος Π, πάνω στην οποία θα στηριζόταν ο ηλεκτρικός κινητήρας. Ως κύρια βάση της πλατφόρμας παρέμενε η προηγούμενη κατασκευή από plexiglass, στην οποία ήταν ήδη προσαρμοσμένα τα γραμμικά έδρανα ολίσθησης του φορείου ρυμούλκησης (βλ. Σχήμα 4-2). Η συγκεκριμένη λύση όμως, καθιστούσε δύσκολη την επίτευξη πλήρους ευθυγράμμισης του άξονα του κινητήρα με τον κατακόρυφο άξονα z, λόγω της πληθώρας διαφορετικών τεμαχίων που έπρεπε να σταθεροποιηθούν. Επιπρόσθετα, η χρήση κοχλιών για την σύνδεση των εξαρτημάτων αύξανε κατά πολύ το συνολικό βάρος κατασκευής. Οι κατεργασίες μορφοποίησης στις οποίες είχε υποβληθεί η υπάρχουσα βάση από plexiglass καθιστούσαν δύσκολη την επίτευξη της επιθυμητής γεωμετρίας στην καινούργια βάση. Λαμβάνοντας υπόψη τους πιο πάνω λόγους, η πρώτη σχεδιαστική λύση κρίθηκε ακατάλληλη.

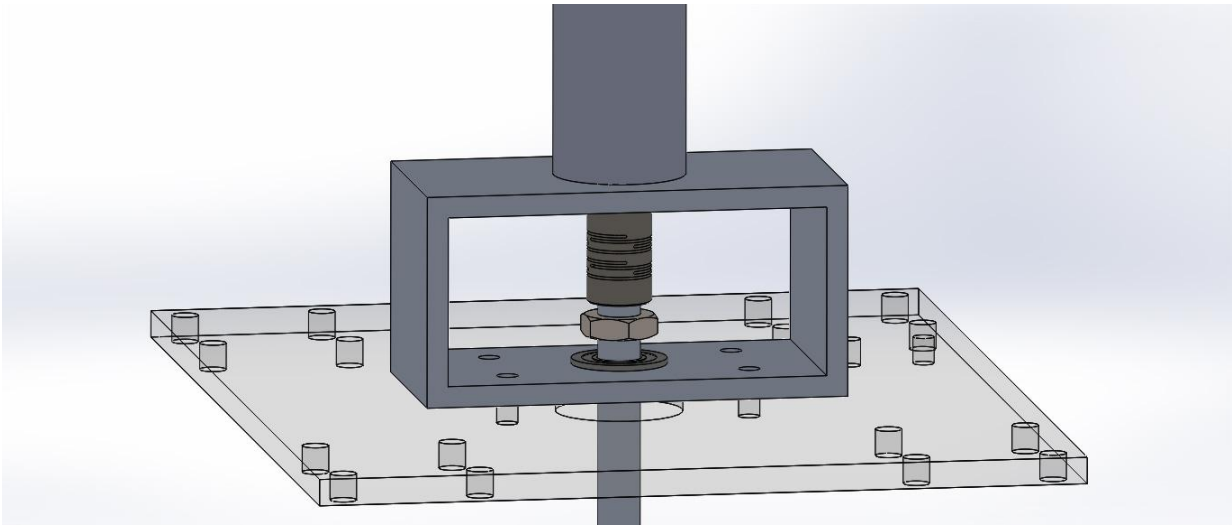


Σχήμα 4-1. Πρώτη σχεδιαστική λύση



Σχήμα 4-2. Προηγούμενη κατασκευή βάσης στο φορείο ρυμούλκησης

Η δεύτερη σχεδιαστική λύση (βλ. Σχήμα 4-3), προέβλεπε μία ενιαία σωλήνα ορθογωνικής διατομής (αντί τριών ξεχωριστών τεμαχίων), διαμορφωμένη κατάλληλα ώστε να στηρίζει τον ηλεκτρικό κινητήρα στο πάνω μέρος της. Η άτρακτος του ρομποτικού ψαριού περνούσε μέσα από ένα έδρανο κύλισης στο κάτω μέρος της βάσης και συνδεόταν με την άτρακτο του κινητήρα μέσω ενός ελαστικού συνδέσμου (κόπλερ). Η ορθογωνική σωλήνα στηριζόταν με χρήση τεσσάρων κοχλιών στην βάση από πλεξιγκλάς (plexiglass), η οποία θα κατασκευαζόταν εκ νέου για επίτευξη της επιθυμητής γεωμετρίας (θέση οπών για κοχλίες). Η συγκράτηση του εσωτερικού δακτυλίου του εδράνου κύλισης με την άτρακτο του ρομποτικού ψαριού επιτυγχάνεται με την χρήση περικοχλίου, με κατάλληλη διαμόρφωση σπειρώματος στην άτρακτο του ρομποτικού ψαριού.

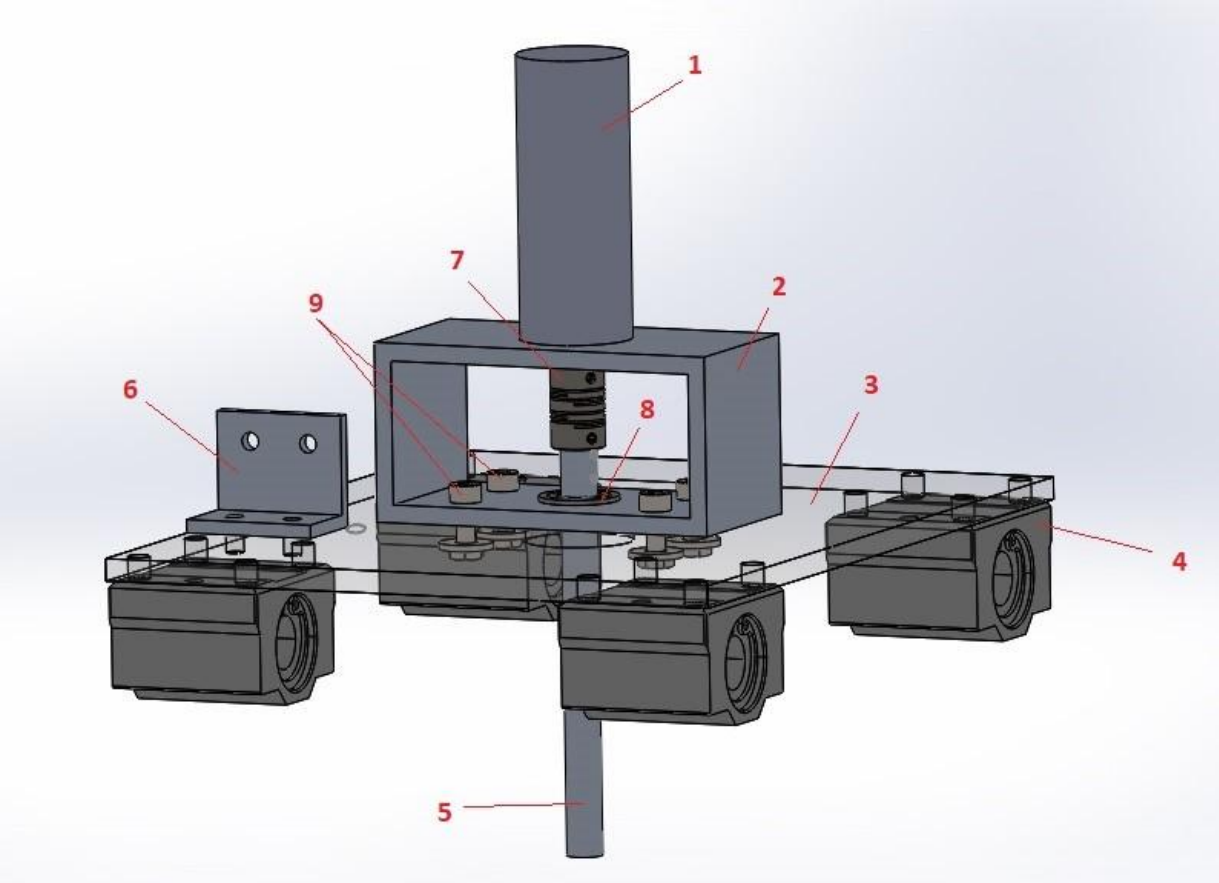


Σχήμα 4-3. Δεύτερη σχεδιαστική λύση

Η λύση αυτή, αν και έδινε απαντήσεις σε όλα τα ερωτήματα που έθετε η προηγούμενη, χαρακτηριζόταν από σχετικά μεγάλο βαθμό πολυπλοκότητας, τόσο για την κατασκευή των επιμέρους εξαρτημάτων, όσο και για την διαδικασία συναρμολόγησης. Με στόχο λοιπόν, την εκπλήρωση των προδιαγραφών σχεδιασμού με μία απλούστερη λύση, εφαρμόστηκαν οι κατάλληλες αλλαγές, που οδήγησαν στην τελική διαμόρφωση όπως περιγράφεται στη συνέχεια.

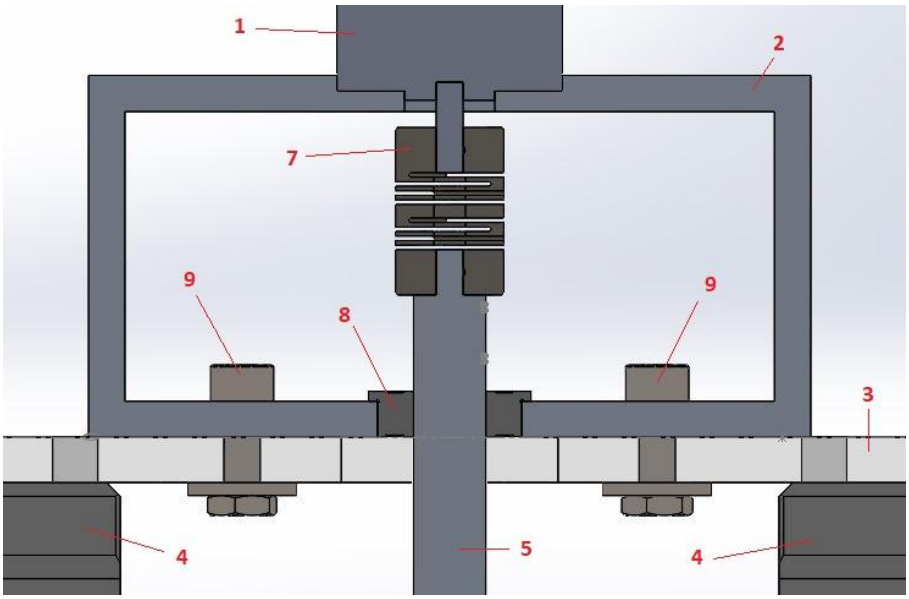
4.4 Τελική Διαμόρφωση

Αφού εξετάστηκαν όλες οι πιθανές σχεδιαστικές λύσεις και απορρίφθηκαν αυτές που δεν πληρούσαν τις προδιαγραφές σχεδιασμού, επιλέχθηκε η τελική λύση προς κατασκευή, που παρουσιάζεται στο Σχήμα 4-4 και στο Σχήμα 4-5 σε τομή.



Σχήμα 4-4. Τελική διαμόρφωση.

Η τελική διαμόρφωση παρουσιάζει αρκετές ομοιότητες με την δεύτερη σχεδιαστική λύση, με κάποιες διαφοροποιήσεις για απλοποίηση της κατασκευής. Το κυρίως σώμα της αποτελείται από σωλήνα ορθογωνικού προφίλ (2). Ο σωλήνας είναι διαμορφωμένος στο πάνω μέρος του ώστε να μπορεί να στηρίζει τον ηλεκτρικό κινητήρα (1), ενώ στην κάτω πλευρά, διαθέτει οπή, για τοποθέτηση εδράνου κύλισης (8). Η άτρακτος του ρομποτικού ψαριού (5) εισέρχεται από την κάτω πλευρά του σωλήνα και συνδέεται με την άτρακτο του ηλεκτρικού κινητήρα μέσω ελαστικού συνδέσμου (7), ο οποίος είναι υπεύθυνος για τη μετάδοση της περιστροφικής κίνησης.



Σχήμα 4-5. Τελική διαμόρφωση (τομή).

Ο σωλήνας είναι συνδεδεμένος πάνω σε ορθογωνική βάση από plexiglass (3) μέσω τεσσάρων κοχλιών (9). Η ορθογωνική βάση συνδέεται με τα τέσσερα γραμμικά έδρανα κύλισης (4) επίσης μέσω κοχλιών. Τα έδρανα κύλισης αυτά, αποτελούν μέρος του μηχανισμού φορείου ρυμούλκησης και παρέχουν δυνατότητα μεταφορικής κίνησης χωρίς τριβές κατά τον οριζόντιο άξονα y (πλάτος δεξαμενής). Η μετάδοση της κίνησης αυτής γίνεται μέσω συνεργασίας οδοντωτών τροχών και πλαστικής αλυσίδας, όπως περιγράφεται στο [25]. Η πλαστική αλυσίδα συνδέεται στο (6) μέσω κοχλιών, ώστε μαζί της να κινείται και ολόκληρη η βάση του 3^{ου} άξονα.

4.4.1 Περιγραφή Μηχανικών Συνιστωσών Τελικής Διαμόρφωσης

Στη συνέχεια, περιγράφονται συνοπτικά τα επιμέρους εξαρτήματα της βάσης 3^{ου} άξονα, με αναφορά στα υλικά και στις μεθόδους κατασκευής τους, καθώς επίσης και στους παράγοντες που οδήγησαν στις επιλογές αυτές.

Ηλεκτρικός Κινητήρας Maxon RE 25 Motor

Ο ηλεκτρικός κινητήρας που στηρίζει η βάση του 3^{ου} άξονα είναι ένας κινητήρας συνεχούς ρεύματος, με διάμετρο 25 mm, ισχύ 10 Watt και με ενσωματωμένο κωδικοποιητή (encoder) της εταιρίας “Maxon Motors” (βλ. Σχήμα 4-6). Τα λεπτομερή τεχνικά χαρακτηριστικά του κινητήρα και του encoder παρουσιάζονται στο Παράρτημα της παρούσας διπλωματικής εργασίας [48].

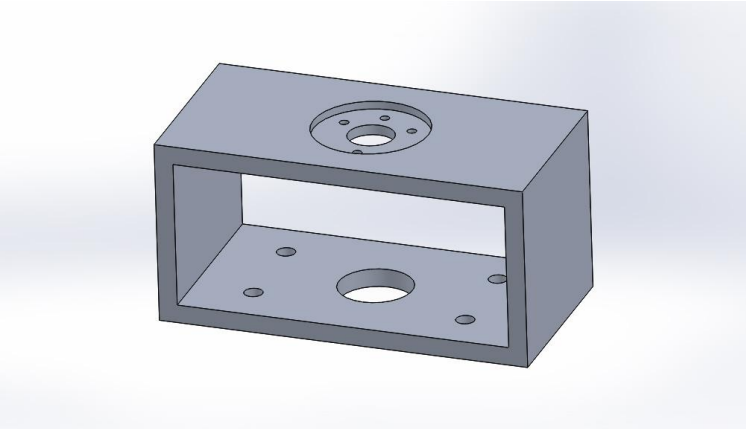
Σχήμα 4-6. Ηλεκτρικός Κινητήρας Maxon RE 25 Motor.

Βάση από Σωλήνα Ορθογωνικού Προφίλ

Η βάση από σωλήνα ορθογωνικού προφίλ (βλ. Σχήμα 4-7), αποτελεί το κυρίως σώμα της βάσης 3^{ου} άξονα. Οι εξωτερικές της διαστάσεις είναι 80x40x40 mm και τα τοιχώματα της έχουν ομοιόμορφο πάχος 4 mm. Στην πάνω πλευρά της διαθέτει ειδική κυκλική διαμόρφωση διαμέτρου 25 mm για την υποδοχή του ηλεκτρικού κινητήρα, μία κεντρική οπή διαμέτρου 10 mm για την άτρακτο του κινητήρα αλλά και τρεις οπές διαμέτρου 2 mm για τους κοχλίες σύνδεσης του. Στην κάτω πλευρά της, διαθέτει μία κεντρική οπή διαμέτρου 16 mm για την εισαγωγή του εδράνου κύλισης και τέσσερεις μικρότερες, συμμετρικές οπές διαμέτρου 4 mm, για την σύνδεση του σωλήνα με την ορθογωνική βάση της πλατφόρμας.

Η προδιαγραφή για όσο το δυνατόν πιο μικρό συνολικό βάρος κατασκευής δημιουργεί την απαίτηση για επιλογή ελαφρών υλικών με μικρή πυκνότητα. Η πληθώρα μηχανουργικών κατεργασιών που απαιτούνται να πραγματοποιηθούν στο σωλήνα και η ανάγκη μόνιμης στήριξης του βάρους του κινητήρα (ή και του ρομποτικού ψαριού) καθιστούν απαραίτητη την επιλογή εύκολα κατεργάσιμου και στιβαρού υλικού. Λαμβάνοντας υπόψη τα πιο πάνω, ως υλικό κατασκευής του εξαρτήματος αυτού επιλέγεται το αλουμίνιο, το οποίο χαρακτηρίζεται από σχετικά χαμηλή πυκνότητα (2810 g/cm^3), υψηλή κατεργασιμότητα και ικανοποιητική αντοχή σε κάμψη (για το μέγιστο φορτίο της συγκεκριμένης εφαρμογής) [19].

Υψηλής σημασίας προδιαγραφή για το συγκεκριμένο εξάρτημα αποτελεί η πλήρης ευθυγράμμιση των δύο κεντρικών οπών στο πάνω και κάτω μέρος του. Οποιαδήποτε εκκεντρότητα μεταξύ των δύο οπών αυτών θα οδηγήσει σε εκκεντρότητα μεταξύ ατράκτου του κινητήρα και ατράκτου του ρομποτικού ψαριού. Αυτό θα έχει ως αποτέλεσμα την καταπόνηση του κινητήρα και του εδράνου κύλισης από φορτία που θα αναπτύσσονται λόγω έκκεντρων μαζών, παρά την χρήση του εύκαμπτου φορέα. Η χρήση εδράνου κύλισης καθιστά απαραίτητη την επίτευξη διάστασης υψηλής ακρίβειας και επιφάνειας χαμηλής τραχύτητας (εσωτερική επιφάνεια της κάτω κεντρικής οπής), με σκοπό της επίτευξη της επιθυμητής συναρμογής. Οι απαιτήσεις αυτές έκριναν ακατάλληλη την χρήση των συμβατικών εργαλειομηχανών (τόρνος, φρέζα, τρυπάνι), λόγω της χαμηλής σχετικά ακρίβειας που μπορούν να πετύχουν. Ως εκ τούτου, η βάση από σωλήνα ορθογωνικού προφίλ αποφασίστηκε να κατασκευαστεί στο αυτόματο κέντρο κατεργασιών HAAS Mini Mill, στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου.

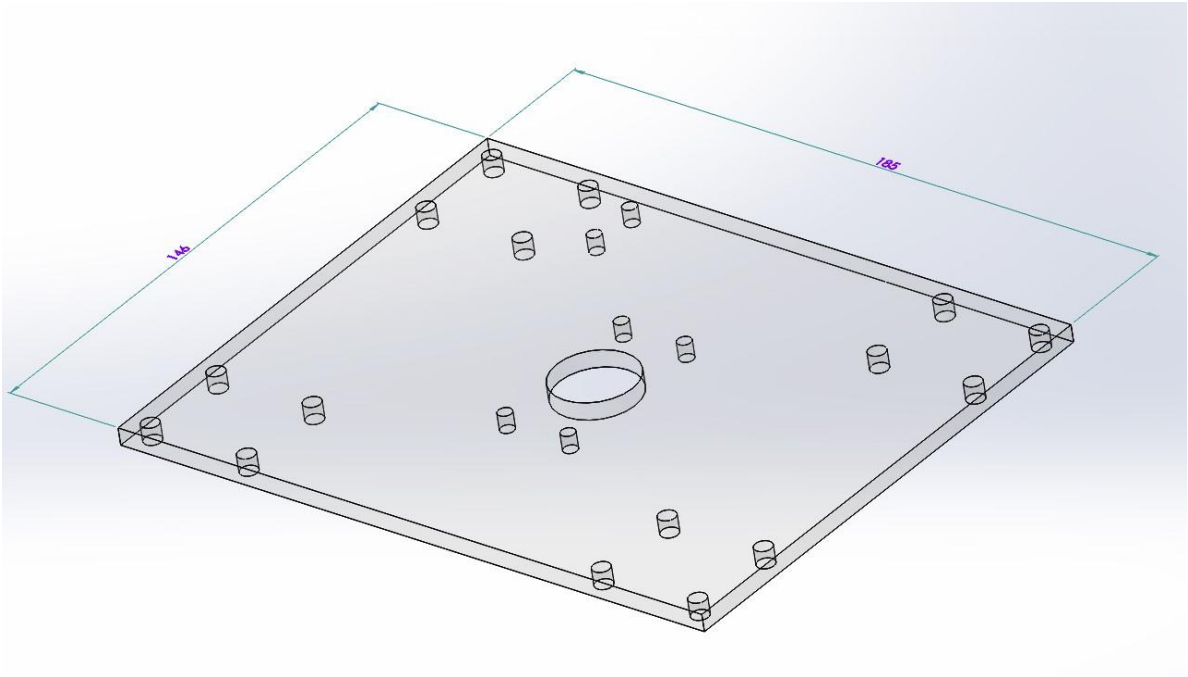


Σχήμα 4-7. Βάση από σωλήνα ορθογωνικού προφίλ.

Ορθογωνική Βάση από Plexiglass

Η πληθώρα κατεργασιών στις οποίες είχε υποβληθεί η βάση της προηγούμενης κατασκευής, και η φθορά των ακμών της με το πέρασμα του χρόνου, κατέστησαν δύσκολη την επαναχρησιμοποίηση της για την παρούσα εφαρμογή. Για τους λόγους αυτούς, κρίθηκε απαραίτητη η κατασκευή μίας εξ ολοκλήρου καινούριας βάσης (βλ. Σχήμα 4-8). Η σχεδίαση της βάσης υλοποιήθηκε με τη χρήση του λογισμικού σχεδίασης SolidWORKS. Οι εξωτερικές διαστάσεις της είναι 185 x 146 x 5 mm, που επιτρέπουν στην κατακόρυφη άτρακτο του ρομποτικού ψαριού να καλύπτει όσο το δυνατόν μεγαλύτερη επιφάνεια της δεξαμενής πειραματικών δοκιμών, κατά την κίνηση του φορείου ρυμούλκησης στο επίπεδο x, y. Η βάση διαθέτει μία κεντρική οπή διαμέτρου 24 mm από την οποία περνάει η κατακόρυφη άτρακτος του ρομποτικού ψαριού. Δίπλα από την οπή αυτή έχουν σχεδιαστεί τέσσερεις μικρότερες οπές διαμέτρου 4 mm για σύνδεση της βάσης με τον σωλήνα ορθογωνικού προφίλ. Περιμετρικά της βάσης υπάρχουν τέσσερεις τετράδες οπών διαμέτρου 5 mm για προσαρμογή της βάσης στα γραμμικά έδρανα κύλισης του φορείου ρυμούλκησης. Τέλος, ένα ζεύγος οπών διαμέτρου 2 mm έχουν σχεδιαστεί με σκοπό της σύνδεση του εξαρτήματος στο οποίο θα στερεώνεται η πλαστική αλυσίδα του φορείου.

Ως υλικό κατασκευής της καινούργιας βάσης επιλέχθηκε να παραμείνει το Plexiglass, λόγω των θετικών χαρακτηριστικών που παρουσιάζει σε φορτίσεις λυγισμού. Λόγω απουσίας εξειδικευμένου μηχανήματος κοπής με Laser στο Μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου, το συγκεκριμένο εξάρτημα αποφασίστηκε να αποσταλεί για κατασκευή σε εξωτερικό μηχανουργείο.



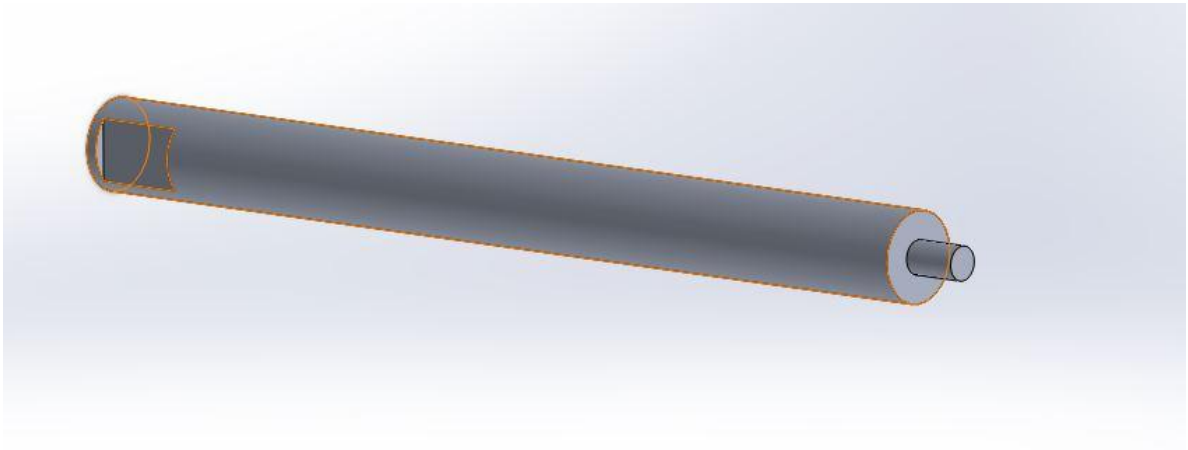
Σχήμα 4-8. Ορθογωνική βάση από Plexiglass

Άτρακτος Ρομποτικού ψαριού

Η άτρακτος σύνδεσης του ρομποτικού ψαριού με το σύστημα περιστροφής αποτελείται από μία συμπαγή ράβδο κυκλικής διατομής, διαμέτρου 8 mm και συνολικού μήκους 96 mm (βλ. Σχήμα 4-9).

Στο ένα της άκρο έχει σχεδιαστεί διαμόρφωση μήκους 5 mm και διαμέτρου 3 mm, ώστε να προσαρμόζεται μέσα στον ελαστικό σύνδεσμο (κόπλερ) που θα μεταφέρει την περιστροφική κίνηση από τον ηλεκτρικό κινητήρα. Το άλλο άκρο της ατράκτου, διαθέτει ορθογωνική εσοχή, η οποία διευκολύνει την σύνδεση ράβδου μεγαλύτερου μήκους για επέκταση της ατράκτου που καταλήγει στο ρομποτικό ψάρι, ή την σύνδεση αισθητήρα ροπής για την ολοκλήρωση πιθανών πειραματικών δοκιμών.

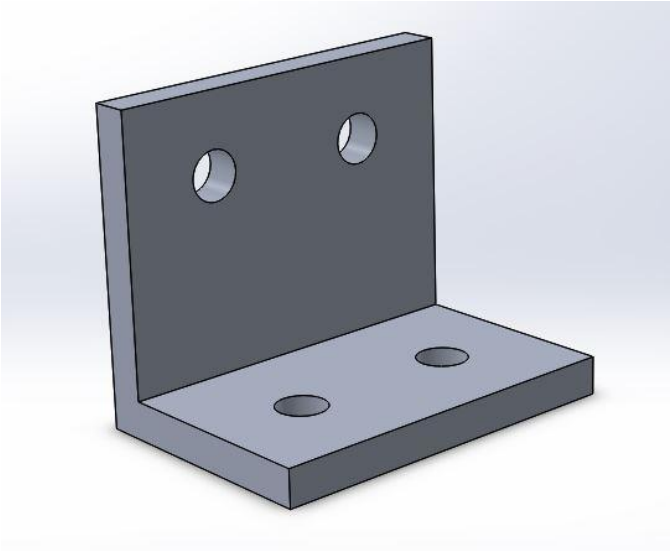
Ως υλικό κατασκευής επιλέχθηκε ράβδος αλουμινίου 8 mm, από την αποθήκη υλικών του Εργαστηρίου Αυτομάτου Ελέγχου. Τα σχετικά χαμηλά φορτία που ασκούνται στο εξάρτημα αυτό κατά την λειτουργία της βάσης 3^{ου} άξονα, θέτουν χαμηλές απαιτήσεις αντοχής υλικού. Λαμβάνοντας υπόψη αυτό, η ράβδος αλουμινίου διαμέτρου 8 mm αποτελεί ικανοποιητική λύση για την εφαρμογή αυτή. Η απλή γεωμετρία της ατράκτου του ρομποτικού ψαριού επέτρεψε την κατασκευή της με την χρήση συμβατικών εργαλειομηχανών (τόρνου, φρέζας), στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου.



Σχήμα 4-9. Άτρακτος σύνδεσης Ρομποτικού Ψαριού.

Εξάρτημα Σύνδεσης Πλαστικής Αλυσίδας

Το εξάρτημα αυτό αποτελεί την απλούστερη δυνατή λύση για μετάδοση της γραμμικής κίνησης της πλαστικής αλυσίδας στην βάση 3^{ου} άξονα. Το μικρό του μέγεθος και η απλή του γεωμετρία δεν επιβαρύνουν το τελικό συναρμολόγημα. Αποτελείται από μία ράβδο διατομής τύπου Γ συνολικού μήκους 30 mm, ενιαίου πάχους 3 mm και διαστάσεων διατομής 25 x 20 mm (βλ. Σχήμα 4-10). Διαθέτει τέσσερεις οπές διαμέτρου 4 mm, δύο στην οριζόντια πλευρά του και δύο στην κατακόρυφη, για σύνδεση με την ορθογωνική βάση από Plexiglass και την πλαστική αλυσίδα αντίστοιχα. Όλες οι συνδέσεις πραγματοποιούνται με την χρήση κοχλία-περικόχλιου. Όπως στην περίπτωση της ατράκτου του ρομποτικού ψαριού, ως υλικό κατασκευής επιλέχθηκε το αλουμίνιο και η κατασκευή του υλοποιήθηκε με την χρήση συμβατικών εργαλειομηχανών (φρέζα, τρυπάνι), στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου.



Σχήμα 4-10. Εξάρτημα Σύνδεσης Πλαστικής Αλυσίδας.

Ελαστικός Σύνδεσμος (κόπλερ)

Η επιλογή του συνδέσμου μετάδοσης κίνησης έγινε μέσα από πληθώρα διαφορετικών ειδών συνδέσμων [21], εκ των οποίων οι κυριότεροι είναι οι:

- Σύνδεσμοι άκαμπτου τύπου (rigid type)
- Αξονικοί ελαστικοί σύνδεσμοι (flexible shaft couplings)
- Ελαστικοί σύνδεσμοι με φλάντζα (flexible flanged)
- Αλυσιδωτοί σύνδεσμοι (chain couplings)

Για την παρούσα κατασκευή, κρίθηκε απαραίτητη η χρήση μικρού μεγέθους συνδέσμου, με διάμετρο οπής της τάξης των 3 mm (διάμετρος άξονα ηλεκτρικού κινητήρα). Λόγω μικρής διαθεσιμότητας στα μεγέθη αυτά αποκλείστηκαν οι αλυσιδωτοί σύνδεσμοι και οι σύνδεσμοι με φλάντζα. Επιπρόσθετα, η υδροδυναμική αντίσταση λόγω της κίνησης του ρομποτικού ψαριού μέσα στο νερό με τη χρήση του φορείου ρυμούλκησης, δημιουργεί καμπτικά φορτία στην άτρακτο του ψαριού, που επιβαρύνουν με τη σειρά τους τον ηλεκτρικό κινητήρα. Για τον λόγο αυτό, επιλέχθηκε αξονικός ελαστικός σύνδεσμος (βλ. Σχήμα 4-11) CPLS12-3-3 της εταιρίας “Misumi”, ο οποίος μπορεί να απορροφήσει τέτοιου είδους φορτία. Λεπτομερή τεχνικά χαρακτηριστικά του συγκεκριμένου συνδέσμου παρουσιάζονται στο Παράρτημα, στο τέλος της διπλωματικής εργασίας αυτής.

Σχήμα 4-11. Ελαστικός Σύνδεσμος.

Έδρανο κύλισης

Τα έδρανα κύλισης μπορούν να κατηγοριοποιηθούν ανάλογα με τον τύπο στοιχείων κύλισης που χρησιμοποιούν σε [20]:

- Μεταλλικά σφαιρικά έδρανα (ball bearings)
- Μεταλλικά κυλινδρικά έδρανα (roller bearings)
- Κεραμικά έδρανα (ceramic bearings)

Τα μεταλλικά κυλινδρικά έδρανα χαρακτηρίζονται από υψηλή αντοχή σε ακτινικά φορτία. Ωστόσο, σε φορτίσεις λόγω κακής ευθυγράμμισης μεταξύ εσωτερικού και εξωτερικού δακτυλίου, όπως στην περίπτωση κλίσης της ατράκτου του ρομποτικού ψαριού λόγω καμπτικών ροπών, εμφανίζουν αυξημένη φθορά στα στοιχεία κύλισης λόγω ανομοιόμορφων φορτίσεων. Τα κεραμικά έδρανα χαρακτηρίζονται από μεγάλη αντοχή σε διάβρωση, σε σύγκριση με τα έδρανα που χρησιμοποιούν μεταλλικά στοιχεία κύλισης. Παρόλο που θα ήταν ανθεκτικά στο διαβρωτικό περιβάλλον της δεξαμενής πειραματικών δοκιμών, το περιορισμένο εύρος διαστάσεων στην επιθυμητή τάξη μεγέθους για την εφαρμογή αυτή (εσωτερική διάμετρος: 8 mm, πλάτος: <= 5 mm) οδηγεί στην απόρριψη τους. Λαμβάνοντας υπόψη τα πιο πάνω, επιλέχθηκε το μεταλλικό σφαιρικό έδρανο κύλισης SFLC688ZZ της εταιρίας “Misumi” (βλ. Σχήμα 4-12). Το έδρανο έχει εσωτερική διάμετρο 8 mm, εξωτερική διάμετρο 16 mm και πλάτος 5 mm. Διαθέτει διαμόρφωση φλάντζας στον εξωτερικό δακτύλιο ώστε να συγκρατείται στην εσωτερική επιφάνεια του σωλήνα ορθογωνικής διατομής και διπλή στεγάνωση για προστασία από σκόνες ή σταγόνες νερού.

Σχήμα 4-12. Έδρανο Κύλισης.

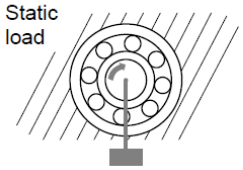
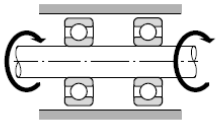
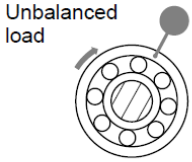
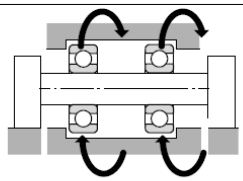
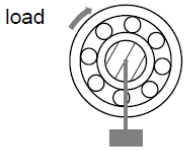
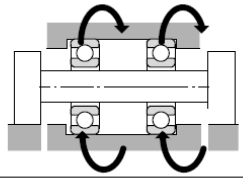
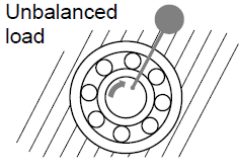
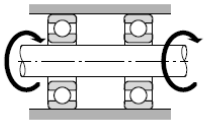
4.4.2 Επιλογή Συναρμογών Εδράνου Κύλισης

Η επιλογή κατάλληλου τύπου συναρμογών μεταξύ των δακτυλίων του εδράνου κύλισης και των επιφανειών που συνεργάζονται μαζί τους, παίζει καθοριστικό ρόλο στη διάρκεια ζωής του εδράνου κύλισης, αλλά και στην συναρμολόγηση της βάσης 3^{ου} άξονα. Γενικά, η επιλογή μεταξύ σφικτής ή ελεύθερης συναρμογής γίνεται λαμβάνοντας υπόψη τα περιστρεφόμενα μέρη ενός εδράνου κύλισης (εσωτερικός ή εξωτερικός δακτύλιος) καθώς και το είδος του ακτινικού φορτίου που ασκείται στο έδρανο (στατικό ή μεταβαλλόμενο), [22].

Ο Πίνακας 4-1 παρουσιάζει τέσσερεις διαφορετικές περιπτώσεις εφαρμογών για στατικό ή μεταβαλλόμενο φορτίο, που ασκείται στον περιστρεφόμενο εσωτερικό ή εξωτερικό δακτύλιο ενός εδράνου κύλισης. Στην παρούσα εφαρμογή ο δακτύλιος που περιστρέφεται είναι ο εσωτερικός, ο οποίος συνεργάζεται με την επιφάνεια της ατράκτου του ρομποτικού ψαριού. Το φορτίο μπορεί να χαρακτηριστεί ως μεταβαλλόμενο, καθώς η διεύθυνση άσκησης του μεταβάλλεται ανάλογα με την κατεύθυνση τη κίνησης του ρομποτικού ψαριού ενώ κινείται με την βοήθεια του φορείου ρυμούλκησης. Λαμβάνοντας υπόψη τον Πίνακας 4-1 και τις συνθήκες λειτουργίας της βάσης 3^{ου} άξονα, επιλέγεται σφικτή συναρμογή μεταξύ εξωτερικού δακτυλίου και σωλήνα ορθογωνικής διατομής, και ελεύθερη συναρμογή μεταξύ εσωτερικού δακτυλίου και ατράκτου ρομποτικού ψαριού.

Επίσης, η συγκεκριμένη επιλογή πληροί την προδιαγραφή για εύκολη συναρμολόγηση και αποσυναρμολόγηση της τελικής κατασκευής, καθώς αποφεύγεται η άσκηση δύναμης και η χρήση εξειδικευμένου εξοπλισμού που θα προκαλούσε πιθανή βλάβη στο έδρανο.

Πίνακας 4-1. Ακτινικά φορτία και συναρμογές εδράνων κύλισης.

Illustration	Bearing rotation	Ring load	Fit
Static load 	 Inner ring: Rotating Outer ring: Stationary	Rotating inner ring load	Inner ring : Tight fit
Unbalanced load 	 Inner ring: Stationary Outer ring: Rotating	Static outer ring load	Outer ring : Loose fit
Static load 	 Inner ring: Stationary Outer ring: Rotating	Static inner ring load	Inner ring : Loose fit
Unbalanced load 	 Inner ring: Rotating Outer ring: Stationary	Rotating outer ring load	Outer ring : Tight fit

Λαμβάνοντας υπόψη τις πιο πάνω επιλογές και τις διαμέτρους των δακτυλίων του εδράνου κύλισης επιλέχθηκαν οι πιο κάτω συναρμογές όπως φαίνονται στις εξισώσεις (4-1) και (4-2), για εξωτερικό και εσωτερικό δακτύλιο αντίστοιχα.

$$16\frac{N6}{h5} = \frac{16^{-9}_{-20}}{16^{0}_{-8}} \tag{4-1}$$

$$8\frac{H5}{f6} = \frac{8^{0}_{-8}}{8^{-13}_{-22}} \tag{4-2}$$

4.4.3 Πίνακας Υλικών (BOM)

Στη συνέχεια, παρουσιάζεται ο συγκεντρωτικός πίνακας υλικών (Πίνακας 4-2) που χρησιμοποιήθηκαν για την κατασκευή της βάσης 3^{ου} άξονα.

Πίνακας 4-2. Πίνακας Υλικών (BOM)

Επίπεδο	Αρ. Τεμαχίου	Περιγραφή	Ποσότητα	Αγορά/Κατασκευή
1	1	Ηλεκτρικός Κινητήρας Maxon RE 25 Motor	1	-
1	2	Βάση από Σωλήνα Ορθογωνικού Προφίλ	1	Κατασκευή στο Εργαστήριο Αυτομάτου Ελέγχου
1	3	Ορθογωνική Βάση από Plexiglass	1	Κατασκευή σε εξωτερικό μηχανουργείο
1	4	Άτρακτος Ρομποτικού Ψαριού	1	Κατασκευή στο Εργαστήριο Αυτομάτου Ελέγχου
1	5	Εξάρτημα Σύνδεσης Πλαστικής Αλυσίδας	1	Κατασκευή στο Εργαστήριο Αυτομάτου Ελέγχου
1	6	Ελαστικός Σύνδεσμος (κόπλερ) CPLS12-3-3	1	Αγορά από εταιρία “Misumi”
1	7	Έδρανο Κύλισης SFLC688ZZ	1	Αγορά από εταιρία “Misumi”
1	8	Γραμμικά Έδρανα Κύλισης	4	-
1	9	Κοχλίες Ορθογωνικού Σωλήνα M4	4	-
1	10	Περικόχλια M4	4	-
1	11	Ροδέλες M4	4	-
1	12	Κοχλίες Ηλεκτρικού Κινητήρα M2	3	-
1	13	Κοχλίες Εξαρτήματος Αλυσίδας M2	2	-
1	14	Περικόχλια M2	2	-
1	15	Ροδέλες M2	2	-
1	16	Κοχλίες Βάσης Plexiglass M5	16	-

4.5 Κατασκευή και Συναρμολόγηση

Από το σύνολο των εξαρτημάτων που αποτελούν την βάση του 3^{ου} άξονα μόνο η βάση από σωλήνα ορθογωνικής διατομής, η άτρακτος του ρομποτικού ψαριού και το εξάρτημα σύνδεσης με την πλαστική αλυσίδα κατασκευάστηκαν στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου. Η ενότητα αυτή επικεντρώνεται στην περιγραφή της κατασκευής του πρώτου, καθώς είναι το μόνο που απαιτεί υψηλή διαστασιολογική ακρίβεια και χρήση εξειδικευμένου εξοπλισμού. Η κατασκευή του συγκεκριμένου εξαρτήματος έγινε με τη χρήση του αυτοματοποιημένου Κέντρου Κατεργασιών HAAS Mini Mill, ενώ ο προγραμματισμός των κατεργασιών έγινε με την βοήθεια του λογισμικού κατεργασιών SolidCAM. Τέλος, παρουσιάζεται το τελικό συναρμολόγημα και η σύνδεση του στο φορείο ρυμούλκησης της δεξαμενής πειραματικών δοκιμών.

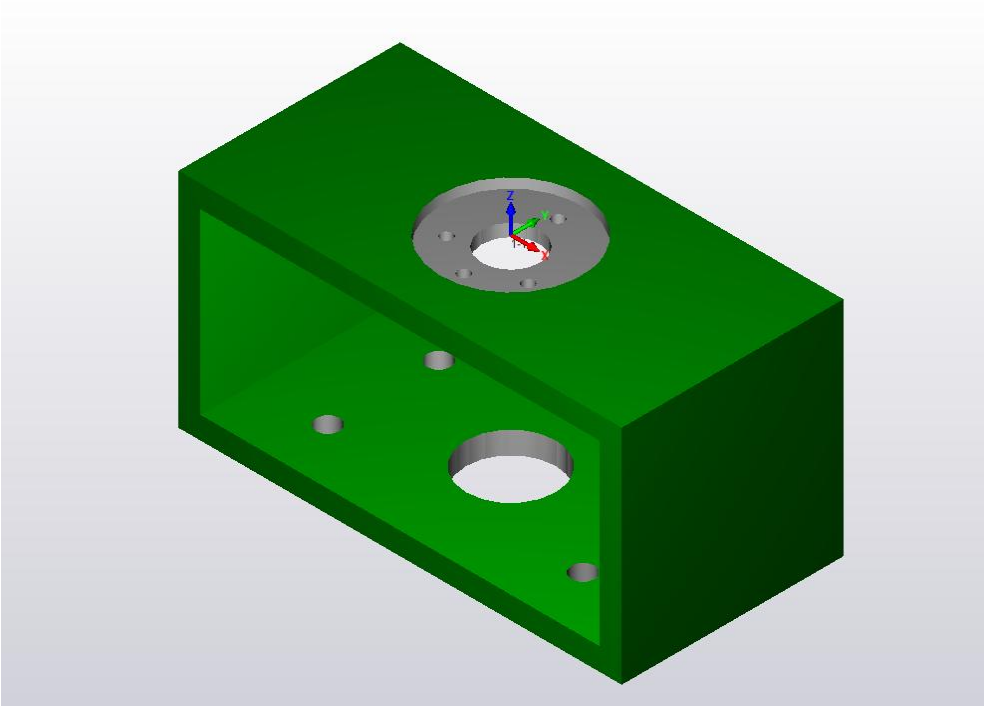
4.5.1 Λογισμικό Σχεδίασης Μηχανουργικών Κατεργασιών SolidCam

Το λογισμικό Σχεδιασμού Μηχανουργικών Κατεργασιών SolidCAM είναι ένα ενσωματωμένο λογισμικό στο σχεδιαστικό πακέτο SolidWORKS. Στόχος του λογισμικού αυτού, είναι ο σχεδιασμός και η προσομοίωση των διαφόρων μηχανουργικών κατεργασιών που μπορούν να γίνουν σε ένα κέντρο κατεργασιών (CNC Machine). Για παράδειγμα, δίνοντας ως είσοδο το τρισδιάστατο μοντέλο του τεμαχίου που πρόκειται να κατασκευαστεί, ο χρήστης έχει τη δυνατότητα μέσω του λογισμικού να επιλέξει παραμέτρους όπως είδος μηχανήματος, είδος κατεργασίας, υλικό κατασκευής, εργαλεία και συνθήκες κατεργασίας. Στη συνέχεια, αφού έχει δοκιμάσει τις επιλογές του στο περιβάλλον προσομοίωσης του λογισμικού, έχει τη δυνατότητα να εξάγει αυτόματα τις εντολές που καθορίζουν τις επιθυμητές κατεργασίες, στην γλώσσα προγραμματισμού αριθμητικού ελέγχου (G-code) που είναι συμβατή με τη εκάστοτε εργαλειομηχανή. Η δυνατότητα προσομοίωσης των κατεργασιών στο περιβάλλον του λογισμικού, πριν αυτές εφαρμοστούν στην πράξη, διευκολύνει την διάγνωση και αποφυγή σχεδιαστικών σφαλμάτων, που θα μπορούσαν να θέσουν σε κίνδυνο το μηχάνημα, το κατεργάσιμο τεμάχιο και προπάντων την υγεία του χρήστη.

Στην ενότητα αυτή, παρουσιάζεται συνοπτικά ο σχεδιασμός των κατεργασιών που έγιναν στο σωλήνα ορθογωνικής διατομής με την βοήθεια του λογισμικού σχεδίασης κατεργασιών SolidCAM.

Συστήματα Συντεταγμένων

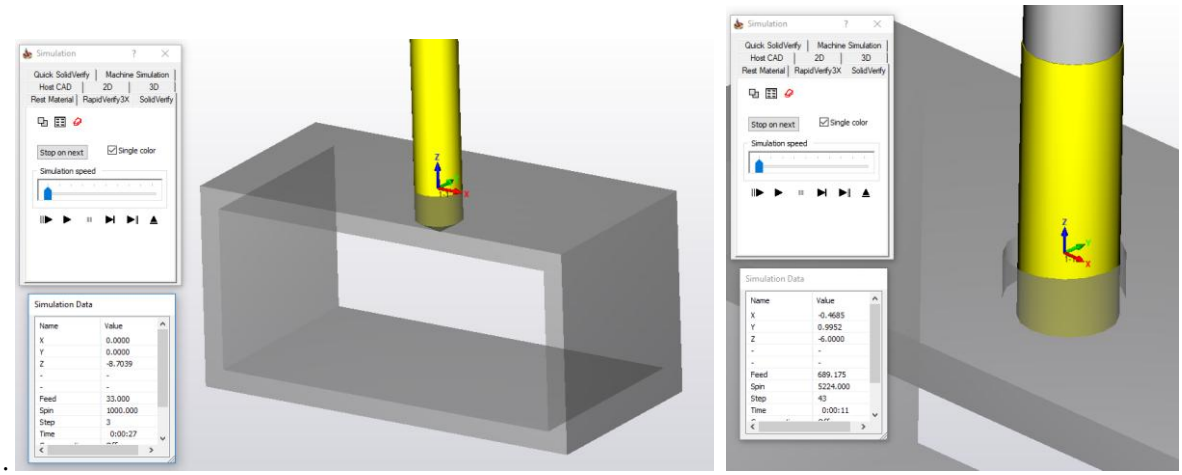
Οι κατεργασίες στη βάση ορθογωνικού σωλήνα χωρίζονται σε δύο κατηγορίες, ανάλογα με το σύστημα συντεταγμένων στο οποίο έχουν σχεδιαστεί. Επιλέχθηκαν δύο συστήματα συντεταγμένων, στο κέντρο της πάνω και κάτω πλευράς του τεμαχίου, ώστε να έχουν κοινό κατακόρυφο άξονα z (βλ Σχήμα 4-13). Η επιλογή αυτή διευκόλυνε την επίτευξη της απόλυτης ευθυγράμμισης των οπών για τον άξονα του κινητήρα και της ατράκτου του ρομποτικού ψαριού.



Σχήμα 4-13. Σύστημα συντεταγμένων πάνω πλευράς.

Διαμόρφωση Στήριξης Κινητήρα (1^{ο} Σύστημα Συντεταγμένων)

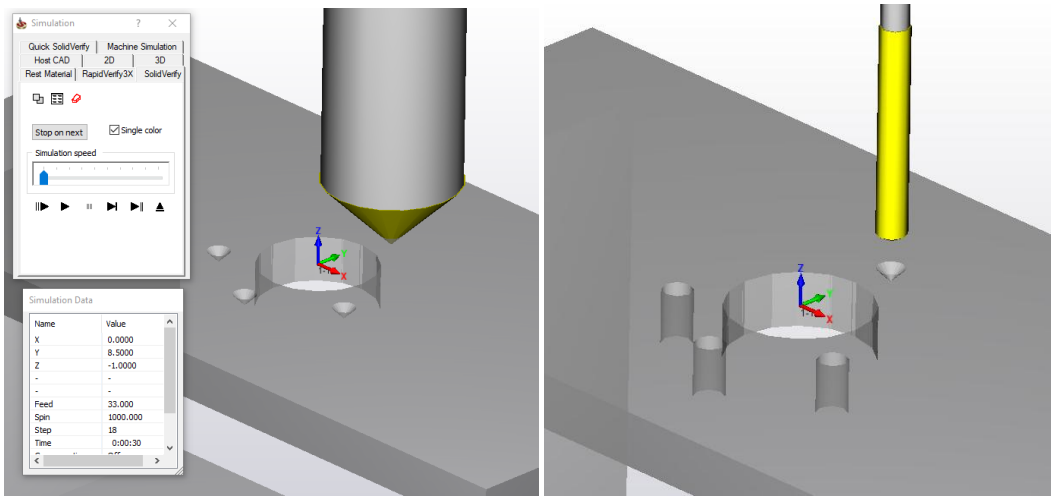
Η διαμόρφωση στήριξης του ηλεκτρικού κινητήρα σχεδιάστηκε με βάση το σύστημα συντεταγμένων της πάνω πλευράς. Η διάνοιξη της οπής διαμέτρου 10 mm πραγματοποιήθηκε σε δύο στάδια. Αρχικά χρησιμοποιήθηκε τρυπάνι διαμέτρου 9 mm (T3) για την διάνοιξη της οπής (βλ. Σχήμα 4-14-i), και στην συνέχεια κονδύλι διαμέτρου 7,98 mm (T1) για πλάτυνση της οπής μέχρι την επιθυμητή διάσταση και φινίρισμα (βλ. Σχήμα 4-14-ii)



(i) **(ii)**

Σχήμα 4-14. (i) Διάνοιξη οπής 9 mm, (ii) Πλάτυνση και φινίρισμα.

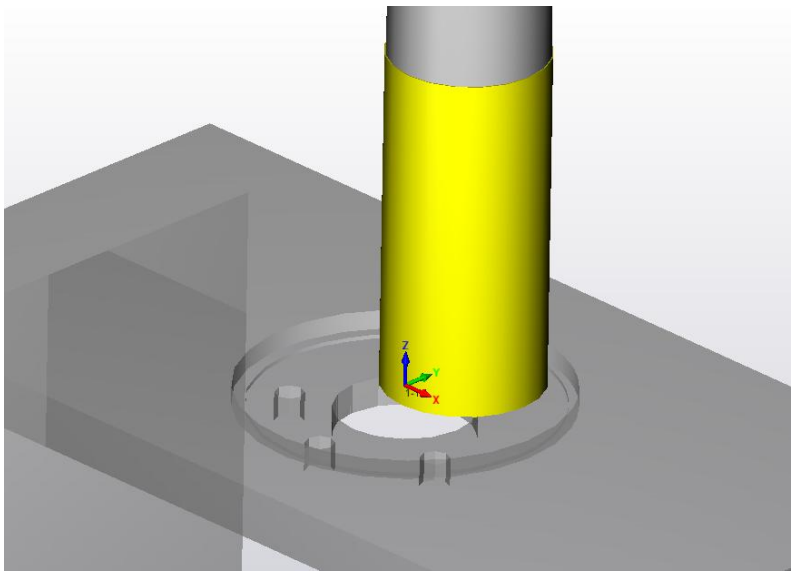
Η διάνοιξη των τεσσάρων οπών για την σύνδεση του κινητήρα πραγματοποιήθηκε σε δύο στάδια. Σε πρώτο στάδιο, για αποφυγή επιβάρυνσης του τρυπανιού και επίτευξη ακρίβειας στη θέση των οπών, χρησιμοποιήθηκε εργαλείο πονταρίσματος (center drill) διαμέτρου 12 mm (T6), το οποίο χάραξε την επιφάνεια στα σημεία των οπών σε βάθος 1 mm (βλ Σχήμα 4-15-i). Σε δεύτερο στάδιο, ολοκληρώθηκε η διάνοιξη των οπών στα σημαδεμένα πλέον σημεία, με τη χρήση τρυπανιού διαμέτρου 2 mm (T5) (βλ. Σχήμα 4-15-ii).



(i) **(ii)**

Σχήμα 4-15. (i) Ποντάρισμα, (ii) Διάνοιξη οπών.

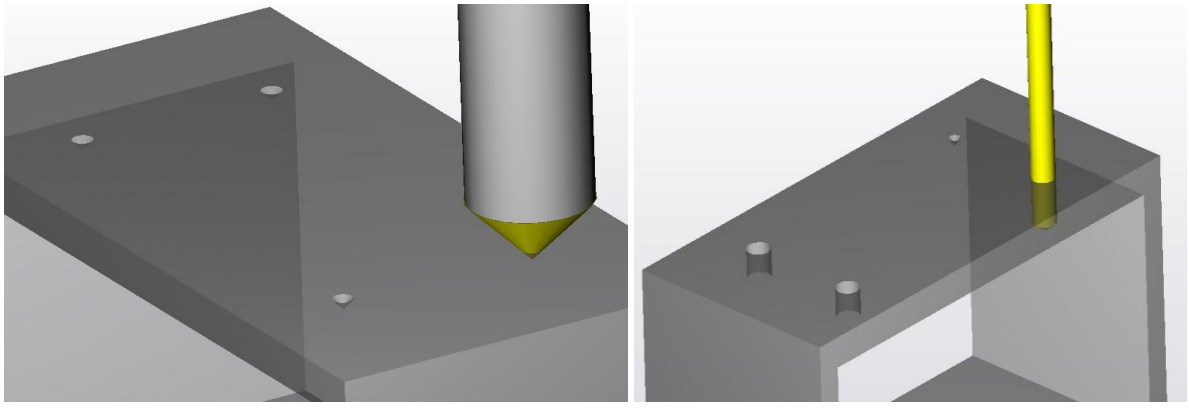
Η τελική διαμόρφωση ποκέτας διαμέτρου 25 mm και βάθους 1,75 mm, στη οποία θα ακουμπήσει η επιφάνεια του κινητήρα, δημιουργήθηκε με την χρήση κονδυλιού διαμέτρου 12 mm (T2) (βλ. Σχήμα 4-16). Το φινίρισμα της διαμόρφωσης αυτής πραγματοποιήθηκε με το ίδιο εργαλείο.



Σχήμα 4-16. Διαμόρφωση ποκέτας.

Κατεργασίες Κάτω Επιφάνειας (2^{o} Σύστημα Συντεταγμένων)

Οι κατεργασίες στην κάτω επιφάνεια του σωλήνα ορθογωνικής διατομής σχεδιάστηκαν με βάση το σύστημα συντεταγμένων της κάτω πλευράς. Οι τέσσερεις οπές για την σύνδεση του σωλήνα στην βάση από plexiglass, πραγματοποιήθηκαν σε δύο στάδια, όπως και στην περίπτωση των οπών για την σύνδεση του κινητήρα. Αρχικά έγινε ποντάρισμα σε βάθος 1 mm με το εργαλείο ποντορίσματος (T6) (βλ. Σχήμα 4-17-i) και ακολούθως ολοκληρώθηκε η διάνοιξη τους με τρυπάνι διαμέτρου 4 mm (T4) (βλ Σχήμα 4-17-ii).

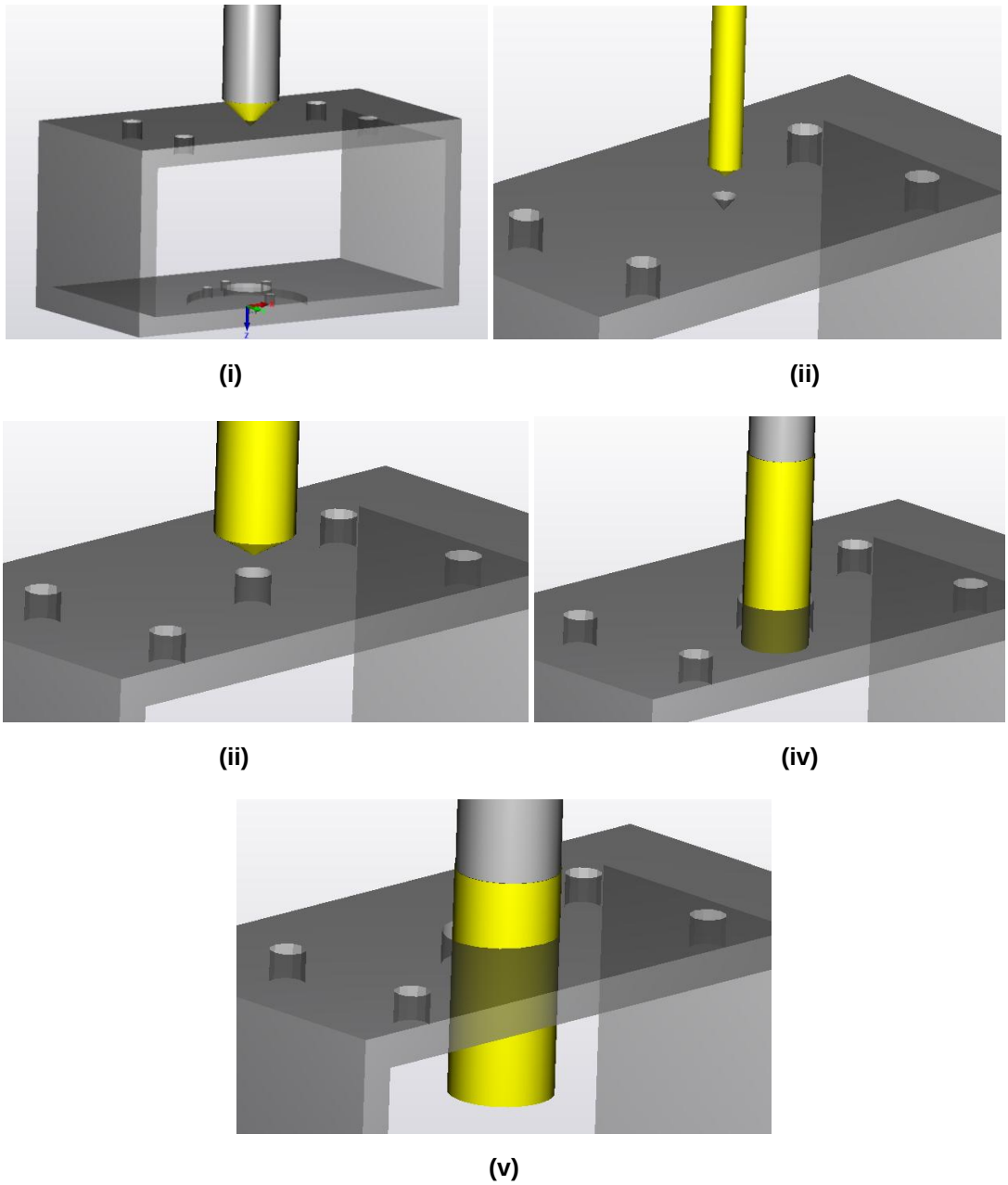


(i) **(ii)**

Σχήμα 4-17. (i) Ποντάρισμα, (ii) Διάνοιξη οπών.

Η διάνοιξη της οπής διαμέτρου 16 mm στην οποία θα τοποθετηθεί το έδρανο κύλισης απαιτεί ιδιαίτερη προσοχή καθώς η εσωτερική της επιφάνεια θα βρίσκεται σε συναρμογή σύσφιξης με τον εξωτερικό δακτύλιο του εδράνου. Για την επίτευξη διαστασιολογικής ακρίβειας (ανοχής όπως ορίστηκε

σε προηγούμενη ενότητα) και υψηλής ποιότητας επιφανείας, η συγκεκριμένη κατεργασία χωρίζεται σε πέντε διαφορετικά στάδια. Σε πρώτο και δεύτερο στάδιο, δημιουργείται οπή διαμέτρου 4 mm με την χρήση του εργαλείου πονταρίσματος (T6) (βλ. Σχήμα 4-18-i) και τρυπανιού διαμέτρου 4 mm (T4) (βλ. Σχήμα 4-18-ii), όπως και στις προηγούμενες κατεργασίες διάνοιξης οπής. Στη συνέχεια, η οπή αυτή διευρύνεται σε διάμετρο 9 mm με τη χρήση τρυπανιού (T3) (βλ. Σχήμα 4-18-iii). Αφού πλέον το επιτρέπουν οι διαστάσεις της οπής, χρησιμοποιείται κονδύλι διαμέτρου 7,98 mm (T1) για διαπλάτυνση της οπής μέχρι διάμετρο 14 mm (βλ. Σχήμα 4-18-iv). Ακολούθως, με τη χρήση μεγαλύτερου κονδυλιού, διαμέτρου 12 mm (T2), γίνεται διεύρυνση της οπής στην τελική της διάσταση και ολοκληρώνεται το τελικό φινίρισμα της επιφάνειας (βλ. Σχήμα 4-18-v).



Σχήμα 4-18. Διάνοιξη οπής εδράνου κύλισης: (i) Χάραξη οπής, (ii) Διάτρηση 4 mm, (iii) Διάτρηση 9 mm, (iv) Διαπλάτυνση 14 mm, (v) Διαπλάτυνση 16 mm.

Η σταδιακή αύξηση στις διαμέτρους των εργαλείων που χρησιμοποιούνται διευκολύνει την μείωση των φορτίων που ασκούνται σε αυτά και την αποφυγή ανάπτυξης ταλαντώσεων (φαινόμενο “chattering”), το οποίο θα ήταν καταστροφικό για την ποιότητα της επιφάνειας.

Μετά την ολοκλήρωση της σχεδίασης των κατεργασιών στο λογισμικό SolidCAM, εξάγονται οι απαραίτητες εντολές σε γλώσσα προγραμματισμού αριθμητικού ελέγχου (G-code), οι οποίες θα χρησιμοποιηθούν στο αυτοματοποιημένο κέντρο κατεργασιών HAAS Mini Mill του εργαστηρίου, για την κατασκευή την βάσης ορθογωνικού προφίλ. Οι εντολές αυτές παρουσιάζονται αναλυτικά στον ψηφιακό δίσκο (CD) που παραδίδεται με την παρούσα εργασία, ενώ η διαδικασία με την οποία εξάγονται περιγράφεται στο Παράρτημα, στο τέλος της εργασίας.

4.5.2 Κέντρο Κατεργασιών HAAS Mini Mill

Το εξάρτημα της βάσης από σωλήνα ορθογωνικής διατομής κατασκευάστηκε στο αυτόματο κέντρο κατεργασιών HAAS Mini Mill (βλ. Σχήμα 4-19 και Σχήμα 4-20), που βρίσκεται στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου.

Σχήμα 4-19. Αυτόματο Κέντρο Κατεργασιών HAAS Mini Mill.

Σχήμα 4-20. Αυτόματο Κέντρο Κατεργασιών – Εσωτερικός Χώρος Εργασίας.

Πριν από την έναρξη οποιασδήποτε κατεργασίας, τα απαραίτητα εργαλεία τοποθετήθηκαν στον εργαλειοφορέα τύπου Carousel του κέντρου κατεργασιών, για να γίνει η αντιστάθμιση των διαστάσεων τους. Η πλήρης περιγραφή των κοπτικών εργαλείων που χρησιμοποιήθηκαν (βλ. Σχήμα 4-21) παρουσιάζεται στον Πίνακας 4-3.

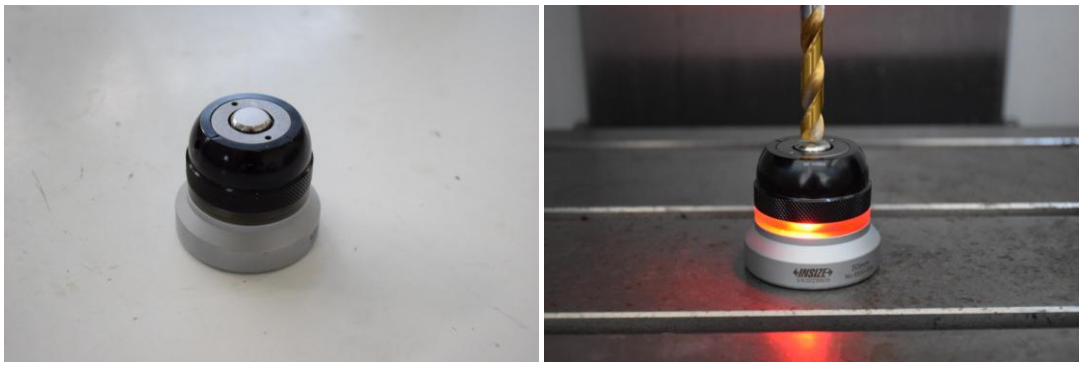
Σχήμα 4-21. Κοπτικά εργαλεία.

Πίνακας 4-3. Κοπτικά Εργαλεία.

Αρ. Εργαλείου	Τύπος	Διάμετρος (mm)	Μήκος κοπτικής ακμής (mm)	Αρ. Πτερυγίων
T1	Κονδύλι ενιαίο	7,98	23,3	4
T2	Κονδύλι ενιαίο	12	26,2	2
T3	Τρυπάνι	9	69,52	2
T4	Τρυπάνι	4	36,6	2
T5	Τρυπάνι	2	16,94	2
T6	Τρυπάνι πονταρίσματος (center drill)	12	15	2
T7	Κονδύλι με αναλώσιμα κοπτικά πλακίδια	32	5	3

Το κοπτικό εργαλείο T7 χρησιμοποιήθηκε για εξομάλυνση της παράπλευρης επιφάνειας του σωλήνα με ορθογωνική διατομή, η οποία εφάπτεται στην επιφάνεια εργασίας του κέντρου κατεργασιών. Η κατεργασία αυτή υλοποιήθηκε χειροκίνητα και για τον λόγο αυτό δεν προσομοιώθηκε στο λογισμικό σχεδίασης κατεργασιών.

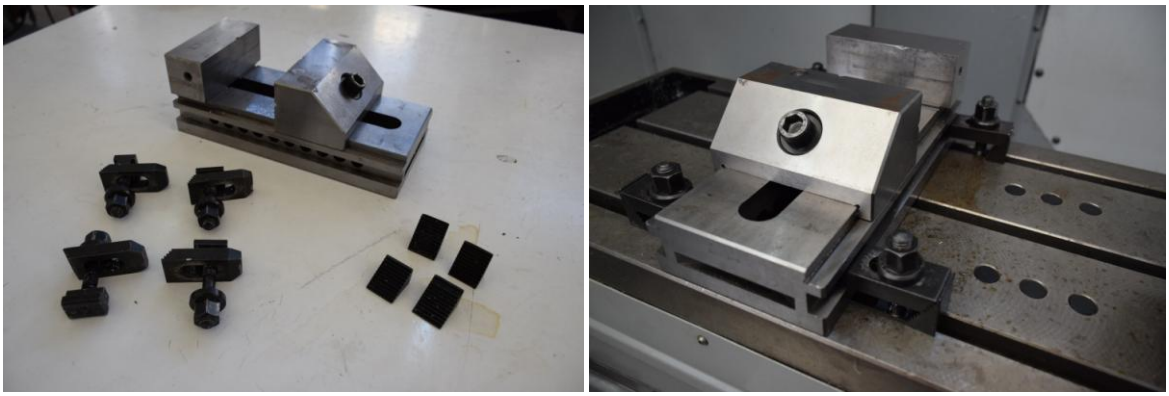
Για τον υπολογισμό των αντισταθμίσεων των εργαλείων και τον ορισμό τον διαστάσεων τους στο σύστημα της εργαλειομηχανής, χρησιμοποιήθηκε ο ηλεκτρονικός ρυθμιστής σημείου μηδέν (electronic zero setter) (βλ. Σχήμα 4-22-i). Η ηλεκτρικά αγώγιμη βάση του εργαλείου αυτού συνδέεται στην επιφάνεια εργασίας της εργαλειομηχανής με τη χρήση ενσωματωμένων μαγνητών. Η άτρακτος της εργαλειομηχανής που φέρει το εργαλείο, μεταφέρεται ακριβώς πάνω από τον ηλεκτρονικό ρυθμιστή και τον προσεγγίζει με πολύ χαμηλή ταχύτητα. Μόλις το κοπτικό εργαλείο έρθει σε επαφή με την επιφάνεια του ρυθμιστή, το ηλεκτρικό κύκλωμα επιφάνειας εργασίας-εργαλειομηχανής-εργαλείου κλείνει και μία φωτεινή ένδειξη ανάβει (βλ. Σχήμα 4-22-ii). Το κύκλωμα αυτό τροφοδοτείται από μία ενσωματωμένη μπαταρία στον ρυθμιστή. Στη συνέχεια, η ακριβής θέση της ατράκτου αποθηκεύεται στο σύστημα και από τη μέτρηση της θέσης του κατακόρυφου άξονα z αφαιρείται το ύψος του ηλεκτρονικού ρυθμιστή. Με τον τρόπο αυτό καταχωρείται στο σύστημα της εργαλειομηχανής η ακριβής διάσταση του εργαλείου. Η διαδικασία γίνεται χειροκίνητα από τον χειριστή μέσω της μονάδας ελέγχου της εργαλειομηχανής και επαναλαμβάνεται μέχρι να καταχωρηθούν οι διαστάσεις όλων των εργαλείων. Λεπτομερής περιγραφή των τεχνικών χαρακτηριστικών του ηλεκτρονικού ρυθμιστή παρουσιάζεται στο Παράρτημα, που βρίσκεται στο τέλος της παρούσας διπλωματικής εργασίας.



Σχήμα 4-22. (i) Ηλεκτρονικός ρυθμιστής σημείου μηδέν, (ii) Μέτρηση διαστάσεων κοπτικού εργαλείου.

Η συγκράτηση του τεμαχίου προς κατεργασία πραγματοποιείται με τη χρήση μεταλλικής μέγγενης (βλ. Σχήμα 4-23-i), η οποία με τη σειρά της συνδέεται στην επιφάνεια εργασίας της εργαλειομηχανής μέσω κοχλιοσύνδεσης (βλ. Σχήμα 4-23-ii). Για την αποφυγή διαστασιολογικών αποκλίσεων και σφαλμάτων στην επιθυμητή γεωμετρία του τεμαχίου, είναι απαραίτητη η πλήρης ευθυγράμμιση της μέγγενης με τους άξονες x και y της επιφάνειας εργασίας της εργαλειομηχανής. Η επίτευξη της ευθυγράμμισης αυτής πραγματοποιείται με τη χρήση αναλογικού μετρητή μετατόπισης υψηλής ακρίβειας (βλ. Σχήμα 4-24-i). Σε πρώτο στάδιο, ο μετρητής συνδέεται στη κατακόρυφη άτρακτο της εργαλειομηχανής με την χρήση ενσωματωμένου μαγνήτη (βλ. Σχήμα 4-24-ii). Στη συνέχεια, τοποθετείται με τέτοιο τρόπο, ώστε η κινούμενη ακίδα του να βρίσκεται σε επαφή με το αριστερό άκρο της κατακόρυφης επιφάνειας της πίσω σιαγώνας της μέγγενης (επιφάνεια που

συγκρατεί το τεμάχιο). Αφού ληφθεί η μέτρηση στο σημείο αυτό, το φορείο της εργαλειομηχανής μετατοπίζεται κατά τον άξονα y, ώστε η ακίδα του μετρητή να βρίσκεται στο δεξί άκρο της ίδιας επιφάνειας, όπου και επαναλαμβάνεται η μέτρηση. Ακολούθως υπολογίζεται η διαφορά στις δύο μετρήσεις και η μέγγενη μετατοπίζεται ελαφρά προς της κατεύθυνση που μειώνει την διαφορά αυτή. Η διαδικασία επαναλαμβάνεται έως ότου η διαφορά μεταξύ των δύο μετρήσεων φτάσει κάτω από ένα επιτρεπτό όριο, γεγονός που επιβεβαιώνει την ευθυγράμμιση της μέγγενης.



(i) **(ii)**

Σχήμα 4-23. (i) Μέγγενη και κοχλίες σύνδεσης, (ii) Σύνδεση με επιφάνεια εργασίας.

(i) **(ii)**

Σχήμα 4-24. (i) Αναλογικός μετρητής μετατόπισης, (ii) Σύνδεση μετρητή στην άτρακτο της εργαλειομηχανής.

Μετά την σταθεροποίηση της μέγγενης στην επιθυμητή θέση, ακολουθεί ο ορισμός του αδρανειακού συστήματος συντεταγμένων για το σύνολο των κατεργασιών που πρόκειται να πραγματοποιηθούν. Η αρχή του συστήματος αυτού επιλέγεται στη σταθερή οριζόντια επιφάνεια της πίσω σιαγώνας της μέγγενης. Για τη διαδικασία αυτή, χρησιμοποιείται το ηλεκτρονικό εργαλείο εύρεσης ακμής (electronic edge finder) (βλ. Σχήμα 4-25-i), το οποίο ακολουθεί την ίδια αρχή λειτουργίας με τον ηλεκτρονικό ρυθμιστή σημείου μηδέν που περιγράφηκε νωρίτερα. Μόλις το κάτω σφαιρικό άκρο του έρθει σε επαφή με την μεταλλική επιφάνεια που πρόκειται να μετρηθεί, το ηλεκτρικό κύκλωμα εργαλείου- μηχανής κλείνει, μία φωτεινή ένδειξη ανάβει και ένας παρατεταμένος βόμβος ακούγεται (βλ. Σχήμα 4-25-ii). Η ακριβής θέση της ατράκτου της εργαλειομηχανής

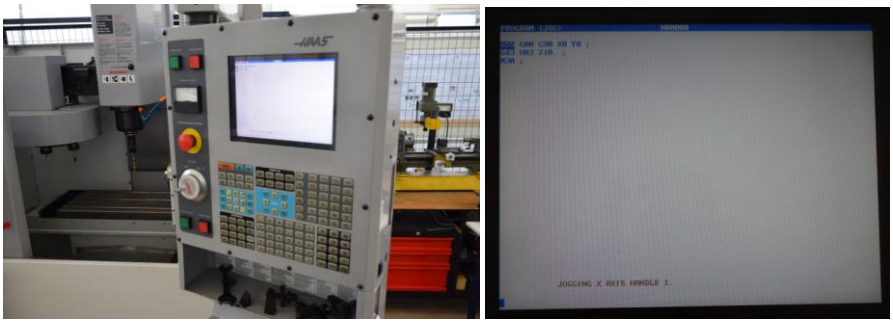
αποθηκεύεται στο σύστημα και η γνωστή ακτίνα του σφαιρικού άκρου αφαιρείται από την διάσταση που μετριέται κάθε φορά. Παρόμοια διαδικασία ακολουθείται και για τον ορισμό του σωματόδετου συστήματος συντεταγμένων, όπως αυτό παρουσιάζεται στην ενότητα της προσομοίωσης των κατεργασιών. Η διαδικασία ελέγχεται χειροκίνητα από τον χρήστη μέσω της μονάδας ελέγχου της εργαλειομηχανής. Λεπτομερής περιγραφή των τεχνικών χαρακτηριστικών του ηλεκτρονικού εργαλείου εύρεσης ακμής παρουσιάζεται στο Παράρτημα, που βρίσκεται στο τέλος της παρούσας διπλωματικής εργασίας.

(i) **(ii)**

Σχήμα 4-25. (i) Ηλεκτρονικό εργαλείο εύρεσης ακμής, (ii) Μέτρηση θέσης επιφάνειας.

Αφού έχουν οριστεί τα επιθυμητά συστήματα συντεταγμένων και έχει περαστεί ο κώδικας με τις εντολές (G-code) στο σύστημα της εργαλειομηχανής, οι κατεργασίες του τεμαχίου μπορούν να πραγματοποιηθούν αυτόματα, όπως έχουν σχεδιαστεί στο λογισμικό σχεδίασης SolidCAM. Αξίζει να σημειωθεί ότι μετά την ολοκλήρωση των κατεργασιών της πάνω επιφάνειας του σωλήνα, η διαδικασία διακόπτεται, ώστε ο χειριστής να περιστρέψει το τεμάχιο χειροκίνητα για την κατεργασία της κάτω επιφάνειας. Το δεύτερο σύστημα συντεταγμένων ορίζεται με τον ίδιο τρόπο όπως περιγράφεται πιο πάνω, πριν από την έναρξη των κατεργασιών στην κάτω πλευρά του σωλήνα.

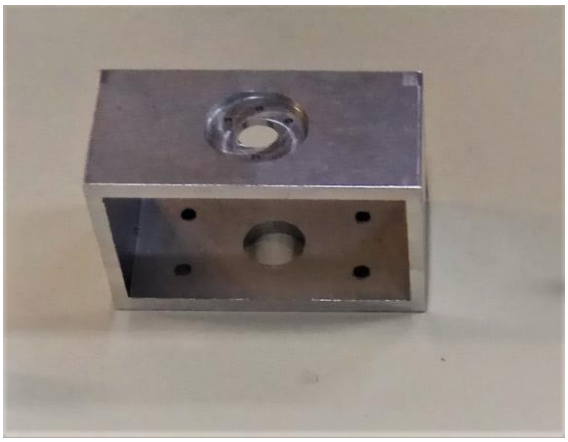
Σε οποιαδήποτε στιγμή της διαδικασίας, ο χειριστής έχει την δυνατότητα να διακόψει μία κατεργασία, να μεταβάλει τις παραμέτρους κοπής και να επιβλέψει της πρόοδο της κατεργασίας μέσω της μονάδας ελέγχου του κέντρου κατεργασιών (βλ. Σχήμα 4-26-i). Επιπρόσθετα, μέσω της ψηφιακής οθόνης της μονάδα ελέγχου (βλ. Σχήμα 4-26-ii), ο χειριστής μπορεί να βλέπει την ακριβή θέση της ατράκτου της εργαλειομηχανής σε πραγματικό χρόνο και να πραγματοποιεί αλλαγές στις εντολές του κώδικα κατεργασιών.



(i) **(ii)**

Σχήμα 4-26. (i) Μονάδα ελέγχου κέντρου κατεργασιών, (ii) Ψηφιακή οθόνη μονάδας ελέγχου.

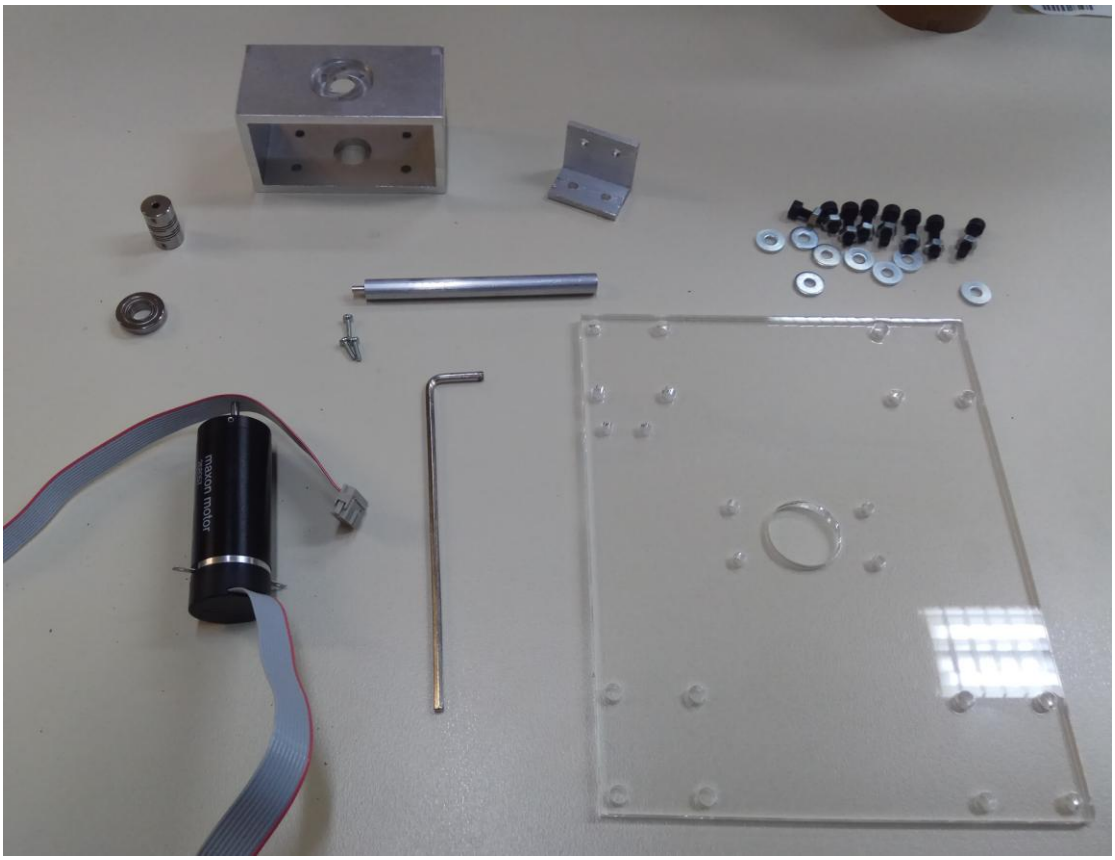
Ακολουθώντας την διαδικασία που περιγράφεται πιο πάνω, κατασκευάστηκε η μεταλλική βάση από σωλήνα ορθογωνικής διατομής (βλ. Σχήμα 4-27), η οποία θα χρησιμοποιηθεί για τη συναρμολόγηση της βάσης 3^{ου} άξονα.



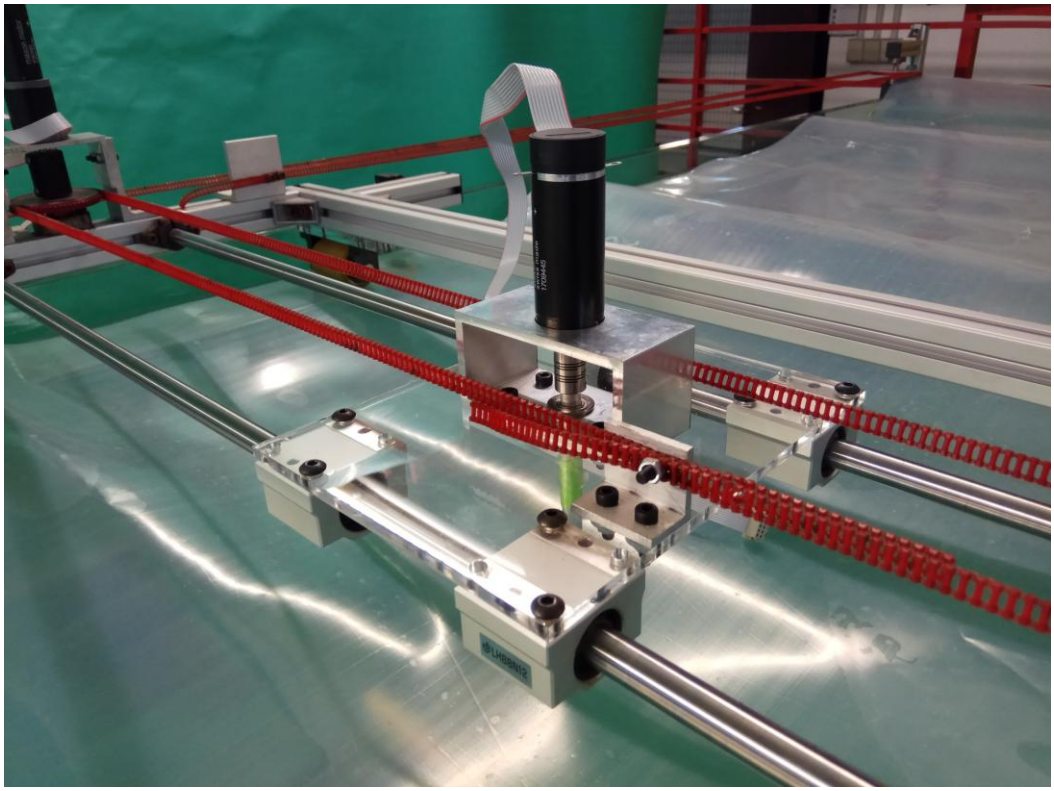
Σχήμα 4-27. Βάση από σωλήνα ορθογωνικής διατομής.

4.5.3 Συναρμολόγηση Βάσης 3^{ου} Άξονα

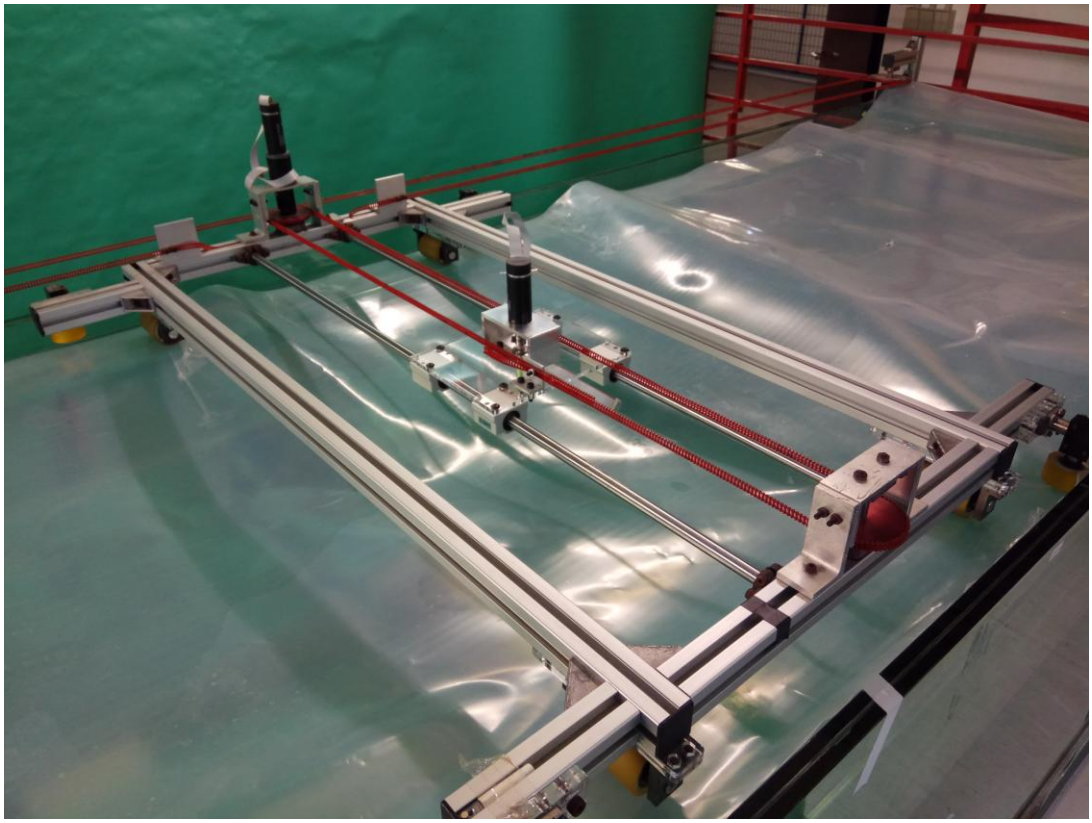
Μετά την ολοκλήρωση της κατασκευής και την προετοιμασία των επιμέρους συνιστωσών (βλ. Σχήμα 4-28), η βάση 3^{ου} άξονα συναρμολογήθηκε και ενσωματώθηκε στο κυρίως μέρος του φορείου ρυμούλκησης της δεξαμενής πειραματικών δοκιμών (βλ. Σχήμα 4-29 και Σχήμα 4-30).



Σχήμα 4-28. Συνιστώσες βάσης 3^{ου} άξονα.



Σχήμα 4-29. Συναρμολογημένη βάση 3ου άξονα.

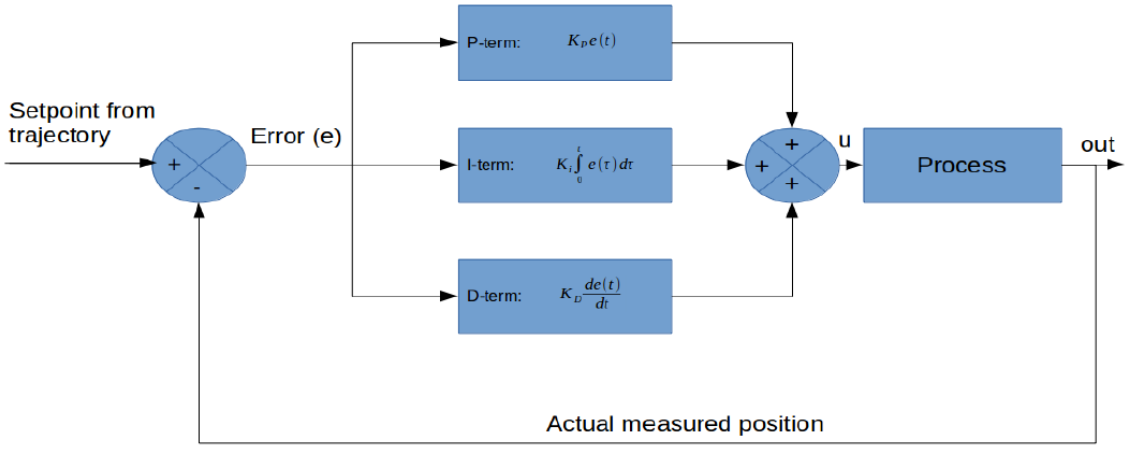


Σχήμα 4-30. Φορείο ρυμούλκησης με βάση 3ου άξονα.

4.6 Έλεγχος ηλεκτρικού κινητήρα τρίτου άξονα

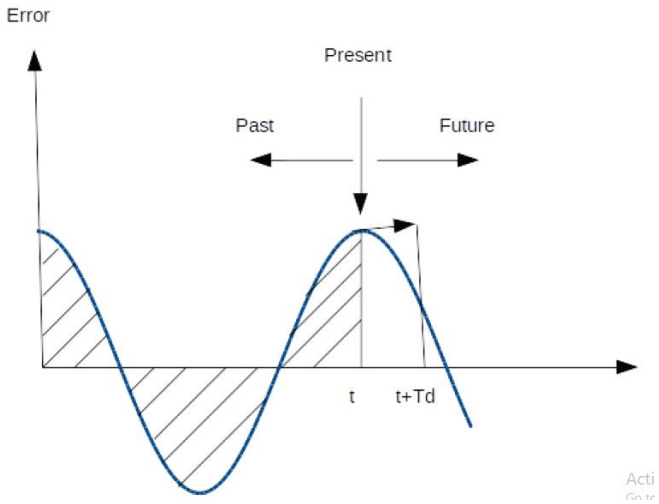
Για τον έλεγχο της γωνίας περιστροφής του ηλεκτρικού κινητήρα του τρίτου άξονα, επιλέγεται ο κλασσικός ελεγκτής (controller) Proportional-Integral-Derivative (PID). Είναι ένας από τους πιο διαδεδομένους ελεγκτές και αποτελεί έναν μηχανισμό ανατροφοδότησης σήματος σε βρόχο ελέγχου.

Ένας ελεγκτής PID (βλ. Σχήμα 4-31) διορθώνει το λάθος μεταξύ μιας μετρημένης από κατάλληλο αισθητήρα μεταβλητής (π.χ. θέση ή ταχύτητα) και ενός επιθυμητού σημείου λειτουργίας (setpoint), που προκύπτει από κατάλληλο σχεδιασμό τροχιάς (trajectory planning). Η διόρθωση αυτή υλοποιείται μέσω της παραγωγής διορθωτικού σήματος, το οποίο τροφοδοτεί την ελεγχόμενη εγκατάσταση.



Σχήμα 4-31. Δομικό διάγραμμα ελέγχου PID.

Ο υπολογισμός της εξόδου του PID περιλαμβάνει τρεις όρους, οι οποίοι επιδρούν στο σφάλμα που προκύπτει από τη σύγκριση του επιθυμητού και του μετρημένου πραγματικού σημείου λειτουργίας. Οι όροι αυτοί είναι ο αναλογικός, ο ολοκληρωτικός και ο διαφορικός. Οι όροι αυτοί σχετίζονται με τις δράσεις του ελέγχου που αφορούν το παρόν, το παρελθόν και το μέλλον του συστήματος (βλ. Σχήμα 4-32).



Σχήμα 4-32. Η επίδραση των όρων του ελεγκτή PID.

Ο αναλογικός όρος (P) καθορίζει την αντίδραση του συστήματος στο τρέχον λάθος, πολλαπλασιάζοντας το σφάλμα με μία σταθερά. Ο ολοκληρωτικός όρος (I) λαμβάνει υπόψη το ιστορικό των μετρήσεων και καθορίζει την αντίδραση του συστήματος, μέσω του υπολογισμού του ολοκληρώματος του σφάλματος στον χρόνο. Τέλος, ο παραγωγικός όρος (D) επιδρά ανάλογα με το ποσοστό μεταβολής του σφάλματος. Λαμβάνει υπόψη την παράγωγο του σφάλματος ως προς τον χρόνο και δίνει την πρόβλεψη για την μελλοντική τιμή του σφάλματος. Οι τρεις όροι αθροίζονται σταθμισμένα, παράγοντας το διορθωτικό σήμα. Το σήμα εξόδου του ελεγκτή PID παρουσιάζεται στην ακόλουθη εξίσωση:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\mathrm{t})\,\mathrm{dt} + \mathrm{T}_d \frac{de(t)}{dt} \right) \tag{4-1}$$

όπου K είναι το αναλογικό κέρδος, T_i η χρονική σταθερά ολοκλήρωσης, T_d η χρονική σταθερά παραγώγισης και e το σφάλμα (desired point – setpoint). Η εξίσωση (4-1) γράφεται και ως εξής:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{4-2}$$

όπου K_p είναι το αναλογικό κέρδος, $K_i = K / T_i$ το κέρδος του ολοκληρωτή και $K_d = K \cdot T_i$ το κέρδος του διαφορικού όρου.

Για τον προσδιορισμό των κερδών K_p , K_i και K_d υπάρχουν διαθέσιμες πολλές μεθοδολογίες. Ιδιαίτερα διαδεδομένες είναι η διαδικασία επαναλαμβανόμενων δοκιμών και σφάλματος (trial and error) και η μέθοδος Ziegler – Nichols. Ο υπολογισμός των κερδών αυτών και η υλοποίηση του ελέγχου θέσης του ηλεκτρικού κινητήρα τρίτου άξονα δεν αποτελούν αντικείμενο της παρούσας εργασίας. Ωστόσο, θα ήταν χρήσιμο να αποτελέσουν αντικείμενο μελλοντικής εργασίας, βασισμένης στην πιο πάνω θεωρητική ανάλυση.

5 Διεξαγωγή Πειραμάτων

5.1 Εισαγωγή

Το τελικό στάδιο μετά την ανάπτυξη λογισμικού για την ενσωμάτωση των δύο υποσυστημάτων στο σύστημα ROS, είναι η διεξαγωγή των πειραματικών δοκιμών που θα επιβεβαιώσουν την ορθή λειτουργία της κεντρικής μονάδας ελέγχου. Έτσι, στο κεφάλαιο αυτό, μετά από μία περιγραφή της πειραματικής διάταξης, παρουσιάζονται και αναλύονται τα πειράματα που εκτελέστηκαν και τα αποτελέσματα τους.

5.2 Περιγραφή πειραματικής διάταξης

Για τον έλεγχο της λειτουργίας των δύο υποσυστημάτων που ενσωματώνονται στην κεντρική μονάδα ελέγχου, εκτελούνται δύο διαφορετικές πειραματικές δοκιμές. Μία για το υποσύστημα ελέγχου ανοικτού βρόχου και μία για το υποσύστημα λήψης δεδομένων από περιφερειακή συσκευή. Στην παρούσα ενότητα περιγράφονται οι πειραματικές διατάξεις για τις δοκιμές αυτές και παρουσιάζονται τα αποτελέσματα τους.

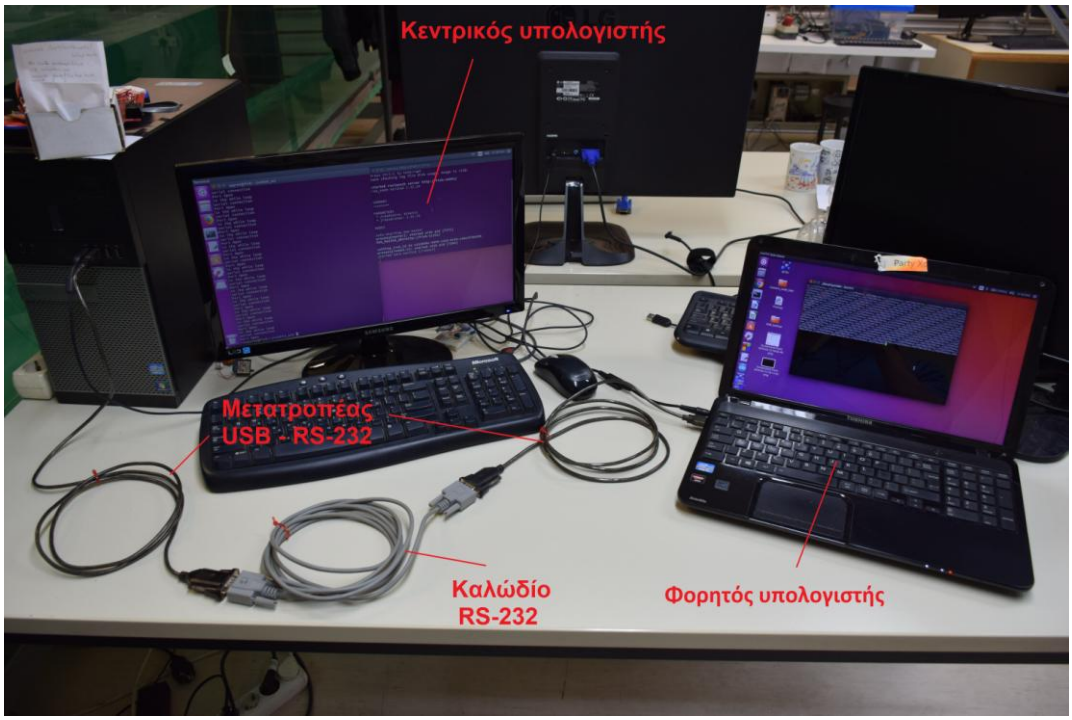
5.2.1 Δοκιμή υποσυστήματος ελέγχου ανοικτού βρόχου

Πριν από την σύνδεση του κεντρικού υπολογιστή με τον μικροελεγκτή του ρομποτικού ψαριού, πραγματοποιήθηκε δοκιμή με τη χρήση του σειριακού τερματικού ενός φορητού υπολογιστή. Πιο συγκεκριμένα, τα αποστελλόμενα δεδομένα από την κεντρική μονάδα ελέγχου, προβάλλονται στο σειριακό τερματικό του φορητού υπολογιστή. Η διαδικασία αυτή επιτρέπει την επίβλεψη της λειτουργίας του υποσυστήματος και διάγνωση πιθανών σφαλμάτων, πριν την σύνδεση με τον μικροελεγκτή. Αφού επαληθευθεί η επιθυμητή λειτουργία του υποσυστήματος, η διαδικασία επαναλαμβάνεται με σύνδεση στον μικροελεγκτή του ρομποτικού ψαριού.

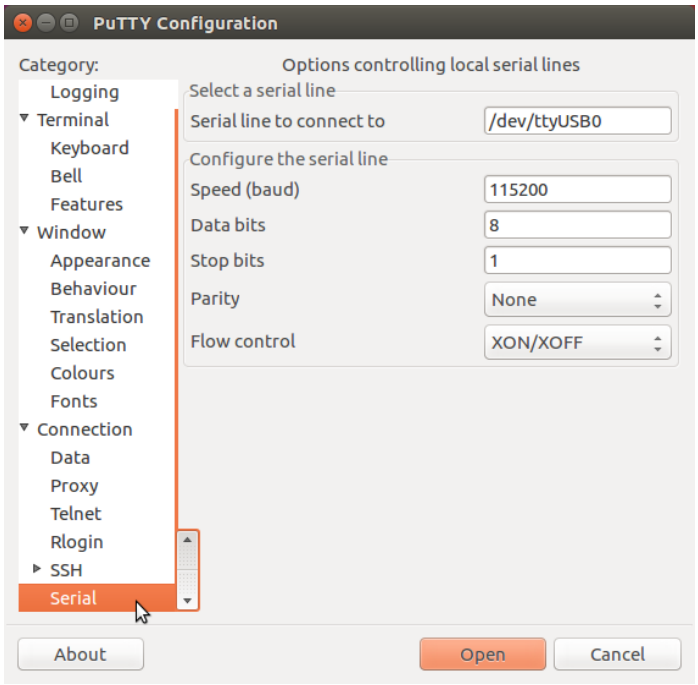
Δοκιμή με χρήση σειριακού τερματικού

Σε πρώτο στάδιο, γίνεται εγκατάσταση του μετατροπέα USB – RS-232 στον κεντρικό υπολογιστή ακολουθώντας τη διαδικασία που περιγράφεται στο Κεφάλαιο 3. Η διαδικασία επαναλαμβάνεται για έναν δεύτερο μετατροπέα, ο οποίος συνδέεται σε μία από τις θύρες USB του φορητού υπολογιστή. Τα ελεύθερα άκρα των δύο μετατροπέων συνδέονται μεταξύ τους μέσω ενός καλωδίου RS-232 σχηματίζοντας την διάταξη που παρουσιάζεται στο Σχήμα 5-1.

Στη συνέχεια, ενεργοποιείται το σειριακό τερματικό στον φορητό υπολογιστή. Στην συγκεκριμένη εφαρμογή χρησιμοποιείται το λογισμικό putty (βλ. Σχήμα 5-2). Με το άνοιγμα του παραθύρου του λογισμικού, επιλέγεται η καρτέλα Serial, από την οποία μπορούν να καθοριστούν οι διάφορες παράμετροι της σειριακής επικοινωνίας. Καταχωρείται το όνομα της θύρας USB, όπως αυτό εντοπίστηκε κατά την εγκατάσταση του μετατροπέα USB – RS-232 (ttyUSB0) και ο ρυθμός μετάδοσης δεδομένων (115200 bits/s). Ο αριθμός αυτός πρέπει να είναι ίσος με αυτόν που ορίστηκε στο εκτελέσιμο αρχείο serial_connection.py κατά την ενεργοποίηση της σειριακής θύρας. Επιλέγοντας Open, το σειριακό τερματικό ενεργοποιείται και είναι έτοιμο για την λήψη των δεδομένων.



Σχήμα 5-1. Διάταξη δοκιμής με φορητό υπολογιστή.



Σχήμα 5-2. Ρυθμίσεις σειριακής θύρας λογισμικού putty.

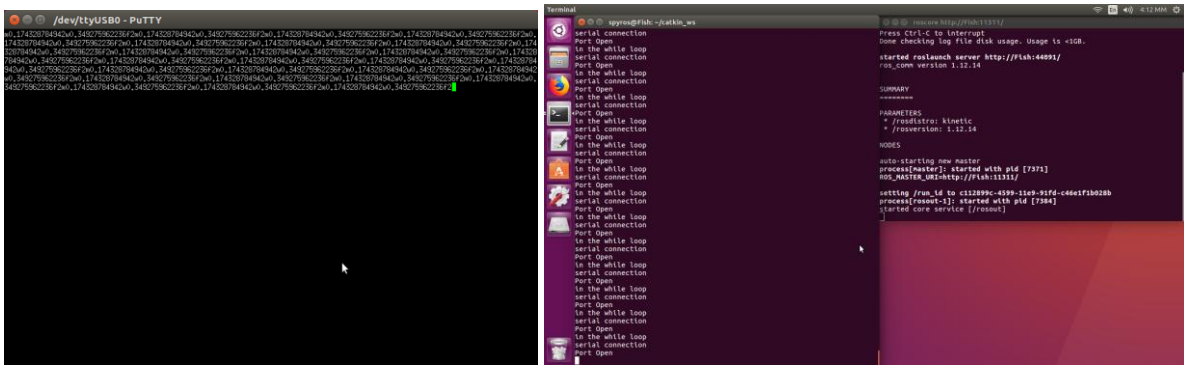
Ακολούθως, γίνεται φόρτωση του περιβάλλοντος ROS στον κεντρικό υπολογιστή (διαδικασία που απαιτείται για κάθε νέο τερματικό που ενεργοποιείται). Ελέγχονται οι παράμετροι της σειριακής θύρας στο εκτελέσιμο αρχείο serial_connection.py, έτσι ώστε το όνομα της θύρας να είναι αυτό που εντοπίστηκε κατά την σύνδεση του μετατροπέα USB – RS-232. Η συχνότητα αποστολής δεδομένων στη σειριακή θύρα μειώνεται από 200 Hz σε 1 Hz για ευκολότερη επιτήρηση των αποτελεσμάτων. Στη συνέχεια, με χρήση της εντολής:

```
catkin_make
```

γίνεται μεταγλώττιση (compiling) του εκτελέσιμου αρχείου, όπως περιγράφεται στην ενότητα 3.3.6. και σε ένα νέο τερματικό ενεργοποιείται ο κόμβος ROS Master. Με χρήση της εντολής:

```
rosrun usb_rs232 serial_connection.py
```

στο αρχικό τερματικό, εκτελείται ο κόμβος serial_connection (ενότητα 3.3.7). Μετά την πάροδο μερικών δευτερολέπτων, η επίλυση του συστήματος φτάνει στο τέλος της και οι μεταβλητές κίνησης του ουραίου πτερυγίου εμφανίζονται στο σειριακό τερματικό με τη μορφή γραμμών κειμένου (βλ. Σχήμα 5-3-i). Παράλληλα, στο τερματικό εκτέλεσης του κόμβου εμφανίζονται με την ίδια συχνότητα μερικές ενδεικτικές γραμμές κειμένου που υποδηλώνουν το σημείο του κώδικα που εκτελείται κάθε στιγμή (βλ. Σχήμα 5-3-ii).



(i) **(ii)**

Σχήμα 5-3. Αποτελέσματα εκτέλεσης κόμβου serial_connection: (i) Σειριακό τερματικό, (ii) Τερματικό εκτέλεσης κόμβου.

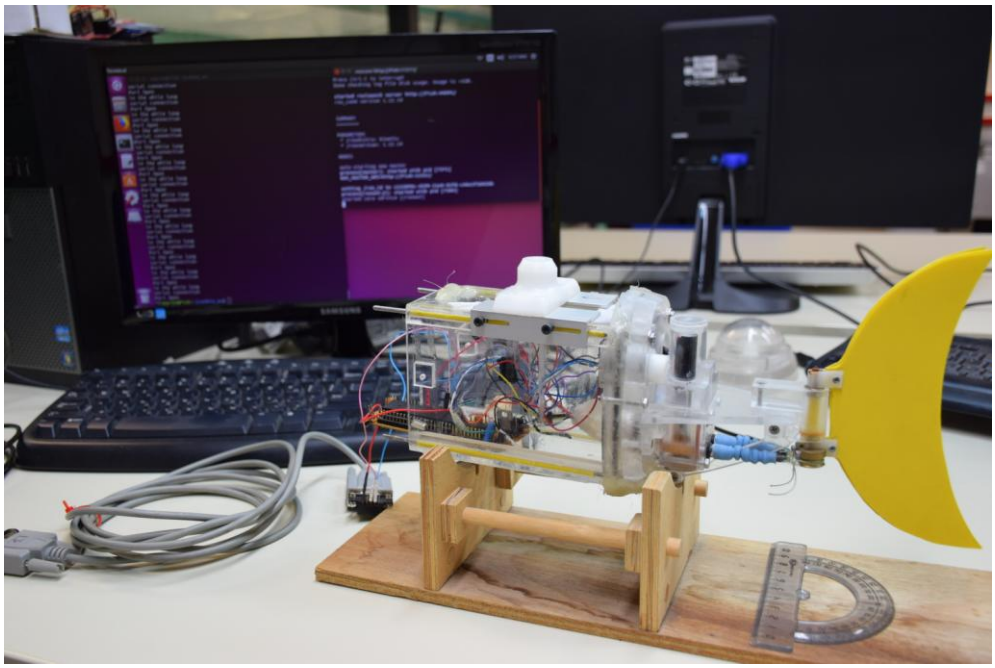
Αφήνοντας τον κόμβο ενεργοποιημένο για μερικά δευτερόλεπτα, εύκολα παρατηρείται ότι:

- Η συχνότητα αποστολής δεδομένων στο σειριακό τερματικό παραμένει σταθερή χωρίς διακυμάνσεις
- Οι γραμμές δεδομένων αποστέλλονται αυτούσιες, χωρίς να παραλείπονται δεδομένα ή να εμφανίζονται κενά
- Οι παράμετροι την κίνησης του ουραίου πτερυγίου που προκύπτουν από την επίλυση του συστήματος, επαληθεύουν τις αντίστοιχες τιμές που προκύπτουν από την προσομοίωση του συστήματος στο περιβάλλον Simulink [1].

Λαμβάνοντας υπόψη τα πιο πάνω αποτελέσματα, συμπεραίνεται ότι το υποσύστημα ελέγχου ανοικτού βρόχου ενσωματώθηκε με επιτυχία στο συνολικό σύστημα. Η αποστολή των δεδομένων μέσω σειριακής θύρας επιτυγχάνεται αξιόπιστα, με σταθερή συχνότητα και χωρίς απώλειες δεδομένων. Τώρα πλέον το υποσύστημα μπορεί να εφαρμοστεί με ασφάλεια, για αποστολή των δεδομένων στον μικροελεγκτή του ρομποτικού ψαριού.

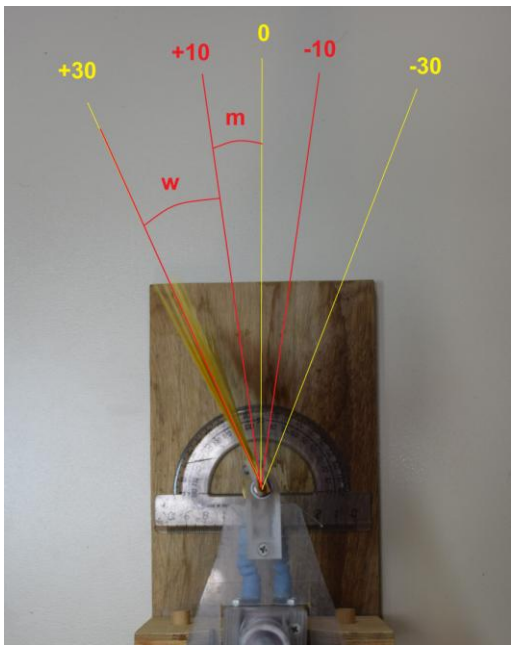
Δοκιμή στον μικροελεγκτή του ρομποτικού ψαριού

Η διαδικασία επαναλαμβάνεται, με τη διαφορά ότι στην θέση του φορητού υπολογιστή συνδέεται ο μικροελεγκτής του ρομποτικού ψαριού (βλ. Σχήμα 5-4), σύμφωνα με το διάγραμμα που παρουσιάζεται στο Σχήμα 3-27.



Σχήμα 5-4. Δοκιμή στον μικροελεγκτή του ρομποτικού ψαριού.

Στη συνέχεια, εκτελείται ο κόμβος serial_connection, αφού πρώτα η συχνότητα εκτέλεσης στο αρχείο serial_connection.py οριστεί στην αρχική της τιμή (200 Hz). Το ουραίο πτερύγιο του ρομποτικού ψαριού ξεκινά να ταλαντώνεται με εύρος ταλάντωσης 20^o και μέση γωνία 10^o (βλ. Σχήμα 5-5).



Σχήμα 5-5. Ταλάντωση ουραίου πτερυγίου.

Λαμβάνοντας υπόψη τα πιο πάνω αποτελέσματα, συμπεραίνεται ότι το υποσύστημα ελέγχου ανοικτού βρόχου λειτουργεί όπως αναμενόταν. Οι μεταβλητές κίνησης του ουραίου πτερυγίου αποστέλλονται στον μικροελεγκτή του ρομποτικού ψαριού με σταθερή συχνότητα, χωρίς απώλειες δεδομένων, ώστε το ρομποτικό ψάρι να εκτελεί την επιθυμητή τροχιά.

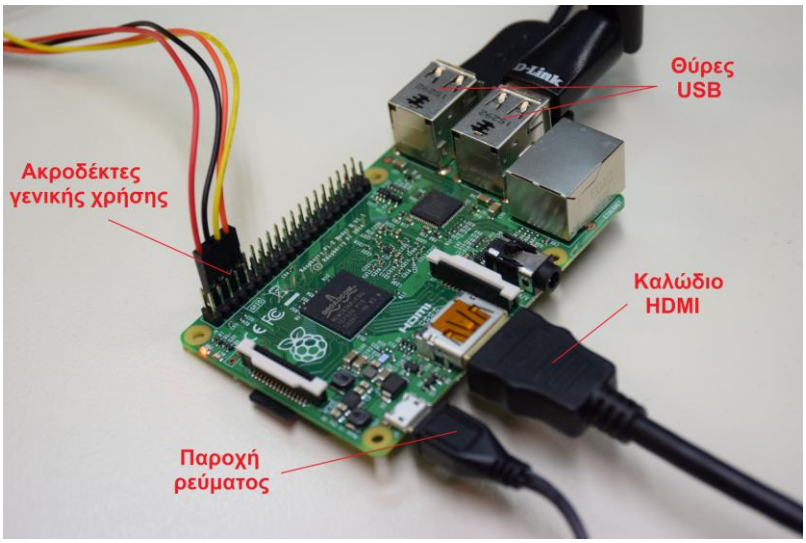
Ακολούθως, περιγράφεται η πειραματική διάταξη που χρησιμοποιήθηκε για δοκιμή του υποσυστήματος λήψης δεδομένων από περιφερειακή συσκευή.

5.2.2 Δοκιμή υποσυστήματος λήψης δεδομένων από περιφερειακή συσκευή

Σε πρώτο στάδιο, γίνεται η προετοιμασία του μικροϋπολογιστή Raspberry Pi, ο οποίος στέλνει τους έξι τυχαίους αριθμούς, υπό τη μορφή γραμμής κειμένου, όπως περιγράφεται στην ενότητα 3.3.2. Οι απαραίτητες περιφερειακές συσκευές του μικροϋπολογιστή συνδέονται στις αντίστοιχες θύρες (βλ. Σχήμα 5-6):

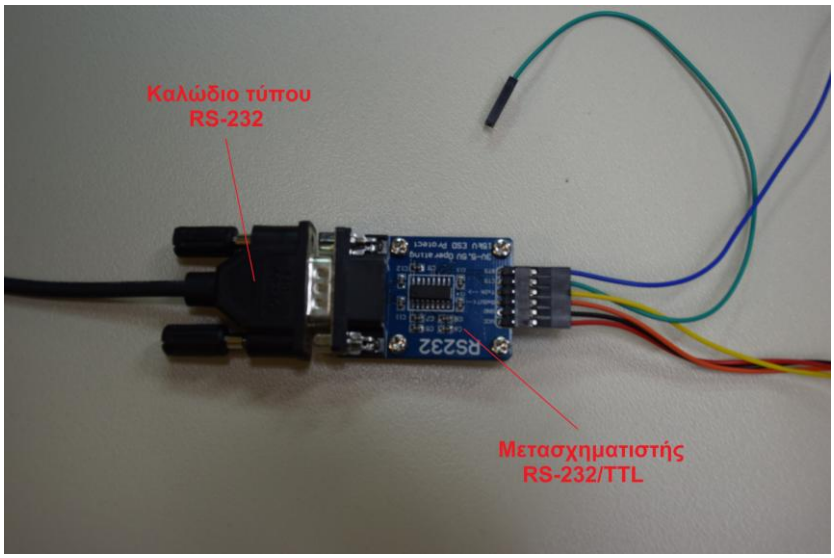
- Οθόνη: συνδέεται στη θύρα HDMI
- Πληκτρολόγιο: συνδέεται σε μία από τις τέσσερεις θύρες USB
- Ποντίκι: συνδέεται σε μία από τις τέσσερεις θύρες USB
- Ασύρματη σύνδεση δικτύου wi-fi: συνδέεται σε μία από τις τέσσερεις θύρες USB

Η ενεργοποίηση του μικροϋπολογιστή γίνεται αυτόματα με την σύνδεση της παροχής ρεύματος στην θύρα micro-USB. Με την πάροδο μερικών δευτερολέπτων, το περιβάλλον εργασίας του μικροϋπολογιστή εμφανίζεται στην συνδεδεμένη οθόνη.

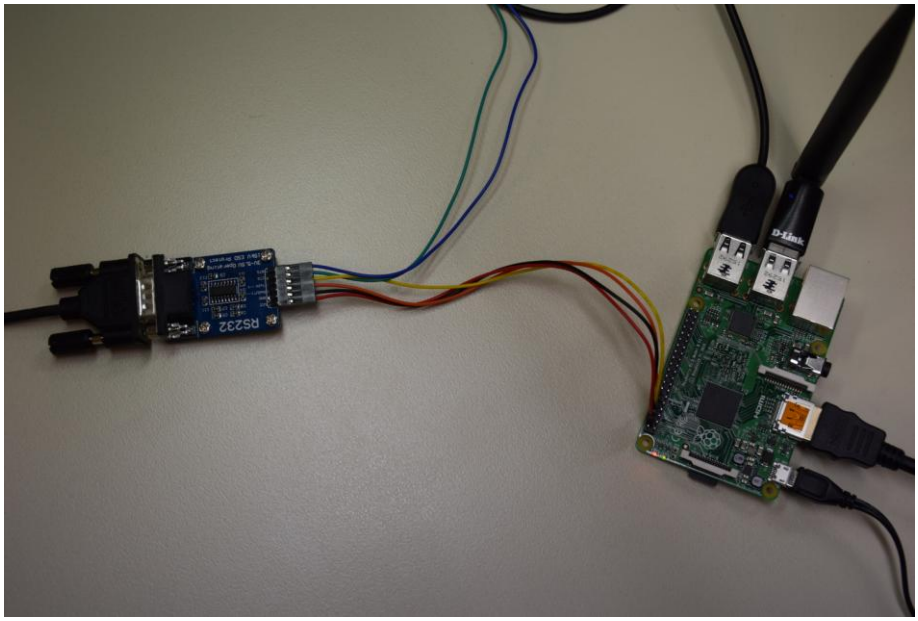


Σχήμα 5-6. Σύνδεση μικροϋπολογιστή Raspberry Pi.

Στη συνέχεια, ο σειριακός μετασχηματιστής RS-232/TTL συνδέεται στους ακροδέκτες γενικής χρήσης 14 και 15 του μικροϋπολογιστή σύμφωνα με το διάγραμμα συνδεσμολογίας (βλ. Σχήμα 3-27-i). Στο άλλο άκρο του μετασχηματιστή (ακροδέκτες τύπου DB9), συνδέεται ένα καλώδιο τύπου RS-232, το οποίο καταλήγει στη σειριακή θύρα του κεντρικού υπολογιστή (βλ. Σχήμα 5-7). Η συνολική διάταξη για τη δοκιμή του υποσυστήματος λήψης δεδομένων από σειριακή συσκευή παρουσιάζεται στο Σχήμα 5-8.



Σχήμα 5-7. Σύνδεση μετασχηματιστή RS-232/TTL.



Σχήμα 5-8. Συνολική διάταξη.

Ακολούθως, πραγματοποιούνται κάποιες αλλαγές στα εκτελέσιμα αρχεία των κόμβων. Για ευκολότερη επίβλεψη των αποτελεσμάτων, απενεργοποιούνται οι λειτουργίες αποστολής δεδομένων στη σειριακή θύρα και επίλυσης του συστήματος εξισώσεων από τον κόμβο serial_connection. Οι λειτουργίες αυτές αφορούν το υποσύστημα ελέγχου ανοικτού βρόχου και δεν απαιτούνται στην παρούσα δοκιμή. Η απενεργοποίηση των λειτουργιών υλοποιείται μετατρέποντας τις αντίστοιχες εντολές σε γραμμές κειμένου, προσθέτοντας πριν από κάθε εντολή τον χαρακτήρα “#” όπως παρουσιάζεται στη συνέχεια. Για την επίλυση του συστήματος εξισώσεων υπεύθυνη είναι η εντολή `solution()`, ενώ για την αποστολή των δεδομένων στη σειριακή θύρα υπεύθυνη είναι η εντολή `serial_connection()`. Με τον τρόπο αυτό το σύστημα αγνοεί τις συγκεκριμένες εντολές κατά την εκτέλεση του κόμβου.

```
1  if __name__ == '__main__':
2      try:
3          rospy.init_node('serial_connection', anonymous = True)
4          ser = serial.Serial('/dev/ttyUSB0', 115200)
5          print (ser.name)
6          time.sleep(2)
7          rospy.Subscriber('real_position', String, callback)
8          rate = rospy.Rate(1)
9          txt_to_arrays()
10       # solution()
11         while not rospy.is_shutdown():
12             print("in the while loop")
13           # serial_connection()
14             rate.sleep()
15         ser.close()
16     except rospy.ROSInterruptException:
17         pass
```

Στη συνέχεια, οι αλλαγές αποθηκεύονται στο εκτελέσιμο αρχείο με το υπάρχον όνομα (serial_connection.py). Επιπρόσθετα, αφού γίνει φόρτωση του περιβάλλοντος ROS σε ένα νέο τερματικό, στο εκτελέσιμο αρχείο camera_node.py, ελέγχεται το όνομα της σειριακής θύρας που χρησιμοποιείται (όπως περιγράφεται στο Κεφάλαιο 3), και ο ρυθμός μετάδοσης δεδομένων (baud rate) ώστε να είναι ίσος με αυτός που ορίζεται στο λογισμικό του Raspberry Pi. Αποθηκεύονται τυχόν αλλαγές με το υπάρχον όνομα και με εκτέλεση της εντολής:

```
catkin_make
```

(για κάθε τερματικό ξεχωριστά) όπως περιγράφεται στο Κεφάλαιο 3, υλοποιείται η μεταγλώττιση (compiling) των εκτελέσιμων αρχείων των κόμβων. Ο κόμβος ROS Master είναι ήδη ενεργοποιημένος από την προηγούμενη πειραματική δοκιμή και δεν χρειάζεται να ενεργοποιηθεί ξανά. Σε περίπτωση που για οποιονδήποτε λόγο ο κόμβος ROS Master απενεργοποιήθηκε, είναι απαραίτητο να ενεργοποιηθεί ξανά, με τον τρόπο που περιγράφεται στο Κεφάλαιο 3.

Ακολούθως, με εκτέλεση της εντολής:

```
rosrun usb_rs232 camera_node.py
```

ενεργοποιείται ο κόμβος camera_node και στο παράθυρο του τερματικού εμφανίζεται η γραμμή κειμένου:

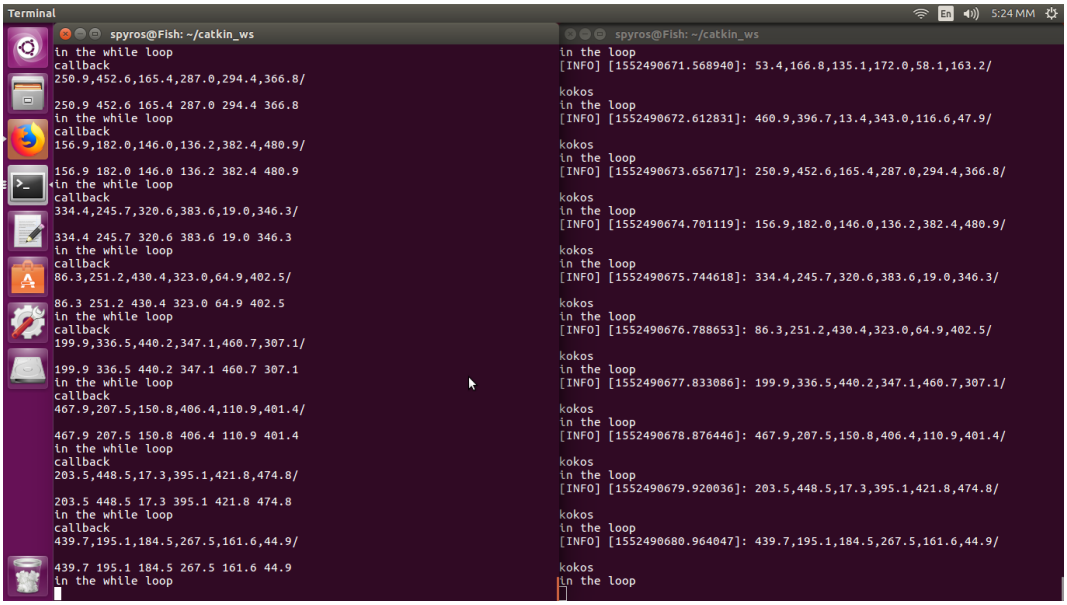
```
in the loop
```

Η γραμμή αυτή δηλώνει πως ο κόμβος έχει ενεργοποιηθεί χωρίς να προκύψει κάποιο σφάλμα και βρίσκεται σε κατάσταση αναμονής μέχρι την λήψη της πρώτης γραμμής δεδομένων από τη σειριακή θύρα. Στο λογισμικό του Raspberry Pi, ενεργοποιείται το ενσωματωμένο περιβάλλον

ανάπτυξης IDLE, και μέσω αυτού εκτελείται το αρχείο RasPi_code.py. Μετα την πάροδο μερικών δευτερολέπτων, οι γραμμές κειμένου με τους έξι τυχαίους αριθμούς (της μορφής που περιγράφεται στην ενότητα 3.3.2.) εμφανίζονται στο τερματικό του κόμβου camera_node, με διάστημα 1 s μεταξύ κάθε νέας γραμμής (βλ. Σχήμα 5-9-δεξιά). Με εκτέλεση της εντολής:

```
rosrun usb_rs232 serial_connection.py
```

στο τερματικό του κόμβου serial_connection, ο κόμβος ενεργοποιείται. Στην οθόνη του τερματικού αυτού εμφανίζονται οι ίδιες γραμμές κειμένου που εμφανίζονται στο τερματικό του κόμβου camera_node με την ίδια ακριβώς συχνότητα (βλ. Σχήμα 5-9-αριστερά).



Σχήμα 5-9. Αποτελέσματα εκτέλεσης κόμβων serial_connection και camera _node.

Οι γραμμές κειμένου με τους έξι τυχαίους αριθμούς που παράγονται στο λογισμικό του Raspberry Pi, αποστέλλονται αυτούσιες και με σταθερή συχνότητα στην κεντρική μονάδα ελέγχου μέσω σειριακής επικοινωνίας. Εκεί, λαμβάνονται από τον κόμβο camera_node, δημοσιεύονται στο topic real_position και στη συνέχεια λαμβάνονται από τον κόμβο serial_connection για περαιτέρω επεξεργασία.

Λαμβάνοντας υπόψη τα πιο πάνω αποτελέσματα, συμπεραίνεται ότι το υποσύστημα λήψης δεδομένων από περιφερειακή συσκευή μέσω σειριακής επικοινωνίας υλοποιήθηκε με επιτυχία. Η σειριακή επικοινωνία εξασφαλίζει αξιόπιστη επικοινωνία μεταξύ περιφερειακής συσκευής και κεντρικής μονάδας ελέγχου, με σταθερή συχνότητα μετάδοσης και χωρίς απώλειες δεδομένων. Το λειτουργικό σύστημα ROS διευκολύνει την μετάδοση δεδομένων μεταξύ των εκτελέσιμων αρχείων και παρέχει την δυνατότητα εύκολης επίβλεψης των επιμέρους λειτουργιών (εμφάνιση αποτελεσμάτων κάθε κόμβου σε διαφορετικό τερματικό).

6 Συμπεράσματα και Μελλοντική Εργασία

6.1 Συμπεράσματα

Η εργασία αυτή αποσκοπούσε πρωταρχικά στη δημιουργία μίας κεντρικής μονάδας ελέγχου με τη χρήση του λειτουργικού συστήματος ROS. Στην κεντρική μονάδα αυτή, ενσωματώνονται τα επιμέρους υποσυστήματα του ρομποτικού ψαριού, όπως ο έλεγχος ανοικτού βρόχου και η λήψη δεδομένων από περιφερειακή συσκευή. Δευτερεύον στόχος της εργασίας, ήταν ο σχεδιασμός και η κατασκευή βάσης τρίτου άξονα για το φορείο ρυμούλκησης του ρομποτικού ψαριού.

Όσον αφορά την επικοινωνία της κεντρικής μονάδας ελέγχου με το ρομποτικό ψάρι και την περιφερειακή συσκευή, υλοποιήθηκε με τη χρήση του σειριακού πρωτοκόλλου επικοινωνίας RS-232. Ως περιφερειακή συσκευή, χρησιμοποιήθηκε ο μικροϋπολογιστής Raspberry Pi. Σχετικά με το τεχνικό κομμάτι, η υλοποίηση της κεντρικής μονάδας ελέγχου πραγματοποιήθηκε επιτυχώς και μάλιστα με την απαίτηση ανεξάρτητης λειτουργίας των δύο υποσυστημάτων, αφού το καθένα μπορεί να λειτουργήσει ξεχωριστά ή να εφαρμοστεί αυτούσιο σε ένα άλλο σύστημα ROS.

Συμπεράσματα όμως εξάγονται και για τη χρήση του λειτουργικού συστήματος ROS στην οργάνωση και στο συντονισμό των επιμέρους λειτουργιών ενός ρομπότ (υλοποίηση ελέγχου ανοικτού βρόχου, αποστολή δεδομένων σε μικροεπεξεργαστή, λήψη δεδομένων από περιφερειακή συσκευή). Όπως φάνηκε από τα αποτελέσματα του συστήματος ελέγχου ανοικτού βρόχου, τα οποία επαληθεύουν τα αντίστοιχα αποτελέσματα προσομοίωσης στο περιβάλλον Simulink, το λειτουργικό σύστημα ROS παρέχει ένα αξιόπιστο περιβάλλον ανάπτυξης λογισμικού. Επιπρόσθετα, αποδεικνύεται ότι ROS αποτελεί ένα ευέλικτο και λειτουργικό τρόπο επικοινωνίας μεταξύ εκτελέσιμων αρχείων που βρίσκονται στο ίδιο σύστημα ανεξάρτητα από την γλώσσα προγραμματισμού στην οποία έχουν αυτά αναπτυχθεί. Η δομή του ROS διευκολύνει τον εντοπισμό σφαλμάτων κατά τον προγραμματισμό, και επιτρέπει τη δημιουργία αυτόνομων πακέτων λογισμικού, τα οποία μπορούν να εφαρμοστούν σε διάφορες ρομποτικές εφαρμογές, χωρίς να απαιτούνται σημαντικές αλλαγές.

Σχετικά με την αποστολή και λήψη δεδομένων σε περιφερειακές συσκευές (μικροελεγκτής, Raspberry Pi) το λειτουργικό σύστημα ROS αποδείχτηκε ιδιαίτερα αποτελεσματικό. Η δυνατότητα χρήσης σειριακής επικοινωνίας για μετάδοση δεδομένων καθιστά το ROS συμβατό με πληθώρα περιφερειακών συσκευών, ακόμα και όταν δεν είναι δυνατή η εγκατάσταση λογισμικού ROS σε αυτές (μικροελεγκτές). Σε εφαρμογές όπου δεν απαιτείται υψηλή ταχύτητα μετάδοσης, ο συνδυασμός λειτουργικού συστήματος ROS και σειριακής επικοινωνίας αποτελεί ικανοποιητική και αξιόπιστη λύση.

Όσον αφορά τον σχεδιασμό και την κατασκευή της βάσης τρίτου άξονα, από την διαδικασία αναζήτησης λύσεων μέχρι και την τελική κατασκευή, τα συμπεράσματα που εξήχθησαν αφορούσαν την δυσκολία επίτευξης ευθυγράμμισης μεταξύ άξονα του κινητήρα και ατράκτου του ρομποτικού ψαριού. Επίσης, στην επιλογή κατάλληλου στοιχείου κύλισης, καθοριστικό ρόλο έπαιξαν οι συνθήκες και το περιβάλλον λειτουργίας της βάσης. Τέλος, προκειμένου η κατασκευή να έχει το ελάχιστο δυνατόν βάρος και κόστος, επιλέχθηκαν υλικά χαμηλής πυκνότητας, που κατεργάστηκαν σχεδόν ως επί το πλείστον στο μηχανουργείο του Εργαστηρίου Αυτομάτου Ελέγχου.

6.2 Μελλοντική Εργασία

Με βάση τα συμπεράσματα που αναπτύχθηκαν, προτείνονται κάποιες βελτιώσεις και προσθήκες που θα μπορούσαν να αποτελέσουν μέρος μελλοντικής εργασίας, με στόχο την περαιτέρω βελτίωση τόσο της κεντρικής μονάδας ελέγχου όσο και των πειραματικών δοκιμών με την βάση τρίτου άξονα.

- Όσον αφορά τη λήψη δεδομένων από περιφερειακή συσκευή, θα ήταν χρήσιμο μία ψηφιακή κάμερα να συνδεθεί στο μικροϋπολογιστή Raspberry Pi. Η κάμερα αυτή, μπορεί να λαμβάνει βίντεο από την κάτοψη της δεξαμενής και μετά από κατάλληλη ψηφιακή επεξεργασία της εικόνας, να στέλλει στην κεντρική μονάδα ελέγχου την θέση και την ταχύτητα του ρομποτικού ψαριού σε πραγματικό χρόνο. Για την αποστολή των δεδομένων μπορεί να χρησιμοποιηθεί το υποσύστημα λήψης δεδομένων από περιφερειακή συσκευή που έχει υλοποιηθεί

- Επίσης, η ψηφιακή κάμερα μπορεί να χρησιμοποιηθεί ως μέσο ανατροφοδότησης για την υλοποίηση του συστήματος ελέγχου κλειστού βρόχου του ρομποτικού ψαριού. Αφού αναπτυχθεί το λογισμικό του κατάλληλου ελεγκτή (controller), αυτό μπορεί να ενσωματωθεί στο σύστημα σε ένα νέο κόμβο ο οποίος θα λαμβάνει τα δεδομένα ανατροφοδότησης από το topic real_position. Η υλοποίηση αυτή, επιτρέπει τον καθορισμό της επιθυμητής τροχιάς για το ρομποτικό ψάρι αλλά και την ελαχιστοποίηση του σφάλματος θέσης με τον υπολογισμό των κατάλληλων κερδών ελέγχου.

- Επιπρόσθετα, η μέθοδος σειριακής επικοινωνίας μεταξύ κεντρικού υπολογιστή και ρομποτικού ψαριού μπορεί να αντικατασταθεί με τη χρήση ακουστικών κυμάτων. Με κατάλληλη προσαρμογή υδροφώνων στο κέλυφος του ρομποτικού ψαριού και στα τοιχώματα της δεξαμενής πειραματικών δοκιμών, τα δεδομένα μπορούν να μεταδίδονται με τη μορφή ακουστικών κυμάτων και στη συνέχεια να αποκωδικοποιούνται από έναν ακουστικό διαμορφωτή (acoustic modem). Η εφαρμογή αυτή θα διευκολύνει τις πειραματικές δοκιμές στην δεξαμενή, αφού το εύρος κίνησης του ρομποτικού ψαριού δεν θα περιορίζεται από το μήκος του καλωδίου που χρησιμοποιείται.

- Όσον αφορά τις πειραματικές δοκιμές με τη χρήση της βάσης τρίτου άξονα, θα ήταν χρήσιμη η υλοποίηση του ελέγχου κίνησης του ηλεκτρικού κινητήρα που είναι υπεύθυνος για την περιστροφική κίνηση στον κατακόρυφο άξονα z. Αυτό θα διευκόλυνε την υλοποίηση πειραμάτων με μεταβλητή γωνία κίνησης του ρομποτικού ψαριού, για ακριβέστερη προσομοίωση της κίνησης του μέσα στο νερό.

7 Βιβλιογραφία

[1] M. Makrodimitris, K. Nanos, and E. Papadopoulos, “A novel trajectory planning method for a robotic fish,” *2017 25th Mediterr. Conf. Control Autom. MED 2017*, pp. 1119–1124, 2017.

[2] M. Quigley, E. Berger, and A. Y. Ng, “STAIR : Hardware and Software Architecture,” *AAAI 2007 Robot. Work. Vancouver, BC*, pp. 31--37, 2007.

[3] J. K. Salisbury, “Stanford University: Personal Robotics Program,” *personalrobotics.stanford.edu*. [Online]. Available: http://personalrobotics.stanford.edu/.

[4] K. A. Wyrobek, E. H. Berger, H. F. M. Van Der Loos, and J. K. Salisbury, “Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2165–2170, 2008.

[5] “ROS.org | Powering the world’s robots.” [Online]. Available: http://www.ros.org/. [Accessed: 16-Jan-2019].

[6] “Rock: The Robot Construction Kit.” [Online]. Available: https://www.rock-robotics.org/. [Accessed: 16-Jan-2019].

[7] “The Orocos Project | Smarter control in robotics & automation!” [Online]. Available: http://www.orocos.org/. [Accessed: 16-Jan-2019].

[8] W. D. S. Morgan Quigley, Brian Gerkey, *Programming Robots with ROS*. OREILLY, 2015.

[9] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” no. Figure 1, pp. 679–686.

[10] J. Klein and E. Papadopoulos, “A simulation framework for a hybrid leg-wheeled robot using Gazebo and ROS,” National Technical University of Athens, University of Duisburg-Essen, 2018.

[11] S. Cousins, B. Gerkey, K. Conley, and W. Garage, “Sharing Software with ROS [ROS Topics],” *Robot. Autom. Mag. IEEE*, vol. 17, no. 2, pp. 12–14, 2010.

[12] Open Source Robotics Foundation, “msg wiki,” 2017. [Online]. Available: http://wiki.ros.org/msg. [Accessed: 23-Jan-2019].

[13] “Robot for Research and Innovation - PR2,” *www.willowgarage.com*. [Online]. Available: http://www.willowgarage.com/pages/pr2/overview. [Accessed: 16-Jan-2019].

[14] S. Milica, “Underwater Acoustic Communication,” *Comput. Eng.*, 2015.

[15] B. Benson *et al.*, “Design of a Low-Cost Underwater Acoustic Modem,” *IEEE Embed. Syst. Lett.*, vol. 2, no. 3, pp. 58–61, 2010.

[16] “How is sounds used to transmit data underwater? – Discovery of Sound in the Sea,” *Discovery of Sound in The Sea*. [Online]. Available: https://dosits.org/people-and-sound/communication/how-is-sounds-used-to-transmit-data-underwater/. [Accessed: 19-Jan-2019].

[17] Ε. Αποστολόπουλος, “Υλοποίηση, Έλεγχος και Πειραματική Αξιολόγηση Τηλεχειριζόμενου Υποβρύχιου Οχήματος με Πρόωση από Ουραίο Πτερύγιο,” Εθνικό Μετσόβιο Πολυτεχνείο, 2005.

[18] W. Wang, J. Zhao, X. Wei, F. Cao, and G. Xie, “Underwater electric current communication of robotic fish: Design and experimental results,” *Proc. - IEEE Int. Conf. Robot. Autom.*, no. June, pp. 1166–1171, 2015.

[19] “Aluminium Magazine | Ενημερωτική έκδοση για τον κλάδο του αλουμινίου.” [Online]. Available: https://aluminium.gr/. [Accessed: 28-Jan-2019].

[20] “Bearings | MISUMI online shop - Select, configure, order.” [Online]. Available: https://uk.misumi-ec.com/vona2/mech/M0800000000/M0802000000/. [Accessed: 29-Jan-2019].

[21] “Shaft Couplings | MISUMI online shop - Select, configure, order.” [Online]. Available: https://uk.misumi-ec.com/vona2/mech/M1000000000/M1001000000/. [Accessed: 28-Jan-2019].

[22] NTN Bearings, “Ball and Roller Bearings,” no. 2500. NTN, pp. 52–53, 2015.

[23] S. Joyeux, R. Alami, S. Lacroix, and R. Philippsen, “A plan manager for multi-robot systems,” *Int. J. Rob. Res.*, vol. 28, no. 2, pp. 220–240, 2009.

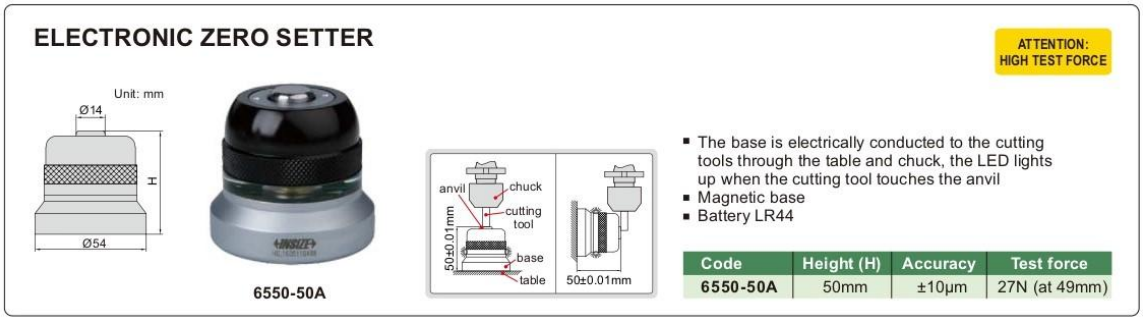
[24] “OpenRobotsFrontPage - Openrobots.” [Online]. Available: https://www.openrobots.org/wiki/OpenRobotsFrontPage?action=fullsearch&context=180&valu

e=Generator+of+modules&titlesearch=Titles. [Accessed: 16-Jan-2019].
[25] V. Tsounis, “Σχεδιασμός και Υλοποίηση Σερβομηχανισμού Ρυμούλκησης Υποβρυχίων Ρομποτικών Συστημάτων Διπλωματική Εργασία,” National Technical University of Athens, 2014.
[26] “PR1 - ROBOTS: Your Guide to the World of Robotics,” 2008. [Online]. Available: https://robots.ieee.org/robots/pr1/. [Accessed: 07-Mar-2019].
[27] P. H. Milne, *Underwater acoustic positioning systems*. Gulf Pub. Co, 1983.
[28] C. Fairchild and T. L. Dr. Harman, *ROS Robotics by Example: Learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame*, Second. Packt Publishing Ltd, 2017.
[29] A. Martinez, A. Mahtani, L. S. Crespo, and P. Fernandez, *Learning ROS for Robotic Programming*, Second. Packt Publishing Ltd, 2015.
[30] “CCS, Inc. - PCWHD IDE C Compiler for Microchip PIC.” [Online]. Available: http://www.ccsinfo.com/product_info.php?products_id=PCWHD_full. [Accessed: 12-Mar-2019].
[31] M. Makrodimitris, I. Aliprantis, and E. Papadopoulos, “Design and implementation of a low cost, pump-based, depth control of a small robotic fish,” *IEEE Int. Conf. Intell. Robot. Syst.*, no. Iros, pp. 1127–1132, 2014.
[32] Microchip Technology, “Pin Enhanced Flash Microcontrollers with nanoWatt Technology,” *Retrieved from http//ww1.microchip.com/downloads/en/devicedoc/30487c.pdf*, pp. 221–242, 2006.
[33] “D-sub 9 Connector Pinout.” [Online]. Available: https://www.db9-pinout.com/. [Accessed: 07-Mar-2019].
[34] “Raspberry Pi 2 Model B - Raspberry Pi.” [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Accessed: 07-Mar-2019].
[35] “RS-232 vs. TTL Serial Communication - SparkFun Electronics,” 2010. [Online]. Available: https://www.sparkfun.com/tutorials/215. [Accessed: 07-Mar-2019].
[36] “RS232 Communication Board - GROBOTRONICS.” [Online]. Available: https://grobotronics.com/rs232-communication-board.html. [Accessed: 07-Mar-2019].
[37] “std_msgs - ROS Wiki.” [Online]. Available: http://wiki.ros.org/std_msgs. [Accessed: 06-Mar-2019].
[38] K. Conley, “rospy - ROS Wiki.” [Online]. Available: http://wiki.ros.org/rospy. [Accessed: 06-Mar-2019].
[39] “Licenses by Name | Open Source Initiative.” [Online]. Available: https://opensource.org/licenses/alphabetical. [Accessed: 06-Mar-2019].
[40] “ROS/Tutorials/CreatingPackage - ROS Wiki.” [Online]. Available: http://wiki.ros.org/ROS/Tutorials/CreatingPackage#ROS.2BAC8-Tutorials.2BAC8-catkin.2BAC8-CreatingPackage.Customizing_Your_Package. [Accessed: 06-Mar-2019].
[41] “Writing a Simple Publisher and Subscriber (C++).” [Online]. Available: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29. [Accessed: 06-Mar-2019].
[42] “How to Install Pip on Ubuntu 16.04 LTS | Liquid Web Knowledge Base.” [Online]. Available: https://www.liquidweb.com/kb/how-to-install-pip-on-ubuntu-16-04-lts/. [Accessed: 08-Mar-2019].
[43] “Installing packages — SciPy.org.” [Online]. Available: https://www.scipy.org/install.html#scientific-python-distributions. [Accessed: 08-Mar-2019].
[44] “scipy.integrate.solve_ivp — SciPy v1.2.1 Reference Guide,” 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html. [Accessed: 09-Mar-2019].
[45] “rostopic - ROS Wiki.” [Online]. Available: http://wiki.ros.org/rostopic. [Accessed: 09-Mar-2019].
[46] “Setting up your Raspberry Pi - Introduction | Raspberry Pi Projects.” [Online]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up. [Accessed: 10-Mar-2019].
[47] “Read and Write From Serial Port With Raspberry Pi (with Pictures).” [Online]. Available: https://www.instructables.com/id/Read-and-write-from-serial-port-with-Raspberry-Pi/. [Accessed: 10-Mar-2019].
[48] “DC motors and drive systems by maxon motor.” [Online]. Available: https://www.maxonmotor.com/maxon/view/content/index. [Accessed: 28-Jan-2019].

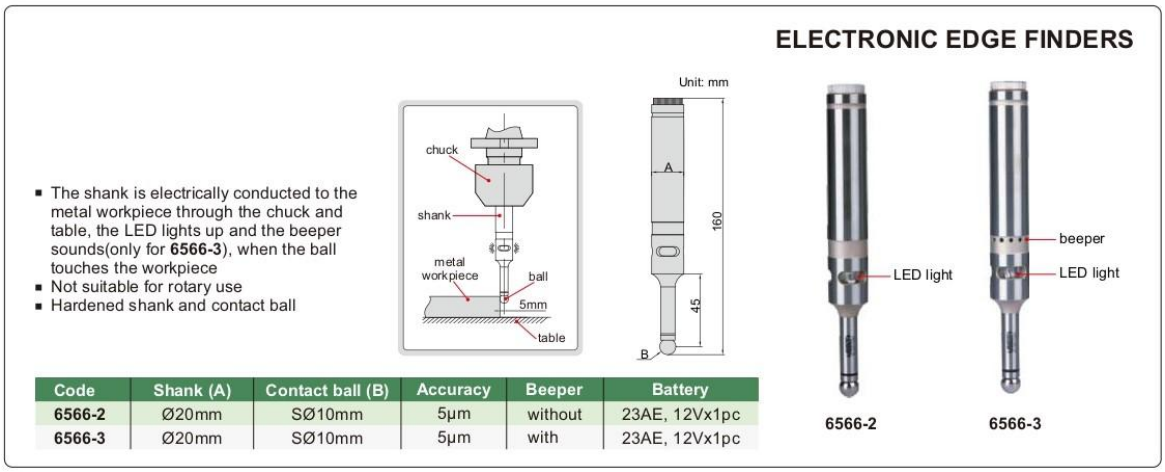
8 Παράρτημα Α

Στο παρόν παράρτημα παρατίθενται τα τεχνικά χαρακτηριστικά από τα εξαρτήματα που χρησιμοποιήθηκαν, το συνολικό περιεχόμενο του εκτελέσιμου αρχείου serial_connection.py και η διαδικασία εξαγωγής του κώδικα στην γλώσσα προγραμματισμού αριθμητικού ελέγχου (G-code), από το λογισμικό SolidCAM.

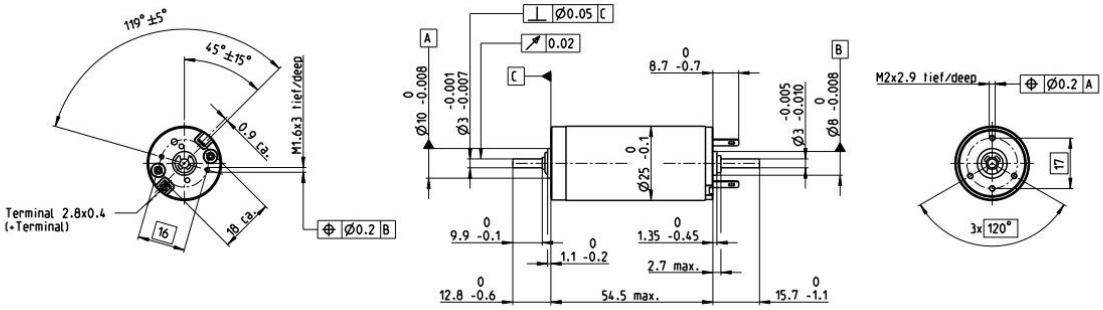
Ηλεκτρονικός ρυθμιστής σημείου μηδέν



Ηλεκτρονικό εργαλείο εύρεσης ακμής



RE 25 ∅25 mm, Precious Metal Brushes CLL, 10 Watt



M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers

	118740	118741	118742	118743	118744	118745	118746	118747	118748
Motor Data									
Values at nominal voltage									
1 Nominal voltage V	4.5	8	9	12	15	18	24	32	48
2 No load speed rpm	5360	5320	5230	4850	4980	4790	5190	5510	5070
3 No load current mA	79.7	44.4	38.7	26.3	21.8	9.88	14.4	11.7	6.96
4 Nominal speed rpm	4980	4520	4220	3800	3920	3710	4130	4450	4000
5 Nominal torque (max. continuous torque) mNm	11.4	20.9	23.9	28.6	28.2	28.7	28	27.9	27.9
6 Nominal current (max. continuous current) A	1.5	1.5	1.5	1.24	1.01	0.811	0.652	0.516	0.317
7 Stall torque mNm	131	132	119	129	131	126	136	144	132
8 Stall current A	16.5	9.23	7.31	5.5	4.57	3.52	3.1	2.61	1.47
9 Max. efficiency %	87	87	86	87	87	90	87	87	87
Characteristics									
10 Terminal resistance Ω	0.273	0.867	1.23	2.18	3.28	5.11	7.73	12.3	32.6
11 Terminal inductance mH	0.0275	0.0882	0.115	0.238	0.353	0.551	0.832	1.31	3.48
12 Torque constant mNm/A	7.99	14.3	16.3	23.5	28.6	35.8	43.9	55.2	89.9
13 Speed constant rpm/V	1200	668	584	406	334	267	217	173	106
14 Speed / torque gradient rpm/mNm	40.9	40.5	44	37.7	38.3	38.2	38.3	38.5	38.6
15 Mechanical time constant ms	4.99	4.4	4.37	4.25	4.23	4.22	4.22	4.22	4.23
16 Rotor inertia gcm²	11.7	10.4	9.49	10.8	10.6	10.6	10.5	10.5	10.5

Specifications

Thermal data

17	Thermal resistance housing-ambient	14 K/W
18	Thermal resistance winding-housing	3.1 K/W
19	Thermal time constant winding	12.5 s
20	Thermal time constant motor	612 s
21	Ambient temperature	-20…+85°C
22	Max. winding temperature	+100°C

Mechanical data (ball bearings)

23	Max. speed	5500 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.2 N
27	Max. force for press fits (static) (static, shaft supported)	64 N 800 N
28	Max. radial load, 5 mm from flange	16 N

Other specifications

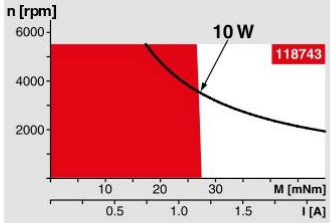
29	Number of pole pairs	1
30	Number of commutator segments	11
31	Weight of motor	130 g

CLL = Capacitor Long Life

Values listed in the table are nominal.
Explanation of the figures on page 64.

Option
Preloaded ball bearings

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

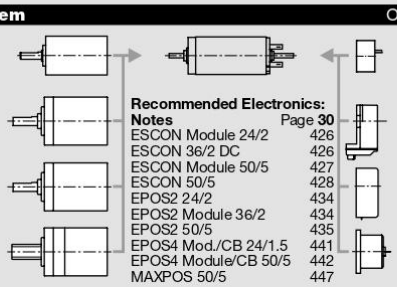
Overview on page 28–36

Planetary Gearhead
∅26 mm
0.75 - 4.5 Nm
Page 332

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 334/335/338

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 343

Spindle Drive
∅32 mm
Page 366–368



Recommended Electronics:

Notes	Page 30
ESCON Module 24/2	426
ESCON 36/2 DC	426
ESCON Module 50/5	427
ESCON 50/5	428
EPOS2 24/2	434
EPOS2 Module 36/2	434
EPOS2 50/5	435
EPOS4 Mod./CB 24/1.5	441
EPOS4 Module/CB 50/5	442
MAXPOS 50/5	447

Encoder MR
128 - 1000 CPT,
3 channels
Page 404

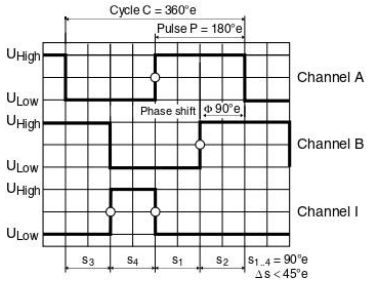
Encoder Enc
22 mm
100 CPT, 2 channels
Page 411

Encoder HED_ 5540
500 CPT,
3 channels
Page 413/415

DC-Tacho DCT
∅22 mm
0.52 V
Page 421

maxon sensor

Encoder MR Type ML, 128–1000 CPT, 3 Channels, with Line Driver



Direction of rotation cw (definition cw p. 60)

Stock program
Standard program
Special program (on request)

Part Numbers	225771	225773	225778	225805	225780
Type					
Counts per turn	128	256	500	512	1000
Number of channels	3	3	3	3	3
Max. operating frequency (kHz)	80	160	200	320	200
Max. speed (rpm)	37500	37500	24000	37500	12000

maxon Modular System

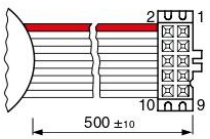
+ Motor	Page	+ Gearhead	Page	Ø Enc [mm]	Overall length [mm] / ● see Gearhead				
RE 25	125/127			25	65.5	65.5	65.5	65.5	65.5
RE 25	125/127	GP 26, 0.75 - 2.0 Nm	332	25	●	●	●	●	●
RE 25	125/127	GP 32, 0.75 - 6.0 Nm	334-339	25	●	●	●	●	●
RE 25	125/127	KD 32, 1.0 - 4.5 Nm	343	25	●	●	●	●	●
RE 25	125/127	GP 32 S	366-368	25	●	●	●	●	●
RE 25, 20 W	126			25	54.0	54.0	54.0	54.0	54.0
RE 25, 20 W	126	GP 22, 0.5 Nm	325	25	●	●	●	●	●
RE 25, 20 W	126	GP 26, 0.75 - 2.0 Nm	332	25	●	●	●	●	●
RE 25, 20 W	126	GP 32, 0.75 - 6.0 Nm	334-339	25	●	●	●	●	●
RE 25, 20 W	126	KD 32, 1.0 - 4.5 Nm	343	25	●	●	●	●	●
RE 25, 20 W	126	GP 32 S	366-368	25	●	●	●	●	●
A-max 26	152-158			25	53.5	53.5	53.5	53.5	53.5
A-max 26	152-158	GP 26, 0.75 - 4.5 Nm	332	25	●	●	●	●	●
A-max 26	152-158	GS 30, 0.07 - 0.2 Nm	333	25	●	●	●	●	●
A-max 26	152-158	GP 32, 0.75 - 6.0 Nm	334-339	25	●	●	●	●	●
A-max 26	152-158	GS 38, 0.1 - 0.6 Nm	344	25	●	●	●	●	●
A-max 26	152-158	GP 32 S	366-368	25	●	●	●	●	●
EC-max 30, 40 W	230			25			54.2		54.2
EC-max 30, 40 W	230	GP 32, 1 - 8.0 Nm	339/341	25			●		●
EC-max 30, 40 W	230	KD 32, 1.0 - 4.5 Nm	343	25			●		●
EC-max 30, 40 W	230	GP 32 S	366-368	25			●		●
EC-max 30, 60 W	231			25			76.2		76.2
EC-max 30, 60 W	231	GP 32, 1 - 8.0 Nm	339/341	25			●		●
EC-max 30, 60 W	231	KD 32, 1.0 - 4.5 Nm	343	25			●		●
EC-max 30, 60 W	231	GP 42, 3 - 15 Nm	346	25			●		●

Technical Data

Supply voltage V_{CC}	5 V ± 5%
Typical current draw	14 mA
Output signal	TTL compatible
Phase shift Φ	90°e ± 45°e
Index pulse width	90°e ± 45°e
Operating temperature range	-25…+85 °C
Moment of inertia of code wheel	≤ 0.7 gcm²
Output current per channel	max. 5 mA

The index signal I is synchronized with channel A or B.

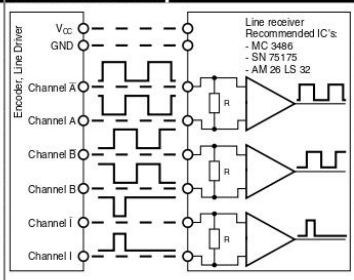
Pin Allocation



1	N.C.
2	V_{CC}
3	GND
4	N.C.
5	Channel $\overline{A}$
6	Channel A
7	Channel $\overline{B}$
8	Channel B
9	Channel $\overline{I}$ (Index)
10	Channel I (Index)

DIN Connector 41651/
EN 60603-13
flat band cable AWG 28

Connection example



Opt. terminal resistance R > 1 kΩ

Μέθοδος επιλογής ελαστικού συνδέσμου

Coupling Selection Method

When Selection is Based on Motor

1 Please select appropriate coupling type based on the motor type and the application in which it is to be used.

* Motor type and Coupling type are not limited to the following combinations. Select after confirming each product page.
* When selection is by torque, start from 2.

Coupling Easy Selection Chart

	Coupling Characteristics			Motor		
Type	Zero Backlash	High Torque	Allowable Lateral Misalignment Allowable Angular Misalignment	Servo	Stepping Motor Compact Servo	General-purpose
Disc	◎	◎	○	◎	○	×
Oldham	×	◎	◎	×	×	◎
Slit	◎	○	○	○	◎	×

Selection Example)

<Prerequisites>

Rotating Direction: One-way rotation
Applications: Transfer Conveyor (Positioning not required)
Motor: General-purpose Motor

<Selection>

As positioning is not required in one-way rotation, zero backlash is not required. General-purpose Motor is used and Oldham Couplings can be selected.

2 Calculate the compensation torque applied to coupling

Connection with Servo Motors / Stepping Motors

Calculate the Compensation Torque by multiplying Compensation Factor with the Motor's Peak Torque. For Compensation Factor, please see products page. Choose a coupling with the torque capacity (shaft slip torque) higher than the calculated compensation torque.

Compensation Torque = Motor's Peak Torque x Compensation Factor

Connection with General-purpose Motor

Calculate the load torque, and the compensation torque should be 1-5 times of the load torque. Choose a coupling with the torque capacity (shaft slip torque) higher than the calculated compensation torque.

$$\text{Load Torque (N} \cdot \text{m)} = 9550 \times \frac{\text{Transmission Power (kW)}}{\text{Rotational Speed (r/min)}}$$

Compensation Torque = Load Torque (N · m) x 1~5

3 Check coupling tolerance

Check that the Tolerance (Angular and Lateral misalignments and Max. Rotational Speed) and the Moment of Inertia mentioned in the catalogue meet the conditions of the device.

4 Select shaft bore

Check if the outer diameter of the connecting shaft is included in the inner diameter range of the coupling. If the outer diameter is not included, select a large size.

5 Select shaft connection method

Select in accordance with the clamp, keyed locking, keyless clamping applications.

6 Final confirmation

Finally check the dimension table to confirm that the coupling is compatible with the device.

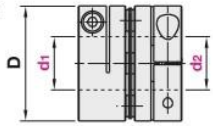
How to Create Coupling Part Number

Part Number (Type, D)	-	I.D. 1 (d_1)	-	I.D. 2 (d_2)	$d_1 \leq d_2$

Ex.) TYPE: GCPS When D=33 d_1=10 d_2=11

GCPS33-10-11

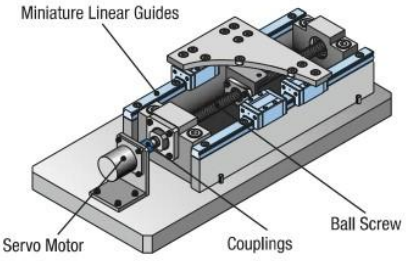
TYPE



Selection Example

Select on the basis of the following Example.

Example



<Prerequisites>

Drive Motor: Servo Motor, Driven Side: Ball Screw
Rotational Direction: Reverse Rotary, Rotational Speed: 3000rpm
Lateral Misalignment: 0.1mm Angular Misalignment: 1° Axial Misalignment: 0.1mm
Shaft Bore Dia. Motor Side: 14mm, Ball Screw Side: 15mm
Coupling O.A.L.: 60mm or less

1. Select Coupling Type
   Required characteristics of coupling from the above conditions
   - Zero Backlash
   - Allow lateral/angular misalignments

   Refer to the above Coupling Easy Selection Chart, and select.

   Compatible Coupling Type ⇒ Disc Couplings: GCPW

2. Calculate the compensation torque applied to coupling
   Condition: Servo Motor's Peak Torque: 3.0N · m
   Servo Motor's Rated Torque: 1.0N · m
   Calculation of Compensation Torque
   Compensation Torque = Motor's Peak Torque (3.0N · m) x Compensation Factor (2.0) = 6.0N · m

   Select D(O.D.) with allowable torque 6.0N · m or more ⇒ GCPW39

* Compensation factor is for reference when using couplings with servo motors in general. Please use the values as reference.

3. Check coupling tolerance
   Allowable Lateral Misalignment: 0.25mm Allowable Angular Misalignment: 1° Allowable Axial Misalignment: ±0.5mm
   Allowable Rotational Speed: 10000rpm
   ⇒ Conditions are met

4. Select shaft bore
   Check if 14mm and 15mm shaft bore can be selected in GCPW39.
   14mm and 15mm both fall under the category of D (O.D.) = 39 specification.

   d_1=14, d_2=15⇒ GCPW39-14-15

5. Selecting the method to fasten shaft bore

   ⇒ GCPW has only clamp, and need not be selected

6. Check that the dimension matches the device
   Overall Length: 49.6mm
   ⇒ Matches with the overall length 60mm or less

   Finally selected Part Number ⇒ GCPW38-14-15

Terminology for Couplings·Assembly Procedure

Terminology

·Allowable Torque

A torque that a coupling can continuously transmit. Select a allowable torque according to selection method given on **P.1061** which would ensure that using load torque is less than the allowable torque of the coupling. In couplings for Servo Motors, compensation factor recommended for each part number is set.

·Slip Torque

A torque which spins the fastened shaft and coupling, and slides out. Select the shaft slip torque when it is less than the allowable torque according to the selection method given on **P.1061**, such that the load torque is less than the shaft slip torque of the coupling.

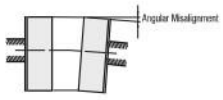
·Misalignment

An error between 2 coupling shaft centers. Misalignments are: Angular Misalignment, Lateral Misalignment and Axial Misalignment. Align (Centering) the shafts to ensure that misalignment between two shafts is less than the mentioned allowable misalignment. When two or more misalignments are combined, the allowable value for respective misalignment is 1/2. (Explanation regarding misalignment is as follows)

Angular Misalignment

Angle Error between two coupling shafts. (Fig. 1)

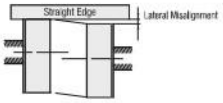
Fig. 1 Angular Misalignment
(Angle Error between Two Shaft Centers)



Lateral Misalignment

Angle Error between two coupling shafts. (Fig. 2)

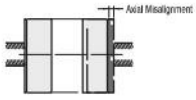
Fig. 2 Lateral Misali ent
(Parallelism Error between Two Shaft Centers)



Axial Misalignment

Displacement in the axial direction of various shafts. (Fig. 3): This misalignment occurs at the time of motor acceleration or, due to swelling at the time of rise in temperature.

Fig. 3 Axial Misalignment
(Shaft's Shift in the Axial Direction)



·Static Torsional Spring Constant (Torsional Rigidity)

Torsional rigidity of the coupling. It shows the phase difference between the rotating direction of input shaft and output shaft, when torque is applied to the coupling. The value given in the catalogue indicates the torsional rigidity for the whole coupling. The responsiveness increases as this value increases, and highly-precise rotation control becomes possible.

·Max. Rotational Speed

Highest available rotational speed. Balancing is necessary when using for high speed rotation, as dynamic balance is not considered in this value.

·Moment of Inertia

Inertia moment of the coupling. Rotatory inertia increases in respect to increase in the value of inertia moment.

·Zero Backlash

A rattling noise occurring in various parts of the coupling, corresponding to the rotating direction. When using Servo Motors, consider using disk coupling or slit couplings with zero backlash for the purpose of highly precise positioning, clock-wise and anti-clockwise rotation.

·Tightening Torque

A bolt tightening torque to fasten the coupling to the shaft. Use torque wrench to tighten the coupling to prescribed torque.

·Temperature Correction Factor

Resin spacer couplings such as Oldham and Jaw Types have allowable torque values that vary depending on the operating temperature. Select couplings by multiplying couplings Allowable Torque, Max. Rotational Speed, etc by Temperature Compensation Factor in the right hand table.

Operating Temperature (°C)	Temperature Correction Factor
-20~ 30	1.00
30~ 40	0.80
40~ 60	0.70
60~100	0.55

Assembly Procedure

1) Confirm that the clamping screws are loosened, and wipe clean the inner bore and shaft surfaces off dust and oils.
2) Insert the shaft into the coupling while taking care not to apply excessive compressive/tensile forces on the disc section.
3) Adjust the disk coupling in left-right hub concentration in precise manner, using the jig. Quickly check the angular and lateral misalignment using coupling as a base.

3-1)Quick Lateral Misalignment Check (Fig. 4): Slide the coupling in axial direction while the bolts are loosened and check that it is moving smoothly. Center the coupling properly as the single type coupling does not allow lateral misalignment.

3-2)Quick Angular Misalignment Check (Fig. 5): Rotate the coupling/shaft and visually check for smooth and even movement.

4)Final assembly: Adjust the shaft insertion as per the dimensions given in the catalogue (Fig. 6) and fasten it with prescribed torque by using the torque wrench.ℓ
Do not fasten till the prescribed torque at once. Alternate between left and right clamp and tighten it two or three times.

Fig. 4 Quick Lateral Misalignment Check

Fig. 5 Quick Angular Misalignment Check

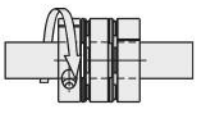
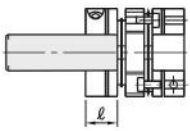


Fig. 6 Shaft Insertion Depth



Slit Couplings

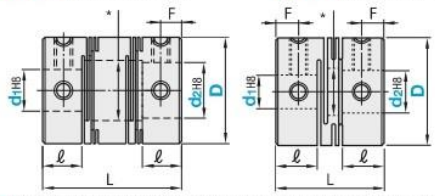
Set Screw, Short / Long

Points of comparison between similar products | Max. Rotational Speed: 19,000~78,000rpm

Features: Because backlash is 0, it is suitable for applications where rotation accuracy is required.

Slit Set Screw

CPL (Aluminum - Long)
CPLS (Stainless Steel - Long)
CPS (Aluminum - Short)
CPSS (Stainless Steel - Short)



*d1, d2 Identical Diameter = d1+0.5
d1, d2 Different Diameters = Large Shaft Diameter ÷ 0.5
* When d1 is 4mm or less and d2 is 5mm or more, there are 3 set screws.
* When both d1 and d2 are less than 4mm, there are 2 set screws.

Type	M Material	S Surface Treatment	A Accessory
CPL, CPS	Aluminum Alloy	Clear Anodize	Set Screw
CPLS, CPSS	Stainless Steel	-	

The lateral, angular, and axial misalignment values shown are for each occurring individually. When multiple misalignments are occurring simultaneously, the allowable maximum value of each will be reduced to 1/2.
For the selection criteria and alignment procedures, see P.1061.

Part Number Type	D	d1	d2	L CPL CPLS	L CPS CPSS	ℓ CPL CPLS	ℓ CPS CPSS	M (Coarse)	F	Unit Price CPL	CPLS	CPS	CPSS
CPL (Aluminum) CPLS (Stainless Steel) CPS (Aluminum) CPSS (Stainless Steel)	8	*2	*2 3	14	10	3.5	3.4	M2	1.7				
		*3	*3										
	12	3	3 4	18.5	14	5	5.2	M2.5	2.5				
		*4	*4 *5										
		*5	*5 6										
	16	4	4 5 6	23	18	6.5	6.8						
		*5	*5 *6 8										
		*6	*6 6.35 8										
		6.35	8										
	20	*5	5 *6 *8	26	20	7.5	7.65	M3	3				
		*6	*6 6.35 7 *8 10										
		6.35	8										
		*8	*8 10										
		10	10										
	25	*5	*6	31	25	8.5	9.6	M4	4				
		*6	*6 6.35 *8 10										
		6.35	8 10										
		*8	*8 *10 12										
		9.525	10										
		*10	*10 12										
		12	12										
	32	6	8	41	32	12	12.6		6				
		6.35	8										
		*8	*8 *10 12										
		*10	*10 11 *12 14										
		*12	*12 *14										
		14	14 16										
	40	8	9.525	56	-	17	-	M5	8.5			-	-
		10	10										
		12	12										
		14	14										
		15	15										
		16	16										
		18	18										

CPS and CPSS are available in * marked sizes only.

Part Number Type	D	Allowable Torque (N·m)	Max. Rotational Speed (r/min)	Moment of Inertia (kg·m²)	Static Torsional Spring Constant (N·m/rad)	Lateral Misalignment (mm)	Angular Misalignment (°)	Allowable Axial Misalignment (mm)	Screw Tightening Torque (N·m)	Mass (g)
CPL (Aluminum)	8	0.1	78000	1.2×10^{-8}	25	0.10	2	±0.2	0.3	1.4
	12	0.4	52000	8.3×10^{-8}	45			±0.3	0.5	3.7
	16	0.5	39000	3.3×10^{-7}	80			±0.4	0.7	8.1
	20	1	31000	9.0×10^{-7}	170					14
	25	2	25000	2.6×10^{-6}	380	0.15		±0.5	1.7	27
	32	4	19000	9.6×10^{-6}	500					60
	40	8	15000	3.2×10^{-5}	700	0.20			4	130

Part Number Type	D	Allowable Torque (N·m)	Max. Rotational Speed (r/min)	Moment of Inertia (kg·m²)	Static Torsional Spring Constant (N·m/rad)	Angular Misalignment (°)	Allowable Axial Misalignment (mm)	Screw Tightening Torque (N·m)	Mass (g)
CPS (Aluminum)	8	0.1	78000	1.0×10^{-8}	24	1	±0.1	0.3	1
	12	0.4	52000	7.0×10^{-8}	80			0.5	3.1
	16	0.5	39000	2.8×10^{-7}	180		±0.2	0.7	7.4
	20	1	31000	7.5×10^{-7}	200				12
	25	2	25000	2.3×10^{-6}	780			1.7	24
	32	4	19000	8.0×10^{-6}	1100				50

CPS does not allow eccentricity.

Part Number Type	D	Allowable Torque (N·m)	Max. Rotational Speed (r/min)	Moment of Inertia (kg·m²)	Static Torsional Spring Constant (N·m/rad)	Lateral Misalignment (mm)	Angular Misalignment (°)	Allowable Axial Misalignment (mm)	Screw Tightening Torque (N·m)	Mass (g)
CPLS (Stainless Steel)	8	0.2	78000	3.1×10^{-8}	50	0.10	2	±0.2	0.3	3
	12	0.3	52000	2.1×10^{-7}	64			±0.3	0.5	9.3
	16	0.5	39000	8.4×10^{-7}	85				0.7	21
	20	1	31000	2.4×10^{-6}	250					38
	25	2	25000	6.8×10^{-6}	330	0.15		±0.4	1.7	71
	32	3.5	19000	2.6×10^{-5}	850			±0.5		160
	40	8	15000	8.7×10^{-5}	1000	0.20			4	350

Part Number Type	D	Allowable Torque (N·m)	Max. Rotational Speed (r/min)	Moment of Inertia (kg·m²)	Static Torsional Spring Constant (N·m/rad)	Angular Misalignment (°)	Allowable Axial Misalignment (mm)	Screw Tightening Torque (N·m)	Mass (g)
CPSS (Stainless Steel)	8	0.2	78000	2.4×10^{-8}	49	1	±0.1	0.3	2.7
	12	0.3	52000	1.8×10^{-7}	140			0.5	7.8
	16	0.5	39000	7.2×10^{-7}	240			0.7	18
	20	1	31000	2.0×10^{-6}	330				32
	25	2	25000	6.1×10^{-6}	720		±0.2	1.7	63
	32	3.5	19000	2.1×10^{-5}	1300				130

CPSS does not allow eccentricity.

Ordering Example: Part Number - Shaft Bore Dia. d1 - Shaft Bore Dia. d2
CPL16 - 5 - 6

To change the shaft bore diameter, see P.1071.

Slit Couplings

Set Screw, Long

Points of comparison between similar products | Max. Rotational Speed: 10,000rpm

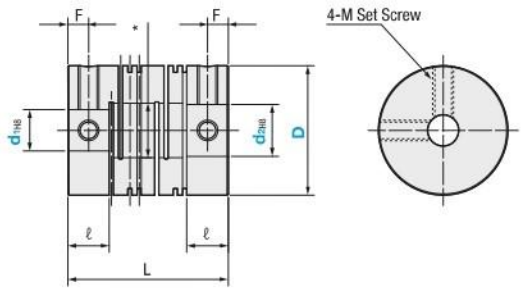
Similar products page P.1073

Features: Product quality and performance same as of the conventional products but at lower price. Replaceable from CPL.

RoHS

GSASL

*d1, d2 Identical Diameter = d1+0.5
d1, d2 Different Diameters = Large Shaft Diameter



The lateral, angular, and axial misalignment values shown are for each occurring individually. When multiple misalignments are occurring simultaneously, the allowable maximum value of each will be reduced to 1/2.
Tolerances for d1 and d2 are values before slit machining.

TYPE	M Material	S Surface Treatment	A Accessory
GSASL	Aluminum Alloy	Clear Anodize	Set Screw

Part Number Type	D	d1	d2	L	ℓ	Set Screw M (Coarse)	Set Screw Tightening Torque (N·m)	F	Unit Price
Set Screw GSASL	16	4	4 5	23	6.4	M3	0.7	3	
		5	5 8						
	20	5	6 8	26	7				
		6	6 8 10						
	25	6	10	31	8	M4	1.7	4	
		6.35	8 10						
		8	8 10 12						
	32	10	10 11 12 14	41	11			6	
		12	12 14						

Characteristic Values

Part Number Type	D	Allowable Torque (N·m)	Max. Rotational Speed (rpm)	Moment of Inertia (kg·m²)	Static Torsional Spring Constant (N·m/rad)	Lateral Misalignment (mm)	Allowable Angular Misalignment (°)	Allowable Axial Misalignment (mm)	Mass (g)
GSASL	16	0.5	10000	6.5×10^{-7}	44	0.1	2	±0.4	9
	20	1		1.5×10^{-6}	110				16
	25	2		4.2×10^{-6}	215	0.15		±0.5	27
	32	4		1.6×10^{-5}	420				64

Static torsional spring constant, inertia moment, and mass values are for cases of maximum shaft diameter.
For the selection criteria and alignment procedures, see P.1061, 1062.

Ordering Example: Part Number - Shaft Bore Dia. d1 - Shaft Bore Dia. d2
GSASL20 - 6 - 8

7. Bearing Fits

7.1 Fitting

For rolling bearings, inner and outer rings are fixed on the shaft or in the housing so that relative movement does not occur between fitting surfaces during operation or under load. This relative movement between the fitting surfaces of the bearing and the shaft or housing can occur in a radial direction, an axial direction, or in the direction of rotation. Types of fitting include tight, transition and loose fitting, which may be selected depending on whether or not there is interference.

The most effective way to fix the fitting surfaces between a bearing's raceway and shaft or housing is to apply a "**tight fit**." The advantage of this tight fit for thin walled bearings is that it provides uniform load support over the entire ring circumference without any loss of load carrying capacity. However, with a tight fit, ease of installation and disassembly is lost; and when using a non-separable bearing as the floating-side bearing, axial displacement is not possible. For this reason, a tight fit cannot be recommended in all cases.

7.2 The necessity of a proper fit

In some cases, improper fit may lead to damage and shorten bearing life, therefore it is necessary to make a careful investigation in selecting a proper fit. Some of the bearing failure caused by improper fit are listed below.

- Raceway cracking, early flaking and displacement of raceway
- Raceway and shaft or housing abrasion caused by creeping and fretting corrosion
- Seizing caused by negative internal clearances
- Increased noise and deteriorated rotational accuracy due to raceway groove deformation

Please refer to insert pages A-96 ~ A-99 for information concerning diagnosis of these conditions.

7.3 Fit selection

Selection of a proper fit is dependent upon thorough analysis of bearing operating conditions, including consideration of:

- Shaft and housing material, wall thickness, finished surface accuracy, etc.
- Machinery operating conditions (nature and magnitude of load, rotational speed, temperature, etc.)

7.3.1 "Tight fit" or "Loose fit"

(1) For raceways under rotating loads, a tight fit is necessary. (Refer to **Table 7.1**) "Raceways under rotating loads" refers to raceways receiving loads rotating relative to their radial direction. For raceways under static loads, on the other hand, a loose fit is sufficient.
(Example) Rotating inner ring load = the direction of the radial load on the inner ring is rotating relatively

(2) For non-separable bearings, such as deep groove ball bearings, it is generally recommended that either the inner ring or outer ring be given a loose fit.

Table 7.1 Radial load and bearing fit

Illustration	Bearing rotation	Ring load	Fit
Static load	Inner ring: Rotating / Outer ring: Stationary	Rotating inner ring load	Inner ring : Tight fit
Unbalanced load	Inner ring: Stationary / Outer ring: Rotating	Static outer ring load	Outer ring : Loose fit
Static load	Inner ring: Stationary / Outer ring: Rotating	Static inner ring load	Inner ring : Loose fit
Unbalanced load	Inner ring: Rotating / Outer ring: Stationary	Rotating outer ring load	Outer ring : Tight fit

7.3.2 Recommended Fits

Bearing fit is governed by the selection tolerances for bearing shaft diameters and housing bore diameters.

Widely used fits for 0 Class tolerance bearings and various shaft and housing bore diameter tolerances are shown in **Fig. 7.1**.

Generally-used, standard fits for most types of bearings and operating conditions are shown in **Tables 7.2 - 7.7**.

Table 7.2: Fits for radial bearings
Table 7.3: Fits for thrust bearings
Table 7.4: Fits for electric motor bearings
Table 7.6: Fits for inch series tapered roller bearings (ANSI Class 4)
Table 7.7: Fits for inch series tapered roller bearings (ANSI Class 3 and 0)
Table 7.5. shows fits and their numerical values.

For special fits or applications, please consult **NTN** Engineering.

7.3.3 Interference minimum and maximum values

The following points should be considered when it is necessary to calculate the interference for an application:

- In calculating the minimum required amount of interference keep in mind that:
  1) interference is reduced by radial loads
  2) interference is reduced by differences between bearing temperature and ambient temperature
  3) interference is reduced by variation of fitting surfaces
- The upper limit value should not exceed 1/1000 of the shaft diameter.

Required interference calculations are shown below.

(1) Radial loads and required interference

Interference of the inner ring and shaft decreases when a radial load is applied to the bearing. The interference required to secure effective interference is expressed by formulae (7.1) and (7.2).

$$F_r \leqq 0.3\,C_{or}$$
$$\Delta_{dF} = 0.08\,(d \cdot F_r / B)^{1/2} \quad \left.\begin{matrix} \\ \end{matrix}\right\} \text{N} \quad \left.\begin{matrix} \\ \{\text{kgf}\} \end{matrix}\right\} \cdots\cdots\cdots(7.1)$$
$$= 0.25\,(d \cdot F_r / B)^{1/2}$$
$$F_r > 0.3\,C_{or}$$
$$\Delta_{dF} = 0.02\,(F_r / B) \quad \left.\begin{matrix} \\ \end{matrix}\right\} \text{N} \quad \left.\begin{matrix} \\ \{\text{kgf}\} \end{matrix}\right\} \cdots\cdots\cdots(7.2)$$
$$= 0.2\,(F_r / B)$$

Where,

Δ_{dF}: Required effective interference according to radial load μm
d : Bearing bore diameter mm
B : Inner ring width mm
F_r : Radial load N {kgf}
C_{or} : Basic static load rating N {kgf}

(2) Temperature difference and required interference

Interference between inner rings and steel shafts is reduced as a result of temperature increases (difference between bearing temperature and ambient temperature, ΔT) caused by bearing rotation. Calculation of the minimum required amount of interference in such cases is shown in formula (7.3).

$$\Delta_{dT} = 0.0015 \cdot d \cdot \Delta T \cdots\cdots\cdots(7.3)$$

Δ_{dT} : Required effective interference for temperature difference μm
ΔT : Difference between bearing temperature and ambient temperature °C
d : Bearing bore diameter mm

(3) Fitting surface variation and required interference

Interference decreases because the fitting surface is smoothened by fitting (surface roughness is reduced). The amount the interference decreases depends on roughness of the fitting surface. It is generally necessary to anticipate the following decrease in interference.

For ground shafts: 1.0～2.5 μm
For lathed shafts: 5.0～7.0 μm

(4) Maximum interference

When bearing rings are installed with an interference fit, tension or compression stress may occur along their raceways. If interference is too great, this may cause damage to the rings and reduce bearing life. You should try to obtain the previously described upper limit.

7.3.4 Other details

(1) Tight interference fits are recommended for,
- Operating conditions with large vibration or shock loads
- Applications using hollow shafts or housings with thin walls
- Applications using housings made of light alloys or plastic

(2) Small interference fits are preferable for,
- Applications requiring high running accuracy
- Applications using small sized bearings or thin walled bearings

(3) Consideration must also be given to the fact that fit selection will effect internal bearing clearance selection. (refer to page insert A-58)

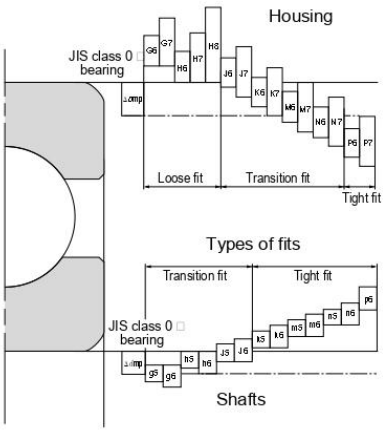


Fig 7.1 State of fitting

MICROCHIP

PIC18F2331/2431/4331/4431

28/40/44-Pin Enhanced Flash Microcontrollers with nanoWatt Technology, High Performance PWM and A/D

14-bit Power Control PWM Module:

- Up to 4 channels with complementary outputs
- Edge- or center-aligned operation
- Flexible dead-band generator
- Hardware fault protection inputs
- Simultaneous update of duty cycle and period:
  - Flexible special event trigger output

Motion Feedback Module:

- Three independent input capture channels:
  - Flexible operating modes for period and pulse width measurement
  - Special Hall Sensor interface module
  - Special event trigger output to other modules
- Quadrature Encoder Interface:
  - 2 phase inputs and one index input from encoder
  - High and low position tracking with direction status and change of direction interrupt
  - Velocity measurement

High-Speed, 200 Ksps 10-bit A/D Converter:

- Up to 9 channels
- Simultaneous two-channel sampling
- Sequential sampling: 1, 2 or 4 selected channels
- Auto-conversion capability
- 4-word FIFO with selectable interrupt frequency
- Selectable external conversion triggers
- Programmable acquisition time

Flexible Oscillator Structure:

- Four crystal modes up to 40 MHz
- Two external clock modes up to 40 MHz
- Internal oscillator block:
  - 8 user selectable frequencies: 31 kHz to 8 MHz
  - OSCTUNE can compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown of device if clock fails

Power-Managed Modes:

- Run CPU on, peripherals on
- Idle CPU off, peripherals on
- Sleep CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep current down to 0.1 μA typical
- Timer1 oscillator, 1.8 μA typical, 32 kHz, 2V
- Watchdog Timer (WDT), 2.1 μA typical
- Two-Speed oscillator start-up

Peripheral Highlights:

- High current sink/source 25 mA/25 mA
- Three external interrupts
- Two Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10 bits
- Enhanced USART module:
  - Supports RS-485, RS-232 and LIN 1.2
  - Auto-Wake-up on Start bit
  - Auto-Baud detect
- RS-232 operation using internal oscillator block (no external crystal required)

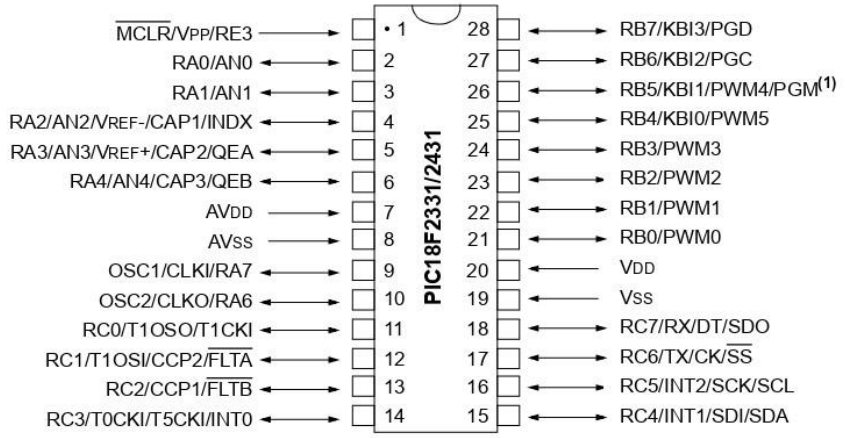
Special Microcontroller Features:

- 100,000 erase/write cycle enhanced Flash program memory typical
- 1,000,000 erase/write cycle data EEPROM memory typical
- Flash/data EEPROM retention: 100 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 X 8 Single-cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Single-supply In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
  - Drives PWM outputs safely when debugging

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP	SSP		EUSART	Quadrature Encoder	14-bit PWM (ch)	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Slave I²C™				
PIC18F2331	8192	4096	768	256	24	5	2	Y	Y	Y	Y	6	1/3
PIC18F2431	16384	8192	768	256	24	5	2	Y	Y	Y	Y	6	1/3
PIC18F4331	8192	4096	768	256	36	9	2	Y	Y	Y	Y	8	1/3
PIC18F4431	16384	8192	768	256	36	9	2	Y	Y	Y	Y	8	1/3

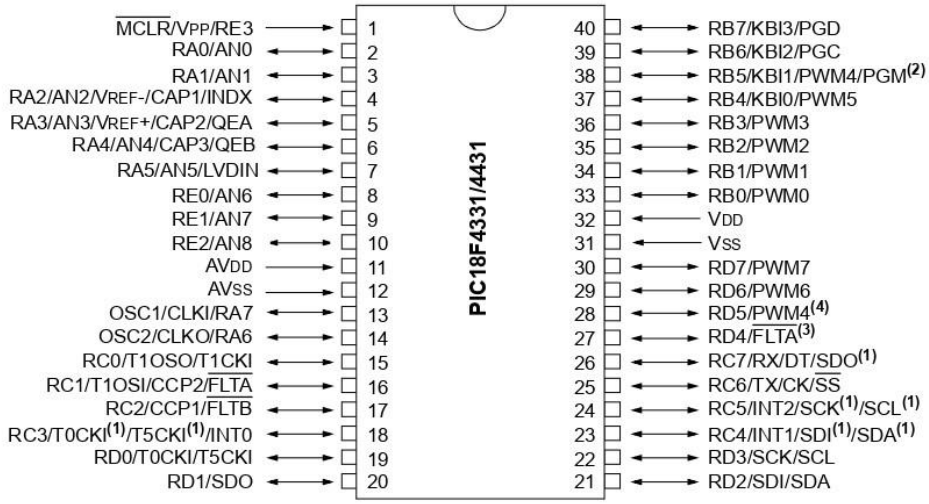
Pin Diagrams

28-Pin SDIP, SOIC



Note 1: Low-voltage programming must be enabled.

40-Pin PDIP



Note 1: RC3 is the alternate pin for T0CKI/T5CKI; RC4 is the alternate pin for SDI/SDA; RC5 is the alternate pin for SCK/SCL.

2: Low-voltage programming must be enabled.

3: RD4 is the alternate pin for FLTA.

4: RD5 is the alternate pin for PWM4.

Αρχείο serial_connection.py

```
 #!/usr/bin/python

import rospy
import serial
import time
from std_msgs.msg import String
from decimal import Decimal
import numpy as np
from scipy.integrate import odeint, solve_ivp
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import os


#camera = [0, 0, 0]
camera = ""
#error = [0, 0, 0]
#sending_data = [0, 0, 0]
#print(sending_data)


added_mass_x = 0.03 # kg
added_mass_y = 0.04
mb = 0.3 # kg
m1 = mb-added_mass_x
m2 = mb-added_mass_y
l1 = 0.07 # m
l2 = 0.05 # m
J = 0.00050797 # kgm^2
Sa = 0.0110 # m^2
Cd = 2.44
Cl = 3.41
Kd = 0.000655 # kgm^2
r = 1000 # kg/m^3

c1 = 0.5*r*Sa*Cd
c2 = 0.5*r*Sa*Cl
c3 = 0.5*mb*(l1**2)
c4 = Kd/J
c5 = (1/(2*J))*(l1**2)*mb*l2
c6 = (1/(3*J))*(l1**3)*mb

theta_0 = 10*(np.pi/180) # rad
theta_A = 20*(np.pi/180) # rad
f = 2 # Hz

#####################################################
t = np.linspace(0,100,8000) # s
#####################################################

xdot_list = []
ydot_list = []
xdotdot_list = []
ydotdot_list = []

def txt_to_arrays():
    # Save data from txt files into numpy arrays
```

```
    # An den thelw na xrisimopoiisw absolute paths gia na anoiksw ta txt
files
    # ta anoigw opws pio katw (na mpei kai gia ta 4)
    # alla sto command line prepei na treksw ton komvo etsi: rosrun
usb_rs232 serial_connection.py _arg_name:=$(rospack find
usb_rs232)/scripts
    #base_dir = rospy.get_param('~arg_name')
    #with open(os.path.join(base_dir, 'xdot.txt'), 'r') as filehandle:

    with
open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/xdot.txt'),
'r') as filehandle:      # path gia arxeia txt mesa sto package
    #with
open(os.path.expanduser('~/Documents/Spyros/Diplwmatiki/ROS/Serial
connection/xdot.txt'), 'r') as filehandle: # path gia arxeia txt sto pc
        filecontents = filehandle.readlines()

        for line in filecontents:
            current_place = line[:-1]
            xdot_list.append(current_place)

    global xdot
    xdot = np.array(xdot_list, dtype=np.float32) #to dtype den kserw an
ine swsto

    with
open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/ydot.txt'),
'r') as filehandle:
    #with
open(os.path.expanduser('~/Documents/Spyros/Diplwmatiki/ROS/Serial
connection/ydot.txt'), 'r') as filehandle:
        filecontents = filehandle.readlines()

        for line in filecontents:
            current_place = line[:-1]
            ydot_list.append(current_place)

    global ydot
    ydot = np.array(ydot_list, dtype=np.float32) #to dtype den kserw an
ine swsto

    with
open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/xdotdot.txt'),
'r') as filehandle:
    #with
open(os.path.expanduser('~/Documents/Spyros/Diplwmatiki/ROS/Serial
connection/xdotdot.txt'), 'r') as filehandle:
        filecontents = filehandle.readlines()

        for line in filecontents:
            current_place = line[:-1]
            xdotdot_list.append(current_place)

    global xdotdot
    xdotdot = np.array(xdotdot_list, dtype=np.float32) #to dtype den
kserw an ine swsto

    with
open(os.path.expanduser('~/catkin_ws/src/usb_rs232/scripts/ydotdot.txt'),
'r') as filehandle:
```

```
    #with
open(os.path.expanduser('~/Documents/Spyros/Diplwmatiki/ROS/Serial
connection/ydotdot.txt'), 'r') as filehandle:
        filecontents = filehandle.readlines()

        for line in filecontents:
            current_place = line[:-1]
            ydotdot_list.append(current_place)

    global ydotdot
    ydotdot = np.array(ydotdot_list, dtype=np.float32) #to dtype den
kserw an ine swsto

    print("kwlos")

    # Paremvoli simeiwn apo numpy arrays
    global interp
    interp = interp1d(t,(xdot,ydot,xdotdot,ydotdot))



def callback(data):  # Gia na mporei na kanei subscribe sto topic gia na
penrei to buffer apo to RPi
    print("callback")
    global camera
    camera = data.data
    print camera
    str_x_c, str_y_c, str_z_c, str_theta_c, str_ux_c, str_uy_c =
camera.strip().rstrip("/").split(",")[:6]
    x_c = float(str_x_c)
    y_c = float(str_y_c)
    z_c = float(str_z_c)                        # apothikeysi twn 6
numerwn se metavlites
    theta_c = float(str_theta_c)                # mporun na xrisimopoithun
allu aen ginun global
    ux_c = float(str_ux_c)
    uy_c = float(str_uy_c)
    print x_c, y_c, z_c, theta_c, ux_c, uy_c



def inverse(t,k):
    print("inverse")
    # epilysi Inverse Dynamic Problem
    # Dinei tin epithymiti thesi tis gwnias theta_i (gwnia uras) se 3
numera
    vcx_i = k[0]
    vcy_i = k[1]
    psi_i = k[2]
    wz_i = k[3]
    theta_i = k[4]
    theta_deg_i = k[5]

   # Apothikeusi eksiswsewn apo paramvoli
    xdot, ydot, xdotdot, ydotdot = interp(t)

    # Subsystem 4
    vcx_i = xdot*np.cos(psi_i)-ydot*np.sin(psi_i)
    vcy_i = ydot*np.cos(psi_i)+xdot*np.sin(psi_i)
    psidot_i = wz_i
```

```
    vcxdot_i = xdotdot*np.cos(psi_i)-xdot*np.sin(psi_i)*psidot_i-
ydotdot*np.sin(psi_i)-ydot*np.cos(psi_i)*psidot_i
    vcydot_i = ydotdot*np.cos(psi_i)-
ydot*np.sin(psi_i)*psidot_i+xdotdot*np.sin(psi_i)+xdot*np.cos(psi_i)*psid
ot_i

    g1 = -(m1/c3)*vcxdot_i+(m2/c3)*vcy_i*wz_i-
(c1/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))+(c2/c3)*vcy_i*np.sqrt((vcx_i
**2)+(vcy_i**2))*np.arctan2(vcy_i,vcx_i)
    g2 =
(m2/c3)*vcydot_i+(m1/c3)*vcx_i*wz_i+(c1/c3)*vcy_i*np.sqrt((vcx_i**2)+(vcy
_i**2))+(c2/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))*np.arctan2(vcy_i,vcx
_i)


    A = 12*np.sin(2*np.pi*f*t+np.pi) # eksiswsi tail_frequency apo
simulink
    if A>=0.1:
        wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-c5*g2-
c6*np.sqrt((g1**2)+(g2**2))
    elif A<-0.1:
        wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-
c5*g2+c6*np.sqrt((g1**2)+(g2**2))
    else:
        wzdot_i = ((m1-m2)/J)*vcx_i*vcy_i-c4*wz_i**2*np.sign(wz_i)-c5*g2

    # Subsystem 5
    if g2>0:
        theta_i = np.arctan2(g1,g2)
    elif g2<0 and g1>=0:
        theta_i = np.arctan2(g1,g2)-np.pi
    elif g2<0 and g1<0:
        theta_i = np.arctan2(g1,g2)+np.pi
    elif g2==0 and g1>0:
        theta_i = -np.pi/2
    elif g2==0 and g1<0:
        theta_i = np.pi/2
    elif g1==0 and g2==0:
        theta_i = 0

    theta_deg_i = (theta_i*180)/np.pi


    return [vcxdot_i, vcydot_i, psidot_i, wzdot_i, theta_i, theta_deg_i]

def solution():
    # arxikes synthikes
    vcx_i_0 = 0.1257
    vcy_i_0 = 0
    psi_i_0 = 0
    wz_i_0 = 0
    theta_i_0 = 0
    theta_deg_i_0 = 0
    #theta_i_0 = 0.1745
    #theta_deg_i_0 = 9.866
    k0 = [vcx_i_0, vcy_i_0, psi_i_0, wz_i_0, theta_i_0, theta_deg_i_0]

    # Epilysi systimatos diaforikwn
    steps = 1
    method = 'RK23'
    atol = 1e-12 # (default = 1e-6)
    k = solve_ivp(inverse, (0, 100), k0, method=method, t_eval=t,
atol=atol, vectorized=True)
```

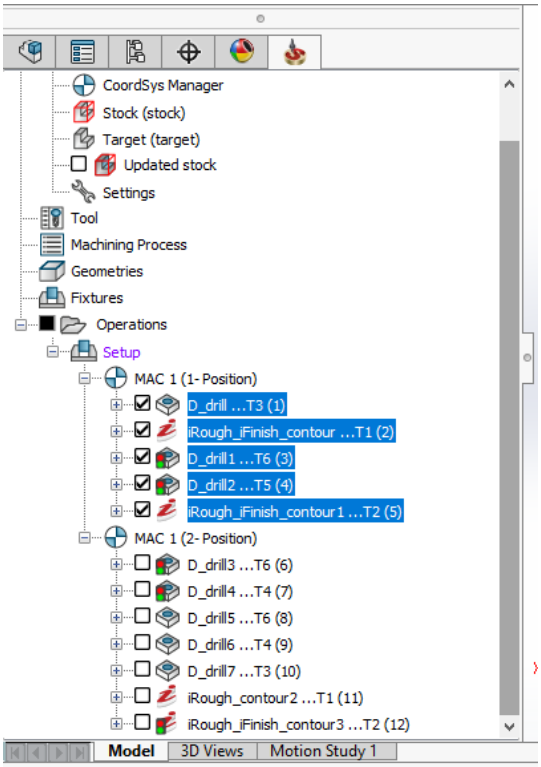
```
    # apothikeysi apotelesmatwn
    vcx_i = k.y[0,:]
    vcy_i = k.y[1,:]
    psi_i = k.y[2,:]
    wz_i = k.y[3,:]
    theta_i = k.y[4,:]
    theta_deg_i = k.y[5,:]

    # Epanalipsi tu Subsystem 5 gia na mporun na plotaristun ta theta_i,
theta_deg_i
    k_new = np.transpose(k.y) # antistrofi diastasewn k.y apo (6,8000) se
(8000,6)
    theta_i = [inverse(t_i, k_i)[4] for t_i, k_i in zip(t, k_new)]
    theta_deg_i = [inverse(t_i, k_i)[5] for k_i, t_i in zip(k_new, t)]

    # Ypologismos mesis gwnias theta kai platus talantwsis
    global mesi_gwnia, platos
    mesi_gwnia = sum(theta_i)/len(theta_i) # rad
    theta_i_trim = theta_i[1000:7000]      # kovw kapoia anepithymita
peaks se arxi kai telos
    platos = (max(theta_i_trim)-min(theta_i_trim))/2
    #platos = (max(theta_i)-min(theta_i))/2

    # Eksiswseis pu den mporun na plotaristun an den ine ektos tu
function
    psidot_i = wz_i
    vcxdot_i = xdotdot*np.cos(psi_i)-xdot*np.sin(psi_i)*psidot_i-
ydotdot*np.sin(psi_i)-ydot*np.cos(psi_i)*psidot_i
    vcydot_i = ydotdot*np.cos(psi_i)-
ydot*np.sin(psi_i)*psidot_i+xdotdot*np.sin(psi_i)+xdot*np.cos(psi_i)*psid
ot_i

    g1 = -(m1/c3)*vcxdot_i+(m2/c3)*vcy_i*wz_i-
(c1/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))+(c2/c3)*vcy_i*np.sqrt((vcx_i
**2)+(vcy_i**2))*np.arctan2(vcy_i,vcx_i)
    g2 =
(m2/c3)*vcydot_i+(m1/c3)*vcx_i*wz_i+(c1/c3)*vcy_i*np.sqrt((vcx_i**2)+(vcy
_i**2))+(c2/c3)*vcx_i*np.sqrt((vcx_i**2)+(vcy_i**2))*np.arctan2(vcy_i,vcx
_i)

def serial_connection():
    print("serial connection")

    if not rospy.is_shutdown():

        if ser.isOpen():
            print ("Port Open")
            for i in range(1):
                #ser.write(b'5')
                #ser.write(chr(5)) auto stelnei kwdikopoiimenus
xaraktires
                #ser.write(5)
                #ser.write(bytes(sending_data) + '\n')
                ser.write('m'+bytes(mesi_gwnia))
                time.sleep(0.001)
                ser.write('w'+bytes(platos))
                time.sleep(0.001)
                ser.write('f'+bytes(f))
                time.sleep(0.001)
            #ser.flush()
```

```
if __name__ == '__main__':
    try:
        rospy.init_node('serial_connection', anonymous = True)
        ser = serial.Serial('/dev/ttyUSB0', 115200)
        print (ser.name)
        time.sleep(2)
        rospy.Subscriber('real_position', String, callback)
        rate = rospy.Rate(1) #isws na min xreiazetai
        txt_to_arrays()
        solution()
        while not rospy.is_shutdown():
            print("in the while loop")
            serial_connection()
            rate.sleep()
        ser.close()
    except rospy.ROSInterruptException:
        pass
```

Διαδικασία εξαγωγής G-Code

Μετά την ολοκλήρωση της προσομοίωσης των κατεργασιών, εξάγεται ο κώδικας στην γλώσσα προγραμματισμού αριθμητικού ελέγχου (G-code) από το λογισμικό SolidCAM. Εφόσον η διαδικασία διακόπτεται ώστε να αντιστραφεί το τεμάχιο στην μέγγενη της εργαλειομηχανής, ο κώδικας θα εξαχθεί σε δύο μέρη (ένα για κάθε σύστημα συντεταγμένων).

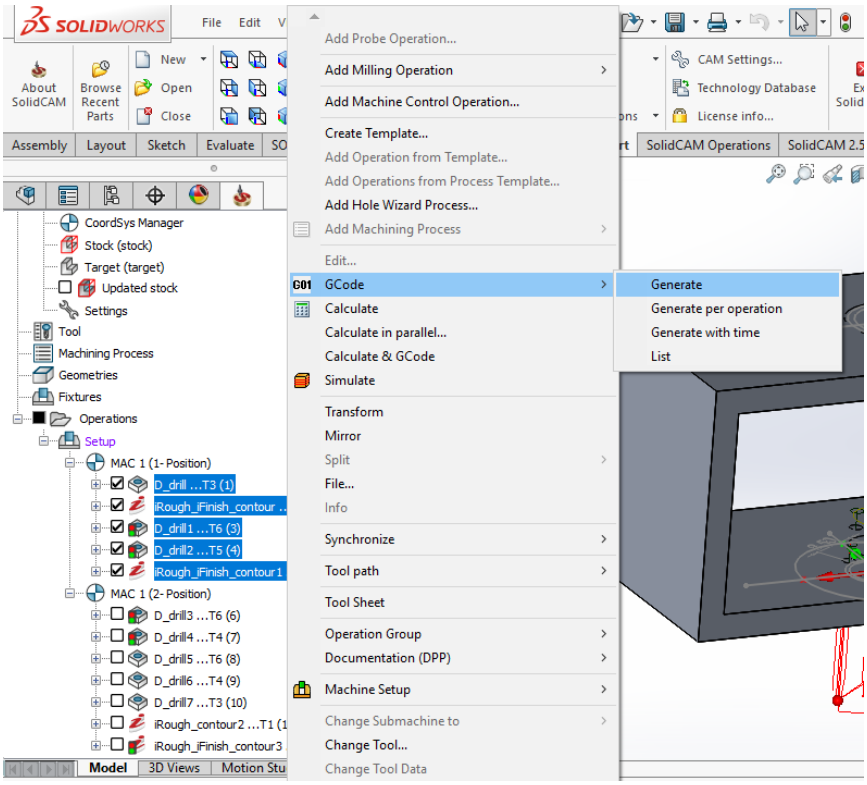
Σε πρώτο στάδιο, από το μενού κατεργασιών στο αριστερό μέρος της επιφάνειας εργασίας, επιλέγονται όλες οι κατεργασίες που υλοποιούνται με βάση το πρώτο σύστημα συντεταγμένων, όπως παρουσιάζεται στο Σχήμα Π-1.



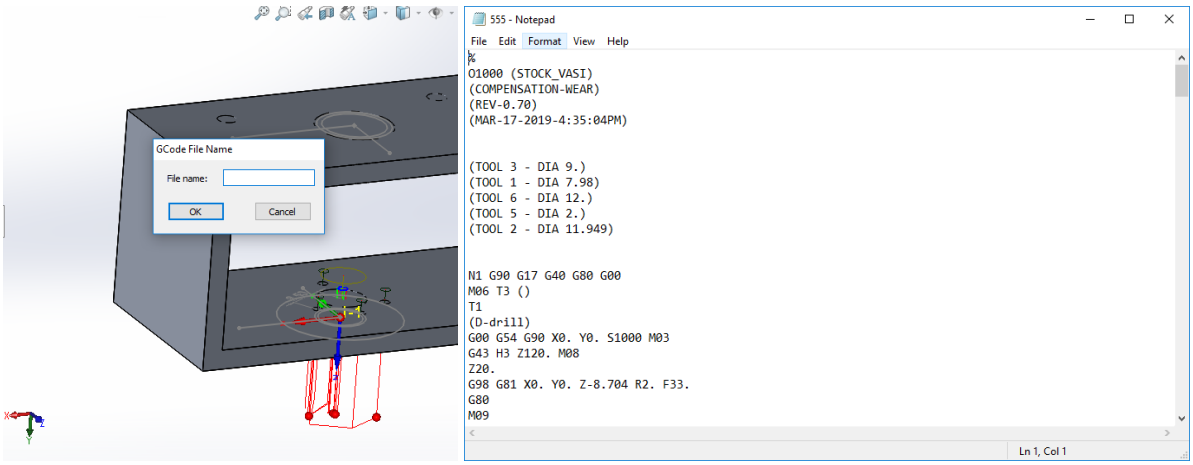
Σχήμα Π-1. Επιλογή κατεργασιών πρώτου συστήματος συντεταγμένων.

Στη συνέχεια, με δεξί κλικ στις επιλεγμένες κατεργασίες, εμφανίζεται ένα μενού επιλογών, από το οποίο επιλέγεται η καρτέλα GCode και από το νέο μενού επιλέγεται η καρτέλα Generate (βλ. Σχήμα Π-2). Αφού δοθεί κατάλληλο όνομα στο αποθηκευόμενο αρχείο (βλ. Σχήμα Π-3-i), ένα νέο παράθυρο εμφανίζεται με το αρχείο του κώδικα, στην μορφή αρχείου τύπου txt (βλ. Σχήμα Π-3-ii). Πλέον, ο κώδικας είναι διαθέσιμος για περαιτέρω τροποποίηση και εισαγωγή του στο σύστημα της εργαλειομηχανής, για έναρξη των κατεργασιών στο τεμάχιο.

Η διαδικασία επαναλαμβάνεται για τις κατεργασίες που έχουν ως βάση το δεύτερο σύστημα συντεταγμένων. Ο συνολικός κώδικας αποτελείται από τα δύο αρχεία τύπου txt (ένα για κάθε σύστημα συντεταγμένων).



Σχήμα Π-2. Εξαγωγή G-Code.



(i) **(ii)**

Σχήμα Π-3. Αποθήκευση αρχείου: (i) Ονομασία αρχείου, (ii) Αρχείο G-Code.