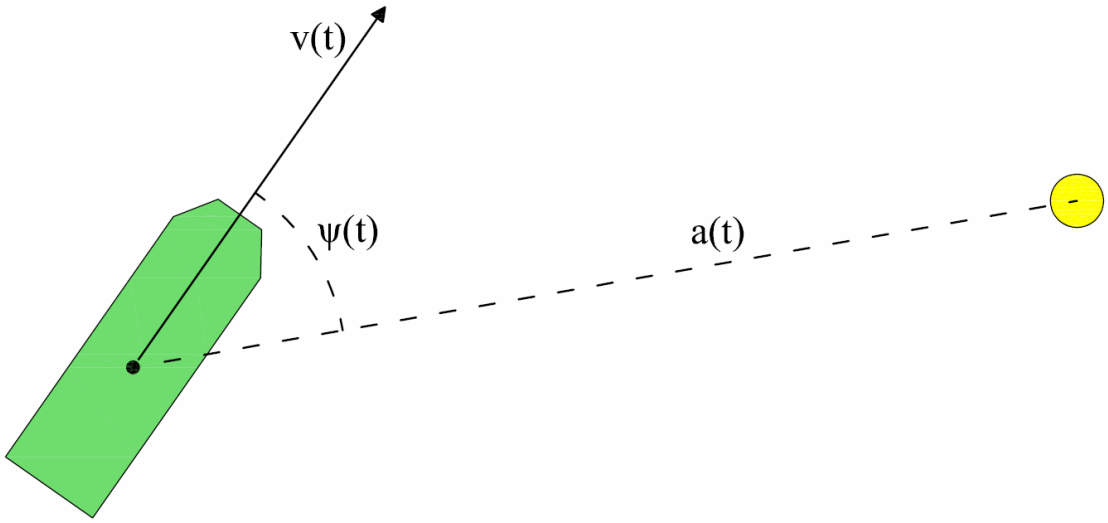
Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ναυπηγών Μηχανολόγων Μηχανικών

Τομέας Ναυτικής Μηχανολογίας

Διπλωματική Εργασία

Ανάπτυξη αυτόνομου υπό κλίμακα προτύπου SES με νόμο ελέγχου βασισμένο σε ανατροφοδότηση σήματος εικόνας



Νικόλαος Κουβαράς

Επιβλέπων: Ι. Γεωργίου, Αναπληρωτής Καθηγητής

Αθήνα, Δεκέμβρης 2008

Η διπλωματική αυτή εργασία ξεκίνησε το Μάη του 2005 και ολοκληρώθηκε το Δεκέμβρη του 2008.

Με την εργασία αυτή μου δόθηκε η δυνατότητα να γνωρίσω το ιδιαίτερα ενδιαφέρον αντικείμενο της ρομποτικής. Έτσι, ασχολήθηκα με προγραμματισμό τόσο υπολογιστών όσο και μικροεπεξεργαστών αλλά και με το σχεδιασμό και κατασκευή ηλεκτρονικών κυκλωμάτων. Για το λόγο αυτό θα ήθελα να ευχαριστήσω τον κ. Νικόλαο Ξηρό που μου έδωσε την ευκαιρία να εκπονήσω αυτή την εργασία αλλά και για την αμέριστη βοήθειά του σε κάθε στάδιο της.

Ευχαριστήσω θερμά τον κ. Ι. Γεωργίου ο οποίος δέχθηκε να αναλάβει και να ολοκληρώσει την εργασία αυτή. Η βοήθεια του είναι πολύτιμη γιατί ανέλαβε μια διπλωματική που ίσως ήταν εκτός των πεδίων ενδιαφέροντός του αλλά δεν αρνήθηκε οποιαδήποτε βοήθεια. Επίσης ευχαριστώ τον υποψήφιο διδάκτορα κ. Ε. Λόγη, τους φοιτητές Ναυπηγούς Μηχ. Μηχ. Κ. Καρδάση και Γ. Σπανό καθώς και τον φοιτητή Πληροφορικής Ν. Φερτάκη για όλη την πολύτιμη βοήθεια που μου προσέφεραν και η οποία ήταν καθοριστική. Προκειμένου να γίνουν κάποια εξειδικευμένα πειράματα χρειάστηκε ο προσωπικός χώρος του Δ. Παξινού τον οποίο μου παραχώρησε και τον ευχαριστώ θερμά για αυτό.

Ευχαριστώ εκ προοιμίου, τον Επίκουρο καθηγητή της σχολής Ναυπηγών Μηχ. Μηχ. κ. Ι. Προυσαλίδη και τον Λέκτορα κ. Χ. Παπαδόπουλο, για την παρουσία τους στην εξέταση και παρουσίαση της εργασίας αυτής.

Τέλος, ευχαριστώ την οικογένεια μου και τους δικούς μου ανθρώπους οι οποίοι με στήριξαν και με ανέχθηκαν καθ' όλη τη διάρκεια των σπουδών μου.

Αφιερώνω την εργασία αυτή στη μνήμη του Αλέξη Γρηγορόπουλου, τον οποίο δολοφόνησε το φασιστικό κράτος που ζούμε...

ΣΥΝΟΨΗ

Στην παρούσα εργασία αναπτύσσεται ένα ολοκληρωμένο σύστημα δυναμικού εντοπισμού και ελέγχου της πορείας και θέσης σκάφους SES (*Surface Effect Ship*) μέσω της ανάπτυξης ενός συστήματος παρακολούθησης φωτεινού στόχου που βρίσκεται στην ακτή ή σε άλλο προπορευόμενο σκάφος.

Σκοπός είναι να δημιουργηθεί ένα σύστημα ναυσιπλοΐας για υπό κλίμακα μοντέλα πλοίων. Κατ' επέκταση, το σύστημα έχει τη δυνατότητα εφαρμογής και σε πραγματικά πλοία. Αυτό που επιζητούμε είναι αρχικά να δημιουργηθεί μια βάση hardware και software με δυνατότητα επέκτασης. Έτσι, αναπτύσσουμε ένα πακέτο το οποίο περιέχει προγραμματιζόμενο μικροελεγκτή, αισθητήρες, ενεργοποιητές καθώς και πομποδέκτες για τηλεχειρισμό και τηλεμετρία.

Το σκάφος κατασκευάζεται και εξοπλίζεται με τον απαραίτητο ηλεκτρομηχανολογικό αλλά και ηλεκτρονικό εξοπλισμό. Στη συνέχεια κάνοντας κάποιες παραδοχές, μοντελοποιούμε μαθηματικά την κίνησή του εξάγοντας μια γενική μορφή διαφορικών εξισώσεων. Προκειμένου να κάνουμε εικονικά πειράματα σε προσομοιωτή, ταυτοποιούμε το σύστημα.

Στη συνέχεια σχεδιάζεται ένας ελεγκτής PI προκειμένου να πραγματοποιηθεί εικονικά το πείραμα ***παρακολούθησης στόχου*** σε Η/Υ. Μέσω εικονικών πειραμάτων προσδιορίζονται οι σταθερές του PI. Σύμφωνα με αυτές τις σταθερές εισάγουμε σε ψηφιακή μορφή τον ελεγκτή σε microcontroller.

Το πείραμα παρακολούθησης στόχου στο φυσικό μοντέλο πραγματοποιείται σε πισίνα δίνοντας εντολή ασύρματα στο σκάφος να παρακολουθήσει το φως. Το πλοίο στέλνει ασύρματα δεδομένα σε σχέση με το οπτικό πεδίο του αισθητήρα εικόνας τα οποία λαμβάνονται από κονσόλα ελέγχου σε Η/Υ.

Γίνεται σύγκριση των αποτελεσμάτων των πειραμάτων παρακολούθησης στόχου μεταξύ εικονικού και πραγματικού πειράματος. Έτσι, εξάγονται συμπεράσματα σχετικά με τη μαθηματική μοντελοποίηση αλλά και με τεχνικά θέματα τα οποία αφορούν τις συνθήκες διεξαγωγής πειραμάτων. Τέλος, προτείνονται τρόποι βελτίωσης της μεθοδολογίας μοντελοποίησης και λειτουργίας του συστήματος αλλά και κάποιες προσθήκες οι οποίες θα φέρουν καλύτερα αποτελέσματα.

ΠΕΡΙΕΧΌΜΕΝΑ

Σελίδα

1. ΕΙΣΑΓΩΓΗ

1.1 Αναφορά στα συστήματα ναυσιπλοΐας

Ναυσιπλοΐα (*navigation*) είναι η διαδικασία σχεδιασμού, αναγνώρισης και ελέγχου της κίνησης ενός οχήματος από ένα μέρος σε ένα άλλο. Η λέξη navigate προέρχεται από τις Λατινικές ρίζες *navis* που σημαίνει «πλοίο» και *agere* που σημαίνει «κινούμαι» ή «κατευθύνομαι». Όλες οι τεχνικές ναυσιπλοΐας περιλαμβάνουν εντοπισμό της θέσης του οχήματος και σύγκρισή της με δεδομένες τοποθεσίες ή ίχνη.

Τα σημεία του ορίζοντα, ή ακόμη και τα αστέρια, χρησιμοποιούνταν από την αρχαιότητα για τον προσανατολισμό των ανθρώπων. Ένα σταθερό άστρο στον ουρανό, με γνωστή γεωγραφική θέση ως προς το σημείο παρατήρησης, αποτελούσε σημείο αναφοράς και βοηθούσε τους ανθρώπους στο να βρουν τη σωστή πορεία τους. Στον προσανατολισμό συνέβαλαν αργότερα και άλλα μέσα, όπως η πυξίδα και ο εξάντας. Ωστόσο ο εξάντας στην πρώιμη μορφή του είχε τη δυνατότητα να παράσχει πληροφορίες μόνο για το γεωγραφικό πλάτος και όχι για το γεωγραφικό μήκος, γεγονός που αποτελούσε ένα σημαντικό μειονέκτημα, ιδιαίτερα για τους ναυτικούς. Το 1761 ο Άγγλος ωρολογοποιός John Harrison, ύστερα από προσπάθειες δώδεκα ετών, κατασκεύασε ένα όργανο, το οποίο δεν ήταν άλλο από το γνωστό σημερινό χρονόμετρο. Σε συνδυασμό με τον εξάντα, το χρονόμετρο επέτρεπε τον υπολογισμό του στίγματος των πλοίων με εξαιρετική ακρίβεια (για τα δεδομένα της εποχής). Πέρασαν αρκετά χρόνια μέχρι να δημιουργηθούν τα πρώτα συστήματα εντοπισμού θέσης που βασίζονταν σε ηλεκτρονικά μέσα (μέσα 20ού αιώνα). Αυτά τα συστήματα χρησιμοποιήθηκαν ευρύτατα κατά τη διάρκεια του Δευτέρου Παγκοσμίου Πολέμου (και χρησιμοποιούνται ακόμη). Τα συστήματα εντοπισμού θέσης της εποχής αποτελούνταν από ένα δίκτυο σταθμών βάσης και κατάλληλους δέκτες.

Τα πιο σύγχρονα συστήματα ναυσιπλοΐας βασίζονται στην επεξεργασία πληροφοριών που μεταδίδονται στο πλοίο μέσω κυμάτων και χωρίζονται σε δύο κατηγορίες. Στην πρώτη κατηγορία υπάγονται τα υποβρύχια συστήματα ναυσιπλοΐας (*underwater navigational aids*) τα οποία βασίζονται στη μετάδοση ηχητικών κυμάτων στο νερό. Η ανίχνευση υποθαλάσσιων στόχων καθώς και η βυθομέτρηση πραγματοποιούνται με τα ηχοβολιστικά μηχανήματα (*Sound Navigation and Ranging - SONAR*). Επίσης το ηχητικό δρομόμετρο το οποίο είναι βασισμένο στο φαινόμενο Doppler και χρησιμοποιείται στη μέτρηση της ταχύτητας του πλοίου, είναι ένα ακόμα σύστημα ναυσιπλοΐας αυτής της κατηγορίας. Στη δεύτερη κατηγορία υπάγονται τα συστήματα εντοπισμού θέσης τα οποία βασίζονται στη μετάδοση ηλεκτρομαγνητικών (Η/Μ) κυμάτων. Αυτή η κατηγορία αναπτύσσεται αναλυτικότερα παρακάτω γιατί αποτελεί την πλέον διαδεδομένη στα συστήματα ναυσιπλοΐας.

Τα συστήματα εντοπισμού θέσης που λειτουργούν με Η/Μ κύματα χωρίζονται με τη σειρά τους σε δύο υποκατηγορίες. Η πρώτη αφορά συστήματα όπου οι σταθμοί εκπομπής Η/Μ κυμάτων βρίσκονται στη στεριά, άρα η θέση τους είναι δεδομένη και αναλύονται από τρεις κύριες μεθόδους. Η πρώτη μέθοδος στηρίζεται στην εύρεση της διεύθυνσης εκπομπής δύο σταθμών που βρίσκονται σε γνωστές θέσεις ως προς το πλοίο και το στίγμα υπολογίζεται πάνω στο χάρτη στην τομή των δύο διευθύνσεων.

Τη μέθοδο αυτή χρησιμοποιεί το ραδιογωνιόμετρο (*radio direction finder*) και το πεπαλαιωμένο σύστημα CONSOL. Η δεύτερη μέθοδος στηρίζεται στον υπολογισμό της διαφοράς της απόστασης του πλοίου από δύο σταθερά σημεία γνωστού στίγματος. Ο γεωμετρικός τόπος των σημείων που παρουσιάζουν σταθερή διαφορά απόστασης από δύο άλλα, είναι μια υπερβολή με πόλους τα δύο αυτά σημεία. Έτσι, στο χάρτη μπορεί να χαραχθεί η αντίστοιχη υπερβολή η οποία ονομάζεται γραμμή θέσης (*Line Of Position – LOF*). Αν χρησιμοποιηθούν δύο ζεύγη σημείων γνωστών συντεταγμένων σχηματίζονται σύμφωνα με τον τρόπο που περιγράφτηκε δύο LOF, στην τομή τον οποίων βρίσκεται το στίγμα του πλοίου. Έτσι, τα συστήματα που χρησιμοποιούν αυτή τη μεθοδολογία ονομάζονται υπερβολικά (*hyperbolic*), με εκπροσώπους το Loran C, το Decca και το Omega. Η τρίτη μέθοδος βασίζεται στον υπολογισμό της απόστασης του πλοίου από κάποιο σταθμό εκπομπής. Η απόσταση αυτή μετράται ως χρονική διαφορά από την εκπομπή μέχρι τη λήψη ενός σήματος. Χαράζεται έτσι ένας κύκλος θέσης του πλοίου. Ένας δεύτερος κύκλος που χαράζεται με βάση την απόσταση του πλοίου από ένα δεύτερο σταθμό, δίδει δύο τομές, μία από τις οποίες αντιστοιχεί στο στίγμα του πλοίου. Συνήθως η θέση του πλοίου είναι τέτοια ώστε δεν υπάρχει αμφιβολία ως προς το ποια από τις δύο τομές αντιστοιχεί στη σωστή θέση. Η μέθοδος αυτή χρησιμοποιείται από συστήματα όπως το Mini-Ranger και ονομάζεται κυκλική (*range – range* ή *circular*).

Η δεύτερη υποκατηγορία αφορά δορυφορικά συστήματα εκπομπής Η/Μ κυμάτων. Η διαφορά με τα συστήματα στεριάς είναι ότι οι δορυφόροι είναι κινούμενοι σταθμοί και αυτό σημαίνει ότι θα πρέπει να παρέχεται στα πλοία πρόσθετη πληροφορία σχετικά με τη θέση τους, κάθε χρονική στιγμή. Εκπρόσωποι δορυφορικών συστημάτων ναυσιπλοΐας είναι το σύστημα Transit και το GPS. Παρά το γεγονός ότι το μεν σύστημα Transit υπολογίζει το στίγμα του κατ' αντιστοιχία με τα υπερβολικά συστήματα και το δε σύστημα GPS υπολογίζει αποστάσεις σε αντιστοιχία με τα κυκλικά συστήματα, διαφέρει ριζικά η διαδικασία προσδιορισμού της θέσης του πλοίου στη γη σε σχέση με τα συστήματα ξηράς.

Τέλος, απαραίτητη είναι η αναφορά σε ένα πολύ σημαντικό ναυτιλιακό βοήθημα, το ραντάρ (*Radio Detection And Ranging – RADAR*). Το RADAR χρησιμεύει στην ανίχνευση αντικειμένων – στόχων και τον υπολογισμό της διεύθυνσης και απόστασης του πλοίου στο οποίο είναι εγκατεστημένο από αυτούς. Τα αντικείμενα μπορεί να είναι πλοία, άλλα πλωτά μέσα ή ακόμη και στεριά. Έτσι, χρησιμεύει ιδιαίτερα στην αποφυγή συγκρούσεων, ιδιαίτερα όταν οι συνθήκες ορατότητας είναι πολύ κακές (ομίχλη, σκοτάδι, κλπ). Σε σχέση με τα άλλα συστήματα ναυσιπλοΐας, έχει την ιδιαιτερότητα ότι δεν απαιτεί τη συνεργασία άλλων σταθμών εκπομπής, είναι λοιπόν ένα αυτοτελές όργανο που μπορεί να εγκατασταθεί στο πλοίο και να λειτουργήσει σε οποιοδήποτε μέρος της Γης. Η αρχή λειτουργίας του RADAR είναι απλή. Παλμοί μικρής διάρκειας εκπέμπονται από κάποιο πομπό ραδιοκυμάτων με χρήση κεραίας που στέλνει τους παλμούς με τη μορφή κατευθυνόμενης μικρής δέσμης. Όταν οι παλμοί αυτοί ανακλαστούν σε κάποιο αντικείμενο επιστρέφουν προς τον πομπό και λαμβάνονται από την ίδια κεραία που τους εξέπεμψε. Επειδή οι παλμοί (Η/Μ κύματα) έχουν σταθερή ταχύτητα, μετρώντας το χρόνο από την εκπομπή μέχρι τη λήψη του παλμού, γνωρίζουμε την απόσταση που διήνυσε η οποία είναι η διπλάσια της απόστασης του πλοίου από το στόχο.

<u>Σχόλιο:</u> *Περισσότερες πληροφορίες σχετικά με τα συστήματα ναυσιπλοΐας υπάρχουν στη βιβλιογραφία (Ταρουδάκης, 1990).*

1.2 Σκοπός της εργασίας

Ο σκοπός της παρούσας εργασίας είναι να δημιουργηθεί ένα σύστημα ναυσιπλοΐας για μοντέλα (υπό κλίμακα) πλοίων. Το συγκεκριμένο σύστημα σχεδιάζεται σύμφωνα με κάποια κριτήρια τα οποία αφορούν το κόστος, τη διαθεσιμότητα υλικών υπό κλίμακα, καθώς και την απαιτούμενη ακρίβεια μετρήσεων, η οποία όπως γίνεται αντιληπτό απέχει τάξη μεγέθους από την ακρίβεια μετρήσεων που απαιτείται σε πραγματικά πλοία. Κατ' επέκταση, το σύστημα έχει τη δυνατότητα εφαρμογής και σε πραγματικά πλοία.

Έτσι, μελετάται σύστημα το οποίο βασίζεται στον δυναμικό εντοπισμό και έλεγχο της πορείας και θέσης σκάφους μέσω της ανάπτυξης ενός συστήματος παρακολούθησης φωτεινού στόχου που βρίσκεται στην ακτή ή σε άλλο προπορευόμενο σκάφος. Κατά συνέπεια, προτείνεται μια εικονική, οπτική γραμμή ρυμούλκησης του μοντέλου πλοίου, η οποία θα αντικαθιστά την υλική γραμμή ρυμούλκησης με καλώδιο.

Στις συνθήκες εργαστηρίου είναι πολύ χρήσιμο ένα τέτοιο σύστημα πλοήγησης, αφού θα υπάρχει η δυνατότητα διεξαγωγής πειραμάτων με αυτοπροωθούμενο σκάφος. Σύμφωνα με τον προτεινόμενο τρόπο διεξαγωγής πειραμάτων θα μετρώνται διάφορες παράμετροι με τη βοήθεια κατάλληλων διατάξεων – αισθητήρων επί του σκάφους και θα αποστέλλονται ασύρματα σε Η/Υ στην ξηρά προς επεξεργασία.

Σε φυσική κλίμακα, το συγκεκριμένο σύστημα ναυσιπλοΐας - ρυμούλκησης έχει τα εξής πλεονεκτήματα:

- Ταχύτερη σύνδεση – αποσύνδεση ρυμουλκούμενου – ρυμουλκού.
- Αυξημένη ασφάλεια.
- Δυνατότητα προσέγγισης απόκρημνων ακτών χωρίς λιμένες.
- Δυνατότητα ρυμούλκησης υπό άσχημες καιρικές συνθήκες.
- Διατήρηση αυτονομίας κίνησης.

Όπως γίνεται αντιληπτό, στόχος της παρούσας διπλωματικής εργασίας δεν είναι να αναλύσει εις βάθος κάθε κομμάτι της εργασίας που σχετίζεται με κάθε επιστημονικό κλάδο της ναυπηγικής (Υδροστατική, Υδροδυναμική, Αριθμητική Ανάλυση, Σύνθετα Υλικά κτλ). Ο λόγος είναι ότι η υλοποίηση περνά από πολλά στάδια: θεωρία – μελέτη – φυσικό μοντέλο – πειράματα – μαθηματικό μοντέλο – σύγκριση αποτελεσμάτων – αυτόματος έλεγχος. Επειδή όμως δεν είναι εφικτό στα πλαίσια της διπλωματικής εργασίας να αναλυθούν όλα τα πεδία εις βάθος χρησιμοποιούνται κάποια κομμάτια έτοιμα, χωρίς ανάλυση. Έτσι, χρησιμοποιώντας όλα αυτά τα πεδία και συνδυάζοντάς τα κατάλληλα επιτυγχάνεται ένα ολοκληρωμένο σύστημα. Άλλωστε η επιστήμη του ναυπηγού έχει αυτό το χαρακτηριστικό στη φύση της. Επιστημονική ανάλυση γίνεται στο κομμάτι του αυτομάτου ελέγχου – μαθηματικής μοντελοποίησης του πλοίου.

Τέλος, στην τελική του μορφή το σύστημα θα μπορούσε να αποτελέσει χρήσιμο εργαλείο των εργαστηρίων της σχολής Ναυπηγών Μηχανολόγων Μηχανικών.

1.3 Δομή της εργασίας

Η παρούσα διπλωματική εργασία απαρτίζεται από τα ακόλουθα κεφάλαια:

- Στο **πρώτο** κεφάλαιο γίνεται αναφορά σε διάφορα συστήματα ναυσιπλοΐας, δίνεται ξεκάθαρα ο σκοπός της εργασίας και παρουσιάζεται η δομή της.

- Στο **δεύτερο** κεφάλαιο αναλύονται τα κριτήρια επιλογής της γάστρας και του βασικού εξοπλισμού ενώ παρουσιάζονται τα βασικά χαρακτηριστικά τους. Επίσης, παρουσιάζεται ο τρόπος κατασκευής του συνολικού πλοίου και προσάρτησης σε αυτό του βασικού εξοπλισμού.

- Στο **τρίτο** κεφάλαιο αναλύονται τα κριτήρια επιλογής ενεργειακού συστήματος πάνω στο σκάφος και περιγράφεται η κατασκευή ενός ρυθμιστή τάσης στα 5 Volt. Τέλος, παρουσιάζεται σχηματικά ο τρόπος τροφοδοσίας όλων των συσκευών επί του πλοίου.

- Στο **τέταρτο** κεφάλαιο αναλύονται τα κριτήρια επιλογής του ηλεκτρονικού εξοπλισμού και περιγράφεται ο τρόπος επικοινωνίας των διαφόρων συσκευών επί του σκάφους αλλά και στο σταθμό ξηράς. Επίσης, περιγράφονται τα βασικά στοιχεία των ηλεκτρονικών μέσων που χρησιμοποιούνται (μικροελεγκτής, αισθητήρας εικόνας, πομποδέκτες κτλ). Τέλος, παρουσιάζονται σχηματικά όλη η «καλωδίωση» πάνω στο σκάφος.

- Στο **πέμπτο** κεφάλαιο πραγματοποιείται η μαθηματική μοντελοποίηση του συνολικού συστήματος. Μετά από την απαραίτητη εισαγωγή δίνεται η γενική μορφή των εξισώσεων κίνησης του πλοίου στο επίπεδο έχοντας σαν παραμέτρους κάποιους σταθερούς συντελεστές. Τέλος, προκειμένου να αντιστοιχήσουμε το οπτικό πεδίο του αισθητήρα εικόνας σε χρήσιμα φυσικά μεγέθη, πραγματοποιούνται κάποια πειράματα βαθμονόμησης του αισθητήρα.

- Στο **έκτο** κεφάλαιο γίνεται ταυτοποίηση συστήματος. Μετατρέπονται οι εξισώσεις του πέμπτου κεφαλαίου σε εξισώσεις στο χώρο κατάστασης. Προκειμένου να προσδιορισθούν οι σταθερές παράμετροι που εμπεριέχονται σε αυτές τις εξισώσεις γίνονται κάποια πειράματα. Σύμφωνα με τα δεδομένα των πειραμάτων υπολογίζονται οι παράμετροι μέσω μιας διαδικασίας βελτιστοποίησης η οποία πραγματοποιείται με τη βοήθεια Η/Υ. Τέλος, δίνεται το δομικό διάγραμμα του συστήματος πλοίο – κάμερα.

- Στο **έβδομο** κεφάλαιο αναφέρεται η έννοια του ελέγχου με ανάδραση και προσδιορίζεται ο νόμος ελέγχου που θα πραγματοποιηθεί στο υπό μελέτη σύστημα. Τέλος, παρουσιάζεται το πείραμα παρακολούθησης στόχου το οποίο θα αξιολογήσει τη μαθηματική μοντελοποίηση και ταυτοποίηση του συστήματος.

- Στο **όγδοο** κεφάλαιο παρουσιάζονται τα αποτελέσματα του πειράματος αξιολόγησης και σχολιάζονται τα διάφορα προβλήματα που προέκυψαν κατά τη διεξαγωγή του. Επίσης γίνεται σύγκριση των αποτελεσμάτων του πραγματικού με το εικονικό πείραμα. Τέλος, προτείνονται διάφορες βελτιώσεις και προσθήκες στο υπάρχον σύστημα για καλύτερα αποτελέσματα.

2. ΣΚΑΦΟΣ AIRCAT

2.1 Επιλογή σκάφους

Το μοντέλο πλοίου επιλέχθηκε μέσα από λίστα πλοίων σύμφωνα με τα εξής κριτήρια:

- Προτίμηση σε ταχύπλοο σκάφος.
- Διαθέσιμα μοντέλα πλοίων.
- Διαστάσεις σκάφους.

Έτσι, μετά από αναζήτηση σε καταλόγους εταιρειών, αποφασίστηκε ότι το σκάφος που θα χρησιμοποιηθεί για το σκοπό της παρούσας διπλωματικής εργασίας θα είναι το AIRCAT (Σχ. 2.1).

Σχήμα 2.1. *Το σκάφος AIRCAT.*

2.2 Χαρακτηριστικά σκάφους

Το σκάφος AIRCAT είναι τύπου SES (*Surface Effect Ship*). Τα SES αποτελούν υβριδικό τύπο πλοίου που συνδυάζει χαρακτηριστικά και πλεονεκτήματα των δίγαστρων πλοίων (*catamaran*) και αερόστρομνων (*hovercraft*). Πρόκειται ουσιαστικά για πλοία catamaran, εφοδιασμένα με ελαστικές «ποδιές» στην πλώρη και την πρύμνη τους, οι οποίες μαζί με τις δύο πλευρικές γάστρες αποτελούν τα όρια του στρώματος αέρα. Κατ'αυτόν τον τρόπο το συνολικό μήκος της «ποδιάς» μειώνεται δραστικά, και μαζί με αυτό μειώνονται τα προβλήματα φθορών και συντήρησης που παρουσιάζονται στα αμιγή hovercraft. Ταυτόχρονα, στερούμενα βέβαια τον «αμφίβιο» χαρακτήρα των hovercraft, τα SES χρησιμοποιούν για την πρόωσή τους, αντί των όχι τόσο αξιόπιστων ελίκων αέρος, συμβατικές έλικες ή συχνότερα προωθητήρες αντίδρασης (*waterjet*). Σε σχέση με τα συμβατικά catamaran, τα SES παρουσιάζουν μειωμένη αντίσταση και καλύτερη συμπεριφορά σε κυματισμούς (μέχρι ένα ύψος κύματος), αλλά παραμένουν πολύπλοκα στην κατασκευή, λειτουργία και συντήρηση.

Όταν χρησιμοποιείται ο ανεμιστήρας ανύψωσης (*blower*), ο οποίος δημιουργεί το στρώμα αέρα, προστίθεται αεροστατική άντωση στην υπάρχουσα υδροστατική. Με αυτό τον τρόπο μειώνεται το βύθισμα κι έτσι ένα μικρό μόνο τμήμα της διπλής γάστρας παραμένει κάτω από την επιφάνεια του νερού με αποτέλεσμα να ελαττώνεται η αντίσταση. Είναι απαραίτητο ένα τμήμα της γάστρας να παραμένει κάτω από την επιφάνεια του νερού ώστε οι εισαγωγές των waterjets να «ρουφάνε» νερό και όχι αέρα. Όταν δεν χρησιμοποιείται το blower, όλο το βάρος του πλοίου ισορροπεί στο νερό μόνο λόγω υδροστατικής άντωσης. Έτσι έχουμε αυξημένο βύθισμα άρα και αυξημένη αντίσταση. Τα βασικά χαρακτηριστικά του AIRCAT παρουσιάζονται παρακάτω (Πνκς 2.1).

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τα πλοία τύπου SES υπάρχουν στη βιβλιογραφία (Ζαραφωνίτης, 2002).*

Πίνακας 2.1
Βασικά χαρακτηριστικά σκάφους.

Όνομα		AIRCAT	
Τύπος		SES	
Εταιρεία		KMB	
Έτος κατασκευής		2005	
Κλίμακα		1:30	
L_{OA}	mm	1250	Ολικό μήκος
L_{BP}	mm	1200	Μήκος μεταξύ καθέτων
B	mm	400	Πλάτος
D	mm	135	Κοίλο
T_H	mm		Βύθισμα υδροστατικής άντωσης
T_A	mm		Βύθισμα μέγιστης αεροστατικής άντωσης
Δ	gr		Εκτόπισμα

2.3 Επιλογή βασικού εξοπλισμού

Η εταιρεία KMB παρείχε πλήρες πακέτο προς συναρμολόγηση, προσφέροντας όλο τον βασικό εξοπλισμό που υποστηρίζει το συγκεκριμένο σκάφος. Δυνατότητα επιλογής είχαμε στους ηλεκτρικούς κινητήρες, τους ηλεκτρονικούς ελεγκτές στροφών και τους σερβομηχανισμούς πηδαλιουχίας και αναπόδισης. Ο συγκεκριμένος εξοπλισμός επιλέχθηκε σύμφωνα με τα εξής κριτήρια:

- Κόστος.
- Διαστάσεις.
- Βάρος.
- Ιδιαίτερα χαρακτηριστικά.

Ο βασικός εξοπλισμός του πλοίου αποτελείται από τα εξής:

- Γάστρα.
- Υπερκατασκευή.
- Σύστημα πρόωσης.
- Σύστημα πηδαλιουχίας – αναπόδισης.
- Σύστημα ανεμιστήρα ανύψωσης.
- Ελαστική ποδιά.

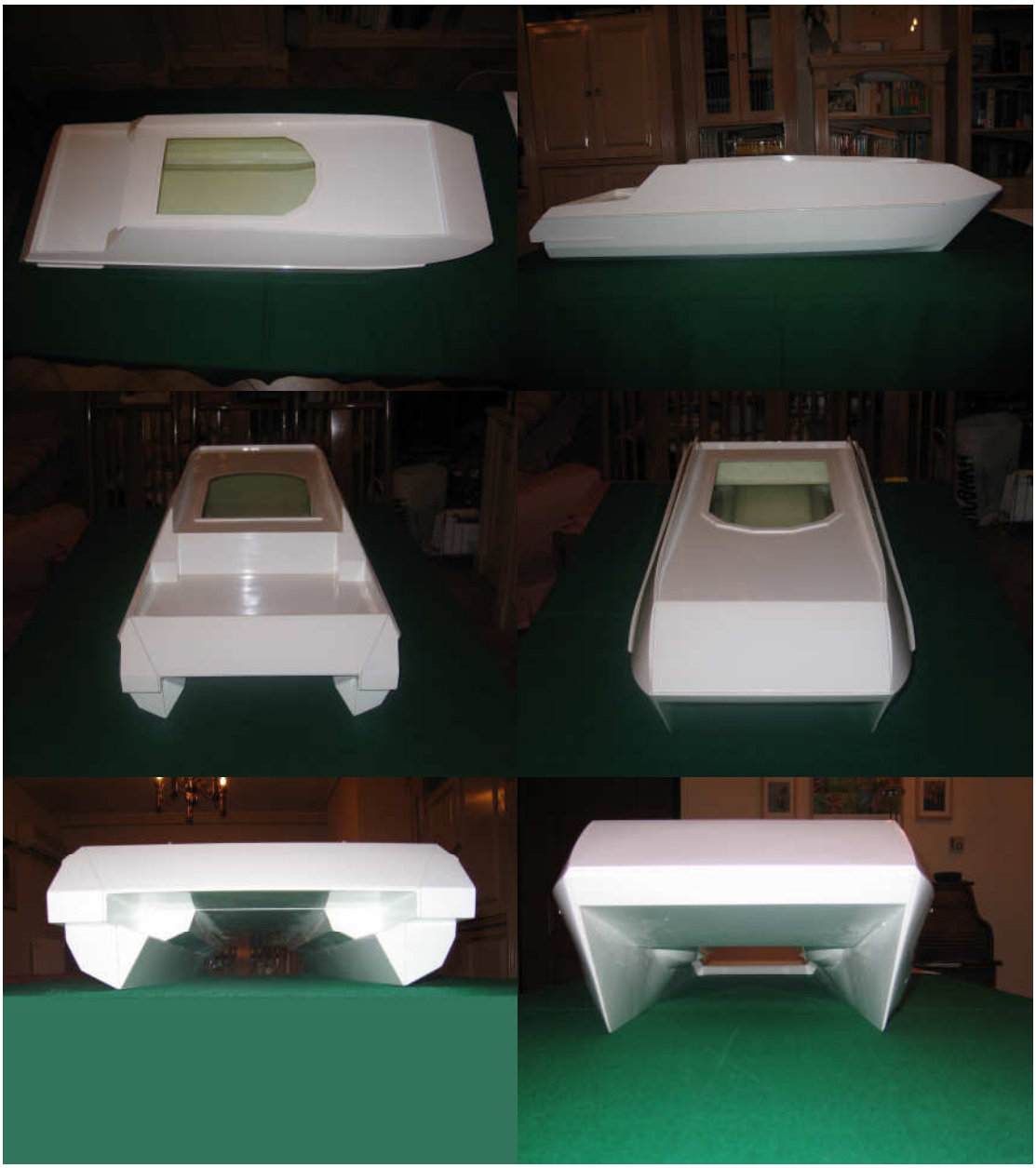
Στη συνέχεια παρουσιάζεται ο βασικός εξοπλισμός ανά κατηγορία. Παρακάτω φαίνεται το πλοίο πριν συναρμολογηθεί (Σχ. 2.2).

Σχήμα 2.2. *Πλοίο και βασικός εξοπλισμός.*

2.4 Βασικός εξοπλισμός πλοίου AIRCAT

2.4.1 Γάστρα

Το υλικό κατασκευής της γάστρας είναι εποξική ρητίνη ενισχυμένη με ϋαλόπανο. Για να είναι λεία η εξωτερική επιφάνεια είναι καλυμμένη με gelcoat. Στη συνέχεια παρουσιάζεται η αρχική μορφή της γάστρας (Σχ. 2.3).



Σχήμα 2.3. *Όψεις γάστρας.*

Αρχικά στο εσωτερικό αλλά και στα εξωτερικά τμήματα της γάστρας δεν υπήρχαν υποδοχές και βάσεις ώστε να στηριχθεί ο υπόλοιπος εξοπλισμός (Σχ. 2.4).

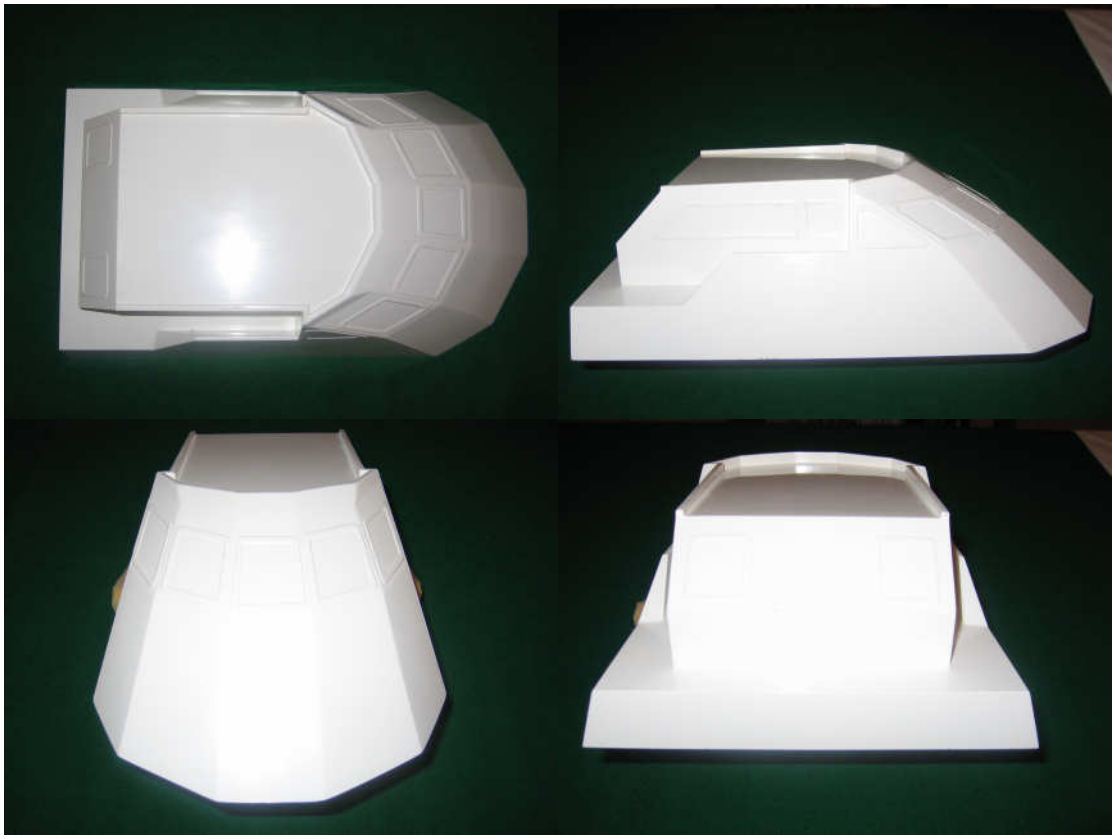
Σχήμα 2.4. *Λεπτομέρειες γάστρας.*

Στη συνέχεια η γάστρα μορφοποιείται ώστε να δεχθεί τα λειτουργικά μέρη (water jet, blower, κτλ). Έτσι, διαμορφώνονται οπές, ενισχυτικά και βάσεις (Σχ. 2.5).

Σχήμα 2.5. *Δημιουργία οπών.*

2.4.2 Υπερκατασκευή

Όπως και η γάστρα έτσι και η υπερκατασκευή είναι κατασκευασμένη από εποξική ρητίνη ενισχυμένη με ϋαλόπανο, έχοντας την εξωτερική της επιφάνεια καλυμμένη με gelcoat. Στη συνέχεια παρουσιάζεται η αρχική μορφή της υπερκατασκευής (Σχ. 2.6).

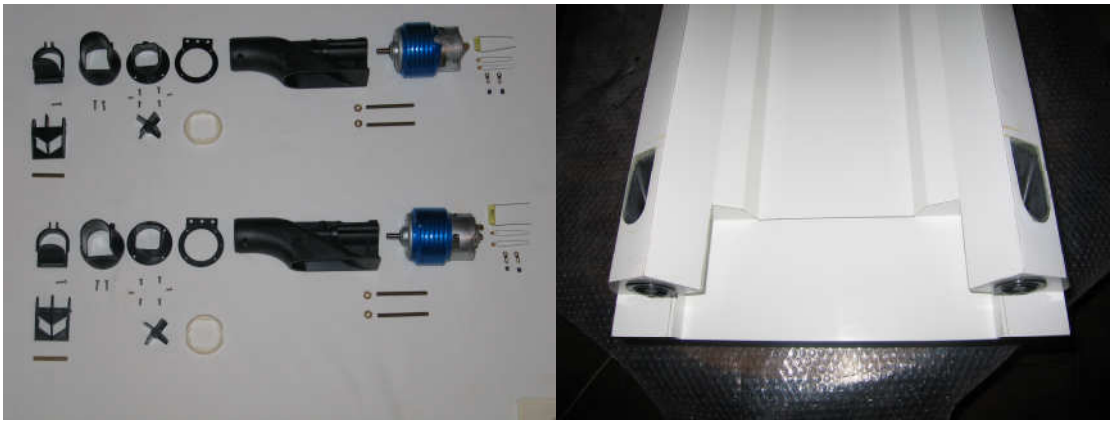


Σχήμα 2.6. *Όψεις υπερκατασκευής.*

Η υπερκατασκευή δε μορφοποιήθηκε σημαντικά, μιας και ο βασικός της ρόλος στο μοντέλο του πλοίου είναι να προστατεύει το εσωτερικό του πλοίου από διαβροχή.

2.4.3 Σύστημα πρόωσης

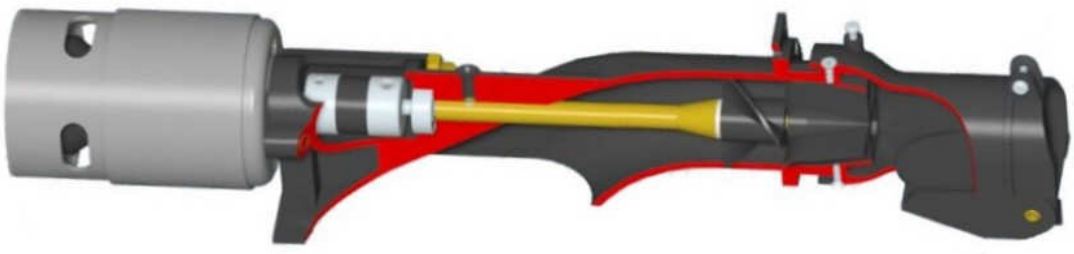
Το σύστημα πρόωσης (*thruster*) του AIRCAT αποτελείται από δύο προωθητήρες αντίδρασης (*waterjet*) οι οποίοι τοποθετούνται στο εσωτερικό του πυθμένα του πρυμναίου τμήματος της διπλής γάστρας (Σχ. 2.7).



Σχήμα 2.7. *Water jets πριν και μετά την τοποθέτησή τους στο σκάφος.*

Αυτό το σύστημα χρησιμοποιείται σε πλοία που κινούνται με ιδιαίτερα μεγάλες ταχύτητες. Από έναν ειδικά διαμορφωμένο αγωγό στην πρύμνη του πλοίου αναρροφάται νερό το οποίο οδηγείται στην φτερωτή (*impeller*) και από εκεί στη συνέχεια στο πρυμναίο ακροφύσιο, πάνω από τη στάθμη του νερού. Από το ακροφύσιο το νερό εκτινάσσεται με ορμή πίσω από το πλοίο. Η λειτουργία του προωθητήρα αντίδρασης βασίζεται στην αρχή διατήρησης της ορμής (η ώση που αναπτύσσεται είναι αποτέλεσμα της ορμής που προσδίδεται στο ρεύμα του νερού μέσω της πτερωτής). Στη συνέχεια παρουσιάζεται μια τομή ενός προωθητήρα αντίδρασης, όπου μπορεί να παρατηρήσει κανείς τον αγωγό εισαγωγής που ξεκινά από τον πυθμένα του σκάφους, την πτερωτή και το ακροφύσιο εξόδου (Σχ.2.8). Επίσης φαίνεται ο άξονας που κινεί την φτερωτή, συνδεδεμένος με τον ηλεκτρικό κινητήρα. Η διάταξη που φαίνεται σαν προέκταση του ακροφυσίου εξόδου χρησιμεύει για τις λειτουργίες της πηδαλιουχίας και της αναπόδισης οι οποίες αναλύονται σε επόμενη παράγραφο.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τα waterjet υπάρχουν στη βιβλιογραφία (Ζαραφωνίτης, 2002 – Ιωαννίδης, 2005).*

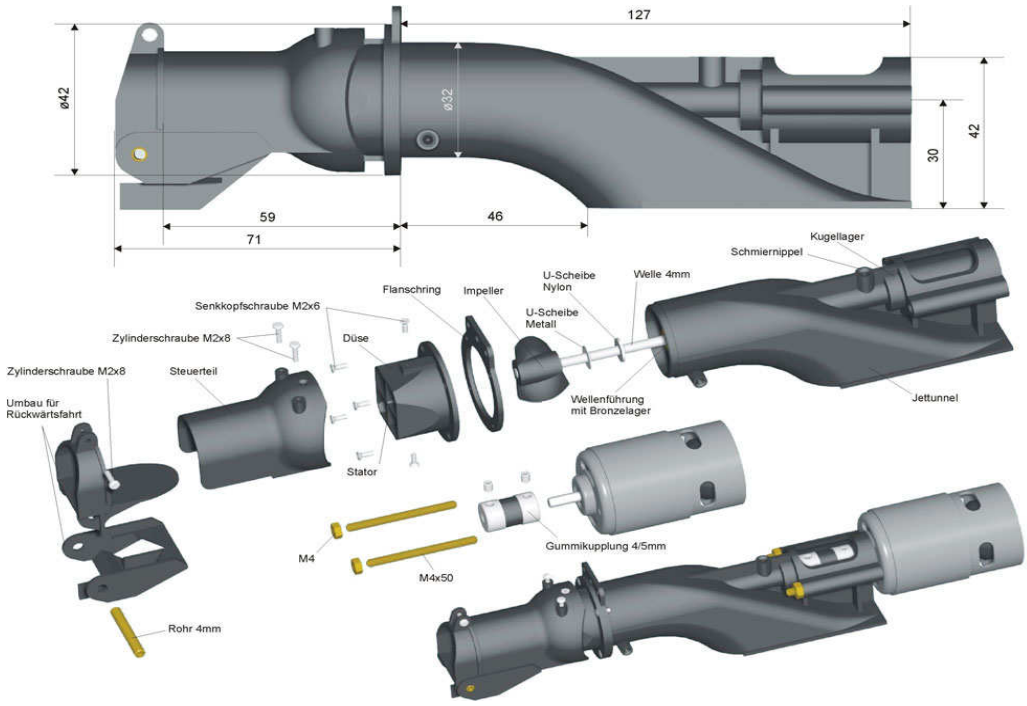


Σχήμα 2.8. *Τομή προωθητήρα αντίδρασης.*

Στη συνέχεια δίνονται τα κύρια χαρακτηριστικά του συστήματος πρόωσης που χρησιμοποιείται στο σκάφος AIRCAT (Πνκς 2.2) καθώς και σκαρίφημα αυτού (Σχ. 2.9).

Πίνακας 2.2
Κύρια χαρακτηριστικά συστήματος πρόωσης.

Εύρος λειτουργίας	RPM	15000 - 20000	Ταχύτητα περιστροφής
Στατική ώση	W	200 - 400	Ισχής
	N	24,5 - 39,2	Δύναμη
Διαστάσεις σήραγγας	mm	130	Μήκος
	mm	38	Πλάτος
	mm	45	Ύψος
Φτερωτή	mm	28	Διάμετρος
Διαστάσεις αξονικού συστήματος	mm	125	Μήκος
	mm	4	Διάμετρος
Ηλεκτρικοί κινητήρες		2 x DC JOHNSON-700-NEODYM 60 mm	
Ηλεκτρονικός ρυθμιστής στροφών		1 x KMB 7-30 NC 70 A NAVY	
Μπαταρίες		2 x 12V NiMH 3300mAh	



Σχήμα 2.9. *Σκαρίφημα χρησιμοποιούμενου συστήματος πρόωσης.*

2.4.4 Σύστημα ανεμιστήρα ανύψωσης

Το σύστημα ανεμιστήρα ανύψωσης (*blower*) του AIRCAT αποτελείται από έναν ηλεκτρικό κινητήρα στου οποίου το σώμα προσαρμόζεται σταθερή φτερωτή ενώ στον άξονά του προσαρμόζεται περιστρεφόμενη φτερωτή. Ο κινητήρας με τις δύο φτερωτές τοποθετείται στο εσωτερικό δακτυλίου ο οποίος στηρίζεται σε έναν αγωγό διοχέτευσης αέρα (Σχ. 2.10).

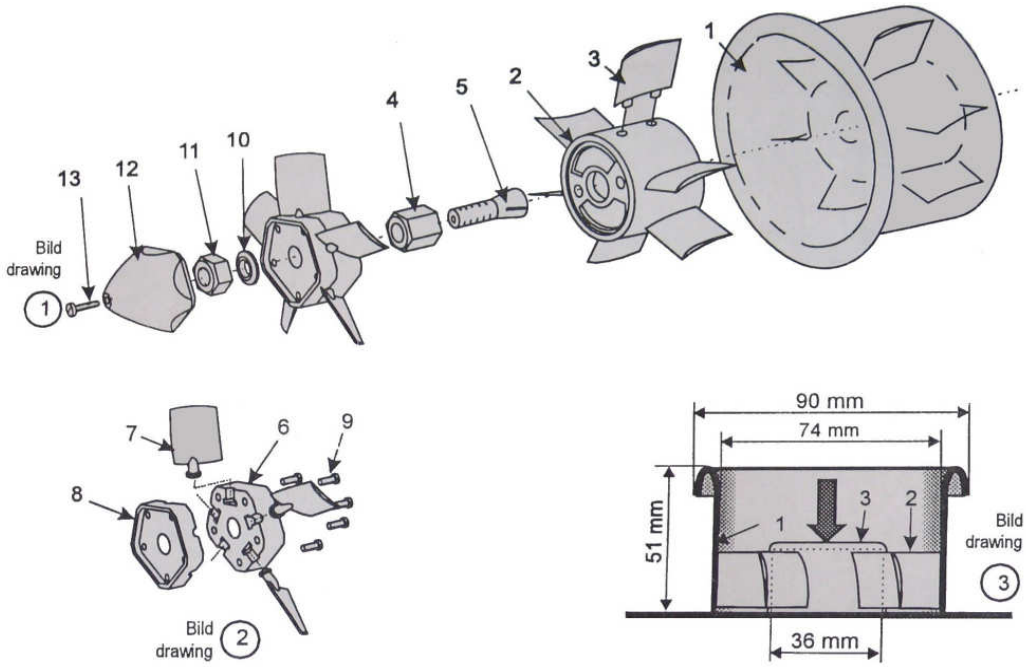
Σχήμα 2.10. *Αγωγός διοχέτευσης αέρα με δακτύλιο, ηλεκτρικός κινητήρας με φτερωτές και συναρμολογημένο blower.*

Το blower τοποθετείται στο μέσον του πλοίου πάνω από τις οπές ώστε το ρεύμα αέρα να περνά στον κενό χώρο ανάμεσα στις δύο γάστρες. Ο άξονας του ανεμιστήρα είναι παράλληλος με τον εγκάρσιο άξονα του πλοίου. Μεταξύ γάστρας και blower παρεμβάλλεται στεγανωτικό αφρώδες υλικό ώστε να μην έχουμε απώλειες αέρα. Στα πλάγια της γάστρας του πλοίου ανοίγονται οπές για εισαγωγή αέρα.

Τέλος, δίνονται τα κύρια χαρακτηριστικά του συστήματος ανεμιστήρα ανύψωσης που χρησιμοποιείται στο σκάφος AIRCAT (Πνκς 2.3) καθώς και σκαρίφημα αυτού (Σχ. 2.11).

Πίνακας 2.3
Κύρια χαρακτηριστικά συστήματος ανεμιστήρα ανύψωσης.

Εύρος λειτουργίας	RPM	15000 - 20000	Ταχύτητα περιστροφής
Στατική ώση	W	130 - 300	Ισχής
	N	5,2 - 6,5	Δύναμη
Διαστάσεις δακτυλίου	mm	90	Εξσωτερική διάμετρος
	mm	74	Εσωτερική διάμετρος
	mm	51	Ύψος
Φτερωτή	mm	73	Διάμετρος
Διαστάσεις αγωγού διοχέτσυσης αέρα	mm	206	Μήκος βάσης
	mm	144	Πλάτος βάσης
	mm	90	Συνολικό ύψος
Ηλεκτρικος κινητήρας		1 x DC GRAUPNER-SPEED-600 50 mm	
Ηλεκτρονικός ρυθμιστής στροφών		1 x KMB 6-14 NC 40 A NAVY	
Μπαταρίες		1 x 12V NiMH 3300mAh	



Σχήμα 2.11. *Σκαρίφημα χρησιμοποιούμενου συστήματος ανεμιστήρα ανύψωσης.*

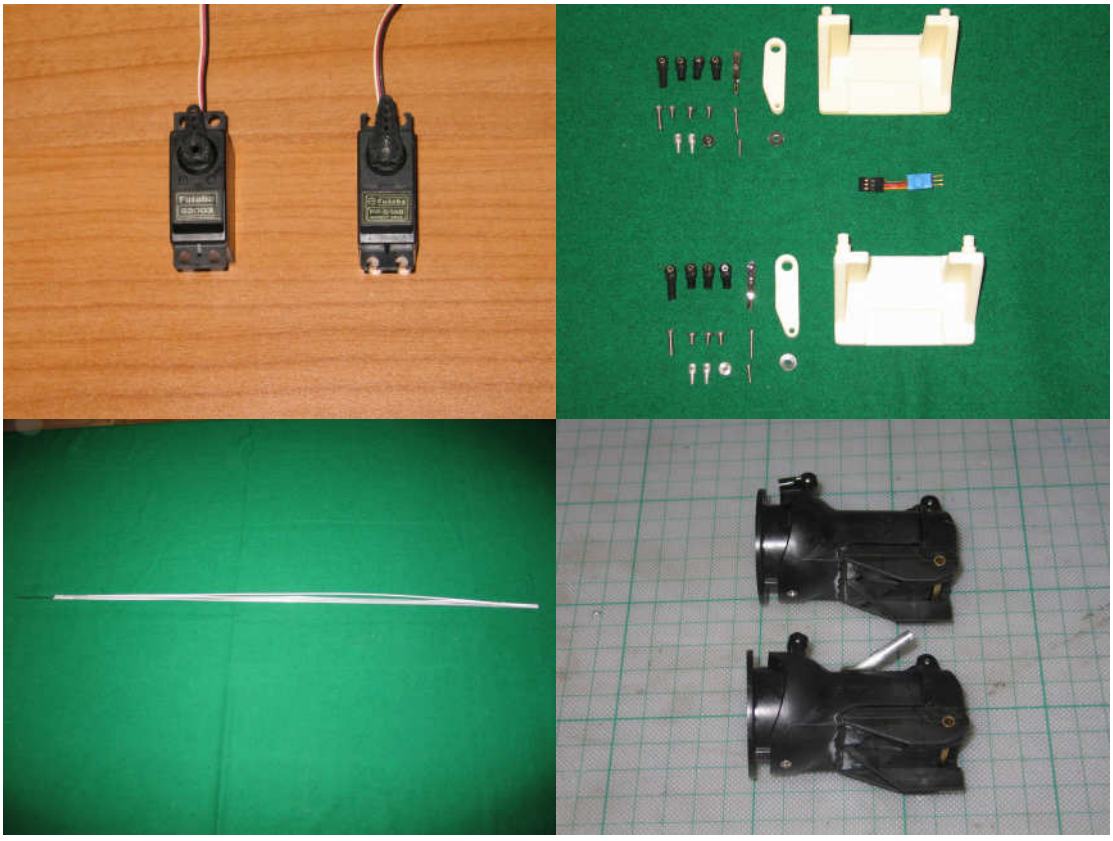
2.4.5. Σύστημα πηδαλιουχίας - αναπόδισης

Η διάταξη που βρίσκεται μετά την έξοδο του ακροφυσίου του waterjet (Σχ. 2.8) εξυπηρετεί δύο σκοπούς:

- Εκτροπή της δέσμης του νερού αριστερά ή δεξιά (πηδαλιουχία).
- Εκτροπή της δέσμης του νερού προς τα εμπρός (αναπόδιση).

Οι βασικοί μηχανισμοί του συστήματος πηδαλιουχίας (*rudder*) – αναπόδισης (*reverser*) που διαθέτει το AIRCAT είναι:

- Σερβομηχανισμοί.
- Βάσεις σερβομηχανισμών.
- Ντίζες μεταφοράς κινήσεων.
- Ακροφύσια ελιγμών.



Σχήμα 2.12. *Μηχανισμοί συστήματος πηδαλιουχίας – αναπόδισης.*

Το πλοίο AIRCAT διαθέτει δύο τέτοια ολοκληρωμένα συστήματα, ένα για κάθε waterjet.

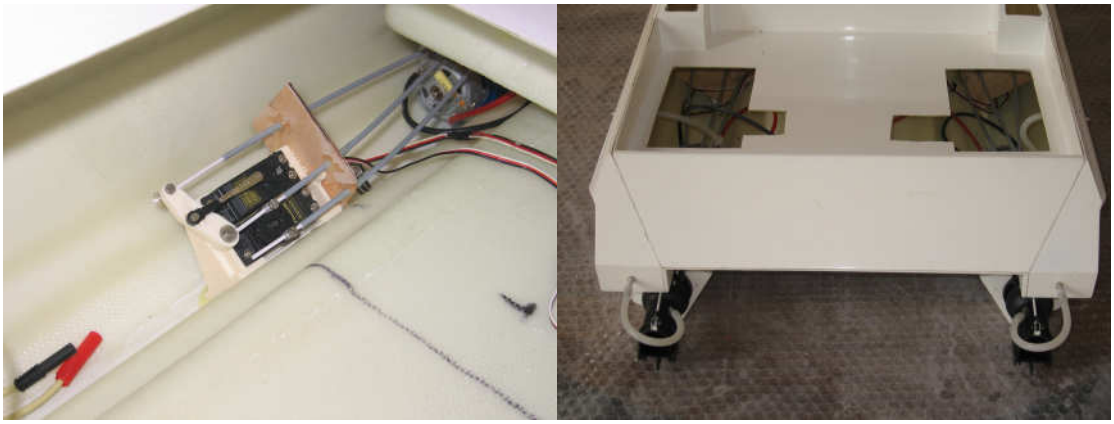
Στο ακροφύσιο ελιγμών του κάθε waterjet είναι ενσωματωμένος αγωγός ο οποίος εκμεταλλεύεται την υπάρχουσα πίεση του νερού και προμηθεύει το κύκλωμα ψύξης του σκάφος με νερό. Αυτό το νερό ψύχει τους ηλεκτρικούς κινητήρες και τους ηλεκτρονικούς ρυθμιστές στροφών (Σχ. 2.13).

Σχήμα 2.13. *Το σύστημα ψύξης.*

Τέλος, δίνονται τα κύρια χαρακτηριστικά του συστήματος πηδαλιουχίας - αναπόδισης που χρησιμοποιείται στο σκάφος AIRCAT (Πνκς 2.4) καθώς και όψεις του εγκατεστημένου συστήματος (Σχ. 2.14).

Πίνακας 2.4
Κύρια χαρακτηριστικά συστήματος πηδαλιουχίας - αναπόδισης.

Διαστάσεις ακροφύσιου ελιγμών	mm	48	Μήκος
	mm	40	Πλάτος
	mm	51	Ύψος
Σερβομηχανισμοί πηδαλιουχίας		2 x FUTABA S3003	
Σερβομηχανισμοί αναπόδισης		2 x FUTABA FP-S148	



Σχήμα 2.14. *Το σύστημα πηδαλιουχίας – αναπόδισης εγκατεστημένο.*

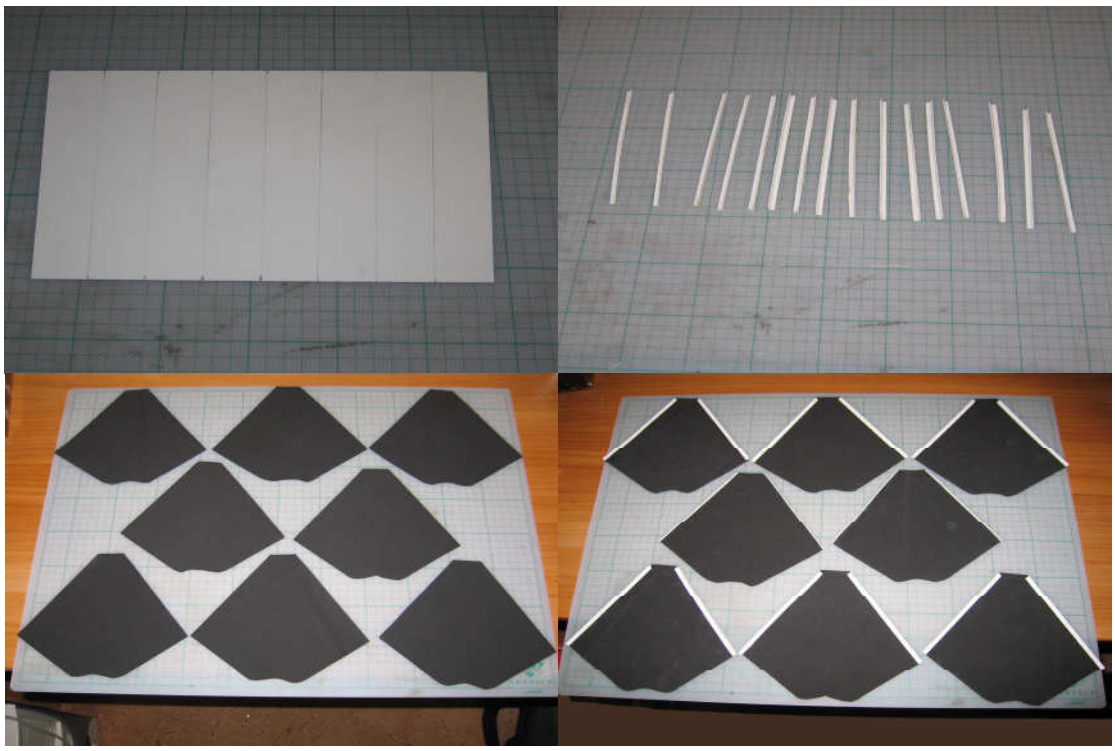
2.4.6 Ελαστική ποδιά

Η ελαστική ποδιά είναι κατασκευασμένη από πλαστικό φύλο πάχους 1mm (βάσεις, στηρίγματα) και αφρώδες ελαστικό υλικό (ποδιά). Παρακάτω παρουσιάζονται τα υλικά πριν διαμορφωθούν (Σχ. 2.15).

Σχήμα 2.15. *Πλαστικό φύλο πάχους 1mm και αφρώδες ελαστικό υλικό.*

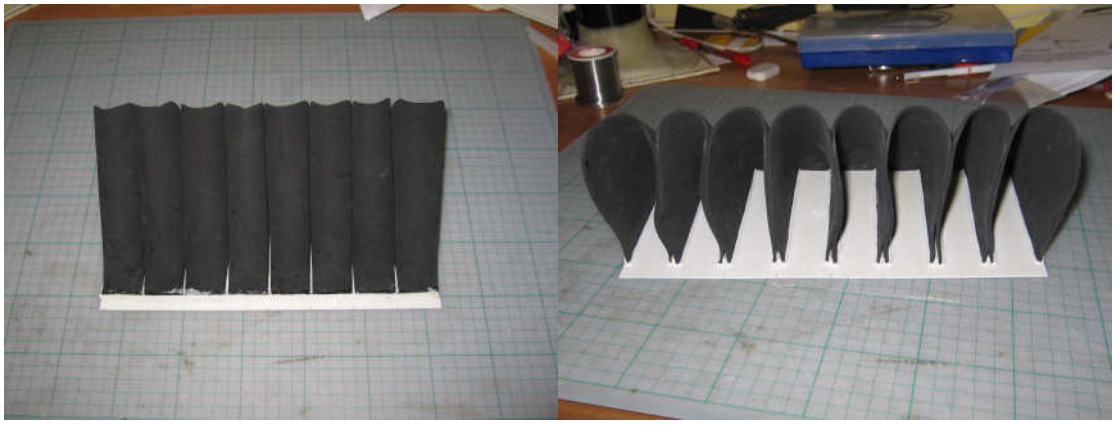
- **Περιγραφή κατασκευής πρωραίου τμήματος ελαστικής ποδιάς**

Στη συνέχεια βλέπουμε την κύρια βάση, τα στηρίγματα της ποδιάς και τα κομμάτια της ελαστικής ποδιάς πριν και αφού προσαρμοστούν σε αυτά τα στηρίγματα (Σχ. 2.16).



Σχήμα 2.16. *Κύρια βάση, στηρίγματα και κομμάτια ελαστικής ποδιάς.*

Ύστερα, τα κομμάτια της ελαστικής ποδιάς με τη βοήθεια των στηριγμάτων προσαρμόζονται στην κύρια βάση (Σχ. 2.17).



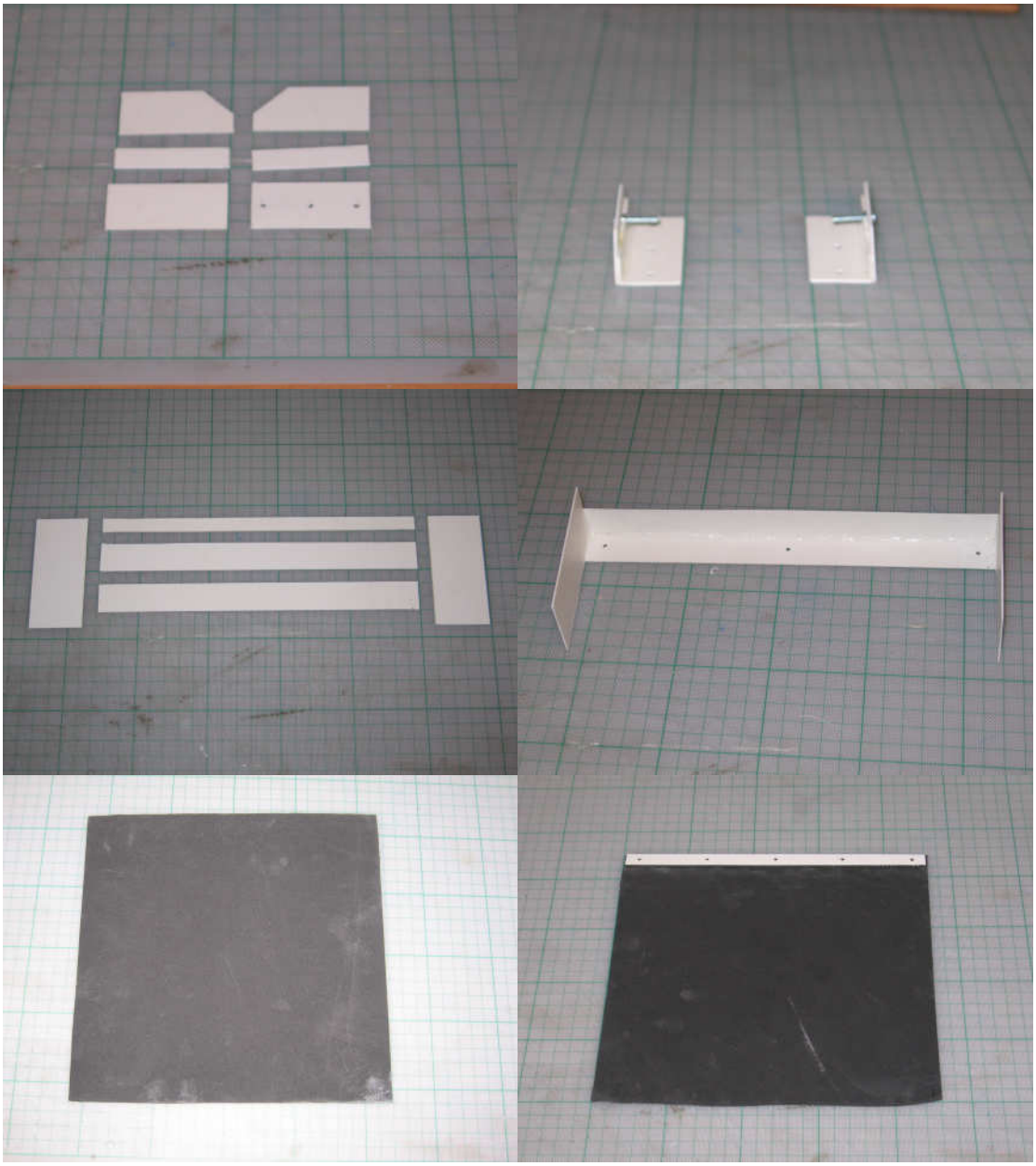
Σχήμα 2.17. *Όψεις ολοκληρωμένου πρωραίου τμήματος ελαστικής ποδιάς.*

Τέλος, το πρωραίο τμήμα τοποθετείται στο κάτω μέρος της γάστρας στεγανοποιώντας τις οπές στήριξης και τα σημεία επαφής με σιλικόνη (Σχ. 2.18).

Σχήμα 2.18. *Εγκατάσταση πρωραίας ποδιάς στη γάστρα.*

- **Περιγραφή κατασκευής πρυμναίου τμήματος ελαστικής ποδιάς**

Στο σχήμα 2.19 βλέπουμε τις βάσεις πριν και μετά τη συναρμολόγησή τους και την ελαστική ποδιά πριν και αφού προσαρμοστεί σε αυτή το στήριγμα.



Σχήμα 2.19. *Μπροστινή βάση, πίσω βάση και ελαστική ποδιά.*

Τέλος, στηρίχτηκαν οι βάσεις του πρυμναίου τμήματος στο κάτω μέρος της γάστρας στεγανοποιώντας τις οπές στήριξης και τα σημεία επαφής με σιλικόνη. Επίσης διαμορφώθηκε η ελαστική ποδιά ώστε να εφαρμόζει καλύτερα στη γάστρα και τοποθετήθηκε στις βάσεις (Σχ. 2.20).

Σχήμα 2.20. *Τελική διαμόρφωση και τοποθέτηση πρυμναίου τμήματος ελαστικής ποδιάς.*

3. ΗΛΕΚΤΡΟΛΟΓΙΚΗ ΕΓΚΑΤΑΣΤΑΣΗ ΠΛΟΙΟΥ

3.1 Επιλογή ενεργειακού συστήματος

Ο εξοπλισμός του πλοίου λειτουργεί με ηλεκτρική ενέργεια συνεχούς ρεύματος. Έτσι, υπάρχει αναγκαιότητα αναζήτησης κατάλληλου ενεργειακού συστήματος το οποίο θα προσφέρει την απαιτούμενη ενέργεια. Επειδή οι διαστάσεις του σκάφους δε μας επιτρέπουν τη χρήση γεννήτριας, η ηλεκτρική ενέργεια πρέπει να παραχθεί με άλλο τρόπο. Τα κριτήρια επιλογής του ενεργειακού συστήματος είναι:

- Κόστος.
- Διάρκεια αυτονομίας.
- Διαστάσεις.
- Βάρος.
- Ιδιαίτερα χαρακτηριστικά.

Η λύση στο πρόβλημα επιλογής ενεργειακού συστήματος δίνεται από πακέτα μπαταριών οι οποίες αποτελούνται από στοιχεία νικελίου - μετάλλου υδριδίου (*NiMH*). Η NiMH είναι τύπος επαναφορτιζόμενης μπαταρίας παρόμοια με τη μπαταρία νικελίου - καδμίου (*NiCd*). Η διαφορά τους είναι ότι η NiMH χρησιμοποιεί κράμα που απορροφά υδρογόνο (*hydrogen - absorbing alloy*) για το αρνητικό ηλεκτρόδιο αντί για κάδμιο. Όπως και στις μπαταρίες NiCd, το θετικό ηλεκτρόδιο είναι οξύ - υδροξείδιο του νικελίου (*NiOOH*).

Το κύριο πλεονέκτημα των μπαταριών NiMH είναι ότι έχουν 2 - 3 φορές μεγαλύτερη χωρητικότητα σε σχέση με μια αντίστοιχου μεγέθους NiCd. Αυτός είναι ο κύριος λόγος που επελέγησαν αυτού του είδους οι μπαταρίες για τη συγκεκριμένη εφαρμογή. Λόγω αυξημένων ενεργειακών αναγκών στο σκάφος, με τη χρήση μπαταριών NiMH μειώνεται σημαντικά το συνολικό βάρος, που είναι επιθυμητό.

Επίσης οι NiMH έχουν μεγάλο ρυθμό εκφόρτισης, δηλ. μπορούν να δώσουν υψηλή ένταση στον καταναλωτή. Επειδή οι ηλεκτρικοί κινητήρες λειτουργούν σε μεγάλες εντάσεις ρεύματος, ο αυξημένος ρυθμός εκφόρτισης ήταν ένα επιπλέον πλεονέκτημα των μπαταριών NiMH.

Ο βασικός εξοπλισμός του πλοίου AIRCAT που λειτουργεί με ηλεκτρική ενέργεια είναι:

- Ηλεκτρικοί κινητήρες προώσεως και ανεμιστήρα ανύψωσης.
- Ηλεκτρονικοί ρυθμιστές στροφών (*Electronic Speed Controller – ESC*) των κινητήρων προώσεως και ανεμιστήρα ανύψωσης.
- Σερβομηχανισμοί (*servo*) πηδαλιουχίας, αναπόδισης και κίνησης βάσης αισθητήρα εικόνας.
- Πομποδέκτης (*transceiver*).
- Αισθητήρας εικόνας (*vision sensor*).
- Προσαρμογέας σειριακής θύρας (*RS232 adapter*).
- Μικροελεγκτής (*microcontroller*).

Έτσι κατασκευάζονται δύο ειδών πακέτα μπαταριών τα οποία καλύπτουν όλες τις ενεργειακές ανάγκες του πλοίου:

- Είδος 1 (ρεύματα ισχύος): 12V, 3300mAh (x 3).
- Είδος 2 (ασθενή ρεύματα): 8,4V, 2100mAh (x1).

3.2 Ρύθμιση τάσης

Ο χρησιμοποιούμενος εξοπλισμός λειτουργεί σε διάφορα επίπεδα τάσης όπως φαίνεται παρακάτω (Πνκς 3.1). Οι μπαταρίες προσφέρουν 8,4V και 12V. Η απαιτούμενες τάσεις δημιουργούνται από ειδικούς ρυθμιστές τάσης (*voltage regulator*).

Πίνακας 3.1
Επίπεδα τάσης λειτουργίας βασικών καταναλωτών.

Καταναλωτές	**Τάση**
-	**V**
Ηλεκτρικοί κινητήρες προώσεως και ανεμιστήρα ανύψωσης	12,0
Ηλεκτρονικοί ρυθμιστές στροφών των κινητήρων προώσεως και ανεμιστήρα ανύψωσης	5,0
Σερβομηχανισμοί πηδαλιουχίας αναπόδισης και κίνησης αισθητήρα εικόνας	5,0
Πομποδέκτης	8,4
Αισθητήρας εικόνας	8,4
Προσαρμογέας σειριακής θύρας	5,0
Μικροελεγκτής	8,4

Ο επεξεργαστής που βρίσκεται στην αναπτυξιακή κάρτα του μικροελεγκτή λειτουργεί σε τάσεις 1,8V και 3,3V, ενώ η κάρτα του αισθητήρα εικόνας σε τάσεις 3,3V και 5,0V. Οι κάρτες αυτές διαθέτουν ενσωματωμένους ρυθμιστές τάσης οι οποίοι δέχονται στην είσοδό τους 8,4V και παράγουν τις επιθυμητές εξόδους. Οι παραπάνω προσφερόμενες τάσεις (1,8V, 3,3V και 5,0V) δε μπορούν να χρησιμοποιηθούν έξω από τα όρια της εκάστοτε κάρτας. Έτσι, δε μπορούμε να εκμεταλλευτούμε τη δημιουργία των 5,0V, η οποία όπως γίνεται αντιληπτό από τον παραπάνω πίνακα χρησιμοποιείται από αρκετούς καταναλωτές.

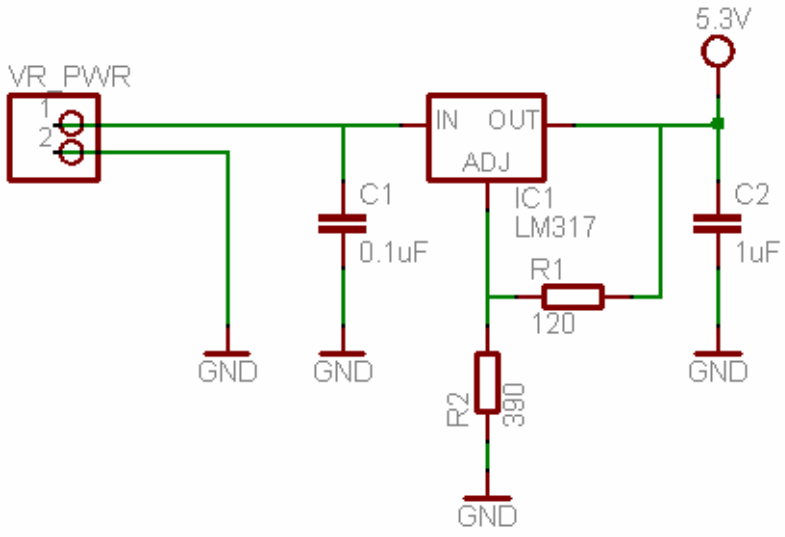
Έτσι, κατασκευάζεται ένας ρυθμιστής τάσης ο οποίος τροφοδοτεί με τάση 5,0V όλους τους σερβομηχανισμούς, τους ηλεκτρονικούς ρυθμιστές στροφών και τον προσαρμογέα σειριακής θύρας.

Πριν κατασκευαστεί ο ρυθμιστής, πρέπει να γίνει ηλεκτρικός ισολογισμός ώστε να γνωρίζουμε τις ενεργειακές ανάγκες των συγκεκριμένων καταναλωτών και να διαστασιολογηθεί η κατασκευή. Η στοιχειοθέτηση του ισολογισμού προκύπτει από πειραματικές μετρήσεις και από δεδομένα εγχειριδίων (Πνκς 3.2).

Πίνακας 3.2
Ηλεκτρικός Ισολογισμός

Καταναλωτές	**Πλήθος**	**Τάση**	**Κατανάλωση Χειρισμών**
-	-	V	mA
Rudder Servos	2	5,0	400
Reverser Servos	2	5,0	400
Pan Servo	1	5,0	200
Tilt Servo	1	5,0	200
Thruster ESC	1	5,0	40
Blower ESC	1	5,0	40
MAX232	1	5,0	10
ΣΥΝΟΛΟ			**1290**

Για να εξασφαλίσουμε ότι σε κάθε περίπτωση διατίθεται η απαιτούμενη ισχύς από το ρυθμιστή τάσης, οι υπολογισμοί έγιναν στη δυσμενέστερη περίπτωση (κατανάλωση χειρισμών). Έτσι, αποφασίστηκε να χρησιμοποιηθεί ρυθμιστής ο οποίος μπορεί να τροφοδοτήσει το σύστημα με ρεύμα έντασης έως 1500mA. Επίσης, αποφασίστηκε να σχεδιαστεί έτσι ο ρυθμιστής ώστε να δίνει τάση εξόδου 5,3V. Στη συνέχεια παρουσιάζεται το σχέδιο της κατασκευής (Σχ. 3.1).



Σχήμα 3.1. *Ρυθμιστής τάσης 5,3V.*

Το εγχειρίδιο του ρυθμιστή (LM317) δίνει την παρακάτω σχέση:

$$V_{OUT} = 1{,}25 \cdot \left(1 + \frac{R_2}{R_1}\right) + I_{ADJ} \cdot R_2$$

Το I_{ADJ} είναι αμελητέο (<100μA) κι έτσι ο παράγοντας $I_{ADJ} \cdot R_2$ μπορεί να αγνοηθεί. Επομένως χρησιμοποιούμε τις εξής αντιστάσεις:

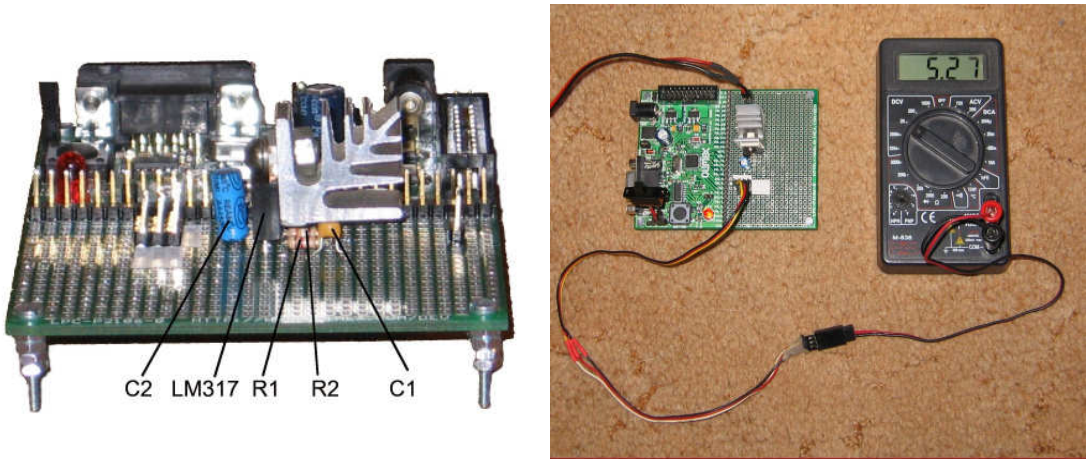
- $R_1 = 120\Omega$
- $R_2 = 390\Omega$

Έτσι προκύπτει η επιθυμητή τάση εξόδου:

$$V_{OUT} = 1{,}25 \cdot \left(1 + \frac{390}{120}\right) = 5{,}3V$$

Οι πυκνωτές που υπάρχουν στην κατασκευή βοηθούν στην εξομάλυνση της τάσης. Αν σαν τάση εισόδου χρησιμοποιούσαμε μόνο μπαταρίες, οι πυκνωτές θα μπορούσαν να παραληφθούν. Ο λόγος είναι ότι η μπαταρία έχει απόλυτα σταθερή τάση η οποία δε χρειάζεται εξομάλυνση. Επειδή όμως τα εργαστηριακά πειράματα γίνονται και με μετασχηματιστές AC/DC οι οποίοι δεν παράγουν απόλυτα σταθερή τάση, έχουμε προσθέσει τους πυκνωτές.

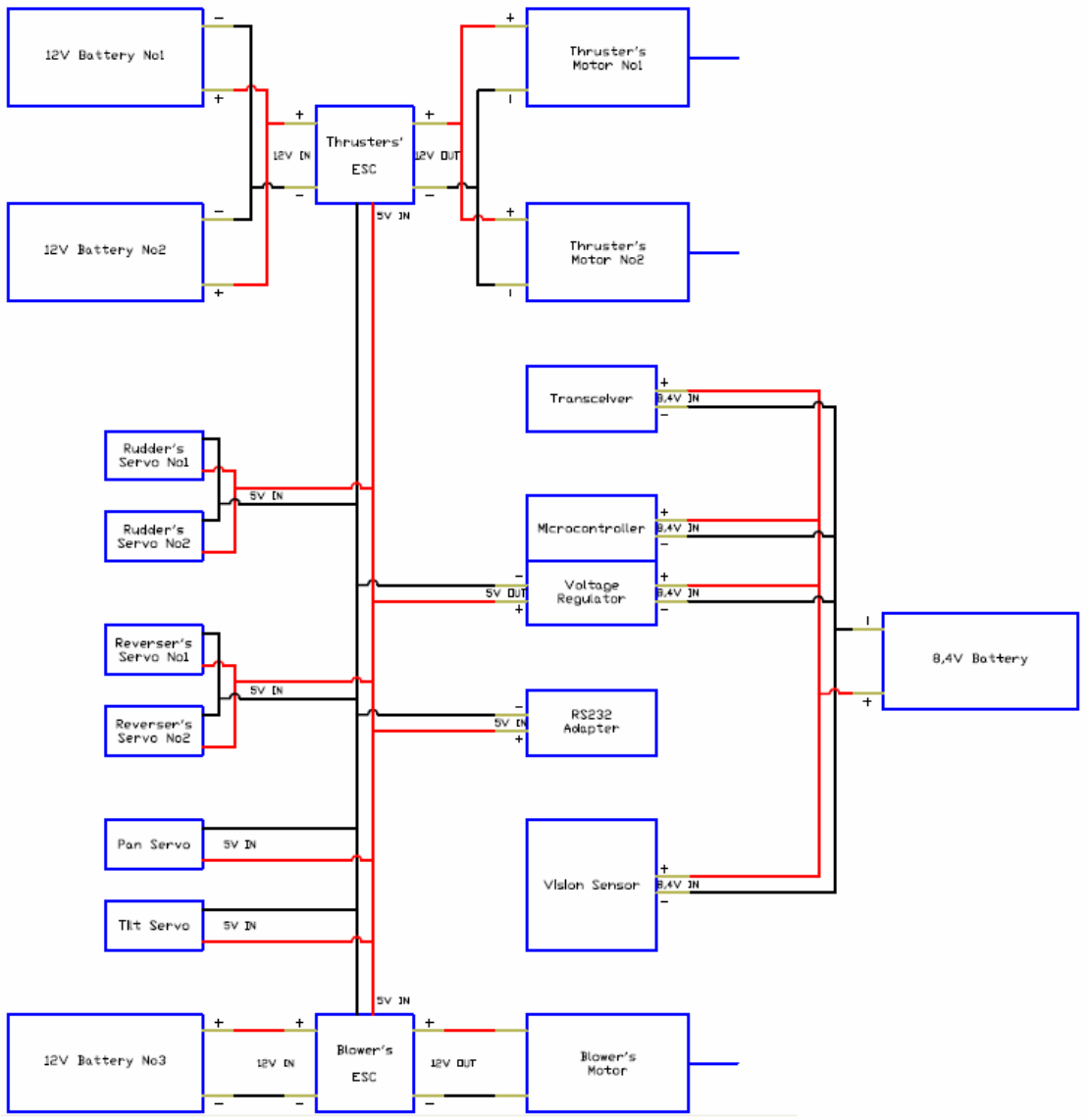
Στη συνέχεια φαίνεται ο ρυθμιστής που τοποθετήθηκέ στην αναπτυξιακή κάρτα του microcontroller καθώς και η ένδειξη του βολτομέτρου (5,27V) το οποίο είναι συνδεδεμένο στην έξοδο του ρυθμιστή (Σχ. 3.2).



Σχήμα 3.2. *Κατασκευή και έλεγχος λειτουργίας του ρυθμιστή τάσης.*

3.3 Ηλεκτρικό δίκτυο πλοίου

Αφού αναφέρθηκαν οι κύριοι καταναλωτές του πλοίου και η τάση λειτουργίας τους, δίνεται διάγραμμα στο οποίο παρουσιάζεται ο τρόπος τροφοδοσίας τους (Σχ.3.3). Τα σήματα ελέγχου αυτών παρουσιάζονται σε άλλη παράγραφο, αναλυτικά για κάθε καταναλωτή.



Σχήμα 3.3. *Διάγραμμα τροφοδοσίας κύριων καταναλωτών πλοίου.*

Στο διάγραμμα παρουσιάζονται δύο ειδών ρεύματα:

- Ισχύος (έως 60A)
- Ασθενή (έως 1,5A)

Οι μπαταρίες των 12V συνδέονται στις καταναλώσεις ισχύος, ενώ η μπαταρία των 8,4V στις ασθενείς καταναλώσεις. Παρακάτω περιγράφεται συνοπτικά ο τρόπος συνδεσμολογίας των καταναλωτών.

Οι δύο μπαταρίες των 12V (No1 και No2) συνδέονται παράλληλα και δίνουν ηλεκτρική ενέργεια στον ηλεκτρονικό ρυθμιστή στροφών των κινητήρων προώσεως (*thruster's ESC*). Ο ESC επεξεργάζεται τη συνεχή τάση και παράγει τετραγωνικό παλμό ισχύος για την οδήγηση των κινητήρων οι οποίοι με τη σειρά τους συνδέονται και αυτοί παράλληλα. Όπως φαίνεται στο διάγραμμα, ο ESC τροφοδοτείται και με τάση 5V. Αυτή η τάση τροφοδοτεί τα ηλεκτρονικά ισχύος του ESC που δημιουργούν και ρυθμίζουν τον τετραγωνικό παλμό ο οποίος είναι απαραίτητος για τον έλεγχο των στροφών των κινητήρων.

Η αντίστοιχη διάταξη υπάρχει και στον κινητήρα του ανεμιστήρα ανύψωσης (*blower*). Εκεί απαιτείται λιγότερη ισχύς άρα η ηλεκτρική ενέργεια δίνεται από μία μπαταρία 12V (No3).

Η μπαταρία 8,4V παρέχει ηλεκτρική ενέργεια στους ασθενείς καταναλωτές:

- Πομποδέκτης.
- Μικροελεγκτής.
- Ρυθμιστής τάσης.
- Αισθητήρας εικόνας.

Ο ρυθμιστής τάσης, όπως έχει αναφερθεί, δίνει τάση 5,3V σε όλους τους σερβομηχανισμούς, τους ηλεκτρονικούς ρυθμιστές στροφών και τον προσαρμογέα σειριακής θύρας.

4. ΗΛΕΚΤΡΟΝΙΚΑ ΜΕΣΑ

4.1 Επιλογή ηλεκτρονικού εξοπλισμού

Το ζητούμενο της παρούσας εργασίας, είναι το σύστημα να διαθέτει χαρακτηριστικά αυτομάτου ελέγχου, τηλεχειρισμού και τηλεμετρίας. Τα παραπάνω χαρακτηριστικά υλοποιούνται με τη βοήθεια ηλεκτρονικού εξοπλισμού ο οποίος χωρίζεται σε δύο επιμέρους συστήματα:

- Σταθμός ξηράς.
- Πλοίο.

Ο εξοπλισμός ο οποίος χρησιμοποιείται επιλέγεται με κύριο στόχο να καλύπτει τις ανάγκες της παρούσας διπλωματικής. Ωστόσο γίνεται η πρόβλεψη για τη δυνατότητα μελλοντικής αναβάθμισης του όλου συστήματος. Έτσι, τα ηλεκτρονικά μέσα που θα χρησιμοποιηθούν πρέπει να πληρούν τα παρακάτω κριτήρια:

- Διαστάσεις.
- Βάρος.
- Ιδιαίτερα χαρακτηριστικά.
- Δυνατότητα αναβάθμισης.
- Αξιοπιστία.
- Κόστος.

Ο ηλεκτρονικός εξοπλισμός του πλοίου αποτελείται από τα εξής:

- Μικροελεγκτής.
- Αισθητήρας εικόνας.
- Πομποδέκτης.
- Σερβομηχανισμοί.
- Ηλεκτρονικοί ρυθμιστές στροφών.

Ο ηλεκτρονικός εξοπλισμός του σταθμού ξηράς αποτελείται από τα εξής:

- Ηλεκτρονικός Υπολογιστής.
- Πομποδέκτης.

Σε αυτό το κεφάλαιο, θα ασχοληθούμε με τα σήματα ελέγχου και την αντίστοιχη συνδεσμολογία και όχι με την τροφοδοσία του εξοπλισμού. Το θέμα της τροφοδοσίας αναλύθηκε στο προηγούμενο κεφάλαιο.

Στις επόμενες παραγράφους περιγράφεται ολόκληρος ο ηλεκτρονικός εξοπλισμός από πλευράς υλικού (*hardware*) και όχι λογισμικού (*software*). Το λογισμικό θα μας απασχολήσει σε επόμενα κεφάλαια.

4.2 Επικοινωνία συσκευών συστήματος

Το κύριο δεδομένο κατά το σχεδιασμό του συστήματος το οποίο είναι και κριτήριο επιλογής του υλικού, είναι ο τρόπος επικοινωνίας μεταξύ των ηλεκτρονικών εξαρτημάτων τα οποία απαρτίζουν το συνολικό σύστημα του πλοίου. Έτσι στον παρακάτω πίνακα δίνονται τα χρησιμοποιούμενα υλικά και ο τρόπος επικοινωνίας τους (Πνκς 4.1).

Πίνακας 4.1
Αντιστοιχία συσκευών - σημάτων.

Hardware	**Signal**
Actuators	
Rudder Servos	PWM
Reverser Servos	PWM
Pan Servo	PWM
Tilt Servo	PWM
Thruster ESC	PWM
Blower ESC	PWM
Sensors	
Vision Sensor	UART
Communication	
Transceiver	UART

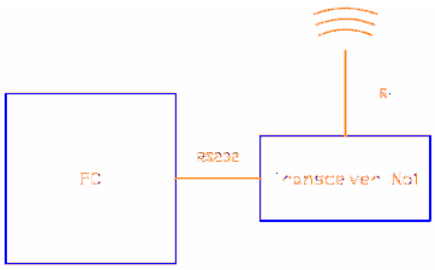
Όπως προκύπτει από τον πίνακα υπάρχει ανάγκη για 6 θύρες διαμόρφωσης εύρους παλμών (*Pulse Width Modulation - PWM*) και 2 σειριακές θύρες (*Universal Asynchronous Receiver Transmitter - UART*). Ο μικροελεγκτής έχει τη δυνατότητα να καλύψει ταυτόχρονα 2 UART και 2 PWM. Επομένως 4 PWM πρέπει να ελεγχθούν από σύστημα εκτός του κύριου μικροελεγκτή. Τη λύση στο πρόβλημα δίνει η κάρτα του αισθητήρα εικόνας η οποία διαθέτει 5 εξόδους PWM.

Άλλο ένα πρόβλημα που ζητά επίλυση είναι ότι η αναπτυξιακή κάρτα του μικροελεγκτή είναι εφοδιασμένη μόνο με έναν προσαρμογέα σειριακής θύρας (UART0). Η UART1 χρειάζεται συγκεκριμένη διαμόρφωση από TTL (*Transistor – Transistor Logic*) σε RS232 (*Recommended Standard 232*). Έτσι κατασκευάζεται ένας εξωτερικός προσαρμογέας σειριακής θύρας (*RS232 Adapter*).

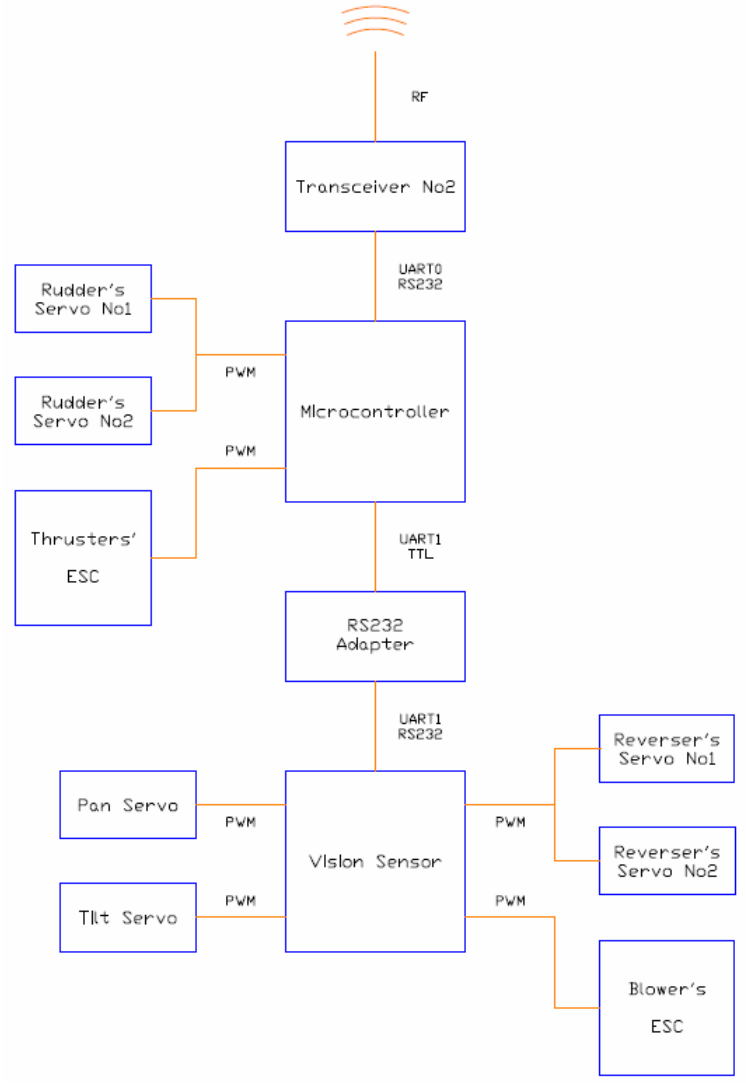
Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τις ανωτέρω έννοιες υπάρχουν στη βιβλιογραφία (Gadre, 2001 – Martin, 2005).*

Τέλος, πέραν του σκάφους υπάρχει και σταθμός ξηράς. Ο ρόλος του σταθμού ξηράς είναι η ασύρματη λήψη δεδομένων των αισθητήρων του σκάφους καθώς επίσης και ο τηλεχειρισμός του συστήματος. Η διάταξη του σταθμού ξηράς αποτελείται από έναν πομποδέκτη συνδεδεμένο μέσω σειριακής θύρας με έναν Η/Υ. Οι πομποδέκτες επικοινωνούν μεταξύ τους με ραδιοκύματα (*Radio Frequency - RF*).

Στη συνέχεια παρουσιάζονται ποιοτικά οι συνδεσμολογίες του σταθμού ξηράς (Σχ. 4.1) και του πλοίου (Σχ. 4.2).



Σχήμα 4.1. *Διάγραμμα συνδεσμολογίας συστήματος σταθμού ξηράς.*



Σχήμα 4.2. *Διάγραμμα συνδεσμολογίας συστήματος πλοίου.*

4.3 Μικροελεγκτής

Βασικό είναι να περιγράψουμε τι είναι ο Μικροελεγκτής. Έτσι, έχουμε τους ακόλουθους όρους:

- **Μικροεπεξεργαστής (*Microprocessor*):** Είναι μια ΚΜΕ (*CPU*) ενσωματωμένη στο ίδιο ολοκληρωμένο κύκλωμα.
- **Μικροϋπολογιστής (*Microcomputer*):** Ένας Μικροεπεξεργαστής μαζί με τα σχετικά κυκλώματα υποστήριξης, διάφορα περιφερειακά εξαρτήματα και κάποιες μονάδες μνήμης (για την αποθήκευση προγραμμάτων όσο και δεδομένων), μπορούν να συνδυαστούν κατάλληλα με σκοπό να σχηματίσουν ένα μικρό υπολογιστικό σύστημα, κυρίως για εφαρμογές ελέγχου και συλλογής δεδομένων.

Σύμφωνα με τους ανωτέρω ορισμούς, **Μικροελεγκτής (*Microcontroller*)** είναι ένα πλήρες σύστημα Μικροϋπολογιστή ενσωματωμένο στο ίδιο ολοκληρωμένο κύκλωμα.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τις ανωτέρω έννοιες υπάρχουν στη βιβλιογραφία (Gadre, 2001 – Martin, 2005).*

Ο μικροελεγκτής που επιλέχθηκε ανήκει στην οικογένεια ARM7 και συγκεκριμένα είναι ο LPC2106 της εταιρείας Philips. Είναι ενσωματωμένος στην αναπτυξιακή κάρτα της Olimex (Σχ. 4.3), ενώ τα χαρακτηριστικά του παρουσιάζονται στο Παράρτημα ΣΤ. Στην παρούσα παράγραφο αναφέρονται τα χαρακτηριστικά εκείνα που χρησιμοποιούνται στην εφαρμογή της συγκεκριμένης εργασίας.

Σχήμα 4.3. *Ο μικροελεγκτής ενσωματωμένος στην αναπτυξιακή κάρτα.*

Ο μικροελεγκτής είναι ο «εγκέφαλος» του πλοίου. Αυτός ελέγχει όλα τα ηλεκτρονικά συστήματα που βρίσκονται στο σκάφος. Όπως αναφέρθηκε παραπάνω δε μπορεί να ελέγξει απευθείας πάνω από 2 θύρες PWM ενώ λειτουργούν ταυτόχρονα και οι δύο σειριακές θύρες. Επομένως, επιλέγεται να ελέγξει τους 2 από τους 6 μηχανισμούς με τη σημαντικότερη λειτουργία. Έτσι, ελέγχει τους σερβομηχανισμούς του πηδαλίου και το ρυθμιστή στροφών των κινητήρων πρόωσης. Οι υπόλοιποι 4 μηχανισμοί συνδέονται στις PWM εξόδους της κάρτας του αισθητήρα εικόνας.

Στη μία σειριακή θύρα του μικροελεγκτή (UART0) συνδέεται ο πομποδέκτης και στην άλλη (UART1) ο αισθητήρας εικόνας. Τα δεδομένα που μπορεί να επεξεργαστεί ο μικροελεγκτής και να τα στείλει ασύρματα στο σταθμό ξηράς προέρχονται από τον αισθητήρα εικόνας και από τους μηχανισμούς κίνησης (αναφορά κατάστασης λειτουργίας). Έτσι, η πληροφορίες που σχετίζονται με την εικόνα και τους μηχανισμούς κίνησης που βρίσκονται συνδεδεμένοι στην κάρτα του αισθητήρα, περνούν σειριακά μέσω της UART1 στον μικροελεγκτή. Αυτές οι πληροφορίες μαζί με την κατάσταση λειτουργίας των μηχανισμών κίνησης που βρίσκονται συνδεδεμένοι στις δύο θύρες PWM του μικροελεγκτή, επεξεργάζονται και οδηγούνται στη UART0. Εκεί, ο πομποδέκτης αναλαμβάνει να στείλει ασύρματα στο σταθμό ξηράς τα δεδομένα που λαμβάνει από τη σειριακή του είσοδο.

Στη συνέχεια περιγράφεται η συνδεσμολογία των αγωγών που μεταφέρουν σήματα από και προς τον μικροελεγκτή.

4.3.1 Σειριακή θύρα UART0

Η αναπτυξιακή κάρτα της Olimex διαθέτει ενσωματωμένο προσαρμογέα σειριακής θύρας για την UART0. Έτσι, η συγκεκριμένη θύρα λειτουργεί κατά το πρότυπο RS232. Αυτό σημαίνει ότι τα σήματα εισόδου και εξόδου μπορούν να ταξιδέψουν μέσα σε καλώδια έως 15m μήκος. Το βύσμα της UART0 είναι τύπου DB-9S θηλυκό (Σχ. 4.9). Από τα 9 pin του βύσματος χρησιμοποιούνται μόνο τα 3. Στη συνέχεια επεξηγούνται τα σήματα που αντιστοιχούν στα διάφορα pins (Πνκς 4.2)

Πίνακας 4.2
Αντιστοιχία σημάτων στα pin του βύσματος.

Pin No	Σήμα	Επεξήγηση
2	TXD0	Σειριακή έξοδος
3	RXD0	Σειριακή είσοδος
5	GND	Γείωση

4.3.2 Σειριακή θύρα UART1

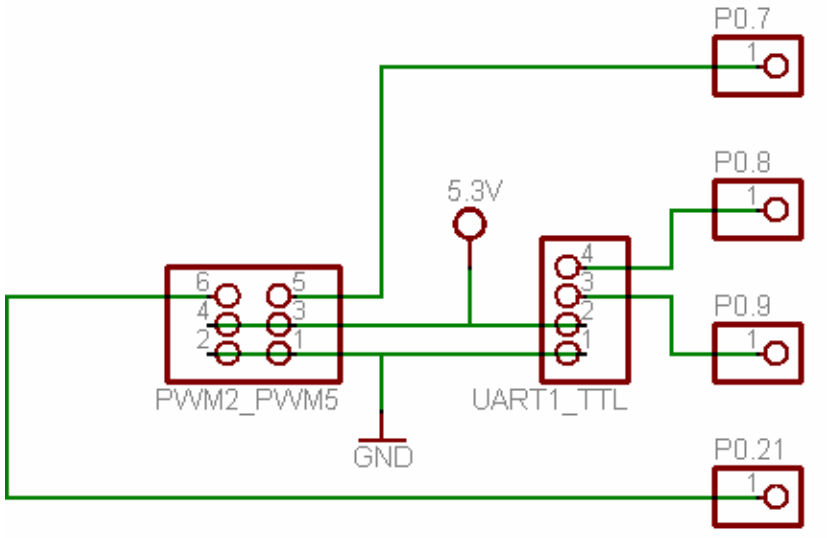
Επειδή η αναπτυξιακή κάρτα της Olimex δε διαθέτει ενσωματωμένο προσαρμογέα σειριακής θύρας για την UART1 η έξοδος λειτουργεί σε επίπεδο TTL. Έτσι, τα σήματα εισόδου και εξόδου μπορούν να ταξιδέψουν μέσα σε καλώδια μήκους το πολύ 30cm. Επειδή οι απαιτήσεις της εφαρμογής μας ζητούν επικοινωνία μέσω της σειριακής θύρας UART1 σε απόσταση πάνω από 30cm, θα κατασκευαστεί ένας προσαρμογέας σειριακής θύρας. Ο προσαρμογέας θα περιγραφεί σε επόμενη παράγραφο. Παρακάτω παρουσιάζονται σχηματικά τα pin εξόδου από την αναπτυξιακή κάρτα (Σχ.4.4) και επεξηγούνται τα σήματα (Πνκς 4.3)

4.3.3 Θύρες PWM

Ο LPC2106 έχει τη δυνατότητα να χειριστεί έως 6 θύρες PWM. Αυτός ο αριθμός εξαρτάται από τη ρύθμιση του μικροελεγκτή και πιο συγκεκριμένα από τι είδους θύρες χρησιμοποιούνται. Ο λόγος που συμβαίνει αυτό είναι ότι η αρχιτεκτονική του συγκεκριμένου μικροελεγκτή δίνει τη δυνατότητα για εναλλακτικές χρήσεις των εξόδων/εισόδων του. Έτσι, όπως έχουμε αναφέρει και νωρίτερα, ο LPC2106 ρυθμίζεται ώστε να μπορεί να ελέγξει και τις 2 διαθέσιμες σειριακές (UART0,1) καθώς επίσης και 2 από τις 6 θύρες PWM (PWM2,5). Η θύρα PWM2 οδηγεί τους σερβομηχανισμούς του πηδαλίου, ενώ η θύρα PWM5 οδηγεί τον ηλεκτρονικό ρυθμιστή στροφών των κινητήρων πρόωσης. Παρακάτω παρουσιάζονται σχηματικά τα pin εξόδου από την αναπτυξιακή κάρτα (Σχ. 4.4) και επεξηγούνται τα σήματα (Πνκς 4.3).

Πίνακας 4.3
Αντιστοιχία σημάτων στα χρησιμοποιούμενα pin.

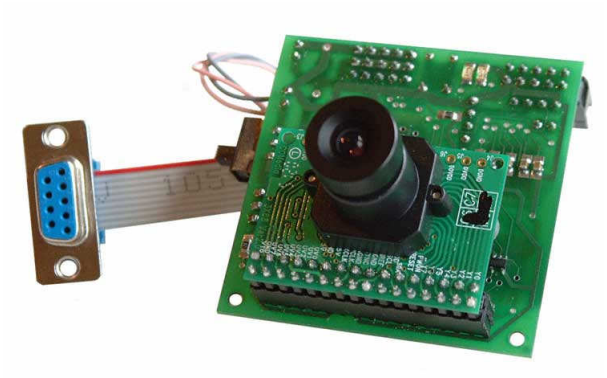
Pin No	Σήμα	Επεξήγηση
P0.7	PWM2	Έλεγχος μηχανισμού πηδαλίου
P0.8	TXD1	Σειριακή έξοδος
P0.9	RXD1	Σειριακή είσοδος
P0.21	PWM5	Έλεγχος ρυθμιστή στροφών κινητήρων πρόωσης
-	GND	Γείωση
-	5,3V	Τάση λειτουργίας σερβομηχανισμών και προσαρμογέα σειριακής θύρας



Σχήμα 4.4. *Απεικόνιση χρησιμοποιούμενων pin στην αναπτυξιακή κάρτα του μικροελεγκτή.*

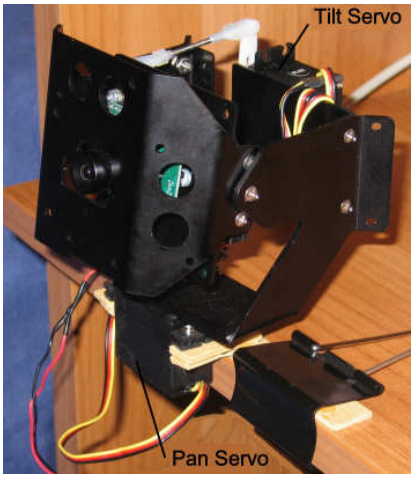
4.4 Αισθητήρας εικόνας

Ο αισθητήρας εικόνας που χρησιμοποιείται είναι ο CMUcam2 (Σχ. 4.5). Αποτελείται από μία συσκευή συλλογής εικόνων (*camera*) προσαρμοσμένη σε ένα ολοκληρωμένο κύκλωμα το οποίο επιτρέπει να εξαχθούν δεδομένα υψηλού επιπέδου από τη συνεχή ροή εικόνων της κάμερας. Αυτά τα δεδομένα οδηγούνται σε έναν μικροελεγκτή (SX52) ο οποίος βρίσκεται στην κάρτα του αισθητήρα. Η κάρτα επικοινωνεί με εξωτερικές συσκευές μέσω σειριακής θύρας (UART). Όμοια με τον μικροελεγκτή, η κάρτα της CMUcam2 διαθέτει ενσωματωμένο προσαρμογέα σειριακής θύρας για την UART, άρα η θύρα λειτουργεί κατά το πρότυπο RS232. Το βύσμα της UART είναι τύπου DB-9S θηλυκό.



Σχήμα 4.5. *Ο αισθητήρας εικόνας CMUcam2.*

Προκειμένου να αυξηθεί το οπτικό πεδίο του αισθητήρα, τοποθετείται σε βάση η οποία έχει τη δυνατότητα να περιστρέφεται σε δύο άξονες, κάθετος (*pan*) και οριζόντιος (*tilt*) (Σχ. 4.6). Η βάση περιστρέφεται με τη βοήθεια 2 σερβομηχανισμών οι οποίοι ελέγχονται από τις θύρες PWM της κάρτας του αισθητήρα.



Σχήμα 4.6. *Ο αισθητήρας εικόνας τοποθετημένος στη βάση του με τους σερβομηχανισμούς περιστροφής.*

Όπως έχει αναφερθεί ήδη, ο μικροελεγκτής που είναι ενσωματωμένος στην κάρτα του αισθητήρα (SX52) έχει τη δυνατότητα να ελέγξει 5 θύρες PWM . Στην εφαρμογή μας χρησιμοποιούμε τις 4 θύρες. Οι 2 εξυπηρετούν τους σερβομηχανισμούς που περιστρέφουν τον αισθητήρα, ενώ οι άλλοι 2 εξυπηρετούν δευτερεύουσες λειτουργίες του πλοίου (*blower ESC, reverser's servos*).

Επειδή ο αισθητήρας περιστρέφεται και μάλιστα σε δύο άξονες, καλό είναι να μην υπάρχουν πολλά εξωτερικά καλώδια συνδεδεμένα στην κάρτα, τα οποία υπάρχει κίνδυνος να κοπούν κατά τη διάρκεια της κίνησης του αισθητήρα. Η λύση σε αυτό το πρόβλημα δίνεται από καλωδιοταινία, με τη βοήθεια της οποίας συγκεντρώνονται όλοι οι αγωγοί μαζί. Στη συνέχεια γίνεται μια μελέτη διαστασιολόγισης αυτής της καλωδιοταινίας ώστε να γνωρίζουμε τον απαιτούμενο αριθμό αγωγών που θα την απαρτίζουν (Πνκς 4.4).

Πίνακας 4.4

Αναζήτηση πλήθους αγωγών συνδεδεμένων στον αισθητήρα.

	Σήμα	**Πλήθος Αγωγών**
UART	TX	1
	RX	1
	GND_{232}	1
Main Power	8,4V	1
	GND_{PWR}	1
Servo Signal	SV2	1
	SV3	1
Servo Power	5,3V	1
	GND_{SV}	1
ΣΥΝΟΛΟ		**9**

Ο αριθμός 9 είναι ιδιαίτερα βολικός γιατί όπως έχουμε αναφέρει, το βύσμα της σειριακής θύρας (το οποίο ήδη φέρει ο αισθητήρας) είναι τύπου DB-9S θηλυκό (Σχ. 4.9). Άρα χρησιμοποιώντας αυτό το βύσμα σε συνδυασμό με την καλωδιοταινία έχουμε πλήρη τροφοδοσία και έλεγχο των ζητουμένων από τον αισθητήρα. Παρακάτω επεξηγούνται και αντιστοιχίζονται τα σήματα στα 9 pin (Πνκς 4.5)

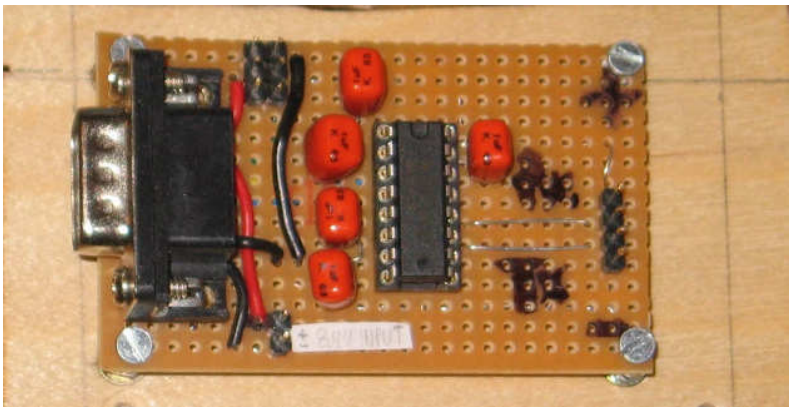
Πίνακας 4.5
Αντιστοιχία σημάτων στα pin του βύσματος της κάμερας.

Pin No	**Σήμα**	**Επεξήγηση**
1	SV3	Έλεγχος μηχανισμού αναπόδισης
2	TX	Σειριακή έξοδος
3	RX	Σειριακή είσοδος
4	SV2	Έλεγχος ηλεκτρονικού ρυθμιστή στροφών του ανεμιστήρα ανύψωσης
5	GND_{232}	Γείωση σειριακής θύρας
6	8,4V	Τάση λειτουργίας αισθητήρα
7	5,3V	Τάση λειτουργίας μηχανισμών
8	GND_{PWR}	Γείωση αισθητήρα
9	GND_{SV}	Γείωση μηχανισμών

Άλλα χαρακτηριστικά καθώς και ο τρόπος λειτουργίας του αισθητήρα εικόνας υπάρχουν στο Παράρτημα ΣΤ και στα manual που περιέχονται στο επισυναπτόμενο CD.

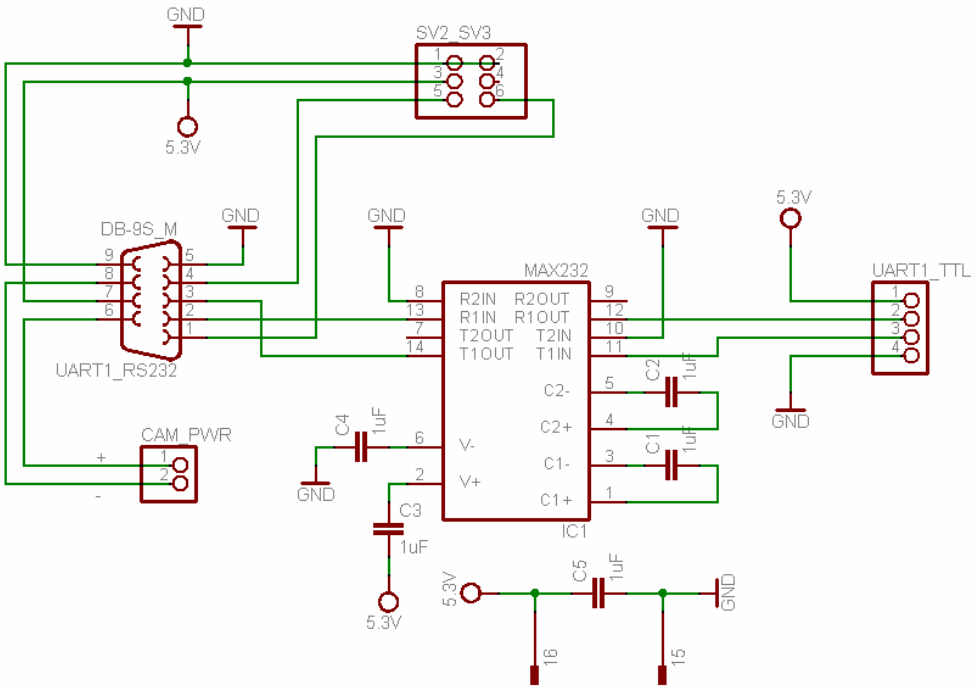
4.5 Προσαρμογέας σειριακής θύρας

Ο σύνδεσμος μεταξύ UART1 του μικροελεγκτή και UART του αισθητήρα εικόνας υλοποιείται με την κατασκευή ενός προσαρμογέα σειριακής θύρας (Σχ. 4.7).



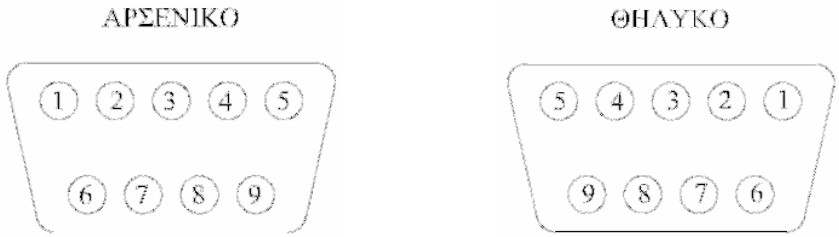
Σχήμα 4.7. *Ο προσαρμογέας σειριακής θύρας.*

Ο κύριος ρόλος του προσαρμογέα είναι να πάρει σαν είσοδο το σήμα TTL που δίνει ο μικροελεγκτής για τη UART1 και να δημιουργήσει το ζητούμενο σήμα RS232. Με αυτή τη διαμόρφωση σήματος γίνεται δυνατή η επικοινωνία του μικροελεγκτή με τον αισθητήρα. Ο δευτερεύον ρόλος του προσαρμογέα είναι να τροφοδοτήσει τον αισθητήρα με ενέργεια αλλά και να δεχθεί τα σήματα ελέγχου των μηχανισμών (*blower ESC, reverser's servos*), τα οποία όπως έχουμε αναφέρει ελέγχονται μέσω της κάρτας του αισθητήρα. Στη συνέχεια παρουσιάζεται το σχέδιο της κατασκευής του προσαρμογέα (Σχ. 4.8).



Σχήμα 4.8. *Το σχέδιο του προσαρμογέα.*

Το βύσμα από την πλευρά του αισθητήρα είναι DB-9S αρσενικό (Σχ. 4.9). Η αντιστοιχία σημάτων στα 9 pin είναι όπως παρουσιάζεται στον πίνακα 4.5.



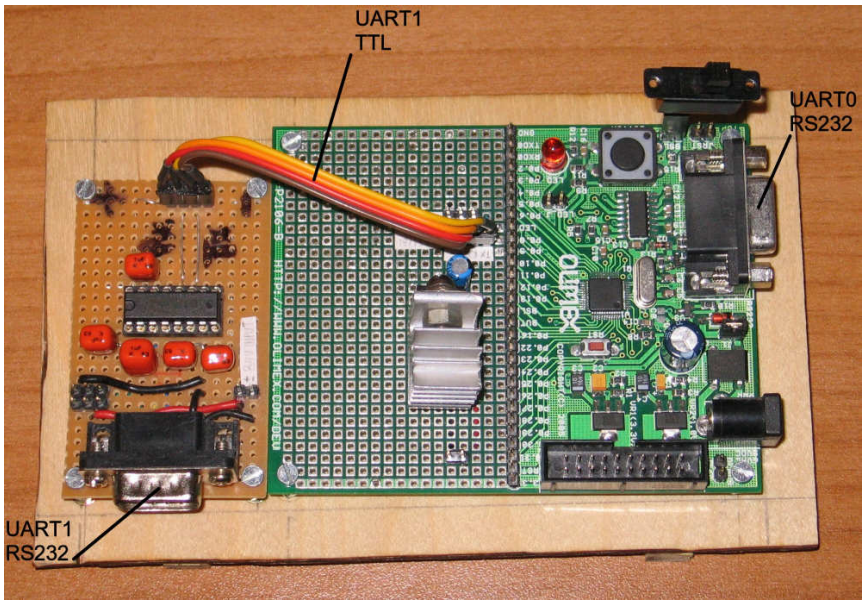
Σχήμα 4.9. *Απεικόνιση των pin στο αρσενικό και θηλυκό βύσμα τύπου DB-9S.*

Στην πλευρά της επικοινωνίας με τον μικροελεγκτή υπάρχει σειρά pin (UART1_TTL) όπου με τη βοήθεια καλωδιοταινίας τεσσάρων αγωγών επιτυγχάνεται η σύνδεση του προσαρμογέα με τον μικροελεγκτή. Στη συνέχεια δίνονται τα σήματα με την αντιστοιχία τους στη σειρά pin καθώς και η εξήγησή τους (Πνκς 4.6).

Πίνακας 4.6
Αντιστοιχία σημάτων στη σειρά pin UART1_TTL.

Pin No	Σήμα	Επεξήγηση
1	5,3V	Τάση λειτουργίας σερβομηχανισμών και προσαρμογέα σειριακής θύρας
2	RXD0	Σειριακή είσοδος
3	TXD0	Σειριακή έξοδος
4	GND	Γείωση

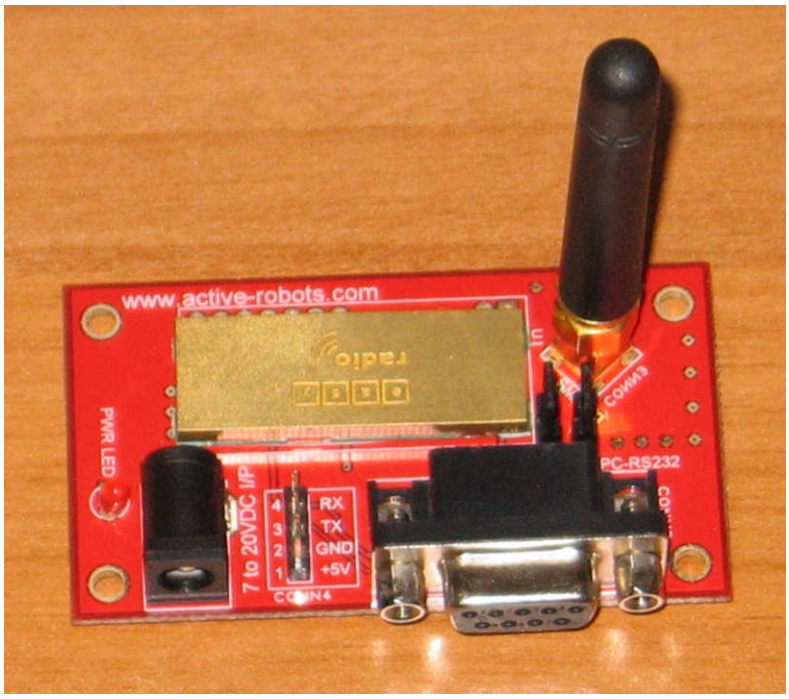
Τέλος, παρουσιάζεται ο μικροελεγκτής συνδεδεμένος μέσω της καλωδιοταινίας τεσσάρων αγωγών με τον προσαρμογέα της σειριακής θύρας UART1 (Σχ. 4.10).



Σχήμα 4.10 *Μικροελεγκτής και προσαρμογέας.*

4.6 Σύστημα πομποδεκτών

Οι πομποδέκτες που χρησιμοποιούνται είναι δύο όμοιες ηλεκτρονικές συσκευές (ER400TRS-02) της εταιρείας easy radio (Σχ. 4.11).



Σχήμα 4.11. *Ο πομποδέκτης.*

Κάθε μία από αυτές διαθέτει μία σειριακή θύρα (UART) και μία κεραία εκπομπής ραδιοκυμάτων (RF). Ο πομποδέκτης που βρίσκεται στο σκάφος συνδέεται στη σειριακή θύρα UART0 του μικροελεγκτή. Η λειτουργία του είναι:

- Δέχεται ασύρματα δεδομένα από το σταθμό ξηράς και τα μεταβιβάζει μέσω της σειριακής θύρας στο μικροελεγκτή.
- Δέχεται δεδομένα σειριακά από το μικροελεγκτή και τα μεταβιβάζει ασύρματα μέσω ραδιοκυμάτων στο σταθμό ξηράς.

Αντίστοιχα η λειτουργία του πομποδέκτη που βρίσκεται στο σταθμό ξηράς είναι:

- Δέχεται δεδομένα σειριακά από τον Η/Υ και τα μεταβιβάζει ασύρματα μέσω ραδιοκυμάτων στο πλοίο.
- Δέχεται ασύρματα δεδομένα από το πλοίο και τα μεταβιβάζει μέσω της σειριακής θύρας στον Η/Υ.

Τα χαρακτηριστικά των πομποδεκτών δίνονται αναλυτικά στο Παράρτημα ΣΤ.

4.8 Ολοκλήρωση συστήματος

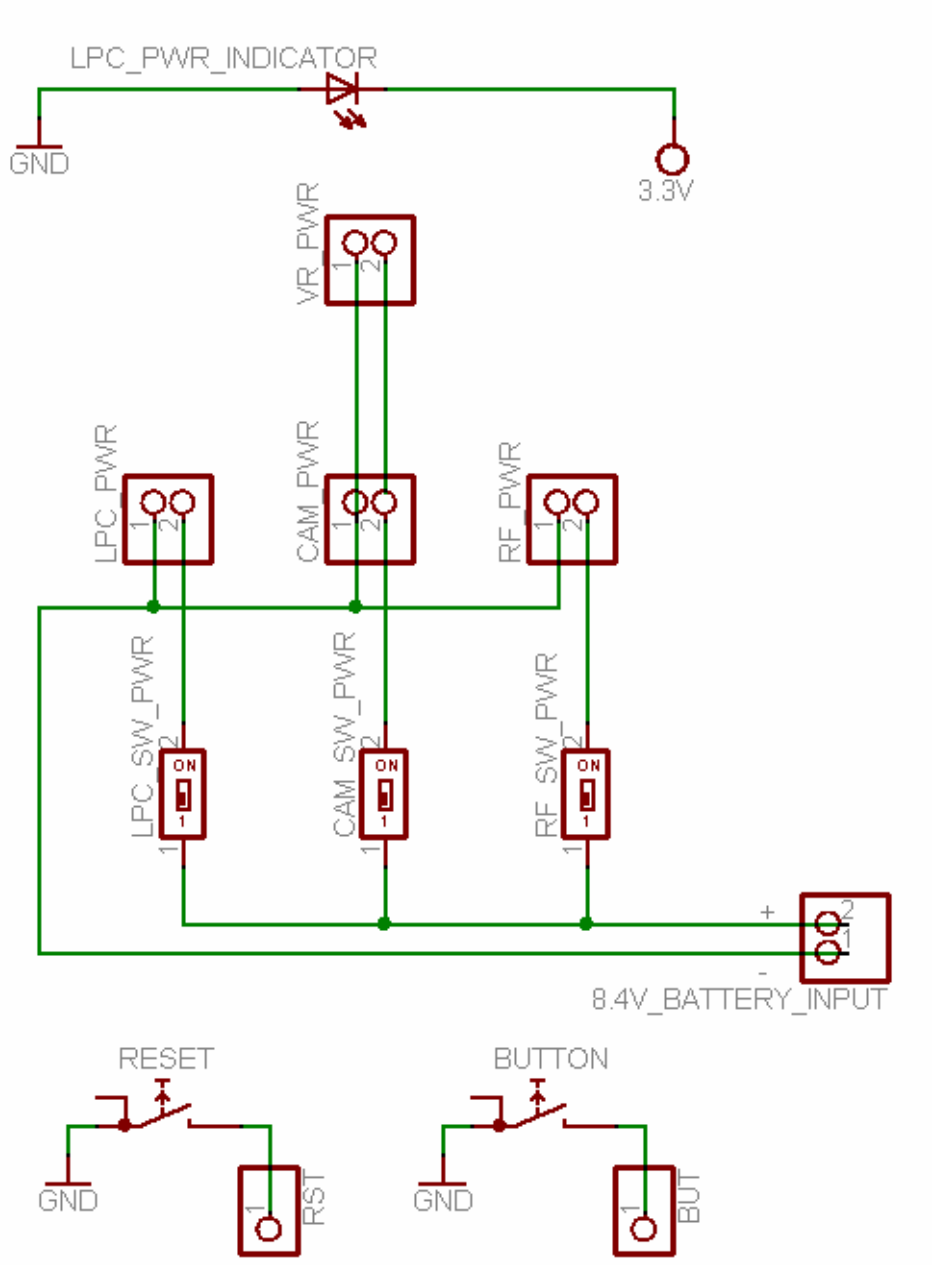
Στόχος της παρούσας παραγράφου είναι να συνδυάσει όλα τα παραπάνω και να δώσει μια συνολική εικόνα του συστήματος από πλευράς hardware. Πρέπει να αναφερθεί ότι κατά τη διάρκεια πειραμάτων προέκυψε η ανάγκη για εύκολη συνδεσμολογία του συστήματος εντός ή εκτός του σκάφους. Έτσι αποφασίστηκε να κατασκευαστεί ένα κουτί το οποίο θα περιείχε τον μικροελεγκτή και τον μετατροπέα σειριακής θύρας. Έτσι, διευκολύνεται η μετακίνηση του συστήματος. Το κουτί για να είναι εύχρηστο είναι εφοδιασμένο με διακόπτες, κουμπιά, φωτοδίοδο (*light emitting diode - led*) καθώς και δίαυλο επικοινωνίας με το εξωτερικό περιβάλλον (καλωδιοταινία για εύκολη σύνδεση - αποσύνδεση όλων των αγωγών ταυτόχρονά) (Σχ. 4.12). Στο εσωτερικό του κουτιού τοποθετείται επίσης και η μπαταρία που τροφοδοτεί τους ασθενείς καταναλωτές (8,4V).

Σχήμα 4.12. *Το κουτί «ελέγχου».*

Ο ρόλος των διακοπτών είναι να τροφοδοτούν με ενέργεια όλες τις συσκευές του σκάφους. Τα κουμπιά εκτελούν συγκεκριμένες διεργασίες του μικροελεγκτή. Το led δηλώνει την κατάσταση του μικροελεγκτή (ON ή OFF).

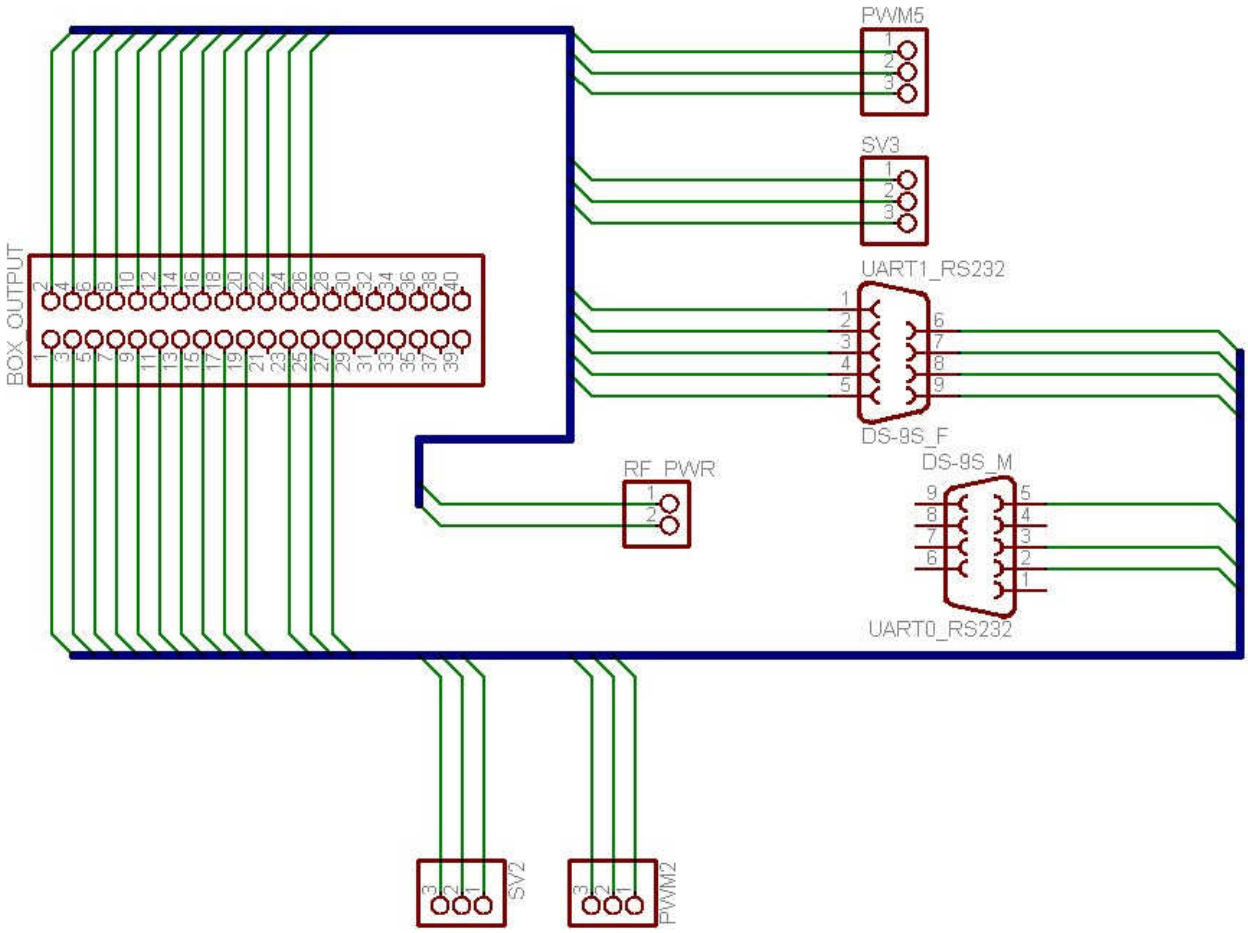
Η καλωδιοταινία έχει 39 αγωγούς. Χρησιμοποιούνται οι 26, ενώ οι υπόλοιποι παραμένουν για πιθανή μελλοντική χρήση.

Η συνδεσμολογία των διακοπτών, των κουμπιών και του led παρουσιάζεται παρακάτω (Σχ.4.13)



Σχήμα 4.13. *Διάγραμμα διακοπτών, κουμπιών και led που βρίσκονται στο κουτί ελέγχου.*

Στη συνέχεια δίνεται το διάγραμμα όλων των καλωδίων που καταλήγουν στην καλωδιοταινία από το εσωτερικό του κουτιού (Σχ. 4.14).



Σχήμα 4.14. *Διάγραμμα συνδεσμολογίας διαύλου.*

Οι παραπάνω συνδεσμολογίες επεξηγούνται αναλυτικά στον πίνακα 4.7. Ο πίνακας αυτός χωρίζεται σε δύο μέρη. Αριστερά περιγράφονται οι λειτουργίες των αγωγών που αντιστοιχούν σε κάθε pin του διαύλου. Δεξιά αντιστοιχίζονται τα pin του κάθε βύσματος στο εσωτερικό του κουτιού με τα pin του διαύλου.

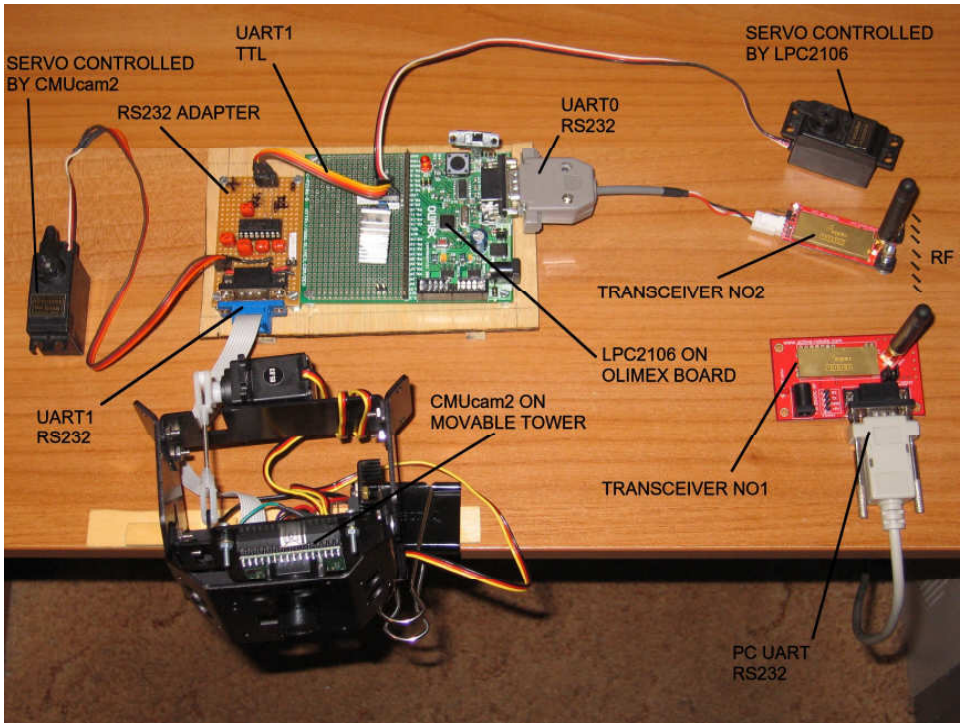
Τέλος, τα καλώδια που φέρει η καλωδιοταινία στην έξοδο του κουτιού ελέγχου, κατά την αντίστροφη διαδικασία, μοιράζονται στις διάφορες ηλεκτρονικές συσκευές.

Πίνακας 4.7.1
Αναλυτική περιγραφή αγωγών.

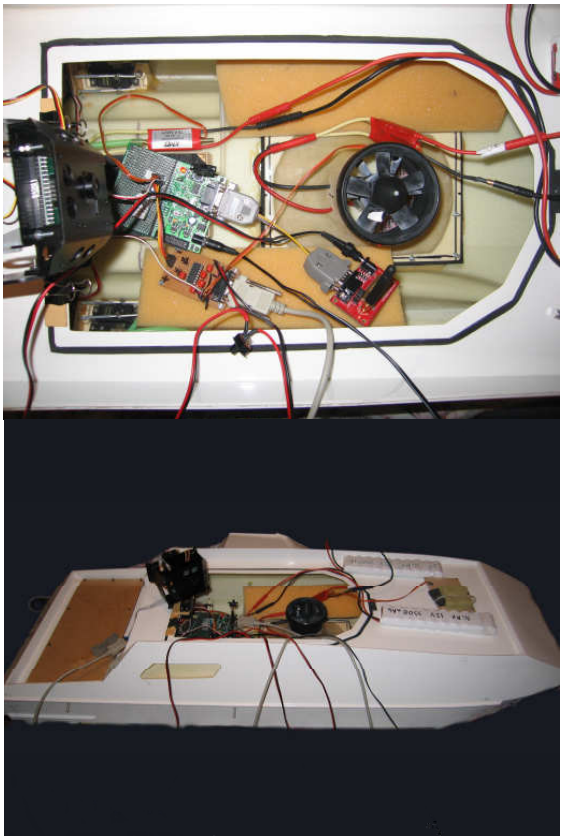
Box output Pin No	Description
1	Rudder's servos GND output
2	Thruster's ESC GND output
3	Rudder's servos 5,3V output
4	Thruster's ESC 5,3V output
5	Rudder's servos PWM output
6	Thruster's ESC PWM output
7	Blower's ESC GND output
8	Reverser's servos GND output
9	Blower's servos 5,3V output
10	Reverser's servos 5,3V output
11	Blower's ESC PWM output
12	Reverser's servos PWM output
13	CMUcam2 main power output 8,4V
14	SV2 input (from CMUcam2)
15	CMUcam2 servo power output 5,3V
16	CMUcam2 UART RXD1
17	CMUcam2 main power GND
18	CMUcam2 UART TXD1
19	CMUcam2 servo power GND
20	SV3 input (from CMUcam2)
21	-
22	CMUcam2 UART GND
23	Transceiver UART GND
24	Transceiver main power GND
25	Transceiver UART TXD0
26	Transceiver main power 8,4V
27	Transceiver UART RXD0

Connector	Connector's Pin No	Box output Pin No
PWM2	1	1
	2	3
	3	5
PWM5	1	2
	2	4
	3	6
SV2	1	7
	2	9
	3	11
SV3	1	8
	2	10
	3	12
UART0_RS232	2	25
	3	27
	5	23
UART1_RS232	1	20
	2	16
	3	18
	4	14
	5	22
	6	13
	7	15
	8	17
	9	19
RF_PWR	1	24
	2	26

Παρακάτω δίνονται μερικές εικόνες της εργαστηριακής εγκατάστασης για πληρέστερη κατανόηση του συστήματος (Σχ. 4.15 – 4.16).



Σχήμα 4.15. *Εργαστηριακή εγκατάσταση δοκιμών χωρίς σκάφος.*



Σχήμα 4.16. *Δοκιμές στο σκάφος.*

5. ΜΑΘΗΜΑΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ

5.1 Εισαγωγή

5.1.1 Περιγραφή προβλήματος

Όπως έχει αναφερθεί στο εισαγωγικό κεφάλαιο σκοπός της παρούσας εργασίας είναι ο δυναμικός εντοπισμός και έλεγχος της πορείας και θέσης σκάφους μέσω της ανάπτυξης ενός συστήματος παρακολούθησης φωτεινού στόχου που βρίσκεται στην ακτή ή σε άλλο προπορευόμενο σκάφος. Επομένως, μελετάται ένα σενάριο αυτοματισμού σύμφωνα με το οποίο θα επιτυγχάνεται το εξής:

Το πλοίο σε σκοτεινό περιβάλλον, θα ακολουθεί μια συγκεκριμένου χρωματικού εύρους φωτεινή πηγή, κινούμενη ή ακίνητη (παρακολούθηση στόχου).

Για να επιτευχθεί αυτό, χρησιμοποιείται δεδομένο hardware το οποίο έχει αναλυθεί σε προηγούμενα κεφάλαια. Σύμφωνα με αυτό, μπορούν να ελεγχθούν τα εξής:

- Φορτίο κινητήρων (*thruster*).
- Φορτίο ανεμιστήρα ανύψωσης (*blower*).
- Θέση ακροφυσίου προωθητήρων (*rudder*).
- Θέση ακροφυσίου αναπόδισης (*reverser*).

Σε πρώτη προσέγγιση, για τον έλεγχο της κίνησης του πλοίου επιστρατεύονται μόνο το φορτίο κινητήρων και η θέση των ακροφυσίων των προωθητήρων. Το φορτίο ανεμιστήρα ανύψωσης καθώς και η θέση του ακροφυσίου αναπόδισης παραμένουν σε προεπιλεγμένες καταστάσεις λειτουργίας. Αυτό συμβαίνει για απλούστευση της μελέτης μας. Ο μόνος αισθητήρας που βρίσκεται στο σκάφος είναι η κάμερα. Έτσι, σύμφωνα με το οπτικό της πεδίο θα δίνονται μέσω του microcontroller κατάλληλες εντολές οι οποίες θα αφορούν thruster και το rudder ώστε να ελέγχεται η πορεία του σκάφους.

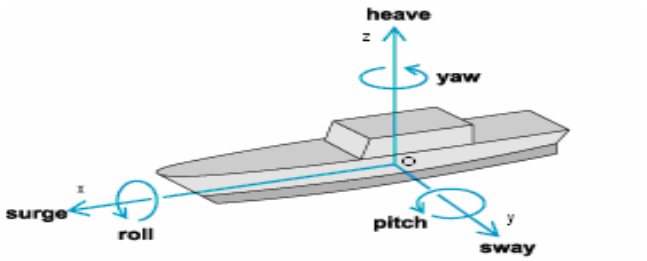
Προκειμένου να διεξαχθεί επιτυχημένα το πείραμα παρακολούθησης στόχου, είναι απαραίτητη η διεξαγωγή προσομοιώσεων σε Η/Υ. Κατά τη διαδικασία αυτή, μας δίνεται η δυνατότητα να ελέγξουμε εικονικά την κίνηση του πλοίου πριν πραγματοποιηθεί το πείραμα του φυσικού μοντέλου σε αληθινό περιβάλλον. Έτσι, διορθώνονται τυχόν προβλήματα χωρίς υλικές απώλειες. Επίσης, η προσομοίωση είναι απαραίτητη για τη δημιουργία του αλγορίθμου ελέγχου ο οποίος θα «κατευθύνει» το πλοίο.

Για να γίνουν οι προσομοιώσεις πρέπει αρχικά να μοντελοποιηθεί μαθηματικά το σύστημα πλοίο – κάμερα. Στη συνέχεια, θα δημιουργηθεί ο αλγόριθμος ελέγχου της κίνησης του πλοίου. Μετά θα γίνει εικονικό πείραμα παρακολούθησης στόχου. Αφού, έχει δημιουργηθεί ένα «εικονικά» αξιόπιστο σύστημα ελέγχου με ανατροφοδότηση (*feedback control*), θα εισαχθεί σε ψηφιακή μορφή στον microcontroller που βρίσκεται ενσωματωμένος στο φυσικό μοντέλο. Στη συνέχεια θα διεξαχθεί το πραγματικό πείραμα παρακολούθησης στόχου και θα ακολουθήσει σύγκριση αποτελεσμάτων εικονικού – πραγματικού πειράματος.

Με αυτό τον τρόπο θα αξιολογήσουμε τη μαθηματική μοντελοποίηση η οποία όπως θα δούμε στη συνέχεια πραγματοποιείται με κάποιες απλοποιήσεις - παραδοχές. Η διαδικασία προσομοιώσεων καθώς και η δημιουργία του αλγορίθμου ελέγχου παρουσιάζονται σε επόμενα κεφάλαια ενώ η μεθοδολογία μαθηματικής μοντελοποίησης περιγράφεται στις επόμενες παραγράφους του παρόντος κεφαλαίου.

5.1.2 Γενική μορφή εξισώσεων κίνησης ελεύθερα επιπλέοντας στερεού σώματος

Οι κινήσεις ενός σκάφους επιφανείας στο νερό μπορούν να αναλυθούν σε 6 βαθμούς ελευθερίας αν αυτό θεωρηθεί ως απολύτως στερεά (άκαμπτη) δοκός (Σχ.5.1).



Σχήμα 5.1. *Βαθμοί ελευθερίας σκάφους επιφανείας.*

Η δυναμική απόκριση ενός σκάφους επιφανείας μοντελοποιείται μαθηματικά με ένα σύστημα διαφορικών ως προς το χρόνο εξισώσεων κινήσεως που πηγάζουν από τη θεώρησή του ως απολύτως άκαμπτου στερεού. Οι εν λόγω δ.ε. αφορούν το ρυθμό μεταβολής της ορμής και στροφορμής του πλοίου και δίνονται παρακάτω ως προς σωματόδετο σύστημα αναφοράς Oxyz:

$$\sum \vec{F}_{ext} = \frac{d}{dt}\left\{ \int_{m_S} (\vec{v}_S + \vec{\omega}_S \times \vec{r}) dm \right\} \quad (5.1)$$

$$\sum \vec{M}_{ext} = \frac{d}{dt}\left\{ \int_{m_S} \vec{r} \times (\vec{v}_S + \vec{\omega}_S \times \vec{r}) dm \right\} + m_S \cdot (\vec{v}_S \times \vec{v}_G)$$

Όπου:

- $\sum \vec{F}_{ext}$: Συνισταμένη εξωτερικών δυνάμεων.
- $\sum \vec{M}_{ext}$: Συνισταμένη εξωτερικών ροπών.
- m_S : Μάζα σκάφους.
- $\vec{v}_S$: Γραμμική ταχύτητα σημείου αναφοράς Ο του σωματόδετου συστήματος.
- $\vec{\omega}_S$: Γωνιακή ταχύτητα σκάφους.
- $\vec{v}_G$: Μεταφορική ταχύτητα κέντρου μάζας του σκάφους.
- $\vec{r}$: Διάνυσμα θέσης κάθε στοιχείου μάζας *dm* επί του πλοίου ως προς το Oxyz.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τις εξισώσεις κίνησης επιπλέοντος σώματος υπάρχουν στη βιβλιογραφία (Αθανασούλης, 2005).*

5.1.3 Μοντέλο μειωμένης τάξης

Αναλύοντας τις διανυσματικές διαφορικές εξισώσεις (5.1), προκύπτουν 6 βαθμωτές δ.ε. οι οποίες αντιστοιχούν σε κάθε βαθμό ελευθερίας. Οι 3 αφορούν τις μεταφορικές κινήσεις (*surge, sway, heave*) ενώ οι άλλες 3 τις περιστροφικές κινήσεις του πλοίου (*roll, pitch, yaw*). Οι 6 αυτές σχέσεις είναι μη γραμμικές και πεπλεγμένες.

Επιπλέον το δικό μας σκάφος AIRCAT είναι τύπου SES (*Surface Effect Ship*). Στην πραγματικότητα δεν είναι απολύτως άκαμπτο στερεό πράγμα που σημαίνει ότι οι κινήσεις του στο νερό περιγράφονται από άπειρους β.ε. Άρα οι παραπάνω σχέσεις που αναφέρονται σε συμβατικά πλοία επιφανείας και απολύτως άκαμπτα δεν ανταποκρίνονται πλήρως στις ανάγκες μας.

Επίσης οι ζητούμενες εξισώσεις για τη δική μας περίπτωση πρέπει να συμβαδίζουν με το διαθέσιμο hardware. Δηλαδή, πρέπει να μπορούν να περιγράψουν και να ελέγξουν τη βασική κίνηση του πλοίου μέσω των ενδείξεων του μοναδικού αισθητήρα ο οποίος βρίσκεται στο σκάφος (κάμερα).

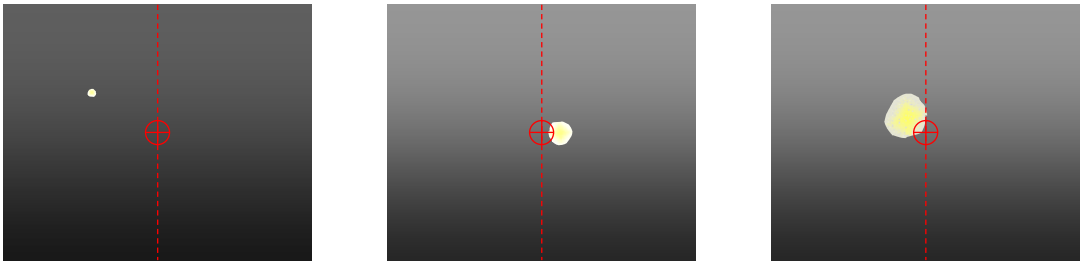
Έτσι, αποφασίζεται η ανάλυση να βασισθεί σε μοντέλο μειωμένης τάξης (*reduced order model*). Με τον όρο αυτό εννοούμε ότι κατά τη μοντελοποίηση του συστήματος θεωρούμε ότι οι κινήσεις του πλοίου μπορούν να περιγραφούν επαρκώς με λιγότερους β.ε. από ότι στην πραγματικότητα. Ο λόγος είναι ότι μια μαθηματική μοντελοποίηση η οποία περιγράφει πεπερασμένο αριθμό β.ε. μπορεί να πραγματοποιηθεί, σε αντίθεση με την περίπτωση των άπειρων β.ε. Δηλαδή η επίλυση συστήματος πεπερασμένου αριθμού δ.ε. είναι εφικτή.

Στην παρούσα ανάλυση γίνονται κάποιες παραδοχές οι οποίες απλοποιούν τη διαδικασία μοντελοποίησης. Έτσι, στο δικό μας πρόβλημα μελετάμε την κίνηση του πλοίου στο επίπεδο xy και μας ενδιαφέρουν 2 β.ε. (αντί για άπειρους). Οι 2 αυτοί β.ε. είναι η μεταφορική κίνηση κατά τον άξονα x και η περιστροφή γύρω από τον άξονα z (yaw). Αυτοί οι 2 β.ε. καλύπτουν τις ανάγκες της δικής μας ανάλυσης. Αυτό σημαίνει ότι μελετάμε ένα σύστημα χωρίς να επιθυμούμε την ακριβή μοντελοποίησή του, αλλά μία πολύ καλή προσέγγιση αυτού. Βασική προϋπόθεση είναι ότι το ρευστό είναι δεδομένο (γλυκό νερό) και η επιφάνειά του επίπεδη, χωρίς κυματισμούς.

Στη συνέχεια περιγράφεται σύντομα ο ρόλος της κάμερας στην εξαγωγή των εξισώσεων που θα περιγράφουν την κίνηση του πλοίου.

5.1.4 Ο ρόλος της κάμερας

Θεωρούμε τις παρακάτω εικόνες (Σχ.5.2). Είναι τυπικά στιγμιότυπα σαν αυτά που αναμένουμε να λαμβάνουμε από την κάμερα. Με τον ερυθρό στόχο καταδεικνύεται το επιθυμητό μέγεθος της φωτεινής πηγής ενώ με την ερυθρή διακεκομμένη γραμμή καταδεικνύεται η επιθυμητή θέση (μέση) της φωτεινής κηλίδας μέσα στο οπτικό πεδίο της κάμερας. Στο παράδειγμα που αναλύεται η φωτεινότητα δεν παίζει ρόλο. Τα μόνα μεγέθη που μας ενδιαφέρουν είναι η απόσταση του κέντρου της κηλίδας από τη μέση και το μέγεθός της. Επίσης, θεωρείται ότι η κάμερα είναι ενσωματωμένη στο πλοίο, δεν κινείται και είναι απόλυτα ευθυγραμμισμένη με το διαμήκη άξονα του σκάφους.



(α) (β) (γ)

Σχήμα 5.2. *Στιγμιότυπα κάμερας.*

Όπως είναι ευνόητο, στην (α) το φως είναι μακριά από το σκάφος και αρκετά προς τα αριστερά. Στη (β) το μέγεθος της κηλίδας είναι σωστό καθώς και η θέση είναι σχεδόν η ζητούμενη (χάνει λίγο προς τα δεξιά). Στη (γ) το μέγεθος της κηλίδας είναι μεγαλύτερο από το επιθυμητό που σημαίνει ότι η απόσταση του σκάφους από το φως είναι μικρότερη από την ζητούμενη ενώ ταυτόχρονα χάνει και λίγο προς τα αριστερά.

Σύμφωνα με τα παραπάνω γίνεται αντιληπτό ότι οι κινήσεις που μπορούμε να ελέγξουμε μέσω της κάμερας είναι:

- **Μεταφορική κίνηση κατά τον άξονα x:** Η μεταφορική κίνηση ελέγχεται μέσω της ένδειξης του μεγέθους της φωτεινής κηλίδας. Όσο μεγαλύτερη είναι η φωτεινή κηλίδα στα στιγμιότυπα της κάμερας, τόσο πιο κοντά στο φάρο βρίσκεται το σκάφος.

- **Περιστροφή γύρω από τον άξονα z (yaw):** Η περιστροφική κίνηση ελέγχεται μέσω της ένδειξης της οριζόντιας θέσης της φωτεινής κηλίδας. Σε κάθε στιγμιότυπο γίνεται σύγκριση της θέσης της κηλίδας με το μέσο κατά τον οριζόντιο άξονα του οπτικού πεδίου της κάμερας.

Έτσι, οι ζητούμενες εξισώσεις κίνησης του πλοίου πρέπει να εκφράζουν αυτές τις δύο κινήσεις. Η μαθηματική περιγραφή των ανωτέρω μπορεί να γίνει με χρήση των δύο παρακάτω μεταβλητών κατάστασης:

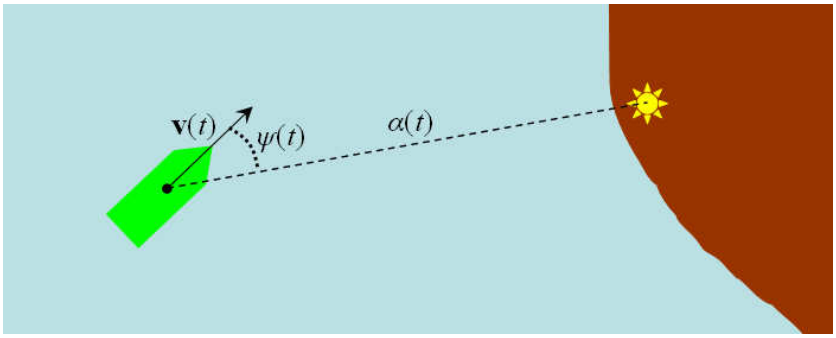
- *pn*(*t*): Αριθμός pixel (pixel number) της φωτεινής κηλίδας στο τρέχον στιγμιότυπο.
- *tp*(*t*): Εγκάρσια θέση (transverse position) της φωτεινής κηλίδας στο τρέχον στιγμιότυπο.

<u>Σχόλιο:</u> *Ουσιαστικά η μεταβλητή pn εκφράζει το ποσοστό της επιφάνειας του στιγμιότυπου το οποίο καλύπτεται από την παρακολουθούμενη κηλίδα φωτός. Έτσι, η μεταβλητή tp εκφράζει την θέση του ΚΒ της φωτεινής κηλίδας στον οριζόντιο άξονα.*

Η μεταφορική κίνηση ελέγχεται μέσω της μεταβλητής *pn*, ενώ η περιστροφική μέσω της μεταβλητής *tp*. Στη συνέχεια αναλύεται η λειτουργία αυτού του μηχανισμού. Πρώτα όμως είναι απαραίτητη η περιγραφή της κίνησης του πλοίου στο επίπεδο, μελετώντας τους 2 προαναφερθέντες β.ε.

5.2 Εξισώσεις κίνησης πλοίου στο επίπεδο

Στη συνέχεια δίνεται η απλοποιημένη δυναμική ανάλυση του σκάφους ως προς ακίνητο παρατηρητή, όπου για απλούστευση θεωρείται ότι είναι η πηγή του φωτός (ο φάρος που φαίνεται στο σχήμα 5.3). Όπως αναφέρθηκε παραπάνω η ανάλυση θα βασισθεί σε μοντέλο μειωμένης τάξης 2 βαθμών ελευθερίας, όπου το πλοίο κινείται στο επίπεδο του ήρεμου νερού (xy). Έτσι η κίνηση του πλοίου θα περιγράφεται από την μεταφορική κίνηση και την περιστροφή του γύρω από τον άξονα z (yaw).



Σχήμα 5.3. *Η κίνηση του πλοίου.*

Οι μεταβλητές που μας ενδιαφέρουν κατά τη συγκεκριμένη προσέγγιση είναι:

- *a(t)*: Απόσταση πλοίου από φωτεινή πηγή.
- *v(t)*: Γραμμική ταχύτητα πλοίου.
- *ψ(t)*: Γωνία που σχηματίζει η ταχύτητα με την ευθεία που συνδέει το σκάφος με το φάρο.

5.2.1 Μεταφορική κίνηση

Για την αλγεβρική τιμή της ταχύτητας του σκάφους, ισχύει η παρακάτω εξίσωση (βλέπε Παράρτημα Α για απόδειξη):

$$v'(t) = \frac{c_{1m}}{v(t)} \cdot u_1(t) - c_{1V} \cdot |v(t)| \cdot v(t), \qquad \text{για } T(t) < T_{max}$$

$$v'(t) = \frac{T_{\max}}{m_S} - c_{1V} \cdot |v(t)| \cdot v(t), \qquad \text{για } T(t) > T_{max} \tag{5.2}$$

Όπου:

- $u_1(t)$: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης ESC κινητήρων (0 ~ 100%).
- c_{1m}: Σταθερά.
- c_{1V}: Σταθερά.
- $T(t)$: Δύναμη ώσης.
- T_{max}: Μέγιστη δύναμη ώσης που μπορεί να αποδώσει το σύστημα.

<u>Σχόλιο:</u> *Στο υπόλοιπο του Κεφαλαίου 5 θα παρουσιάζουμε τη σχέση (5.2) μόνο με τον 1^o κλάδο της για ευκολία. Στη διαδικασία επίλυσης σε Η/Υ υπάρχει αλγόριθμος απόφασης για το ποιος από τους δύο κλάδους θα χρησιμοποιηθεί (βλέπε Παράρτημα Ε).*

Επίσης, ισχύει ότι:

$$a'(t) = v(t) \cdot \cos\psi(t) \Rightarrow$$

$$v(t) = \frac{a'(t)}{\cos\psi(t)} \tag{5.3}$$

Παραγωγίζοντας την (5.3) έχουμε:

$$v'(t) = \left(\frac{a'(t)}{\cos\psi(t)} \right)' \Rightarrow$$

$$v'(t) = \frac{a''(t) \cdot \cos\psi(t) - a'(t) \cdot \left(\cos\psi(t)\right)'}{\left(\cos\psi(t)\right)^2} \Rightarrow$$

$$v'(t) = \frac{a''(t) + a'(t) \cdot \psi'(t) \cdot \tan\psi(t)}{\cos\psi(t)} \tag{5.4}$$

Εξισώνοντας τις σχέσεις (5.2) και (5.4) έχουμε:

$$\frac{c_{1m}}{v(t)} \cdot u_1(t) - c_{1V} \cdot |v(t)| \cdot v(t) = \frac{a''(t) + a'(t) \cdot \psi'(t) \cdot \tan\psi(t)}{\cos\psi(t)} \tag{5.5}$$

Αντικαθιστώντας την ταχύτητα *v(t)* με τη σχέση (5.3) προκύπτει:

$$\frac{c_{1m} \cdot \cos\psi(t)}{a'(t)} \cdot u_1(t) - c_{1V} \cdot \left| \frac{a'(t)}{\cos\psi(t)} \right| \cdot \frac{a'(t)}{\cos\psi(t)} = \frac{a''(t) + a'(t) \cdot \psi'(t) \cdot \tan\psi(t)}{\cos\psi(t)} \Rightarrow$$

$$\frac{c_{1m}}{a'(t)} \cdot u_1(t) \cdot \cos^2\psi(t) - c_{1V} \cdot \left| \frac{a'(t)}{\cos\psi(t)} \right| \cdot a'(t) = a''(t) + a'(t) \cdot \psi'(t) \cdot \tan\psi(t) \Rightarrow$$

$$a''(t) + a'(t) \cdot \left(\psi'(t) \cdot \tan\psi(t) + c_{1V} \cdot \left| \frac{a'(t)}{\cos\psi(t)} \right| \right) - \frac{c_{1m}}{a'(t)} \cdot u_1(t) \cdot \cos^2\psi(t) = 0 \tag{5.6}$$

Λύνοντας ως προς *a''(t)* προκύπτει τελικά η σχέση για την επιτάχυνση του πλοίου:

$$a''(t) = \frac{c_{1m}}{a'(t)} \cdot u_1(t) \cdot \cos^2\psi(t) - a'(t) \cdot \left(\psi'(t) \cdot \tan\psi(t) + c_{1V} \cdot \left| \frac{a'(t)}{\cos\psi(t)} \right| \right) \tag{5.7}$$

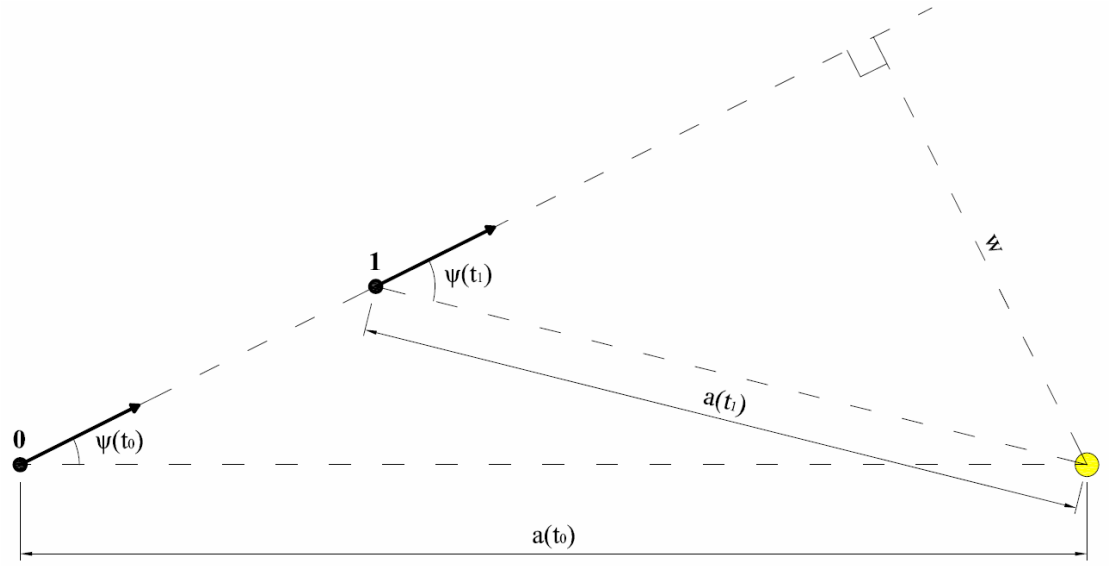
5.2.2 Περιστροφική κίνηση

Για τη γωνία που σχηματίζει η ταχύτητα του σκάφους με την ευθεία που συνδέει το σκάφος με το φάρο ισχύουν τα κατωτέρω.

Αρχικά κάνουμε μια υπόθεση:

Τα πηδάλια θεωρούνται μονίμως ευθυγραμμισμένα.

Σε αυτή την περίπτωση δεχόμαστε το σχήμα 5.4. Παρατηρούμε τις μεταβλητές ψ και a σε δύο διαδοχικές θέσεις (0 και 1) του πλοίου κατά την κίνησή του. Το πλοίο παριστάνεται σαν υλικό σημείο όπου στην πραγματικότητα είναι το Κ.Β. του.



Σχήμα 5.4. *Κίνηση του πλοίου με ευθυγραμμισμένα πηδάλια.*

Επειδή τα πηδάλια παραμένουν ακίνητα ισχύει:

$$w = a(t_0)\cdot\sin\psi(t_0) = a(t_1)\cdot\sin\psi(t_1) \Rightarrow$$

$$a(t)\cdot\sin\psi(t) = w = const$$

Παραγωγίζοντας έχουμε:

$$\frac{d}{dt}\left[a(t)\cdot\sin\psi(t)\right] = 0 \Rightarrow$$

$$a'(t)\cdot\sin\psi(t) + a(t)\cdot\psi'(t)\cdot\cos\psi(t) = 0 \Rightarrow$$

Έτσι, λύνοντας ως προς $\psi'(t)$ (ρυθμός μεταβολής της γωνίας)προκύπτει:

$$\psi'(t) = -\frac{a'(t)}{a(t)}\cdot\tan\psi(t) \qquad (5.8)$$

Αφού ολοκληρώθηκε η θεώρηση υπό προϋπόθεση (ευθυγραμμισμένα πηδάλια) μελετάται ο ρυθμός μεταβολής της γωνίας ψ όταν εκτρέπονται τα πηδάλια. Σε αυτή την περίπτωση ισχύει:

$$\psi'(t) = C_2(v(t)) \cdot u_2(t)$$

Θεωρώντας:

$$C_2(v(t)) \equiv c_2 \cdot v(t)$$

Όπου:

- c_2: Σταθερά.
- $u_2(t)$: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης servo ακροφυσίου προωθητήρων (–100 ~ +100%).

Προκύπτει:

$$\psi'(t) = c_2 \cdot v(t) \cdot u_2(t)$$

Αντικαθιστώντας την ταχύτητα $v(t)$ με τη σχέση (5.3) προκύπτει:

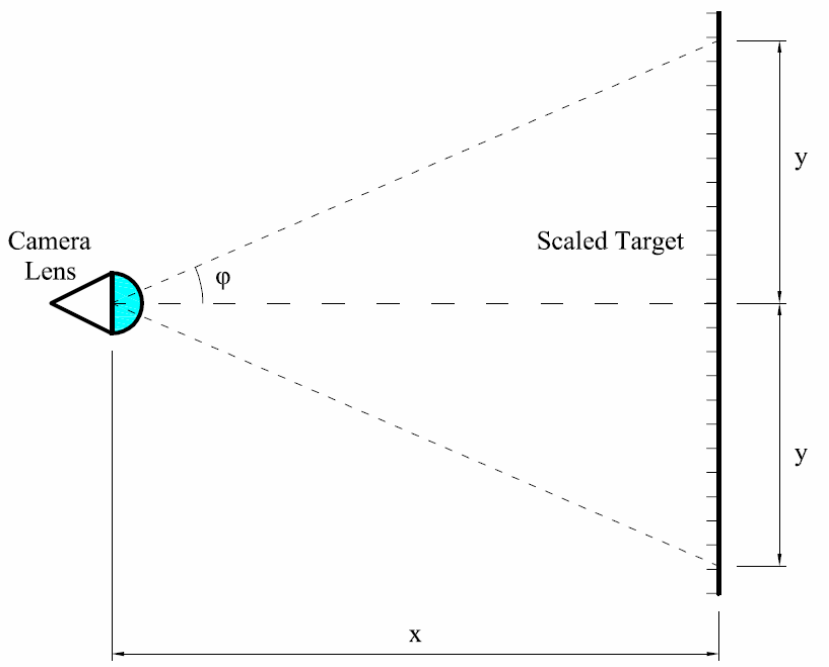
$$\psi'(t) = c_2 \cdot \frac{a'(t)}{\cos\psi(t)} \cdot u_2(t) \qquad (5.9)$$

Έτσι, θεωρούμε ότι αθροίζοντας τις δύο σχέσεις (5.8) και (5.9) καταλήγουμε στη συνολική σχέση που περιγράφει το ρυθμό μεταβολής της γωνίας ψ:

$$\psi'(t) = -\frac{a'(t)}{a(t)} \cdot \tan\psi(t) + c_2 \cdot \frac{a'(t)}{\cos\psi(t)} \cdot u_2(t) \qquad (5.10)$$

5.3 Βαθμονόμηση κάμερας

Απαραίτητος είναι ο υπολογισμός ενός μεγέθους το οποίο δε γνωρίζουμε και πρέπει να προσδιορισθεί πειραματικά. Αυτό το μέγεθος είναι η γωνία του οπτικού πεδίου της κάμερας (2φ) (Σχ. 5.5). Για να υπολογισθεί, παίρνουμε φωτογραφίες με την κάμερα έναν βαθμονομημένο στόχο (Scaled Target) τον οποίο τοποθετούμε σε συγκεκριμένες αποστάσεις (x) από το φακό της κάμερας. Στη φωτογραφία η οποία λαμβάνεται από την κάμερα βλέπουμε ένα μέρος του στόχου του οποίου γνωρίζουμε τη διάσταση (y).



Σχήμα 5.5. *Κάτοψη της διάταξης υπολογισμού της γωνίας του οπτικού πεδίου της κάμερας.*

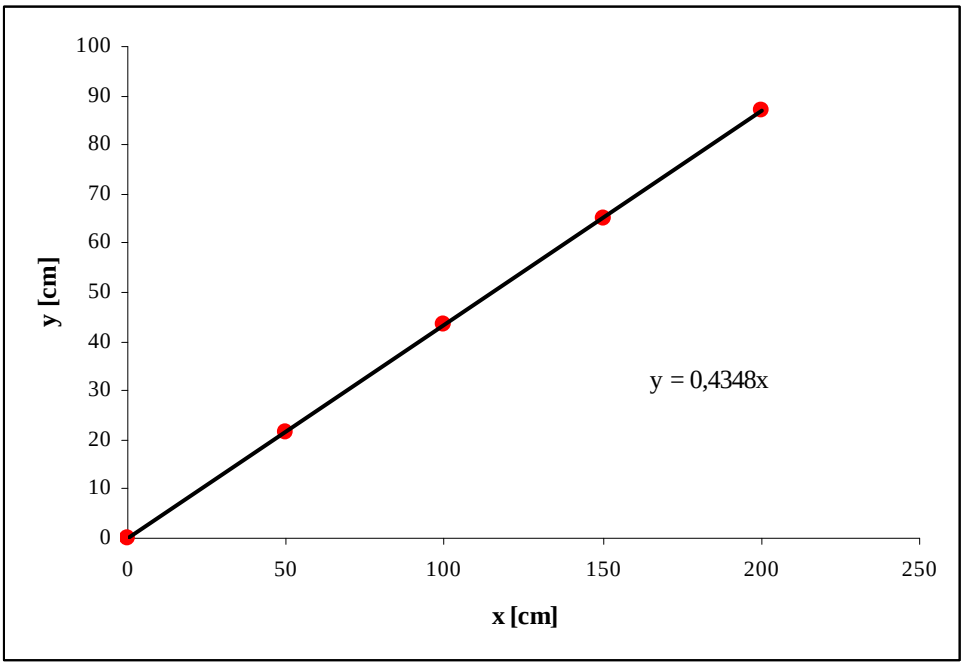
Σχόλιο: *Περισσότερες πληροφορίες σχετικά με την τεχνολογία φωτογραφικών μηχανών υπάρχουν στη βιβλιογραφία (Young, 1994).*

Σύμφωνα λοιπόν με μια σειρά φωτογραφιών συντάσσεται ο πίνακας:

Πίνακας 5.1
Αντιστοιχία $y \rightarrow x$.

x	y
[cm]	[cm]
0	0,0
50	21,7
100	43,5
150	65,2
200	87,0

Από τα ζεύγη του πίνακα 5.1 δημιουργείται η καμπύλη του σχήματος 5.6.



Σχήμα 5.6. *Συσχέτιση των μεταβλητών x – y.*

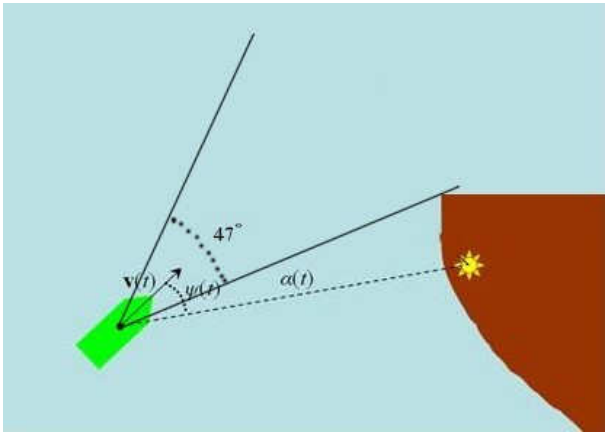
Παρατηρούμε ότι η σχέση των *x – y* είναι γραμμική. Έτσι, ο συντελεστής διεύθυνσης της παραπάνω ευθείας αντιστοιχεί στη γωνία φ, δηλαδή:

$$\lambda = 0{,}4348 = \tan^{-1}\varphi \Rightarrow$$

$$\varphi = 23{,}5^{o}$$

Επομένως, θεωρώντας ότι η κάμερα είναι σταθερή στον άξονά της και δεν περιστρέφεται, η γωνία του οπτικού της πεδίου είναι 2φ = 47°.

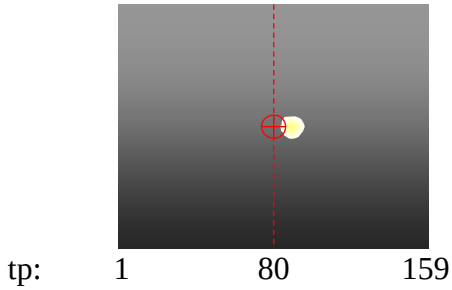
Αν $\psi(t) > \varphi = 23{,}5^{o}$, τότε η κάμερα δε «βλέπει» το φάρο (Σχ. 5.7).



Σχήμα 5.7. *Οπτικό πεδίο κάμερας.*

<u>Σχόλιο:</u> *Το οπτικό πεδίο της κάμερας αυξάνεται δραστικά χρησιμοποιώντας μηχανισμό περιστροφής της κάμερας περί τον κάθετο προς το επίπεδο άξονα. Αυτή η επιλογή όμως δε χρησιμοποιείται στην παρούσα ανάλυση.*

Έστω ότι $\psi(t) < 23{,}5^{o}$. Αυτό σημαίνει ότι ο φάρος βρίσκεται στο οπτικό πεδίο της κάμερας. Η κάμερα έχει βαθμονομημένο τον οριζόντιο άξονα του οπτικού της πεδίου σύμφωνα με το σχήμα 5.8.



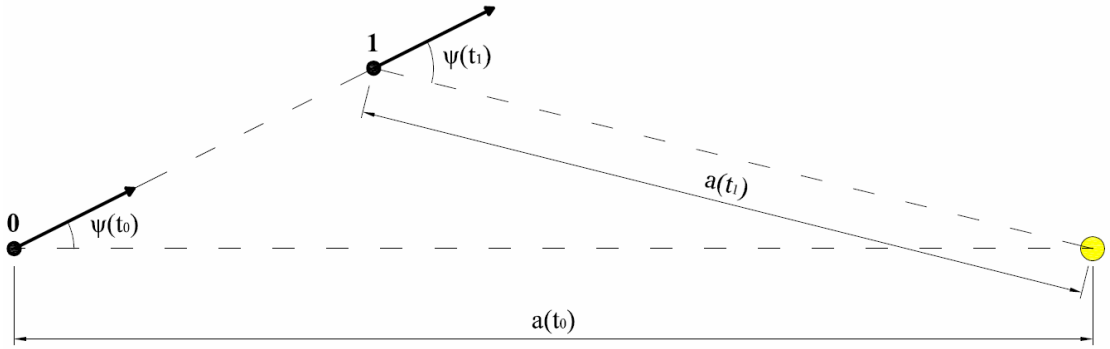
Σχήμα 5.8. *Ο οριζόντιος άξονας της κάμερας.*

Έτσι, σε κάθε στιγμιότυπο η κάμερα επιστρέφει μια τιμή για το *tp* (θέση κέντρου βάρους του φωτός στον οριζόντιο άξονα) από 1 έως 159. Επίσης σε κάθε στιγμιότυπο επιστρέφει μια τιμή για το *pn* (αριθμός pixel του φωτός) από 0 έως 255.

Στη συνέχεια γίνεται μια προσπάθεια «σύνδεσης» των μεταβλητών *ψ*, *a* με τις *tp*, *pn*. Για να γίνει αυτό κάνουμε μια τριγωνομετρική ανάλυση του προβλήματος.

5.3.1 Μεταβλητή tp

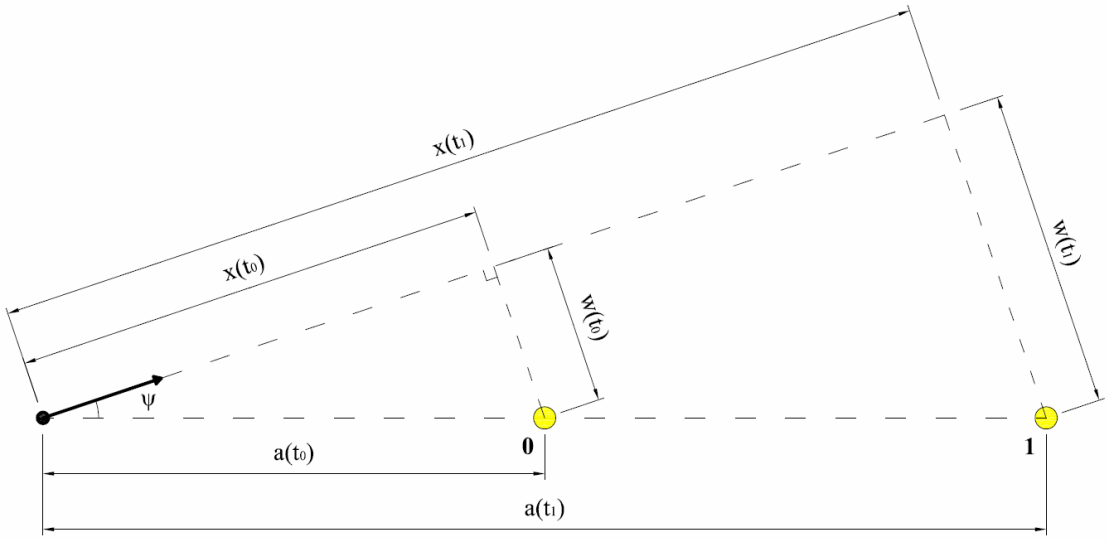
Ο φάρος βρίσκεται σε δεδομένη θέση στο επίπεδο. Έστω, ότι το πλοίο βρίσκεται αρχικά στη θέση 0, με $\psi(t_0)$ και $a(t_0)$ (Σχ.5.9). Σε αυτή τη θέση η κάμερα επιστρέφει μία τιμή $tp(t_0)$ που δίνει την οριζόντια θέση του φωτός στο οπτικό της πεδίο.



Σχήμα 5.9. *Συσχέτιση μεταβλητών ψ, a* με τη μεταβλητή *tp*. *Το πλοίο κινείται σε ευθεία πορεία.*

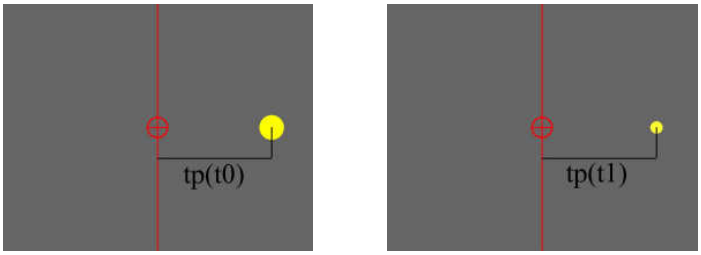
Αν το πλοίο κινηθεί ευθεία, χωρίς να στρέψει το πηδάλιό του, θα βρεθεί στη θέση 1. Τώρα η γωνία της ταχύτητας του σκάφους με την απόσταση του από το φάρο είναι $\psi(t_1) > \psi(t_0)$ και η απόσταση είναι $a(t_1) < a(t_0)$ (θεωρούμε $0^{o} < \psi < 90^{o}$). Η κάμερα επιστρέφει την τιμή $tp(t_1)$, η οποία εξαρτάται μόνο από τη σχετική γωνία του σκάφους με το φάρο *ψ* και όχι από την απόσταση *a*.

Για να γίνει πιο κατανοητό αυτό θα εξετάσουμε δύο περιπτώσεις που αφορούν τα στιγμιότυπα της κάμερας θεωρώντας ακίνητο το πλοίο σε μία θέση στο επίπεδο (Σχ. 5.10).



Σχήμα 5.10. *Συσχέτιση μεταβλητών ψ, a* με τη *μεταβλητή tp. Διαφορετικές θέσεις φάρου.*

Περίπτωση α. Η απόσταση του πλοίου από το φάρο είναι $a(t_0)$ και η γωνία ψ. Η απόλυτη απόσταση του διανύσματος της κατεύθυνσης του πλοίου από το φάρο είναι $w(t_0)$. Το αποτύπωμα του φάρου στο στιγμιότυπο της κάμερας έχει κατά τον οριζόντιο άξονα απόσταση $tp(t_0)$ από το μέσον του οπτικού της πεδίου (Σχ. 5.11).



Σχήμα 5.11. *Στιγμιότυπα κάμερας όταν ο φάρος βρίσκεται στις θέσεις 0 και 1 αντίστοιχα. $tp(t_0) = tp(t_1)$*

Περίπτωση β. Η απόσταση του πλοίου από το φάρο είναι $a(t_1)$ και η γωνία πάλι ψ. Η απόλυτη απόσταση της προέκτασης του διανύσματος της κίνησης του πλοίου από το φάρο είναι $w(t_1)$. Το αποτύπωμα του φάρου στο στιγμιότυπο της κάμερας έχει απόσταση $tp(t_1)$ από το κέντρο του οπτικού της πεδίου ενώ το μέγεθος του είναι μικρότερο από πριν (Σχ. 5.11)

Το σημαντικό συμπέρασμα από την παραπάνω ανάλυση είναι ότι:

$$tp(t_0) = tp(t_1)$$

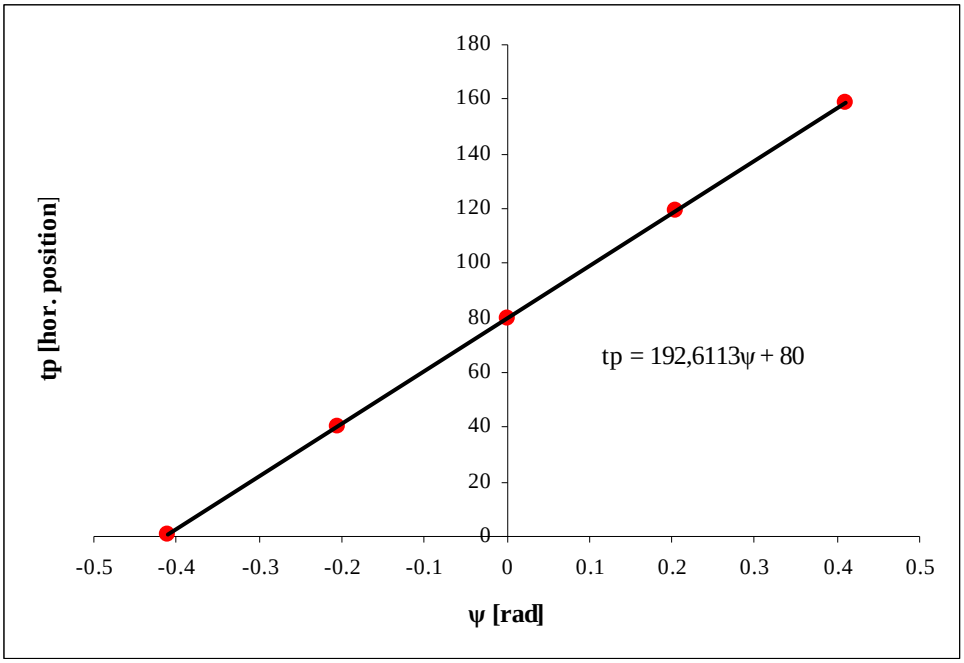
Έτσι αντιλαμβανόμαστε ότι η μεταβλητή *tp* δεν εξαρτάται από την απόσταση *a* του πλοίου από το φάρο, αλλά μόνο από τη γωνία *ψ*.

Στη συνέχεια κάνουμε ένα πείραμα για να προσδιορίσουμε την εξάρτιση του *tp* από το *ψ*. Τοποθετούμε το φως υπό συγκεκριμένες γωνίες και ζητάμε από την κάμερα να μας δώσει πληροφορίες για την οριζόντια θέση του φωτός στο οπτικό της πεδίο. Δηλαδή για τις δεδομένες γωνίες ψ_n θα επιστρέφονται από την κάμερα οι αντίστοιχες τιμές tp_n. Έτσι συντάσσεται ο ακόλουθος πίνακας.

Πίνακας 5.2
Αντιστοιχία $\psi \rightarrow tp$.

tp	**ψ**	**ψ**
[hor. position]	**[°]**	**[rad]**
1	-23.50	-0.41
40	-11.75	-0.21
80	0.00	0.00
120	11.75	0.21
159	23.50	0.41

Από τα ζεύγη του πίνακα 5.2 δημιουργείται η καμπύλη του σχήματος 5.12.



Σχήμα 5.12. *Συσχέτιση των μεταβλητών tp – ψ.*

Παρατηρούμε ότι η σχέση των *tp – ψ* είναι γραμμική. Έτσι, η ζητούμενη εξίσωση για τη μεταβλητή *tp* είναι:

$$tp(t) = 192{,}6113 \cdot \psi(t) + 80 \qquad (5.10)$$

5.3.2 Μεταβλητή pn

Η μεταβλητή *pn* επηρεάζεται από την απόσταση *a* και τη γωνία *ψ*. Σύμφωνα με το σχήμα 5.10 έχουμε:

Αν: $x(t_1) > x(t_0) \quad \rightarrow \quad pn(t_1) < pn(t_0)$

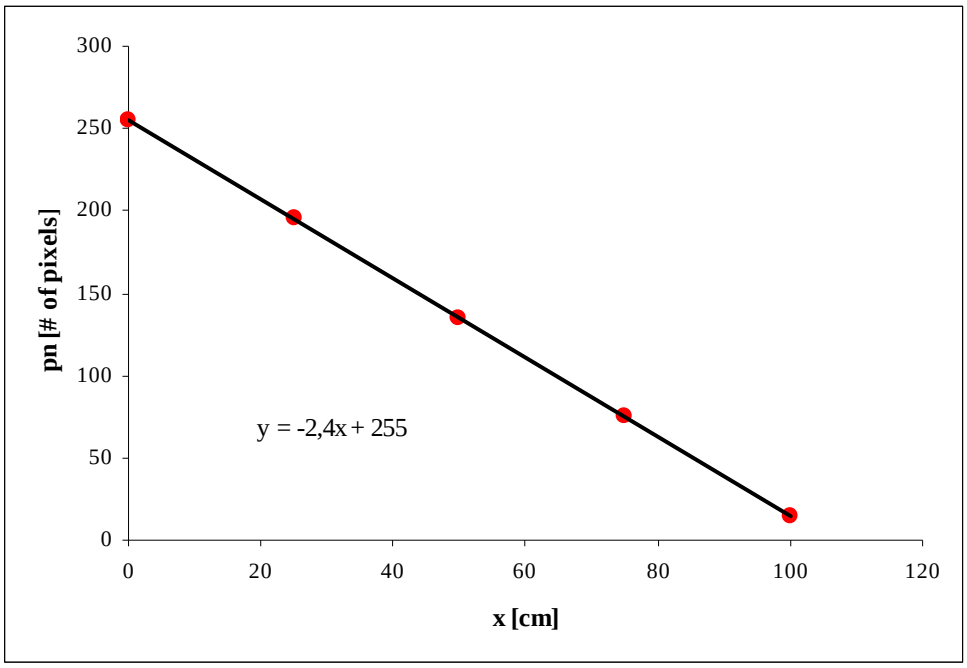
Το παραπάνω συμπέρασμα συνάγεται από το γεγονός ότι όσο πιο κοντά στο φάρο βρίσκεται το πλοίο (άρα και η κάμερα) τόσο μεγαλύτερο ποσοστό του οπτικού της πεδίου καλύπτεται από το φως. Άρα όσο το πλοίο πλησιάζει το φάρο τόσο θα μεγαλώνει ο αριθμός των pixels του παρακολουθούμενου φωτός, ο οποίος όπως γνωρίζουμε εκφράζεται από τη μεταβλητή *pn*.

Για τον προσδιορισμό της εξάρτισης του *pn* από το *x* πρέπει να πραγματοποιηθεί το ακόλουθο πείραμα. Τοποθετούμε το φως σε συγκεκριμένες αποστάσεις στην προέκταση της κατεύθυνσης της κάμερας (όχι υπό γωνία, δηλαδή ψ = 0). Ζητάμε από την κάμερα να μας δώσει πληροφορίες για τον αριθμό των παρακολουθούμενων pixels στο οπτικό της πεδίο σε κάθε προεπιλεγμένη απόσταση. Δηλαδή για τις δεδομένες αποστάσεις x_n θα επιστρέφονται από την κάμερα οι αντίστοιχες τιμές pn_n. Έτσι συντάσσεται ο ακόλουθος πίνακας.

Πίνακας 5.3
Αντιστοιχία $x \rightarrow pn$.

pn	**x**
[# of pixels]	**[cm]**
255	0
195	25
135	50
75	75
15	100

Από τα ζεύγη του πίνακα 5.3 δημιουργείται η καμπύλη του σχήματος 5.13.



Σχήμα 5.13. *Συσχέτιση των μεταβλητών tp – ψ.*

Παρατηρούμε ότι η σχέση των *pn – x* είναι γραμμική. Έτσι, η ζητούμενη εξίσωση για τη μεταβλητή *pn* είναι:

$$pn(t) = -c_3 \cdot x(t) + 255 \qquad (5.11)$$

<u>Σχόλιο:</u> *Η σταθερά 255 ερμηνεύεται ποιοτικά ως εξής. Ο μέγιστος αριθμός pixel παρακολουθούμενου φωτός είναι 255. Δηλαδή όταν το φως καλύπτει το 100% του οπτικού πεδίου της κάμερας επιστρέφεται η τιμή pn = 255. Για την παρούσα ανάλυση θεωρούμε ότι το φως καλύπτει το 100% του οπτικού πεδίου της κάμερας σε απόσταση x=0 από το φακό της. Αυτό δεν είναι απόλυτα σωστό, αφού το ποσοστό επικάλυψης του οπτικού πεδίου της κάμερας εξαρτάται από το μέγεθος του φωτός. Αν για παράδειγμα θεωρούσαμε ένα φάρο πολύ μεγάλων διαστάσεων σε σχέση με το φακό της κάμερας, αυτός θα κάλυπτε το οπτικό της πεδίο κατά 100% σε απόσταση x>0. Στα πλαίσια όμως των δικών μας δοκιμών, οι οποίες γίνονται με φάρους ανάλογων μεγεθών σε σχέση με το μέγεθος του φακού της κάμερας, δεχόμαστε τη γενική σχέση (5.11) σαν μια πολύ καλή προσέγγιση.*

Από σχήμα 5.10 ισχύει:

$$x(t) = a(t) \cdot \cos\psi(t)$$

Άρα η μεταβλητή *pn* εξαρτάται από τις μεταβλητές *a*, *ψ* σύμφωνα με τη σχέση:

$$pn(t) = -c_3 \cdot a(t) \cdot \cos\psi(t) + 255 \qquad (5.12)$$

Η σταθερά c_3 εξαρτάται από το είδος και το μέγεθος του φωτός. Γι' αυτό το λόγο θα επιλέξουμε ένα συγκεκριμένο φως ώστε να γίνουν όλα τα πειράματα βάσει αυτό. Θα ακολουθήσει πείραμα όπως περιγράφτηκε παραπάνω για τον προσδιορισμό της σταθεράς c_3. Έτσι, κατά το πείραμα υπολογισμού της σταθεράς c_3 για το δεδομένο φάρο, έχουμε:

- $\psi = 0{,}337 rad$
- $a = 11m$
- $pn = 3$

Επομένως:

$$pn = -c_3 \cdot a \cdot \cos\psi + 255 \Rightarrow$$

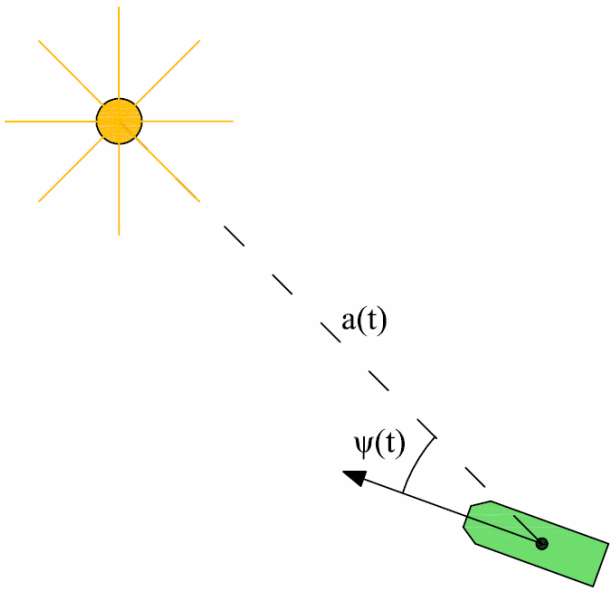
$$c_3 = \frac{255 - pn}{a \cdot \cos\psi} \Rightarrow$$

$$c_3 = \frac{255 - 3}{11 \cdot \cos 0{,}337} \Rightarrow$$

Άρα:

$$c_3 = 24{,}3 \qquad (5.13)$$

Σχόλιο: *Πρέπει να σημειωθεί ότι οι σχέσεις (5.10) και (5.12) ισχύουν σε περίπτωση ομοιοτροπικού φωτός (Σχ. 5.14), δηλαδή φως το οποίο φαίνεται το ίδιο από όλες τις οπτικές γωνίες. Για παράδειγμα, μια φωτεινή σφαίρα η οποία εκπέμπει ακτινικά προς όλες τις κατευθύνσεις είναι ένα τέτοιο (ομοιοτροπικό) φως.*



Σχήμα 5.14. *Ομοιοτροπικό φως.*

6. ΤΑΥΤΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

6.1 Εξισώσεις κίνησης πλοίου στο χώρο κατάστασης

Στο παρόν κεφάλαιο περιγράφεται η διαδικασία εξαγωγής του ακριβούς μαθηματικού μοντέλου του συστήματός μας. Η γενική του μορφή περιγράφεται από τις εξισώσεις (5.7) και (5.10). Όμως σε αυτές τις εξισώσεις υπεισέρχονται οι άγνωστες παράμετροι c_{1m}, c_{1V}, c_2. Ο υπολογισμός αυτών των παραμέτρων αποτελεί τον προσδιορισμό των εξισώσεων κίνησης του σκάφους που μελετάμε, δηλαδή την ταυτοποίηση του συστήματος.

Στη συνέχεια μετατρέπουμε τις εξισώσεις (5.7) και (5.10) σε εξισώσεις κατάστασης του συστήματος πλοίου. Το διάνυσμα καταστάσεως **x** είναι:

$$x = \begin{bmatrix} a & a' & \psi \end{bmatrix}^T$$

- $x_1 = a$
- $x_2 = a'$
- $x_3 = \psi$

<u>Σχόλιο:</u> *Περισσότερες πληροφορίες σχετικά με τις εξισώσεις στο χώρο κατάστασης υπάρχουν στη βιβλιογραφία (Ξηρός, 2004).*

Το διάνυσμα εισόδου είναι:

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$$

- u_1: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης ESC κινητήρων (0 ~ 100%)
- u_2: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης servo ακροφυσίου προωθητήρων (–100 ~ +100%).
- u_3: Θέση ακροφυσίου αναπόδισης (+/–1)
- u_4: κανονικοποιημένος λόγος εύρους παλμού οδήγησης ESC ανεμιστήρα ανύψωσης (0 - 100%)

Παραλείποντας όμως τα στοιχεία u_3 και u_4 τα οποία σχετίζονται με την αναπόδιση και τον ανεμιστήρα ανύψωσης, έχουμε:

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$$

<u>Σχόλιο:</u> *Η παραπάνω παράλειψη στο διάνυσμα εισόδου έγινε διότι στα πειράματα που θα ακολουθήσουν, δε θα χρησιμοποιηθούν όλες οι επιφάνειες ελέγχου παρά μόνο το thruster και το rudder. Αυτά ισχύουν για την παρούσα ανάλυση όπως αναφέρεται στην παράγραφο 5.1.1.*

Παρακάτω υπολογίζονται τα στοιχεία του **x'** συναρτήσει των στοιχείων του **x** και του **u**.

$$x' = \begin{bmatrix} a' & a'' & \psi' \end{bmatrix}^T$$

$$x_1' = a' = x_2$$

$$x_2' = a'' = \begin{cases} \dfrac{c_{1m}}{x_2} \cdot u_1 \cdot \cos^2 x_3 - x_2 \cdot \left(\left(-\dfrac{x_2}{x_1} \cdot \tan x_3 + c_2 \cdot \dfrac{x_2}{\cos x_3} \cdot u_2 \right) \cdot \tan x_3 + c_{1V} \cdot \left| \dfrac{x_2}{\cos x_3} \right| \right), & \gamma\iota\alpha \quad T(t) < T_{\max} \\ \dfrac{T_{\max}}{m_S} \cdot \cos x_3 - x_2 \cdot \left(\left(-\dfrac{x_2}{x_1} \cdot \tan x_3 + c_2 \cdot \dfrac{x_2}{\cos x_3} \cdot u_2 \right) \cdot \tan x_3 + c_{1V} \cdot \left| \dfrac{x_2}{\cos x_3} \right| \right), & \gamma\iota\alpha \quad T(t) > T_{\max} \end{cases} \tag{6.1}$$

$$x_3' = \psi' = -\frac{x_2}{x_1} \cdot \tan x_3 + c_2 \cdot \frac{x_2}{\cos x_3} \cdot u_2$$

Το διάνυσμα εξόδου **y** προκύπτει από την επίλυση του συστήματος διαφορικών εξισώσεων **x'** (6.1). Δηλαδή το διάνυσμα εξόδου **y** ταυτίζεται με το **x**. Έτσι:

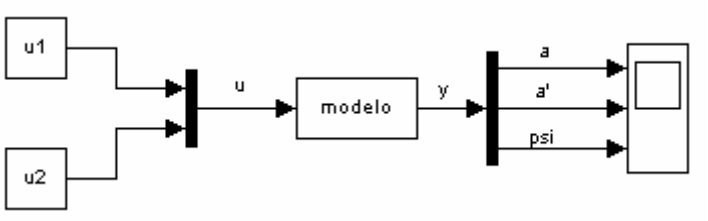
$$y = \begin{bmatrix} a & a' & \psi \end{bmatrix}^T$$

Το παραπάνω σύστημα δ.ε. είναι συζευγμένο και μη γραμμικό. Έτσι, η επίλυση του επιτυγχάνεται με τη βοήθεια Η/Υ. Αυτό είναι απαραίτητο αφού είναι αναγκαία η επίλυση για ένα χρονικό διάστημα [0 t] και όχι για ένα στιγμιότυπο. Το περιβάλλον που χρησιμοποιείται για την επίλυση του συστήματος είναι το Matlab/Simulink©.

Σε αυτό το σημείο πρέπει να αναφερθούμε στην αριθμητική μέθοδο που χρησιμοποιείται για την επίλυση του συστήματος. Αυτή είναι η μέθοδος Dormand – Prince η οποία χρησιμοποιείται στην επίλυση συνήθων διαφορικών εξισώσεων (*Ordinary Differential Equations – ODE*) με μεταβλητό βήμα (variable step). Η μέθοδος Dormand – Prince ανήκει στην οικογένεια Runge – Kutta.

Για να επιτευχθεί λύση του συστήματος (6.1) πρέπει να προσδιορισθούν οι παράμετροι $c = [c_{1m} \quad c_{1V} \quad c_2]$ του μοντέλου.

Για τον προσδιορισμό των παραμέτρων *c* ακολουθείται η παρακάτω μεθοδολογία. Αρχικά δημιουργούμε ένα διάγραμμα block (Σχ. 6.1) στο Simulink το οποίο μας παρέχει τη δυνατότητα να υπολογίσουμε ποια είναι η απόκριση *y* του συστήματος για δεδομένη διέγερση *u* την οποία ορίζουμε εμείς. Δηλαδή, ορίζουμε ένα «σενάριο κίνησης» μέσω των u_1 και u_2 και παρατηρούμε την εικονική κίνηση του πλοίου μέσω των *a*, *a'* και *ψ*.



Σχήμα 6.1. *Διάγραμμα block του συστήματος.*

Στο δομικό διάγραμμα έχουμε εισάγει τις εξισώσεις (6.1) (block modelo) όμως περιέχουν τις άγνωστες παραμέτρους *c*. Για να είναι έγκυρα τα αποτελέσματα του παραπάνω μοντέλου, πρέπει να γίνει ταυτοποίηση συστήματος. Έτσι, πρέπει να έχουμε στη διάθεσή μας πειραματικά δεδομένα από δοκιμές με το σκάφος AIRCAT. Σύμφωνα με τα δεδομένα των πειραμάτων όπου θα έχουμε ορισμένες εισόδους και θα μετρήσουμε τις εξόδους, θα στοιχειοθετηθεί το ανωτέρω μοντέλο με τις αντίστοιχες εικονικές εισόδους. Στη συνέχεια θα δημιουργηθεί ένας αλγόριθμος βελτιστοποίησης όπου θα υπολογίζει τις βέλτιστες σταθερές *c* σύμφωνα με τις οποίες το μοντέλο θα επιστρέφει τις επιθυμητές εξόδους για τις συγκεκριμένες εισόδους. Όλα αυτά παρουσιάζονται αναλυτικά στη συνέχεια.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τις μεθόδους αριθμητικής ολοκλήρωσης και βελτιστοποίησης υπάρχουν στη βιβλιογραφία (Μπακόπουλος, 1999).*

6.2 Πειραματική Διαδικασία

Στη συνέχεια περιγράφονται αναλυτικά οι τρεις δοκιμές που πραγματοποιήθηκαν σε λίμνη με το σκάφος AIRCAT. Σε κάθε πείραμα παρουσιάζεται η διαδικασία διεξαγωγής του πειράματος.

Τα μεγέθη που υπολογίζονται από τα πειράματα είναι:

- Μέγιστη ταχύτητα πλοίου.
- Χρόνος ενός κύκλου στροφής με 100% γωνία πηδαλίου και 100% φορτίο προωστήρων.
- Διανυόμενη απόσταση του σκάφους από τη στιγμή που σταματά να λειτουργεί ο κινητήρας του σκάφους, αφού έχει αποκτηθεί μέγιστη ταχύτητα, έως ότου ακινητοποιηθεί.

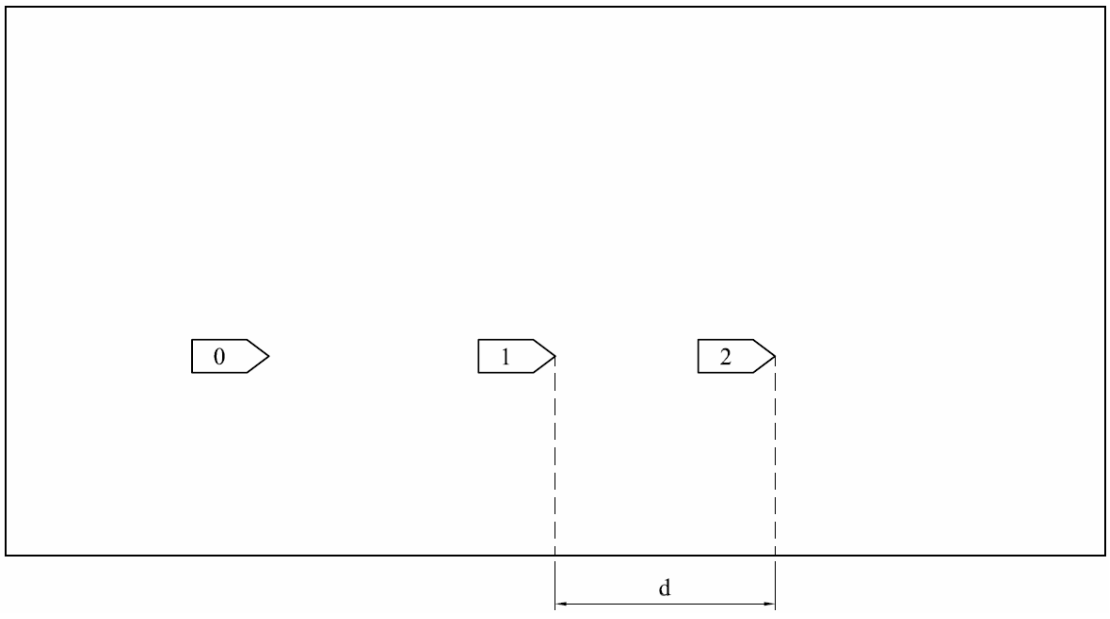
6.2.1 Πείραμα 1: Υπολογισμός μέγιστης ταχύτητας

α. Εκκίνηση πλοίου από τη θέση 0.
β. Επιβολή μέγιστου φορτίου στους προωστήρες και διατήρηση του σκάφος σε ευθεία πορεία παράλληλα προς τις μεγάλες πλευρές τις λίμνης.
γ. Επιτάχυνση σκάφους έως ότου αποκτηθεί μέγιστη ταχύτητα.
δ. Τη στιγμή που το AIRCAT περνά από τη θέση 1 με σταθερή (μέγιστη) ταχύτητα αρχίζει η χρονομέτρηση.
ε. Μόλις περνά από τη θέση 2 με σταθερή (μέγιστη) ταχύτητα σταματάει η χρονομέτρηση.

στ. Η ταχύτητα προκύπτει από τη σχέση: $v = \frac{s_2 - s_1}{t_2 - t_1} = \frac{d}{t}$

Πίνακας 6.1
Μετρήσεις πειράματος 1.

	Μέγιστη Ταχύτητα
Απόσταση	d = 20 m
α/α	[sec]
1	5,9
2	5,7
Μ.Ο.	**5,8**
v [m/s]	**3,448**



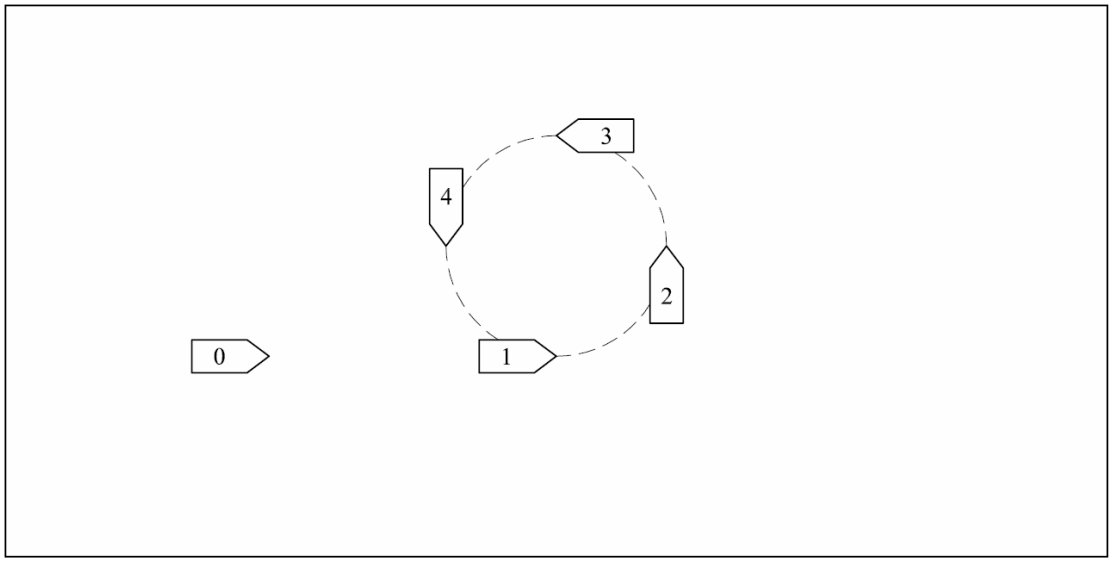
Σχήμα 6.2. *Περιγραφή πειράματος 1.*

6.2.2 Πείραμα 2: Υπολογισμός χρόνου 1 κύκλου στροφής

α. Εκκίνηση πλοίου από τη θέση 0.
β. Επιβολή μέγιστου φορτίου στους προωστήρες και διατήρηση του σκάφος σε ευθεία πορεία παράλληλα προς τις μεγάλες πλευρές τις λίμνης.
γ. Επιτάχυνση σκάφους έως ότου αποκτηθεί μέγιστη ταχύτητα.
δ. Τη στιγμή που το AIRCAT περνά από τη θέση 1 με σταθερή (μέγιστη) ταχύτητα εκτρέπεται βηματικά το πηδάλιο αποκτώντας μέγιστη γωνία κλίσης και αρχίζει η χρονομέτρηση.
ε. Μόλις ολοκληρωθεί 1 κύκλος και περνά το σκάφος ξανά από τη θέση 1 με σταθερή (μέγιστη) ταχύτητα και μέγιστη γωνία πηδαλίου σταματάει η χρονομέτρηση.

Πίνακας 6.2
Μετρήσεις πειράματος 2.

	Χρόνος 1 Κύκλου
α/α	[sec]
1	7,6
2	7,6
Μ.Ο.	**7,6**



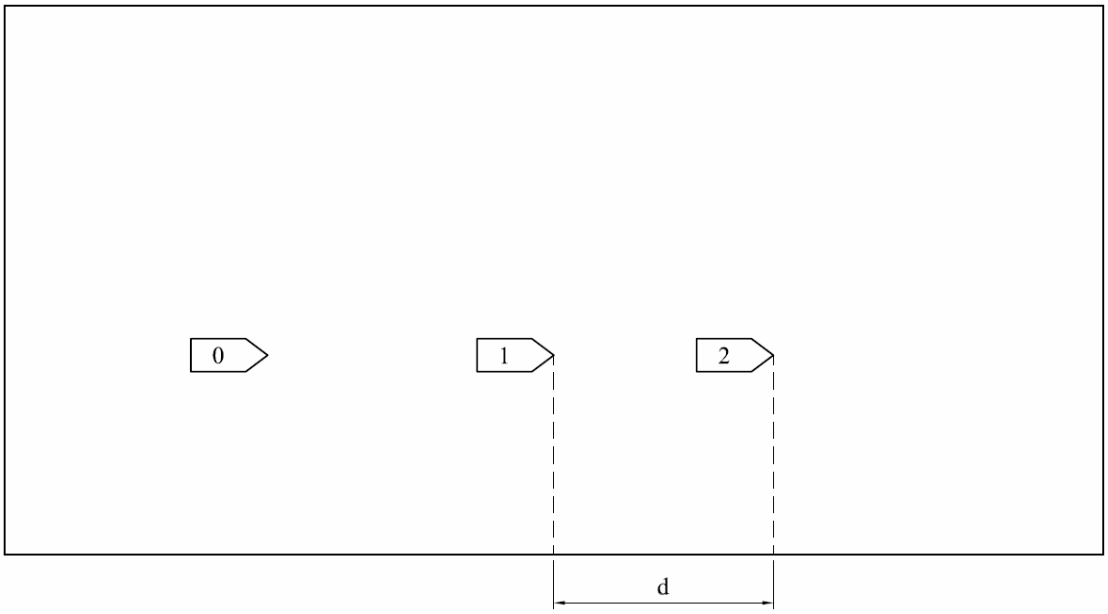
Σχήμα 6.3. *Περιγραφή πειράματος 2.*

6.2.3 Πείραμα 3: Υπολογισμός απόστασης σταματήματος

α. Εκκίνηση πλοίου από τη θέση 0.
β. Επιβολή μέγιστου φορτίου στους προωστήρες και διατήρηση του σκάφος σε ευθεία πορεία παράλληλα προς τις μεγάλες πλευρές τις λίμνης.
γ. Επιτάχυνση σκάφους έως ότου αποκτηθεί μέγιστη ταχύτητα.
δ. Τη στιγμή που το AIRCAT περνά από τη θέση 1 με σταθερή (μέγιστη) ταχύτητα μηδενίζεται το φορτίο των προωστήρων (το σκάφος επιβραδύνει) και αρχίζει η χρονομέτρηση.
ε. Μόλις σταματήσει να κινείται το σκάφος προς τα εμπρός (θέση 2) μετράται η απόσταση d.

Πίνακας 6.3
Μετρήσεις πειράματος 3.

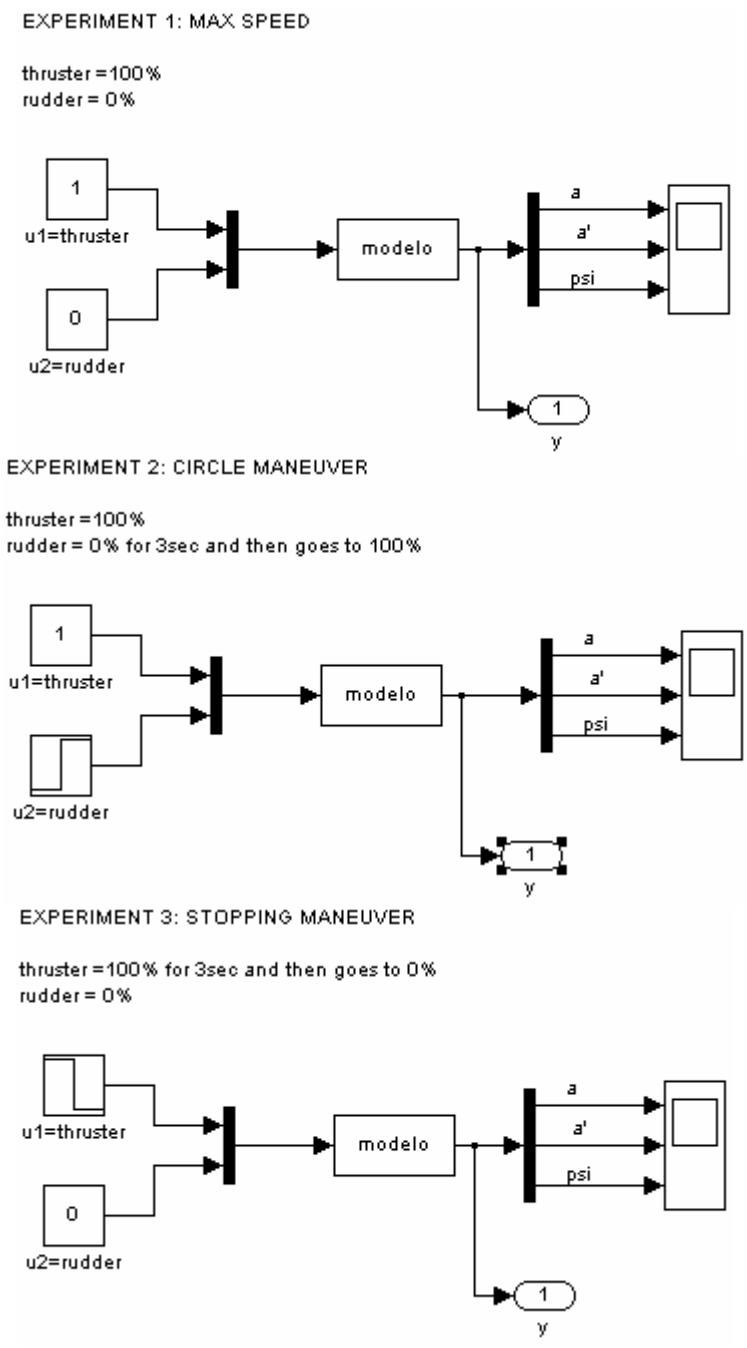
	Σταμάτημα
α/α	[m]
1	5,70
2	6,00
Μ.Ο.	**5,85**



Σχήμα 6.4. *Περιγραφή πειράματος 3.*

6.3 Υπολογισμός παραμέτρων c

Έχοντας πλέον τα 3 πειραματικά μετρημένα μεγέθη μπορούμε να λύσουμε το μη γραμμικό σύστημα εξισώσεων με 3 αγνώστους. Επειδή όμως το σύστημα δεν έχει μοναδική λύση, θα επιστρατευτεί αλγόριθμος βελτιστοποίησης όπου θα υπολογισθούν τα βέλτιστα *c*. Σύμφωνα με το βασικό μοντέλο του διαγράμματος block (Σχ. 6.1) δημιουργούμε τρία όμοια μοντέλα στο Simulink τα οποία αντιστοιχούν στα τρία πειράματα (Σχ. 6.5).

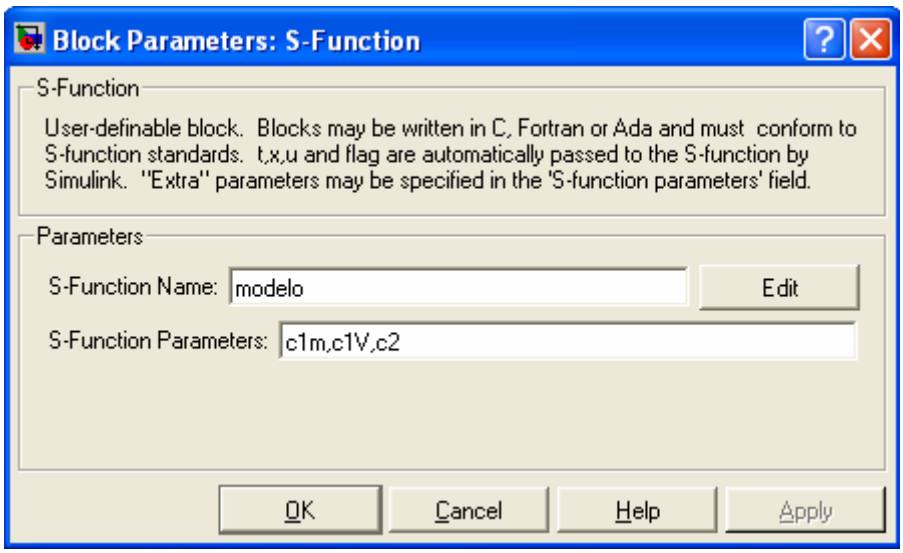


Σχήμα 6.5. *Μοντέλα των τριών πειραμάτων (op_1, op_2, op_3).*

Το block “modelo” περιέχει τις δ.ε. (6.1) σε μορφή S-Function (βλέπε Παράρτημα modelo.m). Οι σταθερές *c* ορίζονται σαν εξωτερικές παράμετροι (Σχ. 6.6), δηλαδή:

```
function [sys,x0,str,ts] = modelo(t,x,u,flag,c1m,c1V,c2)
```

Αυτό γίνεται προκειμένου να μπορούν οι συναρτήσεις βελτιστοποίησης να αλλάζουν τις σταθερές ώστε να υπολογιστούν οι βέλτιστες.



Σχήμα 6.6. *Παράμετροι της S-Function “modelo”.*

Επίσης ορίζουμε μια συνάρτηση η οποία θα επιστρέφει μια παράσταση την οποία θα ελαχιστοποιούμε με τις συναρτήσεις βελτιστοποίησης. Η επιστρεφόμενη παράσταση θα είναι:

$$f(a,v,t) = (a_p - a_{cr})^2 + (v_p - v_{cr})^2 + (t_p - t_{cr})^2$$

Όπου:

- a_{cr}: Μαθηματικά υπολογισμένη απόσταση σταματήματος (μοντέλο 3).
- a_p = 5,85 m: Πειραματικά μετρημένη απόσταση σταματήματος (πείραμα 3).
- v_{cr}: Μαθηματικά υπολογισμένη μέγιστη ταχύτητα (μοντέλο 1).
- v_p = 3,448 m/s: Πειραματικά μετρημένη μέγιστη ταχύτητα (πείραμα 1).
- t_{cr}: Μαθηματικά υπολογισμένος χρόνος στροφής (μοντέλο 2).
- t_p = 7,6 sec: Πειραματικά μετρημένος χρόνος στροφής (πείραμα 2).

Οι τιμές a_{cr}, v_{cr}, t_{cr} εξάγονται από τα μοντέλα του Simulink σύμφωνα με τα εξής κριτήρια:

- Από το πείραμα 1 θέλουμε την ταχύτητα, άρα θα ζητήσουμε μια τιμή της $2^{ης}$ στήλης του διανύσματος εξόδου (ταχύτητα) η οποία αντιστοιχεί στη μέγιστη ταχύτητα.
- Από το πείραμα 2 θέλουμε τη διάρκεια κύκλου στροφής, άρα θα ζητήσουμε την τιμή του διανύσματος του χρόνου για την οποία έχουμε ψ = 2π = 6,28, δηλαδή θα αναζητήσουμε στην $3^η$ στήλη του διανύσματος εξόδου (γωνία), μια τιμή γύρω στο 6,28 και θα δούμε σε ποιο χρόνο αντιστοιχεί.

- Από το πείραμα 3 θέλουμε την απόσταση την οποία διανύει το σκάφος έως ότου η ταχύτητα μηδενιστεί., άρα θα ζητήσουμε την τιμή της $1^{ης}$ στήλη του διανύσματος εξόδου (απόσταση) για την οποία έχουμε v ~ 0, δηλαδή θα αναζητήσουμε στην $2^{η}$ στήλη του διανύσματος εξόδου (ταχύτητα), μία τιμή που να είναι μικρότερη από ένα όριο ταχύτητας (v<0,2m/s) και θα δούμε σε ποια απόσταση αντιστοιχεί.

Σύμφωνα με τα παραπάνω κριτήρια δημιουργούμε τη συνάρτηση c_fun (βλέπε Παράρτημα Ε, c_fun.m) όπου δέχεται σαν όρισμα τις σταθερές *c*, δηλαδή:

```
function F = c_fun(c)

...

F=(ap-acr)^2+(vp-vcr)^2+(tp-tcr)^2;
```

Έτσι ελαχιστοποιώντας την παραπάνω παράσταση αλλάζοντας τα *c*, θα προκύψουν οι ζητούμενες σταθερές. Η ελαχιστοποίηση της παράστασης θα γίνει με τη βοήθεια της συνάρτησης fminsearch (βλέπε Παράρτημα, find_c.m) η οποία χρησιμοποιεί τη μέθοδο SIMPLEX του Lagarias. Η fminsearch θα μας επιστρέψει τις τιμές των *c* για τις οποίες η F ελαχιστοποιείται, άρα τα βέλτιστα *c*.

Συνοψίζοντας,αναφέρουμε ότι για να υπολογιστούν τα βέλτιστα *c*, χρειαζόμαστε τα αρχεία που παρουσιάζονται στον πίνακα 5.3.4.1.

Πίνακας 6.4
Απαιτούμενα αρχεία Matlab/Simulink για την εύρεση των παραμέτρων *c*.

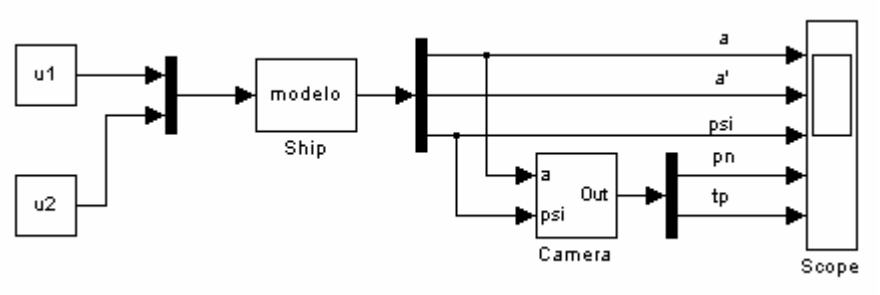
Name	File type	Description
find_c.m	M-File	Calls fminsearch and computes the optimum c constants.
c_fun.m	M-File Function	Called by fminsearch and returns the representation to be minimized.
modelo.m	M-File S-Function	Includes the dynamic system in state space.
op_1.mdl	MDL-File	Executes experiment 1.
op_2.mdl	MDL-File	Executes experiment 2.
op_3.mdl	MDL-File	Executes experiment 3.

Διαθέτοντας αυτά τα αρχεία, το μόνο που πρέπει να κάνουμε είναι να τρέξουμε το «find_c.m». Αυτό θα επιστρέψει τα ζητούμενα c τα οποία προκύπτουν:

- $c_{1m} = 19{,}9694$
- $c_{1V} = 0{,}4872$
- $c_2 = 0{,}2443$

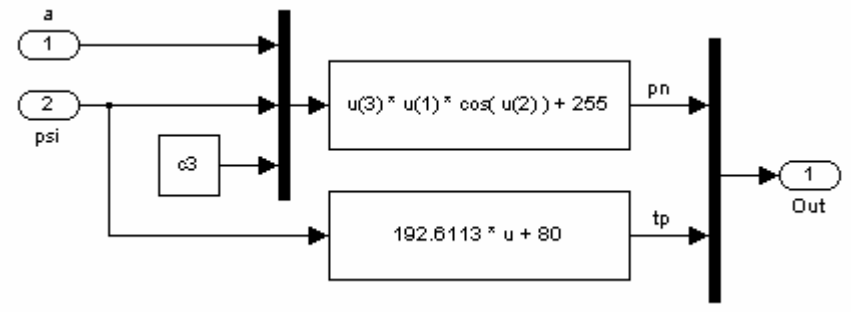
6.4 Το σύστημα πλοίο – κάμερα

Για να μοντελοποιηθεί το συνολικό σύστημα θα προσάψουμε το μετασχηματισμό για τις μεταβλητές της κάμερας (βλέπε Παράγραφος 5.3) στο ήδη υπάρχον μοντέλο κίνησης του πλοίου. Έτσι, δημιουργούμε ένα δομικό διάγραμμα (Σχ. 6.7) στο Simulink το οποίο εκφράζει το σύστημα πλοίο – κάμερα.



Σχήμα 6.7. *Δομικό διάγραμμα συστήματος πλοίου - κάμερας.*

Σύμφωνα με το παραπάνω σχήμα, για κάθε διέγερση u_1 και u_2 έχουμε τη δυνατότητα να παρατηρήσουμε εκτός των *a*, *a'* και *ψ*, τα *pn* και *tp*, δηλαδή το οπτικό πεδίο της κάμερας. Στο block «Ship» υπάρχουν οι εξισώσεις κατάστασης (6.1), αφού έχουμε υπολογίσει τις παραμέτρους *c*. Στο block «Camera» υπάρχουν οι μετασχηματισμοί των σχέσεων (5.10) και (5.12) (Σχ. 6.8).



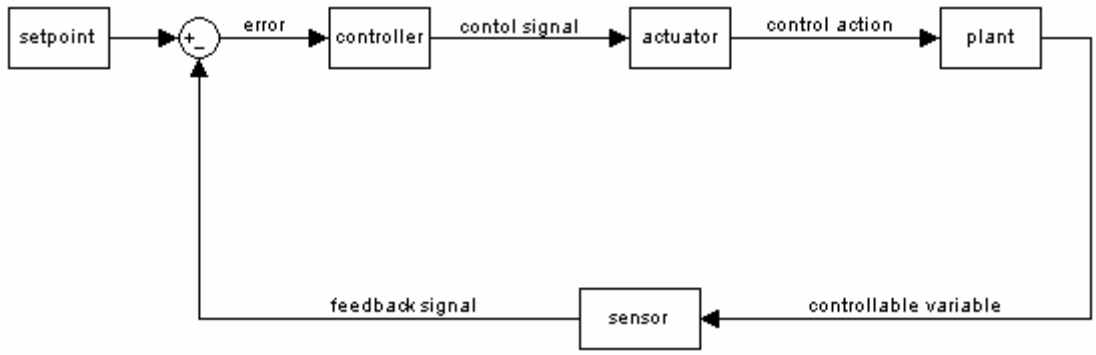
Σχήμα 6.8. *Block μετασχηματισμού κάμερας.*

Το παραπάνω συνολικό μοντέλο είναι απαραίτητο για τη διεξαγωγή των εικονικών πειραμάτων παρακολούθησης στόχου. Η χρήση του παρουσιάζεται στο επόμενο κεφάλαιο.

7. ΑΥΤΟΜΑΤΟΣ ΕΛΕΓΧΟΣ

7.1 Έλεγχος με ανάδραση

Το σύστημα ελέγχου είναι η ενότητα των αλληλοσυνδεδεμένων συνιστωσών, η οποία αποδίδει επιθυμητές αποκρίσεις σε διεγέρσεις εισόδου. Με δεδομένη την επιθυμητή απόκριση του συστήματος, είναι σκόπιμο να παραχθεί ένα σήμα ανάλογο προς τη διαφορά της επιθυμητής και της πραγματικής απόκρισης (*error*). Η χρήση του συστήματος αυτού προς έλεγχο μιας εγκατάστασης (*plant*) οδηγεί σε μια αλληλουχία δράσεων, που αποτελούν «κλειστό βρόχο» (*closed loop*), το δε σύστημα που προκύπτει καλείται σύστημα ανάδρασης (*feedback*) (Σχ. 7.1).



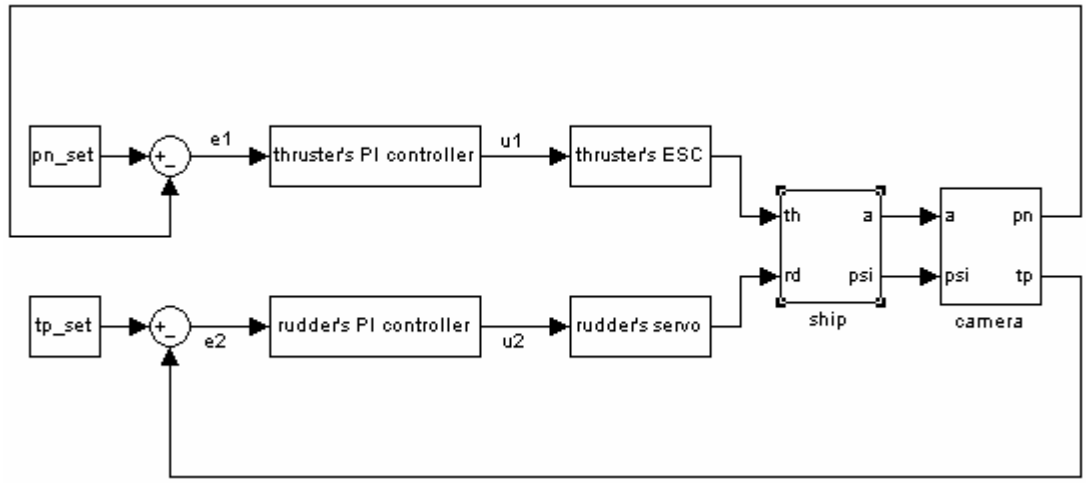
Σχήμα 7.1. *Σύστημα κλειστού βρόχου.*

Η εγκατάσταση που μελετάται στη παρούσα εργασία είναι στην πραγματικότητα το πλοίο χωρίς τους ενεργοποιητές, δηλαδή η γάστρα του πλοίου με τα waterjets.

Ο ρόλος του ενεργοποιητή (*actuator*) είναι να μετατρέπει το σήμα ελέγχου (*control signal*) σε δράση ελέγχου (*control action*). Στο υπό μελέτη σύστημα οι 2 ενεργοποιητές που χρησιμοποιούνται είναι ο σερβομηχανισμός κίνησης του ακροφυσίου προωθητήρων (*rudder's servo*) και ο ηλεκτρονικός ρυθμιστής στροφών των κινητήρων προώσεως (*thruster's ESC*). Οι δράσεις ελέγχου είναι η θέση του ακροφυσίου προωθητήρων και το φορτίο των κινητήρων.

Ο ρόλος του αισθητήρα (*sensor*) είναι να παρέχει το απαραίτητο σήμα αναδράσεως που προκύπτει στην έξοδο του υπό μελέτη συστήματος. Στην περίπτωσή μας ο αισθητήρας είναι η κάμερα ενώ οι μεταβλητές υπό έλεγχο (*controllable variables*) είναι η απόσταση του πλοίου από το φάρο και η γωνία που σχηματίζει η ταχύτητα με την ευθεία που συνδέει το σκάφος με το φάρο. Το σήμα μετρήσεως ή ανατροφοδοτήσεως (*feedback signal*) είναι τα δεδομένα που επιστρέφει η κάμερα σχετικά με το οπτικό της πεδίο (βλέπε ανάλυση προηγούμενων κεφαλαίων). Τέλος η τιμή αναφοράς (*set point*) είναι οι επιθυμητές τιμές των δεδομένων που επιστρέφει η κάμερα.

Η καρδιά ενός συστήματος αυτομάτου ελέγχου είναι ο ελεγκτής (*controller*). Ο ελεγκτής οδηγείται από το σφάλμα (*error*) που παρουσιάζουν τα σήματα μετρήσεως σε σχέση με την τιμή αναφοράς που εκφράζει την επιθυμητή «θέση» του συστήματος και δημιουργεί σήματα ελέγχου. Τα σήματα ελέγχου τα οποία παράγει ο ελεγκτής του συστήματός μας είναι ο κανονικοποιημένος λόγος εύρους παλμού οδήγησης ρυθμιστή στροφών κινητήρων καθώς και ο κανονικοποιημένος λόγος εύρους παλμού οδήγησης σερβομηχανισμού ακροφυσίου προωθητήρων. Σύμφωνα με τις παραπάνω αντιστοιχήσεις, το σύστημα κλειστού βρόχου που μελετάμε παρουσιάζεται στο σχήμα 7.2.



Σχήμα 7.2. *Υπό μελέτη σύστημα κλειστού βρόχου.*

Σε κάθε περίπτωση , ο ελεγκτής επιχειρεί να μειώσει όσο το δυνατόν περισσότερο το σήμα του σφάλματος *(error) = (set point) – (feedback signal)* με το οποίο τροφοδοτείται, παράγοντας κατάλληλα σήματα ελέγχου. Έτσι, πρέπει να αποφασιστεί η τεχνική ελέγχου η οποία θα δημιουργεί αυτά τα σήματα. Μεταξύ των τεχνικών ON/OFF, P, PI και PID, επιλέγεται η τεχνική ελέγχου PI για τους ακόλουθους λόγους:

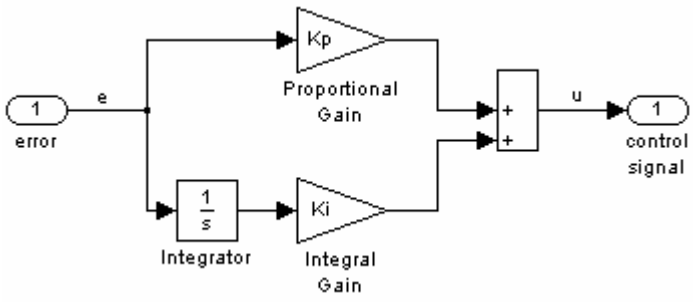
- Ο έλεγχος ON/OFF για μικρού εύρους νεκρή ζώνη (*dead zone*) παρουσιάζει το φαινόμενο των ταλαντώσεων οριακού κύκλου (*limit cycle*), ενώ για μεγάλου εύρους νεκρή ζώνη αυξάνεται το σφάλμα μόνιμης κατάστασης.
- Ο Αναλογικός έλεγχος P (*Proportional*) παρουσιάζει μεγάλη ομοιότητα με τον ON/OFF, αλλά στην ιδανική περίπτωση (χωρίς κορεσμό) μπορεί να εξαλείψει το πρόβλημα των ταλαντώσεων οριακού κύκλου. Όμως το σφάλμα μόνιμης κατάστασης παραμένει.
- Ο Αναλογικός – Ολοκληρωτικός – Διαφορικός έλεγχος PID (*Proportional – Integral – Differential*) ενώ διορθώνει το πρόβλημα του PI το οποίο είναι η ταλαντωτική συμπεριφορά της εξόδου του κλειστού βρόχου, έχει ένα βασικό μειονέκτημα. Ο διαφορικός όρος D με την παραγώγιση ενισχύει το θόρυβο, που κάνει πάντα την εμφάνισή του στο σήμα ανατροφοδοτήσεως.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τις ανωτέρω έννοιες υπάρχουν στη βιβλιογραφία (Κρικέλης, 2000 – Ξηρός 2004).*

7.2 Ελεγκτής PI

7.2.1 Γενική περιγραφή ελεγκτή PI

Σύμφωνα με τα παραπάνω επιλέγεται σαν τεχνική ελέγχου του υπό μελέτη συστήματος, ο Αναλογικός – Ολοκληρωτικός έλεγχος PI (*Proportional – Integral*). Ο έλεγχος PI, εξαλείφει το μειονέκτημα του απλού Αναλογικού ελεγκτή P το οποίο όπως αναφέρθηκε είναι το σφάλμα μονίμου καταστάσεως και επιπλέον δεν έχει τον Διαφορικό όρο του ελεγκτή PID ο οποίος ενισχύει το θόρυβο. Το δομικό διάγραμμα του ελεγκτή PI δίνεται στο σχήμα 7.3.



Σχήμα 7.3. *Δομικό διάγραμμα ελεγκτή PI.*

Η σχέση μεταξύ εισόδου *e* και εξόδου *u* ενός κλασικού ελεγκτή 2 όρων (P και I) δίνεται από τη σχέση:

$$u(t) = P(t) + I(t) \tag{7.1}$$

Όπου:

- $P(t) = K_p \cdot e(t)$ (7.2)

- $I(t) = K_i \cdot \int_0^t e(\xi) d\xi$ (7.3)

Το πρόβλημα συντονισμού (γενικότερα σχεδίασης ή σύνθεσης) ενός ελεγκτή PI συνίσταται στον προσδιορισμό των 2 σταθερών του νόμου ελέγχου K_p και K_i ώστε να ικανοποιούνται συγκεκριμένες προδιαγραφές για τη μεταβατική και μόνιμη συμπεριφορά του συστήματος κλειστού βρόχου. Στη συνέχεια παρουσιάζεται η μεθοδολογία υπολογισμού των σταθερών K_p και K_i του υπό μελέτη συστήματος.

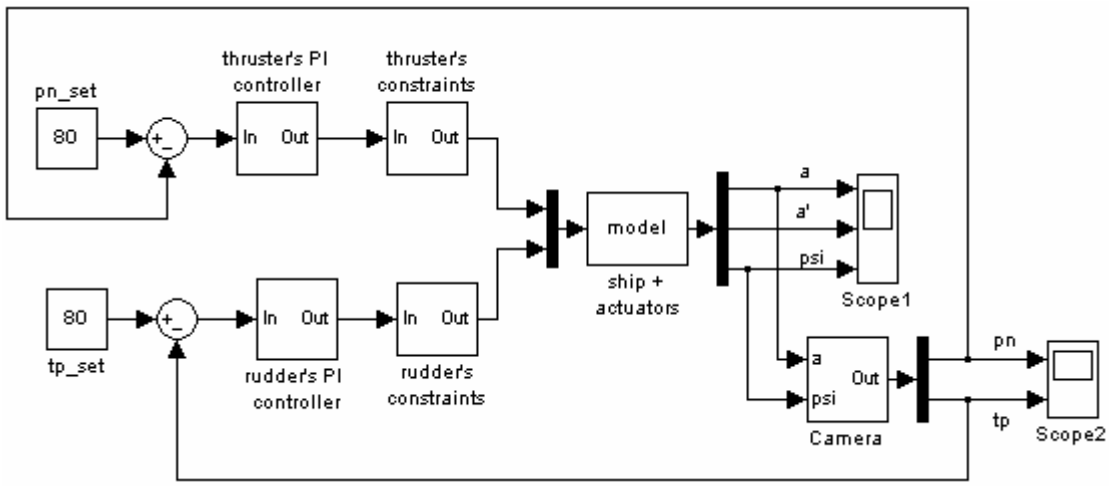
7.2.2 Προσδιορισμός σταθερών K_p, K_i

Αρχικά πρέπει να διευκρινιστεί ότι σκοπός της παρούσας παραγράφου δεν είναι να σχεδιαστεί το ιδανικό σύστημα ελεγκτών για το υπό μελέτη πλοίο. Στόχος μας είναι να δημιουργήσουμε κατάλληλους ελεγκτές οι οποίοι υπό ορισμένες συνθήκες θα εξυπηρετούν τις ανάγκες μας. Αυτό που θέλουμε δηλαδή, είναι η διεξαγωγή ενός πειράματος παρακολούθησης στόχου σε εικονικό και πραγματικό περιβάλλον. Σύμφωνα με τα αποτελέσματα και τη σύγκριση των ανωτέρω θα μπορέσουμε να αξιολογήσουμε την ταυτοποίηση του συστήματος που παρουσιάζεται στα προηγούμενα κεφάλαια. Επομένως, στη συνέχεια παρουσιάζεται η μεθοδολογία προσδιορισμού των κατάλληλων για την εφαρμογή μας K_p και K_i.

Οι σταθερές K_p και K_i θα υπολογισθούν με τη βοήθεια προσομοιώσεων. Σύμφωνα με το μοντέλο που έχουμε δημιουργήσει σε περιβάλλον Matlab/Simulink© θα γίνουν εικονικά πειράματα – δοκιμές αλλάζοντας τις σταθερές έως ότου παρατηρήσουμε κάποια επιθυμητή συμπεριφορά μέσω των εικονικών μεταβλητών *a, a', ψ* (βλέπε Κεφάλαιο 5). Η λογική γύρω από τη διαδικασία υπολογισμού των σταθερών έχει ως ακολούθως.

Έχοντας το συνολικό μοντέλο του σχήματος 6.7 εισάγουμε στην είσοδο του συστήματος, τους ελεγκτές PI και ανατροφοδοτούμε το σύστημα με τα δεδομένα εξόδου της κάμερας. Οι τιμές αναφοράς είναι εκείνες οι οποίες αντιπροσωπεύουν μέσω των ενδείξεων της κάμερας, την επιθυμητή απόσταση του πλοίου από το φάρο *a* και την επιθυμητή γωνία *ψ*.

Θεωρώντας ότι το σκάφος βρίσκεται σε κατάσταση ηρεμίας και παρακολουθεί το φάρο στην επιθυμητή θέση, δεν πραγματοποιείται καμία εντολή από τους ελεγκτές προς τους ενεργοποιητές. Αν όμως θεωρήσουμε αρχικές συνθήκες (Α.Σ.) διαφορετικές από τις επιθυμητές, οι PI controllers του thruster και του rudder δίνοντας τις κατάλληλες εντολές, «προσπαθούν» να επαναφέρουν το πλοίο στην επιθυμητή θέση ως προς τη θέση του φάρου. Έτσι, προκύπτει το ακόλουθο δομικό διάγραμμα (Σχ. 7.4).



Σχήμα 7.4. *Δομικό διάγραμμα εικονικού πειράματος.*

Σχόλιο: *Παρατηρούμε ότι στο παραπάνω δομικό διάγραμμα δεν υπάρχουν οι ενεργοποιητές, αλλά υπάρχουν κάποιοι περιορισμοί (constraints). Ο λόγος είναι ότι η δυναμική των ενεργοποιητών εμπεριέχεται στο block του πλοίου (ship + actuators). Όπως γνωρίζουμε η σχέση εισόδου εξόδου αυτού του block εκφράζεται από τις δ.ε. (6.1) όπου περιέχουν την δυναμική συμπεριφορά των ενεργοποιητών. Οι περιορισμοί τίθενται για πρακτικούς λόγους, έτσι ώστε οι τιμές που δέχονται από τους ελεγκτές να βρίσκονται στα προβλεπόμενα όρια των εισόδων των ενεργοποιητών.*

Κάνοντας δοκιμές για τους συντελεστές των 2 PI controllers, K_{pt}, K_{it}, K_{pr} και K_{ir} παρατηρούμε την εικονική συμπεριφορά του πλοίου μέσω των μεταβλητών *a, a'* και *ψ*. Έτσι, προσδιορίζονται οι παράμετροι των 2 ελεγκτών οι οποίες δημιουργούν την εικονικά επιθυμητή συμπεριφορά του σκάφους όταν διεγερθεί ανάλογα (θέση φάρου διάφορη της επιθυμητής).

Συνοψίζοντας,αναφέρουμε ότι για να γίνουν τα εικονικά πειράματα, ώστε να οι ζητούμενες παράμετροι χρειαζόμαστε τα αρχεία που παρουσιάζονται στον πίνακα 7.1.

Πίνακας 7.1

Απαιτούμενα αρχεία Matlab/Simulink για την εύρεση των παραμέτρων K_p και K_i.

Name	File type	Description
PI_test.mdl	MDL-File	Executes the visual experiment.
modelo1.m	M-File S-Function	Includes the dynamic system in state space.

Διαθέτοντας αυτά τα αρχεία, το μόνο που πρέπει να κάνουμε είναι να τρέξουμε το «PI_test.mdl», δοκιμάζοντας διάφορα K_p και K_i για τους δύο PI controllers. Παρακολουθώντας την εικονική κίνηση επιλέγουμε τις παραμέτρους εκείνες οι οποίες μας ικανοποιούν μέσω των μεταβλητών *a, a'* και *ψ*. Έτσι, επιλέγουμε:

- $K_{pt} = 0{,}01$
- $K_{it} = 0{,}001$
- $K_{pr} = 0{,}0125$
- $K_{ir} = 0{,}001$

Ο κώδικας που περιέχεται στα ανωτέρω αρχεία παρουσιάζεται αναλυτικά στο Παράρτημα Ε.

7.2.3 Ψηφιακή μορφή ελεγκτή PI

Όπως γνωρίζουμε για να πραγματοποιηθεί το πείραμα παρακολούθησης στόχου στο φυσικό μοντέλο, πρέπει οι παραπάνω ελεγκτές να εισαχθούν κατάλληλα στον microcontroller. Ο microcontroller είναι μια διάταξη η οποία επεξεργάζεται ψηφιακά τα δεδομένα. Για το λόγο αυτό πρέπει να μετατρέψουμε τη σχέση (7.1) από το συνεχή στο διακριτό χρόνο. Αυτό θα γίνει διακριτοποιώντας ξεχωριστά τον Αναλογικό P και Ολοκληρωτικό όρο I και αθροίζοντάς τους θα προκύψει η σχέση εισόδου – εξόδου του ελεγκτή PI σε ψηφιακή μορφή.

Στο συνεχή ελεγκτή ο αναλογικός όρος ορίζεται ως:

$$P(t) = K_p \cdot e(t) \tag{7.4}$$

Ο αλγόριθμος διακριτού χρόνου υλοποιείται με απλή αντικατάσταση των συνεχών μεταβλητών με τις διακριτοποιημένες τιμές τους την ίδια στιγμή, δηλαδή:

$$P(k) = K_p \cdot e(k), \qquad k = 0, 1, 2 \ldots \tag{7.5}$$

Το k δηλώνει τις στιγμές δειγματοληψίας, δηλαδή τις στιγμές που η αντίστοιχη συνεχής μεταβλητή μετατρέπεται σε ψηφιακή σε κάθε περίοδο δειγματοληψίας.

Ο ολοκληρωτικός όρος του συνεχούς ισοδύναμου ορίζεται από το ολοκλήρωμα:

$$I(t) = K_i \cdot \int_0^t e(\xi) d\xi \tag{7.6}$$

Παραγωγίζοντας την παραπάνω σχέση προκύπτει:

$$I'(t) = K_i \cdot e(t) \tag{7.8}$$

Υπάρχουν διάφοροι τρόποι προσέγγισης του ολοκληρωτικού όρου. Εδώ θα χρησιμοποιηθεί η μέθοδος της διαφοράς προς τα πίσω. Σύμφωνα με αυτή τη μέθοδο η σχέση (7.8) γίνεται:

$$I'(t) \approx \frac{I(k) - I(k-1)}{T_S} \tag{7.9}$$

Όπου, T_S είναι η περίοδος δειγματοληψίας.

Συνδυάζοντας τις σχέσεις (7.8) και (7.9) και θεωρώντας ότι το δεύτερο μέλος της (7.8) στο διακριτό χρόνο είναι $K_i \cdot e(k)$, $k = 0, 1, 2 \ldots$, μετατρέπεται η σχέση (7.6) από το συνεχή στο διακριτό χρόνο:

$$\frac{I(k) - I(k-1)}{T_S} = K_i \cdot e(k) \Rightarrow$$

$$I(k) = I(k-1) + T_S \cdot K_i \cdot e(k)\ , \qquad k = 0, 1, 2 \ldots \tag{7.10}$$

Η συνολική μορφή του ψηφιακού αλγόριθμου του PI ελεγκτή, που θα υλοποιηθεί από τον μικροεπεξεργαστή προκύπτει από την άθροιση των εξισώσεων (7.5) και (7.10). Έτσι, η σχέση (7.1) σε ψηφιακή μορφή γίνεται:

$$u(k) = P(k) + I(k) \tag{7.11}$$

Αντικαθιστώντας τις σχέσεις (7.5) και (7.10) στην (7.11)

$$u(k) = K_p \cdot e(k) + I(k-1) + T_S \cdot K_i \cdot e(k) \tag{7.12}$$

Για την απαλοιφή του όρου *I(k-1)*, ο οποίος δε μπορεί να αναπαρασταθεί κατά την υλοποίηση του αλγορίθμου, θα πραγματοποιούν τα ακόλουθα βήματα. Αρχικά θα λύσουμε τη σχέση (7.10) ως προς *I(k-1)*, έτσι:

$$I(k-1) = I(k) - T_S \cdot K_i \cdot e(k) \tag{7.13}$$

Στη συνέχεια θα γράψουμε τη σχέση (7.11) για *k-1* χρησιμοποιώντας τη σχέση (7.13), έτσι:

$$u(k-1) = P(k-1) + I(k-1) \Rightarrow$$

$$u(k-1) = K_p \cdot e(k-1) + I(k) - T_S \cdot K_i \cdot e(k) \tag{7.14}$$

Λύνουμε τη σχέση (7.11) ως προς *I(k)* και αντικαθιστούμε την έκφραση για το *P(k)* με τη σχέση (7.5), επομένως:

$$I(k) = u(k) - K_p \cdot e(k) \tag{7.15}$$

Αντικαθιστώντας τη σχέση (7.15) στην (7.14) έχουμε:

$$u(k-1) = K_p \cdot e(k-1) + u(k) - K_p \cdot e(k) - T_S \cdot K_i \cdot e(k)$$

Άρα, λύνοντας ως προς *u(k)* προκύπτει η έκφραση του αλγορίθμου για την υλοποίηση του ελεγκτή PI σε ψηφιακό μικροεπεξεργαστή:

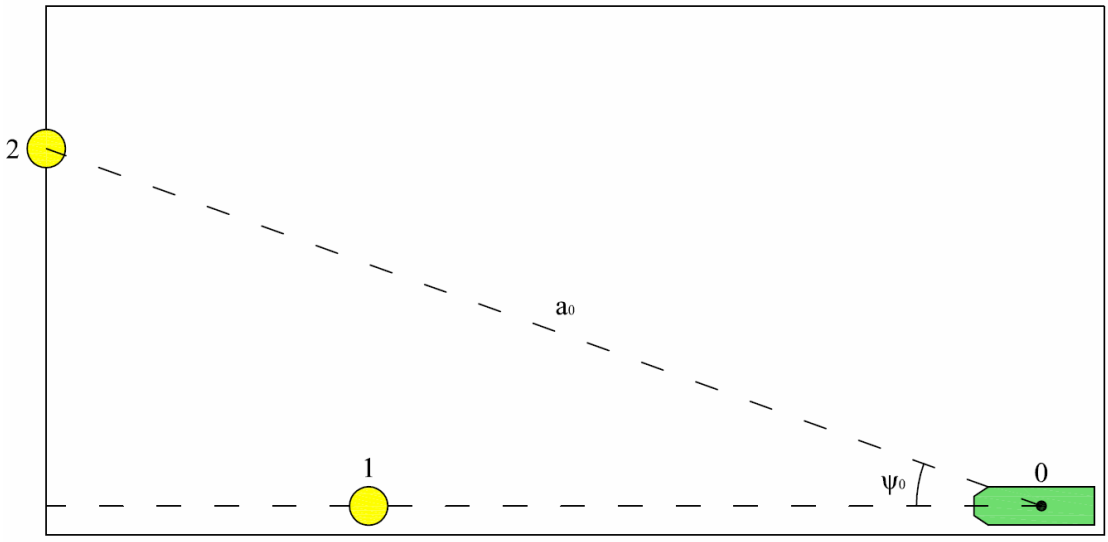
$$u(k) = u(k-1) + K_p \cdot e(k) - K_p \cdot e(k-1) + T_S \cdot K_i \cdot e(k) \tag{7.16}$$

<u>Σχόλιο:</u> *Περισσότερες πληροφορίες σχετικά με τα ψηφιακά συστήματα και μεθοδολογίες υπάρχουν στη βιβλιογραφία (Ξηρός 2005).*

Ο παραπάνω αλγόριθμος καθώς και όλη η επικοινωνία του μικροελεγκτή με τον αισθητήρα εικόνας, τους ενεργοποιητές και το σταθμό ξηράς υλοποιείται σε γλώσσα προγραμματισμού C. Στο Παράρτημα Β υπάρχει περιγραφή των συναρτήσεων και της δομής του κώδικα ενώ στο Παράρτημα Γ παρατίθεται ο πλήρης κώδικας.

7.3 Πείραμα αξιολόγησης

Παρακάτω περιγράφεται αναλυτικά το πείραμα που έγινε με το πραγματικό μοντέλο σε φυσικό περιβάλλον. Το φυσικό περιβάλλον είναι η πισίνα διαστάσεων 11 x 5,5 m η οποία περιέχει γλυκό νερό (Σχ. 7.5).



Σχήμα 7.5. *Πραγματικό πείραμα.*

➢ **Τιμές αναφοράς**

Θεωρούμε τιμή αναφοράς ως προς την απόσταση τα 7,2 m, δηλαδή *a_set = 7,2 m* και ως προς τη γωνία τα 0 rad, δηλαδή *ψ_set = 0 rad*. Σύμφωνα με το σχήμα 7.5 τα set points αντιστοιχούν στις θέσεις:

- Θέση πλοίου → 0.
- Θέση φάρου → 1.

Αυτό σημαίνει ότι αν δώσουμε εντολή στο πλοίο να εκτελέσει την παρακολούθηση στόχου υπό αυτές τις Α.Σ. δε θα γίνει τίποτα γιατί ήδη βρίσκεται στην επιθυμητή θέση. Επειδή ο αισθητήρας όπως έχουμε αναφέρει δε μετράει απόσταση και γωνία, αλλά δύο άλλα μεγέθη (*pn* και *tp*), θα μετατρέψουμε τα set points στα μετρούμενα μεγέθη σύμφωνα με τις εξισώσεις μετασχηματισμού του Κεφαλαίου 5.

Έτσι, σύμφωνα με τη σχέση (5.12) για την απόσταση έχουμε:

$$pn\_set = -c_3 \cdot a\_set \cdot \cos\psi\_set + 255 \Rightarrow$$

$$pn\_set = -24{,}3 \cdot 7{,}2 \cdot \cos 0 + 255 \Rightarrow$$

$$pn\_set = 80$$

Ενώ, για τη γωνία αναφερόμαστε στη σχέση (5.10) όπου έχουμε:

$$tp\_set = 192{,}6113 \cdot \psi\_set + 80$$

$$tp\_set = 192{,}6113 \cdot 0 + 80$$

$$tp\_set = 80$$

- **Αρχικές Συνθήκες (Α.Σ.)**

Οι Α.Σ. υπό τις οποίες θα πραγματοποιηθεί το πείραμα σύμφωνα με το σχήμα 7.5 αποφασίζεται να είναι:

- Θέση πλοίου → 0.
- Θέση φάρου → 2.

Σε αυτή την κατάσταση έχουμε:

- $a_0 = 11m$
- $\psi_0 = 0{,}337rad$

Αν χρησιμοποιήσουμε ξανά τις σχέσεις (5.10) και (5.12) έχουμε:

$$pn_0 = -c_3 \cdot a_0 \cdot \cos\psi_0 + 255 \Rightarrow$$

$$pn_0 = -24{,}3 \cdot 11 \cdot \cos 0{,}337 + 255 \Rightarrow$$

$$pn_0 = 3$$

Παρατηρούμε ότι $pn_0 \neq pn\_set$. Αυτό σημαίνει ότι αν ζητήσουμε από το σκάφος να παρακολουθήσει το στόχο, τότε θα ενεργοποιηθεί ο PI controller του thruster.

Επίσης:

$$tp_0 = 192{,}6113 \cdot \psi_0 + 80$$

$$tp_0 = 192{,}6113 \cdot 0.337 + 80$$

$$tp_0 = 145$$

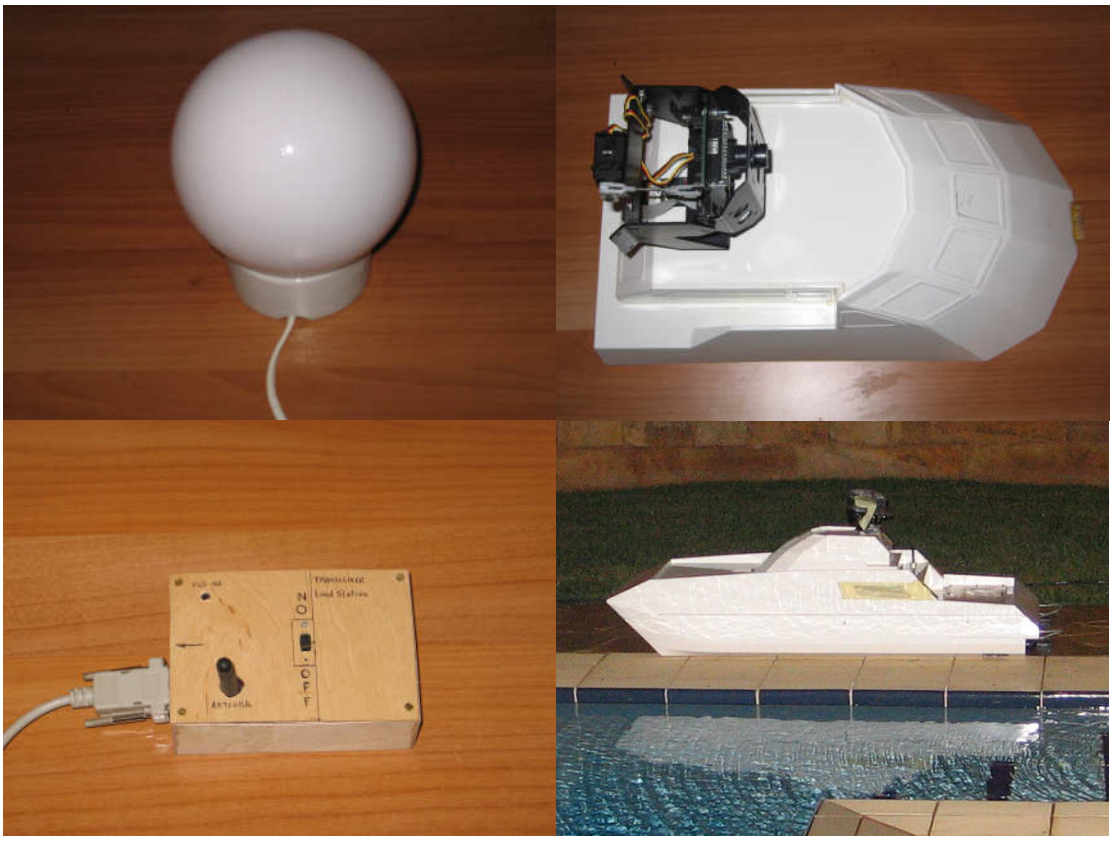
Παρατηρούμε ότι $tp_0 \neq tp\_set$. Αυτό σημαίνει ότι αν ζητήσουμε από το σκάφος να παρακολουθήσει το στόχο, τότε θα ενεργοποιηθεί ο PI controller του rudder.

➢ **Διεξαγωγή πειράματος**

Κατά την πειραματική διαδικασία η οποία διεξήχθη σε σκοτεινό περιβάλλον (νύχτα) ακολουθούνται τα παρακάτω βήματα:

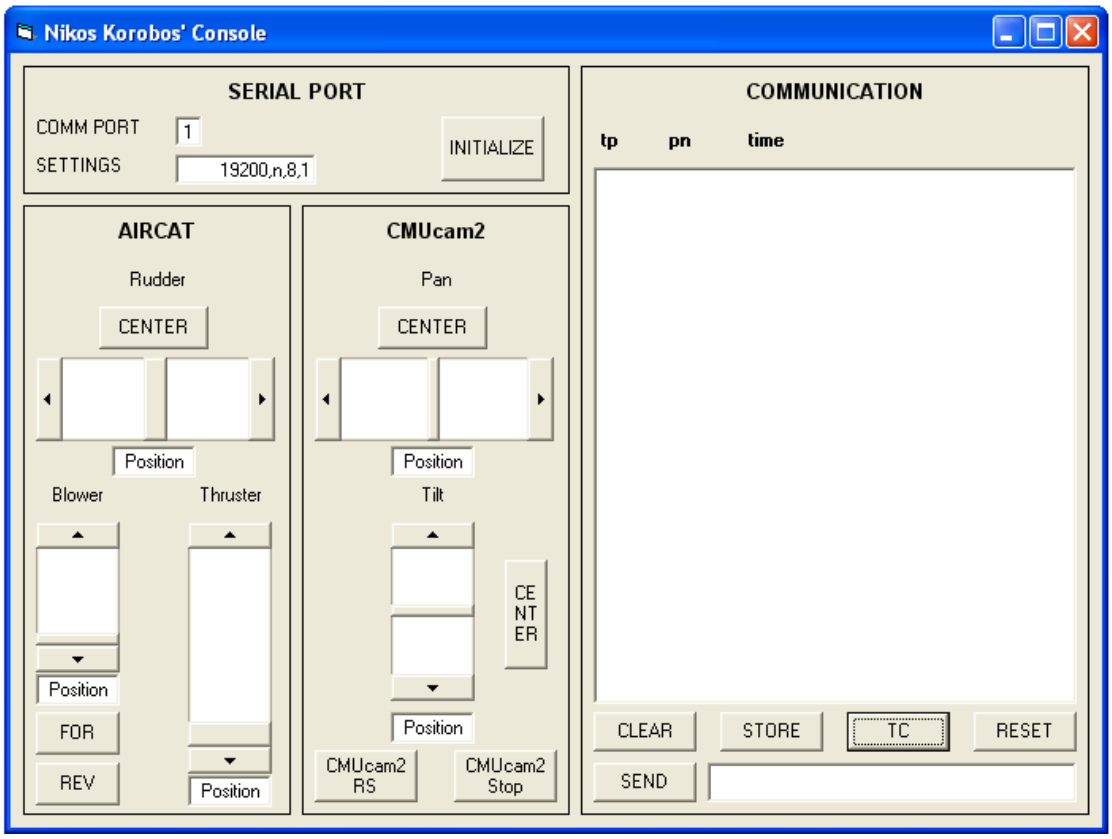
- Τοποθετείται ο φάρος στη θέση 2 (Σχ. 7.5).
- Τοποθετείται η κάμερα στο ΚΒ του πλοίου κατά τον οριζόντιο άξονα.
- Φορτώνεται ο κώδικας στον microcontroller (βλέπε Παράρτημα Β).
- Συνδέονται οι 2 πομποδέκτες, ένας στο σκάφος και ένας σε Η/Υ (σταθμός ξηράς).
- Γίνεται εκκίνηση της κονσόλας ελέγχου στον Η/Υ.
- Ξεκινάει να εκτελείται ο κώδικας στον microcontroller του σκάφους (αυτή τη στιγμή υπάρχει ασύρματη επικοινωνία μεταξύ σκάφους και σταθμού ξηράς).
- Τοποθετείται το σκάφος στη θέση 0 (Σχ. 7.5).
- Μέσω της κονσόλας ελέγχου δίνεται ασύρματα εντολή στο σκάφος να παρακολουθήσει το φάρο.
- Το σκάφος στέλνει ασύρματα στο σταθμό ξηράς δεδομένα σε σχέση με το οπτικό πεδίο της κάμερας (μεταβλητές *pn(t)* και *tp(t)*) ενώ ξεκινάει η **αυτόνομη** κίνησή του για παρακολούθηση του φάρου.
- Μόλις ολοκληρωθεί η κίνηση του σκάφους, τα δεδομένα τα οποία έχουν ληφθεί ασύρματα στο σταθμό ξηράς, αποθηκεύονται σε λογιστικά φύλλα.

Παρακάτω δίνονται κάποιες εικόνες για πληρέστερη κατανόηση των ανωτέρω (Σχ. 7.6).



Σχήμα 7.6. *Ο φάρος, η κάμερα τοποθετημένη στην υπερκατασκευή, ο πομποδέκτης του σταθμού ξηράς και το σκάφος με τοποθετημένη την κάμερα δίπλα από την πισίνα.*

Στη συνέχεια παρουσιάζεται η κονσόλα ελέγχου (Σχ. 7.7) που χρησιμοποιείται στο σταθμό ξηράς για συλλογή δεδομένων κτλ. Έχει δημιουργηθεί σε περιβάλλον Visual Basic 6 με σκοπό να εξυπηρετεί τις ανάγκες των πειραμάτων μας. Στο Παράρτημα Β υπάρχει περιγραφή των συναρτήσεων και της δομής του κώδικα ενώ στο Παράρτημα Δ παρατίθεται ο πλήρης κώδικας.



Σχήμα 7.7. *Η κονσόλα ελέγχου.*

Η κονσόλα ελέγχου όπως φαίνεται στο σχήμα 7.7 χωρίζεται σε 4 περιοχές:

- SERIAL PORT
- AIRCAT
- CMUcam2
- COMMUNICATION

Η περιοχή SERIAL PORT αφορά τις ρυθμίσεις της σειριακής θύρας στην οποία είναι συνδεδεμένος ο πομποδέκτης του σταθμού ξηράς. Η περιοχή AIRCAT εξυπηρετεί τον τηλεχειρισμό του σκάφους σε περίπτωση διαφόρων δοκιμών. Η περιοχή CMUcam2 ελέγχει τα servo της βάσης της κάμερας καθώς επίσης υπάρχει και η δυνατότητα διακοπής της ροής δεδομένων από την κάμερα προς τον microcontroller με το πλήκτρο CMUcam2_Stop. Τέλος, η περιοχή που κυρίως μας ενδιαφέρει στο συγκεκριμένο πείραμα είναι η COMMUNICATION. Έχουμε τη δυνατότητα να λάβουμε δεδομένα από το σκάφος (*τηλεμετρία*) καθώς και να δώσουμε σε αυτό την εντολή να παρακολουθήσει το φάρο μέσο του πλήκτρου TC (*Track Color*). Όταν ολοκληρωθεί η διαδικασία της κίνησης του σκάφους προς το φάρο, τα δεδομένα (*tp, pn , time*) αποθηκεύονται σε λογιστικά φύλλα με τη βοήθεια του πλήκτρου STORE.

8. ΑΠΟΤΕΛΕΣΜΑΤΑ - ΠΡΟΤΑΣΕΙΣ

8.1 Αποτελέσματα πειράματος αξιολόγησης

8.1.1 Προβλήματα διεξαγωγής πειράματος

Δεδομένου ότι στα εργαστήρια του Τομέα Ναυτικής Μηχανολογίας δεν υπήρχε τμήμα το οποίο ασχολείται με ρομποτική, ηλεκτρονικά, αισθητήρες κτλ, έπρεπε η εργαστηριακή εγκατάσταση να δημιουργηθεί από την αρχή. Έτσι, δαπανήθηκε πολύ μεγάλος χρόνος στο να αποκτηθούν τα διάφορα νέα υλικά, να εξοικειωθούμε με αυτά και να βαθμονομηθούν ανάλογα. Επίσης, για τη χρήση, ρύθμιση, λειτουργία και προγραμματισμό των ρομποτικών και ηλεκτρονικών αυτών συστημάτων ήταν απαραίτητη η γνώση πεδίων της επιστήμης πέραν του Ναυπηγού. Έτσι, το χρονικό διάστημα εκμάθησης και πρακτικής εφαρμογής όλων των ανωτέρω δυσχέρανε ακόμα περισσότερο το έργο μας. Μια επιπλέον δυσκολία ήταν τα χρήματα τα οποία δεν ήταν εύκολο να εξασφαλίσουμε από τη σχολή, με αποτέλεσμα σχεδόν όλος ο εξοπλισμός να αποκτηθεί με δικά μας κεφάλαια.

Κατά την διεξαγωγή του πειράματος παρουσιάστηκαν διάφορα προβλήματα τα οποία κυρίως οφείλονται στο γεγονός ότι δεν υπήρχε πανεπιστημιακού επιπέδου εργαστηριακή εγκατάσταση στο χώρο του πειράματος. Έτσι, δεν υπήρχε η δυνατότητα να ελέγχονται οι περισσότερες παράμετροι με ακρίβεια. Πιο συγκεκριμένα:

- Η πισίνα η οποία βρισκόταν σε υπαίθριο χώρο επηρεαζόταν από διάφορους εξωτερικούς φωτισμούς.
- Ο φάρος έπρεπε να είναι πολύ μεγαλύτερου μεγέθους από ότι είχαμε προβλέψει, έτσι αναγκαστήκαμε να προσθέσουμε ανακλαστήρες στον υπάρχοντα φάρο.
- Οι ανακλαστήρες δεν ήταν απόλυτα ελεγχόμενοι με αποτέλεσμα να στέλνουν μέρος του φωτός στην επιφάνεια της πισίνας και αυτό να αντανακλά και να επηρεάζει σε μεγάλο βαθμό την κάμερα.
- Η υπόθεση ομοιοτροπικού φωτός (Σχ. 5.14) δεν ίσχυε πια, δεδομένου ότι οι ανακλαστήρες δεν έστελναν σφαιρικά τις ακτίνες φωτός προς όλες τις κατευθύνσεις.
- Οι διαστάσεις τις πισίνας ήταν τέτοιες, που εύκολα ο δημιουργούμενος από το σκάφος κυματισμός ανακλώνταν στις πλευρές της και επέστρεφε προς το πλοίο, με αποτέλεσμα να επηρεάζει την κίνησή του.

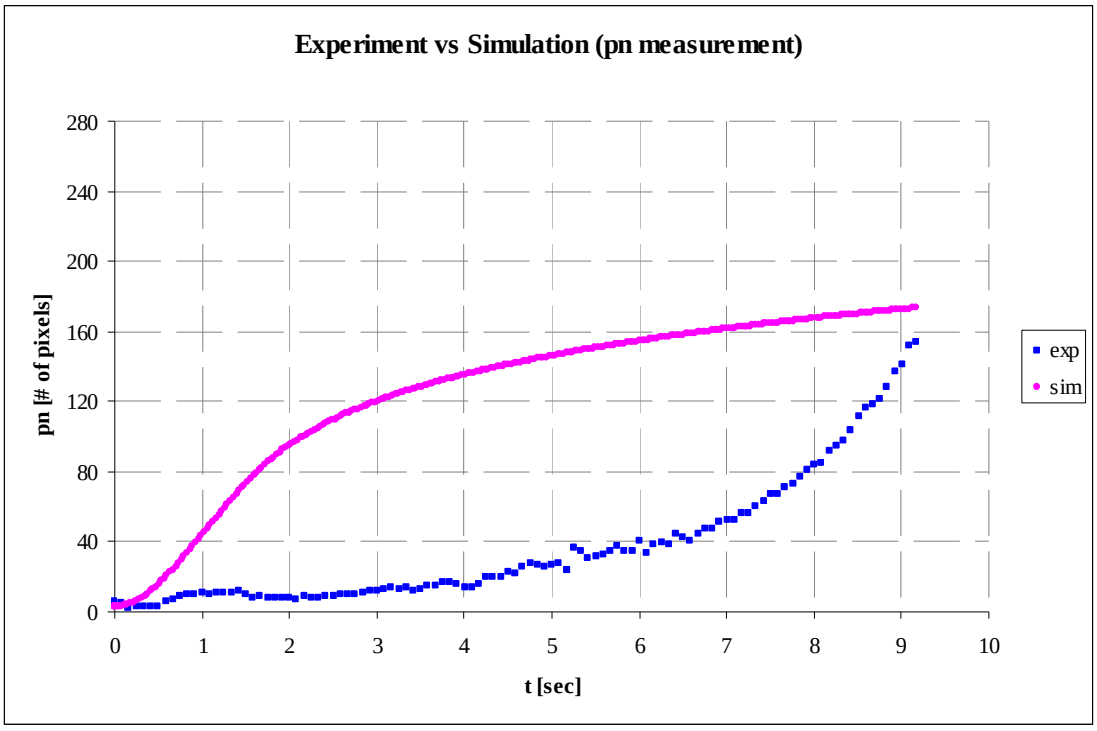
8.1.2 Παρουσίαση αποτελεσμάτων

Παρόλα τα προβλήματα καταφέραμε να πραγματοποιήσουμε το πείραμα και να πάρουμε μετρήσεις. Αυτό είναι πολύ σημαντικό δεδομένου ότι δεν είχαμε εύκολη πρόσβαση σε πισίνα και το ότι πήραμε ενδείξεις μέσα σε λίγες μόνο ώρες είναι αξιοσημείωτο. Παρακάτω παρουσιάζονται οι μετρήσεις τόσο για το εικονικό όσο και για το πραγματικό πείραμα για ένα αντιπροσωπευτικό σετ αρχικών συνήθων (Α.Σ.).

Οι αρχικές συνθήκες για το πείραμα ορίζονται:

- $a = 11m$
- $a' = 0m/s$
- $\psi = 0{,}337rad$

Σύμφωνα με τις παραπάνω Α.Σ. παρουσιάζονται τα αποτελέσματα του εικονικού και του πραγματικού πειράματος συγκρίνοντας την ένδειξη για το οπτικό πεδίο της κάμερας. Έτσι, παρακάτω δίνεται η καμπύλη *pn(t)* (Σχ. 8.1).

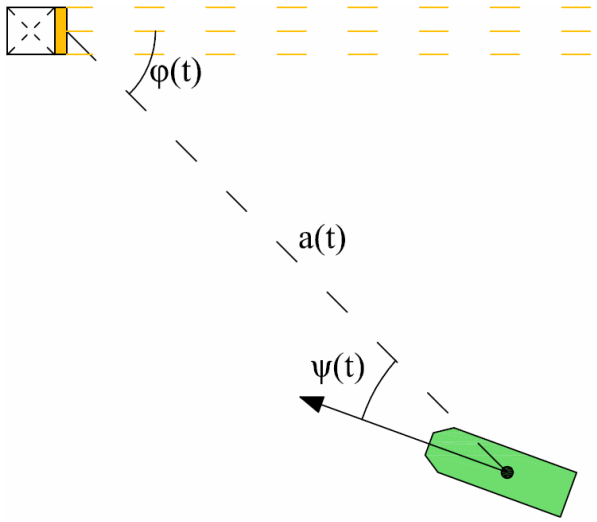


Σχήμα 8.1. *Σύγκριση αποτελεσμάτων πειράματος για τη μεταβλητή pn.*

Η καμπύλη της προσομοίωσης (ροζ) παρατηρείται κοίλη, ενώ αντίθετα η καμπύλη του πραγματικού πειράματος (μπλε) είναι κυρτή. Η διαφορά αυτή εντοπίζεται κυρίως στους παρακάτω λόγους.

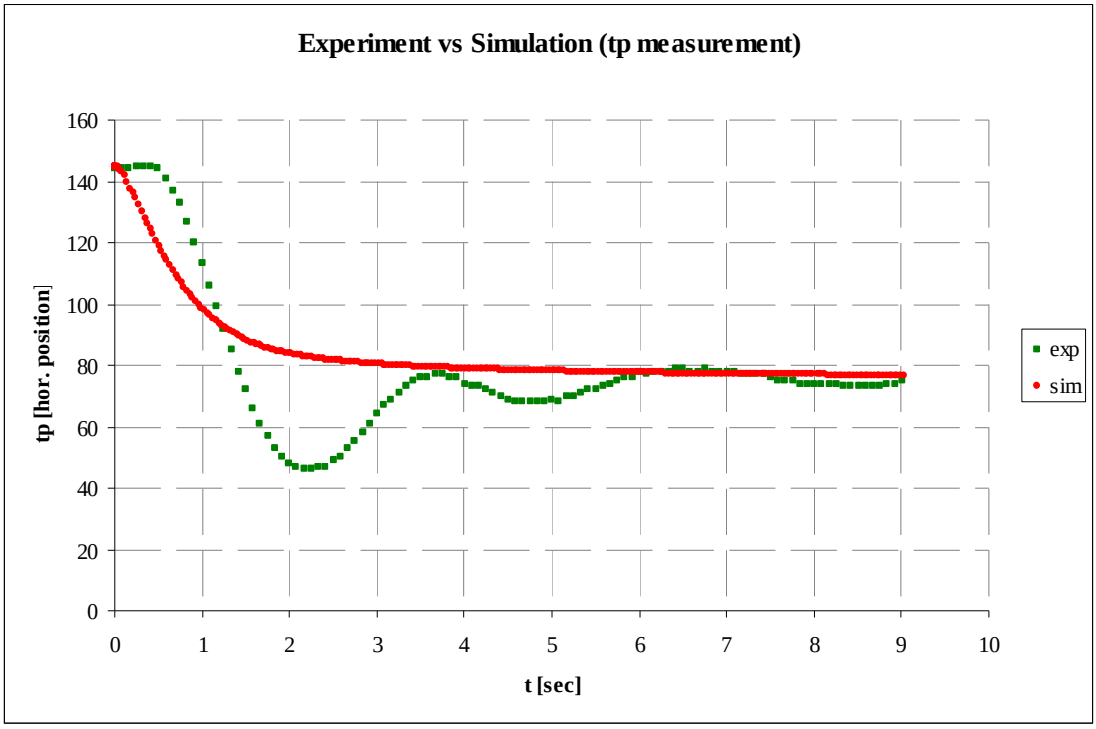
Το φως δεν ήταν ομοιοτροπικό αλλά είχε κάποιου είδους κατευθυντικότητα (Σχ 8.2). Επιπλέον γινόταν αντανάκλασή του στην επιφάνεια του νερού. Αυτό είχε σαν αποτέλεσμα η κάμερα να «βλέπει» περισσότερα από τα αναμενόμενα pixels καθώς πλησίαζε προς το φάρο. Έτσι, η καμπύλη του πραγματικού πειράματος δεν εκφράζει απόλυτα τη γραμμική σχέση (5.12) μεταξύ απόστασης *a* και αριθμού pixels *pn*.

Επίσης, η υπόθεση που έγινε στη μαθηματική μοντελοποίηση για την ώση, δεν αποτελεί ίσως, το πιο ακριβές μοντέλο. Έτσι, ούτε η καμπύλη της προσομοίωσης αποτυπώνει με ακρίβεια την μεταβολή του αριθμού των pixels.



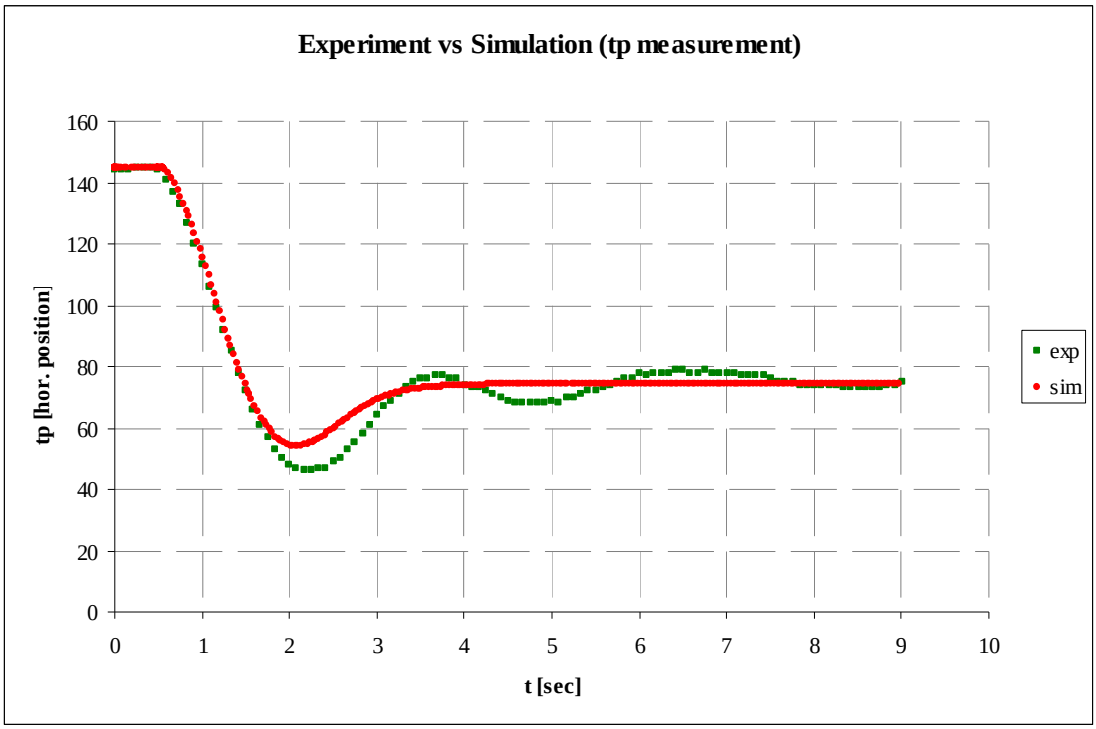
Σχήμα 8.2. *Κατευθυνόμενο φως.*

Στη συνέχεια παρουσιάζεται η σύγκριση μεταξύ εικονικού και πραγματικού πειράματος για τη μεταβλητή *tp(t)* (Σχ. 8.3).



Σχήμα 8.3. *Σύγκριση αποτελεσμάτων πειράματος για τη μεταβλητή tp.*

Η καμπύλη της προσομοίωσης (κόκκινη) παρατηρείται κυρτή, ενώ η καμπύλη του πραγματικού πειράματος (πράσινη) πραγματοποιεί ταλάντωση κοντά στο set point *tp_set = 80*. Εισάγοντας μια χρονική καθυστέρηση στην απόκριση του πλοίου της τάξης του 0,5 sec στην προσομοίωση έχουμε την παρακάτω αλλαγή (Σχ. 8.4).



Σχήμα 8.4. *Εισαγωγή χρονικής καθυστέρησης 0,5 sec.*

Παρατηρούμε ιδιαίτερη βελτίωση της καμπύλης της προσομοίωσης πλησιάζοντας πολύ την πραγματική συμπεριφορά της μεταβλητής *tp*. Αυτή την καθυστέρηση στη απόκριση του πλοίου δε μπορούσαμε να την προβλέψουμε με βάση τα αρχικά πειράματα που προηγήθηκαν της ταυτοποίησης (βλέπε Κεφάλαιο 6).

Τέλος, για να γίνουν πιο κατανοητά τα ανωτέρω μετατρέπουμε τις γραφικές παραστάσεις από *pn(t)* σε *a(t)* (Σχ. 8.5) και από *tp(t)* σε *ψ(t)* (Σχ. 8.6) μέσω των (5.10) και (5.12). Έτσι:

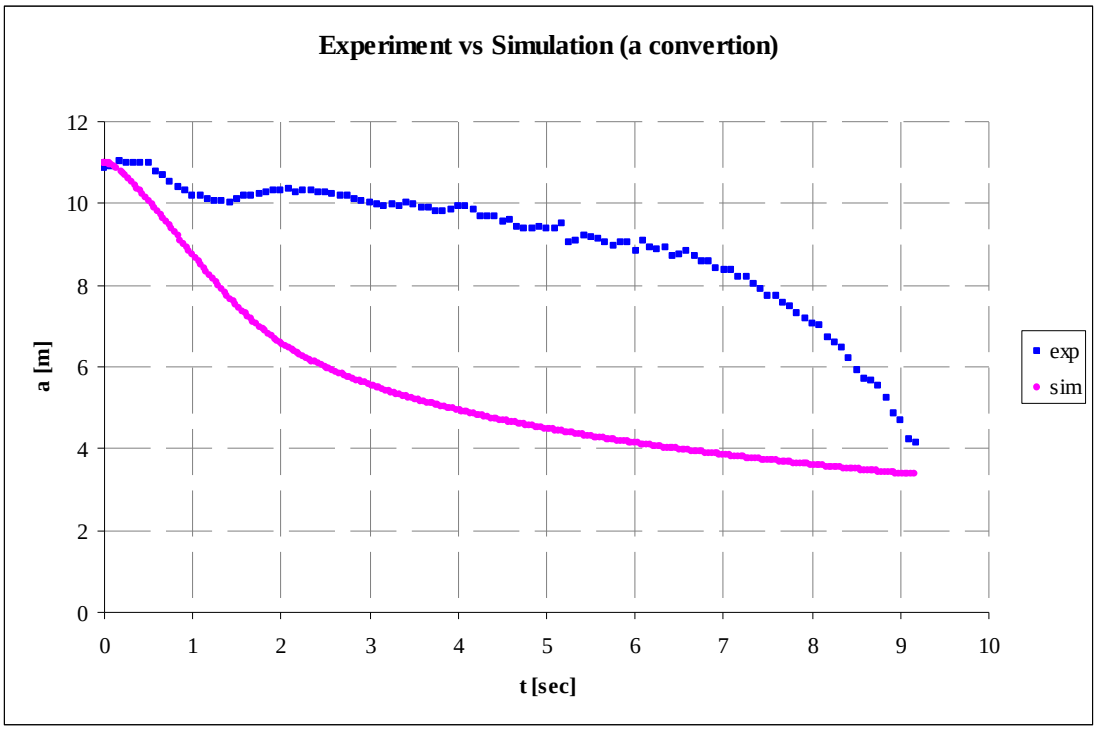
$$pn(t) = -24{,}3 \cdot a(t) \cdot \cos\psi(t) + 255 \Rightarrow$$

$$a(t) = -\frac{pn(t) - 255}{24{,}3 \cdot \cos\psi(t)} \qquad (8.1)$$

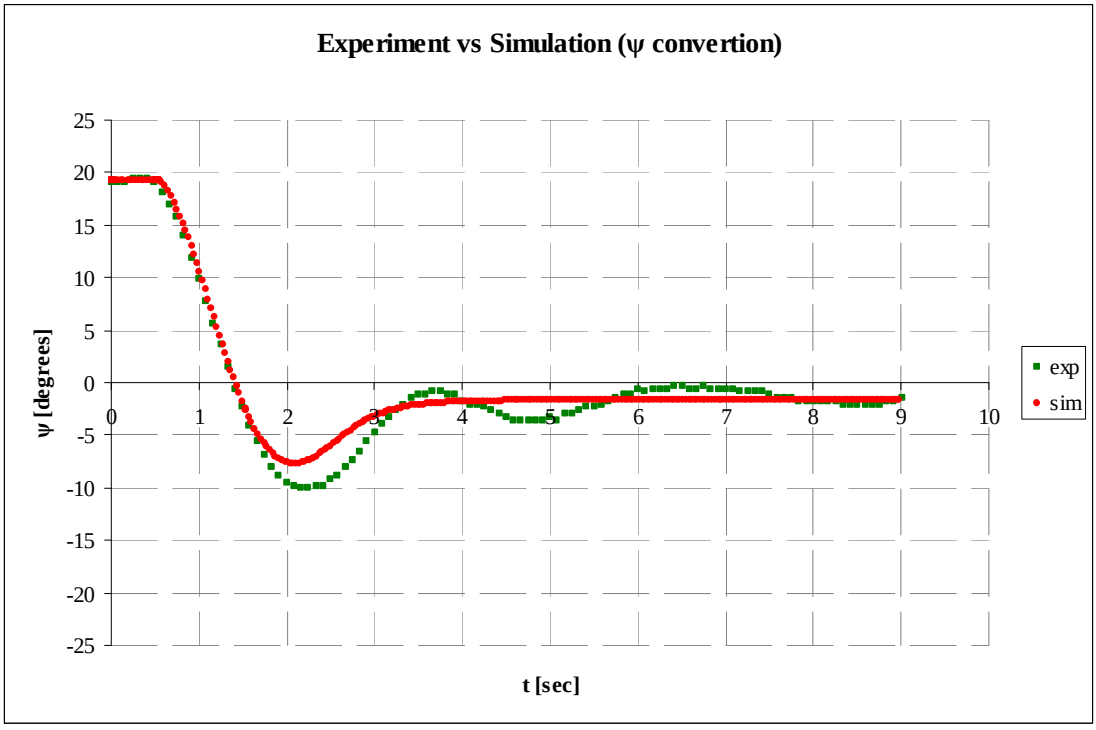
$$tp(t) = 192{,}6113 \cdot \psi(t) + 80 \Rightarrow$$

$$\psi(t) = \frac{tp(t) - 80}{192{,}6113} \qquad (8.2)$$

Σύμφωνα με τους μετασχηματισμούς των σχέσεων (8.1) και (8.2) προκύπτουν οι ακόλουθες γραφικές παραστάσεις, οι οποίες ποιοτικά γίνονται πιο εύκολα αντιληπτές.



Σχήμα 8.5. *Σύγκριση αποτελεσμάτων πειράματος για την απόσταση a του πλοίου από το φάρο.*



Σχήμα 8.6. *Σύγκριση αποτελεσμάτων πειράματος για την γωνία ψ που σχηματίζει η ταχύτητα με την ευθεία που συνδέει το σκάφος με το φάρο.*

8.2 Προτάσεις για περαιτέρω εξέλιξη

Όπως φάνηκε από τα παραπάνω γραφήματα, υπάρχει μια απόκλιση μεταξύ προσομοιώσεων και πραγματικού πειράματος κυρίως στη μεταβλητή *pn* (άρα και *a*). Οι διάφοροι λόγοι αναφέρθηκαν ανωτέρω.

Για να λυθεί το πρόβλημα αυτό αρχικά προτείνεται η δημιουργία ακριβέστερου μοντέλου για την δύναμη ώσης (βλέπε Παράρτημα Α).

Επίσης είναι απαραίτητη η εισαγωγή μιας επιπλέον διαφορικής εξίσωσης, η οποία θα συμπλήρωνε το μοντέλο των δ.ε. (6.1). Αυτή η δ.ε. θα εισάγει το ρόλο της γωνίας φ, μεταξύ του κατευθυνόμενου φωτός και της ευθείας που συνδέει το σκάφος με το φάρο (Σχ. 8.2), στο υπάρχον μαθηματικό μοντέλο.

Σε πρώτη προσέγγιση προτείνεται διεξαγωγή πειραμάτων βαθμονόμησης της κάμερας για τη μεταβλητή *pn* σε διάφορες συνθήκες και για ποικιλία αποστάσεων. Με αυτό τον τρόπο θα μπορέσουμε να εξάγουμε ίσως μια καλύτερη από την (5.12) σχέση μεταξύ *pn* και *a*. Εισάγοντας την νέα αυτή σχέση στο μετασχηματισμό της κάμερας σε συνδυασμό με το νέο μαθηματικό μοντέλο και πραγματοποιώντας νέα προσομοίωση αναμένουμε η ροζ καμπύλη του σχήματος 8.1 να πλησιάσει αρκετά την μπλε.

Επιπλέον είναι αναγκαία νέα πειράματα με το σκάφος σε περισσότερα σημεία λειτουργίας των προωστήρων σε συνδυασμό με δεδομένα από την κάμερα. Με τον τρόπο αυτό η ταυτοποίηση του συστήματος θα δώσει εξισώσεις κίνησης του πλοίου οι οποίες θα ανταποκρίνονται ακόμα περισσότερο στην πραγματικότητα.

Επίσης, προτείνεται η χρήση διαφόρων αισθητήρων και διεξαγωγή πειραμάτων λαμβάνοντας δεδομένα από αυτούς. Έτσι, αρχικά θα μπορέσουμε να βελτιώσουμε την ανάλυση των 2 βαθμών ελευθερίας ενώ μακροπρόθεσμα θα μπορέσουμε να επεκτείνουμε τη μαθηματική μοντελοποίηση σε περισσότερους β.ε. Οι αισθητήρες που προτείνονται είναι:

- Πυξίδα (*Compass*): Έλεγχος της απόλυτης γωνίας του σκάφους κατά τον κατακόρυφο άξονα.
- Ηχοβολιστικό (*Sonar*): Έλεγχος της απόστασης του σκάφους από αντικείμενα.
- Συσκευή μέτρησης αδράνειας (*Inertial Measurement Unit – IMU*): Έλεγχος των γραμμικών επιταχύνσεων και περιστροφικών κινήσεων του σκάφους.
- Αμπερόμετρο (*Current sensor*): Έλεγχος της έντασης του ηλεκτρικού ρεύματος σε διάφορους καταναλωτές (κινητήρες, σερβομηχανισμοί, κτλ)
- Βολτόμετρο (*Voltage sensor*): Έλεγχος της τάσης του ηλεκτρικού ρεύματος σε διάφορους καταναλωτές (κινητήρες, σερβομηχανισμοί, κτλ)
- Στροφόμετρο (*RPM sensor*): Έλεγχος της ταχύτητας περιστροφής των αξόνων των waterjet και του blower.

Τέλος, για την ολοκλήρωση της πλατφόρμας ελέγχου η οποία περιέχει μικροελεγκτή, αισθητήρες, ενεργοποιητές κτλ, προτείνεται η ανάπτυξη λειτουργικού συστήματος (*FreeRTOS*) για τον μικροελεγκτή. Έτσι, θα μπορούμε να εκτελούμε τα διάφορα σενάρια δοκιμών με εύκολο τρόπο.

ΠΑΡΑΡΤΗΜΑ

Α. ΑΠΟΔΕΙΞΗ ΣΧΕΣΗΣ (5.2)

Στόχος μας είναι να αποδείξουμε τη σχέση (5.2) όπου έχει βασισθεί πάνω της όλη η μαθηματική μοντελοποίηση του συστήματός μας. Για να μην ανατρέχουμε στο κεφάλαιο 5 παρατίθεται στη συνέχεια (σχέση (Α.1)).

$$v'(t) = \frac{c_{1m}}{v(t)} \cdot u_1(t) - c_{1V} \cdot |v(t)| \cdot v(t), \quad \text{για } T(t) < T_{max}$$

$$v'(t) = \frac{T_{\max}}{m_S} - c_{1V} \cdot |v(t)| \cdot v(t), \quad \text{για } T(t) > T_{max} \qquad \text{(Α.1)}$$

Α.1 Απόδειξη

Ξεκινώντας από τον 1ο Νόμο του Νεύτωνα έχουμε την ισορροπία δυνάμεων στο σκάφος:

$$\Sigma F(t) = T(t) - R(t) \qquad \text{(Α.2)}$$

Όπου:

- *ΣF(t)*: Συνισταμένη δυνάμεων που ασκούνται στο σκάφος.
- *T(t)*: Δύναμη ώσης.
- *R(t)*: Δύναμη αντίστασης.

Α.1.1 Συνισταμένη δυνάμεων ΣF

Σύμφωνα με το 2ο Νόμο του Νεύτωνα ισχύει:

$$\Sigma F(t) = m_S \cdot v'(t) \qquad \text{(Α.3)}$$

Όπου:

- m_S: Μάζα πλοίου (αμετάβλητη).
- *v(t)*: Ταχύτητα πλοίου.
- *v'(t)*: Επιτάχυνση πλοίου.

Α.1.2 Δύναμη ώσης Τ

Για την ισχύ προώσεως έχουμε:

$$P(t) = T(t) \cdot v(t) \tag{A.4}$$

Επίσης, υποθέτουμε μια σχέση που συνδέει την ισχύ προώσεως με την εισερχόμενη ισχύ. Βάση υπόθεσης έχουμε:

$$P(t) = \eta \cdot P_0 \cdot u_1(t) \tag{A.5}$$

Όπου:

- η: Βαθμός απόδοσης ηλεκτροκινητήρα.
- P_0: Μέγιστη εισερχόμενη ισχύς ηλεκτροκινητήρα.
- $u_1(t)$: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης ESC κινητήρων.

Εξισώνοντας τις (Α.4) και (Α.5) και λύνοντας ως προς $T(t)$, προκύπτει:

$$T(t) \cdot v(t) = \eta \cdot P_0 \cdot u_1(t) \Rightarrow$$

$$T(t) = \begin{cases} \dfrac{\eta \cdot P_0}{v(t)} \cdot u_1(t), & για \quad T(t) < T_{\max} \\ \\ T_{\max}, & για \quad T(t) > T_{\max} \end{cases} \tag{A.6}$$

<u>Σχόλιο:</u> *Η σχέση (Α.6) αποκτά 2 κλάδους γιατί όπως παρατηρούμε η ώση για πολύ μικρές τιμές της ταχύτητας τείνει ασυμπτωτικά στο άπειρο. Έτσι, όταν ξεπεράσει ένα όριο (T_{max}) θεωρούμε ότι δε μπορεί να αυξηθεί άλλο και αποκτά αυτή τη σταθερή τιμή η οποία είναι χαρακτηριστική για το κάθε σύστημα. Η μοντελοποίηση (Α.6) η οποία αφορά τη δύναμη ώσης στηρίζεται σε υποθέσεις, άρα δεν αποτελεί αποδεδειγμένη έκφραση για αυτήν. Ωστόσο στα πλαίσια της παρούσας εργασίας όπου δε μας ενδιαφέρει να ορίσουμε το τέλειο μαθηματικό μοντέλο, δεχόμαστε αυτή την ανάλυση.*

Θέτοντας:

$$C_T(v(t)) \equiv \frac{\eta \cdot P_0}{v(t)} \tag{A.7}$$

Η σχέση (Α.6) γίνεται:

$$T(t) = \begin{cases} C_T(v(t)) \cdot u_1(t), & για \quad T(t) < T_{\max} \\ \\ T_{\max}, & για \quad T(t) > T_{\max} \end{cases} \tag{A.8}$$

Α.1.3 Δύναμη αντίστασης R

Για την αντίσταση του πλοίου όταν αυτό κινείται ευθύγραμμα χωρίς εκτροπή των πηδαλίων του ισχύει (Ιωαννίδης, 2005):

$$R_H(t) = C_{RH} \cdot v^2(t) \qquad \text{(A.9)}$$

Προκειμένου να έχουμε κάποια ένδειξη για την κατεύθυνση του σκάφους γράφουμε την παραπάνω εξίσωση ως εξής:

$$R_H(t) = C_{RH} \cdot |v(t)| \cdot v(t) \qquad \text{(A.10)}$$

Η εκτροπή του πηδαλίου προσθέτει μια αντίσταση:

$$R_R(t) = C_{RR1} \cdot \delta(t) \cdot v^2(t) \qquad \text{(A.11)}$$

$$\delta(t) = C_{RR2} \cdot u_2(t) \qquad \text{(A.12)}$$

Όπου:

- $C_{RH,RR1RR2}$: Σταθερές αναλογίας.
- $\delta(t)$: Γωνία εκτροπής πηδαλίου.
- $u_2(t)$: Κανονικοποιημένος λόγος εύρους παλμού οδήγησης servo ακροφυσίου προωθητήρων.

$$R_R(t) = C_{RR1} \cdot C_{RR2} \cdot u_2(t) \cdot v^2(t) \Rightarrow$$

$$R_R(t) = C_{RR1} \cdot C_{RR2} \cdot u_2(t) \cdot |v(t)| \cdot v(t) \qquad \text{(A.13)}$$

Θέτοντας:

$$C_{RR} \equiv C_{RR1} \cdot C_{RR2} \qquad \text{(A.14)}$$

Η σχέση (A.13) γίνεται:

$$R_R(t) = C_{RR} \cdot u_2(t) \cdot |v(t)| \cdot v(t) \qquad \text{(A.14)}$$

Έτσι, αθροίζοντας τις σχέσεις (A.10) και (A.14) έχουμε:

$$R(t) = R_H(t) + R_R(t) \Rightarrow$$

$$R(t) = C_{RH} \cdot |v(t)| \cdot v(t) + C_{RR} \cdot u_2(t) \cdot |v(t)| \cdot v(t) \Rightarrow$$

$$R(t) = \left(C_{RH} + C_{RR} \cdot u_2(t)\right) \cdot |v(t)| \cdot v(t) \qquad \text{(A.15)}$$

Θέτοντας:

$$C_R(u_2(t)) \equiv C_{RH} + C_{RR} \cdot u_2(t) \tag{A.16}$$

Η σχέση (A.15) γίνεται:

$$R(t) = C_R(u_2(t)) \cdot |v(t)| \cdot v(t) \tag{A.17}$$

A.1.4 Γενική εξίσωση κίνησης

Αντικαθιστώντας τις σχέσεις (A.3), (A.8) και (A.17) στην (A.2), προκύπτει:

$$\begin{aligned} m_S \cdot v'(t) &= C_T(v(t)) \cdot u_1(t) - C_R(u_2(t)) \cdot |v(t)| \cdot v(t), && \text{για } T(t) < T_{max} \\ m_S \cdot v'(t) &= T_{\max} - C_R(u_2(t)) \cdot |v(t)| \cdot v(t), && \text{για } T(t) > T_{max} \end{aligned} \tag{A.18}$$

Διαιρώντας και τα δύο μέλη της παραπάνω εξίσωσης με m_S, έχουμε:

$$\begin{aligned} v'(t) &= \frac{C_T(v(t))}{m_S} \cdot u_1(t) - \frac{C_R(u_2(t))}{m_S} \cdot |v(t)| \cdot v(t), && \text{για } T(t) < T_{max} \\ v'(t) &= \frac{T_{\max}}{m_S} - \frac{C_R(u_2(t))}{m_S} \cdot |v(t)| \cdot v(t), && \text{για } T(t) > T_{max} \end{aligned} \tag{A.19}$$

Θέτοντας:

$$C_{1m}(v(t)) \equiv \frac{C_T(v(t))}{m_S} \tag{A.20}$$

$$C_{1V}(u_2(t)) \equiv \frac{C_R(u_2(t))}{m_S} \tag{A.21}$$

Καταλήγουμε στη γενική εξίσωση που περιγράφει την κίνηση του πλοίου στο επίπεδο λαμβάνοντας υπ' όψην μας 2 βαθμούς ελευθερίας.

$$\begin{aligned} v'(t) &= C_{1m}(v(t)) \cdot u_1(t) - C_{1V}(u_2(t)) \cdot |v(t)| \cdot v(t), && \text{για } T(t) < T_{max} \\ v'(t) &= \frac{T_{\max}}{m_S} - C_{1V}(u_2(t)) \cdot |v(t)| \cdot v(t), && \text{για } T(t) > T_{max} \end{aligned} \tag{A.22}$$

A.1.5 Απλοποίηση εξίσωσης κίνησης

Στη συνέχεια απλοποιούμε την ανωτέρω σχέση. Σύμφωνα με τα παραπάνω έχουμε:

$$C_{1m}(v(t)) = \frac{C_T(v(t))}{m_S} \Rightarrow$$

$$C_{1m}(v(t)) = \frac{\eta \cdot P_0}{m_S \cdot v(t)} \tag{A.23}$$

Έτσι, θέτοντας:

$$c_{1m} \equiv \frac{\eta \cdot P_0}{m_S} \tag{A.24}$$

Προκύπτει:

$$C_{1m}(v(t)) = \frac{c_{1m}}{v(t)} \tag{A.25}$$

Επίσης, εφόσον δεν επηρεάζει πολύ την ανάλυσή μας, θεωρούμε:

$$C_{1V}(u_2(t)) \cong c_{1V} \tag{A.26}$$

Έτσι, τελικά προκύπτει η ζητούμενη απλοποιημένη σχέση η οποία χρησιμοποιείται κατά τη μαθηματική μοντελοποίηση του υπό μελέτη συστήματος:

$$v'(t) = \frac{c_{1m}}{v(t)} \cdot u_1(t) - c_{1V} \cdot |v(t)| \cdot v(t), \quad \text{για } T(t) < T_{max}$$

$$v'(t) = \frac{T_{\max}}{m_S} - c_{1V} \cdot |v(t)| \cdot v(t), \quad \text{για } T(t) > T_{max} \tag{A.27}$$

Α.2 Ένθετο

Α.2.1 Υπολογισμός T_{max}

Από πειραματικά δεδομένα άλλων διπλωματικών καθώς και από χαρακτηριστικά του κατασκευαστή των waterjet έχουμε ότι:

$$T_{\max} \approx 22N$$

Αυτή η τιμή αναφέρεται σε 1 waterjet, δηλαδή στο πρόβλημά μας έχουμε $T_{\max} \approx 44N$. Το μέγεθος αυτό αντιστοιχεί σε στατική ώση, δηλαδή για μηδενική ταχύτητα πλοίου. Αυτό το γεγονός συμπληρώνει την ερμηνεία του διπλού κλάδου της σχέσης (Α.8) για την ώση.

Α.2.2 Υπολογισμός P_0

Για τον υπολογισμό του σταθερού (για την ανάλυσή μας) P_0, έχουμε:

$$P_0 = I \cdot e \qquad \text{(Α.28)}$$

Όπου:

- I: Ένταση ηλεκτρικού ρεύματος κινητήρα σε έμφορτη κατάσταση λειτουργίας.
- e: Τάση λειτουργίας ηλεκτροκινητήρα.

Από πειραματικά δεδομένα άλλων διπλωματικών καθώς και από προσωπική εμπειρία σε δοκιμές ηλεκτροκινητήρων γνωρίζουμε με αρκετά καλή προσέγγιση το επίπεδο της τάσης και έντασης σε έναν ηλεκτροκινητήρα σαν αυτόν της εφαρμογής μας. Έτσι, για τάση λειτουργίας $e \approx 12{,}5V$, έχουμε $I \approx 16A$.

Από τη σχέση (Α.28) σε συνδυασμό με τα παραπάνω, προκύπτει:

$$P_0 \approx 16 \cdot 12{,}5 \Rightarrow$$

$$P_0 \approx 200W$$

Α.2.3 Υπολογισμός m_S

Το m_S όπως γνωρίζουμε είναι η μάζα του σκάφους, δηλαδή το εκτόπισμά του στην κατάσταση των πειραμάτων μας. Σε αυτή την κατάσταση το πλοίο διαθέτει όλο το βασικό εξοπλισμό καθώς και τους αισθητήρες, τα ηλεκτρονικά μέσα και τις μπαταρίες. Ζυγίζοντας όλα τα παραπάνω με ζυγαριά ακριβείας έχουμε ότι:

$$m_S \approx 9{,}000kg$$

Α.2.4 Υπολογισμός η

Από τη διεξαγωγή προσομοιώσεων προέκυψε το c_{1m}. Όπως γνωρίζουμε:

$$c_{1m} = \frac{\eta \cdot P_0}{m_S}$$

Έτσι, λύνοντας την παραπάνω σχέση ως προς η προκύπτει το ζητούμενο:

$$\eta = \frac{c_{1m} \cdot m_S}{P_0} \Rightarrow$$

$$\eta \approx \frac{19 \cdot 9}{200} \Rightarrow$$

$$\eta \approx 0{,}86$$

Β. ΕΓΧΕΙΡΙΔΙΟ ΠΡΟΓΡΑΜΜΑΤΙΣΤΗ

Αφού αναλύσαμε στο κεφάλαιο 4 τα ηλεκτρονικά μέσα που χρησιμοποιούνται στο σύστημα, το παρόν εδάφιο ασχολείται με το απαραίτητο λογισμικό (*software*). Το λογισμικό είναι βασικό συστατικό των προγραμματιζόμενων ηλεκτρονικών συσκευών. Οι ηλεκτρονικές συσκευές του συστήματός μας οι οποίες προγραμματίζονται είναι οι εξής:

- Microcontroller.
- PC.

Για τον microcontroller παράγεται software το οποίο ελέγχει και συντονίζει τις λειτουργίες του σκάφους ενώ στο PC δημιουργείται ένας δίαυλος επικοινωνίας (*interface*) του χρήστη με το σύστημα. Έτσι, το κεφάλαιο χωρίζεται σε δύο κύριες παραγράφους οι οποίες σχετίζονται αντίστοιχα με τα δύο προαναφερθέντα πεδία.

Β.1 Προγραμματισμός microcontroller

Για να λειτουργήσει ο μικροελεγκτής, πρέπει να φορτωθεί στη μνήμη του (FLASH ή RAM) ένα αρχείο το οποίο περιέχει τον κώδικα. Αυτό το αρχείο είναι σε γλώσσα μηχανής (16ικό). Για τη δημιουργία του 16ικού αρχείου main.hex, χρησιμοποιείται το περιβάλλον WinARM με τον GCC compiler ενώ η επεξεργασία γίνεται στον editor Programmers Notepad 2 (*PN2*). Δηλαδή, αρχικά γράφεται το πρόγραμμα σε γλώσσα C και με τη βοήθεια του compiler δημιουργείται το 16ικό αρχείο. Έτσι, έχουμε τη δυνατότητα να το φορτώσουμε σε μορφή η οποία είναι μικρού μεγέθους.

Β.1.1 Οργάνωση Project File

Στη συνέχεια παρουσιάζονται αρχεία τα οποία είναι απαραίτητα για τη δημιουργία οποιουδήποτε προγράμματος με το περιβάλλον WinARM (Πνκς Β.1).

Πίνακας Β.1
Βασικά αρχεία ενός pn project.

Name	File type	Description
Auto_Pilot	Project file	Contains all files.
crt0.S	Startup assembler file	Includes the interrupt vectors and start-up code.
LPC2106-ROM.ld	GNU linker script file	Defines how the code and data emitted by the GNU C compiler and assembler are to be loaded into memory.
Makefile	GNU makefile	Tells make utility how to compile and link a program.
LPC210X.h	Standard LPC2106 header file	Register address location file.
main.c	Main C program	Includes the main() function that is the program entry point.

Εκτός από τα παραπάνω βασικά αρχεία, στο κάθε project μπορεί να συμπεριλαμβάνονται και μερικά άλλα C source και header files. Στο συγκεκριμένο project file (Auto_Pilot) περιλαμβάνονται τα ακόλουθα πρόσθετα αρχεία (Πνκς Β.2).

Πίνακας Β.2
Πρόσθετα αρχεία του project Auto_Pilot.

Name	File type	Description
armVIC.c	C source file	Interface routines for setting up and controlling the various interrupt modes.
armVIC.h	C header file	armVIC.c function definitions.
Button.c	C source file	Interface routines for setting up and using the button pin (P0.15).
Button.h	C header file	Button.c function definitions.
CMUcam2.c	C source file	Interface routines for setting up and using the CMUcam2 which is connected on UART1.
CMUcam2.h	C header file	CMUcam2.c function definitions.
config.h	C header file	Includes handy configuration definitions.
PWM.c	C source file	Interface routines for setting up and controlling servos and ESCs via PWM pulse.
PWM.h	C header file	PWM.c function definitions.
SystemInit.c	C source file	Interface routines for setting up the ARM processor's hardware.
SystemInit.h	C header file	SystemInit.c function definitions.
sysTime.c	C source file	Interface routines for setting up and using an elapsed time system timer.
sysTime.h	C header file	sysTime.c function definitions.
Transceiver.c	C source file	Interface routines for setting up the transceiver which is connected on UART0.
Transceiver.h	C header file	Transceiver.c function definitions.
types.h	C header file	Defines some regularly used typedefs.
uart.c	C source file	Interface routines for setting up and using the two UARTs.
uart.h	C header file	uart.c function definitions.
uartISR.c	C source file	Interrupt Service Routines (ISR) for the two UARTs.
uartISR.h	C header file	uartISR.c function definitions.

Τέλος, δίνονται οι συναρτήσεις που περιέχονται σε κάθε αρχείο (Πνκς Β.3 – Β.11). Σε κάθε συνάρτηση υπάρχει η περιγραφή της λειτουργίας της, τα καλούμενα δεδομένα καθώς και τα επιστρεφόμενα δεδομένα.

Πίνακας Β.3
Συναρτήσεις που περιέχονται στο αρχείο armVIC.

Function name	Description	Calling sequence	Return
__get_cpsr()	This function gets the Current Program Status Register (CPSR).	void	retval: The content of CPSR. (binary)
__set_cpsr()	This function sets the Current Program Status Register (CPSR).	val: The number that we want to load in CPSR. (binary)	void
disableIRQ()	This function sets the IRQ disable bit in the status register.	void	_cpsr: The previous value of CPSR. (binary)
enableIRQ()	This function clears the IRQ disable bit in the status register.	void	_cpsr: The previous value of CPSR. (binary)
restoreIRQ()	This function restores the IRQ disable bit in the status register to the value contained within passed oldCPSR.	void	_cpsr: The previous value of CPSR. (binary)
disableFIQ()	This function sets the FIQ disable bit in the status register.	void	_cpsr: The previous value of CPSR. (binary)
enableFIQ()	This function clears the FIQ disable bit in the status register.	void	_cpsr: The previous value of CPSR. (binary)
restoreFIQ()	This function restores the FIQ disable bit in the status register to the value contained within passed oldCPSR.	void	_cpsr: The previous value of CPSR. (binary)

Πίνακας Β.4
Συναρτήσεις που περιέχονται στο αρχείο Button.

Function name	Description	Calling sequence	Return
ButtonState()	This function checks if a key has been pressed.	void	-1: Key changed or bouncing 0: Key released 1: Key pressed
ButtonPress()	This function calls ButtonState() function and if button pressed calls some other functions.	void	void

Πίνακας Β.5
Συναρτήσεις που περιέχονται στο αρχείο CMUcam2.

Function name	Description	Calling sequence	Return
CamInit()	This function initializes the CMUcam2 for color tracking.	void	void
CamScan()	This function tells the CMUcam2 scanning the area to find the color that we want to track.	void	void
CamTrack()	This function makes the ship track the color that we want.	void	void

Πίνακας Β.6
Συναρτήσεις που περιέχονται στο αρχείο PWM.

Function name	Description	Calling sequence	Return
InitPWM()	This function initializes PWM registers.	void	void
StartPWM()	This function starts PWM pulse.	void	void
ActuatorsInit()	This function initializes actuators (rudder servo and thruster ESC).	void	void
SetRudder()	This function sets rudder's servo dutycycle.	void	void
SetThruster()	This function sets thruster's ESC dutycycle.	void	void
StopPWM()	This function stops PWM pulse.	void	void

Πίνακας Β.7
Συναρτήσεις που περιέχονται στο αρχείο SystemInit.

Function name	Description	Calling sequence	Return
lowInit()	This function starts up the PLL then sets up the GPIO pins before waiting for the PLL to lock. It finally engages the PLL and returns.	void	void
sysInit()	This function is responsible for initializing the program specific hardware.	void	void

Πίνακας Β.8
Συναρτήσεις που περιέχονται στο αρχείο sysTime.

Function name	Description	Calling sequence	Return
initSysTime()	This function initializes the LPC's Timer 0 for use as the system timer.	void	void
getSysTICs()	This function returns the current syetem time in TICs.	void	sysTICs: The current time in TICs. (unsigned long)
getElapsedSysTICs()	This function returns the difference in TICs between the given starting time and the current system time.	startime: The starting time. (unsigned long)	The time difference. (unsigned long)
pause()	This function does not return until the specified 'duration' in TICs has elapsed.	duration: The length of time in TICs to wait before returning. (unsigned long)	void

Πίνακας Β.9
Συναρτήσεις που περιέχονται στο αρχείο Transceiver.

Function name	Description	Calling sequence	Return
TransInit()	This function initializes the transceiver.	void	void

Πίνακας Β.10
Συναρτήσεις που περιέχονται στο αρχείο uart.

Function name	Description	Calling sequence	Return
uart0/1Init()	This function initializes the UART for async mode.	baud: Baudrate divisor. (unsigned short) mode: Data bits, parity, stop bits. (unsigned char) fmode: FIFO (unsigned char)	void
uart0/1Putch()	This function puts a character into the UART output queue for transmission.	ch: Character to be transmitted. (integer)	ch: Character on success. (integer) -1: On error (queue full).
uart0/1Puts()	This function writes a NULL terminated 'string' to the UART output queue, returning a pointer to the next character to be written.	The address of the string. (const char)	A pointer to the next character to be written. (const char) \0: If full string is written
printf0/1()	This function puts the decimal of an ASCII character.	n: Number. (integer)	void
uart0/1TxFlush()	This function removes all characters from the UART transmit queue (without transmitting them).	void	void
uart0/1Getch()	This function gets a character from the UART receive queue.	void	ch: Character on success. (integer) -1: If no character is available.

Πίνακας Β.11
Συναρτήσεις που περιέχονται στο αρχείο uartISR.

Function name	Description	Calling sequence	Return
uart0ISR()	This function implements the ISR for UART0.	void	void
uart1ISR()	This function implements the ISR for UART1.	void	void

Τέλος, για πληρέστερη κατανόηση των ανωτέρω, βασικά τμήματα του κώδικα παρουσιάζονται αναλυτικά στο Παράρτημα Γ ενώ ο πλήρης κώδικας περιέχεται στο επισυναπτόμενο CD.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τη γλώσσα προγραμματισμού C υπάρχουν στη βιβλιογραφία (Kernighan, 2002).*

B.1.2 Εφαρμογή Λογισμικού στον Olimex LPC-P2106

Η παρακάτω περιγραφή προϋποθέτει ότι έχουμε έτοιμο pn project το οποίο διαθέτει τα βασικά αρχεία που αναφέρθηκαν αρχικά (Πνκς Β.1). Ο τρόπος φόρτωσης του λογισμικού στον Olimex διαφοροποιείται ανάλογα με το mode που θέλουμε να χρησιμοποιήσουμε. Έτσι στη συνέχεια παρουσιάζονται οι διάφοροι τρόποι εφαρμογής (*modes*), η προετοιμασία του Makefile και του hardware στο mode που μας ενδιαφέρει καθώς και ο τρόπος φόρτωσης και εκτέλεσης της εφαρμογής.

➢ **Τρόποι λειτουργίας:**

1. Εφαρμογή προγράμματος στη μνήμη FLASH (χωρίς debugging).
2. Εφαρμογή προγράμματος και debugging στη μνήμη FLASH.
3. Εφαρμογή προγράμματος και debugging στη μνήμη RAM.

Σχόλιο: *Εμείς δουλεύουμε με τον 1^{ο} τρόπο.*

➢ **Προετοιμασία Makefile:**

- Ανοίγουμε το pn project κάνοντας διπλό κλικ στο Auto_Pilot PNPROJ File.

- Ανοίγουμε το Makefile και πάμε στη γραμμή 24 ώστε να ρυθμίσουμε τα χαρακτηριστικά του επεξεργαστή που χρησιμοποιούμε:

```
# MCU name and submodel
MCU = arm7tdmi
SUBMDL = LPC2106
```

- Στη γραμμή 31 «ξεσχολιάζουμε» το RUN_MODE=ROM_RUN ώστε να φορτώσουμε το πρόγραμμά μας στη μνήμη FLASH χωρίς debugging:

```
## Create ROM-Image (final)
RUN_MODE=ROM_RUN
## Create RAM-Image (debugging)
#RUN_MODE=RAM_RUN
```

- Στη γραμμή 37 ορίζουμε το output format ώστε κατά το “making” να παράγουμε δεκαεξάρικο αρχείο (.hex file):

```
# Output format. (can be srec, ihex, binary)
FORMAT = ihex
```

- Στη γραμμή 41 συμπληρώνουμε το όνομα του αρχείου που περιέχει τη συνάρτηση main():

```
# Target file name (without extension).
TARGET = main
```

- Στη γραμμή 45 συμπληρώνουμε τα ονόματα των C source αρχείων εκτός του αρχείου που περιέχει τη main():

```
# List C source files here. (C dependencies are automatically
generated.)
SRC = $(TARGET).c
SRC += SystemInit.c sysTime.c uart.c uartISR.c armVIC.c PWM.c
CMUcam2.c Button.c Transceiver.c
```

- Στη γραμμή 151 ρυθμίζουμε τις παραμέτρους του in-system-programming-software, lpc21isp. Οι παράμετροι που ρυθμίζονται εδώ είναι η ονομασία του προγράμματος, η χρησιμοποιούμενη σειριακή θύρα του Η/Υ, το baud rate, η συχνότητα του κρυστάλλου της αναπτυξιακής κάρτας σε kHz και το αρχείο που φορτώνεται στον επεξεργαστή:

```
# Settings and variables:
LPC21ISP = lpc21isp
LPC21ISP_PORT = com1
LPC21ISP_BAUD = 115200
LPC21ISP_XTAL = 14746
LPC21ISP_FLASHFILE = $(TARGET).hex
```

Σχόλιο: *Για περισσότερες λεπτομέρειες σχετικά με τα περιεχόμενα και τις λειτουργίες του Makefile βλέπε τον κώδικα του Makefile στο Παράρτημα Γ.*

➢ Προετοιμασία hardware:

- Συνδέουμε το σειριακό καλώδιο στον Olimex (UART0) και στο PC.
- Τροφοδοτούμε με ηλεκτρικό ρεύμα τον Olimex (9V DC).
- Τοποθετούμε τα jumpers στα pins BSL και JTAG πάνω στην αναπτυξιακή κάρτα του Olimex..

➢ Φόρτωση – Εκτέλεση κώδικα:

- Ανοίγουμε το pn project κάνοντας διπλό κλικ στο Auto_Pilot PNPROJ File.
- Στο Programmers Notepad 2 εκτελούμε: **Tools – Make Clean**.
- Στη συνέχεια εκτελούμε: **Tools – Make Program**. Στο window Output του Programmers Notepad 2 παρουσιάζονται τα πιθανά warnings, errors όπως επίσης και η πιθανή κατάληψη της σειριακής θύρας από άλλο hardware ή software. Αν όλα έχουν ρυθμιστεί σωστά και δεν υπάρξει κανένα σφάλμα, φορτώνεται ο κώδικας με τη μορφή του 16ικού αρχείου main.hex στον επεξεργαστή μέσω της σειριακής θύρας, ενώ στο window Output εμφανίζεται το τελικό μήνυμα:

```
Download Finished... taking x seconds
Now launching the brand new code
```

- Το πρόγραμμα έχει φορτωθεί στον Olimex. Για να εκτελεστεί ο κώδικας, αφαιρούμε το **BSL jumper** και πατάμε το **reset button**. Ο κώδικας εκτελείται.

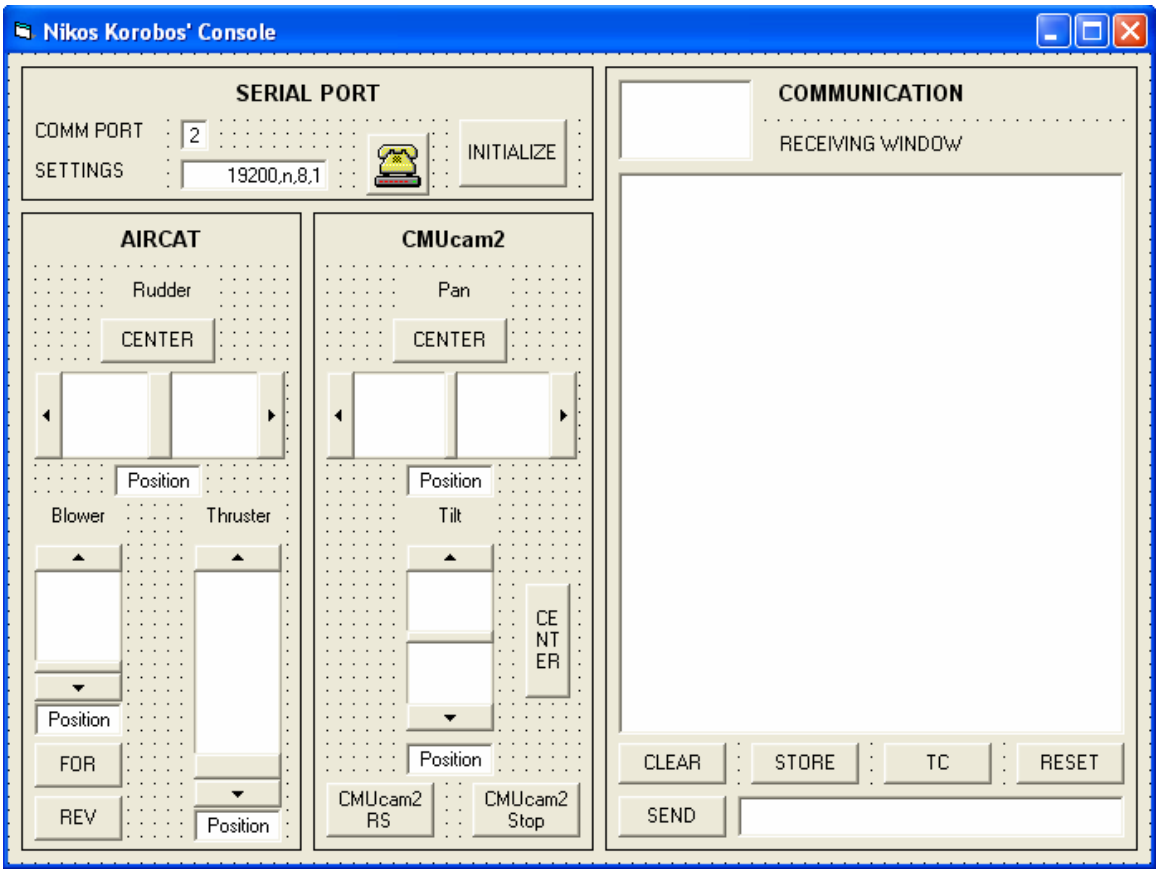
Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τα ανωτέρω υπάρχουν στη βιβλιογραφία (Lynch, 2005).*

Β.2 Δημιουργία κονσόλας ελέγχου

Η κονσόλα ελέγχου όπως είπαμε είναι ένας δίαυλος επικοινωνίας (*interface*) του χρήστη με το σύστημα. Για αυτό το λόγο χρειαζόμαστε ένα περιβάλλον οπτικού προγραμματισμού (*visual programming*) για τη δημιουργία της κονσόλας. Έτσι, επιλέγεται το ολοκληρωμένο περιβάλλον ανάπτυξης (*Integrated Development Environment - IDE*) της Visual Basic 6.0.

Β.2.1 Περιγραφή παραθύρου κονσόλας του Visual Basic Project

Στη συνέχεια παρουσιάζεται η κονσόλα ελέγχου (Σχ. Β.1) και περιγράφονται τα στοιχεία της (*elements*) (Πινκς Β.12 – Β.15).



Σχήμα. Β.1. *Η κονσόλα ελέγχου κατά την ανάπτυξή της.*

Πίνακας Β.12
Elements που περιέχονται στην περιοχή SERIAL PORT της κονσόλας.

Field	Elemet	Function
SERIAL PORT	MSComm	This element links the console window with the PC serial port.
	COMM PORT TextBox	This TextBox allows user to specify serial port number.
	SETTINGS TextBox	This TextBox allows user to specify serial port settings.
	INITILIZE CommandButton	This CommandButton initializes the serial port.

Πίνακας Β.13
Elements που περιέχονται στην περιοχή AIRCAT της κονσόλας.

Field	Elemet	Function
AIRCAT	Rudder HScrollBar	This HScrollBar controls the rudder.
	Rudder CENTER CommandButton	This CommandButton moves the rudder to the center position.
	Rudder Position TextBox	This TextBox shows the rudder's position.
	Blower VScrollBar	This VScrollBar controls the blower.
	Blower Position TextBox	This TextBox shows the blower's position.
	FOR CommandButton	This CommandButton makes AIRCAT to move in forward direction.
	REV CommandButton	This CommandButton makes AIRCAT to move in reverse direction.
	Thruster VScrollBar	This VScrollBar controls the thruster.
	Thruster Position TextBox	This TextBox shows the thruster's position.

Πίνακας Β.14
Elements που περιέχονται στην περιοχή CMUcam2 της κονσόλας.

Field	Elemet	Function
CMUcam2	Pan HScrollBar	This HScrollBar controls the camera's pan servo.
	Pan CENTER CommandButton	This CommandButton moves the pan servo to the center position.
	Pan Position TextBox	This TextBox shows the pan servo position.
	Tilt VScrollBar	This VScrollBar controls the camera's tilt servo.
	Tilt CENTER CommandButton	This CommandButton moves the tilt servo to the center position.
	Tilt Position TextBox	This TextBox shows the tilt servo position.

Πίνακας Β.15
Elements που περιέχονται στην περιοχή COMMUNICATION της κονσόλας.

Field	Elemet	Function
COMMUNICATION	STORE TextBox	This TextBox keeps all received data from the ship.
	RECEIVING WINDOW TextBox	This TextBox shows the last 300 characters received from the ship.
	CLEAR CommandButton	This CommandButton clears the receiving window.
	STORE CommandButton	This CommandButton stores color tracking data from CMUcam2 to a file.
	TC CommandButton	This CommandButton tells CMUcam2 to track color.
	RESET CommandButton	
	SEND CommandButton	This CommandButton sends a user's defined text to the ship.
	SEND TextBox	This TextBox allow user to specify a custom text.

Β.2.2 Παρουσίαση υπορουτινών του Visual Basic Project

Αφού παρουσιάσαμε το οπτικό (*visual*) κομμάτι της κονσόλας ελέγχου, είναι αναγκαία η περιγραφή του κώδικα που περιέχεται «πίσω» από κάθε στοιχείο. Έτσι, δίνονται οι υπορουτίνες που αντιστοιχούν στα elements της κονσόλας (Πνκς Β.16 – Β.19). Σε κάθε υπορουτίνα υπάρχει η περιγραφή της λειτουργίας της.

Πίνακας Β.16
Υπορουτίνες που αντιστοιχούν στα elements της περιοχής SERIAL PORT της κονσόλας.

Field	Subroutine name	Description
SERIAL PORT	MSComm1_OnComm()	This subroutine is called whenever a serial port error or event occurs.
	initialize_Click()	This subroutine initializes the UART and the transceiver.

Πίνακας Β.17
Υπορουτίνες που αντιστοιχούν στα elements της περιοχής AIRCAT της κονσόλας.

Field	Subroutine name	Description
AIRCAT	rudder_Change()	This subroutine controls AIRCAT's rudder.
	center_Click()	This subroutine moves rudder to the center position.
	thruster_Change()	This subroutine controls AIRCAT's thruster.
	blower_Change()	This subroutine controls AIRCAT's blower.
	reverser_Click()	This subroutine makes AIRCAT to move in reverse direction.
	forward_Click()	This subroutine makes AIRCAT to move in forward direction.

Πίνακας Β.18
Υπορουτίνες που αντιστοιχούν στα elements της περιοχής CMUcam2 της κονσόλας.

Field	Subroutine name	Description
CMUcam2	rs_Click()	This subroutine resets CMUcam2 board.
	stop_Click()	This subroutine stops streaming data from camera.
	pan_Change()	This subroutine controls CMUcam2 pan servo.
	centerpan_Click()	This subroutine moves pan servo to the center position.
	tilt_Change()	This subroutine controls CMUcam2 tilt servo.
	centertilt_Click()	This subroutine moves tilt servo to the center position.

Πίνακας Β.19
Υπορουτίνες που αντιστοιχούν στα elements της περιοχής COMMUNICATION της κονσόλας.

Field	Subroutine name	Description
COMMUNICATION	clear_Click()	This subroutine clears the receiving window.
	store_Click()	This subroutine stores all received data from CMUcam2 to data.xls.
	tc_Click()	This subroutine tells CMUcam2 to track color.
	reset_Click()	This subroutine resets Olimex board.
	send_Click()	This subroutine sends a user's defined text through the serial port.

Τέλος, για πληρέστερη κατανόηση των ανωτέρω, όλος ο κώδικας παρουσιάζεται αναλυτικά στο Παράρτημα Δ ενώ υπάρχει και στο επισυναπτόμενο CD.

Σχόλιο: *Περισσότερες πληροφορίες σχετικά με τη γλώσσα προγραμματισμού Visual Basic υπάρχουν στη βιβλιογραφία (Petroutsos, 1998).*

Γ. ΚΩΔΙΚΑΣ C

Γ.1 main.c

```
//////////
// main //
//////////

#include "types.h"
#include "SystemInit.h"
#include "LPC210x.h"
#include "config.h"
#include "armVIC.h"
#include "sysTime.h"
#include "uart.h"
#include "PWM.h"
#include "CMUcam2.h"
#include "Button.h"
#include "Transceiver.h"
#include <stdlib.h> // for atoi() function

/*****************************************************************
*
* Function Name: main()
*
* Description:
*     This function is the program entry point. Its work is:
*       1. System initializing
*       2. Interrupts enabling
*       3. Transceiver initializing
*       4. Greeting sending out UART0
*       5. Actuators initializing
*       6. Camera initializing
*       7. Endless loop that manipulate the characters from 2 UARTs
*          and 2 PWMs
*
* Calling Sequence:
*     void
*
* Returns:
*     void
*
*****************************************************************/
int main(void)
{

sysInit();
enableIRQ();
TransInit();
pause(HALF_SEC);
uart0Puts("Hello Minarikos!\r\n");
ActuatorsInit();
CamInit();
CamTrack();

int ch0, ch1, q, i, j, k;
int e1_k, e1_k_1, e2_k, e2_k_1, tp, tp_set, pn, pn_set;
float th, rd, u1_k, u1_k_1, u2_k, u2_k_1, Kpt, Kit, Kpr, Kir, Ts;
uint32_t TimeLabel;
```

```
////////////////////////////////////////////////////////////////////
// Set points and PI gains.
////////////////////////////////////////////////////////////////////
Ts=0.0834; // Sample Time (0.0833681233723951)

tp_set=80;
pn_set=60;

Kpt=0.01;
Kit=0.001;
Kpr=0.0125;
Kir=0.001;

u1_k_1=0;
u2_k_1=0;

e1_k_1=0;
e2_k_1=0;

while(1){

char temp[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char temp1[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char mx[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
char pixel[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

ch0=uart0Getch();
ch1=uart1Getch();

////////////////////////////////////////////////////////////////////
// This section gets T packets from UART1 while CMUcam2 is in
// autotracking mode, process their data and creates actuators'
// positions.
// T packets include only 2 arguments (mx pixels) after CMUcam2
// setup in CamInit() and CamTrack() functions.
//
// T packets format: "T mx pixels\r"
// mx range:      [0:1:159]
// pixels:        [0:1:255]
////////////////////////////////////////////////////////////////////
if(ch1=='T'){

TimeLabel = getSysTICs();

// Store all the T packet in temp[] matrix.
      ch1=uart1Getch();
      i=0;
      while(ch1!='\r'){
            temp[i]=uart1Getch();
            ch1=temp[i];
            i++;
      }

// Take the first argument of T packet (mx) from temp[] matrix and
// store it in temp1[] matrix.
      q=0;
      while(temp[q]!=' '){
            temp1[q]=temp[q];
            q++;
      }

// Take the data from temp1[] matrix, throw out the ' ' character
// and store only the number "mx" in mx[] matrix.
      j=0;
      while(j<q){
            mx[j]=temp1[j];
            j++;
      }
      tp=atoi(mx);      // convert array to integer
```

```
// Take the data from temp[] matrix and store only the number
// "pixel" in pixel[] matrix.
      k=0;
      for(j=q+1;j<i-1;j++){
            pixel[k]=temp[j];
            k++;
      }
      pn=atoi(pixel);   // convert array to integer

////////////////////////////////////////////////////////////////////
// Here will be make the PI controller that creates the thruster's
// position.
////////////////////////////////////////////////////////////////////
      e1_k = pn_set-pn; // error calculation

      // normilized thruster's PWM [0~100%]
      u1_k = u1_k_1 + Kpt*e1_k - Kpt*e1_k_1 + Ts*Kit*e1_k;

      if(u1_k<=0){ // No negative
      u1_k=0;
      }

      else if(u1_k>0.5){ // No over unity
      u1_k=0.5;
      }

      if(pn==0){ // If the beacon isn't in camera's FOV set thruster
                 // to zero
      e1_k=0;
      u1_k=0;
      th=zero_thruster;
      SetThruster((th*PCLK)/1000);
      }

      else if(pn>90){ // Safety
      e1_k=0;
      u1_k=0;
      th=zero_thruster;
      SetThruster((th*PCLK)/1000);
      }

      else{
      th=(14*u1_k+100)*0.01; // real thruster's PWM [1.00~1.14ms]
      SetThruster((th*PCLK)/1000);
      }

      e1_k_1=e1_k;
      u1_k_1=u1_k;
```

```
//////////////////////////////////////////////////////////////////
// Here will be make the PI controller that creates the rudder's
// position.
//////////////////////////////////////////////////////////////////
      e2_k = tp_set-tp; // error calculation

      // normalized thruster's PWM [0~100%]
      u2_k = u2_k_1 + Kpr*e2_k - Kpr*e2_k_1 + Ts*Kir*e2_k;

      if(u2_k<=-1){ // No under -1
      u2_k=-1;
      }

      else if(u2_k>=1){ // No over 1
      u2_k=1;
      }

      if(tp==0){
      e2_k=0;
      u2_k=0;
      rd=center_rudder;
      SetRudder((rd*PCLK)/1000); // If the beacon isn't in camera's
                                 // FOV set thruster to zero
      }

      else{
      rd=(-35*u2_k+125)*0.01; // real rudder's PWM [0.90~1.60ms]
      SetRudder((rd*PCLK)/1000);
      }

      e2_k_1=e2_k;
      u2_k_1=u2_k;

//////////////////////////////////////////////////////////////////
// Send sensor's data to PC via wireless transceivers.
//////////////////////////////////////////////////////////////////
      printf0(tp);              // show tp number on terminal
      uart0Puts("\t");
      printf0(pn);              // show pn number on terminal
      uart0Puts("\t");
      printf0(TimeLabel);       // show time on terminal
      uart0Puts("\r\n");
}
```

```
////////////////////////////////////////////////////////////////////
// This section gets R packets from UART0, process their data and
// creates rudder's positions.
//
// R packets format:    "R rudder\r"
// rudder range:        [90:1:160] (AIRCAT calibration)
////////////////////////////////////////////////////////////////////
if(ch0=='R'){

// Store all the R packet in temp[] matrix.
      ch0=uart0Getch();
      i=0;
      while(ch0!='\r'){
            temp[i]=uart0Getch();
            ch0=temp[i];
            i++;
      }

// Take the data from temp[] matrix and store only the number
// "rudder" in mx[] matrix.
      k=0;
      for(j=0;j<i-1;j++){
            mx[k]=temp[j];
            k++;
      }
      tp=atoi(mx);      // convert array to integer

      uart0Puts("Rpacket: ");
      printf0(tp);      // show sv number on terminal
      uart0Puts("\r\n");
      rd=tp*0.01;

      SetRudder((rd*PCLK)/1000);    // create rudder's position from
                                    // transmitter's request.
}
```

```
//////////////////////////////////////////////////////////////////
// This section gets W packets from UART0, process their data and
// creates thruster's positions.
//
// W packets format:    "W thruster\r"
// thruster range:      [94:1:114] (AIRCAT calibration)
//////////////////////////////////////////////////////////////////
if(ch0=='W'){

// Store all the W packet in temp[] matrix.
      ch0=uart0Getch();
      i=0;
      while(ch0!='\r'){
            temp[i]=uart0Getch();
            ch0=temp[i];
            i++;
      }

// Take the data from temp[] matrix and store only the number
// "thruster" in mx[] matrix.
      k=0;
      for(j=0;j<i-1;j++){
            mx[k]=temp[j];
            k++;
      }
      tp=atoi(mx);      // convert array to integer

      uart0Puts("Wpacket: ");
      printf0(tp);      // show tp number on terminal
      uart0Puts("\r\n");
      th=tp*0.01;

      SetThruster((th*PCLK)/1000);  // create thruster's position
                                    // from transmitter's request.
}

//////////////////////////////////////////////////////////////////
// Get 'O' character from UART0 and calls main() function for reset.
//////////////////////////////////////////////////////////////////
if (ch0 == 'O') {
            main();
}

//////////////////////////////////////////////////////////////////
// Get characters from UART0 and put them in UART1.
//////////////////////////////////////////////////////////////////
if (ch0 >= 0) {
      uart1Putch(ch0);
}

//////////////////////////////////////////////////////////////////
// Get characters from UART1 and put them in UART0.
//////////////////////////////////////////////////////////////////
//if (ch1 >= 0) {
//    uart0Putch(ch1);
//}

ButtonPress(); // check if button have been pressed; if pressed then
               // execute some functions.

}
  return 0;
}
```

Γ.2 makefile

```
# Hey Emacs, this is a -*- makefile -*-
#
# WinARM template makefile
#
# based on the WinAVR makefile written by Eric B. Weddington, Jφrg
# Wunsch, et al.
# Released to the Public Domain
# Please read the make user manual!
#
#
# On command line:
#
# make all = Make software.
#
# make clean = Clean out built project files.
#
# make program = Download the hex file to the device, using isp21lpc
#
# make filename.s = Just compile filename.c into the assembler code
# only
#
# To rebuild project do "make clean" then "make all".
#


# MCU name and submodel
MCU = arm7tdmi
SUBMDL = LPC2106
#THUMB    = -mthumb
#THUMB_IW = -mthumb-interwork


## Create ROM-Image (final)
RUN_MODE=ROM_RUN
## Create RAM-Image (debugging)
#RUN_MODE=RAM_RUN


# Output format. (can be srec, ihex, binary)
FORMAT = ihex


# Target file name (without extension).
TARGET = main


# List C source files here. (C dependencies are automatically
# generated.)
SRC = $(TARGET).c
SRC  +=  SystemInit.c  sysTime.c  uart.c  uartISR.c  armVIC.c  PWM.c
CMUcam2.c
       Button.c Transceiver.c


# List Assembler source files here.
# Make them always end in a capital .S.  Files ending in a lowercase
# .s will not be considered source files but generated files
# (assembler output from the compiler), and will be deleted upon
# "make clean"!
# Even though the DOS/Win* filesystem matches both .s and .S the,
# same, it will preserve the spelling of the filenames, and gcc
# itself does care about how the name is spelled on its command-ine.
ASRC = crt0.S
```

```
# Optimization level, can be [0, 1, 2, 3, s].
# 0 = turn off optimization. s = optimize for size.
# (Note: 3 is not always the best optimization level. See avr-libc
# FAQ.)
OPT = s

# Debugging format.
# Native formats for AVR-GCC's -g are stabs [default], or dwarf-2.
# AVR (extended) COFF requires stabs, plus an avr-objcopy run.
#DEBUG = stabs
DEBUG = dwarf-2

# List any extra directories to look for include files here.
# Each directory must be seperated by a space.
# EXTRAINCDIRS = ./include
EXTRAINCDIRS =

# Compiler flag to set the C Standard level.
# c89   - "ANSI" C
# gnu89 - c89 plus GCC extensions
# c99   - ISO C99 standard (not yet fully implemented)
# gnu99 - c99 plus GCC extensions
CSTANDARD = -std=gnu99

# Place -D or -U options for C here
CDEFS =  -D$(RUN_MODE)

# Place -I options here
CINCS =

# Place -D or -U options for ASM here
ADEFS =  -D$(RUN_MODE)

# Compiler flags.
# -g*:          generate debugging information
# -O*:          optimization level
# -f...:        tuning, see GCC manual and avr-libc documentation
# -Wall...:     warning level
# -Wa,...:      tell GCC to pass this to the assembler.
# -adhlns...:   create assembler listing
CFLAGS = -g$(DEBUG)
CFLAGS += $(CDEFS) $(CINCS)
CFLAGS += -O$(OPT)
CFLAGS += -Wall -Wstrict-prototypes -Wcast-align -Wcast-qual
          -Wimplicit
CFLAGS += -Wmissing-declarations
CFLAGS += -Wmissing-prototypes -Wnested-externs -Wpointer-arith
          -Wswitch
CFLAGS += -Wredundant-decls -Wreturn-type -Wshadow
          -Wstrict-prototypes -Wunused
CFLAGS += -Wa,-adhlns=$(<:.c=.lst)
CFLAGS += $(patsubst %,-I%,$(EXTRAINCDIRS))
CFLAGS += $(CSTANDARD)
## NONO CFLAGS += -funsigned-char -funsigned-bitfields -fpack-struct
##                -fshort-enums

# Assembler flags.
# -Wa,...:   tell GCC to pass this to the assembler.
# -ahlms:    create listing
# -gstabs:   have the assembler create line number information; note
#            that for use in COFF files, additional information about
#            filenames and function names needs to be present in the
#            assembler source files -- see avr-libc docs [FIXME: not
#            yet described there]
##ASFLAGS = -Wa,-adhlns=$(<:.S=.lst),-gstabs
ASFLAGS = $(ADEFS) -Wa,-adhlns=$(<:.S=.lst),-g$(DEBUG)
```

```
#Additional libraries.

#Support for newlibc-lpc (file: libnewlibc-lpc.a)
#NEWLIBLPC = -lnewlib-lpc
NEWLIBCLPC =

MATH_LIB = -lm

# Linker flags.
# -Wl,...:     tell GCC to pass this to linker.
# -Map:       create map file
# --cref:     add cross reference to  map file
LDFLAGS = -nostartfiles -Wl,-Map=$(TARGET).map,--cref
LDFLAGS += -lc
LDFLAGS += $(NEWLIBLPC) $(MATH_LIB)
LDFLAGS += -lc -lgcc

# Set Linker-Script Depending On Selected Memory
ifeq ($(RUN_MODE),RAM_RUN)
LDFLAGS +=-T$(SUBMDL)-RAM.ld
else
LDFLAGS +=-T$(SUBMDL)-ROM.ld
endif


# ---------------------------------------------------------------------
# Flash-Programming support using lpc21isp by Martin Maurer

# Settings and variables:
LPC21ISP = lpc21isp
LPC21ISP_PORT = com1
LPC21ISP_BAUD = 115200
LPC21ISP_XTAL = 14746
LPC21ISP_FLASHFILE = $(TARGET).hex
# verbose output:
## LPC21ISP_DEBUG = -debug
# enter bootloader via RS232 DTR/RTS (only if hardware supports this
# feature - see Philips AppNote):
## LPC21ISP_CONTROL = -control


# ---------------------------------------------------------------------

# Define directories, if needed.
## DIRARM = c:/WinARM/
## DIRARMBIN = $(DIRAVR)/bin/
## DIRAVRUTILS = $(DIRAVR)/utils/bin/

# Define programs and commands.
SHELL = sh
CC = arm-elf-gcc
OBJCOPY = arm-elf-objcopy
OBJDUMP = arm-elf-objdump
SIZE = arm-elf-size
NM = arm-elf-nm
REMOVE = rm -f
COPY = cp
```

```
# Define Messages
# English
MSG_ERRORS_NONE = Errors: none
MSG_BEGIN = -------- begin --------
MSG_END = --------  end  --------
MSG_SIZE_BEFORE = Size before:
MSG_SIZE_AFTER = Size after:
MSG_FLASH = Creating load file for Flash:
MSG_EXTENDED_LISTING = Creating Extended Listing:
MSG_SYMBOL_TABLE = Creating Symbol Table:
MSG_LINKING = Linking:
MSG_COMPILING = Compiling:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:
MSG_LPC21_RESETREMINDER = You may have to bring the target in
                          bootloader-mode now.

# Define all object files.
OBJ = $(SRC:.c=.o) $(ASRC:.S=.o)

# Define all listing files.
LST = $(ASRC:.S=.lst) $(SRC:.c=.lst)

# Compiler flags to generate dependency files.
### GENDEPFLAGS = -Wp,-M,-MP,-MT,$(*F).o,-MF,.dep/$(@F).d
GENDEPFLAGS = -MD -MP -MF .dep/$(@F).d

# Combine all necessary flags and optional flags.
# Add target processor to flags.
ALL_CFLAGS = -mcpu=$(MCU) $(THUMB_IW) -I. $(CFLAGS) $(GENDEPFLAGS)
ALL_ASFLAGS = -mcpu=$(MCU) $(THUMB_IW) -I. -x assembler-with-cpp
              $(ASFLAGS)


# Default target.
all: begin gccversion sizebefore build sizeafter finished end

build: elf hex lss sym

elf: $(TARGET).elf
hex: $(TARGET).hex
lss: $(TARGET).lss
sym: $(TARGET).sym
# Eye candy.
begin:
      @echo
      @echo $(MSG_BEGIN)

finished:
      @echo $(MSG_ERRORS_NONE)

end:
      @echo $(MSG_END)
      @echo

# Display size of file.
HEXSIZE = $(SIZE) --target=$(FORMAT) $(TARGET).hex
ELFSIZE = $(SIZE) -A $(TARGET).elf
sizebefore:
      @if [ -f $(TARGET).elf ]; then echo; echo $(MSG_SIZE_BEFORE);
                                    $(ELFSIZE); echo; fi

sizeafter:
      @if [ -f $(TARGET).elf ]; then echo; echo $(MSG_SIZE_AFTER);
                                    $(ELFSIZE); echo; fi

# Display compiler version information.
gccversion:
      @$(CC) --version
```

```
# Program the device.
program: $(TARGET).hex
	@echo
	@echo $(MSG_LPC21_RESETREMINDER)
	$(LPC21ISP) $(LPC21ISP_CONTROL) $(LPC21ISP_DEBUG)
	$(LPC21ISP_FLASHFILE) $(LPC21ISP_PORT) $(LPC21ISP_BAUD)
	$(LPC21ISP_XTAL)

# Create final output files (.hex, .eep) from ELF output file.
# TODO: handling the .eeprom-section should be redundant
%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) $< $@

# Create extended listing file from ELF output file.
%.lss: %.elf
	@echo
	@echo $(MSG_EXTENDED_LISTING) $@
	$(OBJDUMP) -h -S $< > $@

# Create a symbol table from ELF output file.
%.sym: %.elf
	@echo
	@echo $(MSG_SYMBOL_TABLE) $@
	$(NM) -n $< > $@


# Link: create ELF output file from object files.
.SECONDARY : $(TARGET).elf
.PRECIOUS : $(OBJ)
%.elf: $(OBJ)
	@echo
	@echo $(MSG_LINKING) $@
	$(CC) $(THUMB) $(ALL_CFLAGS) $(OBJ) --output $@ $(LDFLAGS)

# Compile: create object files from C source files.
%.o : %.c
	@echo
	@echo $(MSG_COMPILING) $<
	$(CC) -c $(THUMB) $(ALL_CFLAGS) $< -o $@

# Compile: create assembler files from C source files.
%.s : %.c
	$(CC) $(THUMB) -S $(ALL_CFLAGS) $< -o $@

# Assemble: create object files from assembler source files.
%.o : %.S
	@echo
	@echo $(MSG_ASSEMBLING) $<
	$(CC) -c $(ALL_ASFLAGS) $< -o $@
```

```
# Target: clean project.
clean: begin clean_list finished end

clean_list :
	@echo
	@echo $(MSG_CLEANING)
	$(REMOVE) $(TARGET).hex
	$(REMOVE) $(TARGET).obj
	$(REMOVE) $(TARGET).elf
	$(REMOVE) $(TARGET).map
	$(REMOVE) $(TARGET).obj
	$(REMOVE) $(TARGET).a90
	$(REMOVE) $(TARGET).sym
	$(REMOVE) $(TARGET).lnk
	$(REMOVE) $(TARGET).lss
	$(REMOVE) $(OBJ)
	$(REMOVE) $(LST)
	$(REMOVE) $(SRC:.c=.s)
	$(REMOVE) $(SRC:.c=.d)
	$(REMOVE) .dep/*


# Include the dependency files.
-include $(shell mkdir .dep 2>/dev/null) $(wildcard .dep/*)


# Listing of phony targets.
.PHONY : all begin finish end sizebefore sizeafter gccversion \
build elf hex lss sym \
clean clean_list program
```

Σχόλιο: *Ο υπόλοιπος κώδικας εφαρμογής για τον microcontroller βρίσκεται στο επισυναπτόμενο CD.*

Δ. ΚΩΔΙΚΑΣ VISUAL BASIC

```
' Nikos Korobos' Console
Option Explicit

' Include a delay function, "Sleep"
Private Declare Sub Sleep Lib "kernel32" _
(ByVal dwMilliseconds As Long)
```

```
'
' SERIAL PORT subroutines
'
' This subroutine is called whenever a serial port error or event
' occurs
Private Sub MSComm1_OnComm()
Dim strNewChar As String
Select Case MSComm1.CommEvent
' Errors
Case comEventBreak ' A Break was received
      MsgBox "Break received"
Case comEventFrame ' Framing Error
      MsgBox "Framing error"
Case comEventOverrun ' Data Lost
      MsgBox "Overrun error"
Case comEventRxOver ' Receive buffer overflow
      MsgBox "Receive Buffer overflow"
Case comEventRxParity ' Parity Error
      MsgBox "Parity Error"
Case comEventTxFull ' Transmit buffer full
      MsgBox "Transmit Buffer full"
Case comEventDCB ' Unexpected error retrieving DCB
      MsgBox "DCB error"

' Events
Case comEvCD ' Change in the CD line
      MsgBox "Carrier Detect changed state"
Case comEvCTS ' Change in the CTS line
      MsgBox "Clear To Send (CTS) changed state"
Case comEvDSR ' Change in the DSR line
      MsgBox "Data Set Ready (DSR) changed state"
Case comEvRing ' Change in the Ring Indicator
      MsgBox "Ring Indicator (RI) changed state"
Case comEvReceive ' Received RThreshold # of characters
      'MsgBox Str$(MSComm1.RThreshold) & " Characters received"
      While (MSComm1.InBufferCount > 0)
            txtStoreText = txtStoreText & MSComm1.Input
            txtReceivedText = txtStoreText
            ' only keep the last 1,000 characters -- throw away any
            ' characters older than that
            txtReceivedText = Right$(txtReceivedText, 300)
      Wend
Case comEvSend          ' There are SThreshold number of characters
                        ' in the transmit buffer
      MsgBox  Str$(MSComm1.SThreshold)  &  "  Characters  remaining  in
      transmit buffer"
Case comEvEOF           ' An EOF charater was found in the input
                        ' stream
```

```
      MsgBox "EOF received"
End Select

End Sub
```

```
'Initialize the UART and the Transceiver
Private Sub initialize_Click()

' read the COM port number from the screen
MSComm1.CommPort = Val(txtCommPort.Text)  ' MSComm1.CommPort = 1
' read the COM port settings from the screen
MSComm1.Settings = Val(txtSettings.Text)  ' 19200 baud, no parity, 8
                                          ' data, and 1 stop bit
' tell the control to read entire buffer when Input is used
MSComm1.InputLen = 0
' generate an event on every character received
MSComm1.RThreshold = 1
' Open the port.
MSComm1.PortOpen = True
' Note: we should set the multiline property to True here in code but
' it is a read-only property do not allow the user to change the
' received text
txtReceivedText.Locked = True
txtStoreText.Locked = True
' clear the initial text boxes
txtReceivedText.Text = vbNullString
txtStoreText.Text = vbNullString
' Initialize the transceiver
MSComm1.Output = "ER_CMD#R1"  ' send initiallizing string to
                              ' transceiver
Sleep (20)        ' 20ms delay
If MSComm1.Input = "ER_CMD#R1" Then  ' check if transceiver responsed
      Sleep (20)  ' 20ms delay
      MSComm1.Output = "ACK"  ' send acknowledge string to
                              ' transceiver
      MsgBox "Transceiver initialized succesfully"
Else
      MsgBox "ERROR: Transceiver not connected"
End If
Sleep (100)       ' 100ms delay
MsgBox "Press RESET button on Olimex board to start"

End Sub
```

```
'
' AIRCAT subroutines
'

' Ship's rudder control
Private Sub rudder_Change()

rudpos.Text = Val(rudder.Value)
' Send "R rudpos\r" through the serial port and LPC calls
' SetRudder(dutycycle) function
MSComm1.Output = "R" & " " & rudpos & vbCr

End Sub
```

```
' Rudder's center position -> straight ship navigation
Private Sub center_Click()

rudpos.Text = 125 ' Set rudder position to the middle on VB console
rudder.Value = rudpos.Text

End Sub
```

```
' Ship's thruster control
Private Sub thruster_Change()

thrpos.Text = Val(thruster.Value)
' Send "W thrpos\r" through the serial port and LPC calls
' SetThruster(dutycycle) function
MSComm1.Output = "W" & " " & thrpos & vbCr

End Sub
```

```
' Ship's blower control
Private Sub blower_Change()

blopos.Text = Val(blower.Value)
' Send "SV 2 ' blopos\r" through the serial port and LPC sends that
' string to the CMUcam2
MSComm1.Output = "SV" & " " & "2" & " " & blopos & vbCr

End Sub
```

```
' Ship's reverser control
Private Sub reverser_Click()

' Send "SV 3 160\r" through the serial port and LPC sends that string
' to the CMUcam2
MSComm1.Output = "SV" & " " & "3" & " " & "160" & vbCr

End Sub
```

```
' Ship's forward control
Private Sub forward_Click()

' Send "SV 3 65\r" through the serial port and LPC ' sends that
' string to the CMUcam2
MSComm1.Output = "SV" & " " & "3" & " " & "65" & vbCr

End Sub
```

```
'
' CMUcam2 subroutines
'

' Reset CMUcam2
Private Sub rs_Click()

' Send "rs\r" through the serial port and LPC sends that string to
' the CMUcam2
MSComm1.Output = "rs" & vbCr
panpos.Text = 128 ' Set pan position to the middle on VB console
pan.Value = panpos.Text
tiltpos.Text = 128 ' Set tilt position to the middle on VB console
tilt.Value = tiltpos.Text

End Sub
```

```
' Stop streaming data from camera
Private Sub stop_Click()

' Send a "\r" through the serial port to the CMUcam2
MSComm1.Output = vbCr

End Sub
```

```
' CMUcam2 pan servo control
Private Sub pan_Change()

panpos.Text = Val(pan.Value)
MSComm1.Output = "SV" & " " & "0" & " " & panpos & vbCr

End Sub
```

```
' pan servo center position
Private Sub centerpan_Click()

panpos.Text = 128 ' Set pan position to the middle on VB console
pan.Value = panpos.Text

End Sub
```

```
' CMUcam2 tilt servo control
Private Sub tilt_Change()

tiltpos.Text = Val(tilt.Value)
MSComm1.Output = "SV" & " " & "1" & " " & tiltpos & vbCr

End Sub
```

```
' tilt servo center position
Private Sub centertilt_Click()

tiltpos.Text = 128 ' Set tilt position to the middle on VB console
tilt.Value = tiltpos.Text

End Sub
```

```
'
' COMMUNICATION subroutines
'

' Clear the receiving window
Private Sub clear_Click()

txtReceivedText = vbNullString

End Sub
```

```
' Store all received data from CMUcam2 to data.xls
Private Sub store_Click()

Open "C:\data.xls" For Append As #1 'define the path
      ' make titles on columns at xls sheet
      Print #1, "tp" & vbTab & "pn" & vbTab & "time"
      ' store the received data
      Print #1, txtStoreText
Close #1

End Sub
```

```
' Track color
Private Sub tc_Click()

' Send a "TC\r" through the serial port to the CMUcam2
MSComm1.Output = "TC" & vbCr
txtStoreText = vbNullString   ' Clear the receiving window
Label3.Caption = " tp" & "    " & "pn" & "      " & "time"
Label3.Alignment = 0
Label3.FontBold = True

End Sub
```

```
' Reset LPC
Private Sub reset_Click()

' Send "O" through the serial port and LPC calls main() function ->
' the program starts from the beginning
MSComm1.Output = "O"
rudpos.Text = 125 ' Set rudder position to the middle on VB console
rudder.Value = rudpos.Text
thrpos.Text = 94 ' Set thruster position to the idle on VB console
thruster.Value = thrpos.Text
blopos.Text = 128 ' Set blower position to the idle on VB console
blower.Value = blopos.Text
panpos.Text = 128 ' Set pan position to the middle on VB console
pan.Value = panpos.Text
tiltpos.Text = 128 ' Set tilt position to the middle on VB console
tilt.Value = tiltpos.Text

End Sub
```

```
' The send button is pressed when the user wants to send text through
' the serial port
Private Sub send_Click()

If (Len(txtSendText) > 0) Then ' make sure that there is text to send
      ' make sure the serial port was initialized
      If (MSComm1.PortOpen = True) Then
            ' send the text as well as a carriage return appended to
            ' the end
            MSComm1.Output = txtSendText & vbCr
      Else
            MsgBox "ERROR: serial port not initialized"
      End If
Else
      MsgBox "ERROR: there is no text to send"
End If

' after sending clear the text out of the input box
txtSendText = vbNullString

End Sub
```

E. ΚΩΔΙΚΑΣ MATLAB

E.1 find_c.m

```
close all;
clear all;
clc;

op_1   %
op_2   % load the model
op_3   %

c0 = [20 0.49 0.24]; % Set initial values

options = optimset('LargeScale','on','Display','iter',
                   'MaxFunEvals',400,'TolX',1e-6,'TolFun',1e-6);

c = fminsearch('c_fun',c0,options);

% put variables back in the base workspace

c

% lusi variable step (ode45)
% c = [c1m c1V c2] = [19.9694 0.4872 0.2443]
```

E.2 c_fun.m

```
function F = c_fun(c)

c1m = c(1); % Move variables into model parameter names
c1V = c(2);
c2 = c(3);

% Choose solver and set model workspace to this function
opt = simset('solver','ode45','SrcWorkspace','Current');

%=======================
%Peirama 1 - Simulation 1
%=======================

[t1,x1,y1]=sim('op_1',[0 20],opt);  % trexw to modelo op_1
vcr=max(y1(:,2));                   % vriskw ti megisti timi tis
                                    % taxititas

%end Peirama 1

%=======================
%Peirama 2 - Simulation 2
%=======================

[t2,x2,y2]=sim('op_2',[0 20],opt);  % trexw to modelo op_2
[m2,n2]=size(t2);                   % vriskw ti diastasi tou pinaka
                                    % tou xronou o opoios einai
                                    % 'pinakas stili' = dianusma
psi=y2(:,3);                        % onomazw ton pinaka ths gwnias
                                    % psi

% Elegxw an ta stoixeia tou dianusmatos tis gwnias einai stin krisimi
% timi 6,28
for (i2=1:m2)                       % diavazw ola ta stoixeia tou
                                    % dianismatos psi
      if(6<psi(i2)&psi(i2)<6.5)     % elegxw an to trexon stoixeio
                                    % tou psi vriskete metaxy 6,0 kai
                                    % 6,5
            kiklos=i2;              % an vriskete stin krisimi timi,
                                    % apothikeuw ton deikti tou
                                    % stoixeiou
      end
end
t_2=t2(kiklos);   % vriskw ton apoluto xrono pou antistoixei sti
                  % simplirwsi enos kiklou strofis
tcr=t_2-3;        % vriskw ti diarkeia tou kuklou strofis, afou ta
                  % pidalia apoktoun klisi meta ta 3sec

%end Peirama 2
```

```
%========================
%Peirama 3 - Simulation 3
%========================

[t3,x3,y3]=sim('op_3',[0 20],opt);  % trexw to modelo op_3
[m3,n3]=size(t3);                   % vriskw ti diastasi tou pinaka
                                    % tou xronou o opoios einai
                                    % 'pinakas stili' = dianusma
v0=y3(:,2);                         % onomazw ton pinaka ths
                                    % taxutitas v0
t0=t3;                              % onomazw ton pinaka tou xronou
                                    % t0

% Elegxw an ta stoixeia tou dianusmatos tis taxutitas einai panw apo
% tin krisimi timi 0
for (i3=1:m3)                   % diavazw ola ta stoixeia tou
                                % dianismatos psi
      if(v0(i3)>=0.2)           % elegxw an to trexon stoixeio tou v0
                                % vriskete panw apo to orio 0
            stamatima=i3;       % an vriskete panw apo tin krisimi
                                % timi, apothikeuw ton deikti tou
                                % stoixeiou
      end
end                             % teleiwnontas to loop i metavliti
                                % 'stamatima' periexei to deikti pou
                                % antistoixei sti zitoumeni timi tis v
                                % (ligo panw apo to 0,2)
a_3st=y3(stamatima,1);          % vriskw tin sunoliki apostasi pou
                                % dianuei to skafos apo tin arxi tis
                                % exomiwsis

% Elegxw an ta stoixeia tou dianusmatos tou xronou einai katw apo ta
% 3sec
for (j3=1:m3)                   % diavazw ola ta stoixeia tou
                                %dianismatos psi
      if(t0(j3)<3.01)           % elegxw an to trexon stoixeio tou t0
                                % vriskete katw apo ta 3sec
            ekinisi=j3;         % an vriskete katw apo ta 3 sec,
                                % apothikeuw ton deikti tou stoixeiou
      end
end                             % teleiwnontas to loop i metavliti
                                % 'ekinisi' periexei to deikti pou
                                % antistoixei sta 3 sec
a_3ek=y3(ekinisi,1);            % vriskw tin apostasi pou dianuei to
                                % skafos kata ta prwta 3sec
acr=a_3st-a_3ek;                % vriskw tin apostasi pou dianuei to
                                % skafos apo ti stigmi pou svinoun ta
                                % thrusters mexri na stamatisei
                                % (v=0,2 m/sec)

%end Peirama 3

ap=5.85;    % m
vp=3.448;   % m/s
tp=7.6;     % s

F=(ap-acr)^2+(vp-vcr)^2+(tp-tcr)^2;
```

E.3 modelo.m

```
function [sys,x0,str,ts] = modelo(t,x,u,flag,c1m,c1V,c2)

%=============================================================================
%   Auto to modelo apeikonizei tin kinisi tou ploiou AIRCAT. To
%   dianusma katastasis tou systimatos einai:
%
%   x = [x1 x2 x3]'
%   x1 = a
%   x2 = a'
%   x3 = psi
%
%   Oi exiswseis katastasis tou systimatos einai:
%
%   x' = [x1' x2' x3']
%   x1' = a' = x2
%   x2' = c1m/x2*u1*(cos(x3))^2
%         -x2*((-tan(x3)*x2/x1+c2*x2/cos(x3)*u2)*tan(x3)
%         +c1V*abs(x2/cos(x3)))
%   x3' = psi' = -a'/a*tan(psi)+c2*u2 = -x2/x1*tan(x3)+c2*u(2)
%
%   Sti sunexeia parousiazontai oi sumvolismoi pou xrisimopoiountai
%   sto paron M-file gia tis anwterw metavlites.
%
%   Dianusma katastasis:
%   x = [x1 x2 x3]'
%   x1 = x(1)
%   x2 = x(2)
%   x3 = x(3)
%
%   Eisodos:
%   u = [u1 u2]'
%   u1 = u(1)
%   u2 = u(2)
%
%   Statheres:
%   c = [c1m c1V c2]
%   c1m = c(1)
%   c1V = c(2)
%   c2 = c(3)
%
%   Based on CSFUNC.
%   See sfuntmpl.m for a general S-function template.
%   See also SFUNTMPL.
%   Copyright 1990-2002 The MathWorks, Inc.
%   $Revision: 1.9 $
%=============================================================================
```

```
switch flag,

      %%%%%%%%%%%%%%%%%%
      % Initialization %
      %%%%%%%%%%%%%%%%%%
      case 0,
            [sys,x0,str,ts]=mdlInitializeSizes(c1m,c1V,c2);

      %%%%%%%%%%%%%%%
      % Derivatives %
      %%%%%%%%%%%%%%%
      case 1,
            sys=mdlDerivatives(t,x,u,c1m,c1V,c2);

      %%%%%%%%%%
      % Outputs %
      %%%%%%%%%%
      case 3,
            sys=mdlOutputs(t,x,u,c1m,c1V,c2);

      %%%%%%%%%%%%%%%%%%%
      % Unhandled flags %
      %%%%%%%%%%%%%%%%%%%
      case { 2, 4, 9 },
            sys = [];

      %%%%%%%%%%%%%%%%%%%
      % Unexpected flags %
      %%%%%%%%%%%%%%%%%%%
      otherwise
            error(['Unhandled flag = ',num2str(flag)]);

end
% end csfunc

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the
% S-function.
%=============================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes(c1m,c1V,c2)

sizes = simsizes;
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [-80 0.00001 0];  % Edw thetoume tis arxikes sintikes A.S.
                        % [a a' psi]
str = [];
ts  = [0 0];

% end mdlInitializeSizes
```

```
%
%=====================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=====================================================================
%
function sys=mdlDerivatives(t,x,u,c1m,c1V,c2)

h=1e-8;     % Elaxisti taxitita gia na min apeirizode kapoioi oroi
            % [m/s]
Tmax=44;    % Megisti wsi twn 2 waterjet = 2*22 [N]
ms = 9;     % Maza skafous AIRCAT [kg]

sys(1)=x(2);

if(x(2)/cos(x(3))<h)
sys(2)=c1m/h*u(1)*cos(x(3))-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*h*u(2))*tan(x(3))+c1V*abs(h));

elseif((c1m/x(2)*cos(x(3))*u(1))>Tmax)
sys(2)=Tmax/ms*cos(x(3))-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2))*tan(x(3))+c1V*abs(x(2)/co
s(x(3))));

else
sys(2)=c1m/x(2)*u(1)*(cos(x(3)))^2-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2))*tan(x(3))+c1V*abs(x(2)/co
s(x(3))));

end

sys(3)=-tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2);

% end mdlDerivatives

%
%=====================================================================
% mdlOutputs
% Return the block outputs.
%=====================================================================
%
function sys=mdlOutputs(t,x,u,c1m,c1V,c2)

sys(1)=x(1);
sys(2)=x(2);

%
% Kanwntas ton parakatw elegxo parousiazete i gwnia sto scope tou
% modelou "op_2" sto diastima: -pi<psi<pi. An den kanw afto ton
% elegxo i gwnia parousiazete sto diastima: 0<psi<+oo. Analoga pws
% thelw na xrisimopoiisw ta noumera rithmizw afto to kommati kwdika.
%
%if (-pi<mod(x(3),2*pi)<=pi)
%x(3)=mod(x(3),2*pi);
%else
%x(3)=-sign(mod(x(3),2*pi))*(2*pi-abs(mod(x(3),2*pi)));
%end

sys(3)=x(3);

% end mdlOutputs
```

E.4 modelo1.m

```
function [sys,x0,str,ts] = modelo1(t,x,u,flag)

switch flag,

      %%%%%%%%%%%%%%%%%%
      % Initialization %
      %%%%%%%%%%%%%%%%%%
      case 0,
            [sys,x0,str,ts]=mdlInitializeSizes();

      %%%%%%%%%%%%%%%
      % Derivatives %
      %%%%%%%%%%%%%%%
      case 1,
            sys=mdlDerivatives(t,x,u);

      %%%%%%%%%%
      % Outputs %
      %%%%%%%%%%
      case 3,
            sys=mdlOutputs(t,x,u);

      %%%%%%%%%%%%%%%%%%%%
      % Unhandled flags %
      %%%%%%%%%%%%%%%%%%%%
      case { 2, 4, 9 },
            sys = [];

      %%%%%%%%%%%%%%%%%%%%
      % Unexpected flags %
      %%%%%%%%%%%%%%%%%%%%
      otherwise
            error(['Unhandled flag = ',num2str(flag)]);

end
% end csfunc
```

```
%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-
function.
%=============================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes()

sizes = simsizes;
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [-11 0.00001 0.337];    % Edw thetoume tis arxikes sintikes
                              % A.S. [a a' psi]
str = [];
ts  = [0 0];

% end mdlInitializeSizes
```

```
%
%=========================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=========================================================================
%
function sys=mdlDerivatives(t,x,u)

h=1e-8;     % Elaxisti taxitita gia na min apeirizode kapoioi oroi
            % [m/s]
Tmax=44;    % Megisti wsi twn 2 waterjet = 2*22 [N]
ms = 9;     % Maza skafous AIRCAT [kg]

% Statheres c = [c1m c1V c2]
c1m =  19.9694;
c1V = 0.4872;
c2 = 0.2443;

sys(1)=x(2);

if(x(2)/cos(x(3))<h)
sys(2)=c1m/h*u(1)*cos(x(3))-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*h*u(2))*tan(x(3))+c1V*abs(h));

elseif((c1m/x(2)*cos(x(3))*u(1))>Tmax)
sys(2)=Tmax/ms*cos(x(3))-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2))*tan(x(3))+c1V*abs(x(2)/co
s(x(3))));

else
sys(2)=c1m/x(2)*u(1)*(cos(x(3)))^2-x(2)*((-
tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2))*tan(x(3))+c1V*abs(x(2)/co
s(x(3))));

end

sys(3)=-tan(x(3))*x(2)/x(1)+c2*x(2)/cos(x(3))*u(2);

% end mdlDerivatives

%
%=========================================================================
% mdlOutputs
% Return the block outputs.
%=========================================================================
%
function sys=mdlOutputs(t,x,u)

sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

% end mdlOutputs
```

ΣΤ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΚΕΥΩΝ

ΣΤ.1 Olimex LPC-P2106 prototype board specifications

Features:

- MCU: 16/32 bit LPC2106 with 128K Bytes Program Flash, 64K Bytes RAM, RTC, 2xUARTs, I2C, SPI, 2x 32bit TIMERS, 7xCCR, 6x PWM, WDT, 5V tolerant I/O, up to 60MHz operation
- standard JTAG connector with ARM 2x10 pin layout for programming/debugging with ARM-JTAG
- push BUTTON with pullup
- status LED
- two on board voltage regulators 1.8V and 3.3V with up to 800mA current
- Power plug-in jack
- single power supply: +5-9VAC/DC required
- power supply filtering capacitor
- RS232 interface circuit
- RESET circuit
- RESET button
- DEBUG jumper for JTAG enable
- BSL jumper for Bootloader enable
- RTCK pullup resistor
- 14.7456 Mhz crystal allow easy communication setup (4 x PLL = 58,9824 Mhz CPU clock)
- extension headers for all uC ports
- PCB: FR-4, 1.5 mm (0,062"), green soldermask, white silkscreen component print
- prototype PCB area with +3.3V and GND bus
- Four mounting holes
- Dimensions: 77x100 mm (3.9x3.05")

Supported devices:
Philips Semiconductors Inc. LPC2106 16/32 bit ARM7TDMI-S™

JTAG interface:
The JTAG connector is 2x10 pin with 0,1" step and ARM recommended JTAG layout. PIN.1 is marked with square pad on bottom and arrow on top.

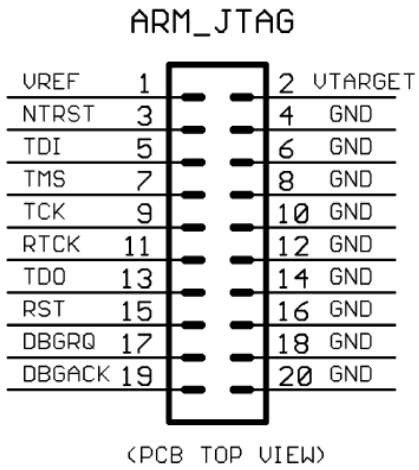
Note: *To enable JTAG interface DBG jumper should be shorted at the time of POWER UP.*

Important: *When JTAG is enabled P0.18-P1.31 ports take their JTAG alternative function no matter of PINSEL register value, so during debugging with JTAG these ports are not available for the user program.*

JTAG signals description:

- **PIN.1 (VTREF)** Target voltage sense. Used to indicate the target's operating voltage to the debug tool.
- **PIN.2 (VTARGET)** Target voltage. May be used to supply power to the debug tool.
- **PIN.3 (nTRST)** JTAG TAP reset, this signal should be pulled up to Vcc in target board.
- **PIN4,6,8,10,12,14,16,18,20** Ground. The Gnd-Signal-Gnd-Signal strategy implemented on the 20-way connection scheme improves noise immunity on the target connect cable.
- **PIN.5 (TDI)** JTAG serial data in, should be pulled up to Vcc on target board.
- **PIN.7 (TMS)** JTAG TAP Mode Select, should be pulled up to Vcc on target board.
- **PIN.9 (TCK)** JTAG clock.
- **PIN.11 (RTCK)** JTAG re-timed clock. Implemented on certain ASIC ARM implementations the host ASIC may need to synchronize external inputs (such as JTAG inputs) with its own internal clock.
- **PIN.13 (TDO)** JTAG serial data out.
- **PIN.15 (nSRST)** Target system reset.
- **PIN.17 (DBGRQ)** Asynchronous debug request. DBGRQ allows an external signal to force the ARM core into debug mode, should be pull down to GND.
- **PIN.19 (DBGACK)** Debug acknowledge signal. The ARM core acknowledges debug-mode in response to a DBGRQ input.

JTAG connector layout:



Power supply:

Power supply is made with two LDO adjustable voltage regulators LM1117. Input voltage should be in range 5-9VAC/DC.

RS232 interface:

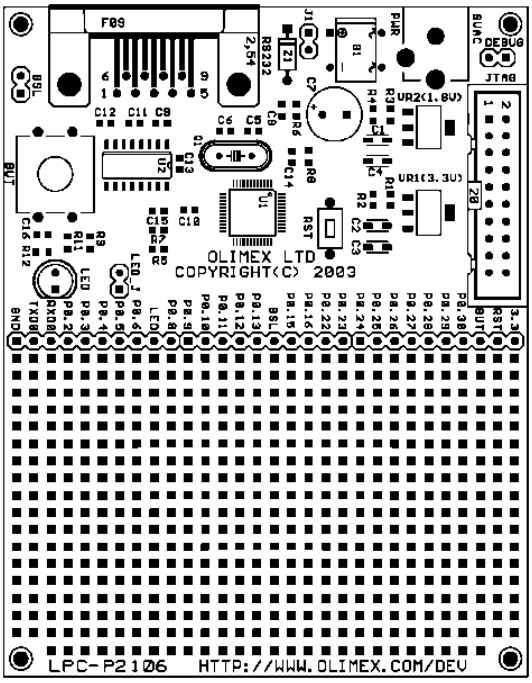
LPC2106 have two RS232 channels. Only Channel 0 is via MAX3232 IC to SUB D 9 pin connector. Channel 0 with TXD0 and RXD0 is used by the Bootloader program to program LPC2106 Flash memory without external programmer. Channel 1 is general purpose RS232 channel and may be used by user program.

RESET:
Reset circuit is made by simple external RC group. There is possibility to apply RESET externally by the small RESET pushbutton on the board.

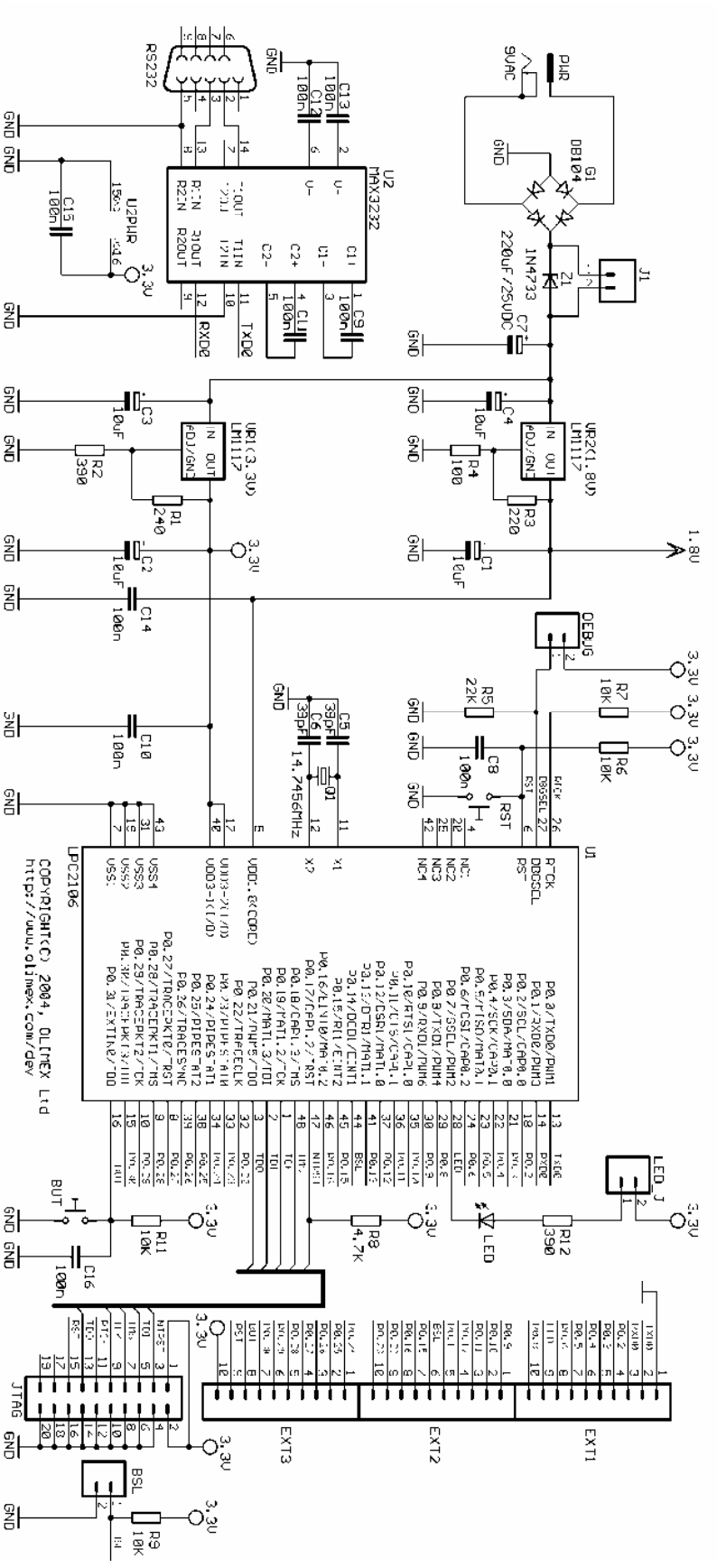
Oscillator:
14.7456 Mhz crystal is used for LPC2106 as it allows easy setup on any communication speed This makes programming with Philips ISP utility possible at any speed up to 115Kbps.

Bootloader:
The Bootloader program is enabled when BSL jumper is shorted at time of power up. In this case Bootloader takes the program control and user may download Flash memory with Philips ISP programming utility. Note that if you want to run code in Flash memory BSL jumper should be open at time of power up, otherwise Bootloader will stay in control and will not allow program in Flash to run.

Board layout:



Ordering codes:
LPC-P2106 - assembled and tested with LPC2106 microcontroller



See also LPC2106 User Manual.

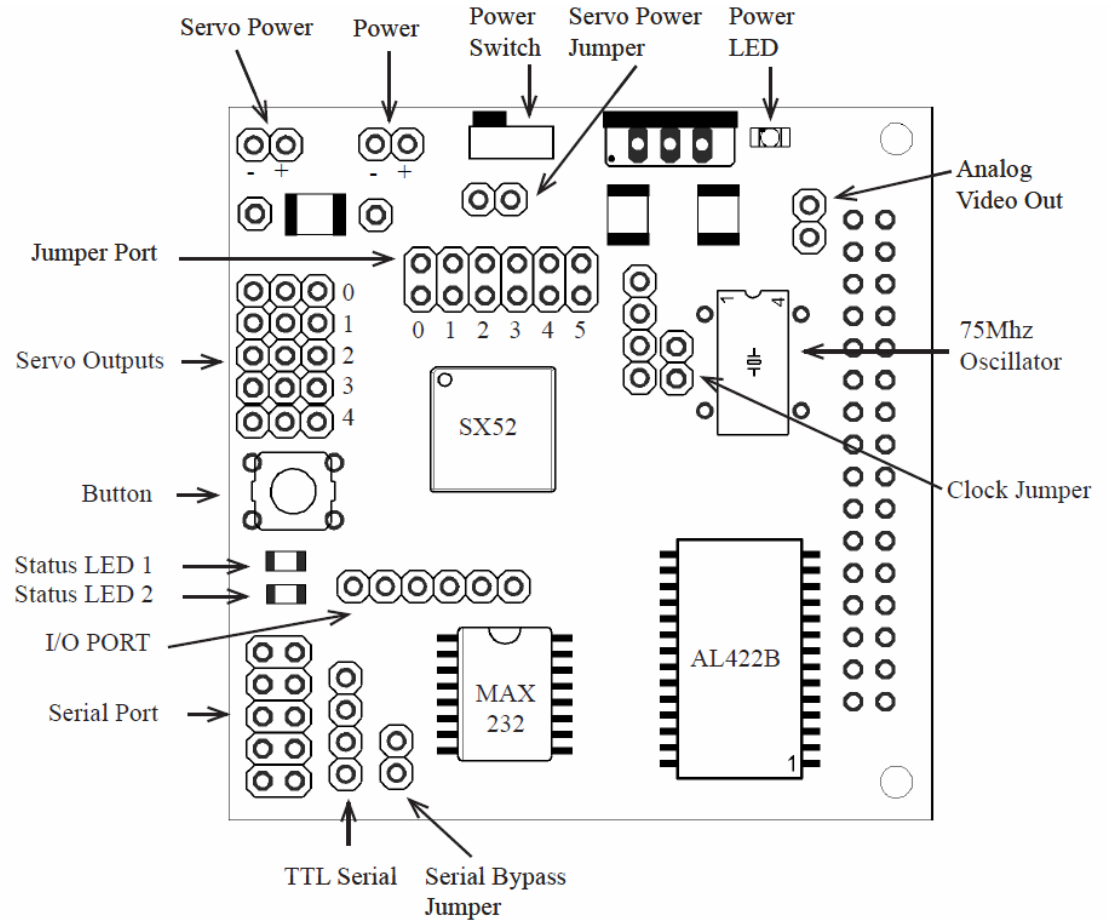
ΣT.2 CMUcam2 vision sensor specifications

The CMUcam2 consists of a SX52 microcontroller (http://www.ubicom.com/products/sx/sx.html) interfaced with an OV6620 or OV7620 Omnivision CMOS camera (http://www.ovt.com) on a chip that allows simple high level data to be extracted from the camera's streaming video. The board communicates via a RS-232 or a TTL serial port and has the following functionality:

- Track user defined color blobs at up to 50 Frames Per Second*
- Track motion using frame differencing at 26 Frames Per Second
- Find the centroid of any tracking data
- Gather mean color and variance data
- Gather a 28 bin histogram of each color channel
- Manipulate Horizontally Pixel Differenced Images
- Transfer a real-time binary bitmap of the tracked pixels in an image
- Arbitrary image windowing
- Adjust the camera's image properties
- Dump a raw image (single or multiple channels)
- Up to 160 x 255 Resolution**
- Supports Multiple Baudrates: 115,200 57,600 38,400 19,200 9,600
- 4,800 2,400 1,200
- Control 5 servo outputs
- Slave parallel image processing mode off of a single camera bus
- Automatically use servos to do two axis color tracking
- B/W Analog video output (PAL or NTSC)**
- Flexible output packet customization
- Multiple pass image processing on a buffered image
- Works with the OV7620 or OV6620 module

*Frame Rate Depends on Window Size
**Camera Properties Depend on Camera Module

Board layout:



See also CMUvam2 User Manual and Omnivision OV7620 datasheet.

ΣT.3 Wireless telemetry data modules specifications

Low Power Radio Solutions (LPRS) – easy radio

General features:

- Crystal controlled synthesizer for frequency accuracy
- High sensitivity receiver – typically –103dBm @ 19.2 Kbps
- Up to 10mW Transmit Power (at 434MHz)
- Low operating Voltage – 2.5-5.5 Volts – Single Lithium Cell
- Low power consumption: Receiver - 21mA
  Transmitter - 25mA
  Sleep – 120μA (V2.01.6 + later)
- User programmable: Frequency of operation
  Data Rate
  Output Power

Applications:

- Handheld Terminals
- Environmental Sense & Control
- Vehicle to Base Station Data Transfer
- Remote Data Acquisition
- Electronic Point of Sale equipment
- Etc

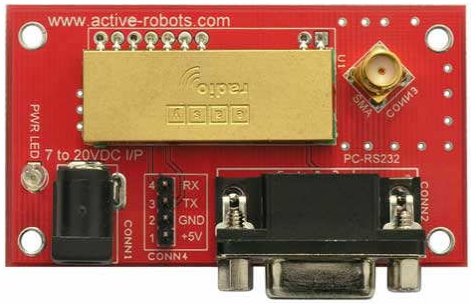
The transceiver board is a simple to use bi-directional wireless RF serial link suitable for use in a wide variety of robotics, embedded control and data-logging applications.

Mini RF telemetry module:

User selectable (UK/EU):

- 10 spot frequencies
- Power output 1-10mW
- Range: Upto 250m line of sight
- Baud rates from 2400 to 38400
- Free configuration software
- RS232 PC Lead (Supplied)
- Straight Antenna (Supplied)
- Battery Snap Lead (Supplied)
- Input power 8.3 to 30Vdc
- Size: 53mm x 20mm

RS232 RF telemetry module:



User selectable (UK/EU):

- 10 spot frequencies
- Power output 1-10mW
- Range: Upto 250m line of sight
- Baud rates from 2400 to 38400
- Free configuration software
- Straight Antenna (Supplied)
- DB9 Cable (Optional)
- Input power 7.0 to 20Vdc
- Size: 70mm x 40mm

See also LPRS User Manual.

ΣT.4 RS232 adapter and LM317 voltage regulator

For RS232 adapter refer to MAXIM MAX232 datasheet.

For LM317 voltage regulator refer to LM317 datasheet.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Gadre, D. V., *Προγραμματίζοντας τον Μικροελεγκτή AVR*, Τζιόλα, 2001
- Kernighan, B. W., Ritchie D. M., *Η Γλώσσα Προγραμματισμού C*, Κλειδάριθμος, δεύτερη έκδοση, 2002
- Lynch, J. P., *ARM Cross Development with Eclipse (Tutorial)*, third version, 2005
- Martin, T., *The Insider's Guide to the Philips ARM7 Based Microcontrollers*, Hitex, 2006
- Petroutsos, E., *Mastering Visual Basic 6*, Sybex, 1998
- Young, H. D., *Φυσική (Τόμος Β') Ηλεκτρομαγνητισμός – Οπτική – Σύγχρονη Φυσική*, Παπαζήση, 1994
- Αθανασούλης, Γ. Α., *Δυναμική Συμπεριφορά Πλοίου σε Κυματισμούς και Πηδαλιουχία Πλοίου (Σημειώσεις)*, ΕΜΠ, 2005
- Ζαραφωνίτης, Γ. Ν., *Εισαγωγή στη Ναυπηγική και τη Θαλάσσια Τεχνολογία (Σημειώσεις)*, ΕΜΠ, 2002
- Ιωαννίδης, Ι.Π., *Ναυτικές Μηχανές – Τεύχος 1*, ΕΜΠ. τρίτη έκδοση, 2005
- Κρικέλης, Ν. Ι., *Εισαγωγή στον Αυτόματο Έλεγχο (Θεωρία και Εφαρμογές)*, Συμμετρία, τρίτη έκδοση, 2000
- Μπακόπουλος, Α., Χρυσόβεργης, Ι., *Εισαγωγή στην Αριθμητική Ανάλυση*, Συμεών,1999
- Ξηρός, Ν. Ι., *Ειδικά Συστήματα Ελέγχου Πλοίου*, ΕΜΠ, 2005
- Ξηρός, Ν. Ι., *Συμπληρωματικές Σημειώσεις Αυτομάτου Ελέγχου για Ναυπηγούς Μηχανολόγους*, ΕΜΠ, 2004
- Ταρουδάκης, Μ. Ι., *Συστήματα Ναυσιπλοΐας*, ΕΜΠ,1990

WEBLIOGRAPHY

- http://en.wikipedia.org/wiki/Dormand_Prince
- http://el.wikipedia.org/wiki/Global_Positioning_System
- http://en.wikipedia.org/wiki/Navigation
- http://en.wikipedia.org/wiki/Nickel_metal_hydride_battery
- http://en.wikipedia.org/wiki/Surface_effect_ship
- http://www.keil.com/
- http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/