ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Θέμα: Ανάπτυξη συστήματος ρομποτικής πλοήγησης αυτόνομου μικρού αερομοντέλου

Φοιτητές: Πέτρου Φώτης
Παλώγος Αποστόλης

Εισηγητής: Καββουσανός Μανόλης

ΗΡΑΚΛΕΙΟ ΚΡΗΤΗΣ, ΦΕΒΡΟΥΑΡΙΟΣ 2007

Πρόλογος

Η μεγάλη εξέλιξη της τεχνολογίας στις μέρες μας έχει σαν αποτέλεσμα να παρουσιάζονται συνεχώς νέα επιτεύγματα ενώ παράλληλα δίνονται εναύσματα για καινούργιες ανακαλύψεις σε διάφορους τομείς. Στο χώρο της ρομποτικής, έχουν γίνει κατά καιρούς μεγάλα άλματα μέσω εφαρμογών που καθιερώνονται ως τα πιο χρήσιμα εργαλεία στον άνθρωπο. Πολλές από αυτές τις εφαρμογές έκαναν πράγματα με εντυπωσιακή δύναμη και ταχύτητα, και με όλο και καλύτερη ποιότητα. Κατά καιρούς μαθαίνουμε για ολοένα και περισσότερες νέες τεχνολογικές εφαρμογές όπως ρομποτικούς βραχίονες στην ιατρική και στη βιομηχανία, ρομπότ διαστημικών αποστολών, ρομπότ με ανθρωπόμορφη κίνηση, ρομποτικά κατοικίδια, οικιακά ρομπότ, ρομποτικά οχήματα μέχρι και ιπτάμενα ρομπότ.

Μέσα στα πλαίσια του Τ.Ε.Ι. ασχοληθήκαμε με τις βασικές αρχές της ρομποτικής και γενικά με τον σχεδιασμό και ανάπτυξη ρομποτικών συστημάτων. Αντικείμενο της πτυχιακής εργασίας που βρίσκεται στα χέρια σας είναι η ρομποτική πλοήγηση ενός μικρού αυτόνομου αεροπλάνου. Στόχος είναι η έρευνα και η μελέτη του αντικείμενου και παράλληλα η υλοποίηση αυτού σε σχέση πάντα με τις δυνατότητες που μας παρέχονται.

Στα επόμενα κεφάλαια θα έρθουμε σε επαφή με κάποιες βασικές αρχές του αερομοντελισμού που χρειάστηκαν για το γνωστικό επίπεδο της πτυχιακής. Στη συνέχεια παρουσιάζουμε την υλοποίηση του συστήματος αρχίζοντας από τα επιμέρους εργαλεία και συσκευές που χρησιμοποιήθηκαν και ακολούθως με την σχεδίαση του κυκλώματος. Επίσης αναλύουμε την μαθηματική θεώρηση που στηρίχθηκε ο ρομποτικός έλεγχος και τέλος την ανάλυση του κώδικα του τελικού προγράμματος.

Σ’ αυτό το σημείο θα θέλαμε να ευχαριστήσουμε τον κύριο Καββουσσανό Μανόλη αρχικά για την ευκαιρία που μας έδωσε να ασχοληθούμε με ένα θέμα τόσο πρωτόγνωρο για εμάς όσο και γοητευτικό και εν συνεχεία για τις πολύτιμες συμβουλές και γνώσεις που μας παρείχε κατά την εκπόνηση της εργασίας. Επίσης θα θέλαμε να εκφράσουμε τις ιδιαίτερες ευχαριστίες μας στον κύριο Θανάση Τουτουντζή για τις χρήσιμες συμβουλές και τη βοήθειά του στη κατασκευή του κυκλωματικού μέρους καθώς και τον κύριο Τοτό Αντζολετάκη για την πολλαπλή συνεισφορά του και την αμέριστη παροχή των μοντελιστικών του γνώσεων.

Θα ήταν μεγάλη παράλειψη να μην αναφέρουμε τα άτομα τα οποία παρακολούθησαν την ανάπτυξη του θεωρητικού αλλά και του πρακτικού μέρους της εργασίας και έμπρακτα συμπαραστάθηκαν καθ’ όλη τη διάρκειά της. Συγκεκριμένα τους συμφοιτητές και φίλους Σιάτρα Βασίλη, Πανάγο Χρήστο, Βλάμη Μόρφω, Περώνα Θανάση και Κοντύλη Παναγιώτη.

Σύντομη Περιγραφή Πτυχιακής Εργασίας

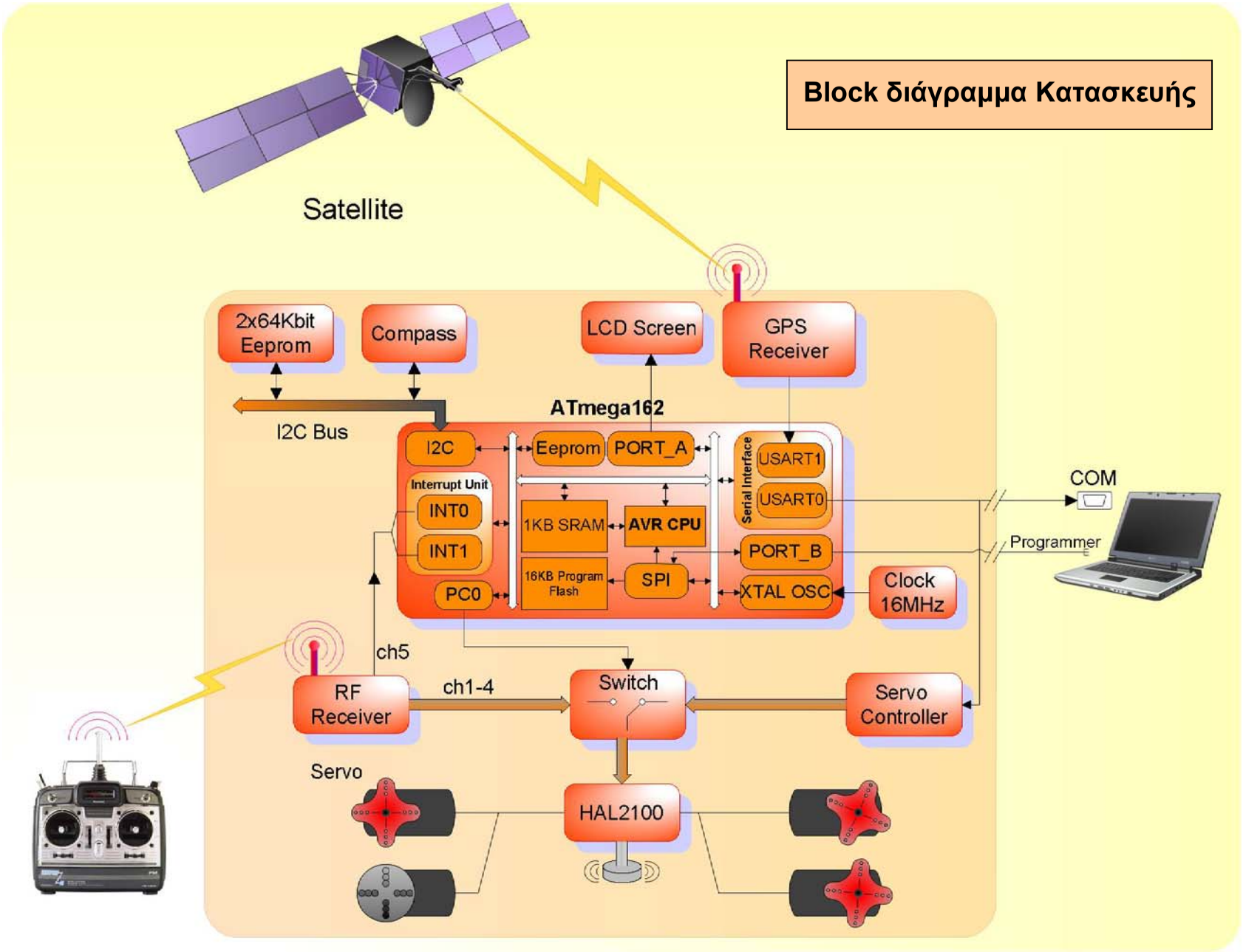
Πρόκειται για ένα σύστημα το οποίο έχει τη δυνατότητα να πλοηγηθεί αυτόνομα σε ένα χώρο μεγάλης απόστασης, πάνω σε μια προϋπολογισμένη πορεία. Στόχος της εργασίας είναι η σταθεροποίηση ενός μοντέλου αεροπλάνου πάνω στη πορεία αυτή, κάτι που προϋποθέτει τον ακριβή έλεγχο βάσει των αισθητηρίων οργάνων. Ο έλεγχος της θέσης του αερομοντέλου γίνεται με χρήση του G.P.S. όπου ένα σύμπλεγμα 24 δορυφόρων μας δίνει τη δυνατότητα με την βοήθεια ενός δέκτη να γνωρίζουμε την θέση μας πάνω στη γη με ακρίβεια 20 m. Επίσης γίνεται χρήση ηλεκτρονικής πυξίδας με την οποία μπορούμε να γνωρίζουμε ανά πάσα στιγμή την κατεύθυνση του αερομοντέλου. Αυτές οι πληροφορίες συλλέγονται μέσω πρωτοκόλλων (Serial, I2C) σε ένα κεντρικό μικροεπεξεργαστή, ο οποίος είναι υπεύθυνος για τον υπολογισμό θέσης, τις αποφάσεις και τον έλεγχο του συστήματος.

Το αερομοντέλο που χρησιμοποιήθηκε διαθέτει πομπό και δέκτη RF για την τηλεκατεύθυνσή του. Για τους σκοπούς της εργασίας υλοποιήθηκε σύστημα με το οποίο γίνεται δυνατή η μεταγωγή της ρομποτικής κατάστασης σε τηλεκατευθυνόμενη και εναλλάξ.

Στο σύστημα επίσης προστέθηκε διεπαφή σύνδεσης με υπολογιστή με την οποία είμαστε σε θέση να προγραμματίσουμε νέα, κάθε φορά, πορεία, αλλά και να λάβουμε διάφορες πληροφορίες που συλλέγονται κατά την πτήση. Ειδικό λογισμικό κατασκευασμένο σε γλώσσα προγραμματισμού Visual Basic αναλαμβάνει την επικοινωνία αυτή μέσω σειριακής σύνδεσης με το αερομοντέλο όταν αυτό βρίσκεται στο έδαφος.

Λόγω της πολυπλοκότητας του χειρισμού που χρειάζεται ένα αεροπλάνο κατά την προσγείωση και την απογείωσή του, στην εργασία αυτή δεν αναπτύχθηκε ρομποτικός έλεγχος κατά τις καταστάσεις αυτές. Το αερομοντέλο είναι σε θέση να ακολουθήσει πορεία οδηγούμενο από συντεταγμένες αφού αυτό βρεθεί εν πτήση. Απογείωση και προσγείωση γίνονται χειροκίνητα μέσω της τηλεκατεύθυνσης και του συστήματος εναλλαγής και πάντα κάτω από τα χέρια ενός έμπειρου αερομοντελιστή.

Block διάγραμμα Κατασκευής



Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ 7

1.1 Ρομποτικη αεροχηματων στις μερες μας 7
1.2 Φαεθων 11

2 ΑΕΡΟΜΟΝΤΕΛΙΣΜΟΣ 12

2.1 Γενικα 12
2.1.1 Τι ονομάζουμε Αερομοντελισμό; 12
2.1.2 Τηλεκατευθυνόμενα 13
2.1.3 Πόσο ψηλά πετάνε, πόσο ζυγίζουν, για πόσο πετάνε κτλ. 13
2.1.4 Πώς φτιάχνεται ένα αερομοντέλο; 14
2.1.5 Άλλες μορφές αερομοντελισμού 14
2.2 Θεωρια πτησης 15
2.2.1 Βασικά 15
2.2.2 Πιο ειδικά 15
2.2.3 Τι είναι το stall 17
2.3 Σκαφος - Κατασκευη 18
2.3.1 Βασικά 18
2.3.2 Συναρμολόγηση και Ζύγισμα 19
2.3.3 Σύστημα προσγείωσης 20
2.4 Μηχανη (αεροπλανα) 21
2.4.1 Γενικά 21
2.4.2 Θερμικοί κινητήρες 21
2.5 Συστηματα Τηλεκατευθυνσης 23
2.6 Σερβο (servo) 25
2.7 Το δικο μας αερομοντελο 26
2.7.1 Λειτουργία Τηλεκατεύθυνσης 28

3 ΕΠΙΜΕΡΟΥΣ ΕΡΓΑΛΕΙΑ(MODULES) 31

3.1 Ο μικροελεγκτης MR162 31
3.1.1 Η θύρα σειριακής μετάδοσης δεδομένων (USART) 32
3.1.2 Timers 32
3.1.3 Προγραμματισμός 32
3.2 GPS(Global Positioning System) 33
3.2.1 Το GPS-PS1E 36
3.2.2 Πρωτόκολλο NMEA 36
3.3 Πρωτοκολλο I2C 38
3.3.1 Η πυξίδα CMPS03 39
3.3.2 Η μνήμη Eeprom 41
3.4 Lcd οθονη PC1602-D 44
3.5 Το HAL 2100 44

4 ΣΧΕΔΙΑΣΜΟΣ ΚΥΚΛΩΜΑΤΟΣ 46

4.1 Το σχεδιαστικο προγραμμα Eagle 47
4.2 Κατασκευη βιβλιοθηκων στο Eagle 52
4.3 Το κυκλωμα εναλλαγης AUTO-MANUAL 56
4.4 Το κυκλωμα 59

5 ΛΟΓΙΚΗ ΠΤΗΣΗΣ – ΡΟΜΠΟΤΙΚΟΣ ΕΛΕΓΧΟΣ 65

5.1 ΑΥΤΟΕΝΤΟΝΤΟΠΙΣΜΟΣ ΘΕΣΗΣ 65
5.1.1 Σύστημα γεωγραφικών συντεταγμένων 65
5.1.2 NMEA GGA 66
5.1.3 Περιορισμοί συντεταγμένων 67
5.2 ΜΑΘΗΜΑΤΙΚΟΣ ΕΛΕΓΧΟΣ 69
5.2.1 Μέθοδος ελαχίστων τετραγώνων 70
5.2.2 Εύρεση διανύσματος κίνησης 72
5.2.3 Εντοπισμός γωνίας – απόστασης 73

6 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ 75

6.1 ΜΙΚΡΟΕΛΕΓΚΤΗΣ 75
6.2 FAETHWN.EXE 82

7 ΣΥΜΠΕΡΑΣΜΑΤΑ 84

7.1 ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ 84
7.2 ΠΡΟΤΑΣΕΙΣ 86

ΒΙΒΛΙΟΓΡΑΦΙΑ 87

ΠΑΡΑΡΤΗΜΑ Α 88

ΠΑΡΑΡΤΗΜΑ Β 98

1 Εισαγωγή

1.1 Ρομποτική αεροχημάτων στις μέρες μας

Στο χώρο της ρομποτικής τα τελευταία χρόνια έχουν γίνει μεγάλα άλματα στην ανάπτυξη συστημάτων αυτόνομων αεροπλάνων ή αλλιώς μη-επανδρωμένων αεροχημάτων όπως λέγονται (UAV, Unmanned Aerial Vehicles). Έχουν γίνει κατασκευές σε ερευνητική φάση από διάφορα πανεπιστήμια και σχολές ανά τον κόσμο καθώς και για αμυντικούς σκοπούς από τον στρατό πολλών κρατών.

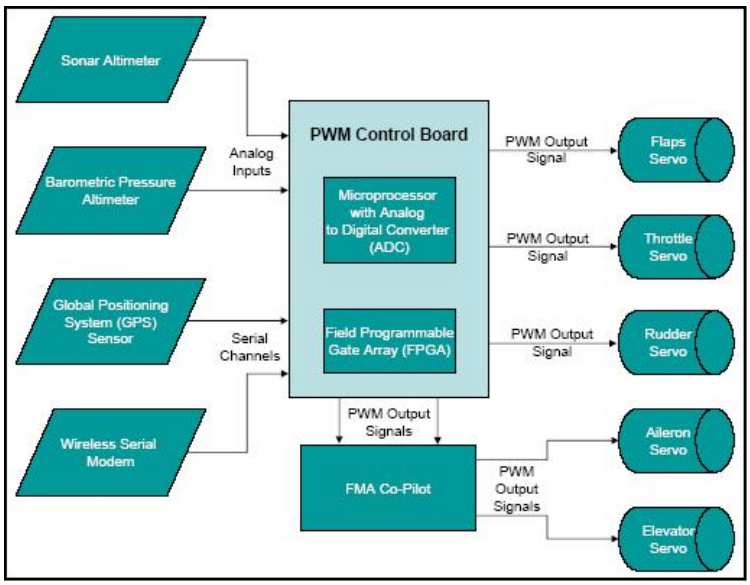
Στο πανεπιστήμιο Berkeley της California το Δεκέμβριο του 2004 μέλη από το πρόγραμμα Berkeley Aerial Robot κατάφεραν επιτυχώς μια σειρά από δοκιμαστικές πτήσεις με ελικόπτερο. Η εργασία της ομάδας για τα αυτόνομα ελικόπτερα άρχισε το 1996 με την υποστήριξη από το ερευνητικό γραφείο αμερικάνικου στρατού και το γραφείο της ναυτικής έρευνας, και συνεχίζεται μέχρι σήμερα. Το σύστημα διαθέτει ρομποτικό έλεγχο πλοήγησης αλλά και αντιδρά για την αποφυγή εμποδίων στην διαδρομή του.

Το πρόβλημα αποφυγής εμποδίων επιλύεται με το συνδυασμό της πρότυπης προβλεπτικής προσέγγισης ελέγχου με την έννοια του πιθανού τομέα. Ο αλγόριθμος είναι σε θέση να υπολογίσει σε πραγματικό χρόνο την τροχιά σε σχέση με τα εμπόδια του χώρου. Τα εμπόδια γίνονται αντιληπτά με έναν ανιχνευτή λέιζερ, με το οποίο δημιουργείτε ένας τοπικός χάρτης εμποδίων κατά μήκος της πορείας πτήσης με ικανοποιητική ακρίβεια και αξιοπιστία[1].

[1] http://robotics.eecs.berkeley.edu/bear

Ένα άλλο σύστημα αυτόνομων αεροπορικών οχημάτων έχει υλοποιηθεί από την εναέρια ομάδα ρομποτικής Waterloo (Waterloo Aerial Robotics Group, WARG) από το Πανεπιστήμιο της Arizona. για το διεθνή εναέριο ανταγωνισμό ρομποτικής (IARC)[1].

Το όχημα έχει τη συνδυασμένη ικανότητα να πετάξει αυτόνομα σε ένα κτήριο-στόχο και να προσδιορίσει τη θέση του κτηρίου και ενός ανοίγματος σε αυτό. Κατά την είσοδό του στο κτήριο, πλοηγείται προκειμένου να βρεθεί και να φωτογραφηθεί ένα αντικείμενο. Το σύστημα έχει αισθητήρα auto pilot, GPS, υψόμετρο βαρομετρικής πίεσης και υπερήχων για καλύτερη ακρίβεια, ενώ επικοινωνεί με το έδαφος με ασύρματο σειριακό modem. Η πλοήγηση έχει στηριχθεί στο λογισμικό JOLIET το οποίο για τους λόγους του διαγωνισμού μπορεί να αναλύσει την εικόνα του εδάφους με χρήση κάμερας και να εντοπίσει κτίσματα[2][3].



[1] http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html
[2] http://clubs.engr.arizona.edu/arc/index.html
[3] http://clubs.engr.arizona.edu/arc/0405site/

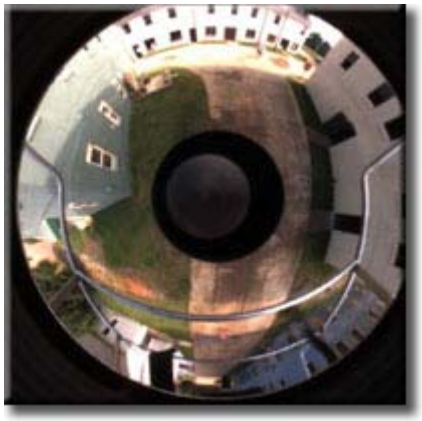
Η ομάδα USC έχει δημιουργήσει μια σειρά από projects τα οποία στηρίζονται κατά βάση σε αυτόνομα ελικόπτερα[1].

Εναέρια ρίψη αντικειμένων

Αυτή η έρευνα εξετάζει τη χρήση του αυτόνομου ελικοπτέρου για να επεκτείνει ακριβώς τα αντικείμενα στις προ-διευκρινισμένες θέσεις GPS. Είτε μια ενιαία προβλεπόμενη θέση, είτε ένα σύνολο από way-points μπορεί να διευκρινιστεί, όπου να ελευθερώνονται τα αντικείμενα. Τα πειραματικά αποτελέσματα δείχνουν ότι η ρίψη είναι σωστή και ακριβής σε περίπου 1,5 μέτρα.

Αυτόνομη προσγείωση

Αυτή η εργασία εξετάζει το σχέδιο και την εφαρμογή ενός πραγματικού χρόνου, vision-based αλγορίθμου προσγείωσης για ένα αυτόνομο ελικόπτερο. Ο αλγόριθμος προσγείωσης είναι ενσωματωμένος με τους αλγορίθμους για την οπτική απόκτηση του στόχου και τη πτήση στο στόχο αυτό, από μια αυθαίρετη αρχική θέση και έναν προσανατολισμό. Το ελικόπτερο ενημερώνεται με τις παραμέτρους του στόχου προσγείωσής του, βασισμένες σε κάμερα και χρησιμοποιεί έναν εν πλω ελεγκτή για να ακολουθήσει μια πορεία στην περιοχή. Τα αποτελέσματα από τις δοκιμές πτήσης δείχνουν ότι οι αλγόριθμοί ανίχνευσης, αναγνώρισης και ελέγχου είναι ακριβείς.



3D Navigation

Ο στόχος αυτής της έρευνας είναι να μπορεί το ελικόπτερο να ανιχνεύσει τα εμπόδια στο περιβάλλον και να τα αποφύγει. Αυτή η ικανότητα είναι μεγάλης σημασίας για να πετάξει αυτόνομα σε μέρη όπως οι αστικές περιοχές. Ο στόχος είναι να κινείται το ελικόπτερο αυτόνομα κάτω από ένα αστικό φαράγγι χωρίς να συγκρουστεί. Μια τέτοια ικανότητα θα έκανε πιθανές υλοποιήσεις όπως η αστική αναζήτηση, η διάσωση και η επιτήρηση.

[1] http://www-robotics.usc.edu/%7Eavatar/projects.htm

Για αμυντικούς σκοπούς έχουν κατασκευαστεί πολλά αυτόνομα μη-επανδρωμένα αεροχήματα.

"Eagle" EADS (European Aeronautic Defence and Space Company 2004)[1]

Μεγάλης απόστασης UAV για τα μέσα ύψη. Ο αετός υποστηρίζει ηλεκτροπτικούς και υπέρυθρους αισθητήρες, ραντάρ, elint (ηλεκτρονική νοημοσύνη), COMINT (νοημοσύνη επικοινωνίας) και η απόκτηση στόχων με χρήση λέιζερ. Ανάλογα με τον τύπο εξοπλισμού φτάνει σε ύψος 25 έως 45000 πόδια με διάρκεια πτήσης 24 έως 20 ωρών και με ταχύτητα 3300 έως 3400 km.

"EuroHawk" EADS (2004)

Μεγάλης απόστασης UAV για τα μεγάλα ύψη. Οι εικόνες είναι από μια επίδειξη με το "RQ-4A GlobalHawk" στη πειραματική ναυτική αεροπορία καταφυγίων, όπου τον Οκτώβριο του 2003 πραγματοποιήθηκαν έξι πτήσεις δοκιμής. Το EuroHawk είναι βασισμένο στο σύστημα UAV "GlobalHawk" από τον Northrop Grumman, με το οποίο η EADS αναπτύσσει το EuroHawk για τις γερμανικές ομοσπονδιακές δυνάμεις και το γαλλικό στρατό[2].

Ο συγκεκριμένος τύπος UAV που διαθέτει ο Ελληνικός Στρατός[3] έχει δυνατότητα πτήσης με μέγιστη ταχύτητα 136 μιλίων την ώρα και με πτήση μέχρι το ύψος των 11.000 ποδιών με χρήση ή χωρίς την χρήση αναμεταδότη. Τα Μη Επανδρωμένα αυτά οχήματα, θα έχουν τη δυνατότητα να μεταδίδουν εικόνα και δεδομένα σε πραγματικό χρόνο σε μία ακτίνα μεγαλύτερη των 200 km από το σημείο ελέγχου τους. Συγκεκριμένου τύπου αεροχήματα μπορεί να αναλάβουν αποστολές:

[1] http://www.eads.net/
[2] http://kai.iks-jena.de/bigb/mav.html
[3] www.army.gr/html/GR_Army/dieuthinseis/DDB/ΣΥΓΧΡΟΝΑ%20ΜΕΣΑ%20ΕΠΙΚΟΙΝΩΝΙΩΝ/UAV.htm

- Επιτήρησης – Αναγνώρισης
- Συλλογής και μετάδοσης πληροφοριών
- Αναμετάδοσης επικοινωνιών
- Υποστήριξης πυρών Πυροβολικού
- Εκτίμησης καταστροφών
- Υποστήριξης επιχειρήσεων έρευνας - διάσωσης
- Βιολογική και χημική αναγνώριση
- Ψυχολογικές επιχειρήσεις

1.2 Φαέθων

Κατά την υλοποίηση εμφανίστηκαν προβλήματα που είχαν να κάνουν με την κίνηση στο χώρο και πολλές φορές με την απρόβλεπτη συμπεριφορά ενός αεροπλάνου στον αέρα. Μερικά από αυτά ήταν, η σχετικά μεγάλη ταχύτητα που απαιτούσε γρήγορους υπολογισμούς και αποφάσεις, οι πλευρικοί άνεμοι κατά την πτήση που ανέτρεπαν την σταθερότητα σε επικίνδυνο επίπεδο και τέλος η αναξιοπιστία μερικών αισθητήριων οργάνων που απαιτούσαν συνεχή βελτίωση. Επειδή δεν ήταν γνωστό εξ αρχής το κατά πόσο επιτυχής θα ήταν ο στόχος, δόθηκε στο project το όνομα “Φαέθων” επηρεασμένοι από την ιστορία του μυθικού αυτού προσώπου.

Ο Φαέθων ήταν γιος του Ήλιου και της Κλυμένης (Ωκεανίδα). Από μικρός εκπαιδευόταν στην αρματοδρομία και έπαιζε πολύ με τον Έπαφο (γιο του Δία και της Ιούς). Κάποτε, σε μια παιδική τους διαμάχη, ο Έπαφος είπε στο Φαέθοντα, ότι δεν ήταν πραγματικό παιδί του Ήλιου. Ο Φαέθων αγανακτισμένος, πήγε στον πατέρα του ο οποίος τον διαβεβαίωσε, ότι ήταν πράγματι γιος του. Ο Φαέθων όμως έμεινε με αμφιβολίες. Έβαλε τον πατέρα του να ορκιστεί στα νερά της Στύγας, προς απόδειξη ότι θα του έδινε οτιδήποτε του ζητούσε. Ο Ήλιος ορκίστηκε και ο νεαρός Φαέθων του ζήτησε να γίνει ηνίοχος έστω και για μια μέρα μόνο της αμαξάς του. Ο Ήλιος κατάλαβε τον κίνδυνο, αλλά δεν μπορούσε να παραβεί τον όρκο του. Έδωσε τις κατάλληλες οδηγίες στο γιο του, του έδωσε και τα χαλινάρια του μεγάλου αστραφτερού άρματος και του ευχήθηκε καλή τύχη. Ο Φαέθων πήρε στα χέρια του την άμαξα και ξεχύθηκε στους αιθέρες. Μόλις απομακρύνθηκε όμως από τη γη και αντίκρισε μπροστά του το απέραντο χάος, ζαλίστηκε και άφησε χαλαρά τα ηνία των δυνατών αλόγων. Αυτά αφήνιασαν και άρχισαν να τρέχουν πότε ψηλά στο απέραντο του ουρανού και πότε χαμηλά, πολύ κοντά στη Γη. Ο δίσκος του Ήλιου κινδύνευε να κατακάψει στο πέρασμα του τη Γη. Ο Δίας, λίγο πριν την μεγάλη καταστροφή που θα προκαλούσε ο αδέξιος ηνίοχος, του έριξε έναν κεραυνό και σταμάτησε την τρελή πορεία του. Το σώμα του Φαέθοντα έπεσε φλεγόμενο σαν κομήτης (επειδή καιγόταν η κόμη του) στον Ηριδανό ποταμό της Ιταλίας (σημερινός Πάδος) ή στον Ηριδανό της Αττικής (παραπόταμος του Ιλισού), Οι αδελφές του, οι Ηλιάδες Μερόπη, Ηλία, Αίγλη, Φοίβη, Λαμπετία, Αιθέριο και Διοξίππη λυπήθηκαν πολύ για το χαμό του Φαέθοντα. Η παράδοση λέει, ότι αναζήτησαν το σώμα του για να το θάψουν και όταν τελικά το βρήκαν στον ποταμό, τον έκλαψαν επί 4 μήνες.

2 Αερομοντελισμός

2.1 Γενικά

Το ερέθισμα της πτήσης, η αεροπορική ιδέα - το αεροπορικό πνεύμα, εμφυτεύεται στον άνθρωπο μόλις συνειδητοποιήσει ότι τα πουλιά πετάνε. Δεν μένει παρά μόνο η ευκαιρία για να τα μιμηθεί ακολουθώντας κάποιο αεράθλημα, ή ασκώντας το επάγγελμα του αεροπόρου. Χωρίς υπερβολή, τις περισσότερες ευκαιρίες τις δίνει ο αερομοντελισμός, από την νεανική ακόμη ηλικία, με το πλέον προσιτό πεδίο, την μεγαλύτερη ποικιλία σε σχέδια και τις αναρίθμητες δραστηριότητες και συγκινήσεις.

2.1.1 Τι ονομάζουμε Αερομοντελισμό;

Αερομοντελισμός είναι η ενασχόληση με την κατασκευή και την πτήση μικρών αεροπλάνων. Τα μικρά αυτά αεροπλάνα ονομάζονται αερομοντέλα ή απλώς μοντέλα. Με άλλα λόγια, αερομοντέλο είναι ένα μικρό αεροπλάνο που μπορεί μεν να πετάξει, αλλά λόγω του μικρού του μεγέθους δεν μπορεί να μεταφέρει άνθρωπο.

Ο αερομοντελισμός είναι αναμφισβήτητα ένα χόμπι για μικρούς και μεγάλους το οποίο δίνει μεγάλη χαρά από τη συναρμολόγηση του μοντέλου του αεροπλάνου της αρεσκείας μας, μέχρι την μεγαλύτερη ικανοποίηση της πτήσης. Πέρα από αυτό όμως ο αερομοντελισμός έχει καθιερωθεί και σαν άθλημα.

Πρέπει όμως να είμαστε προσεκτικοί γιατί παρόλο που ο αερομοντελισμός είναι χόμπι δεν είναι παιχνίδι. Είναι εμφανής η διαφορά μεταξύ των δημιουργημάτων των αερομοντελιστών και των έτοιμων παιχνιδιών με σχήμα αεροπλάνου που μπορεί να αγοράσει ο καθένας. Το παιχνίδι αγοράζεται έτοιμο και μπορεί να χρησιμοποιηθεί μόλις ξετυλιχθεί το περιτύλιγμα. Οι αερομοντελιστές αφιερώνουν πολλές ώρες, ημέρες, ή ακόμα και χρόνια, για να φτιάξουν το αερομοντέλο τους με στόχο την καλύτερη εκμετάλλευση των προσφερόμενων στοιχείων, και πάντα υπό το πρίσμα της συνεχούς βελτίωσης.

2.1.2 Τηλεκατευθυνόμενα

Τα τηλεκατευθυνόμενα (radio control, R/C), ελέγχονται με ηλεκτρομαγνητικά κύματα και εκτελούν τις εντολές του αερομοντελιστή/πιλότου με πολύ μεγάλη ακρίβεια. Ο χειριστής ενός τηλεκατευθυνόμενου μοντέλου κρατάει στα χέρια του τον πομπό του συστήματος τηλεκατεύθυνσης και κινώντας τους κατάλληλους μοχλούς στέλνει σήμα στον δέκτη που είναι στο μοντέλο. Ο δέκτης στέλνει σήμα να κινηθούν οι κεφαλές των σερβομηχανισμών (γνωστότερα σαν “σέρβο”) που είναι μέσα στο μοντέλο. Τα σέρβο με την σειρά τους κινούν τα πηδάλια, αυξομειώνουν τις στροφές του κινητήρα (στα μηχανοκίνητα), ανεβοκατεβάζουν το σύστημα προσγείωσης και ό,τι άλλο χρειάζεται για μία ρεαλιστική πτήση. Με τον τρόπο αυτό ο χειριστής απογειώνει το μοντέλο του, το οδηγεί σε απλούς ή ακροβατικούς ελιγμούς και τέλος το προσγειώνει.

Οι τύποι των δημοφιλέστερων τηλεκατευθυνόμενων μοντέλων είναι τα μηχανοκίνητα, τα ανεμόπτερα, και τα ελικόπτερα. Τα ανεμόπτερα ανεβαίνουν ψηλά με πολλούς τρόπους (ρυμούλκηση με αεροπλάνο, εκτόξευση με καταπέλτη, ή με βοηθητικό κινητήρα). Αν βρίσκονται στην πλαγιά ενός λόφου αρκεί ένα απλό σπρώξιμο. Μετά οδηγούνται στα σημεία που υπάρχουν ανοδικά ρεύματα του αέρα για να παρατείνουν την διάρκεια της πτήσης τους. Αν βρούνε κατάλληλες συνθήκες μπορούν να μείνουν στον αέρα ακόμα και ολόκληρη την ημέρα ή να ταξιδέψουν μεγάλες αποστάσεις. Όταν εξαντληθούν τα περιθώρια ύψους, οδηγούνται στο σημείο προσγείωσης με θαυμαστή ακρίβεια.

Στις κατηγορίες των μηχανοκίνητων διακρίνουμε τα ακροβατικά (aerobatics) και τα ταχύτητας σε κλειστή διαδρομή (pylon racers). Τέλος, τα τηλεκατευθυνόμενα ελικόπτερα έγιναν προσιτά στο ευρύ κοινό την δεκαετία του '80. Είναι αριστουργήματα της τεχνολογίας, με δυνατότητα να κάνουν και αυτά ασκήσεις ακριβείας και ακροβατικούς ελιγμούς, ακόμα και να προσγειωθούν με σβηστό κινητήρα.

2.1.3 Πόσο ψηλά πετάνε, πόσο ζυγίζουν, για πόσο πετάνε κτλ.

Ένα τηλεκατευθυνόμενο μοντέλο δεν έχει κάποιο περιορισμό σε σχέση με το ύψος ή την απόσταση που μπορεί να πετάξει από τον χειριστή του. Τα όρια δεν τίθενται ούτε από τα συστήματα τηλεκατεύθυνσης, η εμβέλεια τους φτάνει υπό φυσιολογικές συνθήκες περισσότερο από περίπου 1 Km. Το μόνο όριο είναι η ορατότητα του χειριστή. Συνήθως αυτή είναι πολύ μικρότερη ακόμα και από τη μισή εμβέλεια της τηλεκατεύθυνσης.

Το βάρος των μοντέλων ποικίλει, μία φυσιολογική τιμή είναι από 1 ως 5+ kgr αν και υπάρχουν μοντέλα πολύ βαρύτερα (βλέπε 15 kgr!). Ο μέγιστος χρόνος πτήσης εξαρτάται από την ποσότητα καυσίμου που φέρει το μοντέλο αν και γενικά βρίσκεται περίπου στα 20 λεπτά. Τα ηλεκτρικά αεροπλάνα κατά κανόνα πετούν για λιγότερο χρόνο.

Παρόλο που μία εντελώς επίπεδη επιφάνεια είναι ιδανική για αποπροσγειώσεις, τα περισσότερα μοντέλα δεν αντιμετωπίζουν πρόβλημα σε διάφορες επιφάνειες: από άσφαλτο ή τσιμέντο ως χωράφια με σχετικά σκληρό χώμα ή χόρτα.

2.1.4 Πώς φτιάχνεται ένα αερομοντέλο;

Ο τρόπος κατασκευής ενός αερομοντέλου είναι ο κλασικός που παραμένει μέχρι σήμερα ο ίδιος. Ακολουθώντας ένα κατασκευαστικό σχέδιο, κόβει και κολλάει κομμάτια από ελαφρύ ξύλο ή κόντρα πλακέ και επικαλύπτει τον ανοιχτό σκελετό με ειδικό χαρτί ή φύλλο πλαστικού. Φυσικά χρησιμοποιούνται και τα μοντέρνα υλικά όπως η διογκωμένη πολυστερίνη και νήματα από γυαλί ή κάρβουνο εμποτισμένα σε εποξικές ρητίνες. Στον τελειωμένο σκελετό προσθέτει, όπου προβλέπεται, τα υπόλοιπα μέρη όπως:

Μηχανή, ηλεκτρική ή θερμική, που θα μπορεί να ανταπεξέλθει στις απαιτήσεις του μοντέλου.

Σύστημα τηλεκατεύθυνσης, βασικό χαρακτηριστικό των συστημάτων αυτών είναι τα κανάλια/channels που υποστηρίζουν. Το πόσα κανάλια χρειάζονται εξαρτάται από το μοντέλο. Υπάρχουν αεροπλάνα που πετούν με 3ch αν και καλύτερα είναι τα 4ch ή περισσότερα. Τα ελικόπτερα χρειάζονται τουλάχιστον 5ch.

2.1.5 Άλλες μορφές αερομοντελισμού

Μία ακόμα μεγάλη πτυχή του αερομοντελισμού είναι ο πυραυλομοντελισμός που ασχολείται με τα μοντέλα των ρουκετών (model rockets), που όμως δεν θα μας απασχολήσουν εδώ. Τέλος πολλοί νέοι γνώρισαν τον αερομοντελισμό συναρμολογώντας στατικά μοντέλα από πολυστερίνη, ομοιώματα κι αυτά γνωστών αεροπλάνων (πλαστικομοντελισμός). Όπως συνάγεται από την ονομασία τους, δεν έχουν πτητικές ικανότητες.

2.2 Θεωρία πτήσης

2.2.1 Βασικά

Τα φτερά των αεροπλάνων είναι έτσι φτιαγμένα ώστε ο αέρας που περνά από την πάνω επιφάνεια τους να έχει μεγαλύτερη ταχύτητα από τον αέρα που περνά από την κάτω επιφάνειά τους. Ο αέρας που κινείται γρηγορότερα ασκεί λιγότερη πίεση στο φτερό, ενώ ο αέρας που κινείται πιο αργά ασκεί μεγαλύτερη πίεση. Έτσι τα φτερά δέχονται συνολική δύναμη προς τα πάνω, η οποία και κρατά το αεροπλάνο στον αέρα. Ο ρόλος της μηχανής είναι να εξασφαλίζει ότι υπάρχει πάντα ροή αέρα στα φτερά (με το να δίνει ταχύτητα στο αεροπλάνο) ώστε να υπάρχει και η παραπάνω ανυψωτική δύναμη.

Ένας άλλος δρόμος επεξήγησης της πτήσης, που εισάγει κάποιες πιο χρήσιμες έννοιες, αναλύει το φαινόμενο χωρίς την χρήση της πίεσης αλλά μόνο των δυνάμεων που εμπλέκονται.

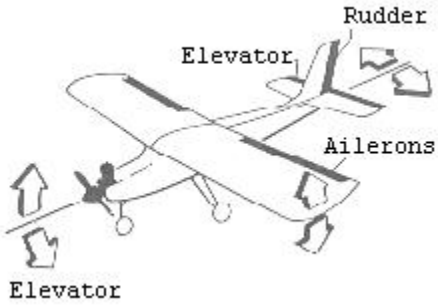
Κατά την κίνηση των αεροπλάνων στο χώρο τα φτερά τους συνήθως συναντούν τον αέρα υπό μία μικρή γωνία που λέγεται γωνία προσβολής (angle of attack) οπότε και πάνω τους ασκείται μία δύναμη πού αναλύεται σε μία κάθετη προς τα πάνω συνιστώσα (αντίθετη του βάρους του αεροπλάνου-lift) και μία οριζόντια που έχει φορά αντίθετη της κίνησης/ταχύτητας του αεροπλάνου (οπισθέλκουσα/drag).

Είναι γεγονός πως τα μοντέλα αεροπλάνων πετούν ακριβώς με τον ίδιο τρόπο όπως και τα πραγματικά αφού ουσιαστικά είναι πραγματικά αεροσκάφη υπό κλίμακα. Εδώ όμως τελειώνουν και οι ομοιότητες...

2.2.2 Πιο ειδικά

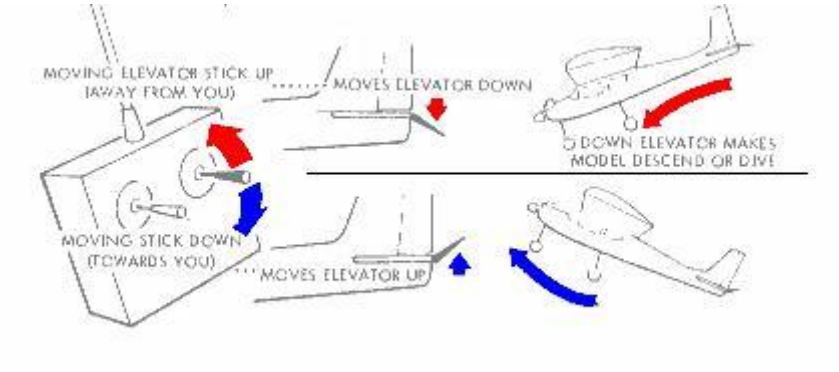
Υπάρχουν πολλά πράγματα που είναι περισσότερο σημαντικά... είναι ενδεικτικό πως συνήθως οι πιλότοι πραγματικών αεροπλάνων κάνουν το λάθος να πιστέψουν πως θα είναι εύκολο γι' αυτούς να πετάξουν ένα μοντέλο. Τα πράγματα είναι εντελώς διαφορετικά όταν πετά κανείς αεροπλάνο έξω από αυτό, ειδικά όταν αυτό έρχεται ίσια καταπάνω στον χειριστή του! (όλοι οι χειρισμοί γίνονται ανάποδα).

Τα φτερά του αεροπλάνου διαθέτουν κινητές επιφάνειες με τις οποίες και αλλάζουν/επηρεάζουν τον τρόπο με τον οποίο κινούνται μέσα στον αέρα. Στο παρακάτω σχήμα διακρίνονται οι επιφάνειες αυτές σε ένα κλασικής σχεδίασης αεροπλάνο.

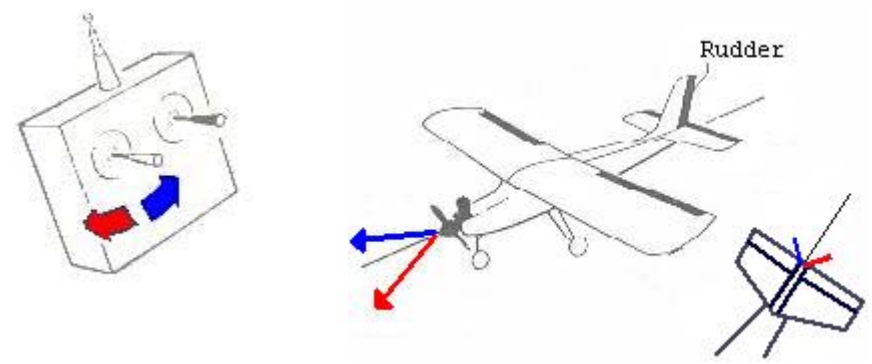


Πάνω στο κεντρικό φτερό βρίσκονται δύο τέτοιες επιφάνειες που κινούνται πάντα αντίθετα η μία σε σχέση με την άλλη και ονομάζονται aileron. Είναι δε υπεύθυνες για την κλίση του αεροπλάνου γύρω από τον οριζόντιο άξονα που διαπερνά την άτρακτο. Στα σχήματα φαίνεται η κίνηση του αεροπλάνου ανάλογα με την κίνηση των aileron και την ανάλογη κίνηση που κάνει ο χειριστής στην τηλεκατεύθυνση.

Η ουρά του αεροπλάνου αποτελείται από δύο μέρη, το οριζόντιο και κάθετο σταθερό. Κάθε μία από αυτές τις επιφάνειες διαθέτει κινητά μέρη που λέγονται αντίστοιχα elevator και rudder. Η κίνηση του elevator (όπως άλλωστε δηλώνει και το όνομα του) καθορίζει την ανύψωση ή την πτώση της μύτης του αεροπλάνου σε σχέση με τον ορίζοντα (αλλάζει δηλ. η γωνία προσβολής). Έτσι δεν μεταβάλλεται αναγκαστικά το ύψος της πτήσης γιατί αυτό εξαρτάται και από την ώθηση του κινητήρα.



Το rudder μεταβάλλει την διεύθυνση κατά την οποία πετά το αεροπλάνο. Τα παρακάτω σχήματα δείχνουν ακριβώς την σχέση rudder-στροφής καθώς και την κίνηση στην τηλεκατεύθυνση.



Ο μόνος χειρισμός που απομένει είναι η κίνηση του αριστερού μοχλού πάνω-κάτω η οποία ελέγχει τις στροφές της μηχανής.

Είναι προφανές πως οποιαδήποτε κίνηση ελέγχου μπορεί να ρυθμιστεί ώστε να "υπακούει" σε οποιοδήποτε από τα joysticks της τηλεκατεύθυνσης. Το παραπάνω setup είναι μία από τις επιλογές που υπάρχουν (γνωστές σαν Mode1, Mode2 κτλ.) και συνηθίζεται στην Ευρώπη και σε άλλα μέρη του κόσμου. Έτσι γίνεται δυνατή η συνεννόηση μεταξύ των μοντελιστών.

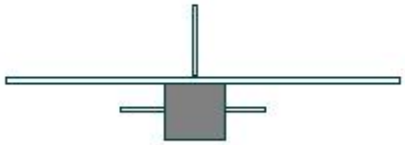
2.2.3 Τι είναι το stall

Αν χρησιμοποιηθεί μόνο το elevator για την κίνηση του αεροπλάνου προς τα πάνω, τότε κάνει την εμφάνιση του το δυσάρεστο φαινόμενο της απώλειας στήριξης (stall). Αρχικά το αεροπλάνο παίρνει ύψος δίνοντας την κινητική του ενέργεια για χάρη της δυναμικής ενέργειας που αυξάνει, όμως αφού η κινητική ενέργεια δεν διατηρείται με την βοήθεια του κινητήρα, κάποια στιγμή φτάνει υπερβολικά κοντά στο μηδέν. Αυτό σημαίνει ότι η ροή του αέρα στα φτερά έχει μειωθεί τόσο πολύ ώστε να μην μπορεί να παραμείνει το αεροσκάφος στον αέρα. Το αεροπλάνο τότε αρχίζει να χάνει ραγδαία ύψος μέχρι είτε να ξαναποκτήσει την απαραίτητη ταχύτητα, είτε να συναντήσει... το έδαφος. Πολλές φορές η απώλεια στήριξης συμβαίνει χωρίς απαραίτητα να έχει γίνει κάποιος λάθος χειρισμός, π.χ. όταν ο κινητήρας δεν έχει άλλα περιθώρια ισχύος.

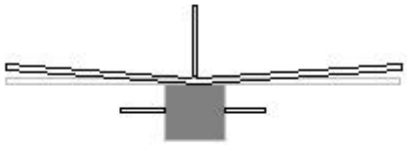
2.3 Σκάφος - Κατασκευή

2.3.1 Βασικά

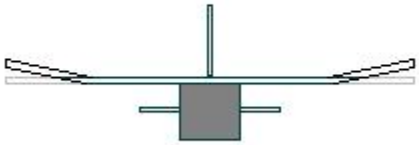
Είναι γεγονός πως όλα τα αεροπλάνα δεν είναι ίδια. Η σχεδίαση κάθε αεροπλάνου είναι τέτοια ώστε να ευνοεί κάποιο/α συγκεκριμένα χαρακτηριστικά πτήσης (π.χ. ακροβατικά, σταθερότητα κτλ.) Τα βασικότερα χαρακτηριστικά στη πτήση ενός αεροπλάνου τα καθορίζει η σχεδίαση των φτερών του. Το φτερό ενός μοντέλου αν παρατηρηθεί προσεκτικά θα διαπιστωθεί καταρχάς ότι συνήθως δεν αποτελείται από ένα και μόνο επίπεδο κομμάτι/επιφάνεια όπως αυτό του αεροπλάνου του παρακάτω σχήματος.



Το αεροπλάνο του παραπάνω σχήματος θα ήταν πολύ ασταθές πράγμα που θα του έδινε πλεονέκτημα στην εκτέλεση κάποιον ακροβατικών στα χέρια ενός έμπειρου χειριστή. Συνήθως, τα εκπαιδευτικά αεροπλάνα διαθέτουν φτερά που αποτελούνται από δύο επιφάνειες (έδρες) που σχηματίζουν μεταξύ τους μία γωνία (δίεδρος γωνία - dihedral), ακριβώς όπως το αεροπλάνο του επόμενου σχήματος:



Το παραπάνω αεροπλάνο θα ήταν πολύ πιο σταθερό από το πρώτο. Στην προσπάθεια για μεγαλύτερη σταθερότητα υπάρχουν επίσης μοντέλα με περισσότερες έδρες στα φτερά τους, τρεις ή τέσσερις. Στην πρώτη περίπτωση δύο έδρες σχηματίζουν γωνία με μία τρίτη στο κέντρο



ενώ στην δεύτερη περίπτωση και η μεσαία έδρα χωρίζεται σε δύο άλλες υπό κάποια γωνία.

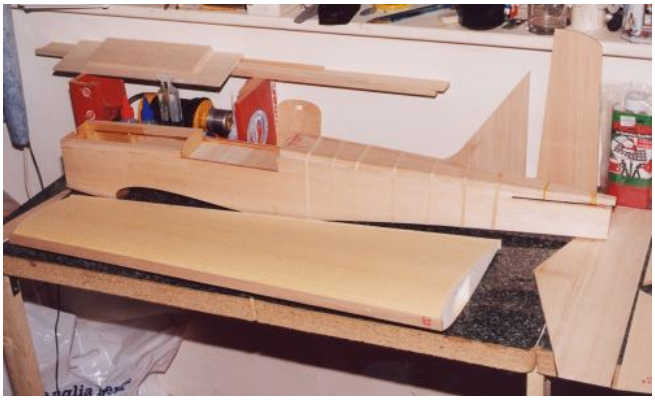
Η κατανόηση της συμπεριφοράς κάθε τύπου φτερού στην πτήση επιτρέπει να μελετηθεί και η στρέψη του αεροσκάφους μόνο με τη χρήση του rudder. Αν λοιπόν χρησιμοποιηθεί μόνο το rudder για να στραφεί το αεροπλάνο, αυτό που θα συμβεί είναι να στραφεί μεν η άτρακτος προς την επιθυμητή διεύθυνση αλλά το διάνυσμα της ταχύτητας του αεροπλάνου δεν θα αλλάξει και αυτό διεύθυνση (αφού δεν θα έχει ασκηθεί καμία δύναμη για το σκοπό αυτό). Το αποτέλεσμα θα είναι το σκάφος να "γλιστρήσει" κατά την φορά της ταχύτητας του και όχι να στρίψει. Όλα αυτά βέβαια αν το φτερό του αεροπλάνου δεν είναι δίεδρο. Γιατί στην περίπτωση της δίεδρης κατασκευής η στροφή της ατράκτου π.χ. προς τα δεξιά του αεροπλάνου σημαίνει ότι η αριστερή έδρα του φτερού βρίσκεται σε πλεονεκτικότερη της δεξιάς θέση σε σχέση με την ταχύτητα του σκάφους και άρα παράγει μεγαλύτερη άνωση. Η διαφορά άνωσης μεταξύ των δύο επιφανειών του φτερού κάνει το αεροσκάφος να πάρει κλίση όμοια με την κλίση που θα του έδιναν τα aileron οπότε στη συνέχεια έχουμε στροφή. Το ίδιο συμβαίνει και στις περιπτώσεις με περισσότερες έδρες. Έτσι καταφέρνουν να πετούν αποτελεσματικά μοντέλα χωρίς aileron.

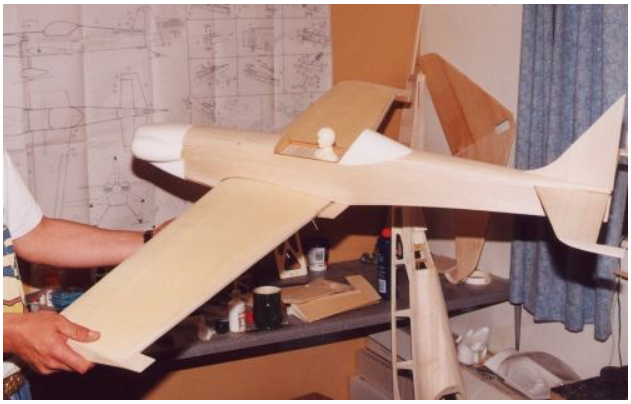
Τα μοντέλα ανάλογα με τη σχεδίαση του σκάφους διακρίνονται επίσης και σε υψηλοπτέρυγα, μεσοπτέρυγα και χαμηλοπτέρυγα. Τα πρώτα είναι πιο σταθερά και ενδείκνυνται για εκπαίδευση νέων χειριστών ενώ τα τελευταία είναι μάλλον ασταθή και χρειάζονται πιο έμπειρους χειριστές.

2.3.2 Συναρμολόγηση και Ζύγισμα

Η κατασκευή κάθε μοντέλου είναι μία πολύ σημαντική διαδικασία. Μια λάθος κατασκευή (ακόμα και σε μία φαινομενικά μικρή λεπτομέρεια) μπορεί να έχει σαν αποτέλεσμα προβληματική πτητική συμπεριφορά άρα προβλήματα στην πτήση, την εκμάθηση και τέλος ίσως και να προξενήσει πτώση και καταστροφή του μοντέλου.

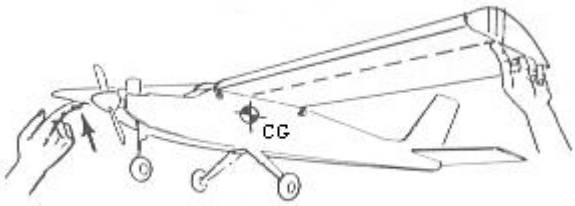
Τα περισσότερα μοντέλα αποτελούνται από ξύλινα κομμάτια που στερεώνονται με κόλλα, συνήθως εποξική ή κυανοακρυλική.





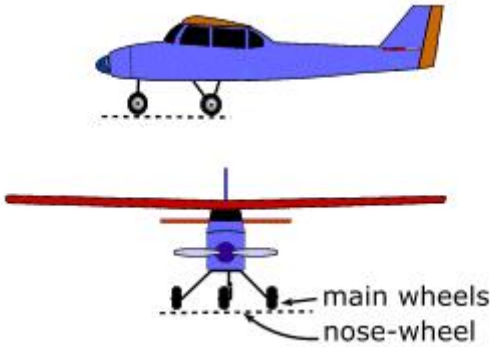
Οι κόλλες αυτές είναι κατάλληλες για μοντέλα αφού κρατούν τα ξύλινα μέρη αρκετά γερά κολλημένα ώστε να αντέξει το μοντέλο στην καταπόνηση.

Μία επιπλέον σημαντική διαδικασία είναι ο τελικός έλεγχος και το ζύγισμα του μοντέλου ώστε να διαπιστωθεί αν υπάρχει η σωστή ισορροπία της κατασκευής και αν το κέντρο βάρους βρίσκεται στη σωστή περιοχή του σκάφους. Συνήθως το σωστό setup έχει σαν αποτέλεσμα όταν το μοντέλο συγκρατείται από τα άκρα των φτερών να παρουσιάζει ελαφριά κλίση προς τα μπροστά.

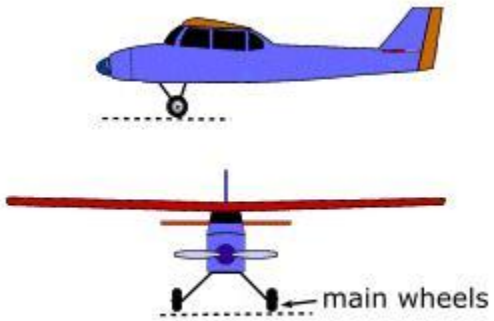


2.3.3 Σύστημα προσγείωσης

Υπάρχουν αερομοντέλα με σύστημα προσγείωσης που αποτελείται από 2 ή από 3 ρόδες. Η διαρρύθμιση που έχουν τα σκέλη σε κάθε μοντέλο παίζει σημαντικό ρόλο στις αποπροσγειώσεις. "Συνήθως" για εκπαιδευτικές πτήσεις πιο βολικό είναι το σύστημα με 3 τροχούς που φαίνεται παρακάτω:



Βέβαια αυτό είναι εντελώς υποκειμενικό. Η άλλη πιο συνηθισμένη διαμόρφωση είναι η παρακάτω:



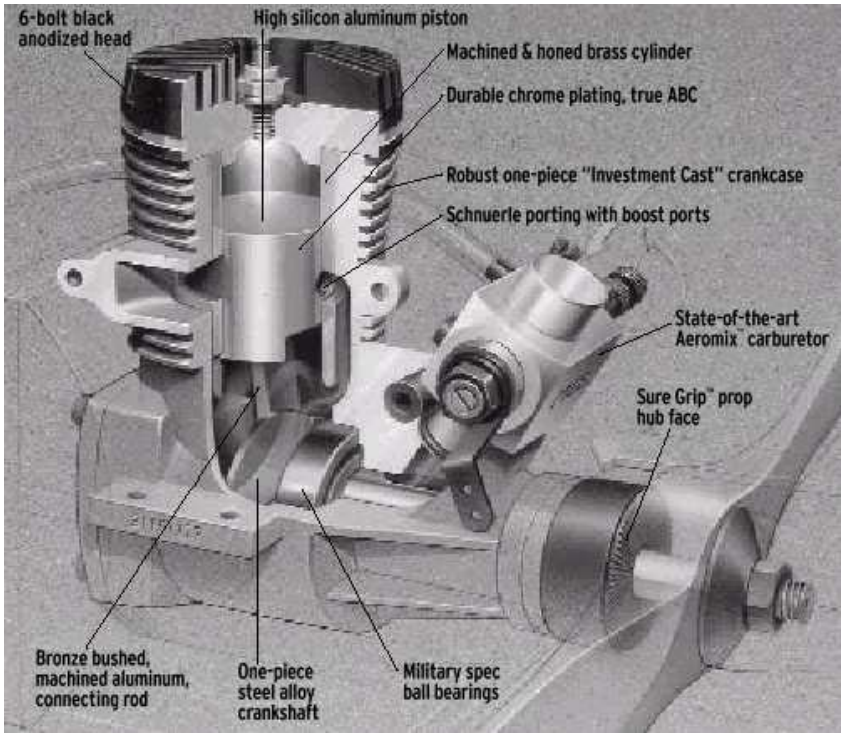
2.4 Μηχανή (αεροπλάνα)

2.4.1 Γενικά

Τα αεροπλάνα διακρίνονται σε ηλεκτροκίνητα και θερμικά. Και τα δύο έχουν πλεονεκτήματα και μειονεκτήματα. Τα ηλεκτροκίνητα είναι σχεδόν αθόρυβα, καθαρότερα και πιο οικολογικά ενώ και δεν χρειάζονται ακριβά καύσιμα. Η πτήση όμως είναι κάπως υποτονική και κακές καιρικές συνθήκες(δυνατός αέρας) ή δύσκολοι ελιγμοί μπορεί να το δυσκολέψουν να ανταπεξέλθει και ίσως κρατήσουν το μοντέλο στο έδαφος. Επίσης χρειάζονται και επαναφορτιζόμενες μπαταρίες που είναι σχετικά ακριβές. Από την άλλη τα θερμικά μοντέλα έχουν περισσότερη δύναμη, αυτονομία και δεν χρειάζονται φόρτιση μπαταριών αλλά κάνουν θόρυβο, είναι πιο βρώμικα (καύσιμα, εξάτμιση κτλ) και χρειάζονται ακριβά καύσιμα όπως και εξοπλισμό για την εκκίνηση τους που είναι πιο δύσκολη.

2.4.2 Θερμικοί κινητήρες

Ένα μοντέλο έχει υπερβολικά μεγάλες απαιτήσεις από μια μηχανή, μερικές όχι τόσο εμφανείς όπως π.χ. την ανάγκη συντήρησης που πρέπει να είναι ελάχιστη(ως ανύπαρκτη) αφού δεν μπορεί να απαιτείται από τον μοντελιστή να διαθέτει ούτε τις γνώσεις ούτε τα υλικά για δύσκολες διαδικασίες συντήρησης. Ο τύπος μηχανής που ικανοποιεί κατά το δυνατό όλες τις απαιτήσεις έχει παγιωθεί στο χώρο του μοντελισμού και πλέον ο χαρακτηρισμός model engine αναφέρεται σχεδόν αποκλειστικά στις μοντελιστικές glow μηχανές. Υπάρχουν εταιρίες που κατασκευάζουν ακριβώς τέτοιες μηχανές για μοντελιστική και μόνο χρήση.



Οι κινητήρες αεροπλάνων που μας ενδιαφέρουν χαρακτηρίζονται από τον κυβισμό τους που αρχίζει από τις 0.010 κυβικές ίντσες (0.16cc). Συνήθως αναφερόμαστε σε μια μηχανή μοντέλου με βάση τις κυβικές ίντσες της. Κλασικά μεγέθη είναι οι 0.10-0.15ci (~2cc) ή κοινώς 15άρα, 0.25ci (~4.0cc) ή 25αρα, 40άρα, 60άρα(~10cc) κτλ. Τα μεγέθη αυτά καθορίζουν και τις κατηγορίες μεγέθους των αεροπλάνων αφού ένα συγκεκριμένου μεγέθους μοντέλο αεροπλάνο μπορεί να "δεχθεί" μόνο αντίστοιχου μεγέθους μηχανή.

Το μοντέλο μπορεί να αποτελεί ουσιαστικά την μικρογραφία ενός πραγματικού αεροπλάνου όμως ο αέρας μέσα στον οποίο πετά δεν είναι δυστυχώς και αυτός υπό κλίμακα. Η άπλετη ισχύς είναι αναγκαία για να πετάει ένα μοντέλο ευκολότερα. Γενικά σαν όριο τίθεται παγκοσμίως από την πλειοψηφία των έμπειρων μοντελιστών στις 0.40ci το ελάχιστο για να μπορεί να πετά ένα μοντέλο ικανοποιητικά και να είναι αρκετά σταθερό.

Οι παραπάνω κινητήρες χρησιμοποιούν σαν μπουζί μια ηλεκτρική πηγή που διοχετεύει ρεύμα κατά την εκκίνηση του κινητήρα, μετά δεν χρειάζεται πλέον, αποδεσμεύεται από την μηχανή και παραμένει στο έδαφος κατά τη διάρκεια της πτήσης. Για την εκκίνηση χρειάζονται το παραπάνω "μπουζί" καθώς και ένας ηλεκτροκινητήρας που δίνει τις αρχικές στροφές στον κινητήρα (κάτι που όταν γίνεται με το χέρι είναι πάρα πολύ επικίνδυνο, έχοντας υπόψη ότι μια έλικα περιστρέφεται χωρίς κόπο με 10000-15000στροφές/λεπτό).

Οι κινητήρες του παραπάνω τύπου διακρίνονται σε δίχρονους και τετράχρονους. Οι δίχρονοι είναι ελαφρότεροι, φθηνότεροι και ανθεκτικότεροι στις πτώσεις, παράγουν όμως περισσότερους κραδασμούς. Οι τετράχρονοι κινητήρες είναι ακριβότεροι, βαρύτεροι και πολυπλοκότεροι αλλά με πιο ρεαλιστικό θόρυβο και λιγότερους κραδασμούς.

Το καύσιμο που χρησιμοποιείται είναι ένα μίγμα βασισμένο στη μεθανόλη. Περιέχει δε ανάμεσα στα αλλά συστατικά λιπαντικά (για τη λίπανση του κινητήρα) και σε ένα ποσοστό 0-30% νιτρομεθάνιο. Τα καύσιμα με μεγαλύτερο ποσοστό νιτρομεθανίου έχουν μεγαλύτερη απόδοση και διατίθενται σε διάφορες διαβαθμίσεις "νίτρου" με πιο συνηθισμένες το 5%, 10%, 15%-25%.

Μερικά είδη κινητήρων φαίνονται παρακάτω:

Κλασική αεροπορική

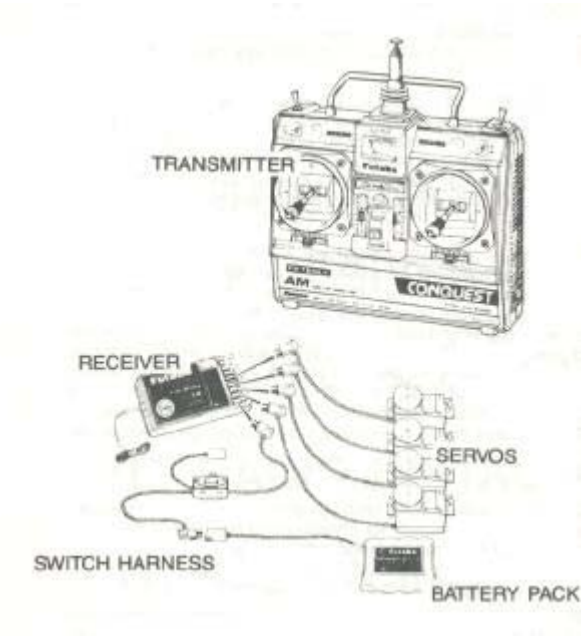
Κινητήρας τύπου Wankel για μοντέλο!...

Κινητήρας με δύο έμβολα

2.5 Συστήματα Τηλεκατεύθυνσης

Ένα σύστημα τηλεκατεύθυνσης για οποιοδήποτε αερομοντέλο αποτελείται από τα εξής μέρη:

- την τηλεκατεύθυνση (έτσι αποκαλείται το "χειριστήριο" που κρατά ο μοντελιστής στα χέρια του) ή αλλιώς τον πομπό
- τον δέκτη (που τοποθετείται πάνω στο μοντέλο)
- τους σερβομηχανισμούς ή απλούστερα τα servo που είναι οι ηλεκτρομηχανικές εκείνες διατάξεις που κινούν τις επιφάνειες ελέγχου.
- τις μπαταρίες για τα παραπάνω και τον φορτιστή



Το βασικότερο χαρακτηριστικό των τηλεκατευθύνσεων είναι ο αριθμός των καναλιών που εκπέμπουν. Αυτό γιατί όσο περισσότερα είναι αυτά τόσες περισσότερες πληροφορίες μπορούν να μεταδοθούν και τόσα servo να ελεγχθούν από μακριά.

Ο όρος κανάλι χρησιμοποιείται γενικά στο μοντελισμό για να εκφράσει την πληροφορία που δέχεται ένα servo ή μία επιφάνεια ελέγχου. Έτσι για να πετάξει ένα αεροσκάφος χρειάζεται «θεωρητικά» 3 κανάλια για τα: elevator, rudder ή aileron(ένα από τα δύο) και το γκάζι.

Τα πρώτα συστήματα τηλεκατεύθυνσης ήταν AM (amptitude modulation). Τα συστήματα αυτά είναι πλέον ξεπερασμένα αφού τώρα χρησιμοποιούνται συστήματα FM (frequency modulation) που έχουν καλύτερη αντοχή στις παρεμβολές. Τα FM συστήματα διακρίνονται στα PPM (pulse position modulation) και PCM (pulse code modulation).

Η εμβέλεια των τηλεκατευθύνσεων που κυκλοφορούν σήμερα είναι πραγματικά εντυπωσιακή. Οι κατασκευαστές δίνουν συνήθως σαν τυπικές τιμές σε ανοικτούς

χώρους αποστάσεις περίπου 1 Km αν και με πλήρη φόρτιση των μπαταριών η εμβέλεια είναι στην πραγματικότητα λίγο μεγαλύτερη.

Οι συχνότητες που χρησιμοποιούνται καθορίζονται από την νομοθεσία της κάθε χώρας. Παγκοσμίως οι συχνότητες που χρησιμοποιούνται για αερομοντέλα κυμαίνονται στην περιοχή 35~72 Mhz. Πάντως ένα σύστημα τηλεκατεύθυνσης μπορεί να λειτουργήσει σε ένα εύρος διαφορετικών συχνοτήτων αν το επιθυμεί ο κάτοχος του. Η συχνότητα στην οποία λειτουργούν ο πομπός και ο δέκτης καθορίζεται από ένα ζευγάρι κρυστάλλων που τοποθετούνται σε κατάλληλες υποδοχές των παραπάνω συσκευών. Δύο ή περισσότερες τηλεκατευθύνσεις δεν μπορεί να λειτουργούν στον ίδιο χώρο και στην ίδια συχνότητα γιατί η μία θα παρεμβάλλεται στην άλλη με ολέθρια αποτελέσματα.

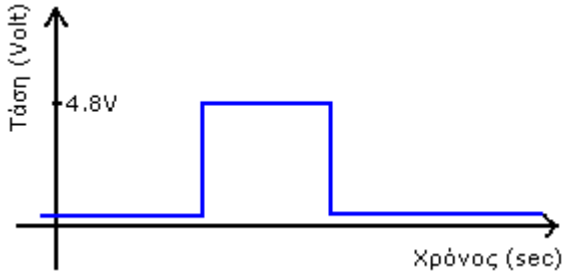
2.6 Σέρβο (servo)

Τα σέρβο είναι μικροί ηλεκτροκινητήρες που κινούνται με ακρίβεια και παρέχουν την μηχανική κίνηση για τον έλεγχο του μοντέλου. Συνδέονται με καλώδιο με τον δέκτη της τηλεκατεύθυνσης που τους παρέχει εκτός από ρεύμα και την απαιτούμενη πληροφορία για την συγκεκριμένη θέση που πρέπει να έχουν.

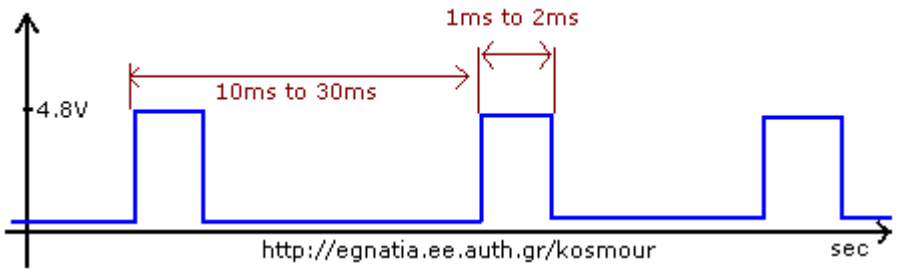
το εσωτερικό ενός servo

Τα servo αυτά εκτός από τον μοντελισμό χρησιμοποιούνται ευρύτατα και σε άλλες εφαρμογές όπως στην ρομποτική, σε συστήματα αυτόματου ελέγχου, σε special effects για τον κινηματογράφο κτλ.

Κάθε servo συνδέεται μέσω καλωδίου με υποδοχή πάνω στο δέκτη. Το καλώδιο αυτό διαθέτει 3 αγωγούς που διακρίνονται εξωτερικά και έχουν διαφορετικό χρώμα. Οι δύο από τους τρεις αγωγούς αποτελούν την παροχή για το σέρβο. Ο τρίτος αγωγός είναι ο πιο ενδιαφέρων αφού "μεταφέρει" από τον δέκτη την πληροφορία θέσης του servo. Η παραπάνω πληροφορία εμπεριέχεται στην χρονική διάρκεια ενός παλμού PWM σαν του παρακάτω που επαναλαμβάνεται κάθε τόσο.



Στα συστήματα των "2 ms" η παλμοσειρά που φτάνει στο servo είναι όπως η παρακάτω:



Η θέση του servo είναι ανάλογη της διάρκειας του παλμού που είναι από 1ms ως 2ms.

2.7 Το δικό μας αερομοντέλο

Το αερομοντέλο που χρησιμοποιήθηκε είναι τύπου Cessna Skylane και στηρίχθηκε στην ιδέα ενός εκπαιδευτικού μοντέλου. Αυτό προϋποθέτει σύμφωνα με τα παραπάνω μια σχετική δίεδρη επιφάνεια φτερών για την καλή συμπεριφορά αλλά και για την πιο εύκολη επαναφορά του κατά τη διάρκεια της πτήσης. Πολύ σημαντικό ήταν η σχεδίαση του, όπου το φτερό βρισκόταν στην επάνω μεριά της ατράκτου κάτι το οποίο προσδίδει μεγαλύτερη σταθερότητα. Αυτού του είδους η σχεδίαση προτείνεται για έναν αρχάριο που θα ασχοληθεί με τον αερομοντελισμό ενώ το καθιστά νωχελικό στις κινήσεις του για ακροβατικές φιγούρες. Αυτός ήταν και ο στόχος γιατί χρειαζόταν ένα αρκετά σταθερό μοντέλο ικανό για ομαλή πτήση έτσι ώστε οι εξωτερικές συνθήκες όπως οι πλευρικοί άνεμοι να μη το επηρεάζουν σε μεγάλο βαθμό.

Επίσης πέρα από τη σταθερότητα υπήρχε και η απαίτηση του χώρου αλλά και του βάρους. Το μοντέλο θα έπρεπε να έχει αρκετό χώρο στο εσωτερικό του έτσι ώστε να μπορεί να φιλοξενήσει το κυκλωματικό μας με σχετική ευρυχωρία αλλά και με την κατάλληλη προστασία του από τυχόν αποκόλλησή του από την άτρακτο. Όλα αυτά όμως προσδίδουν στο μοντέλο επιπλέον βάρος το οποίο θα πρέπει να είναι πάντα σχετικό με την ικανότητα ανύψωσης των φτερών αλλά και του κινητήρα. Γι' αυτό, το μοντέλο σχεδιάστηκε ειδικά για αυτό το σκοπό με αρκετά μεγαλύτερο χώρο στο εσωτερικό του από ότι τα πιο κλασικά αερομοντέλα που κυκλοφορούν στο εμπόριο. Επίσης τοποθετήθηκε κινητήρας στα 0.60ci (~10.0cc) ικανός να ανταπεξέλθει στο τελικό βάρος αλλά και στο μεγάλο μέγεθος του μοντέλου μας. Με αυτή τη διαμόρφωση το μοντέλο μπορεί να σηκώσει από 3-5 κιλά επηρεάζοντας πάντα την πτητική του συμπεριφορά. Το άνοιγμα φτερών είναι 1.85 m ενώ το τελικό βάρος διαμορφώθηκε γύρω στα 5.2 κιλά μαζί με τον κινητήρα από τα 3.8 κιλά αρχικού βάρους.

Στο μοντέλο υπάρχουν τρεις επιφάνειες ελέγχου aileron, elevator, rudder κάτι το οποίο μαζί με τον έλεγχο του κινητήρα απαιτούσε τουλάχιστον 4 κανάλια. Η τηλεκατεύθυνση που χρησιμοποιήθηκε είναι 6 καναλιών της FUTABA αφού χρειαζόμασταν τουλάχιστον ένα επιπλέον κανάλι για τον έλεγχο της ρομποτικής λειτουργίας. Ο τρόπος με τον οποίο έγινε η αναγνώριση του 5^{ου} καναλιού από τον microcontroller θα το δούμε στη παράγραφο 4.3 όπου αναλύεται το κυκλωματικό μέρος.

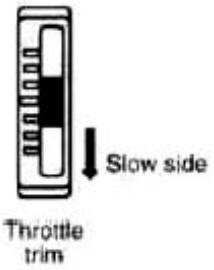
Επιπλέον χρειάστηκε να γίνουν κάποιες επεμβάσεις στο μοντέλο όπως στηρίγματα και τρύπες χωρίς να επηρεάζουν σε μεγάλο βαθμό την αεροδυναμική του. Πέρα από την τοποθέτηση της τελικής πλακέτας, επιπλέον προσαρμόσαμε μια LCD οθόνη, τη κεραία του GPS, εξωτερικές υποδοχές για τον programmer και της σειριακής σύνδεσης με τον υπολογιστή.

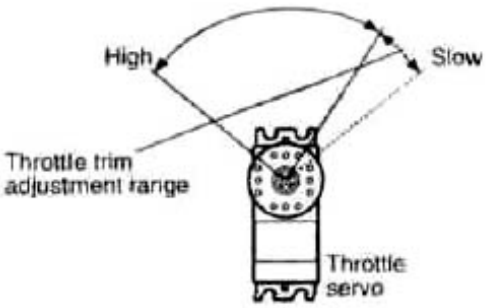
2.7.1 Λειτουργία Τηλεκατεύθυνσης

Η τηλεκατεύθυνση ήταν το μοντέλο της Futaba Skyport FP-6VA σε συνδυασμό με τον δέκτη FP-R127DF της ίδιας εταιρίας.

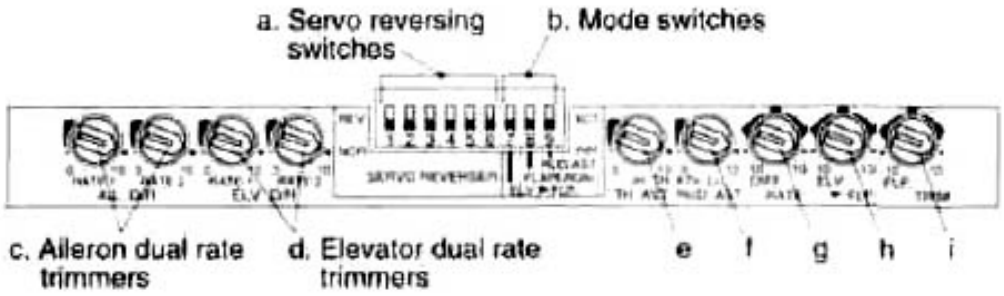
Ο πομπός με τον δέκτη επικοινωνούν στη συχνότητα των 72.150 MHz και μπορεί να αλλαχθεί σε περίπτωση που βρεθεί να χρησιμοποιείται η ίδια συχνότητα στη περιοχή.

Στην τηλεκατεύθυνση υπάρχουν οι μοχλοί ελέγχου των τεσσάρων βασικών καναλιών και τέσσερις ρυθμιστές (trim) αυτών. Οι ρυθμιστές αυτοί χρησιμοποιούνται για την εξισορρόπηση των επιφανειών ελέγχου σε θέσεις έτσι ώστε το αερομοντέλο να έχει ευθεία πορεία (level flight).





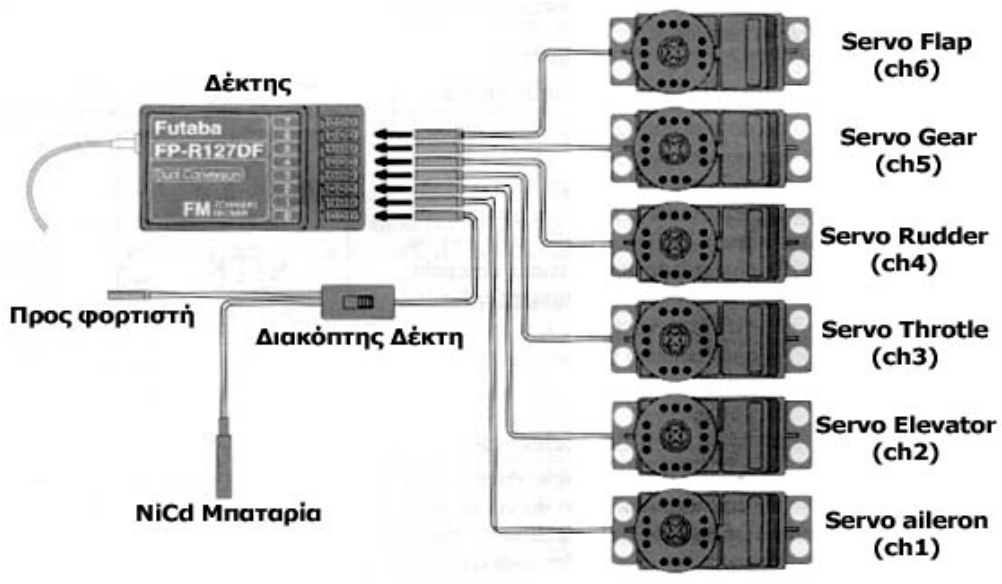
Το 5^{o} και 6^{o} κανάλι βρίσκονται στους διακόπτες δίπλα στη κεραία. Στο πίσω μέρος βρίσκονται οι ρυθμιστές για κάθε servo-κινητήρα. Οι ρυθμιστές αυτοί επηρεάζουν λειτουργίες όπως αντιστοίχηση μοχλού με κανάλι, αντίθετη φορά, μίξεις καναλιών, ρύθμιση Rate, ευαισθησία και κέντρο του κάθε servo.



Στον δέκτη η τροφοδοσία γίνεται με μπαταρία NiMH 5 Volt στη θέση “B”.

Ο διακόπτης που είναι ενδιάμεσα εκτός από το να ανοιγοκλείνει τον δέκτη ανοιγοκλείνει αντίθετα και το βύσμα φόρτισης έτσι ώστε όταν είναι σε χρήση οι μπαταρίες να μη φορτίζουν.

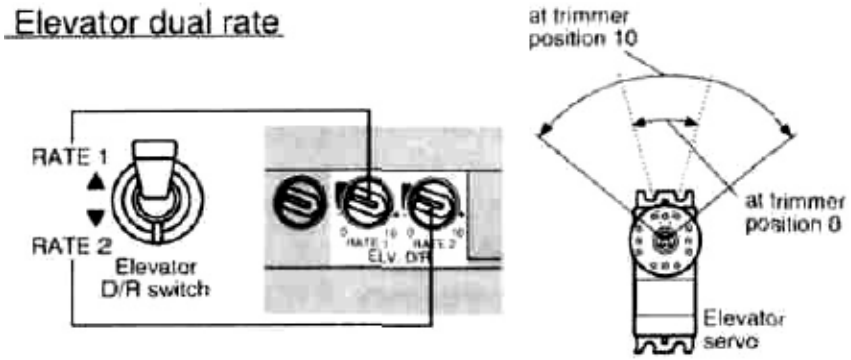
Η συνδεσμολογία του δέκτη με κάθε servo-κινητήρα φαίνεται στο σχήμα:



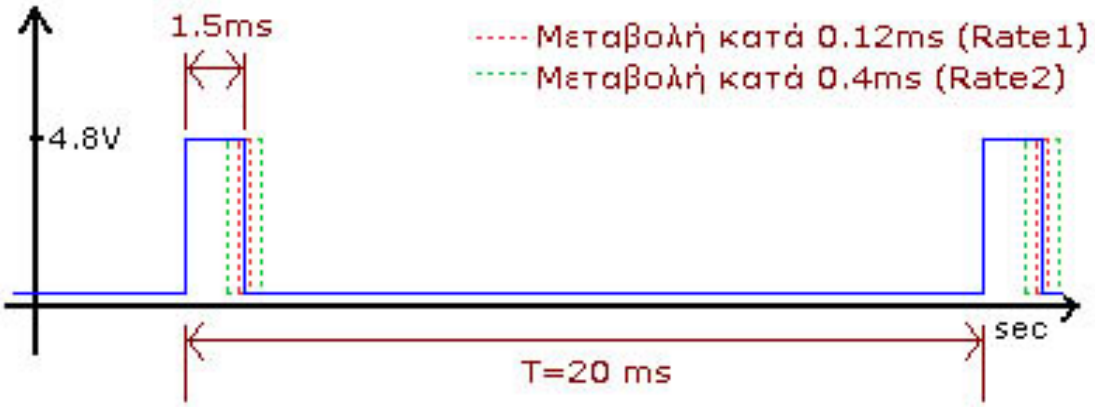
Το 5^{ο} κανάλι προοριζόταν για αερομοντέλο που έχει την ικανότητα να ανοιγοκλείσει το σύστημα προσγείωσης, ενώ το 6^{ο} κανάλι για αερόφρενα ή δεύτερα aileron. Το δεύτερο για παράδειγμα επιτυγχάνεται με μίξη του 1^{ου} και 6^{ου} καναλιού. Η έβδομη θέση είναι για κάποια επιπλέον λειτουργία η οποία ελέγχεται καθαρά από μίξη άλλων δύο. Στην περίπτωσή μας δεν χρειαστήκαμε καμία μίξη γιατί τα servo των aileron(δεξί και αριστερό) είναι βραχυκυκλωμένα στο κανάλι 1 ρυθμισμένα να εκτελούν αντίθετη κίνηση. Σε περίπτωση που και τα δυο servo κατέληγαν στο δέκτη θα έπρεπε να κάνουμε μίξη στη τηλεκατεύθυνση.

Το Rate είναι μια λειτουργία η οποία υπάρχει στα κανάλια των aileron και elevator δηλαδή των βασικών για πλοήγηση του αερομοντέλου. Με αυτή μπορούμε να κάνουμε την αντίδραση των servo πιο “χαλαρή” από ότι αν ήταν σε κανονική φάση. Αυτό βρίσκει

εφαρμογή σε περιπτώσεις εκπαιδευόμενου όπου οι απότομες αντιδράσεις είναι ανεπιθύμητες.

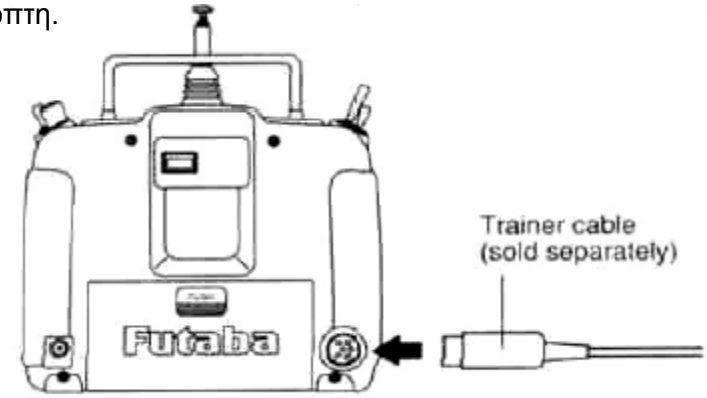


Μετρώντας σε παλμογράφο το σήμα του δέκτη προς τα servo πήραμε παλμό 50 Hz. Στο aileron για παράδειγμα ο παλμός PWM ήταν ρυθμισμένος στα 1,5 msec και Duty Cycle στα 7,5%. Κατά τη αλλαγή μας από Rate1 σε Rate2 μετακινώντας τον μοχλό των aileron καταγράψαμε μεταβολή 0.12 και 0.4 msec αντίστοιχα.



Κάθε servo έχει ρύθμιση κατάλληλη για να ισορροπεί η επιφάνεια ελέγχου στο κάθε φτερό, ανάλογα με τη θέση του κάθε servo στο εσωτερικό του αερομοντέλου.

Σε περίπτωση εκπαιδευόμενου υπάρχει και το καλώδιο trainer. Αυτό ενώνει δύο πομπούς όπου ο ένας είναι του εκπαιδευτή και ο άλλος του εκπαιδευόμενου. Με τον τρόπο αυτό ο εκπαιδευτής έχει την επιλογή του ελέγχου (από ποιο πομπό θα γίνεται) μέσω ενός διακόπτη.



3 Επιμέρους εργαλεία(Modules)

3.1 Ο μικροελεγκτής MR162

Το MR162 είναι μια ολοκληρωμένη πλακέτα που περιέχει έναν μικροελεγκτή Atmel ATmega162, το κύκλωμα προγραμματισμού του (ISP - In-System Programming) και έναν κρύσταλλο 8 MHz, ο οποίος αντικαταστάθηκε με έναν 16 MHz για μεγαλύτερη ταχύτητα επεξεργασίας δεδομένων.

Ο ATmega162 είναι ένας πολύ ισχυρός μικροελεγκτής ο οποίος περιέχει 16 KBytes προγραμματιζόμενης μνήμης Flash, 512 Bytes μνήμης Eeprom για αποθήκευση σταθερών αριθμών και 1 KByte μνήμης SRAM. Περιέχει επίσης τέσσερις timers, δύο των 8bit και δύο των 16bit, δύο USARTs (Universal Synchronous and Asynchronous serial Receiver and Transmitter) για επικοινωνία με σειριακές συσκευές, όπως και πολλά άλλα περιφερειακά. Για σωστή λειτουργία ο μικροελεγκτής χρειάζεται τάση συνεχούς ρεύματος από 2.7 Volt έως τα 5.5 Volt, στην περίπτωσή μας όμως (16MHz κρύσταλλος) η τάση πρέπει να βρίσκεται σίγουρα πάνω από τα 4.5 Volt και κάτω από τα 5.5 Volt, για να δουλεύει ομαλά.

Οι τεχνικοί της active-robots σχεδιάσανε έτσι το MR162 ώστε να είναι εύκολη η προσάρτησή του σε ένα μεγαλύτερο κύκλωμα, ο προγραμματισμός του και να περιέχει κάποιες αναγκαίες λειτουργίες, όπως το κουμπί reset, το led ένδειξης προγραμματισμού και μια ασφάλεια ανάστροφης τάσης μέσω μιας διόδου. Βάσει των παραπάνω ο MR162 αποδεικνύεται ένα πολύ χρήσιμο εργαλείο πάνω στο οποίο “χτίζεται” το υπόλοιπο σύστημα.

Το MR162 είναι η καρδιά του συστήματος, όπου γίνεται η συγκέντρωση και επεξεργασία των πληροφοριών, η επιλογή της επόμενης κίνησης και η αποθήκευση των δεδομένων, ώστε να γίνει τελικά ο έλεγχος της πραγματικής πορείας του σκάφους. Αυτός ακριβώς είναι και ο λόγος για τον οποίο τοποθετήθηκε ένας γρηγορότερος κρύσταλλος από αυτόν που είχε στην αρχή. Από τις δοκιμές που έγιναν φάνηκε ότι δουλεύει πολύ ομαλά και δεν υπάρχει μεγάλη καθυστέρηση ακόμη και σε σύνθετες μαθηματικές πράξεις. Πράγμα το οποίο διευκόλυνε πολύ, καθώς για τον υπολογισμό τις πορείας και την εύρεση τις επόμενης κίνησης θα χρειαστούν πολλές από αυτές.

3.1.1 Η θύρα σειριακής μετάδοσης δεδομένων (USART)

Οι δύο σειριακές θύρες του μικροελεγκτή αποδείχθηκαν αρκετά χρήσιμες καθώς ήταν εύχρηστες, γρήγορες και αξιόπιστες. Αυτοί είναι και οι λόγοι για τους οποίους στηρίχθηκε ο κυρίως κορμός του συστήματος πάνω σε αυτές.

Στον ATmega162 το καθένα από τα USARTs αποτελείται από τρία κύρια μέρη:

- Τον Clock generator (γεννήτρια παλμών)
- Τον Transmitter (πομπός) και
- Τον Receiver (δέκτης)

Η γεννήτρια παλμών είναι υπεύθυνη για να δίνει τον ρυθμό μετάδοσης δεδομένων (BAUD rate), στον πομπό και στο δέκτη. Υπάρχει βέβαια και η δυνατότητα να συνδεθεί με εξωτερική πηγή παλμών, όπου οι παλμοί θα έρχονται από τον master του συστήματος στον slave μικροελεγκτή. Ο πομπός έχει την δυνατότητα να διαχειριστεί διαφόρων ειδών πακέτα δεδομένων και μάλιστα χωρίς καθυστέρηση λόγω του καταχωρητή εγγραφής (write buffer) που περιέχει. Ο δέκτης, που είναι και το πιο σύνθετο κομμάτι του USART, είναι εξοπλισμένος με όλα τα απαραίτητα εργαλεία για έλεγχο λαθών στην μετάδοση δεδομένων, όπως frame error, data Overrun και Parity errors.

3.1.2 Timers

Οι timers είναι καταχωρητές (είτε 8 bit είτε 16 bit), οι οποίοι όταν βρεθούν σε κατάσταση υπερχείλισης (overflow), δημιουργούν interrupt στην σειριακή ροή του προγράμματος του μικροελεγκτή και μας δίνουν την δυνατότητα να τρέξουμε ένα άλλο κομμάτι προγράμματος για να χειριστούμε την συγκεκριμένη κατάσταση. Η ταχύτητα υπερχείλισης μπορεί να ρυθμιστεί είτε εσωτερικά από την ταχύτητα του κρυστάλλου δια έναν διαιρέτη (prescaler), είτε εξωτερικά από κάποια πηγή παλμών που είναι συνδεδεμένη στο κατάλληλο ακροδέκτη του μικροελεγκτή. Ο τρόπος λειτουργίας των timers είναι να αυξάνουν την τιμή που περιέχουν κατά ένα μέχρι να γίνει υπερχείλιση κάθε φορά που «χτυπά» ένας παλμός. Έτσι αναλόγως με τον διαιρέτη, την ταχύτητα του κρυστάλλου ή της εξωτερικής πηγής μπορούμε να μετρήσουμε ένα χρόνο ή εξωτερικά γεγονότα με αρκετά καλή ακρίβεια.

3.1.3 Προγραμματισμός

Για τον προγραμματισμό χρησιμοποιήθηκε το πρόγραμμα CodevisionAVR 1.24.5. Το Codevision είναι ένα ολοκληρωμένο πρόγραμμα το οποίο περιέχει C compiler, assembler, downloader και μαζί με έναν πολύ εύχρηστο wizard για τις βασικές ρυθμίσεις γίνεται ένα ικανοποιητικής απόδοσης εργαλείο στα χέρια ενός προγραμματιστή.

Πριν ξεκινήσει κάποιος να προγραμματίζει θα πρέπει να ανοίξει το CodeWizardAVR, όπου εκεί θα δηλώσει το είδος του μικροελεγκτή που θέλει να προγραμματίσει, την ταχύτητα του κρυστάλλου του, όπως και τις αρχικές ρυθμίσεις για τα περιφερειακά που

χρειάζεται στο πρόγραμμά του. Από το μενού file επιλέγουμε «Generate, save and exit» και δίνουμε τοποθεσία αποθήκευσης για το project. Αμέσως μετά βρισκόμαστε μπροστά σε ένα έτοιμο πρόγραμμα, το οποίο περιέχει όλες τις αρχικοποιήσεις και τις αναγκαίες συναρτήσεις, όπως interrupts και συναρτήσεις ελέγχου διαφόρων καταστάσεων. Αφού γραφεί το πρόγραμμά μας ακολουθεί compile και built και στη συνέχεια από το παράθυρο CodevisionAVR Chip Programmer επιλέγουμε program all και το πρόγραμμα "φορτώνεται" στον μικροελεγκτή.

3.2 GPS(Global Positioning System)

Το GPS είναι ένα σύστημα το οποίο μπορεί να μας δώσει με αρκετά καλή ακρίβεια την θέση μας, οπουδήποτε, πάνω στην γη. Τα ραδιοσήματα που χρησιμοποιεί η τεχνολογία του GPS χρησιμοποιούνται για εύρεση θέσης σημείου και προσανατολισμού από το 1920 με το σύστημα LORAN (Long Range Aid and Navigation). Ο τρόπος που δούλευε αυτό το σύστημα ήταν μετρώντας την χρονική διαφορά άφιξης μεταξύ των σημάτων που προέρχονταν από συγκεκριμένα σταθερά σημεία.

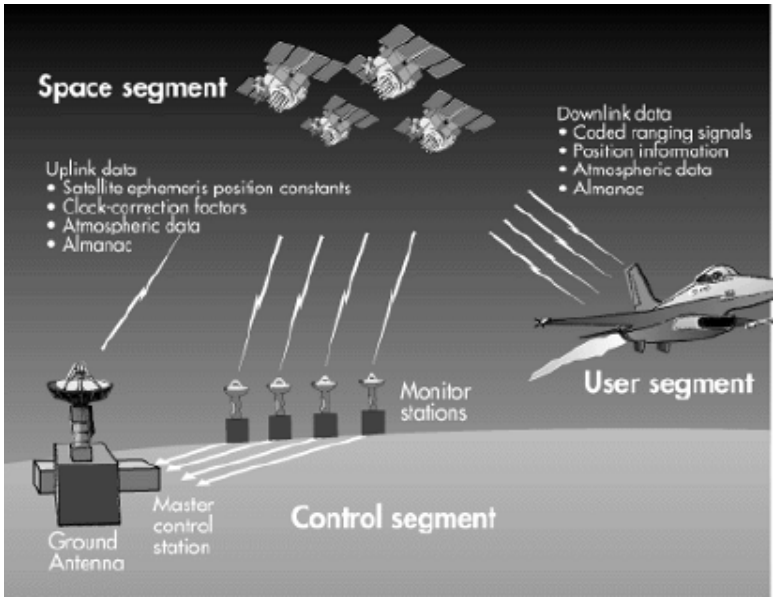
Το πρώτο βήμα για το δορυφορικό σύστημα GPS έγινε το 1957 όταν ο Ρωσικός δορυφόρος Sputnik χρησιμοποίησε ένα πομπό ραδιοσημάτων για να μεταδώσει πληροφορία τηλεμετρίας. Αργότερα στο εργαστήριο «John Hopkins Applied Physics» ανακάλυψαν ότι στα σήματα του δορυφόρου εμφανιζόταν το φαινόμενο Doppler.

Sputnik satellite

Το φαινόμενο Doppler είναι το φαινόμενο που παρατηρούμε όταν ένα ασθενοφόρο διέρχεται από μπροστά μας με μεγάλη ταχύτητα. Καθώς μας πλησιάζει ο ήχος της σειρήνας έχει υψηλότερη συχνότητα και μεγαλύτερο tempo, ενώ καθώς απομακρύνεται από εμάς η συχνότητα και το tempo ελαττώνονται.

Κάποιοι Αμερικάνοι επιστήμονες ανακάλυψαν ότι αν ήταν γνωστή η ακριβής τροχιά του δορυφόρου, θα μπορούσαν να βρουν την ακριβή τοποθεσία τους πάνω στην γη, αν

μετρούσαν τα σήματα που αποστέλλονται από το δορυφόρο και το φαινόμενο Doppler που εμφανιζόταν πάνω σε αυτά.

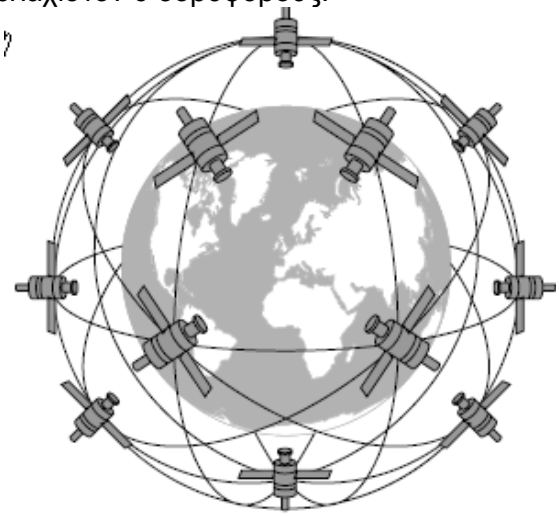


Το 1960 είχαν εξελιχθεί διάφορα δορυφορικά συστήματα εύρεσης θέσης. Ο Αμερικανικός στρατός, το ναυτικό και η αεροπορία εργάζονταν ξεχωριστά για την ανάπτυξη ενός δορυφορικού συστήματος εύρεσης θέσης με αρκετά καλή ακρίβεια και λειτουργία ανεξάρτητη καιρού και ώρας. Το 1973 η αεροπορία επιλέχτηκε για την συνέχιση του έργου μετά την συνένωση των προσπαθειών και των τριών φορέων. Το αποτέλεσμα αυτής της προσπάθειας ήταν το NAVSTAR-GPS (Navigational Satellite Timing and Ranging Global Positioning System). Το 1974 απογειώθηκε ο πρώτος δορυφόρος για δοκιμαστικούς σκοπούς. Στα μέσα της δεκαετίας του ’80 είχαν απογειωθεί αρκετοί δορυφόροι για την λειτουργία του συστήματος και το 1994 ο Αμερικάνικός στρατός θεώρησε το σύστημα σε πλήρη λειτουργία.

Η Αμερικανική κυβέρνηση δεν ενδιαφερόταν, για την χρήση του GPS, μόνο για σκοπούς εύρεσης θέσης πάνω στη γη. Ένα τέτοιο σύστημα θα ήταν πολύ χρήσιμο για την κατεύθυνση οπλικών συστημάτων (π.χ. πυραύλων). Οπλικά συστήματα όπως οι Tomahawk cruise missiles κατασκευάστηκαν ώστε να χτυπάνε με ακρίβεια στόχους, συνδυάζοντας την τεχνολογία του GPS και την τεχνολογία της οπτικής ανάλυσης εικόνας και υπέρυθρης ακτινοβολίας.

Tomahawk cruise missile

Από τους 24 GPS δορυφόρους που είναι σε τροχιά γύρω από την γη, οι 21 λειτουργούν πλήρως και οι άλλοι 3 είναι βοηθητικοί. Οι δορυφόροι πετάνε 19313 χιλιόμετρα πάνω από την επιφάνεια της θάλασσας με ταχύτητα 11265 χιλιόμετρα ανά ώρα. Ένας δορυφόρος χρειάζεται 12 ώρες για να ολοκληρώσει μια τροχιά γύρω από τη γη και περνάει από το ίδιο σημείο κάθε 24 ώρες. Οι τροχιές των δορυφόρων είναι έτσι σχεδιασμένες ώστε ένας δεκτής GPS σε οποιοδήποτε σημείο πάνω στη γη να μπορεί να λαμβάνει σήμα από τουλάχιστον 6 δορυφόρους.



Τα σήματα των δορυφόρων είναι δυο ειδών, C/A-code και P-code:

- C/A-code(Coarse acquisition) είναι αυτό που λαμβάνουν οι εμπορικοί δέκτες GPS και στέλνεται στη συχνότητα 1575,42 MHz. Το πλεονέκτημα του C/A-code είναι ότι είναι πιο γρήγορο στη χρήση, ενώ το μειονέκτημα ότι έχει μικρότερη ακρίβεια και ότι είναι πιο ευάλωτο στα παράσιτα.
- Το P-code (precision) προσφέρει πληροφορία μεγαλύτερης ακρίβειας και είναι λιγότερο ευάλωτο στα παράσιτα, χρησιμοποιείται κυρίως για στρατιωτικούς σκοπούς και εκπέμπει στα 1227,6 MHz.

Η ακρίβεια του GPS για το C/A-code, όπως έχει ανακοινωθεί από την Αμερικάνικη κυβέρνηση είναι 15 μέτρα, αλλά το πόσο ακριβής θα είναι η εκάστοτε μέτρηση εξαρτάται από την ποιότητα του σήματος και από τα εμπόδια που βρίσκονται ανάμεσα.

3.2.1 Το GPS-PS1E

Το PS1E είναι GPS δέκτης που χρησιμοποιήσαμε, είναι κατασκευασμένο από την εταιρία μ-Blox και είναι βασισμένο στο chip SirfStar I/LX, το οποίο κατασκευάζεται από την Sirf Technology. Το μέγεθος του είναι 82.5 mm x 32 mm και παρέχει όλη την επεξεργασία σήματος από την κεραία μέχρι την σειριακή έξοδο δεδομένων του. Για την σωστή λειτουργία του χρειάζεται τάση 5 Volt, η οποία μετασχηματίζεται εσωτερικά σε 3.3 Volt, για μικρότερη κατανάλωση.

Οι αρχικές ρυθμίσεις του PS1E είναι:

Sirf binary protocol
19200 baud rate
8 data bits, no parity, 1 stop bit

Το PS1E μπορεί να ξεκινήσει την λειτουργία του με διάφορους τρόπους, αναλόγως την πληροφορία που έχει στην διάθεσή του και στο πόσο καλή είναι η ορατότητα προς τον ουρανό.

Κάποια από τα πιθανά σενάρια είναι:

- Cold start: σε αυτή την περίπτωση ο δέκτης δεν έχει καμία πληροφορία για την τελευταία του θέση πάνω στην γη, τον πραγματικό χρόνο και το που βρίσκονται οι δορυφόροι. Έτσι ο δέκτης ξεκινάει να ψάχνει τα σήματα των δορυφόρων στα “τυφλά”. Αυτή η περίπτωση είναι συνηθισμένη εάν δεν υπάρχει εφεδρική μπαταρία και είναι χρονοβόρα.
- Warm start: σε αυτή την περίπτωση ο δέκτης ξέρει (λόγω της εφεδρική μπαταρίας) την τελευταία του θέση, τον πραγματικό χρόνο και τις τελευταίες θέσεις των δορυφόρων. Έτσι η εκκίνηση είναι πολύ πιο γρήγορη.
- Hot start: σε αυτή την περίπτωση ο δέκτης ήταν εκτός λειτουργίας για λιγότερο από 2 ώρες και χρησιμοποιεί τα δεδομένα που έχει αποθηκεύσει στην προσωρινή του μνήμη.

3.2.2 Πρωτόκολλο NMEA

Η NMEA δημιούργησε μια διεπαφή για την επικοινωνία μεταξύ διαφόρων ηλεκτρονικών συσκευών που χρησιμοποιούνται για την ακτοπλοΐα. Επίσης ένας από του σκοπούς της δημιουργίας του ήταν η ευκολότερη επικοινωνία των συσκευών αυτών με

ηλεκτρονικούς υπολογιστές, έτσι τα περισσότερα προγράμματα εύρεσης θέσης είτε απαιτούν, είτε δέχονται και δεδομένα σε πρωτόκολλο NMEA.

Η βασική ιδέα του πρωτοκόλλου NMEA είναι η αποστολή μιας πρότασης κάθε φορά η οποία περιέχει την πληροφορία και είναι ανεξάρτητη από τις άλλες προτάσεις. Όλες οι βασικές προτάσεις ξεκινούν με δυο αναγνωριστικούς χαρακτήρες, ως προς την συσκευή που το χρησιμοποιεί (για το GPS, GP), τα οποία ακολουθούνται από τρεις χαρακτήρες που προσδιορίζουν το είδος της πρότασης που ακολουθεί, συνεπώς και τα δεδομένα. Η NMEA έχει δώσει άδεια στους κατασκευαστές να δημιουργούν δικές τους προτάσεις σύμφωνα με τις ανάγκες τους.

Κάθε πρόταση ξεκινάει με το χαρακτήρα '$' και τελειώνει με τον χαρακτήρα αλλαγής γραμμής και μπορεί να έχει μέγιστο μέγεθος 80 χαρακτήρες. Η πληροφορία περιέχεται μέσα στην πρόταση και διαχωρίζεται από κώματα. Είναι πιο σωστό και θεμιτό τα προγράμματα που υποστηρίζουν το NMEA να χρησιμοποιούν τα κώματα για εύρεση της πληροφορίας που χρειάζεται και όχι τη σειρά που εμφανίζονται καθώς αυτό μπορεί να αλλάξει, αναλόγως τα δεδομένα.

Για τις διάφορες ανάγκες έχουν δημιουργηθεί διάφορες προτάσεις. Το PS1E χρησιμοποιεί τις παρακάτω:

GGA- global positioning system fixed data
GLL- geographic position –latitude, longitude
GSA- GNSS DOP and active satellites
GSV- GNSS satellites in view
PMC- recommended minimum specific GNSS data
VTG- course over ground speed

Κάποιες άλλες βασικές προτάσεις είναι:

AAM: waypoint arrival alarm
APA- auto pilot A sentence
RTE- route message
WPL- waypoint information
MSS- beacon receiver status information

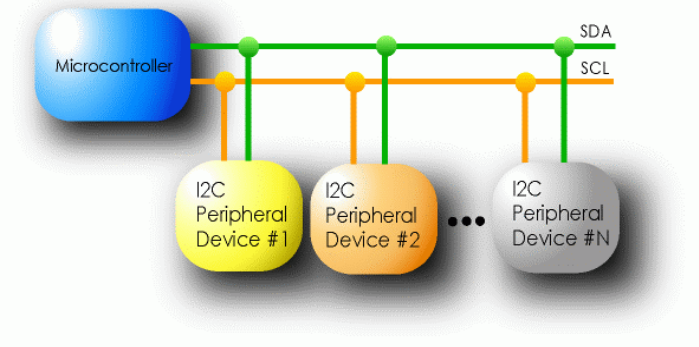
Στο πλαίσιο αυτής της πτυχιακής εργασίας χρησιμοποιήθηκε το GGA, από το οποίο λαμβανόταν η πληροφορία για το γεωγραφικό μήκος, το γεωγραφικό πλάτος και την απόσταση από την επιφάνεια της θάλασσας, συνεπώς ήταν γνωστό το τρισδιάστατο στίγμα του αεροπλάνου. Το μειονέκτημα είναι ότι η πληροφορία ύψους δεν είναι πολύ αξιόπιστη.

Το PS1E όταν ξεκινήσει σε cold start λειτουργεί σε πρωτόκολλο Sirf. Για να αλλάξει πρωτόκολλο πρέπει να σταλεί μια σειρά πληροφοριών μέσω της σειριακής θύρας. Αυτή η σειρά πληροφοριών μπορεί να βρεθεί είτε από το manual του PS1E, είτε από το πρόγραμμα SirfDemo, δίνοντας τις σωστές παραμέτρους για το πρωτόκολλο NMEA και baud rate 19200.

3.3 Πρωτόκολλο I2C

Το πρωτόκολλο επικοινωνίας I2C (ή Inter-IC) δημιουργήθηκε από την Philips Semiconductors στις αρχές της δεκαετίας του 1980 με αρχικό σκοπό την επικοινωνία μεταξύ της κεντρικής μονάδας επεξεργασίας και των περιφερειακών ολοκληρωμένων κυκλωμάτων μιας τηλεόρασης.

Figure 1: I2C 2-wire Interface



Το πρόβλημα που έλυσε το I2c ήταν ότι καθώς αναπτυσσόταν η τεχνολογία η σύνδεση περιφερειακών συσκευών με το σύστημα γινόταν όλο και πιο δύσκολη αλλά και πιο ακριβή, καθώς για την κατασκευή προϊόντων, όπως τηλεοράσεις, βίντεο και εξοπλισμός ήχου, χρειαζόντουσαν πολύ χρόνο για την σχεδίαση τους, λόγω των πολλών καλωδίων που απαιτούνται για την διευθυνσιοδότηση και για την μεταφορά δεδομένων. Αν σε αυτά προσθέσουμε και όλες τις ενδιάμεσες συσκευές που είναι αναγκαίες για την σωστή επικοινωνία το κόστος κατασκευής είναι αρκετά μεγάλο σε προϊόντα μαζικής παραγωγής. Σε τέτοιες περιπτώσεις ό,τι μπορεί να παραλειφθεί σημαίνει μεγαλύτερο κέρδος για τον κατασκευαστή και πιο εύχρηστα προϊόντα για τον χρήστη. Ακόμα, όσοι λιγότεροι αγωγοί υπάρχουν στο PCB τόσο μικρότερα είναι τα εξωτερικά φαινόμενα που μπορούν να το επηρεάσουν, όπως στατικός ηλεκτρισμός και ηλεκτρομαγνητικές παρεμβολές.

Οι έρευνες της Philips έγιναν στα εργαστήρια στο Eindhoven της Ολλανδίας με σκοπό να ξεπεραστούν οι δυσκολίες που προαναφέρθηκαν και είχαν σαν αποτέλεσμα ένα πρωτόκολλο επικοινωνίας, το I2C (Inter-IC), το οποίο χρειαζόταν μόνο δύο καλώδια για τη λειτουργία του.

Το I2C στο φυσικό επίπεδο αποτελείται από δύο, αμφίδρομα στον τρόπο μετάδοσης, καλώδια στα οποία συνδέονται οι συσκευές. Η πιο συνηθισμένη ονομασία που τους δίνεται είναι SDA (Serial Data) και SCL (Serial Clock), όπως φαίνεται το πρώτο είναι το καλώδιο που χρησιμοποιείται για μεταφορά δεδομένων και το δεύτερο για τον συγχρονισμό των συσκευών.

Κάθε συσκευή που είναι συνδεδεμένη πάνω σε έναν I2C δίαυλο έχει μια μοναδική διεύθυνση και μπορεί να λειτουργήσει είτε σαν πομπός δεδομένων, είτε σαν δέκτης, αναλόγως την περίσταση. Βεβαίως, ένας ελεγκτής LCD λειτουργεί σαν δέκτης δεδομένων, ενώ ένα chip μνήμης μπορεί να είναι και πομπός και δέκτης. Για την επικοινωνία εφαρμόζεται αρχιτεκτονική αφέντη-δούλου (master-slave) και μάλιστα το I2C

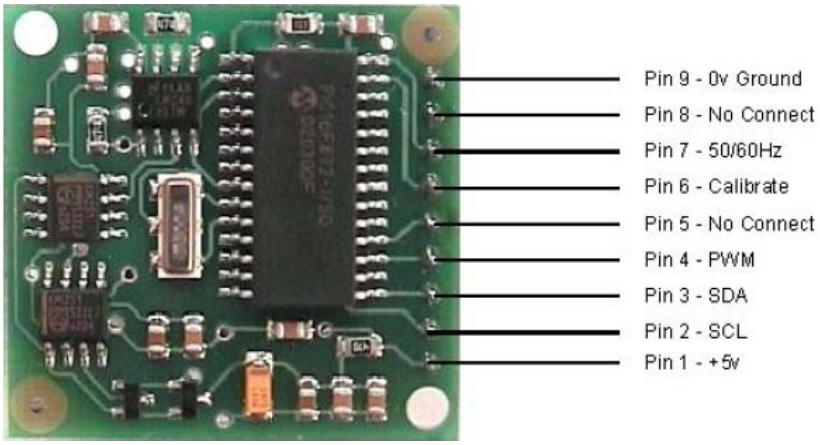
είναι multi-master bus, που σημαίνει ότι μπορούν να συνυπάρξουν περισσότεροι του ενός masters προσκολλημένοι σε έναν δίαυλο I2C.

Η επικοινωνία μεταξύ master και slave γίνεται με μια συγκεκριμένη διαδικασία. Αρχικά ο master δημιουργεί μια κατάσταση START (SDA SCL γειωμένα). Οι slaves όταν αντιληφθούν κατάσταση START στον δίαυλο μπαίνουν σε κατάσταση αναμονής για τα εισερχόμενα δεδομένα. Αμέσως μετά ο master στέλνει στον δίαυλο τον αριθμό της θέσης μνήμης που θέλει να χρησιμοποιήσει με ένα ενδεικτικό για το τι θέλει να κάνει, να διαβάσει ή να στείλει δεδομένα. Τώρα αφού οι slaves έχουν όλοι λάβει την ενεργή διεύθυνση ελέγχουν αν είναι η δική τους και αν δεν είναι περιμένουν μέχρι να ελευθερωθεί ο δίαυλος. Αν είναι η δική τους τότε παράγουν ένα σήμα επιβεβαίωσης (ACKNOWLEDGEMENT). Όταν ο master λάβει το σήμα επιβεβαίωσης, μπορεί να ξεκινήσει η μετάδοση δεδομένων, είτε αποστολή είτε λήψη. Όταν τελειώσει η διαδικασία μετάδοσης ο master δημιουργεί κατάσταση STOP στον δίαυλο, ελευθερώνοντάς τον. Οι συσκευές παραμένουν σε κατάσταση αναμονής για την επόμενη μεταφορά δεδομένων.

Το μέγιστο μέγεθος ενός διαύλου επικοινωνίας I2C μπορεί να φτάσει περίπου 10 με 11 μέτρα. Βασικό μειονέκτημα που παρουσιάζεται είναι ο θόρυβος που προστίθεται στο σήμα από τα μεγάλα καλώδια. Αυτός ο θόρυβος μπορεί να επηρεάσει το σήμα τόσο ώστε να μην μπορεί να διαβαστεί. Μια λύση σε αυτό το πρόβλημα είναι να μειωθεί η συχνότητα επικοινωνίας του διαύλου έτσι ώστε να μην επηρεάζεται από παρεμβολές, με μειονέκτημα βεβαίως, την χαμηλή ταχύτητα μετάδοσης.

3.3.1 Η πυξίδα CMPS03

Η συσκευή πυξίδας που χρησιμοποιήθηκε είναι το CMPS03. Δημιουργήθηκε με σκοπό την εύρεση της κατεύθυνσης ενός ρομπότ πάνω στην γη. Ο κύριος αισθητήρας του CMPS03 είναι το chip ανίχνευσης μαγνητικού πεδίου, Philips KMZ51, το οποίο είναι αρκετά ευαίσθητο ώστε να ανιχνεύει το μαγνητικό πεδίο της γης.



Για την σωστή λειτουργία του το CMPS03 χρειάζεται τάση 5 Volt στα 15 mA και χρησιμοποιεί δύο τρόπους επικοινωνίας, η πρώτη είναι το πρωτόκολλο I2C και η δεύτερη ένας παλμός PWM. Στην εργασία αυτή χρησιμοποιήθηκε μόνο το I2C κερδίζοντας χώρο στο PCB και κάνοντας τα δεδομένα πολύ πιο εύκολα στην ανάκτηση και στην επεξεργασία.

Για την επικοινωνία I2C η πυξίδα χρησιμοποιεί τον τρόπο που περιγράψαμε παραπάνω:

MOS->i2c_start ()
MOS->i2c_write (Device_select_code)
MOS->i2c_write (register_number)
MOS->i2c_start ()
MOS->i2c_write(Device_select_code OR1) (ένδειξη ότι θέλουμε να διαβάσουμε)
MOS<- i2c_read ()

Οι καταχωρητές του CMPS03 ποικίλουν προς διευκόλυνση του προγραμματιστή κατά τη σχεδίαση του προγράμματος.

Register	Function
0	Software Revision Number
1	Compass Bearing as a byte, i.e. 0-255 for a full circle
2,3	Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-359.9 degrees.
4,5	Internal Test - Sensor1 difference signal - 16 bit signed word
6,7	Internal Test - Sensor2 difference signal - 16 bit signed word
8,9	Internal Test - Calibration value 1 - 16 bit signed word
10,11	Internal Test - Calibration value 2 - 16 bit signed word
12	Unused - Read as Zero
13	Unused - Read as Zero
14	Unused - Read as Undefined
15	Calibrate Command - Write 255 to perform calibration step. See text.

Οι τιμές που λαμβάνονται διαφέρουν ανάλογα με τον καταχωρητή.

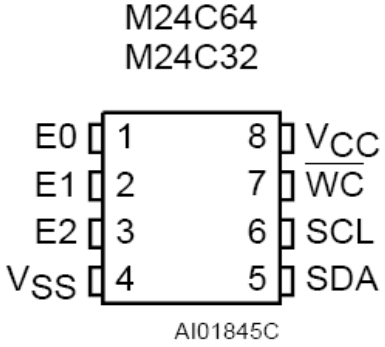
Αν για παράδειγμα, διαβάσουμε τον καταχωρητή 1, ο οποίος είναι 8 bit, θα έχουμε εύρος τιμών από 0 έως 255, πράγμα που θα βοηθούσε κάποιες εφαρμογές αλλά θα προσέθετε σφάλμα στην μέτρηση.

Για μεγαλύτερη ακρίβεια μπορούμε να χρησιμοποιήσουμε τους καταχωρητές 2 και 3, οι οποίοι είναι 16 bit και έχουν εύρος τιμών από 0 έως 3599, αντιπροσωπεύοντας μοίρες 0 έως 359,9.

3.3.2 Η μνήμη Eeprom

Η μνήμη που χρησιμοποιήθηκε για την αποθήκευση δεδομένων κατά την διάρκεια της πτήσης είναι η M24C64, η οποία έχει χωρητικότητα 64 kbit και για την επικοινωνία της με άλλες συσκευές, χρησιμοποιεί το πρωτόκολλο I2C.

Το M24C64 συμπεριφέρεται σαν slave στο δίαυλο. Για την περίπτωση που κάποια εφαρμογή χρειάζεται να συνυπάρχουν περισσότερες της μιας μνήμες ίδιου τύπου, αυτές δηλώνονται μονοσήμαντα στο δίαυλο μέσω των ακροδεκτών E0,E1,E2. Οι άκρες αυτές αντιπροσωπεύουν τα τρία λιγότερο σημαντικά (least significant) bits από το Device_select_code της συσκευής και αναλόγως με το αν είναι συνδεδεμένες στην γείωση ή στην τάση αντιπροσωπεύουν 0 ή 1.



	Device Type Identifier[1]				Chip Enable Address[2]			RW
	b7	b6	b5	b4	b3	b2	b1	b0
Device Select Code	1	0	1	0	E2	E1	E0	RW

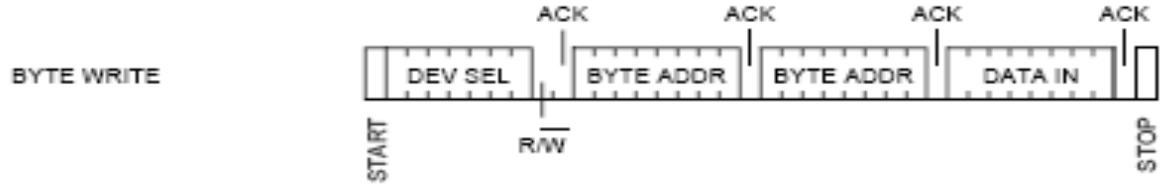
Η άκρη WC είναι η ασφάλεια εγγραφής. Όταν είναι συνδεδεμένη με την γείωση δεν υπάρχει ασφάλεια εγγραφής και η διαδικασία γίνεται κανονικά, ενώ όταν υπάρχει τάση η λειτουργία εγγραφής στην μνήμη απενεργοποιείται.

Η μνήμη είναι οργανωμένη σε 8192 σειρές των 8 bits και κάθε σειρά έχει μια διεύθυνση μνήμης των 16 bits. Υπάρχουν δύο τρόποι εγγραφής στην μνήμη, το Byte Write και το Page Write:

Byte Write:

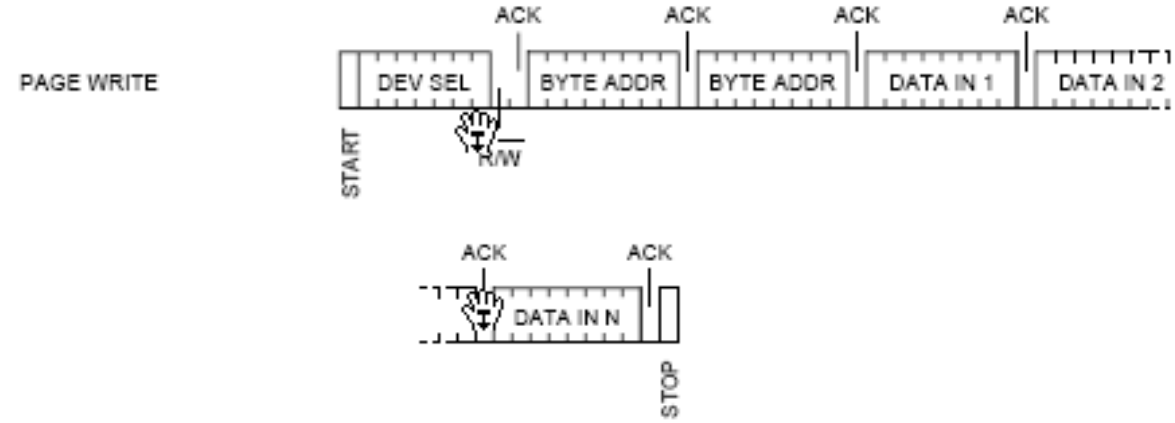
Η διαδικασία είναι παρόμοια με αυτή που περιγράφηκε. Αρχικά ο master του διαύλου δημιουργεί κατάσταση start στον διάδρομο και αποστέλλει το Device_select_code με ένδειξη εγγραφής στο τέλος (OR 1), η μνήμη απαντάει με acknowledgement και ο master αποστέλλει, εν συνεχεία, την διεύθυνση του byte που θέλει να γράψει σε 2 bytes

(πρώτα το περισσότερο σημαντικό byte και μετά το λιγότερο) και το byte πληροφορίας προς αποθήκευση, με την μνήμη να απαντάει με acknowledgement μετά από κάθε byte. Στην συνέχεια ο master δημιουργεί κατάσταση stop στον δίαυλο και η διαδικασία τερματίζεται.



Page Write:

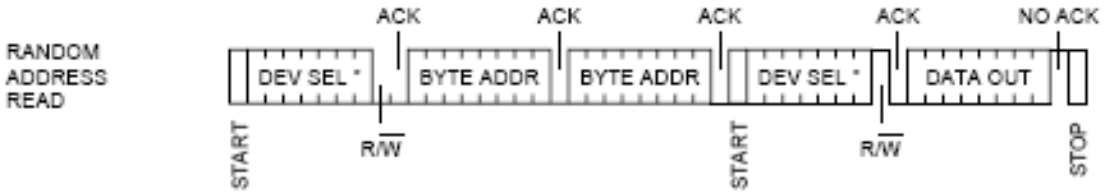
Η διαδικασία εγγραφής Page Write επιτρέπει την αποθήκευση 32 bytes με ένα μόνο κύκλο εγγραφής, αρκεί να βρίσκονται στην ίδια “σειρά” στην μνήμη. Αυτό σημαίνει ότι πρέπει να είναι ίδια τα περισσότερο σημαντικά (most significant) bit των διευθύνσεών τους και να είναι συνεχόμενα. Αν αποσταλούν περισσότερα από 32 bytes τότε η εγγραφή θα συνεχίσει από την θέση μνήμης που ξεκίνησε (roll-over) διαγράφοντας τα υπάρχοντα δεδομένα. Η διαδικασία είναι ίδια με την προηγούμενη με την διαφορά ότι μετά την αποστολή του πρώτου byte πληροφορίας ο master συνεχίζει να στέλνει bytes και η μνήμη τα αποθηκεύει, αυξάνει τον εσωτερικό της μετρητή και απαντάει με acknowledgement. Αυτή η διαδικασία μπορεί να συνεχιστεί μέχρι 32 bytes και τελειώνει με τον master να δημιουργεί κατάσταση stop στο δίαυλο.



Για την διαδικασία προσπέλασης της μνήμης υπάρχουν τρεις τρόποι, το Random Address Read, το Current Address Read και το Sequential Read:

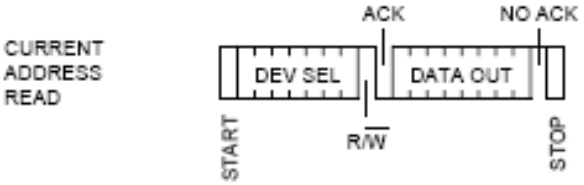
Random Address Read:

Για να προσπελαστεί μια συγκεκριμένη θέση μνήμης πρέπει αρχικά να ξεκινήσουμε μια διαδικασία ψευδο-εγγραφής με την διαφορά ότι ο master μετά την αποστολή της θέσης μνήμης δημιουργεί κατάσταση start και επαναλαμβάνει το Devise_code_select με ένδειξη ανάγνωσης στο τέλος. Στην συνέχεια η μνήμη αποστέλλει στον master το περιεχόμενο της ενεργής θέσης μνήμης. Ο master δεν πρέπει να αποστείλει acknowledgement και τερματίζει την διαδικασία θέτοντας το δίαυλο σε κατάσταση stop.



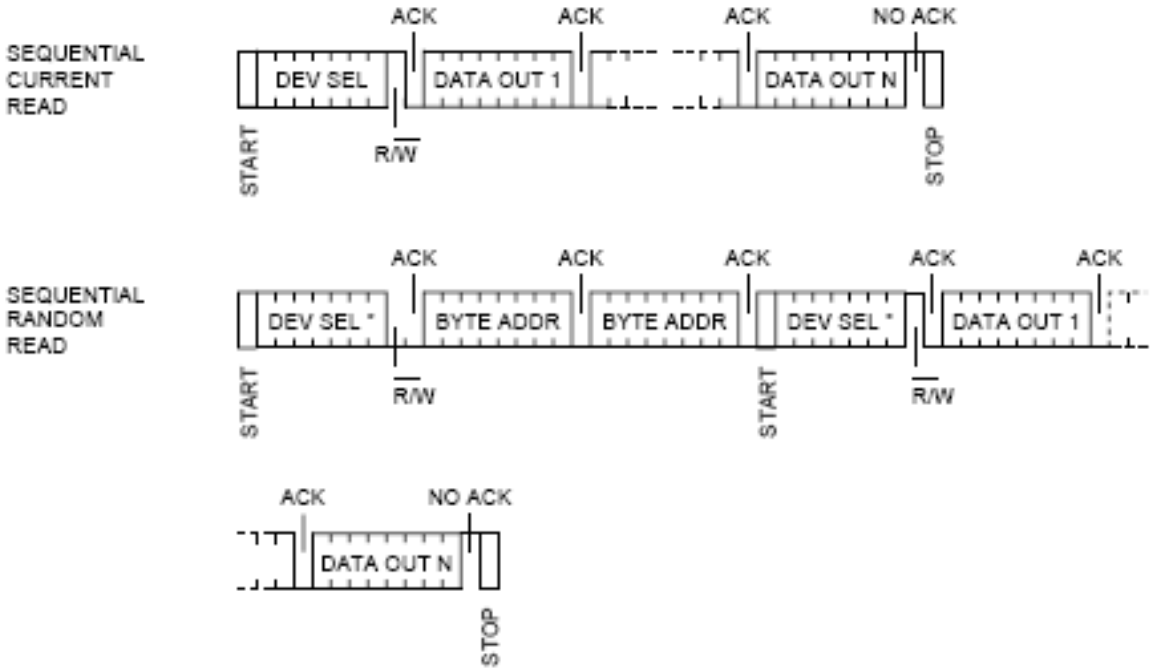
Current Address Read:

Με την διαδικασία Current Address Read γίνεται ανάγνωση της θέσης μνήμης που δείχνει ο εσωτερικός μετρητής της μνήμης. Η διαδικασία είναι πολύ απλή: ο master δημιουργεί κατάσταση start και αποστέλλει το Device_select_code με ένδειξη ανάγνωσης, η μνήμη απαντάει με το περιεχόμενο της θέσης μνήμης που δείχνει ο εσωτερικός μετρητής και ο master δημιουργεί κατάσταση stop στον δίαυλο.

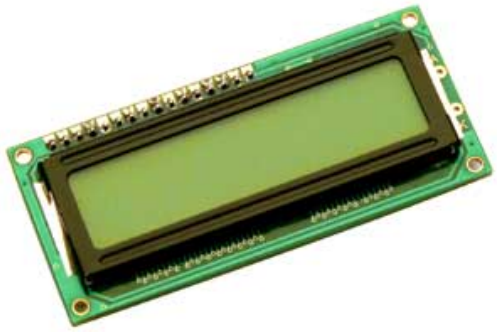


Sequential Read:

Το Sequential Read πρέπει να συνδυαστεί με έναν από τους δύο προηγούμενους τρόπους. Η διαφορά του Sequential Read είναι ότι μετά το τελευταίο byte ο master αποστέλλει acknowledgement και η μνήμη αυξάνει τον εσωτερικό της μετρητή και αποστέλλει το επόμενο byte. Για να τελειώσει η διαδικασία ο master δεν αποστέλλει acknowledgement και δημιουργεί κατάσταση stop στο δίαυλο. Εάν ο μετρητής φτάσει στο τέλος της μνήμης τότε μηδενίζεται και συνεχίζει από την αρχή.



3.4 Lcd οθόνη PC1602-D

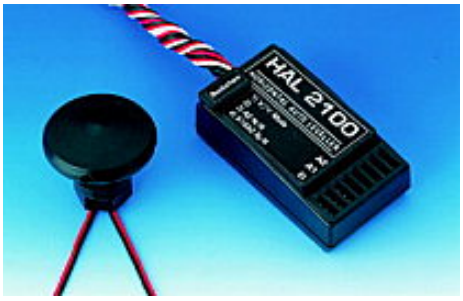


Η οθόνη που χρησιμοποιήθηκε για τον έλεγχο ροής του προγράμματος είναι η PC1602-D. Αποτελείται από 32 στοιχεία, τα οποία είναι χωρισμένα σε δύο γραμμές των 16 χαρακτήρων. Για την επικοινωνία με τον μικροελεγκτή χρειάζεται να συνδεθεί με τα 8 pin μιας θύρας του. Χρησιμοποιήθηκαν οι έτοιμες συναρτήσεις του CodevisionAVR για να τυπωθούν οι επιθυμητές κάθε φορά πληροφορίες.

Οι συναρτήσεις που χρησιμοποιήθηκαν είναι οι εξής:

Lcd_puts(): δέχεται σαν όρισμα έναν δείκτη σε string και εμφανίζει στην οθόνη όλους τους χαρακτήρες που ακολουθούν μέχρι να βρεθεί ο χαρακτήρας '\0'.
Lcd_putsf(): ίδια με την Lcd_puts(), μόνο που το string πρέπει να βρίσκεται στην μνήμη flash του μικροελεγκτή.
Lcd_printf(): η δεσμευμένη εντολή για την C, printf για εμφάνιση τόσο χαρακτήρων, όσο και μεταβλητών.

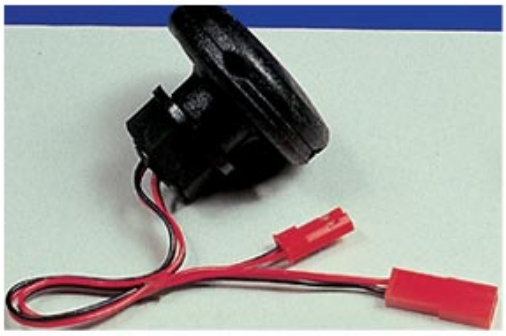
3.5 Το HAL 2100



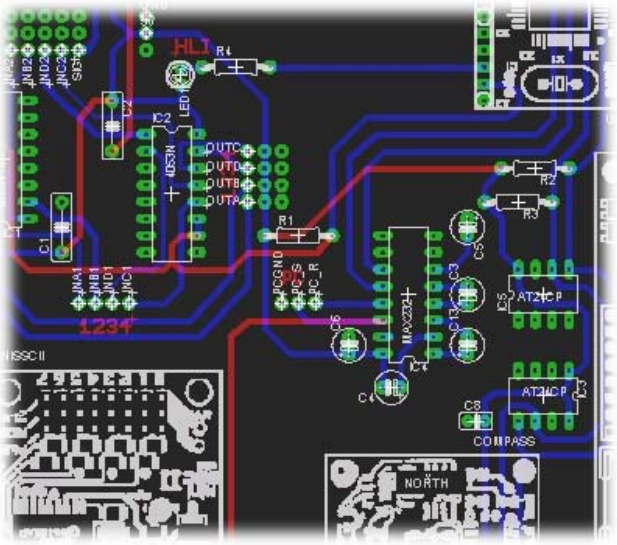
Για το σωστό έλεγχο της πτήσης το σύστημα είναι απαραίτητο να γνωρίζει κάθε στιγμή την κατάσταση στο χώρο στην οποία βρίσκεται. Αυτό μπορεί να επιτευχθεί με αισθητήρες όπως κλισιόμετρο, επιταχυνσιόμετρο ή γυροσκόπιο, τα οποία δίνουν το καθένα διαφορετική πληροφορία με την οποία μπορούμε να συνεπάγουμε εάν το σύστημα βρίσκεται για παράδειγμα υπό κλίση (δηλαδή σε στροφή) ή αν η κλίση αυτή διατηρείται. Αντί αυτών στο σύστημα χρησιμοποιήθηκε μια φτηνότερη κατασκευαστικά λύση από τα παραπάνω.

Το HAL 2100 Horizontal Auto Levelling system είναι ένα καθαρά μοντελιστικό σύστημα το οποίο βοηθά νέους χρήστες στην εκμάθηση πτήσης αερομοντέλων. Το HAL χρησιμοποιεί ένα οπτικό αισθητήρα με τον οποίο μπορεί να παρακολουθεί την συμπεριφορά του αερομοντέλου σε σχέση με τον ορίζοντα. Ο αισθητήρας εγκαθίσταται στο κάτω μέρος του αερομοντέλου και παρακολουθεί την διαφορά της έντασης του φωτός σε τέσσερις κατευθύνσεις στο επίπεδο του μοντέλου. Επίσης συνδέεται μεταξύ δέκτη και servo κινητήρων έτσι ώστε να ενεργεί στη διόρθωση της πτήσης. Αυτό σημαίνει ότι σε περίπτωση που το αερομοντέλο τείνει να χάσει ύψος, ο αισθητήρας ενημερώνει ότι υπάρχει διαφορά στο φως και ενεργεί για να διορθώσει ανάλογα το κατάλληλο servo. Το σύστημα αυτό έχει φτιαχτεί έτσι ώστε να επαναφέρει το αερομοντέλο σε επίπεδο πτήσης και να ενεργεί όταν ο χρήστης αφήσει ελεύθερα τα stick ελέγχου από τον πομπό. Παρέχεται η δυνατότητα να ελέγχεται η ευαισθησία του αισθητήρα από ένα κανάλι στο πομπό.

Στο δικό μας σύστημα το HAL χρησιμοποιείται για την σταθεροποίηση του αερομοντέλου μετά από κάθε στρέψη των επιφανειών ελέγχου. Αυτό μας δίνει τη δυνατότητα να επαναφέρουμε το μοντέλο σε επίπεδο πτήσης προτού ληφθεί η νέα θέση από το GPS και η εκτίμηση της επόμενης κίνησης να είναι σωστή. Επίσης είναι ένας τρόπος να σταθεροποιηθεί το σύστημα σε ένα ύψος για το λόγω του ότι η ένδειξη του υψομέτρου από το GPS δεν είναι τόσο αξιόπιστη.



4 Σχεδιασμός Κυκλώματος



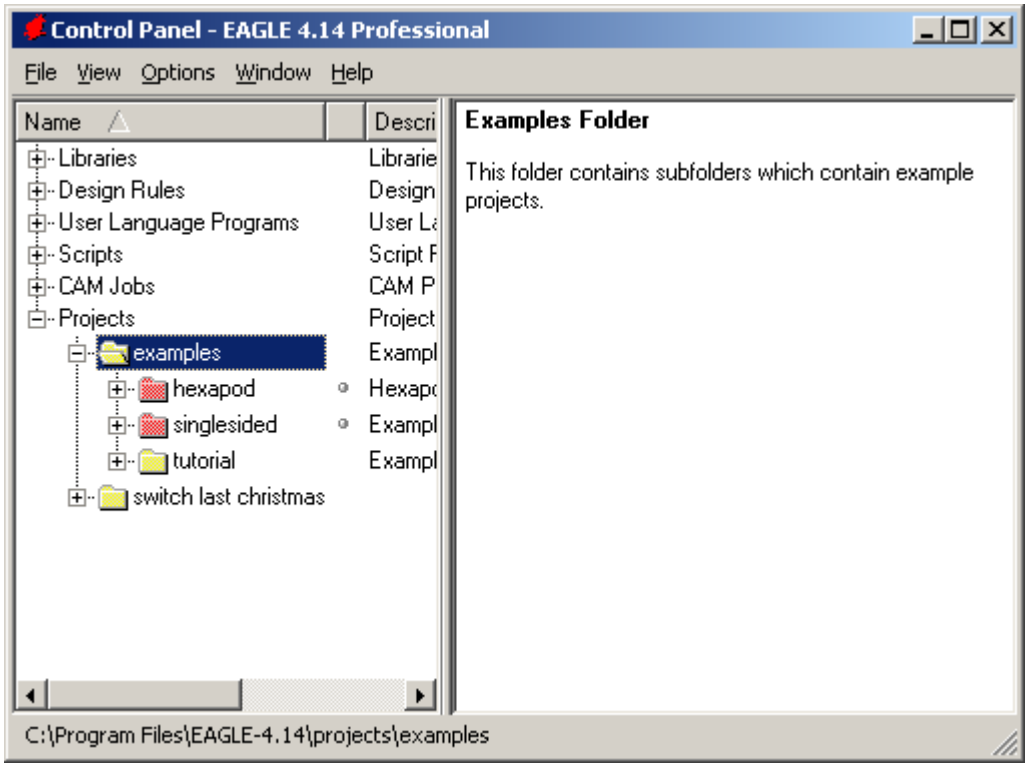
Η υλοποίηση ενός κυκλώματος το οποίο να περιλαμβάνει όλα τα modules που περιγράφηκαν στο προηγούμενο κεφάλαιο ήταν απαραίτητη για την επίτευξη του στόχου.

Ο βασικός τρόπος με τον οποίο υλοποιούνται τα τυπωμένα κυκλώματα είναι περίπου ο ίδιος με εξαίρεση τις εργοστασιακές κατασκευές. Για την κατασκευή χρειάζεται μια πλακέτα επιστρωμένη από χαλκό είτε από την μία μεριά αν πρόκειται να φτιάξουμε πλακέτα μίας πλευράς, είτε κι από τις δύο. Καλύπτοντας για παράδειγμα με μαρκαδόρο τις επιφάνειες όπου πρόκειται να βρίσκονται οι «δρόμοι» και οι «πίστες» τοποθετούμε την πλακέτα σε αποχαλκωτικό οξύ. Το οξύ συνήθως αποτελείται από τριχλωριούχο σίδηρο το οποίο χρειάζεται αραίωση. Αυτό με τη σειρά του διαβρώνει τις ελεύθερες επιφάνειες χαλκού αφήνοντας πίσω μόνο ότι χρειάζεται. Μετά από τον καθαρισμό της η πλακέτα είναι έτοιμη να τρυπηθεί και να δεχθεί τα εξαρτήματά.

Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε φωτοευαίσθητη πλακέτα. Αυτή η πλακέτα είναι μια πλακέτα επίστρωσης χαλκού που από πάνω έχει μία ακόμη στρώση η οποία μόλις εκτεθεί στο φως καταστρέφεται (φιλμ). Εκτυπώθηκε το σχέδιο του κυκλώματος για δύο πλευρές σε διαφάνειες που στη συνέχεια τις ευθυγραμμίστηκαν στην πάνω και κάτω πλευρά της φωτοευαίσθητης πλακέτας. Τοποθετήθηκε κάθε πλευρά της πλακέτας με τη διαφάνεια σε λάμπα υπεριώδους ακτινοβολίας (κλειστό θάλαμο) για περίπου τρία λεπτά κάθε μία. Στη συνέχεια και πολύ γρήγορα εφόσον το φιλμ εκτέθηκε στο φως βυθίστηκε η πλακέτα σε διάλυμα νιτρικής σόδας με νερό. Με αυτό τον τρόπο εμφανίστηκαν οι δρόμοι και οι πίστες πάνω στη πλακέτα αφού διαλύθηκε το καμένο φιλμ από τα ανεπιθύμητα σημεία. Στη συνέχεια, σειρά είχε το αποχαλκωτικό οξύ το οποίο απέδωσε και την τελική μορφή της πλακέτας.

Τον τρόπο με τον οποίο έγινε ο σχεδιασμός πριν την υλοποίηση του τελικού κυκλώματος θα τον δούμε παρακάτω.

4.1 Το σχεδιαστικό πρόγραμμα Eagle

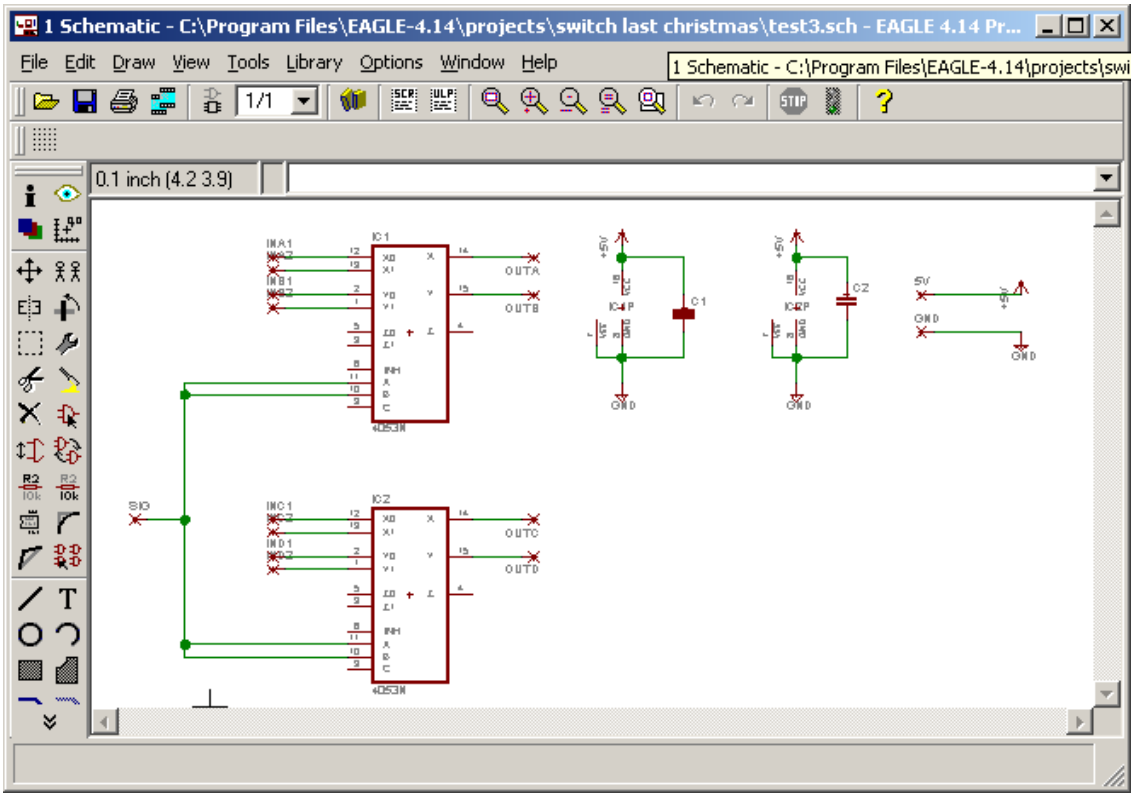


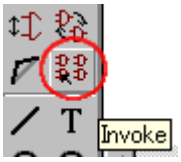
Το Eagle της εταιρίας Cadsoft είναι ένα ιδανικό εργαλείο για την σχεδίαση του τυπωμένου κυκλώματος. Το πλεονέκτημα του προγράμματος είναι κυρίως το ότι είναι freeware και επίσης το ότι αποδείχτηκε αρκετά εύκολο στη χρήση του, χωρίς όμως να το καθιστά κορυφαίο στη κατηγορία του. Όπως και τα άλλα προγράμματα αυτού του είδους πριν την υλοποίηση του board όπως λέγεται, χρειάζεται να δημιουργηθεί το schematic.

Ανοίγοντας το Eagle μπορούμε να δούμε κάποια παραδείγματα κάτω από το πεδίο Projects>examples. Κάνοντας δεξί click στο Projects μπορούμε να δημιουργήσουμε ένα δικό μας project και ένα νέο Schematic.

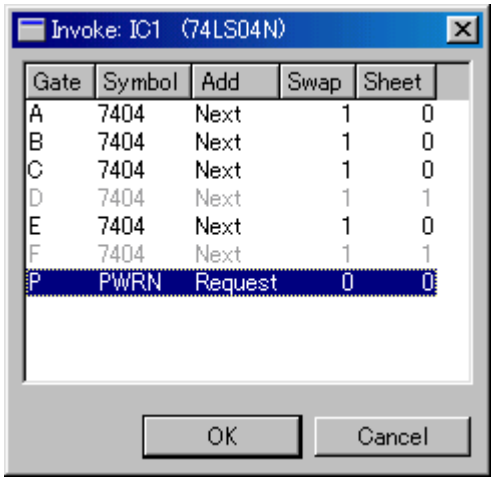
Πατώντας το κουμπί ADD βρισκόμαστε σε μια λίστα από τύπους εξαρτημάτων τα οποία μπορούν να τα προστεθούν στο σχεδιαστικό μας. Τα εξαρτήματα αυτά είναι αποθηκευμένα σε βιβλιοθήκες με όνομα για παράδειγμα «40xx.lbr» και περιέχουν πληροφορίες σχετικά με το σχήμα του εξαρτήματος στο schematic αλλά και στο board όπως
μέγεθος, διάμετρος και πάχος των pads και άλλα.

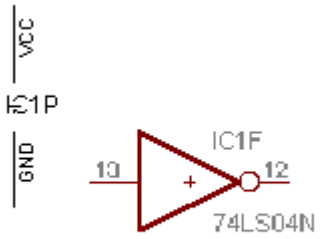
Τοποθετώντας όλα τα εξαρτήματα που χρειάζονται μπορούμε να τα ενώσουμε και να δώσουμε κάποιες τιμές όπως η τιμή ή το όνομα σε περίπτωση που έχουμε πολλά εξαρτήματα του ίδιου είδους.

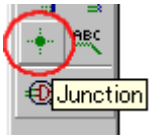




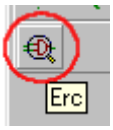
Για να δηλωθεί η τροφοδοσία των εξαρτημάτων επιλέγουμε το κουμπί invoke όπου εμφανίζεται ένα μενού με τους διαθέσιμους ακροδέκτες. Επιλέγοντας PWRN request και πατώντας OK εμφανίζεται στο schematic ένα σχήμα με τους ακροδέκτες της τροφοδοσίας του εξαρτήματος.



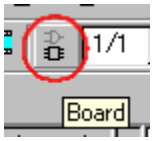




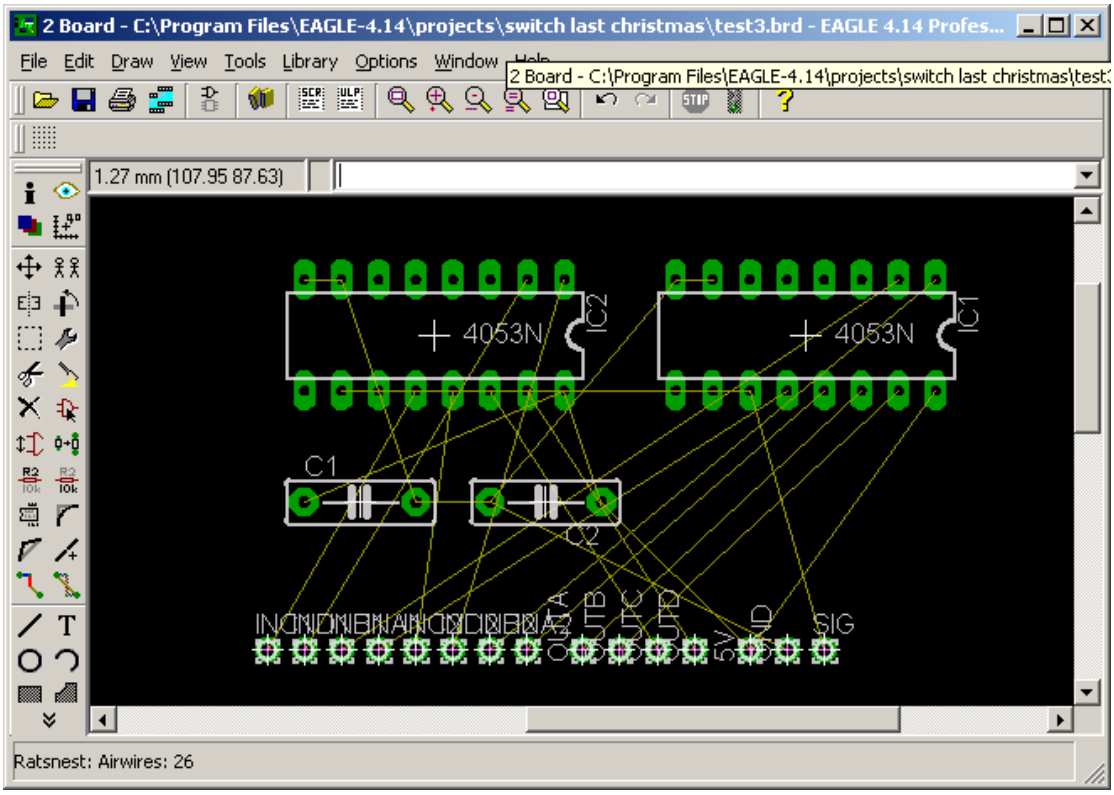
Κατά την ένωση των εξαρτημάτων το πρόγραμμα προειδοποιεί για την αποδοχή πολλών ενώσεων ενώ μεγάλη προσοχή χρειάζεται κατά την ένωση κόμβων όπου χρειάζεται να εισάγουμε το ενδεικτικό junction.



Αφού ολοκληρωθεί ο σχεδιασμός του κυκλώματος, αυτό θα πρέπει να ελεγχθεί για τυχών λάθη η μη συνδεδεμένα εξαρτήματα στο κύκλωμα. Αυτό επιτυγχάνετε με την επιλογή ERC (Electrical Rule Check) από τον πίνακα των εργαλείων.

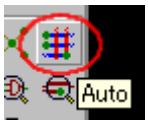


Κατόπιν το σχέδιο μεταφέρεται στο κομμάτι του προγράμματος για τον σχεδιασμό της πλακέτας πατώντας το κουμπί Board. Εδώ θα ανοίξει αυτόματα το αντίστοιχο πρόγραμμα φέρνοντας τα εξαρτήματα και τις συνδέσεις από το προηγούμενο.



Σε αυτήν την φάση θα πρέπει να τοποθετηθούν τα υλικά όπως εμείς θέλουμε, αλλά θα πρέπει να προσέξουμε και την ευκολότερη διαδρομή για τις πίστες που θα δημιουργηθούν αργότερα.

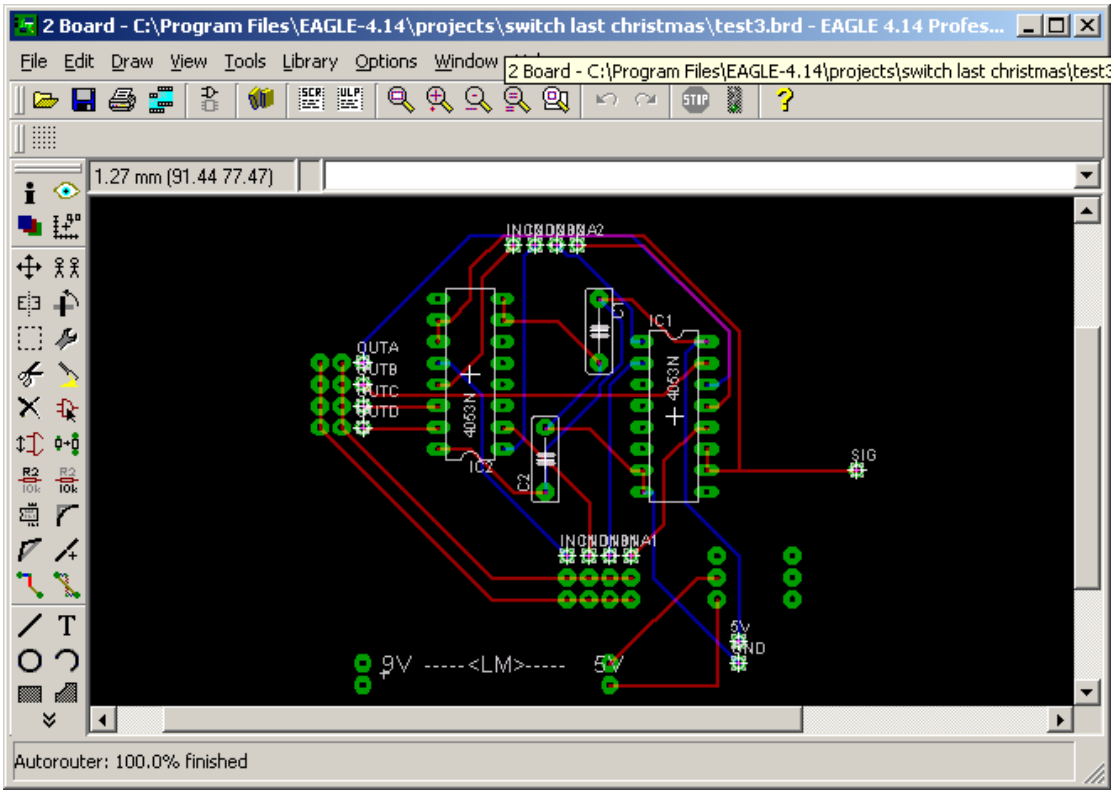
Αφού ολοκληρωθεί η τοποθέτηση των υλικών χρησιμοποιούμε το πλήκτρο Rastnet από την μπάρα εργαλείων ώστε το πρόγραμμα να ξανά δρομολογήσει τις δικτυώσεις και να είναι όσο το δυνατών πιο σύντομες.



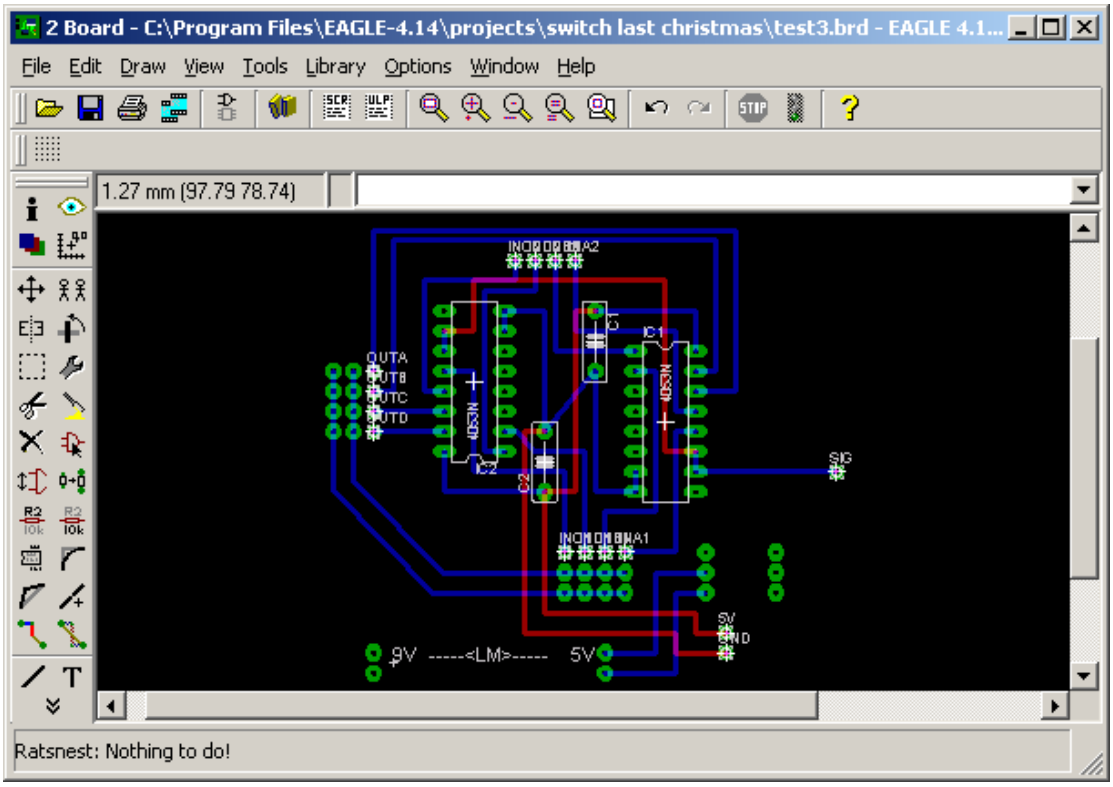
Τέλος πατάμε το πλήκτρο Auto επιλέγοντας τα Layer που επιθυμούμε και πατώντας το πλήκτρο OK δημιουργούνται οι πίστες. Ρυθμίσεις του Autorouter 1 TOP [*] - 16 Bottom [*] τα υπόλοιπα όπως έχουν.

Σε περίπτωση που χρειαζόμαστε επιπλέον πίστες στις οποίες σκοπεύουμε να ενώσουμε εξαρτήματα τα που δεν περιλαμβάνονται στο σχέδιο μπορούμε να τα προσθέσουμε και σε αυτό το στάδιο.

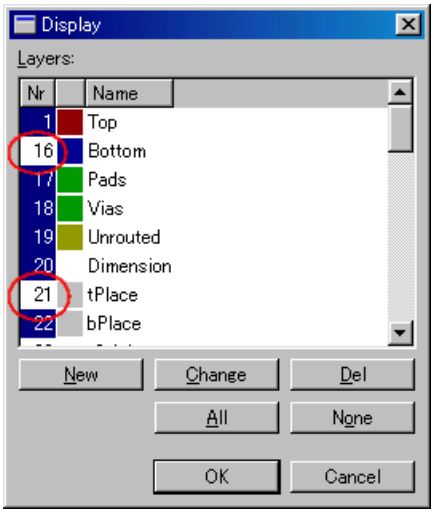
Το αποτέλεσμα του autorouter θα είναι η παρακάτω εικόνα όπου φαίνεται να έχουν σχηματιστεί οι δρόμοι και οι πίστες τόσο στο επάνω με κόκκινο χρώμα επίπεδο όσο και στο κάτω με μπλε.

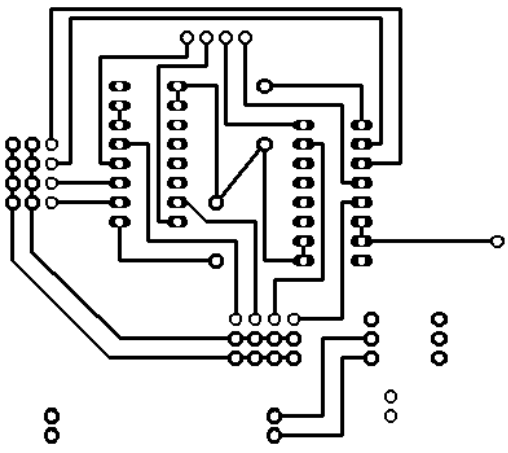


Είναι προφανές πως με κατάλληλες επεμβάσεις στο board μπορούμε να κάνουμε συντομότερους δρόμους και γενικά να το φέρουμε στα μέτρα μας για λόγους λειτουργικότητας αλλά και οικονομίας. Με την εντολή ripup; στο πεδίο εντολών επαναφέρουμε τις συνδέσεις του autorouter. Ένας καλός σχεδιασμός του παραπάνω κυκλώματος που αναπαριστά την εναλλαγή του χειροκίνητου από του ρομποτικού ελέγχου του συστήματός μας είναι η παρακάτω.



Σε αυτό το σημείο μπορούμε να επιλέξουμε από την επιλογή Display ποια επίπεδα θέλουμε να εμφανίζονται για να εκτυπωθούν σε διαφάνεια το σχέδιο. Θα πρέπει να αποεπιλέξουμε το layer tPlace το οποίο αποκρύπτει τα αντικείμενα μαζί με τα ονόματα και τις τιμές και αφήνει ορατές μόνο τις πίστες αυτών. Ανάλογα αποκρύπτουμε όποιο layer χρειάζεται για να εκτυπώσουμε την κατάλληλη πλευρά. Κάνοντας ρυθμίσεις στο print setup ή στο export μπορούμε όπως το Monochrome μπορούμε να πάρουμε το επιθυμητό αποτέλεσμα και να εκτυπώσουμε το κύκλωμά μας με ακρίβεια.

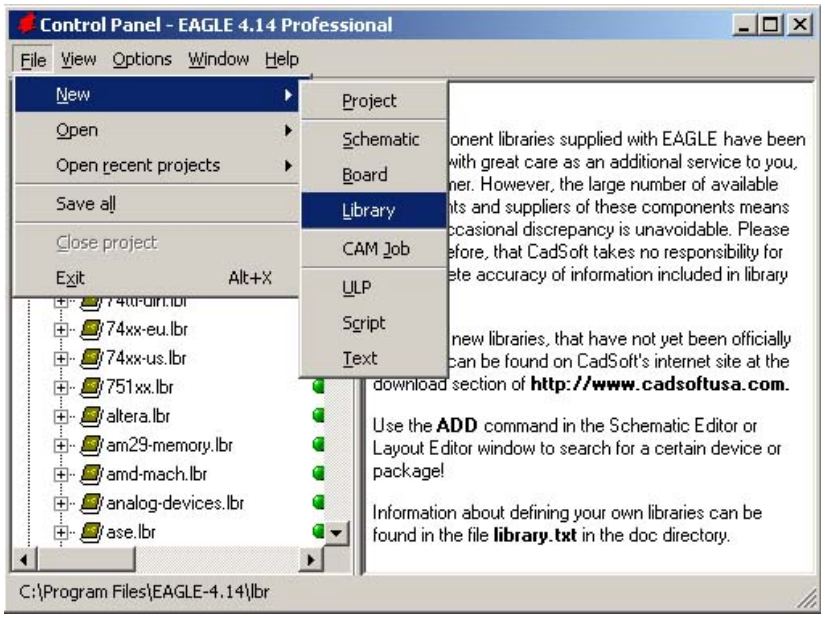




4.2 Κατασκευή βιβλιοθηκών στο Eagle

Κατά τη σχεδίαση του κυκλώματος χρειάζεται να εισάγουμε στο schematic τα απαραίτητα υλικά όπως αντιστάσεις, πυκνωτές, led που υπάρχουν στις βιβλιοθήκες του Eagle. Τα επιμέρους module όπως ο microcontroller της Atmel, το GPS, η πυξίδα και ο servo controller δεν υπήρχαν. Σε μια προσπάθεια τυχαίας τοποθέτησης σύμφωνα με τις διαστάσεις τους, παρατηρήθηκε σοβαρό πρόβλημα σε σχέση με την ευθυγράμμιση των ακίδων κάθε module. Γι' αυτό το λόγο κατασκευάστηκε καινούργια βιβλιοθήκη στο Eagle προσθέτοντας κάθε ηλεκτρονικό εξάρτημα που θα χρειαζόταν.

Από το Control Panel του Eagle επιλέγουμε New Library

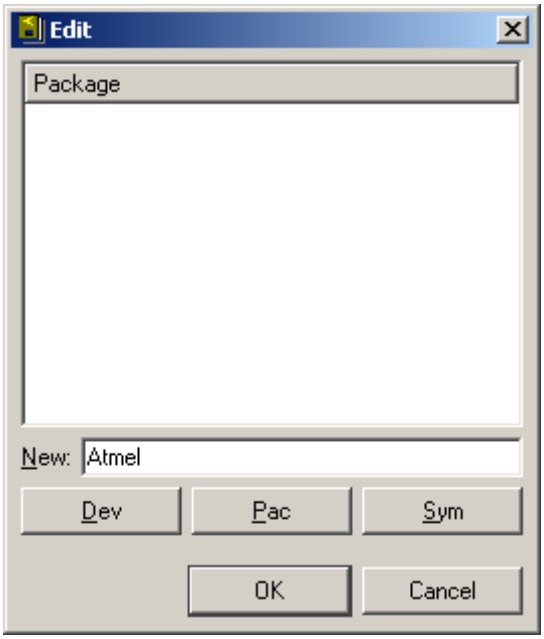


Πατώντας Edit μπορούμε να δημιουργήσουμε το δικό μας εξάρτημα. Κάθε εξάρτημα αποτελείτε από τρία στοιχεία:

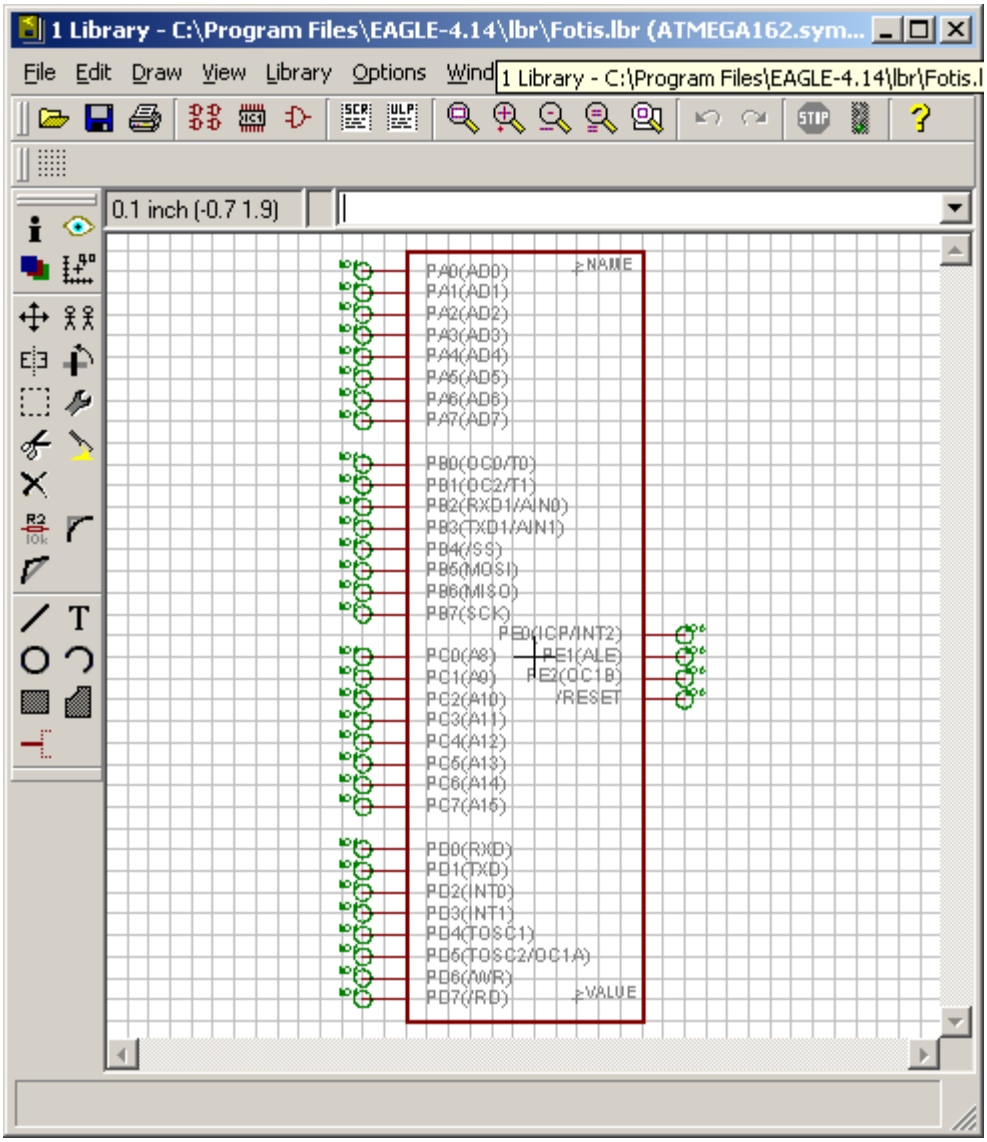
Το Package το οποίο είναι το σχέδιο που θα φαίνεται κατά την επεξεργασία του στο Board

Το Symbol το οποίο είναι το σχέδιο που θα φαίνεται κατά την επεξεργασία του στο Schematic

Το Device που είναι η συνένωση των δύο παραπάνω και θα φαίνεται σαν εξάρτημα μέσα στη βιβλιοθήκη μας

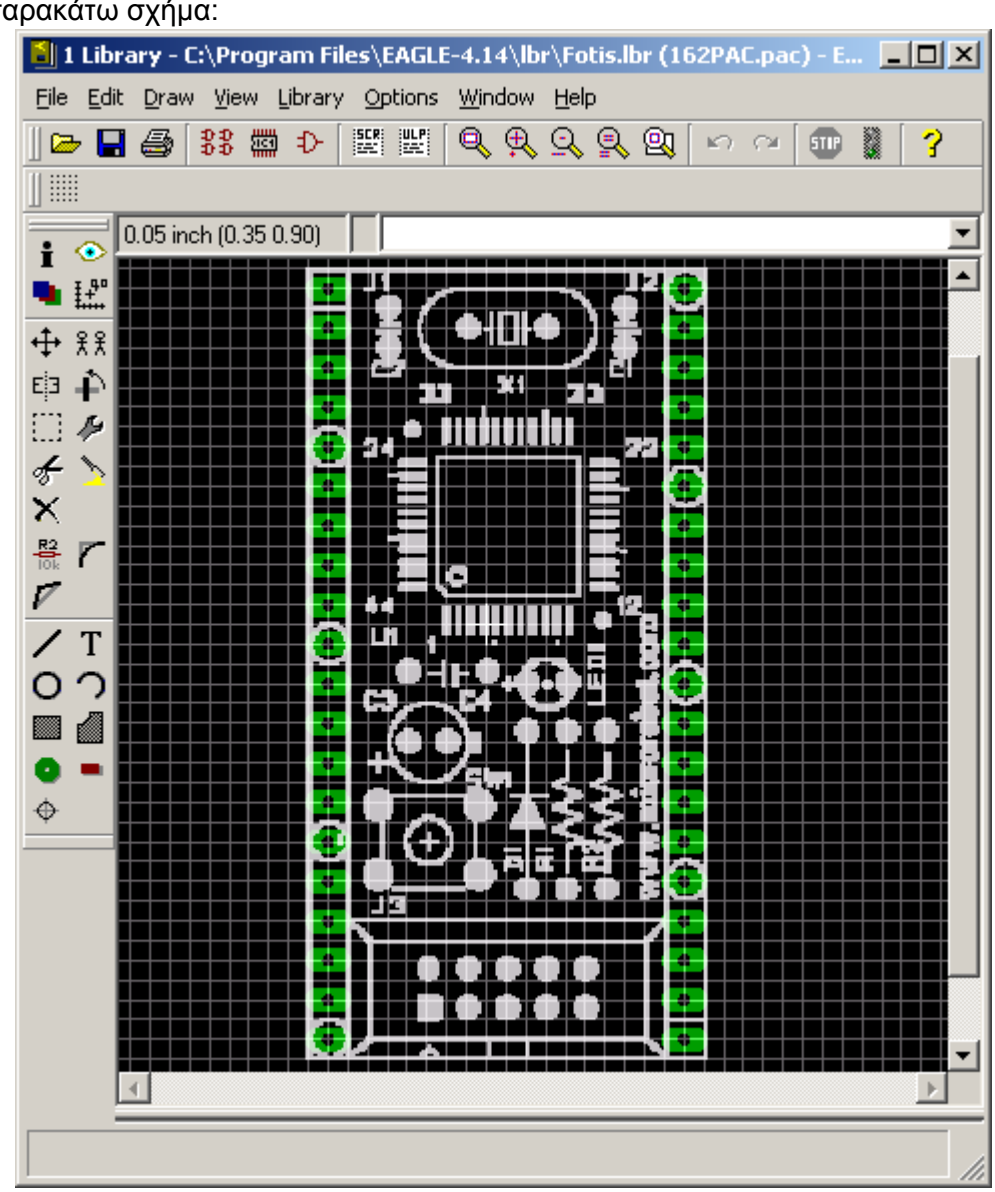


Για παράδειγμα στο αντικείμενο του MR-162 δημιουργήθηκε πρώτα ένα Symbol πατώντας στο edit ένα όνομα, το κουμπί Sym και OK. Αφού σχεδιάστηκε ένα περίγραμμα τοποθετήθηκαν οι ακροδέκτες με το κουμπί Pin.



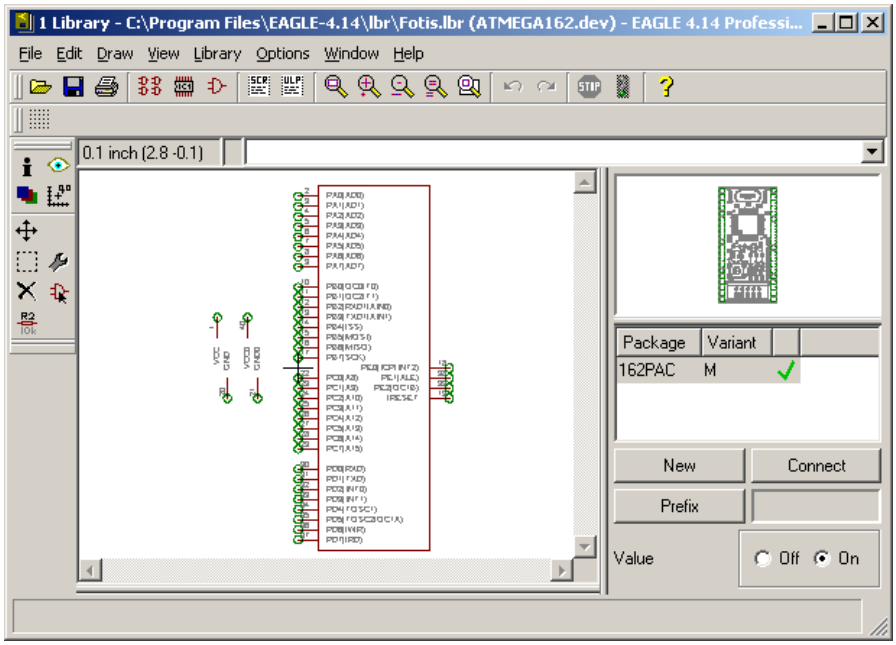
Στη συνέχεια και αφού σώζουμε κάθε φορά τη βιβλιοθήκη επιλέγουμε Package όπου σχεδιάζουμε σε πραγματικό μέγεθος το εξάρτημά. Εδώ θα πρέπει να προσέξουμε να ρυθμίσουμε σωστά το Grid έτσι ώστε να τοποθετηθούν σε σωστή απόσταση οι πίστες του MR-162. Επίσης προσοχή θέλει η ακριβής μέτρηση των ακίδων με παχύμετρο.

Για την κατασκευή του πλέγματος θα μπορούσε να σχεδιαστεί ένα περίγραμμα και να δηλωθούν οι άκρες της συσκευής αλλά προτιμήθηκε ένας άλλος τρόπος. Στο Eagle έχουμε τη δυνατότητα να εισάγουμε κάποια script τα οποία κάνουν κάποιες διεργασίες αυτόματα. Με ένα script το οποίο μετατρέπει μια εικόνα bmp σε σχήμα αποτελούμενο από πολλές γραμμές μετατρέψαμε εικόνες των εξαρτημάτων σε σχήματα αληθινών διαστάσεων στο Package του κάθε ενός. Επίσης χρειάζεται προσοχή έτσι ώστε τα αντικείμενα να αναπαρίστανται με ακρίβεια, ειδικά στα σημεία όπου συμβολίζονται οι τρύπες. Μετά από αρκετές δοκιμές το αποτέλεσμα του Package του MR-162 φαίνεται στο παρακάτω σχήμα:

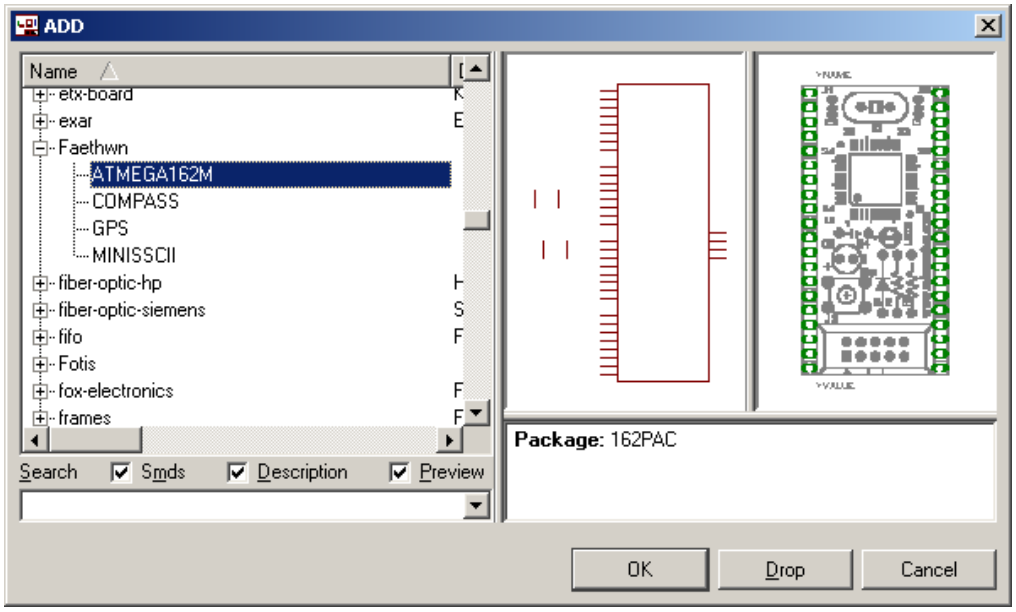


Σώζοντας και πάλι επιλέγουμε το κουμπί Device. Σε αυτό το σημείο ενώνουμε τα δύο σχέδια σε ένα εξάρτημα. Εδώ αντιστοιχούμε κάθε Pin του Board με το κατάλληλο Pin στο Schematic. Πριν γίνει αυτό θα πρέπει να έχει φτιαχτεί και ένα σύμβολο τροφοδοσίας έτσι ώστε να εισαχθεί κι αυτό. Πατάμε ADD και εισάγουμε τα απαραίτητα σύμβολα και

στα δεξιά μας το NEW για να εισάγουμε και το επιθυμητό Package. Πατώντας το connect μπορούμε να ενώσουμε τα κάθε Pin. Ένα εξάρτημα χαρακτηρίζεται από το Device όχι από το Package ή το Symbol και με αυτό τον τρόπο μπορούμε να φτιάξουμε και άλλα εξαρτήματα με συνδυασμούς σαν Instances.



Σώζοντας την βιβλιοθήκη είμαστε έτοιμοι να χρησιμοποιήσουμε το εξάρτημα στο Schematic. Αντιστοίχως με τον ίδιο τρόπο έγινε και η κατασκευή των υπολοίπων συσκευών που έλαβαν μέρος στο κύκλωμα. Η βιβλιοθήκη που περιέχει όλες τις συσκευές είναι διαθέσιμη για χρήση στο CD με όνομα “Faethwn.lbr”.

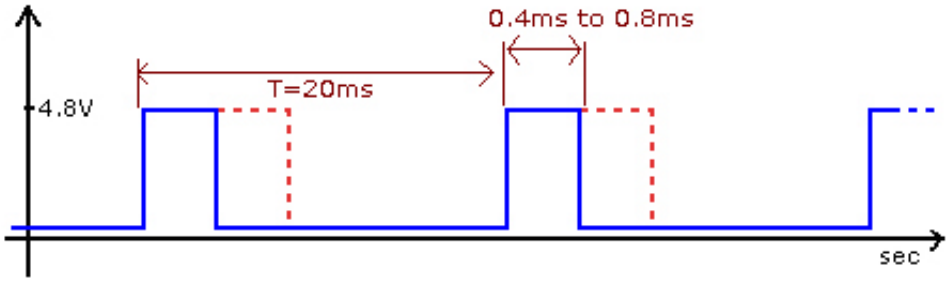


4.3 Το κύκλωμα εναλλαγής AUTO-MANUAL

Η αρχική ιδέα της πτυχιακής εργασίας ήταν το αερομοντέλο να έχει μια χειροκίνητη προσγείωση και απογείωση. Λόγω της πολυπλοκότητας και της δυσκολίας αυτών των ενεργειών σωστό είναι να γίνουν με τη βοήθεια ενός έμπειρου αερομοντελιστή. Ο ρομποτικός έλεγχος θα άρχιζε από τη στιγμή που το αερομοντέλο έχει αποκτήσει κάποιο ύψος για να ακολουθήσει μια προκαθορισμένη πορεία. Επομένως, πρέπει να υπάρχει ένας τρόπος με τον οποίο να μεταβάλλουμε την κατάσταση του κυκλώματος από ρομποτική λειτουργία σε χειροκίνητη. Αυτό χρειάζεται ακόμα και στην περίπτωση των δοκιμών, ώστε αν κάτι πάει λάθος στον αλγόριθμο ελέγχου να είμαστε σε θέση να επαναφέρουμε το αερομοντέλο στα «χέρια μας»!

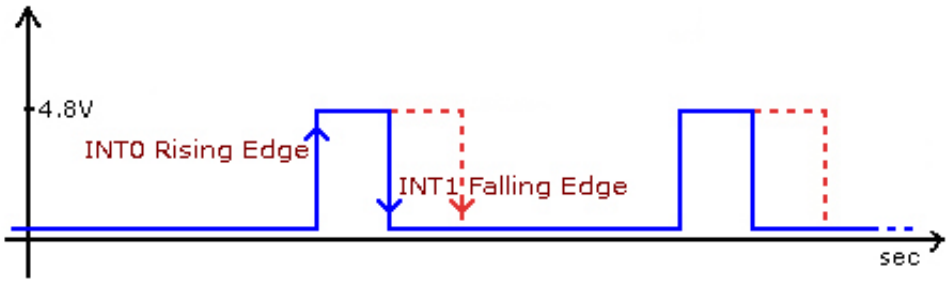
Η μοναδική επικοινωνία με το κύκλωμα κατά τη πτήση είναι η τηλεκατεύθυνση. Για αυτό το λόγο επιλέξαμε τηλεκατεύθυνση άνω των τεσσάρων καναλιών. Τα τέσσερα βασικά κανάλια για την πλοήγηση και ένα πέμπτο για την εναλλαγή. Το 5^{ο} κανάλι στη τηλεκατεύθυνσή προοριζόταν για servo το οποίο θα εκτελούσε μια επιπλέον λειτουργία όπως ο έλεγχος κινητού συστήματος προσγείωσης. Ο έλεγχος αυτός γίνεται μέσω ενός διακόπτη στον πομπό.

Κατά τη μέτρησή του 5^{ου} καναλιού σε παλμογράφο μετρήθηκε παλμός 50Hz με Duty Cycle να μεταβάλλεται από 2 σε 4% ανάλογα με την αλλαγή του διακόπτη.



παλμός ch5 εκτός κλίμακας

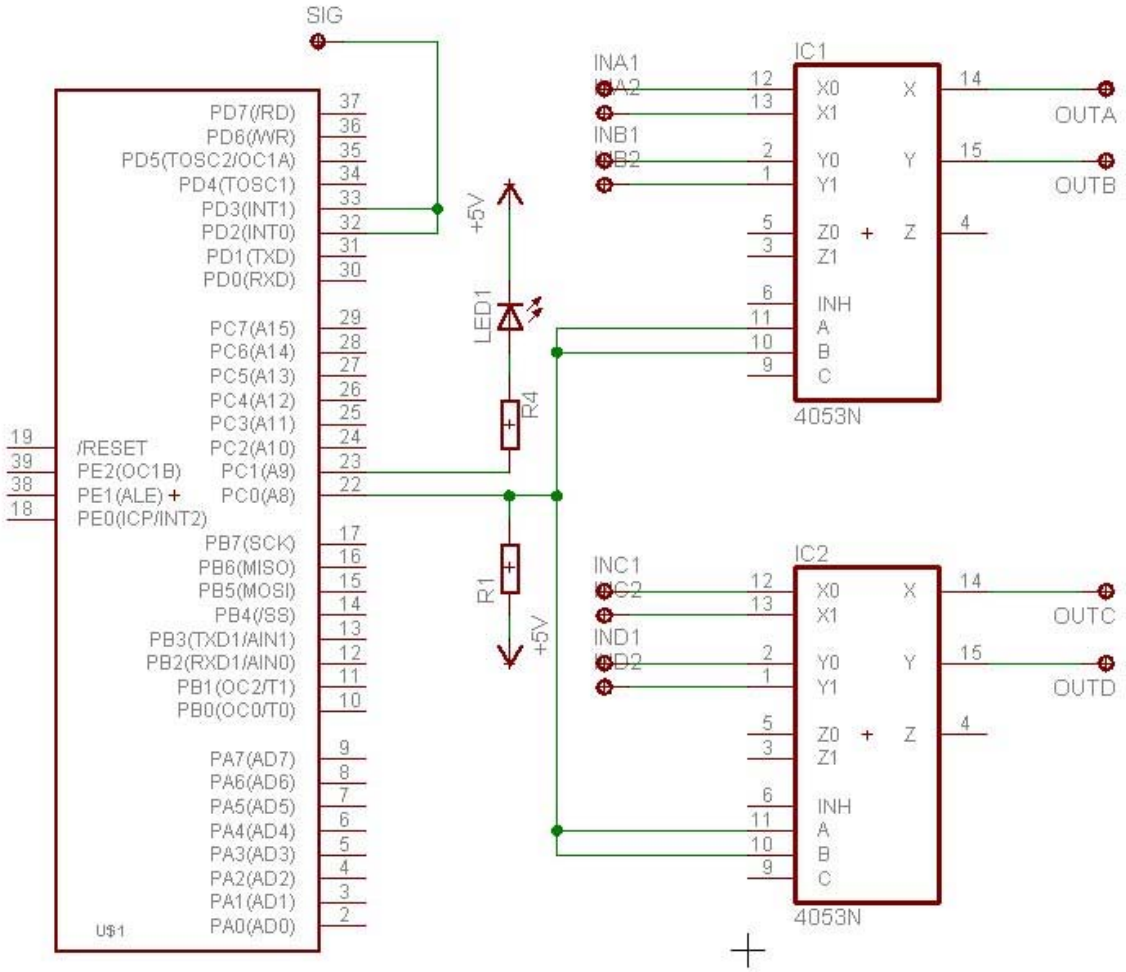
Ο παλμός αναγνωρίστηκε από τον microcontroller με τη βοήθεια των εξωτερικών interrupt και ενός timer. Βραχυκυκλώνοντας τα δύο interrupts INT0 και INT1 τα ενεργοποιήσαμε για Rising και Falling edge αντίστοιχα. Με τον τρόπο αυτό κάθε φορά που ο παλμός αλλάζει από Low σε High εκτελείτε η ρουτίνα του interrupt0 ενώ όταν αλλάζει αντίθετα εκτελείτε η ρουτίνα του interrupt1.

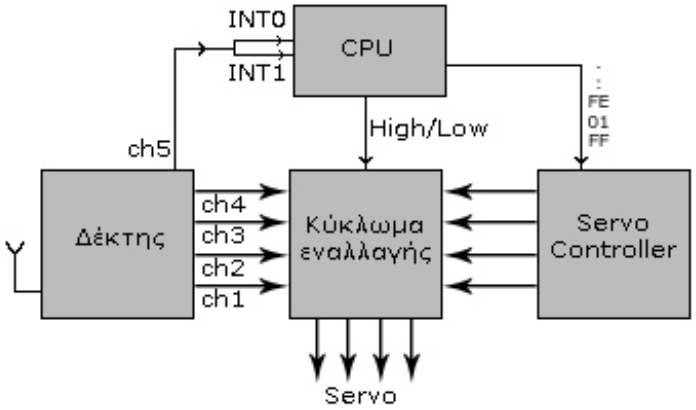


Όταν εκτελείτε το interrupt0 ξεκινάμε τον 8bit Timer0. Η λειτουργία του Timer γενικά μπορεί να μας μετρά χρόνο παίρνοντας clock από τον κρύσταλλο του μικροελεγκτή, ή και μικρότερο clock (prescaled CK). Μετά από 256 βήματα αύξησης γεμίζει και λέμε ότι υπερχειλίζει. Τη στιγμή αυτή μπορεί να δώσει ένα interrupt στον μικροελεγκτή οπότε διαπιστώνουμε ότι έχει περάσει το υπολογισμένο χρονικό διάστημα. Στην δικιά μας περίπτωση ρυθμίσαμε τον Timer με prescale στα 2KHz (CK/8). Επομένως μετρώντας το πόσες φορές υπερχείλισε ο Timer είμαστε σε θέση να αναγνωρίσουμε σε ποιο Duty Cycle βρίσκεται ο παλμός.

Το κύκλωμα εναλλαγής στηρίχθηκε στο ολοκληρωμένο κύκλωμα CD4053B το οποίο είναι τύπος αναλογικού διακόπτη. Κάθε ολοκληρωμένο περιέχει τέσσερις διακόπτες οι οποίοι ενεργοποιούνται από συγκεκριμένη είσοδο ο καθένας. Επειδή απαιτούνταν τέσσερα servo, χρησιμοποιήθηκαν δύο 4053 σε συνδυασμό ρυθμίζοντάς τους για μεταγωγή του κάθε παλμού.

Schematic κυκλώματος εναλλαγής





Στην είσοδο SIG είναι η είσοδος του ch5 παλμού και από το Pin1 της πόρτας C ενεργοποιούμε τις εισόδους του κάθε διακόπτη. Για την περίπτωση όπου ο μικροελεγκτής για κάποιο λόγο σταματήσει να αποκρίνεται έχει προστεθεί μια αντίσταση (R1) στη τροφοδοσία έτσι ώστε να παρέχει στους αναλογικούς διακόπτες High. Η High κατάσταση φέρνει τον έλεγχο προκαθορισμένα στο δέκτη. Επίσης από το Pin2 ενεργοποιούμε ταυτόχρονα και ένα Led για δική μας διευκόλυνση.

Το πρόγραμμα που προστέθηκε στον μικροελεγκτή για αυτό το σκοπό είναι το παρακάτω:

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)          //interrupt όταν έχουμε Rising Edge
{
i=0;
TCCR0=0x02;                                           //Έναρξη του Timer0 με prescale CK/8
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)          //interrupt όταν έχουμε Falling Edge
{
TCCR0=0x00;                                           //Κλείσιμο του Timer0
        if (i>=10){                                   //Αν τα overflow είναι πάνω από 10
                PORTC=0x03;                           //High τα 2 πρώτα pin της PORTC
        }
        else{                                         //Αλλιώς
                PORTC=0x00;                           //Low τα 2 πρώτα pin της PORTC
}
}

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)        //interrupt όταν ο Timer υπερχειλίσει
{
i++;                                                  //αύξησε τα overflow
}
```

4.4 Το κύκλωμα

Αφού κατασκευάστηκαν όλες τις απαραίτητες βιβλιοθήκες για όλα τα εξαρτήματα ακολούθησε η κατασκευή του ηλεκτρονικού σχεδίου στο Eagle.

Η πόρτα A δεσμεύτηκε εξολοκλήρου από μια LCD οθόνη 2χ16. Η οθόνη τοποθετήθηκε στο πίσω κεντρικό μέρος του αερομοντέλου με σκοπό την εμφάνιση διάφορων πληροφοριών όπως η έναρξη του προγράμματος, η αρχικοποίηση και την εμφάνιση των στιγμάτων από το GPS.

Το i2c bus υλοποιήθηκε στα δύο πρώτα pin της πόρτας B. Επειδή επιλέξαμε τρόπο μετάδοσης PWM συνδέσαμε τα SCL και SDA με αντιστάσεις των 47ΚΩ στη τάση. Πάνω στο i2c συνδέθηκαν σε σειρά οι δύο μνήμες EEPROM και η ηλεκτρονική πυξίδα.

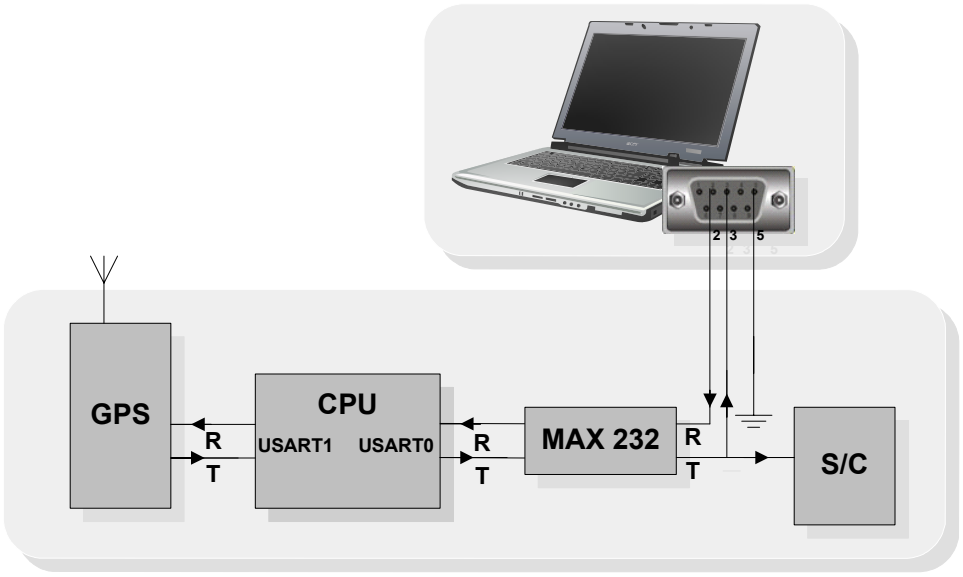
Τα δύο πρώτα pin της πόρτας C δεσμεύτηκαν για το κύκλωμα εναλλαγής ενώ το σήμα του 5^{ου} καναλιού στα pin 3 και 4 της πόρτας D βραχυκυκλωμένα όπου οδηγούν στα εξωτερικά interrupt του επεξεργαστή.

Το GPS καθώς και ο Servo Controller συνδέθηκαν στις δύο σειριακές θύρες του επεξεργαστή.

Το GPS μπορεί να λειτουργήσει σε επίπεδα τάσης σύμφωνα με τον επεξεργαστή και για αυτό το λόγο συνδέθηκε απευθείας σε αυτόν. Ο Servo Controller από κατασκευής του λειτουργεί σε επίπεδα τάσης 10V -10V, δηλαδή σύμφωνα με τη σειριακή RS-232 που διαθέτουν οι ηλεκτρονικοί υπολογιστές. Για αυτό το λόγο κάνουμε χρήση drivers όπως το MAX-232 το οποίο επιτρέπει την επικοινωνία των διαφορετικών μεταξύ τους σημάτων.

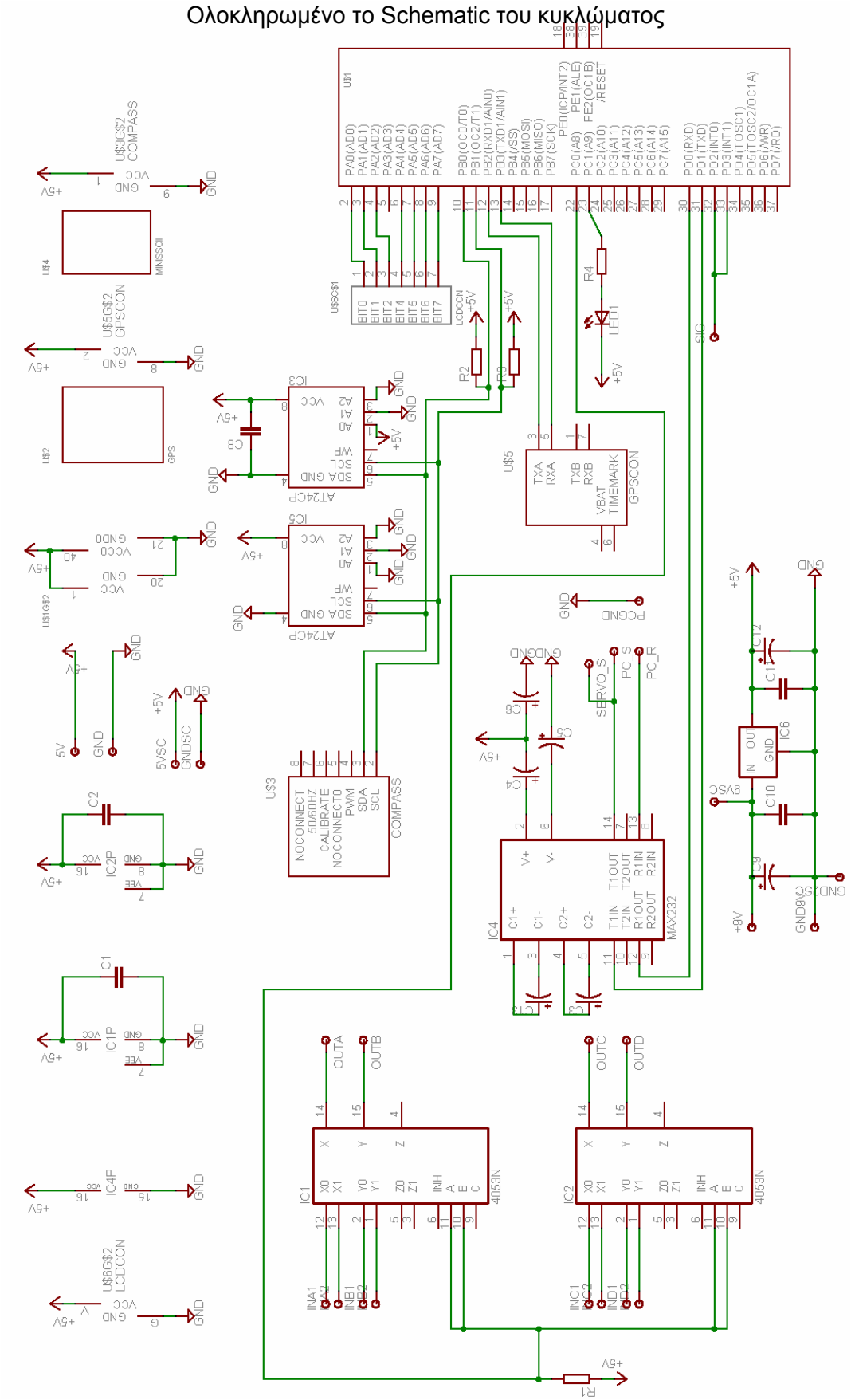
Στο κύκλωμα επιθυμούσαμε να προσθέσουμε μια σειριακή σύνδεση με υπολογιστή έτσι ώστε να μπορούμε να παίρνουμε πληροφορίες από τον επεξεργαστή αλλά και τις αποθηκευμένες πληροφορίες από τη μνήμη. Ο επεξεργαστής όμως περιοριζόταν μόνο σε δύο σειριακές συνδέσεις. Σε αυτό το σημείο δημιουργήθηκε ένα βραχυκύκλωμα στο Transmit της σειριακή του Servo Controller. Ο S/C έχει ως είσοδο μόνο το Transmit καλώδιο από την σειριακή, για να παίρνει τα δεδομένα για τις κινήσεις των Servo. Αυτό σημαίνει ότι αν βραχυκυκλωθεί με μια είσοδο προς τον υπολογιστή και συνδεθεί με αυτόν θα δεχόμαστε τα δεδομένα χωρίς να επηρεάζουμε τη λειτουργία του κυκλώματος. Με αυτό τον τρόπο μπορούμε κάθε φορά που συνδέουμε τον υπολογιστή να στέλνουμε ένα «συνθηματικό» από το Receive καλώδιο της σειριακής, το οποίο δεν χρησιμοποιείτε, προς τον επεξεργαστή. Στη συνέχεια θα αναγνωρίζεται το «συνθηματικό» αυτό από το Receiver Interrupt της σειριακής και από το πρόγραμμα του επεξεργαστή με σκοπό αφενός την αποκοπή επικοινωνίας με τον S/C και αφετέρου την έναρξη επικοινωνίας επεξεργαστή – υπολογιστή. Ο έλεγχος αυτής της επικοινωνίας από τη μεριά του

υπολογιστή υλοποιείται μέσα από διεπαφή προγραμματισμένη σε Visual Basic κάτι το οποίο θα αναλύσουμε παρακάτω.



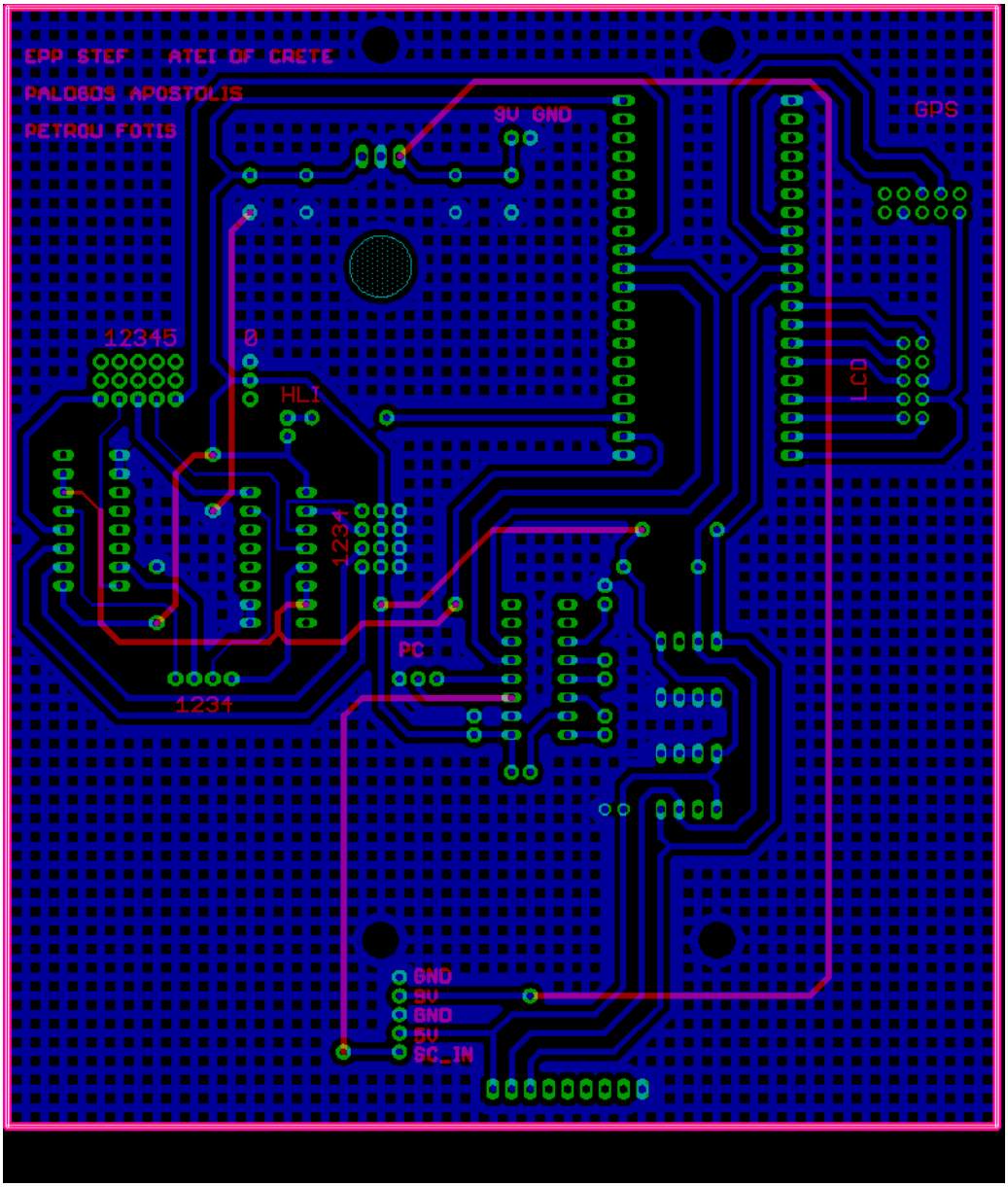
Όλα τα εξαρτήματα του κυκλώματος απαιτούν τροφοδοσία 5V εκτός από το Servo Controller που απαιτεί 5V και 9V. Έτσι για την τροφοδοσία του κυκλώματος χρησιμοποιήθηκε μπαταρία 9V σε συνδυασμό με ρυθμιστή τάσης ο οποίος μπορεί να μεταβιβάσει την τάση σε 5V. Η μπαταρία είναι συστοιχία 8 στηλών 1,2V στα 2100mAh για περισσότερη διάρκεια. Ο ρυθμιστή τάσης είναι LM341 Voltage regulator που χρησιμοποιήθηκε έχει την ικανότητα να παρέχει σταθερή τάση 5V στο κύκλωμά, με τάση εισόδου από 9,6V έως και 7,2V.

Ολοκληρωμένο το Schematic του κυκλώματος

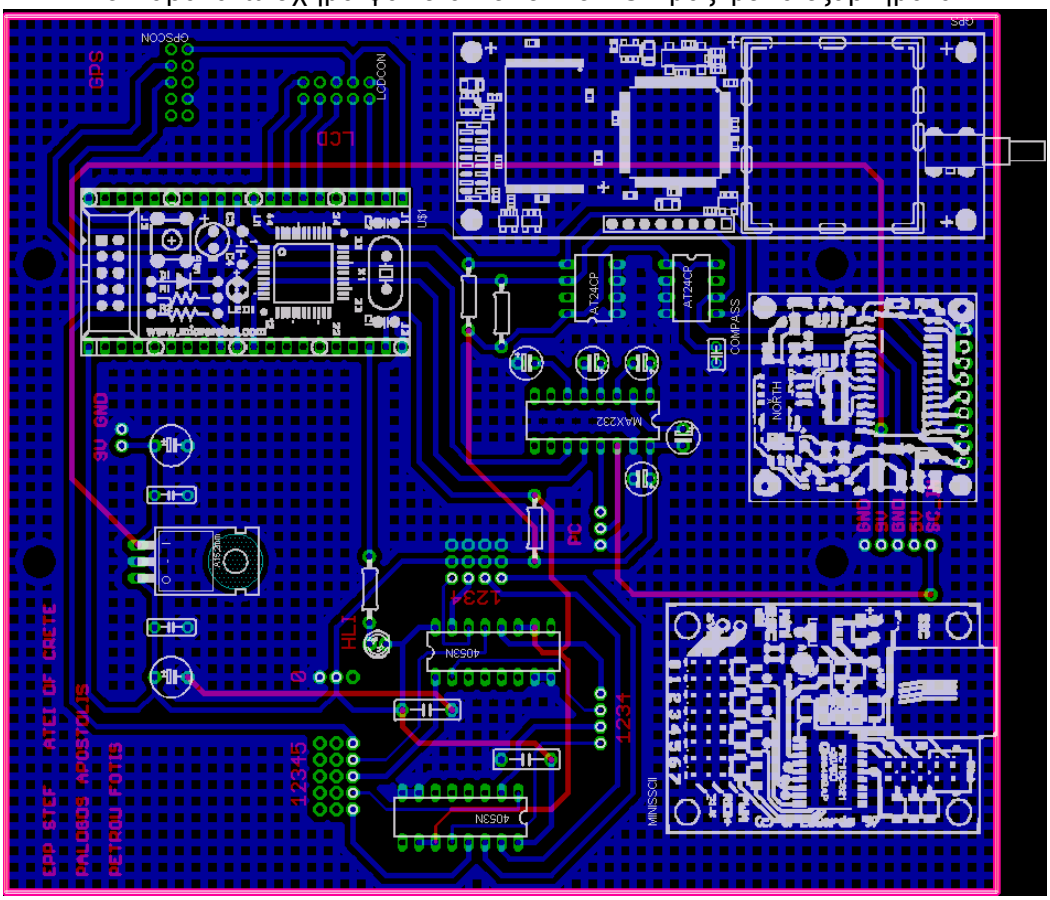


Από το Schematic κατασκευάστηκε το παρακάτω PCB σε δύο επίπεδα. Πολύ σημαντικό για την καλή γείωση του κυκλώματος ήταν η ελεύθερη επιφάνεια του κάτω επιπέδου (μπλε χρώμα) να παραμείνει με χαλκό. Το πλέγμα που έχει δημιουργηθεί είναι ενωμένο στη γείωση ενώ μερικά σημεία που αποκόπτονται έχουν βραχυκυκλωθεί από το πάνω επίπεδο. Στη συνέχεια δημιουργήθηκε πλέγμα από χαλκό και στο πάνω επίπεδο για τον ίδιο λόγο. Στη συνέχεια το πλέγμα του πάνω επιπέδου αποσυνδέεται από τη γείωση γιατί παρατηρήθηκε ότι πολλά εξαρτήματα δέχονται γείωση μέσα από βίδες και την ψύξη του LM341.

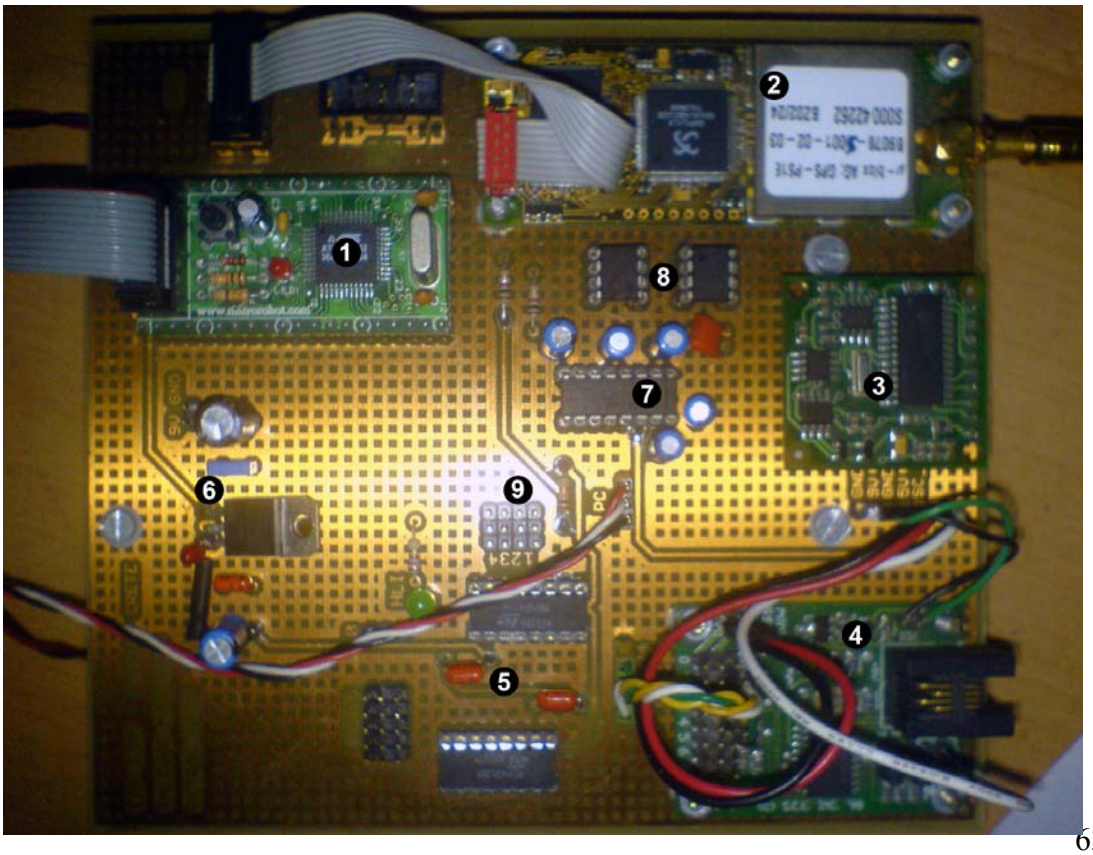
Το project του κυκλώματος για το Eagle 4.14 είναι διαθέσιμο στο φάκελο “Circuit Eagle” του CD. Το PCB με όνομα “Jet3.brd” και το Schematic με όνομα “ Jet3.sch”.



Στο παρακάτω σχήμα φαίνεται το τελικό PCB μαζί με τα εξαρτήματα.



Το κύκλωμα ολοκληρωμένο μετά την κατασκευή



Περιγραφή κυκλώματος

1	Ο Μικροεπεξεργαστής Atmega162V (Κιτ MR-162 της εταιρίας Microrobot). Ο programmer SPI προεκτάθηκε σε εξωτερική υποδοχή στο αερομοντέλο.
2	Ο δέκτης GPS (GPS-PS1E της εταιρίας μBlox)
3	Η ψηφιακή πυξίδα CMPS03 συνδεδεμένη πάνω στο i2c bus. Κατά τη τοποθέτησή της ρυθμίστηκε ο βορράς προς το μπροστά μέρος του αερομοντέλου
4	Το Mini SSC II Serial Servo Controller. Φαίνονται οι δύο τροφοδοσίες του 5 και 9V. Με άσπρο καλώδιο το καλώδιο δεδομένων και από το πολλαπλό οι έξοδοι των Servo PWM παλμών προς το κύκλωμα εναλλαγής.
5	Το κύκλωμα εναλλαγής Auto-Manual αποτελούμενο από τα δύο ολοκληρωμένα CD4053B και από πυκνωτές για ασφαλή τροφοδοσία. Φαίνεται το led το οποίο διακρίνει την Auto-Manual κατάσταση. Δεξιά του οι τέσσερις είσοδοι των παλμών PWM από το S/C. Αριστερά του οι τέσσερις είσοδοι των παλμών PWM από τον δέκτη της τηλεκατεύθυνσης καθώς και το πέμπτο κανάλι στη θέση “0”. Επάνω οι έξοδοι προς τους τέσσερις Servo-κινητήρες.
6	Η τροφοδοσία του κυκλώματος αποτελούμενη από το ρυθμιστή τάσης LM341. Παρέχει 9V στο S/C και 5V σε όλα τα υπόλοιπα εξαρτήματα. Αρχικά είχε σχεδιαστεί να ακουμπά στο πάνω επίπεδο του PCB για ψύξη. Λόγω της γείωσης όμως που ενώνει έχει σηκωθεί ελαφρώς προς τα επάνω.
7	Το MAX-232 της εταιρίας Maxim μετατρέπει τα επίπεδα τάσης της σειριακής USART0 του επεξεργαστή προς τη αναγνώρισή του από το S/C αλλά και από τον υπολογιστή. Διακρίνεται το καλώδιο “PC” το οποίο οδηγεί σε εξωτερική υποδοχή του αερομοντέλου που οδηγεί στη σύνδεση του κυκλώματος με τον υπολογιστή. Επειδή αρκούσαν μόνο τρία καλώδια για τη σύνδεση αυτή η υποδοχή έγινε με βύσμα στερεοφωνικού ηχου (mini jack) αντί για υποδοχή RS-232.
8	Οι δύο μνήμες τύπου EEPROM M24C64 με 64KBit η κάθε μία συνδεδεμένες σε σειρά στο i2c bus.
9	Η έξοδος προς τους τέσσερις Servo-κινητήρες. Σε αυτό το σημείο διορθώθηκε ένα πρόβλημα στη τροφοδοσία που παρατηρήθηκε κατά τη δοκιμή του κυκλώματος. Η μεγάλη κατανάλωση των κινητήρων δημιουργούσε γενική πτώση τάσης στη τροφοδοσία του κυκλώματος 5V, με αποτέλεσμα να υπάρξουν επαναλαμβανόμενα RESET του μικροεπεξεργαστή. Αυτό διορθώθηκε με αποκοπή των δρόμων της τροφοδοσίας των Servo από το κύκλωμα, ενώ προστέθηκε μια επιπλέον μπαταρία 5V 2100mAh εξολοκλήρου για αυτή τη χρήση.

5 Λογική πτήσης – Ρομποτικός έλεγχος

Στην υλοποίηση της εφαρμογής, χρησιμοποιήθηκε ένα μεγάλο κομμάτι μαθηματικών υπολογισμών για τον ρομποτικό έλεγχο. Στο κομμάτι αυτό περιλαμβάνονται υπολογισμοί ως προς τον εντοπισμό θέσης, την εκτίμηση της νέας κάθε φορά αντίδραση και τη λογική της πτήσης του αερομοντέλου γενικότερα. Ο στόχος είναι το σύστημα να είναι ικανό να ακολουθεί μια προδιαγεγραμμένη πορεία σε σχέση με το χώρο.

Το μοντέλο που χρησιμοποιήθηκε για τους υπολογισμούς είναι δυσδιάστατο, σε αντίθεση με το τρισδιάστατο μοντέλο το οποίο θα ήταν πιο σωστό. Αυτό έγινε για λόγους που θα εξηγήσουμε στη συνέχεια.

5.1 Αυτοεντοντοπισμός θέσης

5.1.1 Σύστημα γεωγραφικών συντεταγμένων

Το ευρύτερα χρησιμοποιούμενο σύστημα συντεταγμένων βασίζεται στην πολύ γνωστή απεικόνιση παραλλήλων και μεσημβρινών πάνω στη γήινη σφαίρα. Οι μεσημβρινοί (Meridian) έχουν για άκρα τους πόλους και είναι βαθμονομημένοι σε μοίρες, με το μηδέν στο μεσημβρινό του Greenwich (Prime Meridian) και αρίθμηση από 0 έως 180 μοίρες, Ανατολικά και Δυτικά, αντίστοιχα, του Greenwich. Οι τιμές αυτές εκφράζουν τη γωνία που σχηματίζει το κατακόρυφο επίπεδο που περνά από τον κάθε μεσημβρινό σε σχέση με το κατακόρυφο επίπεδο που περνά από το μεσημβρινό αναφοράς και αναφέρονται ως γεωγραφικό μήκος (Longitude). Οι παράλληλοι (Parallels) είναι νοητοί κύκλοι παράλληλοι με τον ισημερινό (Equator). Οι παράλληλοι είναι και αυτοί βαθμονομημένοι σε μοίρες με το μηδέν στον Ισημερινό και το 90 στο Βόρειο και το Νότιο πόλο. Η αρίθμηση αυτή, από 0 έως 90 μοίρες βόρεια ή νότια του ισημερινού, εκφράζει τη γωνία που σχηματίζεται ανάμεσα στο οριζόντιο επίπεδο που περνά από τον ισημερινό και μια ευθεία που ενώνει το κέντρο της γης με τον κάθε παράλληλο, και αναφέρεται ως γεωγραφικό πλάτος (Latitude).

Η Ελλάδα βρίσκεται ανάμεσα στον 34ο και τον 42ο παράλληλο (βόρειο πλάτος) και τον 19ο και 29ο μεσημβρινό (ανατολικό μήκος).

Κάθε μοίρα διαιρείται σε 60 πρώτα λεπτά. Κάθε πρώτο λεπτό διαιρείται σε 60 δεύτερα λεπτά (τα GPS εκτός από δεύτερα λεπτά δίνουν και μονάδες του δεκαδικού συστήματος, συνήθως χιλιοστά).

Με το σύστημα αυτό, κάθε σημείο της γήινης σφαίρας προσδιορίζεται ως η τομή ενός παραλλήλου με έναν μεσημβρινό και εκφράζεται με γωνίες γεωγραφικού μήκους και πλάτους (φ και λ).

Οι ερασιτεχνικού τύπου δέκτες GPS επιτρέπουν τον προσδιορισμό της θέσης μας με σχετική ακρίβεια (τις συντεταγμένες με ακρίβεια περίπου 15 μ. και το υψόμετρο μέσα σε ένα εύρος 50 μ.). Το σφάλμα υπολογισμού της θέσης από το GPS εξαρτάται από τις συνθήκες της λήψης (φυσικά εμπόδια, αντανακλάσεις) και τον αριθμό και τη διάταξη των δορυφόρων. Το σφάλμα είναι μεγαλύτερο στον υπολογισμό του υψομέτρου της θέσης. Μερικά μοντέλα GPS προσπαθούν να παρακάμψουν το πρόβλημα αυτό προσθέτοντας ένα βαρομετρικό αλτίμετρο στη συσκευή (έτσι ωστόσο προκύπτουν δύο πηγές πληροφοριών με διαφορετικό τύπο σφάλματος). Η μέτρηση του δέκτη που χρησιμοποιήθηκε είναι 1 με 2 μέτρα για το γεωγραφικό πλάτος και μήκος ενώ για το ύψος είναι 10 με 15 μέτρα. Η ακρίβεια του υψόμετρου του GPS χαρακτηρίζεται αρκετά χαμηλή για την εφαρμογή. Έτσι ο έλεγχος δεν βασίζεται στη μέτρηση αυτή κάτι το οποίο αν γινόταν, η εφαρμογή θα γινόταν δυσλειτουργική έως και επικίνδυνη. Αυτός είναι και ο λόγος για τον οποίο χρησιμοποιήθηκε επίπεδο δύο διαστάσεων για τους υπολογισμούς όπως θα δούμε παρακάτω.

Παρόλα αυτά η μέτρηση λαμβάνεται υπόψη μόνο κατά την καταγραφή των σημείων στη μνήμη για μετέπειτα επεξεργασία και συμπεράσματα. Σε περίπτωση σημαντικής απώλειας ύψους, το αερομοντέλο επαναφέρεται χειροκίνητα χρησιμοποιώντας το κύκλωμα εναλλαγής που αναλύσαμε στο προηγούμενο κεφάλαιο.

5.1.2 NMEA GGA

Για τις διάφορες ανάγκες έχουν δημιουργηθεί διάφοροι τρόποι εξόδου της πληροφορίας από τους δέκτες GPS. Στην περίπτωση της εφαρμογής χρησιμοποιήθηκε η πρόταση εξόδου GGA του πρωτοκόλλου NMEA. Η έξοδος αυτή παράγεται κάθε δευτερόλεπτο και είναι της μορφής:

$GPGGA,225719.605,3520.0790,N,02507.3319,E,1,05,1.8,63.0,M,,,,0000*3D

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator[a]	1		
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude[b]	9.0	meters	
Units	M	meters	
Geoid Separation[b]		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
CR LF			End of message termination

Από την προηγούμενη εικόνα φαίνεται ότι οι συντεταγμένες γεωγραφικού μήκους και πλάτους είναι της μορφής dd°mm.mmmm. Για παράδειγμα οι συντεταγμένες του εργαστηρίου ρομποτικής έτσι όπως δίνονται από το GGA είναι:

Γεωγραφικό πλάτος(Latitude) 3519.0489 N
Γεωγραφικό μήκος(Longitude) 2506.0957 E

Δηλαδή για το γεωγραφικό πλάτος έχουμε 35 μοίρες 19 λεπτά της μοίρας, 04,89 εκατοστά του λεπτού της μοίρας (όχι δευτερόλεπτα), στο βόρειο πλάτος. Μια μοίρα αντιστοιχεί σε 60 λεπτά της μοίρας. Για την απλότητα των υπολογισμών στον μικροεπεξεργαστή χρειάζεται οι παραπάνω αριθμοί να μετατραπούν στη μορφή dd.ddddd° δεκαδικών μοιρών. Παίρνοντας μόνο το κομμάτι των λεπτών mm.mmmm και διαιρώντας με 60 παίρνουμε την δεκαδική μορφή των λεπτών. Αυτός ο αριθμός θα είναι ο δεκαδικός μετά την υποδιαστολή στη μορφή dd.ddddd° σε μοίρες. Για παράδειγμα τα δεκαδικά του γεωγραφικού πλάτους θα είναι 19,0489 / 60 = 0,317481 και αντίστοιχα οι συντεταγμένες για το εργαστήριο ρομποτικής θα είναι σε μοίρες:

Γεωγραφικό πλάτος(Latitude) 35,317481 N
Γεωγραφικό μήκος(Longitude) 25,101595 E

5.1.3 Περιορισμοί συντεταγμένων

Στη μορφή με μοίρες των συντεταγμένων, όπως η παραπάνω, το τελευταίο ψηφίο έχει πολύ μεγάλη ακρίβεια, τόσο που το καθιστά άχρηστο για το σκοπό της εφαρμογής. Με ένα πρόχειρο υπολογισμό αν υποθέσουμε ότι μια μέση ταχύτητα του αερομοντέλου είναι 70 km/h τότε κάθε δευτερόλεπτο διανύει περίπου 20 m. Η ακρίβεια που δίνει το τελευταίο ψηφίο είναι λιγότερο του 1 m και πολλές φορές έχει μεγάλες διακυμάνσεις ακόμα και σε σταθερό σημείο. Για αυτό το λόγο δεν λαμβάνεται υπόψη το ψηφίο αυτό.

Επίσης για λόγους οικονομίας της μνήμης στην οποία καταγράφονται όλα τα σημεία, δεν λαμβάνονται υπόψη τα δύο πρώτα ψηφία. Τα ψηφία αυτα παριστάνουν τον 35^{o} παράλληλο και τον 25^{o} μεσημβρινό σε μοίρες. Έτσι όλα τα σημεία θεωρείται ότι βρίσκονται μέσα σε αυτά τα όρια που ορίζουν οι δύο αυτές μοίρες, με την προϋπόθεση ότι δεν θα δοκιμαστεί πτήση έξω από αυτά.

Η πρόταση GGA φαίνεται παρακάτω με τονισμένα τα νούμερα ενδιαφέροντος (Latitude, Longitude, Altitude):

$GPGGA,225719.605,**3520.0790**,N,**02507.3319**,E,1,05,1.8,**63.0**,M,,,,0000*3D
20 0790 07 3319 63

Από τα νούμερα αυτά αφαιρώντας από το γεωγραφικό μήκος και πλάτος τα δύο πρώτα ψηφία, καθώς και το τελευταίο, δημιουργείται ένας 5ψήφιος αριθμός για το καθένα. Επίσης πάλι για λόγους οικονομίας αφαιρείται και το δεκαδικό ψηφίο του ύψους. Το πρόγραμμα έχει φτιαχτεί έτσι ώστε να μη λαμβάνεται υπόψη η ενδιάμεσα υποδιαστολή κατα τη λήψη των αριθμών αυτών. Συνεπώς χρειάζεται να διαιρέσουμε με 60 και πολλαπλασιάσουμε με 100 για να έχουμε σωστά μεγέθη χωρίς να χάσουμε ακρίβεια. Με απλοποίηση λοιπόν το νούμερο πολλαπλασιάζεται με 5/3 για την μετατροπή του μέσα στο πρόγραμμα ενώ κρατούνται μόνο τα ακέραια μέρη.

Κατά τη διάρκεια όμως της υλοποίησης του προγράμματος προέκυψε ένας ακόμα περιορισμός. Οι μεταβλητές που θα αποθηκεύονται οι συντεταγμένες δηλώθηκαν σκόπιμα ως μη προσημασμένοι ακέραιοι τόσο για οικονομία σε χώρο, όσο και για ταχύτητα των πράξεων. Έτσι ένας unsigned int καταλαμβάνει 16bit και εφόσον γίνεται εκμετάλλευση του χώρου που καταλαμβάνει το πρόσημο (σε αντίθεση με έναν απλό int) τα όρια του είναι από 0 μέχρι 65535. Αυτό σημαίνει ότι υπάρχει νέος περιορισμός στο χώρο στον οποίο πρόκειται να πετάξει ρομποτικά το αερομοντέλο. Ουσιαστικά τα όρια δεν θα πρέπει να ξεπερνάνε το 35,65535° σε πλάτος και το 25,65535° σε μήκος, κάτι που όπως φαίνεται και στη παρακάτω εικόνα, δεν αποτελεί σοβαρή απώλεια για τους στόχους της εργασίας.

5.2 Μαθηματικός έλεγχος

Σύμφωνα με τους παραπάνω χωρικούς περιορισμούς είναι γνωστό το πεδίο τιμών των σημάτων που πρόκειται να ληφθούν. Για τον υπολογισμό αποστάσεων (λίγων Km) βασισμένα σε συντεταγμένες χρησιμοποιούνται τύποι που σχετίζουν το γεωγραφικό μήκος της κάθε περιοχής μιας και δεν είναι σταθερό. Για παράδειγμα αν χρησιμοποιηθεί το αερομοντέλο για πτήση από το νοτιότερο σημείο του εύρους(Χάρακας ή Βιάννος) μέχρι το βορειότερο(νήσος Ντία), τότε θα πρέπει οι αποστάσεις να υπολογιστούν σύμφωνα με τον τύπο: (με Lat1, Lon1, Lat2, Lon2 γνωστά)

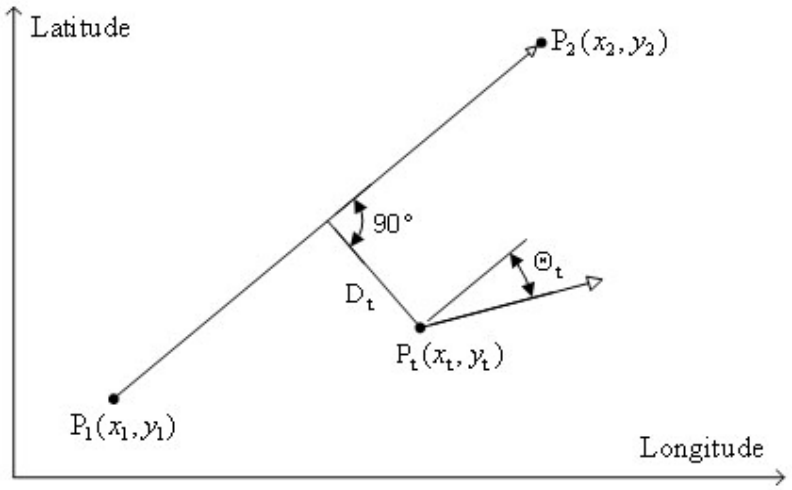
dLat = Lat1 - Lat2
dLon = Lon1 - Lon2

dMLat = dLat * 111'199.233
dMLon = dLon * (111'199.233 * cos (Lat * PI / 180))

D = square_root($dMLon^2$ + $dMLat^2$)

Επειδή όμως ο χώρος στον οποίο γίνεται χρήση των αερομοντέλων δεν είναι πάνω από 0,5x0,5 km μπορούμε να θεωρήσουμε ένα καρτεσιανό επίπεδο για να κάνουμε τους υπολογισμούς μας. Έτσι σε αυτό το επίπεδο ας θεωρήσουμε ένα ευθύγραμμο τμήμα το οποίο αντιπροσωπεύει την θεωρητική πορεία που θα πρέπει να ακολουθήσει το αεροπλάνο. Τα δύο αυτά σημεία $P_1(x_1, y_1)$ και $P_2(x_2, y_2)$ που ορίζουν την πορεία, είναι γνωστά εξ' αρχής και θα υπάρχει η δυνατότητα να επιλέγονται πριν την πτήση, προγραμματίζοντας κατάλληλα το σύστημα.

Μπορούμε τώρα να θεωρήσουμε ένα σημείο $P_t(x_t, y_t)$ στο επίπεδο αυτό και να το αντιστοιχήσουμε με την θέση του αεροπλάνου την χρονική στιγμή t. Το σημείο P_t υποθέτουμε ότι έχει απόσταση D_t από το ευθύγραμμο τμήμα και κατεύθυνση που σχηματίζει απόκλιση γωνίας Θ_t με αυτό.



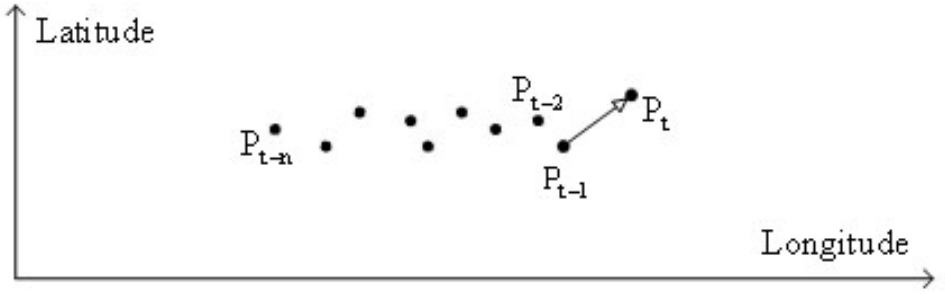
Το σύστημα θα πρέπει να είναι ικανό να αντιδρά προς τη μείωση της απόστασης που έχει από την πορεία του, καθώς και της απόκλισης από την κατεύθυνσή της. Αν κάθε στιγμή επιτυγχάνεται σωστά η μείωση αυτή, τότε το αερομοντέλο θα είναι ικανό να παραμείνει πάνω στη τροχιά του. Υποθέτοντας ότι η διόρθωση θα γίνεται κάθε δευτερόλεπτο (ίδια με τη συχνότητα των σημείων από το GPS) τότε το σύστημα θα διορθώνει χωρίς διακοπή κάθε 20 ή και λιγότερα μέτρα πτήσης. Ανάλογα με το μέγεθος των δύο αυτών μεταβλητών, είναι και το μέγεθος της αντίδρασης. Ο τρόπος με τον οποίο μπορεί να γίνει αυτό ποικίλει στο χώρο της ρομποτικής. Γενικά χρησιμοποιούνται γραμμικές, διαφορικές μέθοδοι ή μέθοδοι ολοκλήρωσης έτσι ώστε να επιτευχθεί η σωστή αντίδραση (απότομη, νωχελική αντίδραση κ.α.) ή ακόμα και συνδυασμοί αυτών. Στη περίπτωση του αερομοντέλου χρησιμοποιήθηκε γραμμική μέθοδος ελέγχου, που χαρακτηρίζει και την συμπεριφορά διόρθωσής του. Θεωρούμε λοιπόν μια εντολή F της οποίας το ποσό θα καθορίσει σε γενικές γραμμές το πόσο μεγάλη ή μικρή θα είναι η αντίδραση των επιφανειών ελέγχου του αερομοντέλου (aileron), ενώ το πρόσημό της θα καθορίσει την φορά (δεξιά ή αριστερά). Η εντολή αυτή πρέπει να είναι γραμμικά ανάλογη των δύο μεταβλητών, κατά αναλογία των D_t και Θ_t κάθε χρονική στιγμή. Η εντολή αυτή περιγράφεται με τον παρακάτω τύπο:

$$\boxed{F_t = K_1 \cdot D_t + K_2 \cdot \Theta_t}$$

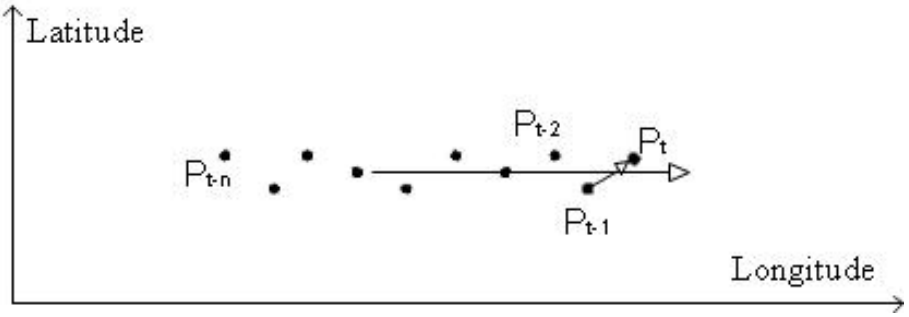
Οι σταθερές K_1 και K_2 επηρεάζουν το τελικό μέγεθος της εντολής F και κρίνουν τη συμπεριφορά αντίδρασης του συγκεκριμένου αερομοντέλου. Ο καλύτερος τρόπος για να οριστούν οι σταθερές αυτές είναι μετά από πειραματικές δοκιμές, όπου με κάθε διόρθωση προσεγγίζουμε όλο και πιο πολύ την επιθυμητή αντίδραση.

5.2.1 Μέθοδος ελαχίστων τετραγώνων

Η κατεύθυνση του αερομοντέλου μπορεί πολύ εύκολα να βρεθεί, βάση του GPS, συγκρίνοντας την τρέχουσα θέση του, με την προηγούμενη. Πολλές φορές αυτή η εκτίμηση μπορεί να είναι λανθασμένη σε σχέση με την πραγματική. Πολλοί λόγοι όπως ο θόρυβος στο σήμα του GPS ή η απουσία δορυφόρων τη στιγμή της λήψης έχουν σαν αποτέλεσμα να δημιουργούνται απότομες, μη αναμενόνες, μετρήσεις ακόμα και αν το σύστημα είναι σταθερό σε ένα σημείο. Στην παρακάτω εικόνα παρατηρούμε οτι η φορά του κινούμενου σήματος είναι προς τα δεξιά, ενώ η εκτίμηση της φοράς από τα δύο τελευταία σημεία είναι αρκετά αλλοιωμένη.



Σε πολλά πειράματα υπάρχει μία γραμμική σχέση ανάμεσα στα μετρούμενα μεγέθη. Τοποθετώντας τα σημεία σε ένα διάγραμμα όπως το παραπάνω, βλέπουμε ότι αυτά προσεγγίζουν μία ευθεία γραμμή. Το επόμενο βήμα είναι να βρούμε την κλίση της ευθείας η οποία προσεγγίζει περισσότερο αυτά τα σημεία, και το σημείο στο οποίο αυτή τέμνει τον άξονα y (τεταγμένη). Σε κάθε περίπτωση, δεν περιμένουμε η ευθεία να περνά από όλα τα σημεία, λόγω της παρουσίας τυχαίων σφαλμάτων. Η μέθοδος με την οποία μπορούμε να βρούμε την εξίσωση τη ευθείας αυτής είναι η μέθοδος των ελαχίστων τετραγώνων.



Σύμφωνα με τη μέθοδο θεωρούμε μια γενική γραμμική εξίσωση $y = m \cdot x + b$ όπου η κλίση m και η τεταγμένη b βρίσκονται με την βοήθεια οριζουσών:

$$D = \begin{vmatrix} \Sigma(x_i^2) & \Sigma x_i \\ \Sigma x_i & i \end{vmatrix} \qquad m = \begin{vmatrix} \Sigma(x_i * y_i) & \Sigma x_i \\ \Sigma y_i & i \end{vmatrix} / D \qquad b = \begin{vmatrix} \Sigma(x_i^2) & \Sigma(x_i * y_i) \\ \Sigma x_i & \Sigma y_i \end{vmatrix} / D$$

Με απλοποίηση των παραπάνω τύπων καταλήγουμε στις παρακάτω εκφράσεις για την κλίση και την τεταγμένη της καλύτερης ευθείας των σημείων:

$$m = \frac{N\sum(x_i y_i) - \left(\sum x_i\right)\left(\sum y_i\right)}{N\sum\left(x_i^2\right) - \left(\sum x_i\right)^2}$$

$$b = \frac{\sum y_i - m\sum x_i}{N}$$

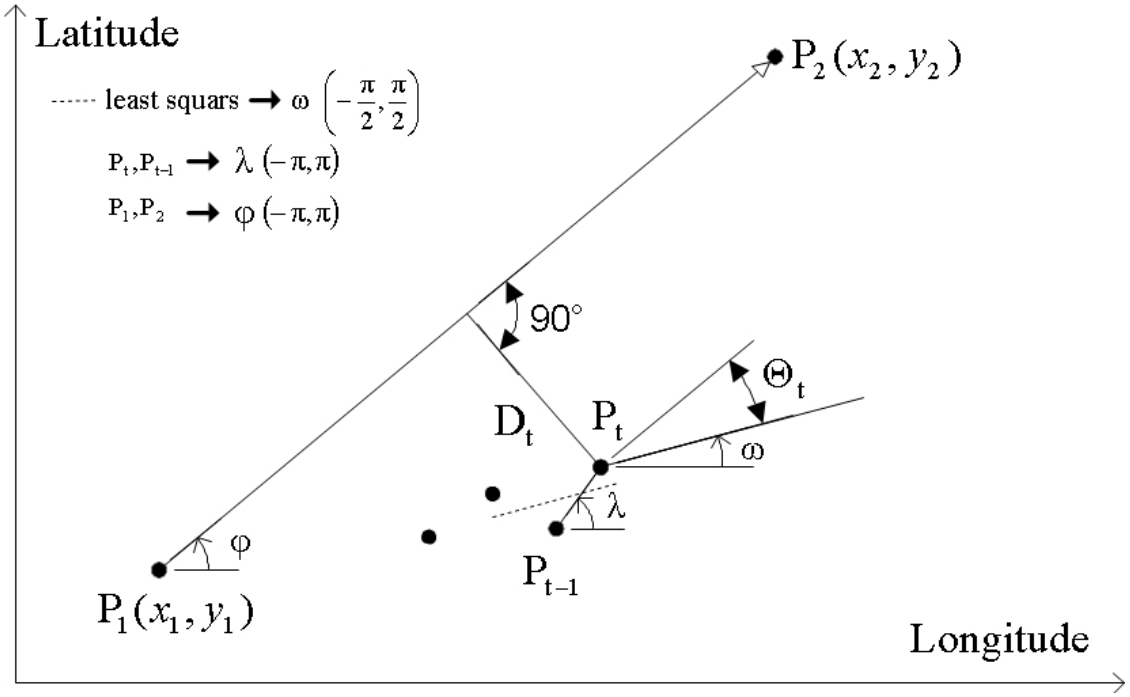
Όπου N είναι ο αριθμός των σημείων που μετέχουν για την εύρεση της εξίσωσης της ευθείας $y = m \cdot x + b$. Οι παραπάνω εκφράσεις είναι πιο απλές και ευκολότερες σε υλοποίηση από τον μικροεπεξεργαστή. Στην δικιά μας περίπτωση, για καλύτερη προσέγγιση της ευθείας εφαρμόζουμε τον τύπο μόνο για τα τρία τελευταία σημεία. Ο λόγος είναι γιατί χρειάζεται να προσδιοριστεί η φορά σε κάποια χρονική στιγμή με κάποια ακρίβεια, χωρίς να συμπεριλαμβάνονται πολλά σημεία κάτι το οποίο θα είχε σαν αποτέλεσμα οι αλλαγές να γίνονται πολύ αργά αντιληπτές. Βρίσκοντας όμως την εξίσωση της καλύτερης ευθείας ουσιαστικά προσδιορίζουμε την σωστότερη ευθεία διεύθυνσης και όχι την φορά. Για αυτήν μένουν δύο επιλογές για να προσδιορισθεί το προς τα που κινείται το αεροπλάνο.

5.2.2 Εύρεση διανύσματος κίνησης

Το διάνυσμα της κίνησης θα έχει ως αρχή το τελευταίο σημείο GPS που λαμβάνεται και κατεύθυνση ίδια με αυτή που περιγράφει η καλύτερη ευθεία. Για να γίνει γνωστό προς τα ποια κατεύθυνση, πάνω στην ευθεία, κινείται το αεροπλάνο συγκρίνουμε την γωνία της καλύτερης ευθείας με τη γωνία που προκύπτει από τα δύο προηγούμενα σημεία. Από τα σημεία P_1, P_2 έχουμε τη γωνία φ η οποία είναι εκφρασμένη σε γωνία με πεδίο ορισμού $(-\pi,\pi)$ με τη βοήθεια της συνάρτησης atan2. Εάν γινόταν χρήση της εφαπτομένης atan θα είχαμε γωνία $\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$ και σαν αποτέλεσμα μια ευθεία να αντιστοιχεί σε μια θετική και μια αρνητική γωνία. Από τα σημεία P_{t-1}, P_t εφόσον είναι γνωστά με τον ίδιο τρόπο βρίσκουμε τη γωνία λ. Από την ευθεία με τη μέθοδο ελαχίστων τετραγώνων των τριών τελευταίων σημείων προκύπτει η γωνία ω με χρήση της εφαπτομένης, δηλαδή σε πεδίο $\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$. Άρα το τελικό διάνυσμα έχει γωνία ω ή $\omega+\pi$. Για πιο σωστά έτσι ώστε να έχουμε και τη γωνία αυτή σε πεδίο $(-\pi,\pi)$ θα έχουμε τις δύο πιθανές γωνίες α,β:

$$\begin{cases} \alpha = \omega \\ \beta = \begin{cases} \omega+\pi \text{ αν } \omega<0 \\ \omega-\pi \text{ αν } \omega>0 \end{cases} \end{cases}$$

Η σωστή γωνία είναι αυτή που έχει την απολύτως μικρότερη διαφορά από τη γωνία λ. Έτσι αν ισχύει $|\lambda-\alpha| \geq |\lambda-\beta|$ τότε το ζητούμενο διάνυσμα έχει γωνία α, αλλιώς ισχύει η β.



5.2.3 Εντοπισμός γωνίας – απόστασης

Η γωνία που μας ενδιαφέρει είναι αυτή που σχηματίζεται μεταξύ της πραγματικής και της επιθυμητής κατεύθυνσης. Από τα προηγούμενα έχει προσεγγιστεί με καλή ακρίβεια το πραγματικό διάνυσμα που ορίζεται από το σημείο P_t και από τη γωνία ω. Έτσι η γωνία Θ_t θα είναι η διαφορά $\varphi - \omega$ διορθώνοντας την γωνία αυτή έτσι ώστε να προκύψει τιμή μεταξύ $(-\pi, \pi)$:

$$\Theta_t = \begin{cases} \Theta_t - 2\pi, & \text{αν } \Theta_t > \pi, \\ \\ \Theta_t + 2\pi, & \text{αν } \Theta_t < -\pi \end{cases}$$

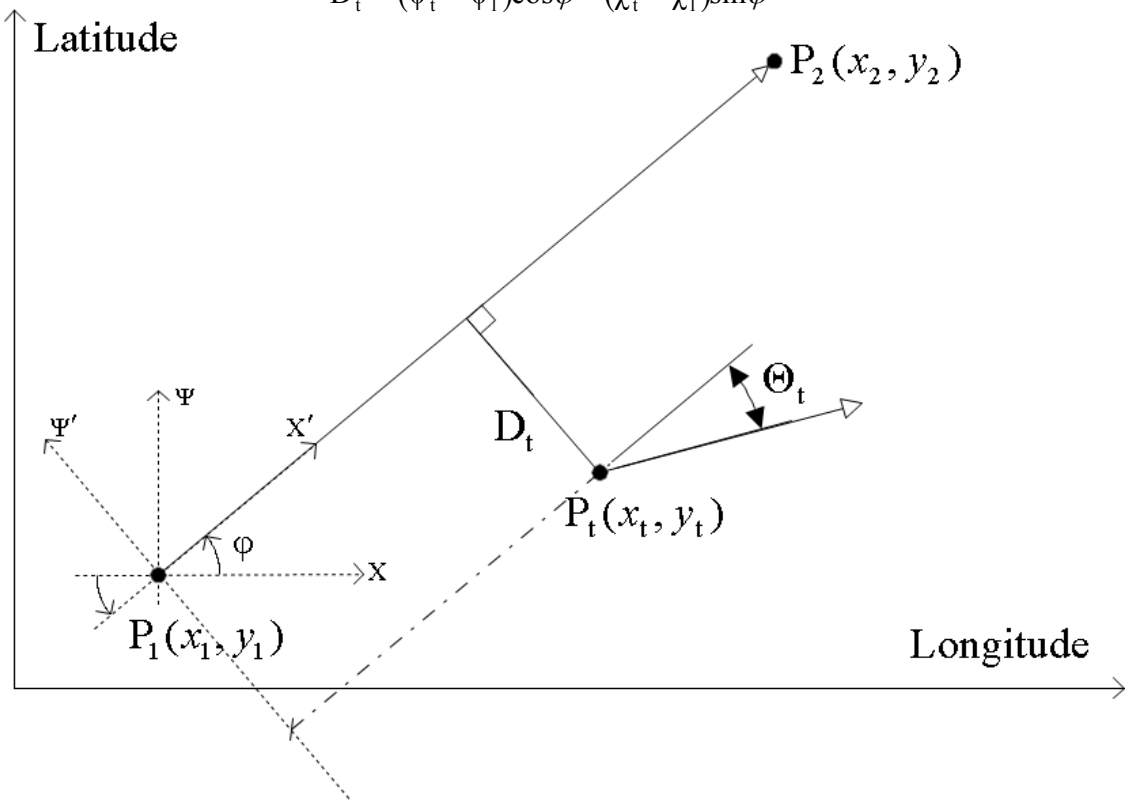
Στην εφαρμογή μας θετικές τιμές της Θ_t σημαίνουν δεξιές αποκλίσεις σε σχέση με την θεωρητική πορεία, ενώ αρνητικές, αριστερές αποκλίσεις.

Η απόσταση D_t μπορεί να βρεθεί ποσοτικά χρησιμοποιώντας τον τύπο απόστασης σημείου από ευθεία $d = \frac{|A\chi + B\psi + \Gamma|}{\sqrt{A^2 + B^2}}$. Ο τύπος δίνει σωστά την απόσταση χωρίς πρόσημο λόγω του απόλυτου. Αν αφαιρεθεί το απόλυτο το πρόσημο μεταβάλλεται ανάλογα με το πρόσημο της κλίσης της ευθείας, με αποτέλεσμα να μην είναι σίγουρο πάντοτε αν η θετική απόσταση είναι, για παράδειγμα, από τα δεξιά της ευθείας που είναι και το ζητούμενο. Για το λόγο αυτό έγινε εφαρμογή της μεταφοράς και της στρέψης αξόνων.

Αν μετατοπιστεί η αρχή των αξόνων σε σημείο (χ_0, ψ_0) τότε ένα σημείο που χαρακτηριζόταν με (χ, ψ) στο προηγούμενο σύστημα αξόνων, στο νέο θα είναι $\begin{pmatrix} X = \chi - \chi_0 \\ \Psi = \psi - \psi_0 \end{pmatrix}$. Αν τώρα αυτό το στρέψουμε και κατά γωνία θ τότε το σημείο θα περιγράφεται από τις συντεταγμένες $\begin{pmatrix} X' = X\cos\theta + \Psi\sin\theta \\ \Psi' = \Psi\cos\theta - X\sin\theta \end{pmatrix}$. Αν λοιπόν μετακινήσουμε το σύστημα των αξόνων πάνω στο σημείο P_t και το στρέψουμε κατά γωνία Θ_t θα είμαστε σε θέση να γνωρίζουμε σε ποιο σημείο του άξονα ψ τέμνεται η ευθεία και συνεπώς την απόσταση του από αυτή. Για καλύτερη ταχύτητα των πράξεων στον μικροελεγκτή μεταφέρουμε τους άξονες πάνω στο σημείο P_1 και με γωνία φ, δηλαδή πάνω στη θεωρητική πορεία αντί πάνω στο τρέχων διάνυσμα. Αυτό έχει σαν αποτέλεσμα τα ημίτονα και συνημίτονα να είναι σταθερά και να υπολογίζονται μια φορά στην αρχή του προγράμματος εφόσον η γωνία φ είναι γνωστή εξαρχής. Σε αντίθετη περίπτωση θα πρέπει να υπολογίζονται κάθε δευτερόλεπτο με νέα κάθε φορά γωνία

Θ_t. Επομένως υπολογίζουμε την Ψ' από τους προηγούμενους τύπους αντικαθιστώντας όπου (χ,ψ) τις συντεταγμένες του σημείου P_t και όπου (χ_0,ψ_0) τις συντεταγμένες του σημείου P_1. Η τιμή αυτή θα είναι και η απόσταση D_t με πρόσημο που θα αναφέρει αν είναι δεξιά ή αριστερά της θεωρητικής πορείας.

$$D_t = (\psi_t - \psi_1)\cos\varphi - (\chi_t - \chi_1)\sin\varphi$$



Για την συνολική πορεία του αερομοντέλου υπολογίζεται και απόσταση του P_t, P_2 με τον τύπο απόστασης σημείου προς σημείου. Η τιμή αυτή συγκρίνεται κάθε φορά με μία σταθερά (πχ 30m) και αν βρεθεί μικρότερη τότε γίνεται αλλαγή του ευθυγράμμου τμήματος που υπολογίζονται οι τιμές Θ_t, D_t. Με αυτό τον τρόπο γίνεται η μεταγωγή τμημάτων έτσι ώστε να ολοκληρωθεί μία πορεία. Παρόλα αυτά χρειάζεται μεγάλη προσοχή κατά τον ορισμό των σταθερών K_1 και K_2 στον τελικό νόμο ελέγχου γιατί υπάρχει περίπτωση το αερομοντέλο να ισορροπήσει ή να ταλαντεύεται γύρω από την θεωρητική πορεία σε τέτοια απόσταση που η μεταγωγή τμημάτων να μην είναι δυνατή. Σε μια τέτοια περίπτωση θα αγνοηθεί το επόμενο ευθύγραμμο τμήμα με κίνδυνο την μεγάλη απομάκρυνση του αερομοντέλου.

6 Προγραμματισμός

6.1 Μικροελεγκτής

Στη παράγραφο αυτή θα αναφερθούν κάποιες βασικές συναρτήσεις που υλοποιήθηκαν κατά τον προγραμματισμό του ATmega162. Μια ρύθμιση του CodevisionAVR για τον άμεσο προγραμματισμό του μικροελεγκτή μετά από το Compile είναι στο Project>Configure>After make tab>Program the Chip. Επίσης χρησιμοποιήθηκε αρκετά η σειριακή θύρα προς τον υπολογιστή για επαλήθευση των μαθηματικών αποτελεσμάτων. Αν γίνει χρήση της τροποποιημένης printf του Codevision, μια ρύθμιση για την σωστή απεικόνιση των αποτελεσμάτων είναι πάλι στο Configure στο C Compiler tab>Code Generation tab κάτω από το (s)printf features η επιλογή float, width, precision.

Το πρόγραμμα αποτελείται από τέσσερα αρχεία κατάληξης *.c source code. Το βασικό πρόγραμμα είναι το main.c όπου γίνεται η αρχικοποίηση των περισσότερων καταχωρητών από το Codevision. Το αρχείο controlfaethon.c περιέχει συναρτήσεις που αφορούν τις τελικές κινήσεις των servo κινητήρων. Το αρχείο eepromfaethon.c περιλαμβάνει συναρτήσεις οι οποίες διαχειρίζονται τις δύο εξωτερικές μνήμες eeprom ενώ το lcdfaethon.c κάποιες διαμορφωμένες συναρτήσεις για την οδήγηση της lcd οθόνης. Για κάθε αρχείο εκτός από το main.c έχει δημιουργηθεί και ένα επιπλέον κατάληξης *.h τα οποία είναι header files και χρησιμοποιούνται για να δηλώνονται οι συναρτήσεις που υπάρχουν σε αυτά. Για να χρησιμοποιηθούν τα αρχεία και συνεπώς οι συναρτήσεις αυτών αρκεί να δηλώσουμε στην αρχή του main.c την εντολή #include <..> και το όνομα του header αρχείου. Ένα πρόβλημα είναι όταν σε περίπτωση που ένα include file χρειαστεί ένα άλλο και πρέπει να δηλωθεί στην αρχή του, τότε ο Compiler αποδίδει σφάλμα διπλής δήλωσης. Για να αποφευχθεί αυτό υπάρχει ρύθμιση στο Project>Configure>Files tab όπου δηλώνουμε στον Compiler ποια αρχεία θα χρησιμοποιηθούν.

Από το wizard του Codevision επιλέχθηκε ο τύπος και η συχνότητα του μικροελεγκτή (16MHz), τέθηκε η οθόνη lcd στην PORTA και το i2c bus στην PORTB στο bit 0 το sda και στο bit 1 το scl. Επίσης ενεργοποιήθηκαν τα δύο εξωτερικά interrupts 0 και 1 το ένα σε rising και το άλλο σε falling edge όπως περιγράψαμε στο κεφάλαιο του κυκλωματικού. Ακόμη ενεργοποιήθηκε ο 8 bit timer με επιλογή του overflow interrupt και οι δύο σειριακές USART0 και USART1 με receive interrupt και στα δύο. Η USART0 χρησιμοποιείται για την επικοινωνία με τον υπολογιστή και του servo controller και ρυθμίστηκε στα 9600 baud rate λόγω περιορισμού του δεύτερου. Η USART1 χρησιμοποιείται εξολοκλήρου από το GPS, ρυθμίστηκε στα 19200 baud rate και η λήψη των χαρακτήρων NMEA έγινε με χρήση του receive interrupt. Ένα ακόμη που προστέθηκε ήταν το broun out detector για την ασφάλεια της μνήμης eeprom του μικροελεγκτή. Έχει παρατηρηθεί ότι όταν η τάση λειτουργίας πέσει κάτω από ένα συγκεκριμένο κατώφλι ο επεξεργαστής είναι ασταθής και υπάρχει πιθανότητα να

γραφούν τυχαία δεδομένα στην μνήμη eeprom του. Με το broun out detector δίνουμε εντολή να μεταπηδήσει το πρόγραμμα σε ένα label (π.χ. goto x;) όταν συμβεί κάτι τέτοιο.

Η λήψη του NMEA string από το GPS γίνεται με χρήση receive interrupt της σειριακής USART1 και πραγματοποιείται μόνο κατά τη διάρκεια της ρομποτικής λειτουργίας. Αυτό επιτυγχάνεται με χρήση μιας global bit μεταβλητής auto. Από το string λαμβάνονται μόνο τα επιθυμητά ψηφία όπως είναι το longitude, το latitude και το elevation. Οι υπόλοιπες ενδείξεις αγνοούνται για εξοικονόμηση ταχύτητας από το interrupt. Δηλώνεται ένας πίνακας χαρακτήρων rx_buffer1[RX_BUFFER_SIZE1] ο οποίος αρχίζει να συμπληρώνεται με τα παραπάνω στοιχεία όταν ληφθεί ο χαρακτήρας εκκίνησης '$'. Όταν προσπελαστούν οι συγκεκριμένες θέσεις από τους λαμβανόμενους χαρακτήρες γίνεται και η καταχώρησή τους στο πίνακα. Μόλις ολοκληρωθεί η λειτουργία του interrupt τίθεται η τιμή 1 στη global bit μεταβλητή rx_buffer_overflow1. Μετά από τη διαδικασία αυτή καλείται η συνάρτηση CalcPosition(), η οποία φιλτράρει το συμπληρωμένο πίνακα και μετατρέπει τα αλφαριθμητικά σε αριθμούς.

```
#define RX_BUFFER_SIZE1 17
char rx_buffer1[RX_BUFFER_SIZE1];
bit rx_buffer_overflow1;

interrupt [USART1_RXC] void usart1_rx_isr(void)
{
      if (auto==1){
            char data;
            static unsigned char ptr,index;
            data=UDR1;

            if (data=='$') {

                  ptr=0;index=0;
                  rx_buffer1[index]=data;
                  putchar(rx_buffer1[index]);
                  index=1;
            };
if (((ptr>19)&&(ptr<26))||((ptr>32)&&(ptr<39))||((ptr>51)&&(ptr<56))){
                  rx_buffer1[index]=data;
                  putchar(rx_buffer1[index]);
                  index++;
                  if (index>16)  rx_buffer_overflow1=1;
            };
            ptr++;
      }
}
```

Στο τέλος της λειτουργίας του interrupt για ένα σήμα του GPS πχ: `$GPGGA,151254.585,3520.0727,N,02507.3299,E,1,04,4.1,78.8,M,,,,0000*30`
ο rx_buffer1 θα έχει μορφή:

2	0	.	0	7	2	0	7	.	3	2	9	7	8	.	8	,

Η συνάρτηση startNMEA καλείται στην αρχή του προγράμματος και σκοπό έχει την αρχικοποίηση του GPS. Το string που στέλνεται στο GPS δίνει εντολή για την μετάδοση δεδομένων σε NMEA πρωτόκολλο αντί για Sirf που είναι το προκαθορισμένο. Μια καθυστέρηση τριών δευτερολέπτων είναι απαραίτητη έτσι ώστε να γίνει σωστά η λήψη των χαρακτήρων. Κατά τη διάρκεια λειτουργίας της συνάρτησης τα interrupt είναι απενεργοποιημένα.

```
void startNMEA(void)
{
 unsigned char i;
 unsigned char InitGPS[32]={  0xA0,0xA2,0x00,0x18,0x81,0x02,
                              0x01,0x01,0x00,0x01,0x00,0x01,
                              0x00,0x01,0x00,0x01,0x00,0x01,
                              0x00,0x01,0x00,0x01,0x00,0x01,
                              0x00,0x01,0x4B,0x00,0x00,0xD9,
                              0xB0,0xB3  };

 #asm("cli")
 for(i=0;i<32;i++)
 putchar1(InitGPS[i]);
 lcd_putsf("NMEA string send\nwaiting...");
 putsf("NMEA string send\nwaiting...");
 delay_ms(3000);
 #asm("sei")
}
```

Στο πρόγραμμα χρησιμοποιούνται κάποιες global μεταβλητές έτσι ώστε να είναι εύκολη η διαχείρισή τους από τις συναρτήσεις. Οι πιο σημαντικές είναι:

- `float D,F;` απόσταση, απόκλιση από τη θεωρητική πορεία
- `eeprom unsigned int RouteGPS[5][2]={{0,0},{0,0},{0,0},{0,0},{0,0}};`

Μεταβλητή στην eeprom του μικροελεγκτή. Περιέχει τα σημεία της θεωρητικής πορείας. Ενημερώνεται από πρόγραμμα στον υπολογιστή. Τα σημεία παραμένουν στον μικροελεγκτή και μετά από reset του συστήματος.

- `float RouteF[4],RouteCosF[4],RouteSinF[4];`

Μεταβλητές όπως υπολογίζονται από την RouteCalculations() στην αρχή του προγράμματος. RouteF οι κλίσεις των ευθύγραμμων τμημάτων της θεωρητικής πορείας και RouteCosF RouteSinF τα συνημίτονα και ημίτονα των κλίσεων αυτών.

- `unsigned int Point1[2]={4620,35970};`
  `unsigned int Point2[2]={4520,34970};`
  `unsigned int Point3[2]={3720,37360};`

global μεταβλητές με τα τρία τελευταία σημεία του GPS. Η αρχικοποίηση είναι αυθαίρετη.

- `unsigned int latitude, longitude, height;`

global μεταβλητές για τις τιμές του τελευταίου σήματος GPS

Η συνάρτηση CalcPosition() φιλτράρει τον πίνακα rx_buffer1 και αποδίδει τις κατάλληλες τιμές στις μεταβλητές latitude, longitude, height. Καλείται κάθε φορά που λαμβάνεται ένα νέο σήμα GPS.

```
void CalcPosition(void)
{
      unsigned char charLat[6],charLon[6];
      int ic,ii=0;

      for (ic=1;ic<7;ic++)                //προσπέλαση του rx_buffer1
      {
            if (ic==3)continue;
            charLon[ii]=rx_buffer1[ic+6]; //προσωρινές μεταβλητές
            charLat[ii]=rx_buffer1[ic];
            ii++;
      }
      charLat[5]='\0';              //προσθήκη τερματικού χαρακτήρα
      charLon[5]='\0';              //στις προσωρινές μεταβλητές

      latitude =(atol(charLat)*5)/3;      //μετατροπή σε αριθμό και σε
      longitude=(atol(charLon)*5)/3;      //μορφή δεκαδικών μοιρών

      lcd_clear();                        //εμφάνιση στην LCD
      lcd_gotoxy(0,0);
      lcd_putsf("lat:");
      lcd_print_uint(latitude);
      lcd_putsf(" l:");

      ic=0;
      height=0;
      while(rx_buffer1[ic+13]!='.') //φιλτράρισμα της τιμής του ύψους
      {
            lcd_putchar(rx_buffer1[ic+13]);
            if ((rx_buffer1[ic+13]!=',')&&(rx_buffer1[ic+13]!='-'))
                  height=(unsigned int)height*10+rx_buffer1[ic+13]-48;
            ic++;
            if (ic>=4) break;
      }
      lcd_gotoxy(0,1);                    //εμφάνιση στην LCD
      lcd_putsf("lon:");
      lcd_print_uint(longitude);

      printf("latitude:%u ",latitude);    //εμφάνιση των τελικών τιμών
      printf("longitude:%u ",longitude);  //στην σειριακή προς τον
      printf("height:%u ",height);        //υπολογιστή
}
```

Το βασικό loop του προγράμματος παρουσιάζεται παρακάτω. Στην αρχή αρχικοποιούνται οι επιφάνειες ελέγχου σε μηδενική θέση. Καλείται η RouteCalculations για την εύρεση των πρώτων υπολογισμών της θεωρητικής πορείας. Στη συνέχεια γίνεται κυκλική ροή του κώδικα και διακόπτεται μόνο από τα interrupt. Αν το bit του rx_buffer_overflow1 έχει τιμή 1 τότε σημαίνει ότι ένα νέο σημείο GPS έχει ληφθεί. Αν το bit του operationFlag έχει τιμή 0 τότε σημαίνει ότι το σύστημα δεν βρίσκεται σε λειτουργία επικοινωνίας με τον υπολογιστή. Αν ισχύουν αυτές οι δύο συνθήκες τότε το σύστημα προχωρά στον βασικό έλεγχο. Αρχικά απενεργοποιούνται τα interrupt έτσι ώστε να μη διακοπεί η λειτουργία. Εκτελείται η CalcPosition() η οποία υπολογίζει τις τιμές του GPS. Οι τιμές αυτές οδηγούνται προς αποθήκευση στην εξωτερική eeprom με χρήση της συνάρτησης toEeprom(longitude, latitude, height). Αφού αποθηκευτούν προσωρινά τα τρία τελευταία σημεία καλείται η GPSCalculations η οποία έχει όρισμα τον αριθμό του ευθύγραμμου τμήματος και υπολογίζει την απόκλιση και την απόσταση από το τμήμα αυτό της πορείας. Αμέσως μετά εκτελείται η εντολή για να κινηθούν τα aileron του αερομοντέλου σύμφωνα με τις τιμές των F και D. Στη συνέχεια καλείται η pathControl, επαναφέρεται το flag του GPS interrupt rx_buffer_overflow1 στο 0 και ενεργοποιούνται ξανά όλα τα interrupt.

```
elevator(0);
throtle(0);
eleron(0,0);
rader(0,0);
RouteCalculations();
while (1)
      {
        if ((rx_buffer_overflow1==1)&&(operationFlag==0)){
                  #asm("cli")             //απενεργοποίηση interrupt
                  CalcPosition();
                  toEeprom(longitude, latitude, height);

                  Point1[0]=Point2[0];
                  Point1[1]=Point2[1];
                  Point2[0]=Point3[0];
                  Point2[1]=Point3[1];
                  Point3[0]=longitude;
                  Point3[1]=latitude;

                  GPSCalculations(route);
                  eleron(F,D);
                  route=pathControl(route);
                  rx_buffer_overflow1=0;
                  #asm("sei")             //ενεργοποίηση interrupt
        }
      };
```

Η συνάρτηση pathControl βρίσκει εφαρμογή μέσα στο βασικό loop και σκοπό έχει να διορθώσει την μεταβλητή αναφοράς route. Έχει σαν όρισμα την προηγούμενη μεταβλητή route, δηλαδή το ευθύγραμμο τμήμα της θεωρητικής πορείας. Ελέγχει αν το τρέχων σημείο GPS βρίσκεται σε απόσταση μικρότερη 30 μέτρων από την αρχή του επόμενου τμήματος και αν ισχύει τότε η route αυξάνεται. Με αυτό τον τρόπο γίνεται ο έλεγχος της πορείας με πρόβλεψη επανεκκίνησης αν τα σημεία προσπελαστούν όλα. Το όλο σύστημα θα μπορούσε να έχει αρκετά σημεία για θεωρητική πορεία αλλά στο πλαίσιο αυτής της πτυχιακής εργασίας περιοριστήκαμε στα πέντε και επομένως στα τέσσερα ευθύγραμμα τμήματα. Εδώ φαίνεται και ο λόγος για τον οποίο αν το αερομοντέλο δεν προσεγγίσει σωστά το επόμενο σημείο τότε θα συνεχίσει να ακολουθεί την πρώτη πορεία.

```
char pathControl(char route)
{

      float DX,DY,D;

      #pragma warn-//εντολή στο Compiler για παράβλεψη προειδοποίησης
      DX=(float)(Point3[0]-(float)RouteGPS[route+1][0]);
      DY=(float)(Point3[1]-(float)RouteGPS[route+1][1]);
      D=sqrt(DX*DX+DY*DY);

      if(D<30)
      {
            if(route==3)
                  route=0;
            else
                  if(RouteF[route+1]!=0)
                        route=route+1;
                  else
                        route=0;
      }
      #pragma warn-
      return route;


}
```

Η τελική συνάρτηση eleron(float Fr,float Dr) καλείται κάθε φορά που πρέπει να κινηθούν τα aileron του αερομοντέλου. Έχει σαν ορίσματα της τιμές της απόστασης και της απόκλισης από την πορεία και υπολογίζει την εντολή ελέγχου $P = K_F \cdot F + K_D \cdot D$. Δηλώνεται όπως και στις άλλες συναρτήσεις ελέγχου το κέντρο της επιφάνειας (ELERON_CENTER), η μέγιστη δυνατή απόκλιση από το κέντρο (eleronDC) και οι σταθερές eleronKF και eleronKD του νόμου ελέγχου. Κατά τον υπολογισμό της P αν η τιμή είναι μεγαλύτερη από τη μέγιστη δυνατή κλίση των aileron που έχει οριστεί τότε η

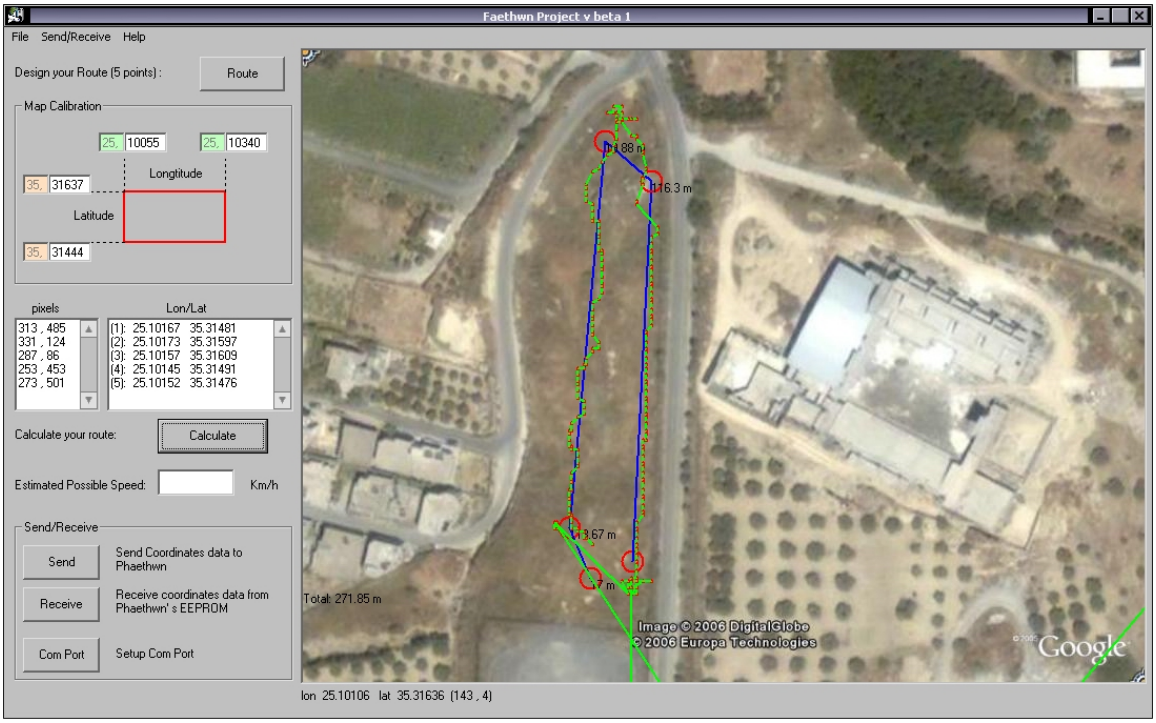
κλίση γίνεται ίση με τη μέγιστη. Εκτελείται η κίνηση στέλνοντας τους χαρακτήρες 0xff 0x01 Ptel στο Servo Controller και παραμένει για δύο δευτερόλεπτα πριν ευθυγραμμιστούν και πάλι. Όταν ευθυγραμμιστούν τα aileron τότε αυτόματα μπαίνει σε λειτουργία ο αισθητήρας ορίζοντα εφόσον “αντιλαμβάνεται” ότι τα χειριστήρια δεν ελέγχονται και επαναφέρει το αερομοντέλο σε επίπεδο πτήσης (level flight). Με αυτό τον τρόπο η επόμενη εντολή που θα δοθεί στα aileron από τον αλγόριθμο θα είναι σε θέση να εκτελεστεί σωστά αφού το αερομοντέλο θα βρίσκεται σε οριζόντια μηδενική θέση.

```
void eleron(float Fr,float Dr)
{
      char ELERON_CENTER=0x83;            //κέντρο των aileron
      #define eleronKF 17                 //σταθερά για την απόκλιση
      #define eleronKD 0.50               //σταθερά για την απόσταση
      #define eleronDC 0x0a               //μέγιστη κλίση aileron

      signed char P;
      float Ptemp;
      unsigned char Ptel;

      Ptemp=eleronKF*Fr+eleronKD*Dr;      //υπολογισμός εντολής P
      if(Ptemp>=eleronDC)                 //έλεγχος επιτρεπτών ορίων
            P=eleronDC;
      else
            if(Ptemp<-eleronDC)
                  P=-eleronDC;
            else
                  P=(signed char)Ptemp;

      Ptel=(unsigned char)ELERON_CENTER+P;
      printf(" %d ",ELERON_CENTER);       //τύπωση κέντρου και τελικής
      printf(" %d ",P);                   //κλίσης στη σειριακή του
      printf(" %d ",Ptel);                //υπολογιστή

      putchar(0xff);                      //εκτέλεση μέσω Servo
      putchar(0x01);                      //Controller
      putchar(Ptel);

      delay_ms(2000);                     //καθυστέρηση

      putchar(0xff);                      //επαναφορά aileron σε
      putchar(0x01);                      //αρχική θέση
      putchar(ELERON_CENTER);
}
```

Ο κώδικας που γράφηκε για τον μικροελεγκτή παρατίθεται στο παράρτημα Α ολοκληρωμένος καθώς και στο CD της πτυχιακής το project για το Codevision με όλα τα αρχεία.

6.2 Faethwn.exe

Το πρόγραμμα Faethwn.exe υλοποιήθηκε με χρήση της γλώσσας προγραμματισμού Visual Basic στο περιβάλλον Microsoft Visual Studio 6.0. Η εύκολη χρήση της γλώσσας καθώς και οι μικρές απαιτήσεις του σκοπού για τον οποίο χρησιμοποιήθηκε, ήταν τα κριτήρια για την επιλογή της. Η ιδέα του προγράμματος βασίστηκε σε ένα ήδη καθιερωμένο το οποίο χρησιμοποιείται ως εμπορική εφαρμογή για αυτόνομα αεροχήματα[1]. Στο πρόγραμμα αυτό ο χρήστης είναι σε θέση να απεικονίσει ένα χάρτη σε μορφή *.jpeg 800x600 pixels συνοδευόμενο από ένα αρχείο *.map το οποίο να περιέχει τις συντεταγμένες των δύο γωνιών του. Οι συντεταγμένες πρέπει να είναι σε μορφή δεκαδικών μοιρών και το όνομα των αρχείων *.jpeg και *.map να είναι ίδιο. Αυτόματα στην πάνω αριστερή γωνία εμφανίζονται οι συντεταγμένες του χάρτη που έχει ανοιχτεί. Πατώντας το κουμπί Route ο χρήστης είναι σε θέση να σχεδιάσει μια πορεία από πέντε σημεία. Για να τερματιστεί η διαδικασία χρειάζεται να πατηθεί το ίδιο κουμπί με την ετικέτα Finish πλέον. Πατώντας Calculate γίνονται οι απαραίτητοι υπολογισμοί και ενεργοποιείται το κουμπί Send.



Το πρόγραμμα κάνει χρήση ενός αρκετά χρήσιμου ActiveX control, το MScomm32 με το οποίο μπορούμε να διαχειριστούμε τις σειριακές ενός υπολογιστή. Για να λειτουργήσει σωστά το πρόγραμμα θα πρέπει αν δεν έχουν εγκατασταθεί τα βασικά αρχεία VB6 KB290887 να μεταφερθεί στον φάκελο συστήματος System32 το αρχείο mscomm32.ocx το οποίο περιλαμβάνεται στο CD που συνοδεύει την αναφορά.

[1] http://www.dgadv.com/aerotracker/

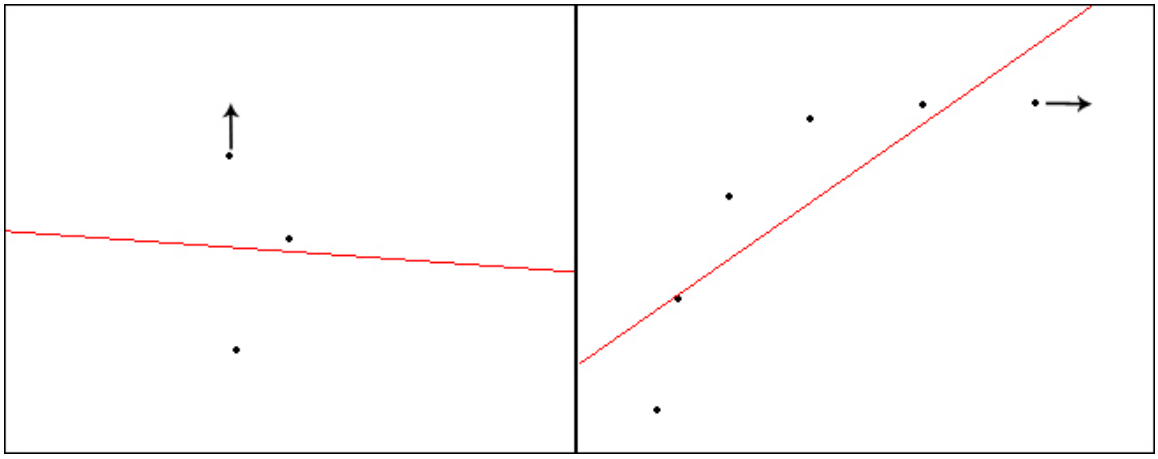
Από το κουμπί Com Port του προγράμματος είμαστε σε θέση να επιλέξουμε την επιθυμητή σειριακή θύρα και μετά με το κουμπί Send να στείλουμε την νέα πορεία στο αερομοντέλο. Η διαδικασία απαιτεί το καλώδιο το οποίο έχει φτιαχτεί για αυτό το σκοπό και είναι τύπου null modem με ακροδέκτη στερεοφωνικού βύσματος προς το αερομοντέλο. Η πορεία θα αποθηκευτεί στην eeprom του μικροελεγκτή και για να ισχύσει απαιτείται ένα reset του συστήματος. Από το κουμπί Receive έχουμε την επιλογή να λάβουμε στον υπολογιστή τα δεδομένα όλης της εξωτερικής μνήμης eeprom που έχουν αποθηκευτεί τα σημεία GPS σε ένα αρχείο κειμένου με κατάληξη *.fth. Με επεξεργασία σε κάποιο πρόγραμμα όπως το Excel μπορούμε να εισάγουμε τα σημεία αυτά στο πρόγραμμα μ-Center της ίδιας εταιρίας με το GPS module. Από την επιλογή File>Open *.fth μπορούμε να φορτώσουμε το αρχείο αυτό επάνω σε κάποιο χάρτη και να δούμε τα σημεία από τα οποία πέρασε το αερομοντέλο. Μια άλλη λειτουργία είναι στο Send/Receive>Read Route όπου μπορούμε να διαβάσουμε την πορεία που έχει αποθηκευτεί μια δεδομένη στιγμή στο αερομοντέλο. Σε ένα νέο παράθυρο εμφανίζεται η πορεία με δυνατότητα εμφάνισης στο χάρτη, αν αυτός υπάρχει. Το πεδίο Estimated Possible Speed είναι σε δοκιμαστικό στάδιο και προσφέρει τη δυνατότητα να προσομοιώσει σημεία σε μια πορεία βάσει της ταχύτητας του αερομοντέλου.

Ο κώδικας που δημιουργήθηκε για το πρόγραμμα παρατίθεται στο παράρτημα Β ολοκληρωμένος καθώς και στο CD της πτυχιακής όλο το Visual Basic project για περαιτέρω διόρθωση και επεξεργασία. Επίσης στο CD περιλαμβάνεται και ένα project για έναν απλό και εύχρηστο terminal που χρησιμοποιήθηκε κατά την σειριακή μετάδοση δεδομένων από το κύκλωμα.

7 Συμπεράσματα

7.1 Πειραματικά αποτελέσματα

Στις πειραματικές δοκιμές που πραγματοποιήθηκαν αναθεωρήθηκαν πολλά μέρη της εργασίας με σκοπό την καλύτερη συμπεριφορά του συστήματος. Ένα από αυτά ήταν η μαθηματική μέθοδος των ελαχίστων τετραγώνων. Η μέθοδος έπαιρνε μέρος στην εύρεση του διανύσματος κίνησης από τα τελευταία τρία στίγματα του GPS. Ένα μεγάλο μειονέκτημα το οποίο εντοπίστηκε κατά τις δοκιμές ήταν όταν το σύστημα μετατοπιζόταν βόρεια ή νότια με τα σημεία του GPS να ισαπέχουν λόγω της σταθερής ταχύτητας. Η μέθοδος σε αυτή τη περίπτωση παράγει ευθεία σχεδόν κάθετη στην ευθεία κίνησης του αερομοντέλου με αποτέλεσμα το διάνυσμα και η γωνία απόκλισης να παρεκκλίνουν αρκετά από τα πραγματικά δεδομένα. Ένα παράδειγμα μπορεί να βρεθεί στη σχετική ιστοσελίδα[1]. Η μέθοδος δουλεύει σωστά για περισσότερα σημεία αλλά με αυτό τον τρόπο θα χανόταν η ακρίβεια του διανύσματος.



Για το λόγο αυτό η μέθοδος ελαχίστων τετραγώνων αφαιρέθηκε από τον νόμο ελέγχου και το διάνυσμα προσεγγίστηκε από τα δύο τελευταία σημεία του GPS. Αποτέλεσμα αυτού βέβαια ήταν λανθασμένα σήματα να επηρεάζουν αρκετά τη συμπεριφορά αφού δεν εξομαλύνεται πια το διάνυσμα κίνησης με κάποια μέθοδο όπως η προηγούμενη.

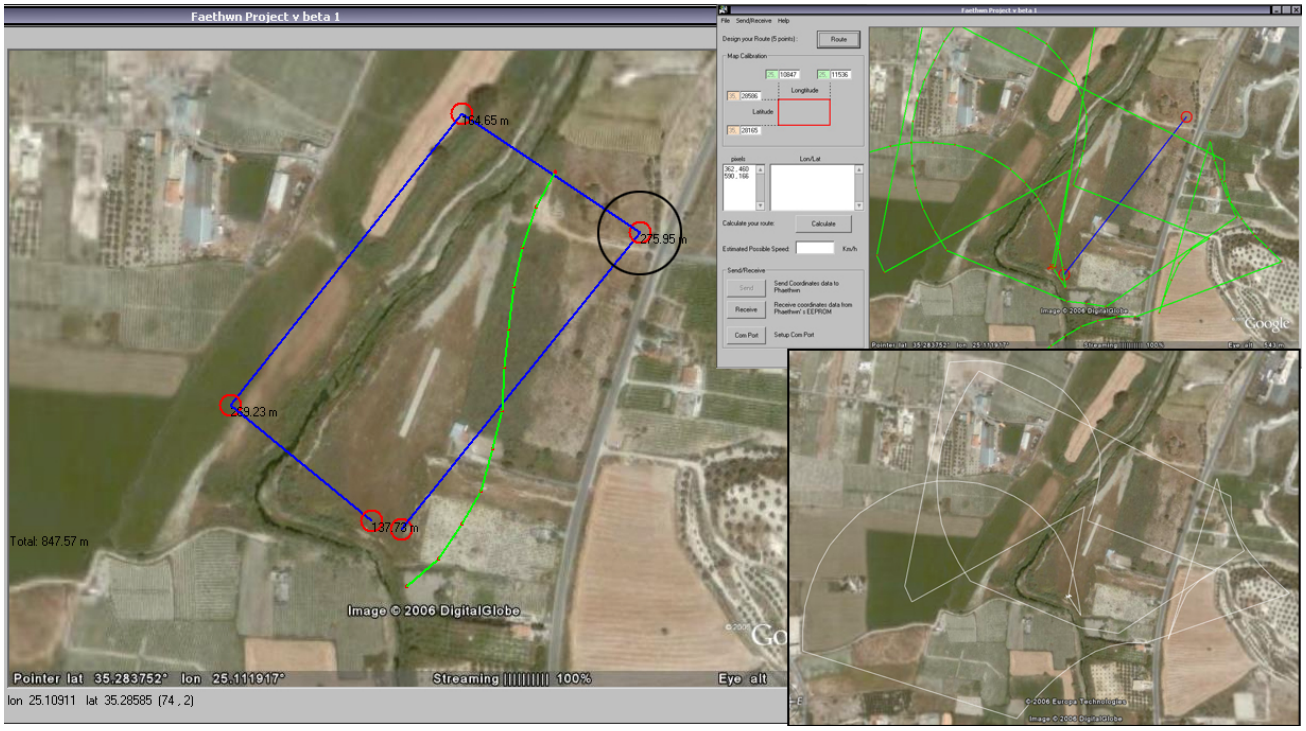
Η ηλεκτρονική πυξίδα δεν χρησιμοποιήθηκε κατά τον έλεγχο γιατί παρουσίασε λανθασμένες ενδείξεις κατά την κλίση της σε διάφορες γωνίες. Αυτό γίνεται γιατί επηρεάζεται από το βάρος ο μαγνητικός μηχανισμός που περιέχει με αποτέλεσμα να συμπεριφέρεται σωστά μόνο σε επίπεδη θέση.

Το μεγαλύτερο πρόβλημα το παρουσίασε ο αισθητήρας ορίζοντα ο οποίος κρίθηκε ακατάλληλος για την επαναφορά του αερομοντέλου σε επίπεδο πτήσης. Μετά από πολλές δοκιμές που έγιναν διαπιστώθηκε πως το HAL ανάτρεπε αρκετά τη πτητική

[1] http://www.barrodale.com/java_demo/javademo.htm

συμπεριφορά του αερομοντέλου και πάντα με διαφορετικό τρόπο ανάλογα τις καιρικές συνθήκες. Σε μια συννεφιασμένη ημέρα το σύστημα αργούσε αρκετά να επανέλθει. Μεγάλο ρόλο έπαιξε και η χρονική στιγμή καθώς σε περίπτωση που ο ήλιος δεν ήταν κάθετα όπως το μεσημέρι το σύστημα έτεινε να μπει σε πορεία προς αυτόν. Επίσης μετά από κάποιο χρονικό διάστημα παρατηρήθηκε απώλεια ύψους του αερομοντέλου με αποτέλεσμα τη συνεχή διακοπή του ρομποτικού αλγόριθμου μέσω του κυκλώματος εναλλαγής. Όλες οι μετρήσεις που πήραμε ήταν για διάστημα το πολύ 20 δευτερολέπτων λόγω της απότομης ανατροπής της σταθερότητας του ύψους από τον αισθητήρα ορίζοντα. Παρόλα αυτά ο νόμος ελέγχου ήταν αρκετά σωστός αν και δεν δόθηκε η δυνατότητα να τον παρακολουθήσουμε σε μια ολοκληρωμένη πορεία λόγω του παραπάνω προβλήματος.

Οι σταθερές K_D και K_Θ του νόμου ελέγχου αρχικά ρυθμίστηκαν στο 0.2 και 20 αντίστοιχα. Με διαδοχικές δοκιμές και σταδιακές αλλαγές οι τιμές διαμορφώθηκαν στο 0.5 και 17 αντίστοιχα προσφέροντας όσο πιο πολύ γίνεται την καλύτερη συμπεριφορά επαναφοράς του αερομοντέλου στην πορεία. Το μοναδικό μειονέκτημα που παρατηρήθηκε ως προς τον νόμο ήταν ότι επηρεαζόταν η σταθεροποίηση ανάλογα του σημείου έναρξης του ρομποτικού ελέγχου. Για παράδειγμα, εάν το αερομοντέλο έμπαινε σε ρομποτική λειτουργία σε μικρή απόσταση από την θεωρητική πορεία, τότε η πορεία που ακολουθούσε ήταν σωστή και με μικρές παρεκκλίσεις. Σε αντίθετη περίπτωση δημιουργούταν πορεία με ταλάντωση γύρω από την επιθυμητή. Αποτέλεσμα αυτού ήταν να μη μπορέσει να προσεγγίσει το τελικό σημείο της ευθείας σε λιγότερο από 30 μέτρα και τελικά να μην μπορέσει να μεταβεί στο επόμενο ευθύγραμμο τμήμα προς ολοκλήρωση της πορείας.



Επίσης μια πολύ σημαντική διόρθωση που έγινε ήταν να αυξηθεί ο χρόνος παραμονής της κλίσης του αερομοντέλου με αντίστοιχη μείωση αυτής. Αρχικά η μέγιστη κλίση των aileron είχε ρυθμιστεί στο 0x1e και με χρόνο παραμονής λιγότερο από 500 msec έτσι ώστε το αερομοντέλο να επανέλθει στον υπόλοιπο μισό χρόνο πριν την επόμενη ένδειξη GPS. Η συμπεριφορά ήταν σχετικά σωστή αλλά όχι τόσο ομαλή αφού κάθε δευτερόλεπτο γίνονταν πολλές διορθώσεις που είχαν ως μειονέκτημα τη σταθερότητα. Η μέγιστη κλίση μίκρυνε στο 0x0a ενώ ο χρόνος παραμονής στην κλίση αυτή μεγάλωσε στα δύο δευτερόλεπτα διαμορφώνοντας μια συμπεριφορά πιο ομαλή ως προς την διόρθωση της πορείας.

7.2 Προτάσεις

Τελειώνοντας, σε αυτό το σημείο θεωρήθηκε σκόπιμο για την καλύτερη εξέλιξη του συστήματος η αναφορά κάποιων προτάσεων.

Το μεγαλύτερο μειονέκτημα του συστήματος που μελετήθηκε και κατασκευάστηκε ήταν ο αισθητήρας ορίζοντα για την επαναφορά του αερομοντέλου. Κάτι τέτοιο μπορεί να επιτευχθεί με τη χρήση δύο γυροσκοπίων και με καλύτερα μάλιστα αποτελέσματα. Κάθε ένα μπορεί να τοποθετηθεί έτσι ώστε να παρακολουθείται η μεταβολή του επιπέδου στους δύο βασικούς άξονες του αεροπλάνου (δεξιά-αριστερά, πάνω-κάτω). Ακόμα αντί αυτού μπορεί να χρησιμοποιηθεί ένα μοντελιστικό gyro το οποίο είναι αρκετά αξιόπιστο και συνήθως βρίσκει εφαρμογή στην εξισορρόπηση του rudder ενός μοντελιστικού ελικοπτέρου.

Επίσης η κατασκευή ενός κλειστού κυκλώματος στο εσωτερικό του αερομοντέλου θα διευκόλυνε αρκετά στις δοκιμαστικές πτήσεις όπου πολλά καλώδια και επιμέρους κυκλώματα δεν διατρέχουν τον κίνδυνο αποκόλλησης και της ανάγκης συνεχούς επίβλεψης.

Ως προς τον νόμο ελέγχου για την καλύτερη σταθεροποίηση του αερομοντέλου σε μια πορεία θα ήταν αποτελεσματικότερο να γίνει με PID. Οι τρεις όροι του PID ελέγχου τείνουν ανάλογα τις σταθερές του αναλογικού, του ολοκληρωτικού και του παραγωγικού όρου να εξομαλύνουν τη συμπεριφορά ενός ελέγχου στην καλύτερη δυνατή. Ένα παράδειγμα δίνεται από τη θεωρία χειρισμού του εξομοιωτή πτήσης FlightGear[1].

[1] http://www.flightgear.org/Docs/XMLAutopilot/XMLAutopilot.html

Βιβλιογραφία

- Προγράμματα που χρησιμοποιήθηκαν:
  - CodeVisionAVR http://www.hpinfotech.ro/
  - Eagle http://www.cadsoft.de/
  - Proteus VSM http://www.labcenter.co.uk/index_uk.htm
  - Google Earth http://earth.google.com/
  - Servo http://www.superrobotica.com/VisualSC2.htm
  - uCenter http://www.u-blox.com/products/u_center.html
  - Sirf demo
- Εξομοιωτές πτήσης:
  - FMS http://n.ethz.ch/student/mmoeller/fms/index_e.html
  - FlightGear http://www.flightgear.org/
- Αερομοντελισμός
  - http://jkon.aeromodelling.gr/
  - http://egnatia.ee.auth.gr/%7Ekosmour/index.htm
- LCD http://ouwehand.net/~peter/lcd/lcd0.shtml#general
- Compass http://www.robot-electronics.co.uk/htm/cmps3doc.shtml
- I2C http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm
- Eagle Tutorial http://www.interq.or.jp/japan/se-inoue/e_eagle.htm
- https://www.aaai.org/AITopics/html/autveh.html#air
- http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html

Παραρτημα Α

Κώδικας c για τον μικροελεγκτή ATmega162 γραμμένος στο CodeVisionAVR
Αρχείο main.c

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.5 Standard
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project :
Version :
Date    : 17/5/2006
Author  : Fodan
Company : Home
Comments:

Chip type           : ATmega162
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega162.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>
#include <i2c.h>
#include <eepromFAETHON.h>
#include <controlFAETHON.h>
#include <lcdFAETHON.h>
#include <lcd.h>
#include <string.h>
#include <ctype.h>
// I2C Bus functions
#asm
   .equ __i2c_port=0x18 ;PORTB
   .equ __sda_bit=0
   .equ __scl_bit=1
#endasm

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm

#define pi 3.1415902654
unsigned int swover;
float D,F;
char points;
eeprom unsigned int RouteGPS[5][2]=
{{0,0},{0,0},{0,0},{0,0},{0,0}};
float RouteF[4],RouteCosF[4],RouteSinF[4];
unsigned int Point1[2]={4620,35970};
unsigned int Point2[2]={4520,34970};
unsigned int Point3[2]={3720,37360};
unsigned int latitude,longitude,height;
bit operationFlag;
bit auto;

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

void saveRoute(unsigned int,char);
void Show(void);
void RouteCalculations(void);
void GPSCalculations(char );
void CalcPosition(void);
char pathControl(char route);

// External Interrupt 0 service routine

interrupt [EXT_INT0] void ext_int0_isr(void)
{
swover=0;
TCCR0=0x02;
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
TCCR0=0x00;
        if (swover>=10){
        PORTC=0x03;
        auto=0;
        }
        else{
        auto=1;
        PORTC=0x00;
        }
}
```

```
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
swover++;
}

/*-----------------------------------------------------*
*Interrupt gia thn Seriakh 0(PC):
*
* To receive Interrupt ekteleite kathe fora pou stalei
* enas xalakthras sth seriakh 0.
* O algoritmos einai tropopoihmenos gia ehn epikoinvnia
* me to programma ston H/Y.
*-----------------------------------------------------*/
// USART0 Receiver buffer
#define RX_BUFFER_SIZE0 8
char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0<256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;

// USART0 Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
char data;
static bit flagin,pointflag;
static char index,ptr;
unsigned int temp;

data=UDR0;
  if (data=='!') {
      #asm("cli")
      putchar('!');
      delay_ms(10);
      Show();
      delay_ms(10);
      putchar('&');
      #asm("sei")
  };
  if (data=='#') {
      #asm("cli")
      fromEeprom();
      #asm("sei")
  };
  if(flagin==1){
       if (index<5){
               rx_buffer0[index]=data;
               index++;
       }
       if (index>=5){
               temp=atol(rx_buffer0);
               saveRoute(temp,ptr);
               ptr++;
               if(ptr>9) {
                       printf("*");
                       flagin=0;
                       #asm("cli")
                       RouteCalculations();
                       #asm("sei")
                       operationFlag=0;
               }
               else{
                        delay_ms(10);
                        printf("%d",ptr);
                        index=0;
               }
        }
  };
  if (pointflag==1){
         flagin=1;
         pointflag=0;
         points=data;
         delay_ms(100);
         putchar('0');
         delay_ms(100);
  };
  if (data=='@'){
       operationFlag=1;
       pointflag=1;
       index=0;
       ptr=0;
       delay_ms(100);
       putchar('@');
       delay_ms(100);
 };
}

/*-----------------------------------------------------*
*Interrupt gia thn Seriakh 1(GPS):
*
* To receive Interrupt ekteleite kathe 1 sec opou
* lambanontai oi xarakthres apo to GPS pou einai
* syndedemeno sth Seriakh 1.
* O algoritmos apothikeyei kathe fora ston rx_buffer1[]
* longtitude,latitude kai yspos me th seira.
*-----------------------------------------------------*/
// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 17
char rx_buffer1[RX_BUFFER_SIZE1];

#if RX_BUFFER_SIZE1<256
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
#else
unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
#endif

// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;


// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
```

```
{
if (auto==1)
{
char data;
  static unsigned char ptr,index;
  data=UDR1;
  if (data=='$') {
        ptr=0;
        index=0;
        rx_buffer1[index]=data;
        putchar(rx_buffer1[index]);
        index=1;
  };
  if
(((ptr>19)&&(ptr<26))||((ptr>32)&&(ptr<39))||((ptr>51)&
&(ptr<56))){
    rx_buffer1[index]=data;
    putchar(rx_buffer1[index]);
    index++;
    if (index>16)  rx_buffer_overflow1=1;
  };
  ptr++;
}
}
// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
// Standard Input/Output functions
// Declare your global variables here
static unsigned char k;
/*-----------------------------------------------------*
* FUNCTION: startNMEA()
*
* Stelnei to InitGPS[] string sto GPS gia thn
* arxikopoihsh tou se NMEA protokollo
*-----------------------------------------------------*/
 void startNMEA(void)
{
 unsigned char InitGPS[32]={
0xA0,0xA2,0x00,0x18,0x81,0x02,
0x01,0x01,0x00,0x01,0x00,0x01,
0x00,0x01,0x00,0x01,0x00,0x01,
0x00,0x01,0x00,0x01,0x00,0x01,
0x00,0x01,0x4B,0x00,0x00,0xD9,0xB0,0xB3};

 unsigned char i;
 #asm("cli")
 for(i=0;i<32;i++)
 putchar1(InitGPS[i]);
 lcd_putsf("NMEA string send\nwaiting...");
 putsf("NMEA string send\nwaiting...");
 delay_ms(3000);
 #asm("sei")
}
/*-----------------------------------------------------*
* FUNCTION: CalcPosition()
*
* Exei ws eisodo ton global pinaka rx_buffer1[] kai
* xexwrizei kai ypologizei se unsigned int tis global
* metavlites latitude,longitude,height
*-----------------------------------------------------*/
void CalcPosition(void)
{
 unsigned char charLat[6],charLon[6];
 int ic,ii=0;

 for (ic=1;ic<7;ic++)
 {
  if (ic==3)continue;
  charLon[ii]=rx_buffer1[ic+6];
  charLat[ii]=rx_buffer1[ic];
  ii++;
 }
 charLat[5]='\0';
 charLon[5]='\0';
 latitude =(atol(charLat)*5)/3;
 longitude=(atol(charLon)*5)/3;

 lcd_clear();
 lcd_gotoxy(0,0);
 lcd_putsf("lat:");
 lcd_print_uint(latitude);
 lcd_putsf(" l:");

 ic=0;
 height=0;


 while(rx_buffer1[ic+13]!='.')
 {
 lcd_putchar(rx_buffer1[ic+13]);
   if ((rx_buffer1[ic+13]!=',')&&(rx_buffer1[ic+13]!='-
')){
   height=(unsigned int)height*10+rx_buffer1[ic+13]-48;
   }
  ic++;
  if (ic>=4) break;
 }
 lcd_gotoxy(0,1);
 lcd_putsf("lon:");
 lcd_print_uint(longitude);

 printf("latitude:%u ",latitude);
 printf("longitude:%u ",longitude);
 printf("height:%u ",height);
}
```

```
/*=====================================================*
*
*                    MAIN FUNCTION
*
*=====================================================*/
void main(void)
{
// Declare your local variables here
  int i;
  char route=0;
// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

//broun out detector
if (MCUCSR & 1)
   {
   MCUCSR&=0xE0;
   }
else if (MCUCSR & 2)
   {
   MCUCSR&=0xE0;
   }
else if (MCUCSR & 4)
   {
   // Brown-Out Reset
   MCUCSR&=0xE0;
   // Place your code here
x: goto x;
   }
else if (MCUCSR & 8)
   {
   MCUCSR&=0xE0;
   }
else if (MCUCSR & 0x10)
   {
   MCUCSR&=0xE0;
   };

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Port E initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
```

```
// Timer/Counter 3 initialization
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: On
// INT1 Mode: Falling Edge
// INT2: Off
// Interrupt on any change on pins PCINT0-7: Off
// Interrupt on any change on pins PCINT8-15: Off
GICR|=0xC0;
MCUCR=0x0B;
EMCUCR=0x00;
GIFR=0xC0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x02;
ETIMSK=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x86;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud rate: 19200
UCSR1A=0x00;
UCSR1B=0x98;
UCSR1C=0x86;
UBRR1H=0x00;
UBRR1L=0x33;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1:
Off
ACSR=0x80;

// I2C Bus initialization
i2c_init();

// LCD module initialization
lcd_init(16);

lcdshow(5,0,"FAETHON");
printf("FAETHON");

delay_ms(1500);

if((RouteGPS[0][0]==0)&&(RouteGPS[0][1]==0)){
        lcd_clear();
        delay_ms(500);
        lcd_putsf("   !No Route!");
        for(i=0;i<2;i++){
        delay_ms(500);
        _lcd_ready();
        _lcd_write_data(0x08);
        _lcd_ready();
        delay_ms(500);
        _lcd_write_data(0x0c);
        _lcd_ready();    }
        delay_ms(1500);
 }
 else{
        lcd_clear();
        delay_ms(500);
        lcd_putsf("     Route!");
        delay_ms(1000);
 }
  lcd_clear();

// Global enable interrupts
#asm("sei")

startNMEA();
//eraseEeprom();

elevator(0);
throtle(0);
eleron(0,0);
rader(0,0);
RouteCalculations();
GPSCalculations(0);
```

```
while (1)
      {
      if((rx_buffer_overflow1==1)&&(operationFlag==0)){
                  #asm("cli")
                  CalcPosition();
                  toEeprom(longitude,latitude,height);

                  //Point1[0]=Point2[0];
                  //Point1[1]=Point2[1];

                  Point2[0]=Point3[0];
                  Point2[1]=Point3[1];

                  Point3[0]=longitude;
                  Point3[1]=latitude;

                  GPSCalculations(route);
                  eleron(F,D);
                  putchar(13);
                  route=pathControl(route);
                  lcd_gotoxy(0,8);
                  lcd_putsf(" path:");
                  lcd_putchar(route+48);
                  rx_buffer_overflow1=0;
                  #asm("sei")
        }
      };
}
/*-----------------------------------------------------*
* FUNCTION: pathControl ()
*
* Kaleite kathe fora sto vasiko loop
* ypologgizei thn apostash apo to trexwn shmeio
* kai analogws epistrefei to swsto eu8ygrammo tmhma
* ths poreias pou prepei na akolou8y8uei
*-----------------------------------------------------*/
char pathControl(char route){

float DX,DY,D;


   #pragma warn-
   DX=(float)(Point3[0]-(float)RouteGPS[route+1][0]);
   DY=(float)(Point3[1]-(float)RouteGPS[route+1][1]);
   D=sqrt(DX*DX+DY*DY);

   if(D<30){
     if(route==3)
             route=0;
     else
             if(RouteF[route+1]!=0)
                     route=route+1;
             else
                     route=0;
   }
   #pragma warn-
   return route;
}
```

```
/*-----------------------------------------------------*
* FUNCTION: saveRoute()
*
* Kaleite apo to interrupt 1 gia na apothikeysei ta
* shmeia ths programmatizomenhs poreias pou lambanetai
* apo to PC.
* Ta shmeia apothikeyonatai sth eeprom global metavliti
* RouteGPS[][]
*-----------------------------------------------------*/
void saveRoute(unsigned int num,char index)
{
  if (index<5)RouteGPS[index][0]=num;
  if (index>=5)RouteGPS[index-5][1]=num;
}
/*-----------------------------------------------------*
* FUNCTION: Show()
*
* Kaleite apo to interrupt 1 gia na emfanisei ta shmeia
* ths poreias sto PC pou vriskontai sthn eeprom global
* metavliti RouteGPS[][]
*-----------------------------------------------------*/
void Show(void){
  char r;
   for(r=0;r<5;r++)
  printf(" (%u,%u) ",RouteGPS[r][0],RouteGPS[r][1]);
}
/*-----------------------------------------------------*
* FUNCTION: RouteCalculations()
*
* Kaleite sthn arxh i kathe fora pou lambanetai nea
* poreia
* Ypologizei apo thn global eeprom metavliti
* RouteGPS[][]
*     - thn klish kathe eythygrammou tmhmatos RouteF[]
*     - to synimhtono ths kathe klishs RouteCosF[]
*     - to hmitono ths kathe klishs RouteSinF[]
*-----------------------------------------------------*/
void RouteCalculations(void)
{
unsigned int x1,x2,y1,y2;
int tx,ty;
char i;
        for (i=0;i<4;i++){
                x1=RouteGPS[i][0];    //long1
                y1=RouteGPS[i][1];    //lat1
                #pragma warn-
                x2=RouteGPS[i+1][0];  //long2
                y2=RouteGPS[i+1][1];  //lat2
                #pragma warn+
                if ((x2==0)||(y2==0)){
                        RouteF[i]=0;
                        RouteCosF[i]=0;
                        RouteSinF[i]=0;
                }
                else{   ty=y2-y1;
                        tx=x2-x1;
                        RouteF[i]=(float)atan2(ty,tx);
                        RouteCosF[i]=cos(RouteF[i]);
                        RouteSinF[i]=sin(RouteF[i]);
```

```
                };
        };
}
/*-----------------------------------------------------*
* FUNCTION: GPSCalculations()
*
* Kaleite kathe fora pou lambanetai neo shmeio GPS
* Ypologizei:
*     - th methodo elaxistwn tetragwnwn
*     - th gwnia apoklishs apo th poreia sth global F
*     - thn apostash apo th poreia sth global D
*-----------------------------------------------------*/
void GPSCalculations(char route)
{
static float t,d;
int tx,ty;

if((Point3[0]!=Point2[0])||(Point3[1]!=Point2[1])){
if(RouteF[route]!=0){

 tx=Point3[0]-Point2[0];
 ty=Point3[1]-Point2[1];
 t=(float)atan2(ty,tx);

 t= RouteF[route]- t;

 if(t>pi)
        t=t-2*pi;
 else
        if(t<-pi)
                t=t+2*pi;

 d=(float)(Point3[1]-
(float)RouteGPS[route][1])*RouteCosF[route]-
(float)(Point3[0]-
(float)RouteGPS[route][0])*RouteSinF[route];
 //d=-d;

}
else{
        t=0;
        d=0;
};
}
else{
        t=0;
        d=0;
};

 F=t;
 D=-d;
 printf(" (%.4f,%.4f) ",F,D);
```

Αρχείο controlfaethon.h

```
void rader(float Fr,float Dr);
void elevator(signed char);
void throtle(signed char);
void eleron(float Fr,float Dr);
```

Αρχείο controlfaethon.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>
#include <lcdFAETHON.h>
/*-----------------------------------------------------*
* FUNCTION: rader()(xrhsh mono gia dokimh edafous)
*
* Exei eisodo thn gwnia apoklishs kai thn apostash apo
* thn poreia.
* Ypologizei thn synarthsh P=KF*F+KD*D me K1,K2
* sygekrimena gia to Rader.
* Stelnei sto Servo Controller thn entolh kinhshs tou
* Rader.
*-----------------------------------------------------*/
void rader(float Fr,float Dr)
{
#define RADER_CENTER 0x79
#define raderKF 10
#define raderKD 0.054
#define raderDC 0x4F
signed char P;
float Ptemp;
unsigned char Ptel;

Ptemp=raderKF*Fr+raderKD*Dr;
if(Ptemp>=raderDC)
        P=raderDC;
else
        if(Ptemp<-raderDC)
                P=-raderDC;
        else
                P=(signed char)Ptemp;

Ptel=(unsigned char)RADER_CENTER+P;
printf(" %d ",RADER_CENTER);
printf(" %d ",P);
printf(" %d ",Ptel);
```

```
putchar(0xff);
putchar(0x04);
putchar(Ptel);
}
/*-----------------------------------------------------*
* FUNCTION: elevator()
*       Under Constraction!!
*-----------------------------------------------------*/
void elevator(signed char move)
{
char ELEVATOR_CENTER=0x79;

putchar(0xff);
putchar(0x02);
putchar(ELEVATOR_CENTER+move);
}
/*-----------------------------------------------------*
* FUNCTION: throtle()
*       Under Constraction!!
*-----------------------------------------------------*/
void throtle(signed char move)
{
char THROTLE_CENTER=0x93;

putchar(0xff);
putchar(0x03);
putchar(THROTLE_CENTER+move);
}
/*-----------------------------------------------------*
* FUNCTION: eleron()
*
* Exei eisodo thn gwnia apoklishs kai thn apostash apo
* thn poreia.
* Ypologizei thn synarthsh P=KF*F+KD*D me K1,K2
* sygekrimena gia ta Eleron.
* Stelnei sto Servo Controller thn entolh kinhshs tou
* Eleron.
*-----------------------------------------------------*/
void eleron(float Fr,float Dr)
{
char ELERON_CENTER=0x83;
#define eleronKF 17
#define eleronKD 0.50
#define eleronDC 0x0a


signed char P;
float Ptemp;
unsigned char Ptel;

Ptemp=eleronKF*Fr+eleronKD*Dr;
if(Ptemp>=eleronDC)
        P=eleronDC;
else
        if(Ptemp<-eleronDC)
                P=-eleronDC;
        else
                P=(signed char)Ptemp;

Ptel=(unsigned char)ELERON_CENTER+P;
printf(" %d ",ELERON_CENTER);
printf(" %d ",P);
printf(" %d ",Ptel);

putchar(0xff);
putchar(0x01);
putchar(Ptel);

delay_ms(2000);

putchar(0xff);
putchar(0x01);
putchar(ELERON_CENTER);
}
```

Αρχείο eepromfaethon.h

```
void lcdprog(unsigned char l,unsigned int cur,unsigned int min, unsigned int max);
void Eread(unsigned char chip_addr,unsigned int address);
void Ewrite(unsigned char chip_addr,unsigned int address);
void toEeprom(unsigned int lat,unsigned int lon,unsigned int hei);
void fromEeprom(void);
void eraseEeprom(void);
```

Αρχείο eepromfaethon.c

```
#include <i2c.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>
#include <lcd.h>

#define EEPROM0 0xa0
#define EEPROM1 0xa2

unsigned int inputdata;  //metabliti gia to I2C
```

```
/*-----------------------------------------------------*
* FUNCTION: Eread()
*
* Diavazei apo to Eeprom Chip me dieythinsi
* "chip_addr".
* H prospelash ths mnhmhs ginetai synexwmena(sequential
* read) arxizonats apo th dieythnsi "address".
* Oi times typononται me katallhlo tropo etsi wste na
* diavastoun apo to programma ston H/Y.
*-----------------------------------------------------*/
void Eread(unsigned char chip_addr,unsigned int
address)
{
 unsigned int i,c;
 i2c_start();
 i2c_write(chip_addr);
 i2c_write ((unsigned char)(address*256));
 i2c_write ((unsigned char)address);
 i2c_start();
 i2c_write(chip_addr|1);
 for (c=0;c<4094;c++) {
        i=((i2c_read(1)*256));
        i+=i2c_read(1);
        lcdprog(chip_addr,c,1,4094);
        printf("%u",i);
        delay_ms(15);
        putchar('&');
        delay_ms(5);
 }
 i=((i2c_read(1)*256));
 i+=i2c_read(0);
 i2c_stop();
 delay_ms(10);
 printf("%u",i);
 delay_ms(10);
 putchar('&');
 delay_ms(100);
}
/*-----------------------------------------------------*
* FUNCTION: Ewrite()
*
* Grafei sto Eeprom Chip me dieythinsi "chip_addr" sth
* dieythnsi "address" ton global unsigned int
* "inputdata" xwrizontas ton se 2 byte
*-----------------------------------------------------*/
void Ewrite(unsigned char chip_addr,unsigned int
address)
{
i2c_start();
i2c_write(chip_addr);
i2c_write (address/256);
i2c_write ((unsigned char) address);
i2c_write((unsigned char)(inputdata>>8));
i2c_stop();
delay_ms(10);
address+=1;
i2c_start();
i2c_write(chip_addr);
i2c_write (address/256);
i2c_write ((unsigned char) address);
i2c_write((unsigned char)inputdata);
i2c_stop();
delay_ms(10);
}

/*-----------------------------------------------------*
* FUNCTION: toEeprom()
*
* Kanei xrhsh ths synarthshs Ewrite().
* Diaxeirizetai to grapsimo twn metavlitvn lat,lon,hei
* sta eeprom chip metavainontas apo to 1o chip sto 2o.
*-----------------------------------------------------*/
void toEeprom(unsigned int lat,unsigned int
lon,unsigned int hei)
{
 static unsigned int memAddress,ptr1;
 if (memAddress<=1364)   //euros memaddress gia eeprom0
 {
  inputdata=lat;
  Ewrite(EEPROM0,ptr1);  //kataxorhsh metavlhtwn
  ptr1+=2;
  inputdata=lon;
  Ewrite(EEPROM0,ptr1);
  ptr1+=2;
  inputdata=hei;
  Ewrite(EEPROM0,ptr1);
  ptr1+=2;
  memAddress++;
 }
 else
 if ((memAddress>=1365)&&(memAddress<=2729))
                         //euros memaddress gia eeprom1
 {
   inputdata=lat;
   Ewrite(EEPROM1,ptr1-8190);   //kataxwrhsh metavlhtwn
   ptr1+=2;
   inputdata=lon;
   Ewrite(EEPROM1,ptr1-8190);
   ptr1+=2;
   inputdata=hei;
   Ewrite(EEPROM1,ptr1-8190);
   ptr1+=2;
   memAddress++;
 }
  else
  {
   lcd_putsf("memory full");
   printf("memory full!");    //memory full.do nothing
  };
 }

/*-----------------------------------------------------*
* FUNCTION: fromEeprom()
*
* Kanei xrhsh ths synarthshs Eread().
* Diaxeirizetai to diavasma twn timwn apo ta eeprom
* chip metavainontas apo to 1o chip sto 2o.
*-----------------------------------------------------*/
```

```
void fromEeprom(void)
{
  delay_ms(100);
  printf("#");
  delay_ms(100);
  printf("#");
  delay_ms(100);
 Eread(EEPROM0,0x0000);
 Eread(EEPROM1,0x0000);
 putchar('*');
}



/*-----------------------------------------------------*
* FUNCTION: eraseEeprom()
*
* Proeretikh synarthsh pou diagrafei oola ta dedmena
* ths eeprom gia na mhn yparxoun dedomena apo
* prohgoumenh pthsh
*-----------------------------------------------------*/
void eraseEeprom(void)
{
 unsigned int i;
 for(i=0;i<8192;i++){
        i2c_start();
        i2c_write(EEPROM0);
        i2c_write(i/256);
        i2c_write((unsigned char)i);
        i2c_write(0x00);
        i2c_stop();
        delay_ms(10);
        i2c_start();
        i2c_write(EEPROM1);
        i2c_write(i/256);
        i2c_write((unsigned char)i);
        i2c_write(0x00);
        i2c_stop();
        delay_ms(10);
        printf("%u,",i);
 }
 printf("memory erased");
 lcd_clear();
 lcd_putsf("memory erased");
}
```

Αρχείο lcdfaethon.h

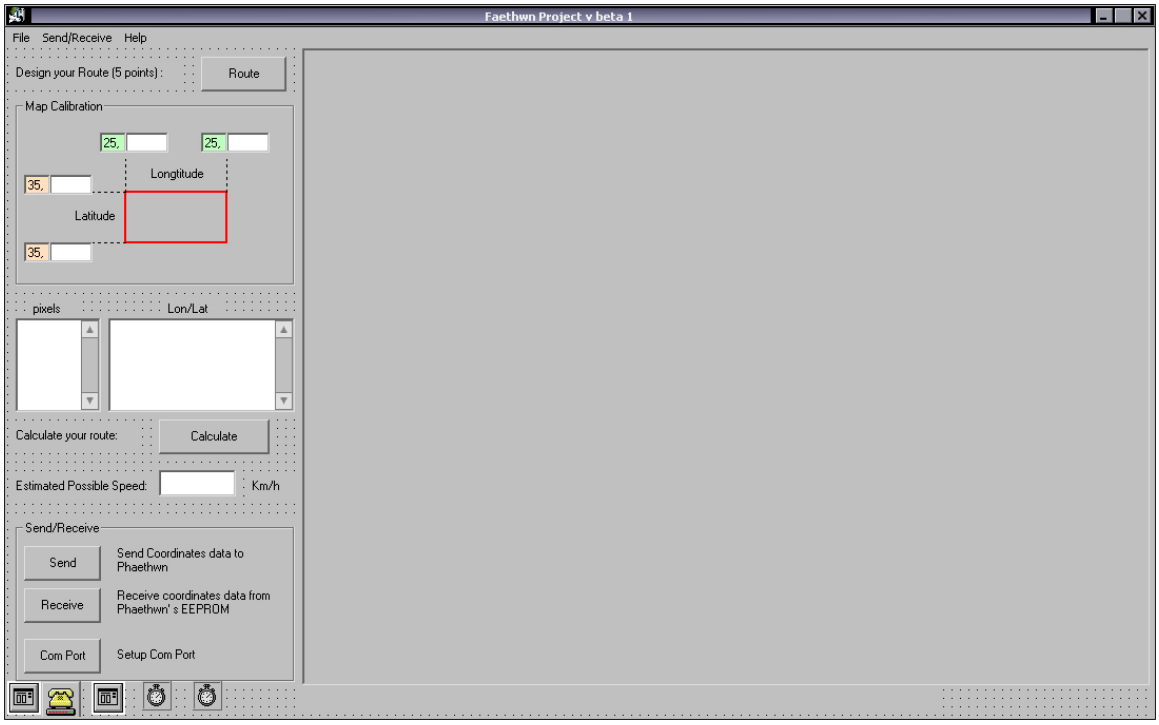
```
void lcdshow(int x,int y,char flash *l);
void lcdprog(unsigned char l,unsigned int cur,unsigned int min, unsigned int max);
void lcd_print_uint(unsigned int x);
```

Αρχείο lcdfaethon.c

```
#include <lcd.h>
#include <string.h>
#include <delay.h>
#include <math.h>

void lcdshow(int x,int y,char flash *l){
char i;
unsigned int k,s;
   s=strlenf(l);
   for(k=0;k<s;k++){
   for(i=0;i<3;i++){
   lcd_gotoxy(x,y);
   lcd_putchar('|');
   delay_ms(50);
   lcd_gotoxy(x,y);
   lcd_putchar('-');
   delay_ms(50);
   lcd_gotoxy(x,y);
   lcd_putchar('/');
   delay_ms(50);
   }
   lcd_gotoxy(x,y);
   lcd_putchar(*l++);
   x++;
   }
}
void lcdprog(unsigned char l,unsigned int cur,unsigned
int min, unsigned int max){
char k;


      lcd_clear();
      if(l==0xa0)
      lcd_putsf("Transferring ep1");
      else if(l==0xa2)
                lcd_putsf("Transferring ep2");
      lcd_gotoxy(0,1);
      for(k=0;k<abs(cur/((max-min)/16));k++){

      lcd_putchar(0xff);
                     }
}
void lcd_print_uint(unsigned int x)
{
unsigned int y;
y=x/10000;lcd_putchar(y+0x30);x-=(y*10000);
y=x/1000;lcd_putchar(y+0x30);x-=(y*1000);
y=x/100;lcd_putchar(y+0x30);x-=(y*100);
y=x/10;lcd_putchar(y+0x30);x-=(y*10);
lcd_putchar(x+0x30);}
```

Παράρτημα Β

Κώδικας VB για το πρόγραμμα Faethwn.exe γραμμένος στο VisualStudio6.0
Form1:




```
Dim DrawNow As Boolean
Dim X1 As Integer
Dim Y1 As Integer
Dim PX(20) As Integer
Dim PY(20) As Integer
Dim GPSX(20) As Double
Dim GPSY(20) As Double
Dim i As Integer
Dim Points As Integer
Dim Status As String
Dim pixX As Double
Dim pixY As Double
Dim apost(20) As Double
Dim total As Double
Dim lonx As Double
Dim laty As Double

Dim flag As String
Dim counter As Integer
Dim FName As String
Dim inBuffer As String
Dim del As Long
```

```
Private Sub About_Click()
frmAbout.Show vbModal
End Sub
```

```
Private Sub com_Click()
Call Command5_Click
End Sub
```

```
Private Sub Command1_Click()
If Command1.Caption = "Route" Then

    For r = 0 To 5
        GPSX(r) = 0
        GPSY(r) = 0
    Next
    Picture1.Cls
    Text1.Text = ""
    Text8.Text = ""
    X1 = 0
    Y1 = 0
    i = 0
    Picture1.MousePointer = 2
    DrawNow = True
    Command1.Caption = "Finish"
```

```
    Command2.Enabled = False
    Command3.Enabled = False
    send.Enabled = False
    Exit Sub
End If

If Command1.Caption = "Finish" Then
    DrawNow = False
    Picture1.MousePointer = 0
    Command1.Caption = "Route"
    Points = i
    Command2.Enabled = True
    If Points = 1 Then
        Points = 0
        MsgBox ("Only 1 point is not a route!!")
        GPSX(0) = 0
        GPSY(0) = 0
        Picture1.Cls
        Text1.Text = ""
        Text8.Text = ""
        X1 = 0
        Y1 = 0
        i = 0
        Command2.Enabled = False
    End If
End If
End Sub
```

```
Private Sub Command2_Click()
If Text2.Text <> "" Or Text3.Text <> "" Or Text4.Text
<> "" Or Text7.Text <> "" Then
Text8.Text = ""
MapXGPS = Abs(Int(Text4.Text) - Int(Text2.Text))
MapYGPS = Abs(Int(Text7.Text) - Int(Text3.Text))

lonx = ("25" & Text2.Text)
laty = ("35" & Text7.Text)
pixX = MapXGPS / 800
pixY = MapYGPS / 600
For r = 0 To Points - 1
GPSX(r) = pixX * PX(r) + lonx
GPSY(r) = pixY * (600 - PY(r)) + laty
Text8.Text = Text8.Text & "(" & r + 1 & "):  " &
FormatNumber((GPSX(r) / 100000), 5) & "   " &
FormatNumber((GPSY(r) / 100000), 5) & vbNewLine
Next

RedrawPic
Command3.Enabled = True
send.Enabled = True
Else
MsgBox ("No map, no points to Calculate!")
End If
End Sub
```

```
Private Sub Command3_Click()
On Error Resume Next
MSComm1.PortOpen = True
If Err Then
    MsgBox Error$, 48
    Exit Sub
End If
 MSComm1.Output = "@"
 Timer1.Enabled = True
 Command4.Enabled = False
 Command5.Enabled = False
 receive.Enabled = False
 Readroute.Enabled = False
 com.Enabled = False
End Sub
```

```
Private Sub Command4_Click()
On Error GoTo Errorhandler

With CommonDialog2
        .DialogTitle = "Save"
        .CancelError = True
        .Filter = "All Files (*.*)|*.*"
        .Filter = "fth (*.fth)|*.fth"
        .ShowSave
        If Len(.FileName) = 0 Then
            Exit Sub
        End If
        FName = .FileName
    End With

On Error Resume Next
MSComm1.PortOpen = True
If Err Then
    MsgBox Error$, 48
    Exit Sub
End If

    MSComm1.Output = "#"
    Timer1.Enabled = True
Open FName For Output As #1
Exit Sub
Errorhandler:
If Err = vbCancel Then
Exit Sub
End If
End Sub
```

```
Private Sub Command5_Click()
frmProperties.Show vbModal
End Sub
```

```
Private Sub exit_Click()
End
End Sub
```

```
Private Sub Form_Load()
Command2.Enabled = False
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
End Sub
```

```
Private Sub fth_Click()
Dim data As String
If Text2.Text <> "" Or Text3.Text <> "" Or Text4.Text
<> "" Or Text7.Text <> "" Then
MapXGPS = Abs(Int(Text4.Text) - Int(Text2.Text))
MapYGPS = Abs(Int(Text7.Text) - Int(Text3.Text))
pixX = MapXGPS / 800
pixY = MapYGPS / 600

    With CommonDialog1
        .DialogTitle = "Open"
        .CancelError = False
        .Filter = "All Files (*.*)|*.*"
        .Filter = "fth (*.fth)|*.fth"
        .ShowOpen
        If Len(.FileName) = 0 Then
            Exit Sub
        End If
        FName = .FileName
    End With

    Open FName For Input As #1
    Do Until EOF(1)
    Line Input #1, data
        temp1 = Mid(data, 1, 5)
        temp2 = Mid(data, 15, 5)
        If temp1 <> "" Then
            lon = Int(temp1)
        Else
            lon = 0
        End If

        If temp2 <> "" Then
            lat = Int(temp2)
        Else
            lat = 0
        End If
        If lon <> 0 And lat <> 0 Then
            lonpix = (lon - Text2.Text) / pixX
            latpix = 600 - ((lat - Text7.Text) / pixY)
            asd = asd + 1
            If asd = 1 Then
                lonpix1 = lonpix
                latpix1 = latpix
            End If

Picture1.Line (lonpix1, latpix1)-(lonpix, latpix),
RGB(0, 255, 0)
Picture1.Circle (lonpix, latpix), 1, RGB(255, 0, 0)
lonpix1 = lonpix
latpix1 = latpix
        End If
Loop
Close #1

lonx = ("25" & Text2.Text)
laty = ("35" & Text7.Text)
Else
MsgBox ("No map to print on!!")
End If
End Sub
```

```
Private Sub MSComm1_OnComm()
Dim InBuff As String

         If MSComm1.CommEvent = comEvReceive Then
               InBuff = MSComm1.Input
               Call ParseChars(InBuff)
            End If
End Sub
```

```
Private Sub Open_Click()
Dim sFile As String
Dim objPic As Picture

    With CommonDialog1
        .DialogTitle = "Open"
        .CancelError = False
        .Filter = "All Files (*.*)|*.*"
        .Filter = "Jpg (*.jpg)|*.jpg"
        .ShowOpen
        If Len(.FileName) = 0 Then
            Exit Sub
        End If
        tFile = .FileTitle
        sFile = .FileName
    End With
    Path = sFile
    Label1 = Path

    map = Left(Path, Len(Path) - 3)
    map = map & "map"

    Set objPic = LoadPicture(Path)
If objPic.Height = 15875 And objPic.Width = 21167 Then
    On Error Resume Next
    Open map For Input As #5
    If Err Then
        MsgBox ("Calibration file not found (*.map)")
        Exit Sub
    End If

    Line Input #5, mapdata
Text3.Text = Mid(mapdata, 1, InStr(1, mapdata, ",")- 1)
Text2.Text = Mid(mapdata, InStr(1, mapdata, ",") + 1,
Len(mapdata))
Line Input #5, mapdata
Text7.Text = Mid(mapdata, 1, InStr(1, mapdata, ",") -
1)
Text4.Text = Mid(mapdata, InStr(1, mapdata, ",") + 1,
Len(mapdata))
```

```
    Close #5
    Picture1.Picture = LoadPicture(Path)
Else
    MsgBox ("Only 800x600 jpeg images are supported!")
End If
End Sub
```

```
Private Sub Picture1_MouseDown(Button As Integer, Shift
As Integer, x As Single, Y As Single)
leftDown = (Button And vbLeftButton) > 0

If leftDown Then
If Status <> "edit" Then
If DrawNow Then
If X1 <> 0 And Y1 <> 0 Then
     Picture1.Line (X1, Y1)-(x, Y), RGB(0, 0, 255)
End If

Picture1.Circle (x, Y), 10, RGB(255, 0, 0)
Text1.Text = Text1.Text & x & " , " & Y & vbNewLine
PX(i) = x
PY(i) = Y
i = i + 1

If i >= 5 Then  'Περιορισμός για 5 σημεία.(20 max)
    MsgBox ("Sorry! Only 5 Points are supported!")
    DrawNow = False
    Picture1.MousePointer = 0
    Command1.Caption = "Route"
    Points = i
    Command2.Enabled = True
End If

X1 = x
Y1 = Y
End If
End If
End If
End Sub
```

```
Private Sub Picture1_MouseMove(Button As Integer, Shift
As Integer, x As Single, Y As Single)

leftDown = (Button And vbLeftButton) > 0
Label1 = "lon  " & FormatNumber((pixX * x + lonx) /
100000, 5) & "   lat  " & FormatNumber((pixY * (600 -
Y) + laty) / 100000, 5) & "  (" & x & " , " & Y & ")"
If Points <> 0 Then
    For w = 0 To Points - 1
        If x < PX(w) + 10 And x > PX(w) - 10 And Y <
PY(w) + 10 And Y > PY(w) - 10 Then
            Picture1.MousePointer = 15
            Label1 = Label1 + " Point:" & w + 1
            Exit For
        Else
            Picture1.MousePointer = 2
Label1 = "lon  " & FormatNumber((pixX * x + lonx) /
100000, 5) & "   lat  " & FormatNumber((pixY * (600 -
Y) + laty) / 100000, 5) & "  (" & x & " , " & Y & ")"
    End If
    Next
End If
End Sub
```

```
Private Sub Readroute_Click()
On Error Resume Next
MSComm1.PortOpen = True
If Err Then
    MsgBox Error$, 48
    Exit Sub
End If
    MSComm1.Output = "!"
    Timer1.Enabled = True
End Sub
```

```
Private Sub receive_Click()
Call Command4_Click
End Sub
```

```
Private Sub send_Click()
Call Command3_Click
End Sub
```

```
Private Sub Text1_Change()
Text1.SelStart = Len(Text1.Text)
If Mid(Text1.Text, 1, 2) = vbCrLf Then
    Text1.Text = Mid(Text1.Text, 3, Len(Text1.Text))
End If
End Sub
```

```
Private Sub Text8_Change()
Text8.SelStart = Len(Text8.Text)
If Mid(Text8.Text, 1, 2) = vbCrLf Then
    Text8.Text = Mid(Text8.Text, 3, Len(Text8.Text))
End If
End Sub
```

```
Private Sub Text9_Change()
Dim i, A, B, G, diak, Xok, Yok, dok, d, u As Long

If (Text2.Text <> "" Or Text3.Text <> "" Or Text4.Text
<> "" Or Text7.Text <> "") And apost(0) <> 0 Then
If Text9.Text = "" Then
    RedrawPic
ElseIf IsNumeric(Text9.Text) Then
    RedrawPic
    u = Abs(Int(Text9.Text))

    If u <> 0 Then
        d = u * 5 / 18
        dok = d
        j = 0
        i = 0

        Do While (True)
            If dok < apost(j) Then
d1 = (GPSY(j + 1) - GPSY(j)) / (GPSX(j + 1) - GPSX(j))
b1 = GPSY(j) - d1 * GPSX(j)
```

```
A = 1 + d1 ^ 2
B = 2 * d1 * (b1 - GPSY(j)) - 2 * GPSX(j)
G = GPSX(j) ^ 2 - dok ^ 2 + (b1 - GPSY(j)) ^ 2
                diak = B ^ 2 - 4 * A * G

                If GPSX(j) < GPSX(j + 1) Then
                    Xok = (-B + Sqr(diak)) / (2 * A)
                    Yok = d1 * Xok + b1
Picture1.Circle (Int((Xok - lonx) / pixX), Int(600 -
(Yok - laty) / pixY)), 5
                ElseIf GPSX(j) > GPSX(j + 1) Then
                    Xok = (-B - Sqr(diak)) / (2 * A)
                    Yok = d1 * Xok + b1
Picture1.Circle (Int((Xok - lonx) / pixX), Int(600 -
(Yok - laty) / pixY)), 5
                End If

                dok = dok + d
                i = i + 1
                If i = Int(total / d) Then
                    Exit Do
                End If

            ElseIf dok > apost(j) Then
                dok = dok - apost(j)
                j = j + 1
            End If
        Loop
    End If
Else
    RedrawPic
End If
Else
MsgBox ("Calculation of a route required!")
End If
End Sub
```

```
Private Sub RedrawPic()
total = 0
Picture1.Cls
Picture1.Circle (PX(0), PY(0)), 10, RGB(255, 0, 0)
For i = 0 To Points - 2
Picture1.Line (PX(i), PY(i))-(PX(i + 1), PY(i + 1)),
RGB(0, 0, 255)
Picture1.Circle (PX(i + 1), PY(i + 1)), 10, RGB(255, 0,
0)
apost(i) = Sqr((GPSY(i + 1) - GPSY(i)) ^ 2 + (GPSX(i +
1) - GPSX(i)) ^ 2)
total = total + apost(i)
Picture1.Print Round(apost(i), 2) & " m"
    Next
Picture1.Print "Total: " & Round(total, 2) & " m"
End Sub
```

```
Sub ParseChars(ByVal InString As String)
         Dim Temp As String
         Dim x As Long

If flag = "send" Then
         For x = 0 To 10
            Temp = Str(x)
            If " " & InString = Temp Then
                Call Delay
                If x <= 4 Then
                    If (GPSX(x) <> 0) Then
         MSComm1.Output = Mid(Str(GPSX(x)), 4, 5)
                    Else
                        MSComm1.Output = "00000"
                    End If
                Else
                    If (GPSY(x - 5) <> 0) Then
         MSComm1.Output = Mid(Str(GPSY(x - 5)), 4, 5)
                    Else
                        MSComm1.Output = "00000"
                    End If
                End If
            End If
         Next x
End If

If flag = "show" Then
            If InString <> "!" And InString <> "&" Then
                inBuffer = inBuffer & InString
            End If
            If InString = "&" Then
                flag = ""
                MSComm1.PortOpen = False
                Form3.Text1.Text = ""
                Form3.Text1.Text = inBuffer
                Form3.Show vbModal
                inBuffer = ""
            End If
End If

If flag = "receive" Then
        If InString <> "*" And InString <> "#" Then
            inBuffer = inBuffer & InString
            If InString = "&" Then
                counter = counter + 1

                Select Case counter Mod 3
                Case 1
Print #1, Mid(inBuffer, 1, Len(inBuffer) - 1); Tab;
                Case 2
Print #1, Mid(inBuffer, 1, Len(inBuffer) - 1); Tab;
                Case 0
Print #1, Mid(inBuffer, 1, Len(inBuffer) - 1)
                End Select

                inBuffer = ""
                If counter <= 16378 Then
                    Form2.ProgressBar1.Value = counter
                End If
            End If
        End If
End If

        If InString = "!" Then
```

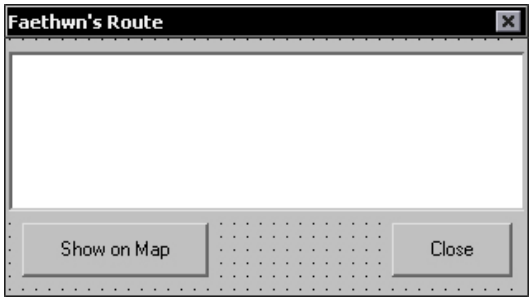
```
    flag = "show"
    Timer1.Enabled = False
End If

If InString = "@" Then
    flag = "send"
    Timer1.Enabled = False
    Call Delay
    MSComm1.Output = Mid(Str(Points), 2)
    Call Delay
End If

If InString = "#" Then
    flag = "receive"
    counter = 0
    Timer1.Enabled = False
    If Form2.Visible = False Then
        Form2.Show vbModal
    End If
End If

If InString = "*" Then
    flag = "end"
    MSComm1.PortOpen = False
    Unload Form2
    counter = 0
    Close #1
    Command4.Enabled = True
    Command5.Enabled = True
    receive.Enabled = True
    Readroute.Enabled = True
            com.Enabled = True
            MsgBox ("Operation completed succesfully!")
End If
End Sub
```

```
Private Sub Timer1_Timer()
Timer1.Enabled = False
MsgBox ("Faethwn is not responding!!!")
flag = "end"
MSComm1.PortOpen = False
Unload Form2
counter = 0
Close #1
Command4.Enabled = True
Command5.Enabled = True
receive.Enabled = True
Readroute.Enabled = True
com.Enabled = True
End Sub
```

```
Private Sub Delay()
del = 0
Do
dummy = DoEvents()
Loop Until (del = 2)

End Sub
```

```
Private Sub Timer2_Timer()
del = del + 1
End Sub
```

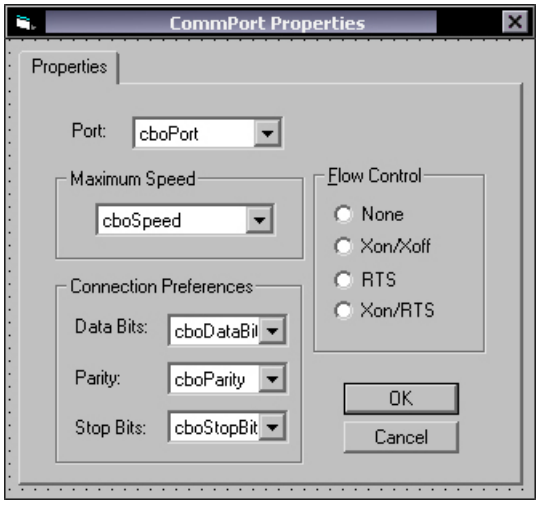
Form3:




```
Private Sub Command1_Click()
Unload Me
End Sub
```

```
Private Sub Command2_Click()

MapXGPS = Abs(Int(Form1.Text4.Text) -
Int(Form1.Text2.Text))
MapYGPS = Abs(Int(Form1.Text7.Text) -
Int(Form1.Text3.Text))
pixX = MapXGPS / 800
pixY = MapYGPS / 600

Temp = Text1.Text
For i = 0 To 4

A = InStr(1, Temp, "(")
B = InStr(1, Temp, ",")
c = InStr(1, Temp, ")")

If (A = 0) Or (B = 0) Or (c = 0) Then
 Exit For
 End If

 lon = Int(Mid(Temp, A + 1, B - A - 1))
 lat = Int(Mid(Temp, B + 1, c - B - 1))
If lon <> 0 And lat <> 0 Then

lonpix = (lon - Form1.Text2.Text) / pixX
latpix = 600 - ((lat - Form1.Text7.Text) / pixY)

            asd = asd + 1
            If asd = 1 Then
                lonpix1 = lonpix
                latpix1 = latpix
            Form1.Picture1.Circle (lonpix, latpix), 4

            End If

        Form1.Picture1.Line (lonpix1, latpix1)-(lonpix,
latpix), RGB(255, 128, 255)
        lonpix1 = lonpix
        latpix1 = latpix
 End If


Temp = Mid(Temp, c + 1, Len(Temp) - c)
Next
Unload Me
End Sub
```

frmProperties(frmProps):




```
Private iFlow As Integer, iTempEcho As Boolean
```

```
Sub LoadPropertySettings()
Dim i As Integer, Settings As String, Offset As Integer

' Load Port Settings
For i = 1 To 16
    cboPort.AddItem "Com" & Trim$(Str$(i))
Next i

' Load Speed Settings
cboSpeed.AddItem "110"
cboSpeed.AddItem "300"
cboSpeed.AddItem "600"
cboSpeed.AddItem "1200"
cboSpeed.AddItem "2400"
cboSpeed.AddItem "4800"
cboSpeed.AddItem "9600"
cboSpeed.AddItem "14400"
cboSpeed.AddItem "19200"
cboSpeed.AddItem "28800"
cboSpeed.AddItem "38400"
cboSpeed.AddItem "56000"
cboSpeed.AddItem "57600"
cboSpeed.AddItem "115200"
cboSpeed.AddItem "128000"
cboSpeed.AddItem "256000"

' Load Data Bit Settings
cboDataBits.AddItem "4"
cboDataBits.AddItem "5"
cboDataBits.AddItem "6"
cboDataBits.AddItem "7"
cboDataBits.AddItem "8"

' Load Parity Settings
cboParity.AddItem "Even"
cboParity.AddItem "Odd"
cboParity.AddItem "None"
cboParity.AddItem "Mark"
cboParity.AddItem "Space"

' Load Stop Bit Settings
cboStopBits.AddItem "1"
cboStopBits.AddItem "1.5"
cboStopBits.AddItem "2"

' Set Default Settings

Settings = Form1.MSComm1.Settings

If InStr(Settings, ".") > 0 Then
    Offset = 2
Else
    Offset = 0
End If

cboSpeed.Text = Left$(Settings, Len(Settings) - 6 -
Offset)
Select Case Mid$(Settings, Len(Settings) - 4 - Offset,
1)
Case "e"
    cboParity.ListIndex = 0
Case "m"
    cboParity.ListIndex = 1
Case "n"
    cboParity.ListIndex = 2
Case "o"
    cboParity.ListIndex = 3
Case "s"
    cboParity.ListIndex = 4
End Select

cboDataBits.Text = Mid$(Settings, Len(Settings) - 2 -
Offset, 1)
cboStopBits.Text = Right$(Settings, 1 + Offset)
```

```
cboPort.ListIndex = Form1.MSComm1.CommPort - 1
optFlow(Form1.MSComm1.Handshaking).Value = True
End Sub
```

```
Private Sub cmdCancel_Click()
Unload Me
End Sub
```

```
Private Sub cmdOK_Click()
Dim OldPort As Integer, ReOpen As Boolean

On Error Resume Next
Echo = iTempEcho
OldPort = Form1.MSComm1.CommPort
NewPort = cboPort.ListIndex + 1

If NewPort <> OldPort Then
    If Form1.MSComm1.PortOpen Then
           Form1.MSComm1.PortOpen = False
           ReOpen = True
    End If

    Form1.MSComm1.CommPort = NewPort
    If Err = 0 Then
        If ReOpen Then
            Form1.MSComm1.PortOpen = True
        End If
    End If

    If Err Then
        MsgBox Error$, 48
        Form1.MSComm1.CommPort = OldPort
        Exit Sub
    End If
End If

Form1.MSComm1.Settings = Trim$(cboSpeed.Text) & "," &
Left$(cboParity.Text, 1) _
    & "," & Trim$(cboDataBits.Text) & "," &
Trim$(cboStopBits.Text)

If Err Then
    MsgBox Error$, 48
    Exit Sub
End If

Form1.MSComm1.Handshaking = iFlow
If Err Then
    MsgBox Error$, 48
    Exit Sub
End If

Unload Me
End Sub
```

```
Private Sub Form_Load()

Me.Left = (Screen.Width - Me.Width) / 2
Me.Top = (Screen.Height - Me.Height) / 2
fraSettings.Move tabSettings.ClientLeft,
tabSettings.ClientTop
fraSettings.ZOrder
LoadPropertySettings

End Sub
```

```
Private Sub optFlow_Click(Index As Integer)
iFlow = Index
End Sub
```