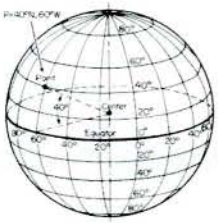
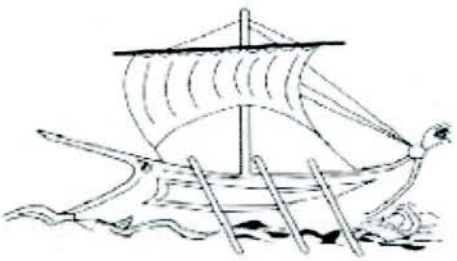
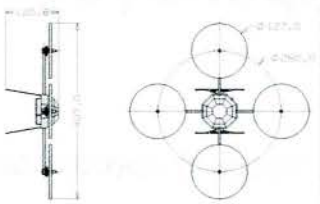
ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ **Πτυχιακή Εργασία**

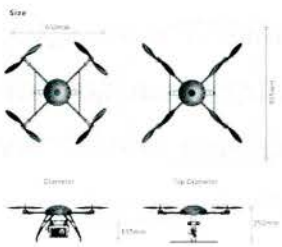


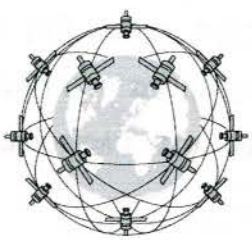


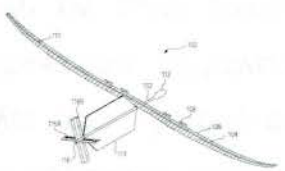
Τεχνολογικό Εκπαιδευτικό Ίδρυμα
Τ.Ε.Ι. ΠΕΙΡΑΙΑ

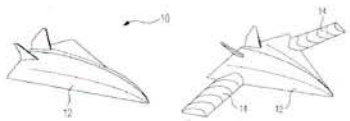


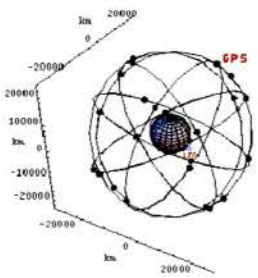
Αυτόματο σύστημα εύρεσης διαδρομής και πλοήγησης μη επανδρωμένου ιπτάμενου οχήματος

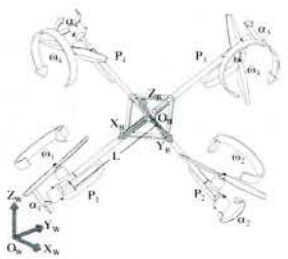












Συγγραφέας
Γιακουμίδης Νικόλαος

Επιβλέπον καθηγητής
Αλαφοδήμος Κων/νος
Νικολάου Γρηγόρης

ΑΘΗΝΑ 2012

Πρόλογος

Η πτυχιακή εργασία αυτή αναλήφθηκε τον Δεκέμβριο του 2010 και ολοκληρώθηκε στο διάστημα δεκατεσσάρων μηνών.

Ο κύριος σκοπός της πτυχιακής εργασίας είναι η εφαρμογή των περισσότερον δυνατό γνώσεων που έχω διδαχτεί στο τμήμα Αυτοματισμού του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Πειραιά, και σαν απόδειξή των γνώσεων αυτών να συνθέσω, σχεδιάσω, υλοποιήσω ένα σύστημα με στόχο την αυτόματη μεταφορά ενός μη επανδρωμένου οχήματος, από τη γεωγραφική θέση που βρίσκετε σε μια άλλη.

Για να υλοποιηθεί ο στόχος αυτός ,αρχικά έγινε μια γενική έρευνα σε αντίστοιχες εφαρμογές που έχουν υλοποιηθεί, έπειτα αναλύθηκε η θεωρία που πρέπει να γνωρίζει κάποιος ώστε να υλοποιήσει το έργο αυτό. Στη συνεχεία επιλέχτηκε ο κατάλληλος εξοπλισμός που εκτελέστηκαν τα πειράματα που ήταν απαραίτητα για την υλοποίηση του συστήματος , και τέλος σχεδιάστηκε και υλοποιήθηκε το λογισμικό που εκτελεί τον αυτόματο έλεγχο του συστήματος.

Ευχαριστίες

Με την εφαρμογή της παρούσας εργασίας θέλω να ευχαριστήσω τους καθηγητές μου Γρήγορη Νικολάου και Αλαφοδήμο Κων/νο για την υποστήριξη τους, και την καθοδήγηση τους, όπως επίσης και τους ανθρώπους , που με τις γνώσεις και την εμπειρία τους με βοήθησαν στην υλοποιήση αυτού του έργου, και αναφέρω τα ονόματα τους στη παρακάτω λίστα :

Νικόλαο Μαυρίδη
Γεώργιο Κορρέ
Εμμανουήλ Φραγκουλόπουλο
Κωνσταντίνο Γκούμα

Πίνακας περιεχομένων

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

ΜΗ ΕΠΑΝΔΡΩΜΕΝΑ ΙΠΤΑΜΕΝΑ ΟΧΗΜΑΤΑ (Unmanned Aerial Vehicle (UAV)) ΚΑΙ ΜΙΚΡΟ ΙΠΤΑΜΕΝΑ ΟΧΗΜΑΤΑ (Micro Aerial Vehicle (MAV)).

Κεφάλαιο 2

ΘΕΩΡΗΤΙΚΟ ΠΑΡΑΡΤΗΜΑ

Κεφάλαιο 3

ΕΠΙΛΟΓΗ ΕΞΩΠΛΙΣΜΟΥ ΚΑΙ ΥΛΙΚΩΝ

Κεφάλαιο 1

Μη Επανδρωμένα Εναέρια Οχήματα (Unmanned Aerial Vehicle (UAV))και Μικρο Εναέρια Οχήματα (Micro Aerial Vehicle (MAV))

Ορισμοί, Ιστορία, Διαχωρισμός, και Εφαρμογές

Πριν από οποιαδήποτε συζήτηση για τα UAV και για τις τεχνολογίες που χρησιμοποιούνται, είναι απαραίτητο να δοθούν κάποιες διευκρινίσεις σχετικά με την ορολογία, τον διαχωρισμό, και πιθανές εφαρμογές των UAV.

Ορισμός (1.1)

Ένα μη επανδρωμένο αεροσκάφος (Unmanned Aerial Vehicle (UAV)) ή ένα εξ αποστάσεως χειριζόμενο όχημα (remotely operated aircraft (ROA)), ή ένα εξ αποστάσεως κατευθυνόμενο όχημά (remotely piloted vehicle(RPV)) σύμφωνα με τη γενική έννοια ορίζεται ένα αεροσκάφος το οποίο δεν περιέχει πιλότο στο χώρο του αεροσκάφους και η κινητήρια δύναμη, η οποία παρέχει την ώθηση ανύψωσης, όπως και τα πτερύγια κατεύθυνσης, ελέγχονται είτε από αυτόνομο σύστημά πλοήγησης είτε από τηλεχειριζόμενο σύστημα πλοήγησης. Όμως ούτε ένας πύραυλος, ο οποίος πετά σε μια βαλλιστική τροχιά, ούτε ένας πλοηγούμενος πύραυλος (cruise missile) ανήκουν σε αυτή την κατηγορία.

Από την άλλη πλευρά, το AIAA (American Institute of Aeronautics and Astronautics) ορίζει ως UAV ένα αεροσκάφος, το οποίο έχει σχεδιαστεί ή έχει τροποποιηθεί, να μην πλοηγείται από ανθρώπινο δυναμικό και λειτουργεί μέσω ηλεκτρονικών εντολών των οποίων είναι υπεύθυνο ένα ενσωματωμένο αυτόνομο σύστημα διαχείρισης ελέγχου πτήσης που δεν απαιτεί την παρέμβαση ανθρώπινου ελεγκτή. Παρά το γεγονός ότι δεν υπάρχει αυστηρός ορισμός της διαφοράς μεταξύ ενός UAV και MAV, σύμφωνα με έναν ορισμό από την DARPA (Defense Advanced Research Projects Agency) του αμερικανικού υπουργείου Άμυνας, ένα MAV (Micro Aerial Vehicle) έχει μέγιστες διαστάσεις (μήκος, πλάτος, ή ύψος) 15 εκατοστά ή λιγότερο.

Κεφάλαιο 7

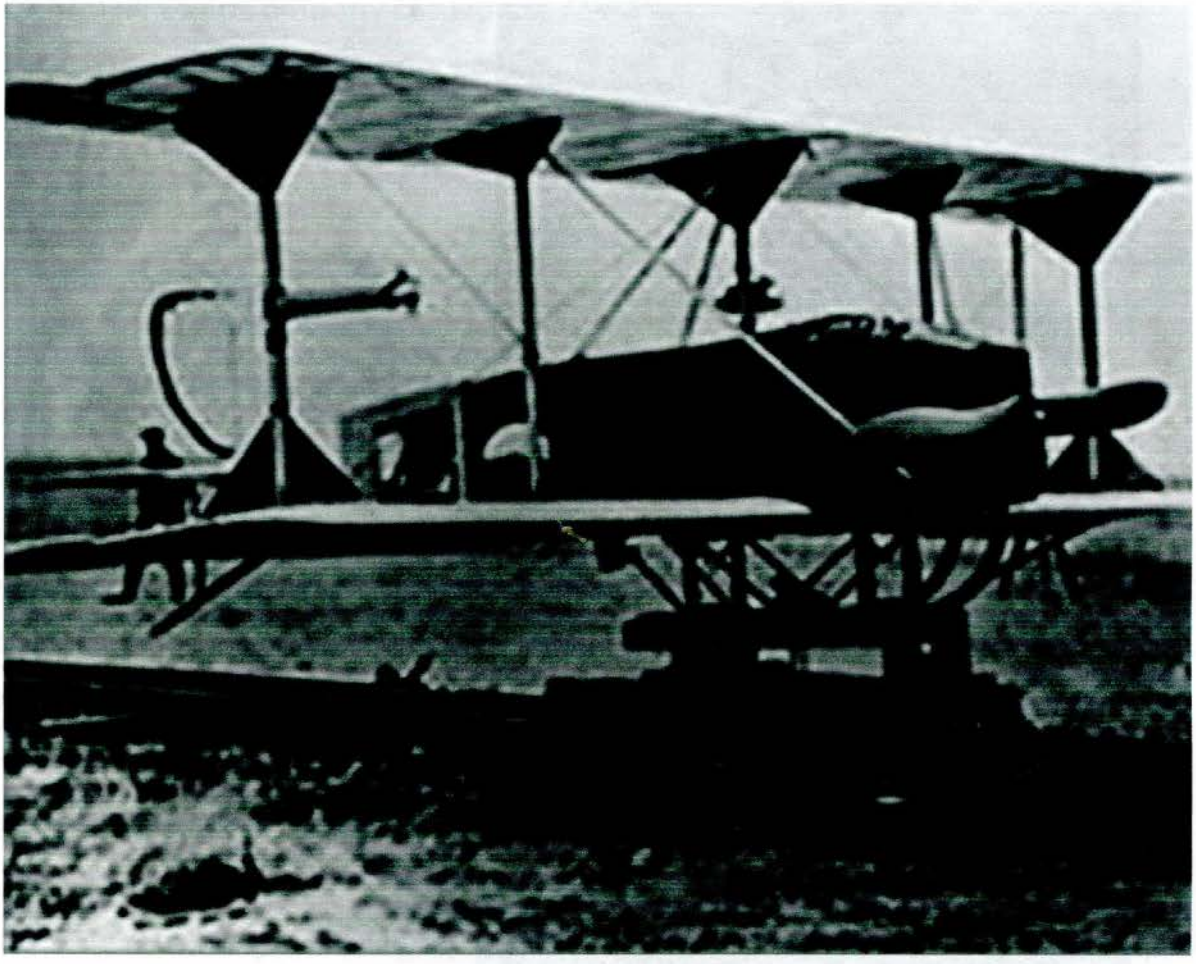
ΑΛΓΟΡΙΘΜΟΣ ΜΙΚΡΟ ΕΛΕΓΚΤΗ

Κεφάλαιο 8

ΒΙΒΛΙΟΓΡΑΦΙΑ

Μια σύντομη ιστορική αναδρομή (1.2)

Το πρώτο UAV κατασκευάστηκε από τους Αμερικανούς και Sperry και Lawrence το 1916 οπου φαίνεται στο σχήμα (1.1). Ανέπτυξαν ένα γυροσκόπιο για να σταθεροποιούν το σώμα, προκειμένου να κατασκευάσουν ένα αυτόματο πιλότο. Αυτό είναι γνωστό ως η αρχή του "αυτόματου ελέγχου στάσης (Attitude)" η οποία χρησιμοποιείται για την αυτόματη διεύθυνση του αεροσκάφους.



(1.1) Aviation Torpedo 1916

Οι Lawrence και Sperry ονόμασαν τη συσκευή τους "aviation torpedo (αέρια τορπίλη)" και πέταξε σε απόσταση που υπερβαίνει περίπου τα 30 μίλια. Ωστόσο, λόγω της ανεπαρκούς ανάπτυξης της τεχνολογίας, φαίνεται ότι τα UAV δεν χρησιμοποιήθηκαν στον Πρώτο Παγκόσμιο Πόλεμο ούτε στον Δεύτερο Παγκόσμιο Πόλεμο.

Η ανάπτυξη των UAV ξεκίνησε ουσιαστικά στα τέλη της δεκαετίας του 1950, στον πόλεμο του Βιετνάμ και τον ψυχρό πόλεμο, με συνεχή έρευνα και ανάπτυξη πλήρου κλίμακας την δεκαετία του 1970. Το διάγραμμα (1.2) δείχνει ένα UAV που ονομάζεται Firebee. Μετά τον πόλεμο του Βιετνάμ, οι ΗΠΑ και το Ισραήλ άρχισαν να αναπτύσου μικρότερα και φθηνότερα UAV. Αυτά ήταν μικρά αεροσκάφη εξοπλισμένα με μικρούς κινητήρες, όπως αυτές που χρησιμοποιούνταν στις μοτοσικλέτες. Επίσης εκείνη τη περίοδο χρησιμοποιήθηκαν και βιντεοκάμερες μεταδίδοντας εικόνες στον χειριστή.

(1.2) Firebee 1970

Οι ΗΠΑ χρησιμοποίησαν τα UAVs σε πραγματική χρήση στον πόλεμο του Κόλπου το 1991.Τα UAVs για στρατιωτικές εφαρμογές αναπτύχθηκαν γρήγορα μετά από αυτό. Το πιο διάσημο μη επανδρωμένο εναέριο όχημα για στρατιωτική χρήση είναι τ Predator, το οποίο φαίνεται στο σχήμα (1.3).

(1.3) Predator 1990

Από την άλλη πλευρά, η NASA ήταν το επίκεντρο της έρευνας των UAV για μη στρατιωτικούς σκοπούς κατά τη διάρκεια αυτής της περιόδου. Το πιο χαρακτηριστικό παράδειγμα της έρευνας αυτής ήταν απο την ERAST (Environmental Research Aircraft and Sensor Technology) που ξεκίνησε στη δεκαετία του 1990, και ήταν ένα συνθετικό εγχείρημα έρευνας για ένα UAV, που περιελάμβανε την ανάπτυξη της τεχνολογίας που απαιτείται για να πετάει σε μεγάλα υψόμετρα μέχρι τα 30.000 m, με μια εξελιγμένη τεχνολογία πτήσης, κινητήρα, αισθητήρων, κλπ. Στα αεροσκάφη που αναπτύχθηκαν σε αυτή την έρευνα περιελάμβανε τα Helios, Proteus, Altus, Pathfinder, κλπ., τα οποία παρουσιάζονται στα σχήματα (1.4 , 1.5).Και αυτά σχεδιάστηκαν για να πραγματοποιούν περιβαλλοντικές μετρήσεις.

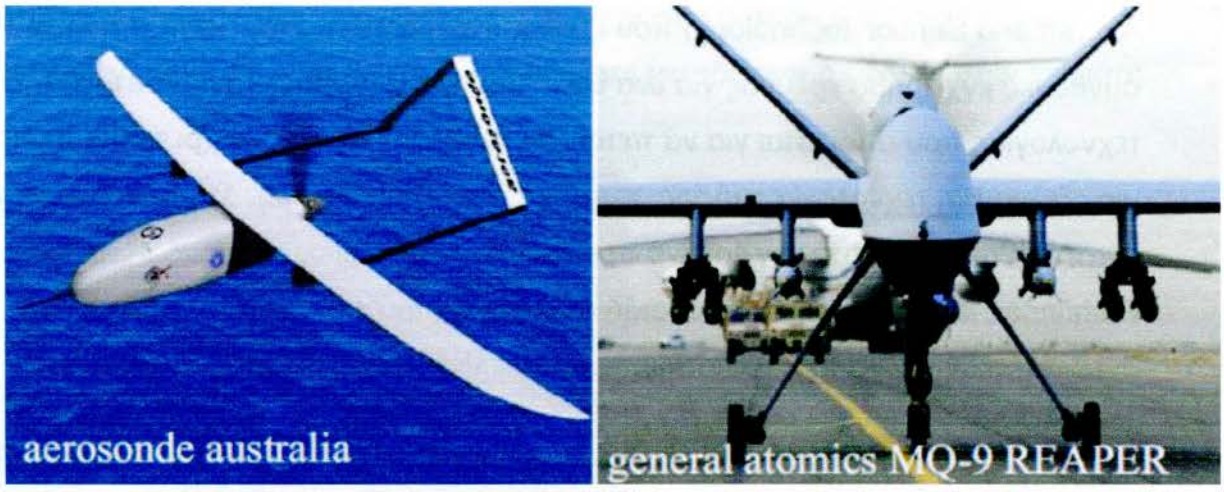
(1.4) NASA (Helios)

(1.5) NASA (Proteus)

Κατάταξη των πλατφορμών UAV (1.3)

Κατά τις τελευταίες δεκαετίες, έχουν γίνει σημαντικές προσπάθειες γύρω από την αύξηση της διάρκειας πτήσεως , το ωφέλιμο φορτίο, και της ανοχής στις διάφορες καιρικές συνθήκες , με αποτέλεσμα διαφορετικές συνθέσεις UAV με διαφορετικά μεγέθη, διάρκεια αυτονομίας και ικανοτήτων. Εδώ θα, επιχειρήσουμε να ταξινομήσουμε τα UAVs, σύμφωνα με τα χαρακτηριστικά τους (αεροδυναμική διαμόρφωση, το μέγεθος, κ.λ.π.).Οι πλατφόρμες των UAV's εμπίπτουν σε μία από τις ακόλουθες τέσσερις κατηγορίες:

1. UAV σταθερής πτέρυγας, τα οποία αναφέρονται σε μη επανδρωμένα αεροπλάνα (με φτερά) που απαιτούν ένα διάδρομο για την απογείωση και την προσγείωση ή εκτόξευση με καταπέλτη. Αυτά έχουν συνήθως αυτονομία λειτουργίας και μπορούν να πετάξουν σε υψηλές ταχύτητες πλεύσης, (βλ. Σχήμα. 1.6 μερικά παραδείγματα).



(1.6)

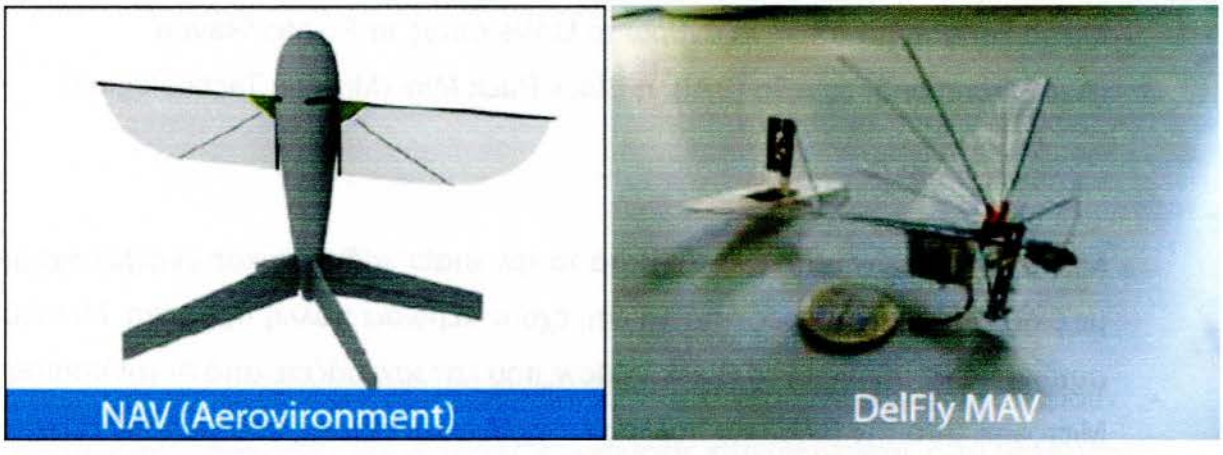
2. Με περιστροφικά πτερύγιά UAV's , που ονομάζονται επίσης ελικοφόρα UAVs ή κάθετης απογείωσης και προσγείωσης (VTOL) UAV, τα οποία έχουν τα πλεονεκτήματα να αιωρούνται και να διαπράττουν απότομους ελιγμούς. Οι δυνατότητες αυτές είναι χρήσιμες για πολλές ρομποτικές αποστολές, ειδικά σε μη στρατιωτικές εφαρμογές. Τα ελικοφόρα UAV μπορεί να έχουν διαφορετικές διαμορφώσεις, όπως :με ένα κύριο στροφείο και ενα ουραίο στροφείο (συμβατικό ελικόπτερο), ομοαξονικό στροφείο, παράλληλα στροφεία, πολλά στροφεία π.χ. (τετρα-κόπτερα), κλπ. (βλ. Σχήμα. 1.7 μερικα παραδείγματα).

(1.7)

3. Πηδαλιουχούμενα εύκαμπτα αερόστατα τα οποία είναι ελαφρύτερα από τον αέρα και έχουν μεγάλη ανθεκτικότητα, πετούν σε χαμηλές ταχύτητες, και συνήθως είναι μεγάλου μεγέθους (βλέπε σχήμα. 1.8 για ορισμένα παραδείγματα).

(1.8)

4. Flapping-wing (φτερωτά) UAVs τα οποία έχουν ευέλικτα μικρά φτερά και είναι εμπνευσμένα από τα πουλιά και τα ιπτάμενα έντομα, (βλ. Σχήμα 1.9)



5. Επίσης υπάρχουν κάποιες άλλες υβριδικές συνθέσεις ή μετατροπές συνθέσεων, τα οποία μπορούν να απογειωθούν κάθετα και μετά να αλλάξουν την κλίση στροφείων τους από κάθετα σε οριζόντια και πετούν σαν αεροπλάνα, (βλ. Σχήμα 1.10) όπως το Eagle Eye UAV.

Άλλος τύπος ταξινόμησης των UAV

Ένα άλλο κριτήριο που χρησιμοποιείται επί του παρόντος να γίνει διάκριση μεταξύ των αεροσκαφών είναι το μέγεθος και η διάρκεια πτήσης μερικά παραδείγματα αναφέρονται παρακάτω:

- Μεγάλης αυτονομίας, μεγάλου υψομέτρου (HALE) UAV, όπως για παράδειγμα, των Northrop- Grumman Ryan' s Global Hawks (65.000 πόδια ύψος, χρόνος πτήσης 35 ώρες, και ωφέλιμο φορτίο 1.900 λίβρες).

- Μεσαίου υψομέτρου Long Endurance (MALE) UAV, όπως για παράδειγμα της General Atomics Predator (27.000 πόδια ύψος, 30/40 ώρες πτήσης, και ωφέλιμο φορτίο 450 λίβρες).

- Τακτική χρησης UAVs όπως Hunter, Shadow 200, και Pioneer (15.000 πόδια ύψος, χρόνος πτήσης 5-6 ώρες, και 25 κιλά ωφέλιμο φορτίο).
- Μικρά και φορητά από ένα άνθρωπο UAVs οπως τα Pointer/Raven (AeroVironment), Javelin (BAI), ή Black Pack Mini (Mission Technologies).

- Μικρο εναέρια οχήματα (MAV): Κατά τα τελευταία χρόνια, μικρο εναέρια οχήματα, με διαστάσεις μικρότερες από 15 cm, έχουν κερδίσει πολλή προσοχή. Μεταξύ αυτών περιλαμβάνεται το Black Widow που κατασκευάζετε από AeroVironment, το Microstar από την BAE, και πολλά νέα σχέδια και έννοιες που παρουσιάζονται από διάφορα πανεπιστήμια, όπως το Entomopter (Georgia Institute of Technology), Micro Bat (California Institute of Technology), και των NXI (Πανεπιστημίου Berkeley), και άλλα σχέδια από ευρωπαϊκά ερευνητικά κέντρα (Εικ. 1.11).

(1.11)

Εφαρμογές των UAV's (1.4)

Σήμερα, οι κύριες εφαρμογές μη επανδρωμένων εναέριων οχημάτων αφορούν στρατιωτικές εφαρμογές και οι κύριες επενδύσεις κατευθύνονται από μελλοντικά στρατιωτικά σενάρια. Τα περισσότερα στρατιωτικά μη επανδρωμένα αεροσκάφη χρησιμοποιούν συστήματα κυρίως για τη συλλογή πληροφοριών, για επιτήρηση, αναγνώριση (ISR), και επιθέσεις.

Η επόμενη γενιά των UAV θα εκτελεί πιο πολύπλοκες αποστολές, όπως εναέριες μάχες, εντοπισμό στόχων, αναγνώριση και καταστροφή, επίθεση και εξουδετέρωση

αεράμυνας του εχθρού, ηλεκτρονικό πόλεμο, δικτυακούς κόμβους και αναμετάδοση επικοινωνιών, εναέρια παράδωση ανεφοδιασμού, κλπ.
Σήμερα, οι πολιτικές εφαρμογές για τα UAV συνεχίζουν να εμφανίζονται. Ωστόσο, οι προσδοκίες για την ανάπτυξη αγοράς αστικών και εμπορικών UAVs είναι πολύ υψηλές για την επόμενη δεκαετία (Εικ. χχ).Μερικές πιθανές πολιτικές εφαρμογές των UAV είναι:

- Η επιβολή του νόμου και εφαρμογές ασφάλειας.
- Επιθεώρηση του εδάφους, αγωγών, επιχειρήσεις κοινής ωφέλειας, κτίριων, κλπ.
- Επιτήρηση των θαλάσσιων συνόρων, την οδική κυκλοφορία, κλπ.
- Έρευνα και διάσωση σε περίπτωση κρίσης, ή μεγάλων καταστροφών.
- Η παρακολούθηση και προστασία του περιβάλλοντος.
- Εφαρμογές στη Γεωργία και δασοκομία.
- Εφαρμογές πυρόσβεσης.
- Εφαρμογές αναμετάδοσης επικοινωνιών και τηλεπαρακολούθησης.
- Εναέρια χαρτογράφηση και μετεωρολογία.
- Έρευνα από πανεπιστημιακά εργαστήρια.
- Και πολλές άλλες εφαρμογές που ακόμα δεν γνωρίζουμε .

Τα τελευταία χρόνια, υπήρξε ταχεία ανάπτυξη των αυτόνομων μη επανδρωμένων αεροσκαφών που είναι εξοπλισμένα με αυτόνομες συσκευές ελέγχου και ονομάζονται "Μη Επανδρωμένα Εναέρια Οχήματα (UAV)¨ και ¨Μικρο Εναέρια Οχήματα (MAVs). Αυτά έχουν γίνει γνωστά ως "Ρομποτικά Αεροσκάφη," και η χρήση τους έχει γίνει ευρέως διαδεδομένη.

Μπορούν να ταξινομηθούν ανάλογα με την εφαρμογή τους για Στρατιωτική ή Πολιτική χρήση.

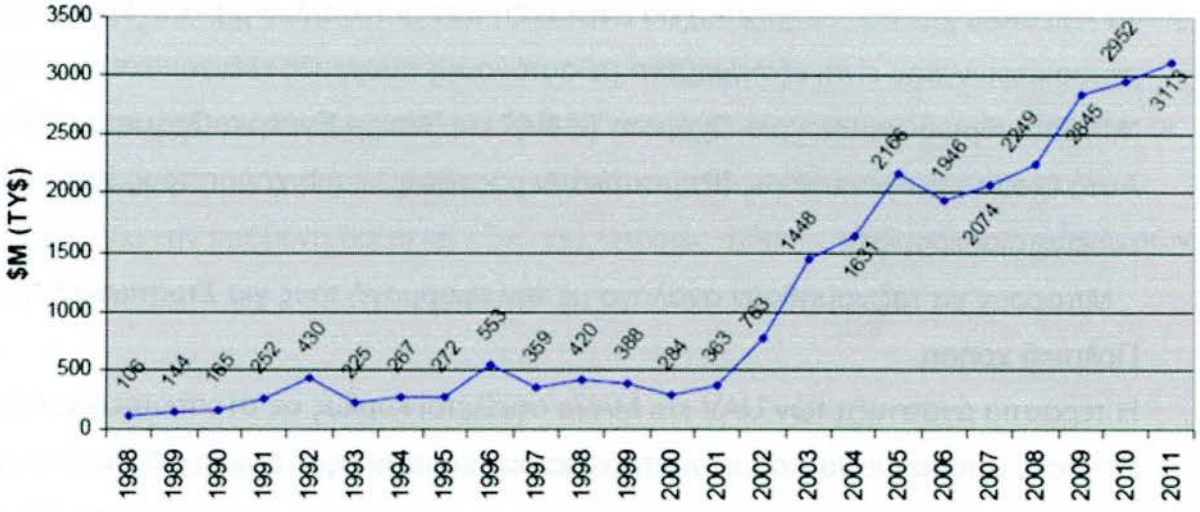
Η τεράστια ανάπτυξη των UAV και MAVs οφείλεται κυρίως σε στρατιωτική χρήση. Ωστόσο, μπορούμε να πούμε ότι παραμένουν κρυμμένες οι άπειρες δυνατότητες των UAV's χρησιμοποιώντας τα εξαιρετικά τους χαρακτηριστικά για πολιτικές εφαρμογές.

Τα UAVs παρέχουν σημαντικά πλεονεκτήματα σε σχέση με τα επανδρωμένα αεροσκάφη, όταν χρησιμοποιούνται σε επικίνδυνες αποστολές για εναέρια επιτήρηση, αναγνώριση, και επιθεώρηση σε πολύπλοκα και επικίνδυνα περιβάλλοντα.

Συγκεκριμένα, τα UAV έχουν τη δυνατότητα να προσαρμοστούν καλύτερα σε περιβάλλον χαμηλής η ακόμα και μηδενικής ορατότητας, η σε περιβάλλον που είναι επικίνδυνα για τη ανθρωπινή διαβίωσή. Ο χαμηλότερος κίνδυνος αρνητικών εξελίξεων και η μεγαλύτερη αξιοπιστία για την επιτυχία της αποστολής είναι δύο ισχυρά κίνητρα για τη συνεχή εξέλιξη των συστημάτων που χρησιμοποιούν τα μη επανδρωμένα αεροσκάφη .

Επιπλέον, πολλοί τεχνολογικοί, οικονομικοί,και πολιτικοί παράγοντες ενθαρρύνουν την ανάπτυξη και λειτουργία των UAV.

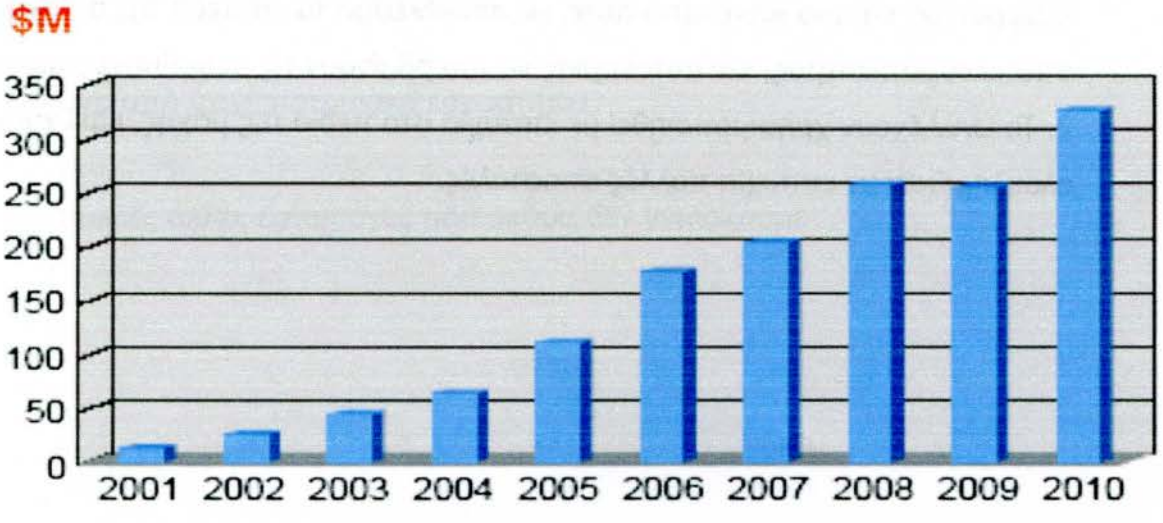
1. Οι τεχνολογικές εξελίξεις παρέχουν σημαντική επιρροή όπως οι νεότεροι αισθητήρες, μικροελεγκτές , και τα συστήματα προώθησης είναι μικρότερα, ελαφρύτερα και πιο ικανά από ποτέ, με αποτέλεσμα τα επίπεδα της αντοχής, αποτελεσματικότητας, και αυτονομίας να υπερβαίνουν τις ανθρώπινες ικανότητες.
2. Τα UAV έχουν χρησιμοποιηθεί με επιτυχία στο πεδίο της μάχης, και έχουν ολοκληρώσει με επιτυχία πολλές αποστολές.



(1.12) Χρηματοδότηση υπουργείου αμύνης των Η.Π.Α

Ρυθμός ανάπτυξης των UAV (1.5)

Οι παράγοντες αυτοί έχουν οδηγήσει σε περισσότερη χρηματοδότηση και ένα μεγάλο αριθμό παραγγελιών παραγωγής. Αρκετές μελέτες αγοράς έχουν προβλέψει ότι η παγκόσμια αγορά UAV θα διευρύνει σημαντικά την επόμενη δεκαετία.Αυτές οι μελέτες, εκτιμούν ότι οι δαπάνες των UAV's θα τριπλασιαστούν κατά την επόμενη δεκαετία, με συνολικό ύψος κοντά στα 55 δις $ για τα επόμενα 10 έτη . Όπως αναφέρεται στο διάγραμμα (1.12) τα επόμενα 5-7 χρόνια, η αγορά στα UAV στις ΗΠΑ θα φθάσει 16 δισεκατομμύρια δολάρια, ακολουθούμενη από την Ευρώπη, η οποία θα δαπανήσει περίπου 3 δισεκατομμύρια δολάρια διάγραμμα (1.13). Στις ΗΠΑ για παράδειγμα, στους προϋπολογισμούς ανάπτυξης αυξήθηκε ραγδαία μετά το 2001, όπως φαίνεται στην εικόνα (1.12) .



(1.13) Χρηματοδότηση για τα UAV στη Ευρώπη

Σήμερα, υπάρχουν αρκετές εταιρείες που σχεδιάζουν, αναπτύσσουν και παράγουν εκατοντάδες UAV καθημερινά. Στις περισσότερες χώρες, με εξαίρεση την Ιαπωνία και κάποιες άλλες χώρες, η πλειοψηφία της έρευνας και της ανάπτυξης υποστηρίζεται από στρατιωτικές δαπάνες. Ωστόσο, η αστική αγορά UAV αναμένεται να προκύψει κατά την επόμενη δεκαετία, ξεκινώντας πρώτα από κυβερνητικούς οργανισμούς για χρήση επιτήρησης, από την ακτοφυλακή, οργανισμούς περιπολίας συνόρων, ομάδων διάσωσης, αστυνομικές δυνάμεις, κλπ.

Αν και οι ένοπλες δυνάμεις σε όλο τον κόσμο εξακολουθούν να έχουν το μεγαλύτερο ενδιαφέρον για έρευνα και την ανάπτυξη τεχνολογιών που έχουν τη δυνατότητα να προωθήσουν τις δυνατότητες των UAV.

Οι τεχνολογίες των UAV (1.6)

Οι τεχνολογίες που αναπτύσσονται για τα UAV είναι συγκεκριμένες, υπό την έννοια ότι για να αντισταθμίζουν την απουσία του πιλότου και να καταστεί έτσι δυνατή η πτήση των μη επανδρωμένων οχημάτων και η αυτόνομη συμπεριφορά τους στηρίζονται κυρίως στις παρακάτω τεχνολογίες :

Predator Targeting System

(1.6.1) **Αισθητήρες πλοήγησης και μικροεπεξεργαστές :**

Αισθητήρες αποτελούν ένα από τα πιο σημαντικά στοιχεία των συστημάτων των μη επανδρωμένων αεροσκαφών και είναι απαραίτητοι για την πλοήγηση αεροναυτιλίας και την επίτευξη της αποστολής. Επεξεργαστές σε συνδυασμό με τον προγραμματισμό τους επιτρέπουν στα UAV να πετούν όλο και πιο αυτόνομα στις αποστολές του με ελάχιστη ή ακόμα με μηδενική ανθρώπινη παρέμβαση.

(1.6.2) **Συστήματα επικοινωνίας (ζεύξης δεδομένων):**

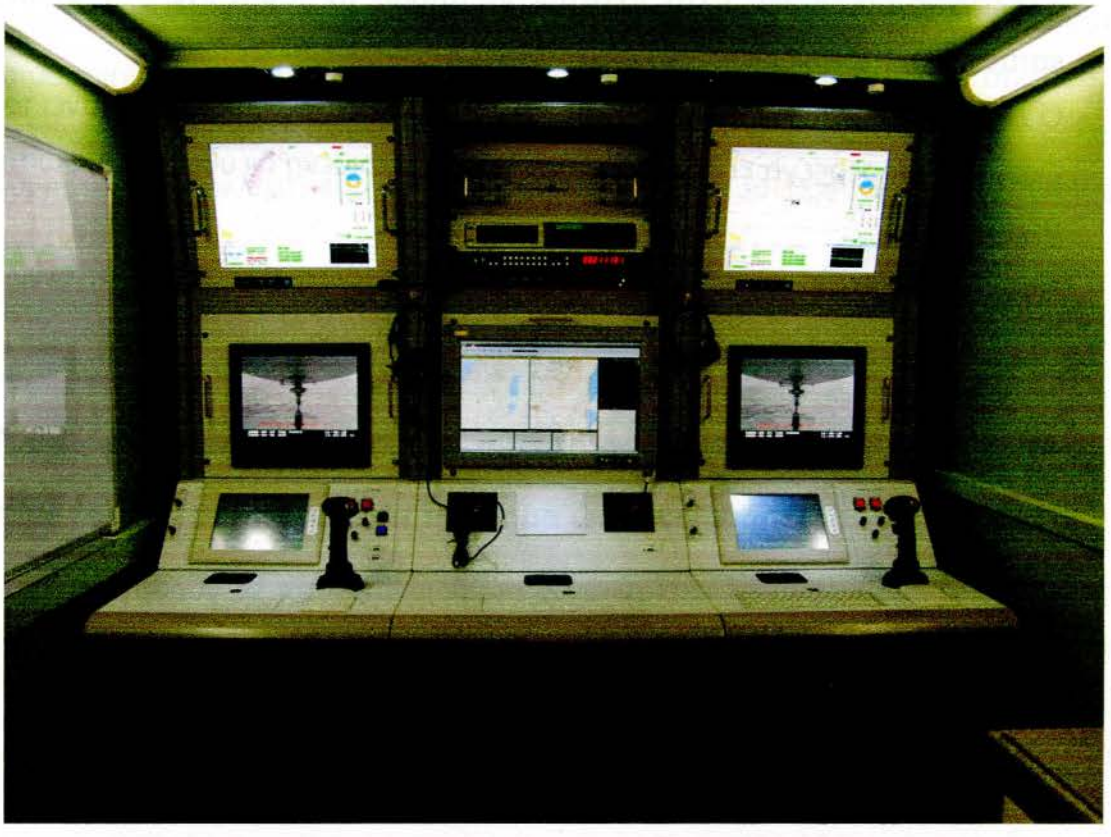
Τα κύρια χαρακτηριστικά που μας ενδιαφέρουν για τις τεχνολογίες επικοινωνίας είναι: η αξιοπιστία , η προσαρμοστικότητα, η ασφάλεια, και η δυνατότητα ελέγχου των δεδομένων που ανταλλάσσονται ανάμεσα στον σταθμό ελέγχου και το μη επανδρωμένο όχημα και αποτελούν έναν απο τους κύριους σκοπούς των UAV όπου είναι η απομακρυσμένη πληροφόρηση. Και όλα αυτά σε συνδυασμό με τη ταχύτητα μεταφοράς των δεδομένων και τη συχνότητα μετάδοσης .

(1.6.3) **Επίγειος σταθμός ελέγχου (Ground Control Station (GCS))**

Ο επίγειος σταθμός ελέγχου είναι το τμήμα ελέγχου και επικοινωνιών με το UAV. Υπάρχουν πολλές και διαφορετικές λειτουργίες όπου είναι υπεύθυνος ο σταθμός ελέγχου. Η πιο σημαντική από αυτές είναι:

Η επαφή του ανθρώπινου δυναμικού με το όχημα ώστε να ορίσει την αποστολή , να συλλέξει τις χρήσιμες πληροφορίες που θα μας στείλει το όχημα , να αντιδράσει σε μια μη επιθυμητή κατάσταση του οχήματος κλπ.

Με λίγα λόγια ο σταθμός εδάφους είναι ένα βασικό και αναπόσπαστο κομμάτι του UAV όπου το ανθρώπινο δυναμικό δίνει και παίρνει πληροφορίες απομακρυσμένα από το όχημα.



(1.14) Επίγειος σταθμός ελέγχου (Ground station control)

(1.6.4) **Επί του αεροσκάφους νοημοσύνη (καθοδήγηση, πλοήγηση και έλεγχος)**

Η νοημοσύνη του UAV είναι μια πολύπλοκη διεργασία η οποία αναλαμβάνει να χειριστεί τη κατάσταση του οχήματος, και είναι αντιστρόφως ανάλογη με το ποσοστό της εποπτείας που απαιτείται από τον ανθρώπινο φορέα.

Ουσιαστικά είναι το πρόγραμμα που τρέχει στον ή στους επεξεργαστές του UAV ώστε να επιτευχθούν οι απαραίτητες ενέργειες ώστε το όχημα να βγάλει εις πέρας την αποστολή του. Μερικές από αυτές τις διεργασίες είναι :

- Ο έλεγχος ισορροπίας ώστε το όχημα να μπορεί να παραμείνει στην επιθυμητή κατάσταση στάσης (Attitude) .
- Ο έλεγχος πλοήγησης που σημαίνει ποιές ενέργειες θα πρέπει να κάνει το όχημα ώστε να μπορέσει να κατευθυνθεί σε μια συγκεκριμένη διεύθυνση , ύψωση , ταχύτητα ,
  ποιές ενέργειες θα κάνει όταν υπάρξει κάποιο σφάλμα στα συστήματά του , κλπ.

Κεφάλαιο 2

ΘΕΩΡΗΤΙΚΟ ΠΑΡΑΡΤΗΜΑ

Ο σκοπός του συστήματος που σχεδιάστηκε είναι η αυτόματη μεταφορά του οχήματος από τη γεωγραφική θέση που βρίσκετε σε μια άλλη. Για να γίνει αυτό θα πρέπει να είναι γνωστές οι συντεταγμένες της γεωγραφική θέσης που βρίσκετε το όχημά σε πραγματικό χρόνο και οι συντεταγμένες της γεωγραφική θέση που είναι επιθυμητό να πάει το όχημα. Για να γίνει γνωστή η γεωγραφική θέση του οχήματος , χρησιμοποιήθηκε ένα αισθητήριο με δέκτη GPS. Οι συντεταγμένες του σημείου που έχει σαν στόχο να πάει το όχημά ,είναι γνωστές από το χρήστη. Οπότε πρέπει να υπολογιστεί η κατάλληλη και πιο σύντομη διαδρομή που πρέπει να ακολουθήσει το όχημα.

Για να επιτευχθεί ο στόχος αυτός, πρέπει να απαντηθούν τα δύο παρακάτω ερωτήματα :

1. **Ποια είναι η απόσταση του στόχου από το σημείο που βρίσκετε το όχημα ;**
2. **Ποια πορεία πρέπει να έχει το όχημα ώστε να φτάσει τον στόχο του ;**

Για να απαντηθούν τα παραπάνω ερωτήματα πρέπει να αναλυθούν οι έννοιες οπού χρησιμοποιήθηκαν όπως επίσης και το θεωρητικό υπόβαθρο που οδήγησε στη λύση του προβλήματος.

(2.1) Συντεταγμένες γεωγραφικής θέσης ενός σημείου στη Γη

Η γεωγραφική θέση ενός κινητού σε μια χρονική στιγμή καθορίζεται από τις γεωγραφικές συντεταγμένες που είναι το γεωγραφικό μήκος και το γεωγραφικό πλάτος.

(2.1.1) **Γεωγραφικό πλάτος**

Το γεωγραφικό πλάτος ενός σημείου στην επιφάνεια της γης είναι η γωνιακή απόσταση βόρεια ή νότια από τον Ισημερινό και εκφράζεται σε μοίρες και το εύρος των τιμών του είναι από 0 που αντιστοιχεί στον Ισημερινό έως 90 που αντιστοιχεί στους πόλους της Γης .

(2.1.2) **Γεωγραφικό μήκος**

Όπως και στο γεωγραφικό πλάτος, το γεωγραφικό μήκος ενός σημείου στην επιφάνεια της γης είναι η γωνιακή απόσταση ανατολικά ή δυτικά του πρώτου παράλληλου που βρίσκεται στο Γκρίνουϊτς και το εύρος των τιμών του είναι απο 0 έως 180 μοίρες το οποίο αντιστιχεί στο Sydney της Αυστραλίας.

(2.3.1) **Παγκόσμιο Σύστημα Θέσης GPS (Global Position System)**

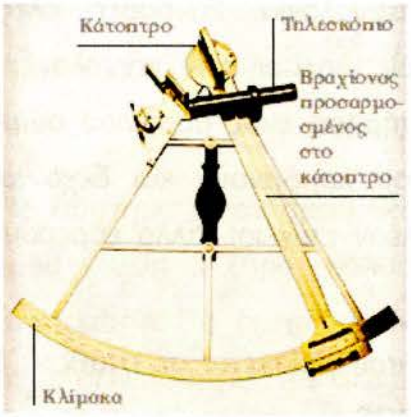
Το **GPS (Global Positioning System), Παγκόσμιο Σύστημα Θέσης** είναι ένα παγκόσμιο σύστημα εντοπισμού θέσης, το οποίο βασίζεται σε ένα "πλέγμα" εικοσιτεσσάρων δορυφόρων στην ατμόσφαιρα Γης, στους οποίους υπάρχουν ειδικές συσκευές χρονομέτρησης και πομποί οι οποίοι εκπέμπουν ηλεκτρομαγνητικά κύματα που περιέχουν κάποιες πληροφορίες για τη θέση και το υψόμετρο του δορυφόρου μαζί με την ακριβή ώρα αποστολής των δεδομένων αυτών μέσα από τα υψηλής ακρίβειας χρονόμετρα που είναι εξοπλισμένοι οι δορυφόροι σε συνεργασία με τους ειδικούς δέκτες αυτών των σημάτων μας παρέχουν ακριβείς πληροφορίες για τη θέση ενός σημείου, το υψόμετρο του, την ταχύτητα και την κατεύθυνση της κίνησης του. Επίσης, σε συνδυασμό με ειδικό λογισμικό χαρτογράφησης μπορούν να απεικονίσουν γραφικά τις πληροφορίες αυτές.

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

(2.3.2) **Ιστορικά στοιχεία για τον εντοπισμό θέσης και σύστημα GPS**

Από την αρχαιότητα η πλοήγηση ήταν από τα πιο σημαντικά εγχειρήματα της ανθρωπότητας.

Τα σημεία του ορίζοντα, ή ακόμη και τα αστέρια, χρησιμοποιούνταν από την αρχαιότητα για τον προσανατολισμό των ανθρώπων. Ένα σταθερό άστρο στον ουρανό, με γνωστή γεωγραφική θέση ως προς το σημείο παρατήρησης, αποτελούσε σημείο αναφοράς όπου βοηθούσε τους ανθρώπους στο να βρουν τη σωστή πορεία τους . Στον προσανατολισμό συνέβαλαν αργότερα και άλλα μέσα, όπως είναι η πυξίδα και ο εξάντας. Ωστόσο ο εξάντας είναι εύχρηστος μόνο για τον προσδιορισμό του γεωγραφικού πλάτους, ενώ η χρήση του για τον προσδιορισμό του γεωγραφικού μήκους είναι δύσκολη και εξαιρετικά σύνθετη, πράγμα που αποτελεί ένα σημαντικό μειονέκτημα για προσδιορισμό του στίγματος στην θάλασσα. Ως αποτέλεσμα, τον 17ο αιώνα, το Ηνωμένο Βασίλειο συνέστησε ένα συμβούλιο επιστημόνων, το οποίο θα επιβράβευε χρηματικά όποιον θα μπορούσε να εφεύρει ένα όργανο, το οποίο θα επέτρεπε τον ακριβή υπολογισμό και των δύο γεωγραφικών συντεταγμένων, δηλαδή γεωγραφικού μήκους και πλάτους.



Το 1761, ο Άγγλος ωρολογοποιός Τζον Χάρισσον (John Harrison), ύστερα από προσπάθειες δώδεκα ετών, κατασκεύασε ένα όργανο, το οποίο δεν ήταν άλλο από το γνωστό σημερινό χρονόμετρο. Σε συνδυασμό με τον εξάντα, το χρονόμετρο επέτρεπε τον υπολογισμό του στίγματος των πλοίων με εξαιρετική ακρίβεια (για τα δεδομένα της εποχής). Πέρασαν αρκετά χρόνια μέχρι να δημιουργηθούν τα πρώτα συστήματα εντοπισμού θέσης που βασίζονταν σε ηλεκτρομαγνητικά κύματα (ραντάρ, στα μέσα του 20ού αιώνα. Τα συστήματα αυτά χρησιμοποιήθηκαν ευρύτατα κατά τη

διάρκεια του Δευτέρου Παγκοσμίου Πολέμου (και χρησιμοποιούνται ακόμη). Τα συστήματα εντοπισμού θέσης της εποχής αποτελούνταν από ένα δίκτυο σταθμών βάσης και κατάλληλους δέκτες.

Ανάλογα με την ισχύ του σήματος που λάμβανε ο κάθε δέκτης από σταθμούς γνωστής γεωγραφικής θέσης, σχηματίζονταν δύο ή περισσότερες συντεταγμένες, μέσω των οποίων προσδιοριζόταν η θέση των σημείων ενδιαφέροντος επάνω σε ένα χάρτη. Στην περίπτωση αυτή, όμως, υπήρχαν δύο διαφορετικά προβλήματα: Στην πρώτη περίπτωση η χρήση σταθμών βάσης, που εξέπεμπαν σήμα σε υψηλή συχνότητα, διέθεταν μεν υψηλή ακρίβεια εντοπισμού, αλλά είχαν μικρή εμβέλεια. Στη δεύτερη περίπτωση συνέβαινε το ακριβώς αντίθετο, δηλαδή ο σταθμός βάσης χρησιμοποιούσε μεν χαμηλή συχνότητα εκπομπής σήματος, προσφέροντας έτσι υψηλότερη εμβέλεια, αλλά η ακρίβεια του ήταν πολύ χαμηλή.

Έστω και με αυτά τα προβλήματα, η αρχή της χρήσης ραδιοκυμάτων για τον εντοπισμό της θέσης ενός σημείου είχε ήδη γίνει. Το Global Positioning System στη σημερινή του μορφή βασίζεται σε παρεμφερή τεχνολογία. Συνδυάζει όλες τις μεθόδους που είχαν χρησιμοποιηθεί στο παρελθόν, δηλαδή την τεχνολογία των ηλεκτρομαγνητικών κυμάτων καθώς και την παρατήρηση ενός ουράνιου σώματος (τεχνητού αυτή τη φορά). Οι σταθμοί βάσης που λαμβάνουν και δέχονται τα απαραίτητα ηλεκτρομαγνητικά κύματα δεν είναι πλέον επίγειοι, αλλά εδρεύουν σε τεχνητούς δορυφόρους.

Ένα δίκτυο πολυάριθμων δορυφόρων (24 - 32) που βρίσκεται σε σταθερή θέση γύρω από τον πλανήτη μας, βοηθά τους δέκτες GPS να υπολογίσουν το ακριβές στίγμα ενός σημείου οπουδήποτε στον κόσμο. Όταν, το 1957, πραγματοποιήθηκε η εκτόξευση του πρώτου δορυφόρου Σπούτνικ, οι άνθρωποι είχαν ήδη αντιληφθεί ότι ένα τεχνητό ουράνιο σώμα κοντά στη Γη είναι δυνατό να χρησιμοποιηθεί για να εντοπιστεί η θέση ενός σημείου πάνω στον πλανήτη.

Αμέσως μετά την εκτόξευσή του, οι ερευνητές του Ινστιτούτου Τεχνολογίας της Μασαχουσέτης (MIT) διαπίστωσαν ότι το σήμα που λαμβανόταν από τον δορυφόρο αυξανόταν καθώς αυτός πλησίαζε προς το επίγειο σημείο παρατήρησης και μειωνόταν όταν ο δορυφόρος απομακρυνόταν από αυτό. Αυτό ήταν και το πρώτο βήμα για την υλοποίηση της τεχνολογίας που σήμερα αποκαλείται Global Positioning System. Με τον ίδιο τρόπο που η θέση ενός δορυφόρου μπορούσε να εντοπιστεί ανάλογα με την ισχύ του σήματος που λαμβάνεται από αυτόν, υπήρχε και η

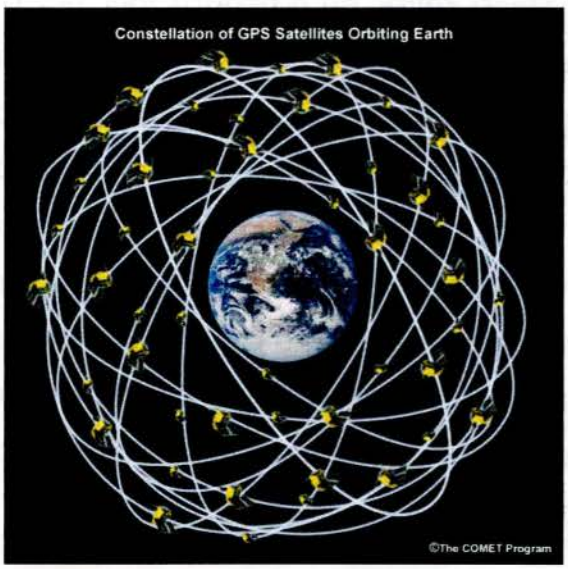
δυνατότητα να συμβεί το ακριβώς αντίθετο: Ο δορυφόρος να εντοπίσει την θέση ενός σημείου με ιδιαίτερη ακρίβεια. Στην πραγματικότητα ένας δορυφόρος δεν είναι αρκετός για να υπάρξουν ακριβή αποτελέσματα, αλλά απαιτούνται τουλάχιστον τρεις, ώστε να υπολογιστεί η δισδιάστατη θέση ενός σημείου πάνω στη Γη και τουλάχιστον τέσσερις δορυφόροι για να υπολογιστεί η τρισδιάστατη θέση αυτού του σημείου , δηλαδή , γεωγραφικό πλάτος , μήκος και υψόμετρο από τη στάθμη της θάλασσας.

Το GPS αρχικά δημιουργήθηκε αποκλειστικά για στρατιωτική χρήση και ανήκε στη δικαιοδοσία του αμερικανικού Υπουργείου Εθνικής Άμυνας. Στα μέσα της δεκαετίας του 1960 το σύστημα δορυφορικής πλοήγησης, γνωστό τότε με την ονομασία Transit System, χρησιμοποιήθηκε ευρέως από το αμερικανικό ναυτικό. Απαιτήθηκαν αρκετές δεκαετίες, ώστε να εξελιχθεί το σύστημα GPS δηλαδή μέχρι τα μέσα της δεκαετίας του 1990, και να γίνει ιδιαίτερα ακριβές όπου άρχισε να διατίθεται για ελεύθερη χρήση από το ευρύ κοινό.

(2.3.3) **Λειτουργία GPS**

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

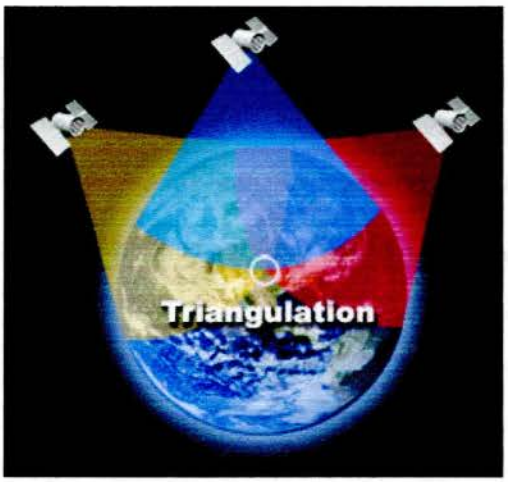
- **Διαστημικό τμήμα**: Αποτελείται από το δίκτυο των 24 - 32 δορυφόρων που ήδη αναφέραμε. Οι δορυφόροι αυτοί «σκεπάζουν» ομοιόμορφα με το σήμα τους ολόκληρο τον πλανήτη, γεγονός που αποδεικνύει τη φιλοσοφία που κρύβεται πίσω από τη λειτουργία του συστήματος GPS, δηλαδή τη διαθεσιμότητά του σε κάθε σημείο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



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

- **Επίγειο τμήμα ελέγχου**: Οι δορυφόροι, όπως είναι αναμενόμενο, είναι πολύ πιθανό να αντιμετωπίσουν ανά πάσα στιγμή προβλήματα στη σωστή λειτουργία τους. Οι έλεγχοι που πραγματοποιούνται σε αυτούς αφορούν τη σωστή τους ταχύτητα το υψόμετρο τους την κατάσταση του αποθέματος τους σε ηλεκτρική ενέργεια. Παράλληλα, εφαρμόζονται όλες οι διορθωτικές ενέργειες που αφορούν στο σύστημα χρονομέτρησης των δορυφόρων, ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος. Το τμήμα επίγειου ελέγχου αποτελείται από ένα επανδρωμένο και τέσσερα μη επανδρωμένα κέντρα, εγκατεστημένα σε ισάριθμες περιοχές του πλανήτη.Οι περιοχές των κέντρων αυτών είναι οι εξής: α) Κολοράντο (ΗΠΑ) β) Χαβάη (Ανατολικός Ειρηνικός Ωκεανός) γ) Ascension Island (Ατλαντικός Ωκεανός) δ) Diego Garcia (Ινδικός Ωκεανός) ε) Kwajalein (Δυτικός Ειρηνικός Ωκεανός).Ο κυριότερος σταθμός βάσης είναι αυτός του Κολοράντο, ο οποίος είναι μάλιστα και ο μοναδικός που βρίσκεται στην ξηρά. Αναλαμβάνει τον έλεγχο της σωστής λειτουργίας των υπόλοιπων τεσσάρων σταθμών, καθώς και τον συντονισμό τους. Σημειώνοντας τη θέση των σταθμών αυτών πάνω σε έναν παγκόσμιο χάρτη, παρατηρεί κανείς ότι η διάταξή τους δεν είναι τυχαία, αλλά ακολουθούν μια γραμμή παράλληλη με τα γεωγραφικά μήκη της Γης. Επίσης μια από τις κύριες αρμοδιότητες των κέντρων αυτών είναι ο απόλυτος και καθημερινός συντονισμός των ατομικών ρολογιών των δορυφόρων όπου καθημερινά επηρεάζονται από τις κοσμικές ακτινοβολίες και στην περίπτωση που συμβεί κάτι τέτοιο έστω δηλαδή ότι ένας από τους τρεις δορυφόρους που χρησιμοποιεί ένας επίγειος δεκτής GPS για να υπολογίσει το στίγμα του , έχει απόκλιση μερικά κλάσματα του δευτερόλεπτου από τους υπόλοιπους δυο δορυφόρους , τότε το σφάλμα της θέσης που υπολογίζετε φτάνει τα δυο χιλιόμετρα από την πραγματική του θέση.

- **Τμήμα δέκτη GPS:** Ο Δέκτης χρησιμοποιεί κυκλώματα εξαιρετικά χαμηλού θορύβου και ειδικές τεχνικές επεξεργασίας σήματος (DSP) ώστε να ξεχωρίζει τα εξαιρετικά ασθενή σήματα των δορυφόρους, από τον ισχυρό τηλεπικοινωνιακό θόρυβο ο οποίος έχει τη μορφή τυχαίου σήματος. Ο κυρίως δέκτης αποτελείται από το αναλογικό τμήμα εισόδου και το ψηφιακό, το οποίο περιέχει σύνθετο ψηφιακό υλικό (hardware), συνήθως από κάποιο εξειδικευμένο ολοκληρωμένο κύκλωμα τύπου ASIC και από ένα μικροελεγκτή (microcontroller) χαμηλής κατανάλωσης ισχύος. Αυτό το hardware χρησιμοποιεί λογισμικό με πολύ εξελιγμένους αλγορίθμους επεξεργασίας, για να μπορέσει να εξάγει χρήσιμο στίγμα υπό συνθήκες urban canyon ή γενικά δύσκολης λήψης. Σε τέτοιες περιπτώσεις, η ακρίβεια λήψης, λόγω των πολλαπλών σημάτων, τα οποία λαμβάνει η κεραία από τον ίδιο δορυφόρο, με χρονική καθυστέρηση μεταξύ τους (φαινόμενο ηχούς), μπορεί να υποβαθμίσει σημαντικά την ακρίβεια θέσης. Το αποτέλεσμα εξαρτάται έντονα από την ποιότητα των αλγορίθμων και βελτιώνεται σημαντικά από την μία γενιά δεκτών στην επόμενη.Ενδείξεις συσκευής πλοήγησης GPS.Η τελική έξοδος του δέκτη είναι το στίγμα (θέση) του και η ακριβής παγκόσμια ώρα UMT. Αυτά τα δύο δεδομένα, μαζί με άλλες χρήσιμες πληροφορίες όπως ο αριθμός των λαμβανόμενων δορυφορικών σημάτων και η στάθμη τους, αποστέλλονται σε μια θύρα επικοινωνίας του δέκτη, συνήθως σειριακής μορφής, δηλαδή ασύγχρονη (UART) ή σύγχρονη (π.χ. SPI). Ο ρυθμός με τον οποίο βγαίνει νέο στίγμα στην έξοδο του δέκτη είναι συνήθως 1 φορά το δευτερόλεπτο (δηλαδή 1 Hz), αν και υπάρχουν δέκτες που μπορούν να δίνουν στίγμα με ταχύτερους ρυθμούς (π.χ. 10 Hz).



(2.4) Θέση του οχήματος

Ένα από τα απαραίτητα στοιχεία που οδηγούν στη λύση του προβλήματος είναι η πραγματική θέση του οχήματος. Το όχημά είναι εξοπλισμένο με ένα δέκτη GPS , που παρέχει ανά πάσα στιγμή τις παρακάτω σημαντικές πληροφορίες :

- Τις συντεταγμένες θέσης του οχήματος , δηλαδή το γεωγραφικό πλάτος και γεωγραφικό μήκος του σημείου που βρίσκετε.
- Τη ταχύτητα εδάφους.
- Κατεύθυνση πορείας.
- Το Υψόμετρο
- Την ακρίβεια στίγματος
- Τον αριθμό των δορυφόρων που έχει κλειδώσει (των δορυφόρων που το σήμα είναι αρκετά ισχυρό ώστε οι πληροφορίες που μας δίνει είναι αρκετά αξιόπιστες) ο δέκτης GPS
- Και άλλες πληροφορίες που δεν είναι απαραίτητες στη συγκεκριμένη εφαρμογή.

Υπολογισμός απόστασης και διόπτευσης επιθυμητού σημείου (2.5)

(2.5.1) Ευκλείδεια γεωμετρία

Αν θεωρηθεί ότι σε ένα επίπεδο τοπίο πρέπει να υπολογιστεί η απόσταση δυο σημείων γνωρίζοντας τις συντεταγμένες αυτών ο πιο απλός τρόπος είναι να χρησιμοποιηθεί , η Ευκλείδεια γεωμετρία και συγκεκριμένα ο τύπος της Ευκλείδειας απόστασης για διάφορα συστήματα συντεταγμένων.

Ο τύπος Ευκλείδειας απόστασης σε καρτεσιανές συντεταγμένες προέρχεται από το Πυθαγόρειο θεώρημα. Αν (x1, y1) και (x2, y2) είναι τα σημεία στο επίπεδο, τότε η

απόσταση μεταξύ τους, που ονομάζεται Ευκλείδεια απόσταση, και δίνεται από τον μαθηματικό τύπο:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Γενικότερα, στην Ευκλείδεια γεωμετρία για ν-χώρο, η Ευκλείδεια απόσταση μεταξύ δύο σημείων, ορίζεται, με γενίκευση του πυθαγόρειου θεωρήματος, ως:

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}.$$

Υπολογισμός ορθοδρομικής απόστασής δυο συντεταγμένων πάνω στη Γη

(2.5.2) **Σφαιρική γεωμετρία**

Στην περίπτωση όμως που πρέπει να υπολογιστεί η ελάχιστη απόσταση δυο σημείων σε κυρτή επιφάνεια και συγκεκριμένα πάνω σε μια σφαίρα, όπως η Γη ,πρέπει να υπολογιστεί η ορθοδρομική απόσταση. Στην πραγματικότητα, η γη είναι ελαφρά ελλειψοειδής, χρησιμοποιώντας λοιπόν ένα σφαιρικό μοντέλο συνήθως το σφάλμα που δημιουργείται είναι το πολύ μέχρι 0,3% .

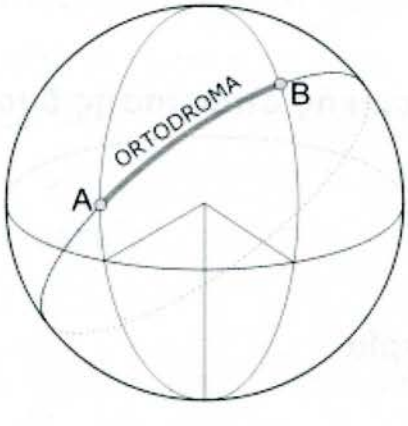
Ίσως στις αποστάσεις που έγιναν οι πειραματικές δοκιμές του συστήματος να μη γινόταν αντιληπτή η περισσότερη απόσταση που θα υπολογιζόταν χρησιμοποιώντα τον τύπο της Ευκλείδιας απόστασης , αλλά στη μελέτη που έγινε έπρεπε ν συμπεριληφθεί και ο παράγοντας της κυρτής επιφάνειας.

Για τους λόγους που αναφέρθηκαν ο μαθηματικός τύπος που χρησιμοποιήθηκ ονομάζετε τύπος Harvesine και υπολογίζει τη ορθοδρομική απόσταση δύο σημείω σε μια σφαίρα.

(2.5.3) **Ορθοδρομία**

Σε μία σφαίρα βραχύτερη διαδρομή μεταξύ δύο σημείων της επιφάνειάς της είναι τ τόξο του μέγιστου κύκλου που διέρχεται από τα δύο σημεία, λεγόμενη ορθοδρομία

Με τον ναυτικό όρο *ορθοδρομική πλεύση* ή *ορθοδρομικός πλους* ή απλούστερ **Ορθοδρομία** (spherical sailing, ή great circle sailing)* χαρακτηρίζεται ο πλου εκείνος που πραγματοποιείται σε τόξο μικρότερο των 180° επί του μεγίστου κύκλο (της επιφάνειας της Γης - θάλασσας) που ενώνει δύο τόπους, και που τελικά είναι μικρότερη μεταξύ αυτών των τόπων απόσταση.(*) Εκτός των Άγγλων οι περισσότερο ναυτικοί λαοί χρησιμοποιούν τον ελληνικό όρο "Orthodromia".



Ορθοδρομική απόσταση από το σημείο Α στο Β

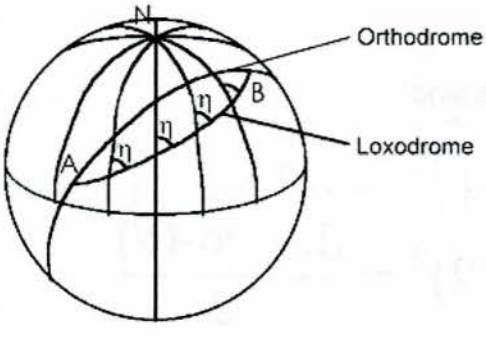
Είναι γνωστό από τη Στερεομετρία τόξο μεγίστου κύκλου μεταξύ δύο σημείων τη επιφάνειας μιας σφαίρας είναι η συντομότερη απόσταση μεταξύ των σημείων αυτών.

Έτσι το είδος αυτό της πλεύσης επειδή συμβαίνει επί μεγίστου κύκλου καλείται και "πλους μεγίστου κύκλου". Το δε μέτρο του ορθοδρομικού τόξου εκφρασμένο σε πρώτα της μοίρας αποδίδει ακριβώς την απόσταση σε ναυτικά μίλια.Το χαρακτηριστικό της ορθοδρομίας είναι ότι αυτή στρέφει το κυρτό της προς τους πόλους, και το κοίλο της προς τον Ισημερινό ενώ τέμνει τους μεσημβρινούς υπό διαφορετικές γωνίες, αφού οι μεσημβρινοί της Γης δεν είναι παράλληλοι, αλλά συγκλίνοντας τέμνονται στους Πόλους.

- Για την εύρεση της ορθοδρομικής απόστασης χρησιμοποιούνται ειδικοί πίνακες του Βρετανικού Ναυαρχείου, οι καλούμενοι "H.O. 214", ή της Αμερικανικής Υδρογραφικής, οι καλούμενοι "H.D. 486".
- Η **Ορθοδρομία** και η Λοξοδρομία αποτελούν τα δύο κύρια και βασικά είδη πλεύσεων των πλοίων ειδικά σε μεγάλες αποστάσεις.

(2.5.4) **Ιδιότητες Ορθοδρομίας**

- Η Ορθοδρομία στο Μερκατορικό χάρτη παρίσταται ως καμπύλη γραμμή, με το κυρτό της στραμμένο προς τους Πόλους. Συνεπώς τέμνει τους (παράλληλους) μεσημβρινούς υπό διαφορετική γωνία, από σημείου αναχώρησης μέχρι σημείου άφιξης, σε αντίθεση της Λοξοδρομίας που παρίσταται σε ίδιο ναυτικό χάρτη ως ευθεία.
- Η Ορθοδρομική πλεύση έχει το σημαντικό πλεονέκτημα της διάνυσης της συντομότερης απόστασης από τόπο εκκίνησης στο τόπο άφιξης, αλλά συγχρόνως και το μειονέκτημα της συνεχούς αλλαγής της πορείας σε τακτά χρονικά διαστήματα.
- Για μικρές αποστάσεις, η διαφορά μεταξύ Ορθοδρομίας και Λοξοδρομίας είναι αμελητέα. Έτσι αφού το κέρδος δεν είναι ουσιώδες προτιμάται η λοξοδρομική πλεύση με το πλεονέκτημα της δια τήρησης σταθερής πορείας. Συνήθως για τα μέσα γεωγραφικά πλάτη μέχρι 600 ν.μ. η πλεύση είναι πάντα λοξοδρομική.



■Η Ορθοδρομία και η Λοξοδρομία συμπίπτουν στη περίπτωση πλόων επί

του Ισημερινού, καθώς και επί των μεσημβρινών, δηλαδή για πορείες 000° ή 180° για τη δεύτερη περίπτωση. Το γεγονός αυτό έχει ως συνέπεια σε πλόες που ο τόπος προορισμού έχει μικρή "διαφορά γεωγραφικού μήκους", Δλ,(δείτε σχετικά 2ο υπολογισμό στο γεωγραφικό στίγμα) και ανεξάρτητα του γεωγραφικού πλάτους ακολουθείται η λοξοδρομία. Σε πορείες πλοίων προς Απηλιώτη (Α) ή προς Ζέφυρο (Β) παρατηρείται η μεγαλύτερη διαφορά μεταξύ Λοξοδρομίας και Ορθοδρομίας. Η διαφορά αυτή αυξάνει σημαντικά με την αύξηση του γεωγραφικού πλάτους. Έτσι όταν οι δύο τόποι αναχώρησης και προορισμού βρίσκονται πλησίον του Ισημερινού αυτή να είναι αμελητέα.

(2.5.5) Μαθηματικός τύπος Harvesine

Ο τύπος Haversine είναι μια εξίσωση πολύ σημαντική για την πλοήγησης, υπολογίζοντας την ορθοδρομική απόσταση μεταξύ δύο σημείων σε μια σφαίρα, από το μήκη και πλάτη τους. Πρόκειται για μια ειδική περίπτωση ενός γενικότερου τύπου σφαιρικής τριγωνομετρίας.Ο νόμος Haversine, αφορά τις πλευρές και γωνίες σφαιρικών "τριγώνων".

Ιστορικά, η haversine είχε δημοσιευθεί από RW Sinnott στο Sky & Telescope, το 1984, αν και ήταν γνωστός εδώ και πολύ περισσότερο καιρό στους πλοηγούς.

Μαθηματικός τύπος Harvesine :

$$\operatorname{haversin}\left(\frac{d}{r}\right) = \operatorname{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\operatorname{haversin}(\psi_2 - \psi_1)$$

όπου:

- Haversin είναι η συνάρτηση Haversine:

$$\operatorname{haversin}(\theta) = \sin(\theta/2)^2 = \frac{1-\cos(\theta)}{2}$$

- "d" είναι η Ορθοδρομική απόσταση ανάμεσα σε δύο σημεία μιας σφαίρας
- "r" είναι η ακτίνα της σφαίρας
- ϕ_1, ϕ_2: Το γεωγραφικό πλάτος του σημείου 1 και το γεωγραφικό πλάτος του σημείου 2
- ψ_1, ψ_2: Γεωγραφικό μήκος του σημείου 1 και γεωγραφικό μήκος του σημείου 2

Στην αριστερή πλευρά του συμβόλου ίσον, το όρισμα της συνάρτησης haversine είναι σε ακτίνια. Για να μετατραπεί σε μοίρες, haversin (d / R) ο τύπος θα γίνει haversin (180 ° d / πR).

Μπορεί κανείς να λύσει ως προς "d" είτε με την απλή εφαρμογή του αντιστρόφου haversine (εάν υπάρχει) είτε χρησιμοποιώντας τη συνάρτηση τόξου ημίτονου (αντίστροφο ημίτονο):

$$d = r\,\mathrm{haversin}^{-1}(h) = 2r\arcsin\left(\sqrt{h}\right)$$

οπού :

h είναι Haversin(d/R)

$$d = 2r\arcsin\left(\sqrt{\mathrm{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\,\mathrm{haversin}(\psi_2 - \psi_1)}\right)$$

$$= 2r\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right)}\right)$$

(2.5.6) **Ο νόμος Haversines**

Σε μια μοναδιαία σφαίρα, ένα «τρίγωνο» στην επιφάνεια της σφαίρας ορίζεται από τις Ορθοδρομικές ευθείες που συνδέουν τα τρία αυτά σημεία u, v, w. Αν τα μήκη των τριών πλευρών είναι "α" (από το u στο v), "b" (από το u για να w), και "c" (από το v σε w), και η γωνία C βρίσκετε απέναντι από τη πλευρά c τότε ο νόμος των haversines ειναι:

$$\operatorname{haversin}(c) = \operatorname{haversin}(a-b) + \sin(a)\sin(b)\operatorname{haversin}(C).$$

Δεδομένου ότι πρόκειται για μοναδιαία σφαίρα, τα μήκη α, β, γ είναι ίσα με τις γωνίες (σε ακτίνια) που τέμνονται από εκείνες τις πλευρές από το κέντρο της σφαίρας (για μια μη μοναδιαία σφαίρα, κάθε ένα από αυτά τα μήκη τόξου είναι ίσα με τη κεντρική γωνία του πολλαπλασιάζεται από την ακτίνα της σφαίρας). Για να προκύψει ο μαθηματικός τύπος Haversine που αναφέρθηκε παραπάνω, αρκεί να λάβει κανείς υπόψη οτι στην περίπτωσή που αντί για σφαίρα έχουμε τη Γη, θεωρείτε το σημείο "u" ο βόρειος πόλος, ενώ "v" και "w" είναι τα δύο σημεία των οποίων είναι επιθυμητό να υπολογιστεί η απόσταση τους "d". Στη περίπτωση αυτή, α και β είναι: π / 2 - φ1, 2 (δηλαδή, 90 ° - γεωγραφικό πλάτος), "C" είναι η διαφορά Δλ του γεωγραφικού μήκους, και το "c" είναι η επιθυμητή απόσταση ως προς την ακτίνα της Γης d / R. Σημειώνοντας ότι sin(π / 2 - φ) = cos (φ), και έτσι παράγετε ο μαθηματικός τύπος Haversine.

Για να προκύψει ο νομός haversines, κάποιος πρέπει να ξεκινήσει με το νόμο των σφαιρικών συνημίτονων:

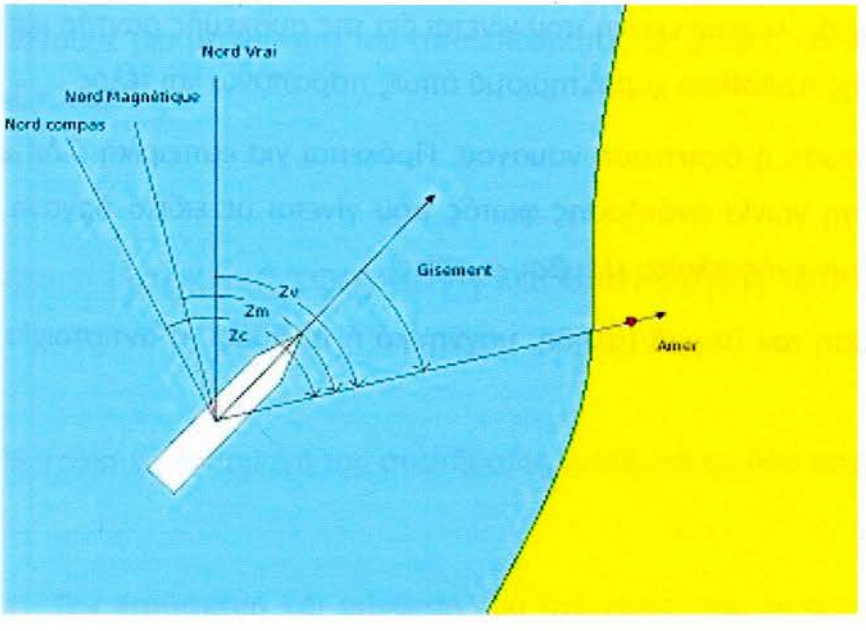
$$\cos(c) = \cos(a)\cos(b) + \sin(a)\sin(b)\cos(C).$$

Υπολογισμός διόπτευσης στόχου (2.6)

(2.6.1) **Διόπτευση**

Διόπτευση (bearing) λέγεται, η κατά συγκεκριμένη κατεύθυνση, παρατήρηση ενός αντικειμένου.Τόσο στη ναυτιλία, όσο και γενικότερα, η κατεύθυνση παρατήρησης αντικειμένου προσδιορίζεται δια του κέντρου του ανεμολογίου πυξίδας. Συνεπώς η διόπτευση μετριέται σε μοίρες (°).

Υφίστανται πολλών ειδών διόπτευσης, τόσο κατά αντικείμενο παρατήρησης όσο και εκ του λαμβανόμενου σημείου αρχής μέτρησης της διόπτευσης, καθώς και εκ του οργάνου με το οποίο πραγματοποιείται* αυτή:



Σχεδιάγραμμα διαφόρων διοπτεύσεων του σημείου Amer.

Zc: Διόπτευση πυξίδας, **Zm**: Μαγνητική διόπτευση, **Zv**: Αληθής διόπτευση, και **Gisement**: Σχετική διόπτευση

- Αληθής ή αστρονομική διόπετυση (true bearing): λέγεται εκείνη που λαμβάνεται, στο ανεμολόγιο, με αρχή μέτρησης τον αληθή Βορρά.
- Μαγνητική διόπτευση) (magnetic bearing): λέγεται εκείνη με αρχή μέτρησης τον μαγνητικό Βορρά.

- Διόπτευση πυξίδας (compass bearing): λέγεται εκείνη που λαμβάνεται από τον Βορρά που δείχνει η πυξίδα ή Βορρά πυξίδας.
- Απόλυτος διόπτευση : χαρακτηρίζεται οποιαδήποτε διόπτευση από 000° μέχρι 360°
- Τεταρτοκυκλική διόπτευση: χαρακτηρίζεται οποιαδήποτε διόπτευση από Βορρά ή Νότο προς Ανατολή ή Δύση με όριο μέτρησης 0° - 90° (ανατολικά ή δυτικά)
- Σχετική διόπτευση: λέγεται η διόπτευση που λαμβάνεται από την πλώρη του πλοίου, (ανεξάρτητα διεύθυνσης Βορρά) είτε περιφερειακά 0° - 360°, είτε ημικυκλικά 0° - 180° δηλαδή δεξιά ή αριστερά του πλοίου.
- Ραδιοδιόπτευση" ονομάζεται εκείνη που γίνεται με ραδιογωνιόμετρο και που χαρακτηρίζεται αντίστοιχα αληθής, ή μαγνητική, ή πυξίδας ή απόλυτη.
- Διόπτευση ραντάρ λέγεται εκείνη που γίνεται δια της συσκευής ραντάρ και που λαμβάνει επίσης πρόσθετο χαρακτηρισμό όπως παραπάνω και τέλος
- Σωσίβια διόπτευση ή διόπτευση ναυαγού: Πρόκειται για εμπειρική διόπτευση στηριζόμενη στη γωνία ανάκλασης φωτός που γίνεται με ειδικό όργανο των σωστικών μέσων ενός πλοίου (λέμβοι, σχεδίες).

Στη διόπτευση με βάση τον Βορρά (αληθή, μαγνητικό ή πυξίδας), η αντιστοιχία έχει ως εξής:

- 000°: Βορράς
- 090°: Ανατολή
- 180°: Νότος
- 270°: Δύση

(2.6.2) **Μαθηματικός υπολογισμός Διόπτευσης**

Η τρέχουσα πορεία μεταβάλεται καθώς το όχημα ακολουθεί μια ορθοδρομική πορεία ανάμεσα σε δυο σημεία. Η τελική πορεία θα διαφέρει από την αρχική με κάποια διάφορα μοιρών που είναι , ανάλογη με την απόσταση και το γεωγραφικό πλάτος. (για παράδειγμα αν κάποιος ξεκινήσει μια ορθοδρομική πορεία από τις συντεταγμένες 35 ° N, 45 ° E (Βαγδάτη) προς τις συντεταγμένες 35 ° N, 135 ° E

(Οσάκα), θα πρέπει να ξεκινήσει με πορεία στις 60 ° και να καταλήξει με πορεία στις 120 °).

Ο μαθηματικός τύπος που χρησιμοποιήθηκε υπολογίζει τη αρχική διόπτευση (και μερικές φορές αναφέρεται ως αζιμούθιο προς τα εμπρός), με την οποία αν ακολουθήσουμε μια ευθεία γραμμή κατά μήκος ενός Ορθοδρομικού τόξου, θα μας μεταφέρει από το σημείο εκκίνησης στο σημείο τερματισμού είναι ο εξής :

$$\theta = \text{atan2}(\ \sin(\Delta\text{long}).\cos(\text{lat}_2).\cos(\text{lat}_1).\sin(\text{lat}_2) - \sin(\text{lat}_1).\cos(\text{lat}_2).\cos(\Delta\text{long})\ )$$

Δεδομένου ότι το atan2 μας επιστρέφει τιμές με εύρος -π ... + π (δηλαδή -180 ° ... 180 °), για να μετατραπεί το αποτέλεσμα σε μια διόπτευση πυξίδας πρέπει να κάνουμε μια μετατροπή του αποτελέσματος σε μοίρες και να χρησιμοποιήσουμε τον (θ+360).

(2.6.3) Πρακτική εφαρμογή και υλοποίηση του συστήματος

Η πρακτική εφαρμογή του συστήματος χωρίζεται σε δύο υποκατηγορίες :

1. Την κατασκευή και σύνθεση του οχήματος που περιέχει όλα τα αισθητήρια και ενεργοποιητές που είναι απαραίτητα για το όχημα μας ώστε να είναι εφικτή η αυτόματη ανεύρεση διαδρομής , και η πλοήγηση του συστήματος

2. Την υλοποίηση της εφαρμογής επίγειου σταθμού ελέγχου (Ground Control Station (GCS)) όπου αναλαμβάνει να συλλέξει όλες τις απαραίτητες πληροφορίες από τα αισθητήρια του οχήματος και σύμφωνα με αυτά να δώσει τις κατάλληλες εντολές ώστε να επιτευχθεί ο σκοπός του συστήματος .

Επιλογή εξοπλισμού και υλικών (3.1)

(3.1.1) Προδιαγραφές

Οι **Προδιαγραφές** που πρέπει να καλύπτει το όχημα είναι οι εξής:

1. Το όχημα θα έπρεπε να έχει την δυνατότητα να κινηθεί προς όλους τους άξονες X,Y,Z.

2. Το όχημα θα έπρεπε να έχει αρκετή αυτονομία ώστε να δυνατή η ολοκλήρωση των δοκίμων και πειραμάτων που είναι απαραίτητα για την επίτευξη της μελέτης.

3. Το όχημα πρέπει να έχει τη κατάλληλη ισχύ για να μεταφέρει τα ηλεκτρονικά εξαρτήματα και τα κατάλληλα αισθητήρια που θα προσαρμοστούν πάνω του ώστε να είναι διαθέσιμες οι μετρήσεις που είναι απαραίτητες
4. Το κόστος οχήματος θα έπρεπε να είχε τη καλύτερη δυνατή αναλογία ποιότητας - κόστους, ενώ το κόστος του να είναι εντός των οικονομικών δυνατοτήτων που διατίθενται.
5. Να υπάρχει πρόσβαση να κάνουμε οποιαδήποτε αλλαγή (Modification) στο Hardware και Firmware του μοντέλου.
6. Να υπάρχουν τα εξαρτήματα του διαθέσιμα άμεσα σε περίπτωση καταστροφή τους.

Δυνατότητες και χαρακτηριστικά του οχήματος που επιλέχτηκε (3.2)

Το όχημα που επιλέχτηκε είναι τύπου ελικοπτέρου με τέσσερις έλικες (QuadRotor)

(3.2.1) Quadrotor ελικόπτερο

Το Quadrotor ελικόπτερο, ή Quadrocopter όπως αλλιώς ονομάζεται, είναι ένα αεροσκάφος που αιωρείται και προωθείται από τέσσερις έλικες. Τα Quadrotors έχουν ταξινομηθεί ως στροφειόπτερα, γιατί σε αντίθεση με τα αεροσκάφη σταθερής πτέρυγας, η ώση τους προέρχεται από τέσσερις έλικες. Μπορούν επίσης να χαρακτηριστούν ως ελικόπτερα, αν και σε αντίθεση με τα τυπικά ελικόπτερα, τα quadrotors χρησιμοποιούν σταθερού βήματος έλικες, των οποίων η κλίση των στροφείων δεν μεταβάλλεται καθώς οι έλικες περιστρέφονται. Ο έλεγχος της κίνησης του οχήματος μπορεί να επιτευχθεί με τη μεταβολή της σχετικής ταχύτητας του κάθε στροφείου με αποτέλεσμα να αλλάζει η ώθηση και η ροπή που παράγεται από το κάθε έλικα.

Υπάρχουν δύο γενιές σχεδίων για quadrotor ελικόπτερο. Τα πρώτης γενιάς quadrotors σχεδιάστηκαν για να μεταφέρουν έναν ή περισσότερους επιβάτες. Τα οχήματα αυτά ήταν από τα πρώτα επιτυχή οχήματα κάθετης απογείωσης και προσγείωσης (VTOL). Ωστόσο, τα πρώτα πρότυπα υπέφεραν από κακή απόδοση, και στα τελευταία πρωτότυπα απαιτούνταν πάρα πολύ μεγάλες πιλοτικές ικανότητες από το χειριστή του οχήματος, λόγω της μεγάλης αστάθειας του οχήματος .

Η πιο πρόσφατη γενιά quadrotors σχεδιάστηκαν για χρήση ως μη επανδρωμένα εναέρια οχήματα (UAV). Αυτά τα οχήματα χρησιμοποιούν ένα ηλεκτρονικό σύστημα ελέγχου καθώς και ηλεκτρονικά αισθητήρια για τη σταθεροποίηση του αεροσκάφους. Με το μικρό τους μέγεθος, την ευελιξία τους, και τη δυνατότητα ελιγμών, έχουν την ικανότητα να πετάξουν σε εσωτερικούς καθώς και σε εξωτερικούς χώρους.

(3.2.2) Συγκρίσεις QuadRotor ελικοπτέρου - Αεροπλάνου

Πλεονεκτήματα:

1. Το **QuadRotor ελικόπτερο** έχει την δυνατότητα να παραμείνει στο ίδιο σημείο μέχρι να τελειώσουν τα αποθέματα ενέργειας που διαθέτει, ενώ το αεροπλάνο πρέπει να κινηθεί με κάποια ελάχιστη ταχύτητα πλεύσης για να παραμείνει στον αέρα.
2. Το **QuadRotor ελικόπτερο** μπορεί να απογειωθεί και να προσγειωθεί σε ελάχιστο χώρο σε σχέση με τον χώρο που απαιτείται για το αεροπλάνο.
3. Το **QuadRotor ελικόπτερο** με ένα βέλτιστο έλεγχο ισορροπίας μπορεί να πετάξει και να δοκιμαστεί ακόμα και σε κλειστό χώρο ώστε να γίνουν τα κατάλληλα πειράματα και εξομοιώσεις για την ανάπτυξη της μελέτης.

Μειονεκτήματα:

1. Μικρότερη αυτονομία σε σχέση με το αεροπλάνο
2. Μικρότερη ταχύτητα πτήσης από το αεροπλάνο
3. Περισσότεροι κραδασμοί στο σκελετό του οχήματος

(3.2.3) **Συγκρίσεις QuadRotor ελικοπτέρου - τυπικού ελικοπτέρου**

Πλεονεκτήματα:

1.Το QuadRotor ελικόπτερο έχει πολύ πιο απλή μηχανική κατασκευή με αποτέλεσμα να απλοποιείται η σχεδίαση του οχήματος, και να μειώνετε ο χρόνος και το κόστος συντήρησης.

2.Το **QuadRotor** ελικόπτερο με έναν καλό αλγόριθμο ελέγχου και τα κατάλληλα αισθητήρια είναι πολύ πιο σταθερό από ένα απλό ελικόπτερο γιατί υπάρχουν 4 σημεία στήριξης .

3. Το **QuadRotor** ελικόπτερο έχει λιγότερες τριβές στο σύστημα κίνησης άρα , μεγαλύτερη αυτονομία και λιγότερη φθορά υλικών.

4. Το **QuadRotor** ελικόπτερο μπορεί να κατευθυνθεί προς όλες τις κατευθύνσεις με την ίδια ταχύτητα γιατί η κατασκευή του είναι συμμετρική προς όλες τις πλευρές.

5. Το **QuadRotor** ελικόπτερο μπορεί να κινηθεί με πιο επιθετικούς ελιγμούς από ένα απλό ελικόπτερο και αυτό μας δίνει την δυνατότητα να κάνουμε πτήση σε περιβάλλον με δυνατότερους ανέμους.

6. Το **QuadRotor** ελικόπτερο έχει απλούστερο σύστημα διεύθυνσης γιατί βασίζεται στον αλγόριθμο ελέγχου και έτσι σε περίπτωση μη επιθυμητής πτώσης οι ζημιές που θα πάθει είναι λιγότερες από ότι σε ένα κανονικό ελικόπτερο , λόγω της πολυπλοκότητας του μηχανικού συστήματος που διαθέτει στο σύστημα διεύθυνσης.

7. Το **QuadRotor** ελικόπτερο επειδή χρησιμοποιεί τέσσερις έλικες ο κάθε έλικας έχει μικρότερη διάμετρο από το ισοδύναμο έλικα του τυπικού ελικοπτέρου, με αποτέλεσμα να αποθηκεύεται λιγότερη κινητική ενέργεια κατά τη διάρκεια της πτήσης. Αυτό

μειώνει τη ζημιά που μπορεί να προκληθεί σε περίπτωση που ένας από τους έλικες συγκρουστεί σε οποιαδήποτε αντικείμενο.

Μειονεκτήματα:

1. Το **QuadRotor** απαιτεί αρκετά πολύπλοκο ηλεκτρονικό έλεγχο σταθεροποίησης σε σχέση με το τυπικό ελικόπτερο.

Το QuadRotor ελικόπτερο που επιλέχτηκε (3.3)

Το τελικό μοντέλο **QuadRotor** που επιλέχτηκε είναι το **Ar. Drone** της εταιρείας **Parrot** και αυτό έγινε σύμφωνα με τα κριτήρια που αναφέρθηκαν στις προηγούμενες παραγράφους .

Σημαντικά προτερήματα και χαρακτηριστικά του Ar. Drone σε σχέση με άλλα Quadcopters της ίδιας κατηγορίας κόστους.

- Ο αλγόριθμος ελέγχου ισορροπίας και κατεύθυνσης που συνθέτεται από τέσσερις ελεγκτές PID (έναν για κάθε βαθμό ελευθερίας Pitch , Roll , Yaw , Altitude) είναι σημαντικά καλύτερος.

- Η επικοινωνία με το όχημα γίνεται μέσω ασύρματου δικτύου WiFi που είναι ένας πολύ κοινότυπος τύπος δικτύωσης με αποτέλεσμα να εκτελεστεί η εφαρμογή που

θα σχεδιαστεί σε ένα κοινό υπολογιστή χωρίς την απαίτηση ειδικού εξοπλισμού για τον χειρισμό του οχήματος .

- Το Ar.Drone χρησιμοποιεί τη μια από τις δύο κάμερες που διαθέτει σαν αισθητήριο περνώντας την εικόνα που λαμβάνει μέσα από μια διαδικασία που ονομάζεται "**Estimation of the optical flow**" και σαν έξοδο δίνει την κατεύθυνση που μετακινούνται τα pixels και την ταχύτητα με την οποία κινούνται

- Να υπάρχουν διαθέσιμα ανταλλακτικά στο εμπόριο για οποιαδήποτε υλική ζημία συμβεί κατά τις δοκιμές.

Ar.Drone απο την Parrot (3.3.1)

Το **Ar.Drone** παρουσιάστηκε για πρώτη φορά στο Λας Βέγκας στη Διεθνούς φήμης έκθεση Consumer Electronics Show (CES) το 2010. Είναι κατασκευασμένο από πλαστικό και αφρώδες υλικό. Συνδέεται με iPhone / iPod touch / iPad

χρησιμοποιώντας το Wi-Fi για τον ελέγχο του, είναι εξοπλισμένο με δύο κάμερες που μπορούν να προβάλλουν την εικόνα τους μέσω του iPhone / iPod / iPad , και με αισθητήρια που σε συνδυασμό με ένα πολύπλοκο αλγόριθμο ελέγχου ,δίνει την δυνατότητα μιας πολύ σταθερής πτήσης.

Τεχνικά χαρακτηριστικά

Ενσωματωμένο υπολογιστικό σύστημα

- ARM9 468 MHz
- DDR 128 Mbyte at 200 MHz
- Wi-Fi b/g
- USB high speed
- Linux OS

Αδρανειακά αισθητήρια καθοδήγησης

- 3-αξόνων επιταχυνσιόμετρο
- 2 -αξόνων γυροσκόπιο
- 1-αξόνων γυροσκόπιο μεγάλης ακρίβειας

Πτητικά χαρακτηριστικά

- Ταχύτητα πλεύσης: 5 m/s; 18 km/h
- Βάρος: 380 g με το προστατευτικό εξωτερικής πτήσης , 420 g με το προστατευτικό εσωτερικής πτήσης

- Διάρκεια πτήσης: περίπου 12 λεπτά

Συστήματα προστασίας

- EPP hull for indoor flight
- Αυτόματο σύστημα απενεργοποίησης κινητήρων σε περίπτωση σύγκρουσης ή επαφής των ελίκων με κάποιο αντικείμενο
- μπαταρία τύπου UL2054
- Διακόπτης άμεσης απενεργοποίησης των κινητήρων

Αεροναυτική δομή

- Έλικες υψηλής απόδοσης
- Δομή κύριου σκελετού από ανθρακόνημα (Carbon-fiber)

Κινητήρες και παροχή ενεργείας

- 4 brushless κινητήρες, (35,000 rpm, power: 15 W)
- Μπαταρία πολυμερών λιθίου (3 cells, 11.1 V, 1000 mAh)
- Μέγιστος ρυθμός αποφόρτισης: 10C
- χρόνος φόρτισης μπαταρίας: 90 λεπτά

Εμπρόσθια κάμερα

- Ευρυγώνια κάμερα με 93 ° ευρυγώνιος φακός και, αισθητήρα CMOS
- Κωδικοποίηση και live streaming των εικόνων στο iPhone
- Ανάλυση κάμερας 640x480 pixels (VGA)

Αισθητήρας μέτρησης ύψους υπερήχων

- Συχνότητα εκπομπής 40 kHz
- Εμβέλεια 6 μέτρων (19.7 ft)

Κάθετη κάμερα

- Κάμερα υψηλής ταχύτητας 60 fps . 64° ευρυγώνιος φακός, και, αισθητήρα CMOS

Επιπλέον υλικά και εξοπλισμός (3.4)

Επιλογή πρόσθετου εξοπλισμού για την επίτευξη του στόχου μας

Το **Parrot Ar. Drone** σαν όχημα περιέχει αρκετούς αισθητήρες και πολύ καλό αλγόριθμο σταθεροποίησης, αλλά λόγω της εμπορικής του χρήσης δεν είναι δυνατό να γίνει προσθήκη περαιτέρω ηλεκτρονικού εξοπλισμού όπως ένας δέκτη GPS και ένα αισθητήριο μαγνητικού πεδίου που μας είναι απαραίτητα για την αυτόματη πλοήγηση .Η λύση δόθηκε με το να υπάρχουν δύο διαφορετικά υποσυστήματα πάνω στο όχημα τα οποία ναι μεν συνεργάζονται μεταξύ τους , αλλά έμμεσα και όχι άμεσα , δηλαδή δεν υπάρχει απευθείας σύνδεση μεταξύ τους.

Ο τρόπος που συνεργάζονται αυτά τα δύο υποσυστήματα είναι δια μέσω του σταθμού εδάφους.

(3.4.2) **Προδιαγραφές μικρο ελεγκτή που θα τοποθετηθεί στο όχημα**

Οι προδιαγραφές που πρέπει να πληρεί ο μικροελεγκτής που θα επιλεχθεί, είναι οι εξής:

- Ο προγραμματισμός του να γίνεται εύκολα χωρίς να είναι απαραίτητος πρόσθετος εξοπλισμός.
- Να είναι αρκετά γρήγορος στην επεξεργασία των δεδομένων ώστε να βγάλει εις πέρας τις ανάγκες του συστήματος.
- Να υπάρχει αρκετή βιβλιογραφία διαθέσιμη για την εκτέλεση των διαφόρων διαδικασιών που είναι απαραίτητες.
- Να υπάρχουν πλήθος βιβλιοθηκών προγραμματισμού ώστε να μην είναι απαραίτητο να προγραμματιστούν εξαρχής οι δευτερεύουσες διαδικασίες.
- Να μπορεί να συνδεθεί και να συλλέγει σήματα από τύπους αισθητηρίων χωρίς πολλές μετατροπές και προσθήκες ηλεκτρονικών κυκλωμάτων.
- Να είναι δυνατή η επέκταση του συστήματος και η παραμετροποιίσημη για τη συμβατότητα διαφορετικού τύπου οχημάτων.

Πλατφόρμα Μικρο-ελεγκτή ArduPilot Mega (3.5)

Η εργασία που εκτελεί ο μικρο ελεγκτής ArduPilot Mega είναι να συλλέγει τις απαραίτητες για τη πλοήγηση πληροφορίες από τα αισθητήρια και να τις στέλνει στον επίγειο σταθμό εδάφους διαμέσου ασύρματης επικοινωνίας.

Η επιλογή του Ardupilot Mega (APM) που δόθηκε σαν λύση ίσως είναι λίγο ακριβή σε σχέση με άλλους μικρο ελεγκτές γενικού σκοπού που κυκλοφορούν στο εμπόριο αλλά η επιλογή του έγινε συμπεριλαμβάνοντας τη παράμετρο αλλαγής ή αναβάθμισης του οχήματος. Με λίγα λόγια το σύστημα που σχεδιάστηκε έχει την δυνατότητα να προσαρμοστεί σε οποιοδήποτε είδος και μέγεθος οχήματος .Οι

πληροφορίες που είναι απαραίτητες για να γίνει η πλοήγηση του οχήματος είναι οι εξής:

1. Η γεωγραφική θέση του οχήματος μας η οποία δίνεται από το δέκτη GPS.

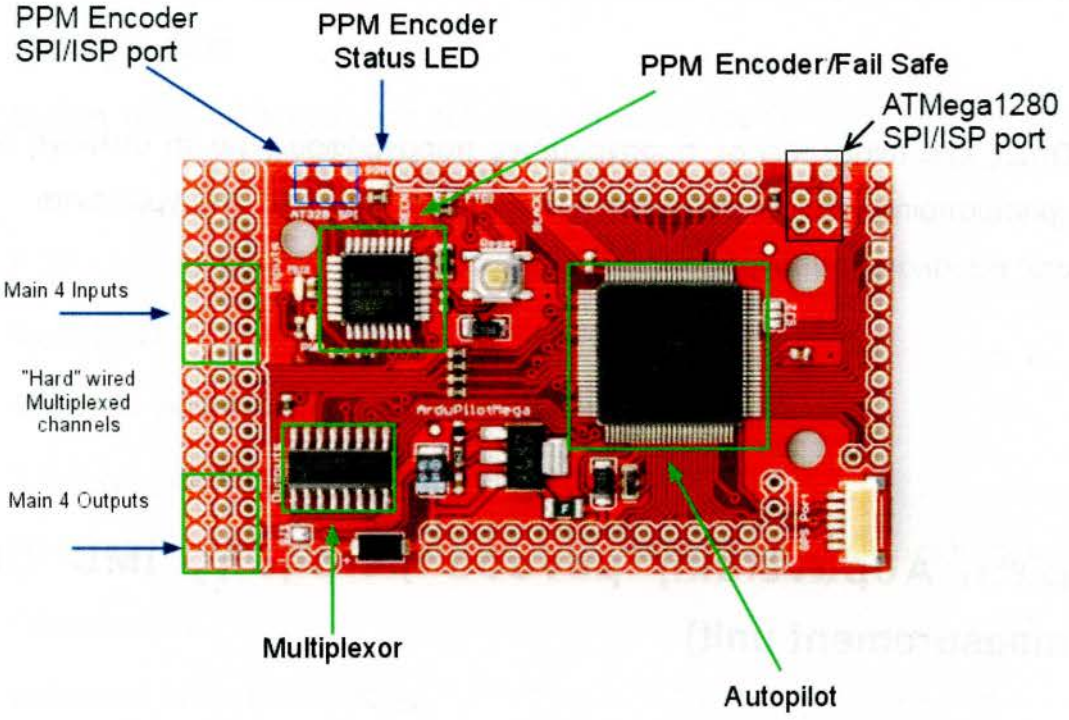
2. Μετρήσεις από 3 αισθητήρια πυξίδας τοποθετημένα σε τέτοια θέση ώστε να να είναι γνωστή η κατεύθυνση του μαγνητικού βορά (αζιμούθιο) ανεξάρτητα απο την γωνία πτήσης.

Πέρα απο τις παραπάνω πληροφορίες που είναι απαραίτητες για τη πλοήγηση του οχήματος , το λογισμικό που εκτελεί ο μικροελεγκτής **ArduPilot Mega** συλλέγει και άλλες πληροφορίες που είναι χρήσιμες αλλά όχι **απαραίτητες.** Αυτές είναι οι εξής:

- Τον αριθμό δορυφόρων από τους οποίους ο δέκτης GPS θεωρεί το σήμα τους αξιόπιστο για να το χρησιμοποιήσει στον υπολογισμό της θέσης του.
- Την ακρίβεια του υπολογισμού θέσης.
- Το υψόμετρο που υπολογίζεται απο το δέκτη GPS.
- Το υψόμετρο που μετριέται απο το αισθητήριο ατμοσφαιρικής πίεσης.
- Τη τάση της μπαταρίας.
- Τη ταχύτητα εδάφους που υπολογίζεται απο το δέκτη GPS.
- Τη πορεία του οχήματος ως προς τον πραγματικό Βορρά που υπολογίζεται από το δέκτη GPS
- Τη στάση του οχήματος (attitude), δηλαδή Pitch ,Roll , Yaw που υπολογίζεται απο τις μετρήσεις των τριών επιταχυνσιομέτρων (έναν για κάθε άξονα) και από τα τρία γυροσκόπια (ένα για κάθε άξονα) που είναι συνδεδεμένα στο μικροελεγκτή.

\

(3.5.1) Μικρο ελεγκτής ArduPilotMega



Τεχνικά χαρακτηριστικά :

- 16MHz Atmega 2560 processor.
- Περιλαμβάνει την ικανότητα να επανεκκίνησης του κύριου επεξεργαστή κατά τη διάρκεια της πτήσης
- σχεδίαση διπλού επεξεργαστή με ισχύ 32 MIPS
- 256k Flash Program Memory, 8K SRAM, 4K EEPROM
- Σειριακή σύνδεση με 6-pin σύνδεσμο για GPS
- 16 αναλογικές (με μετατροπέα ADC στη κάθε είσοδο) και 40 ψηφιακές εισόδους/εξόδους
- 4 σειριακές θύρες.

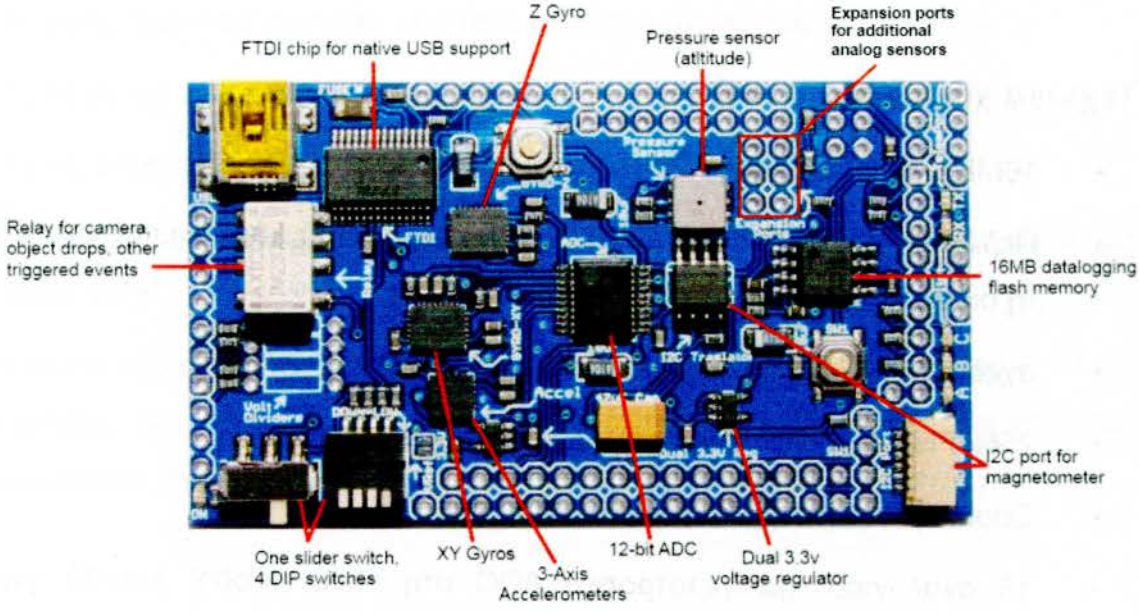
- Αποπλέκτη σημάτων PPM για την αποφυγή φόρτωσης με επιπλέον διεργασίες του κύριου επεξεργαστή
- 8 εισόδους PWM για σύνδεση με δέκτη RC
- LEDs ένδειξης κατάστασης

Αισθητήρια που επιλέχτηκαν για τη συλλογή δεδομένων

(3.6)

Όπως έχει αναφερθεί σε προηγούμενες παραγράφους, για τη συλλογή δεδομένων χρησιμοποιήθηκαν τα απαραίτητα αισθητήρια τα οποία περιγράφονται αναλυτικά στις παρακάτω παραγράφους.

(3.6.1) Αδρανειακή μονάδα μέτρησης IMU (inertial measurement unit)



Τεχνικά χαρακτηριστικά:

- 2 ρυθμιστές τάσης 3.3V (όπου ο ένας χρησιμοποιείται αποκλειστικά για τους αναλογικούς αισθητήρες)
- 1 ρελέ για λειτουργία απομακρυσμένου διακόπτη
- 12-bit ADC μετατροπέας για τα γυροσκόπια, επταχυνσιόμετρα ,αισθητήρα ταχύτητας αέρα
- Μνήμη 16MB για καταγραφή δεδομένων (μαύρο κουτί)..

- FTDI chip, για σύνδεση μικροελεκτή με USB.
- θύρα OSD.
- θύρα I2C για σύνδεση πολλαπλών αισθητηρίων
- 2 προγραματιζόμενοι διακόπτες τύπου button
- 10-Bit ADC μετατροπέας για επέκταση σύνδεσης αναλογικών αισθητηρίων.
- Reset button.
- ενδεικτικά LEDs κατάστασης.
- Γυροσκόπιο (τριών αξόνων) με αντκραδασμικό σύστημα.
- Αναλογικά επιταχυνσιόμετρα ADX330 (τριών αξόνων).
- Θύρα για σύνδεση αισθητήρα ταχύτητας αέρα.
- Αισθητήριο απόλυτης ατμοσφαιρικής πίεσης και θερμοκρασίας για μέτρηση υψομέτρου.

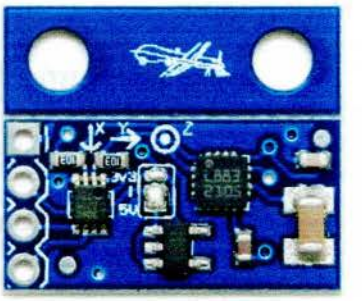
(3.6.2) Δέκτης GPS GS407 U-Blox5 GPS 4Hz

Τεχνικά χαρακτηριστικά:

- Τύπος Chip : uBlox 5
- Κεραία: Sarantel SL1206 ,Συχνοτητας: 1575.42Mhz Ενίσχυση: +24 dBic πλάτος ακτίνας: > 120 Degrees Noise Figure: 1.2 dB
- Τύπος δέκτη :50 καναλιών συχνότητας GPS L1, ανοιχτή λειτουργία συχνότητας C/A Code GALILEO L1
- Χρόνος για καθορισμό θέσης: Κρύο ξεκίνημα < 29s , Ζεστό ξεκίνημα <1s
- Sensitivity: Tracking & Navigation Reacquisition Cold Start (Autonomous)
- <-160dBm <-160dBm <-144dBm
- Ακρίβεια οριζόντιας θέσης:<2.5m
- Ακρίβεια χρονικού παλμού: RMS 99% 30ns <60ns
- 8. Μέγιστος ρυθμός ανανέωσης θέσης : 4Hz
- 9. Ακρίβεια ταχύτητας : 0.1m/s
- 10. Ακρίβεια πορείας : 0.5 degress
- 11. Βάρος: <15g
- 12. Τροφοδοσία : 3.3V +- 5% DC input , < 50mVpp
- 13.Σύνδεση: Panasonic AXK6F10547YG 9600 TTL
- 14. Θερμοκρασία λειτουργίας: -10~50 ° C

- 15. Μέγεθος Flash :4Mbit
- AGPS :Υποστήριξη uBlox AssistNow

(3.6.3) Αισθητήριο ψηφιακής πυξίδας τριών διαστάσεων HMC5883L



Τεχνικά χαρακτηριστικά:

- Διασύνδεση I2C Digital Interface με μετατροπέα σήματος για συμβατότητα 2.5 V έως 3.3V
- Τάση τροφοδοσίας 4V εως 5.5V
- Ακρίβεια μέτρησης 1° με 2° μοίρες
- Μέγιστος ρυθμός εξόδου 160 Hz
- Κατανάλωση ισχύος 100 μA
- Μετατροπέας ADC ανάλυσης 12-bit

(3.6.4) Ασύρματη επικοινωνία XBee-PRO® 900 RF

Τεχνικά χαρακτηριστικά:

Performance:

- Ισχύς εκπομπής: 50 mW (+17 dBm)
- Εμβέλεια σε εσωτερικούς χώρους: 140 m
- Εμβέλεια σε εξωτερικούς χώρους: 3 km, και με κεραίες υψηλής ενίσχυσης: 10 km
- Ρυθμός ασύρματης μετάδοσης δεδομένων: 156 Kbps
- Ρυθμός ενσύρματης μετάδοσης δεδομένων: μέχρι 230 Kbps
- Συχνότητα λειτουργίας εκπομπής: 900 MHz
- Ευαισθησία δέκτη: -100 dBm

Networking:

- Τύπος διάδοσης ραδιοφάσματος: FHSS (Διάδοση ραδιοφάσματος με Αναπήδηση Συχνότητας)
- Τοπολογία δικτύωσης: Peer-to-Peer Mesh, point-to-point & point-to-multipoint
- Διαχείριση αποσφαλμάτωσης: επαναεκπομπή & και επιβεβαίωση
- Επιλογές φιλτραρίσματος: PAN ID, καναλιού, και 64-bit διευθύνσεις
- Χωρητικότητα καναλιού: 8 μεταπήδημενες συχνότητες σε 12 κανάλια

Τροφοδοσία:

- Supply voltage: XBee-PRO: 3.0 - 3.6 VDC
- Κατανάλωση ισχύος μετάδοσης: 210 mA στα 3.3 V

- Κατανάλωση ισχύος λήψης: 80 mA στα 3.3V
- Κατανάλωση ισχύος αναμονής: 60 uA στα 3.3V

Γενικά:

- Μπάντα συχνοτήτων: 902 - 928 MHz
- Διασύνδεση: 3V CMOS UART

Φυσικές Ιδιότητες

- Μέγεθος:2.438 cm x 3.294 cm
- Βάρος:3g
- Ιδιότητες κεραίας: U.FL, Reverse Polarity SMA (RPSMA), or wired whip antenna
- Θερμοκρασία λειτουργίας: -40° C to +85° C (industrial)

Πιστοποιήσεις:

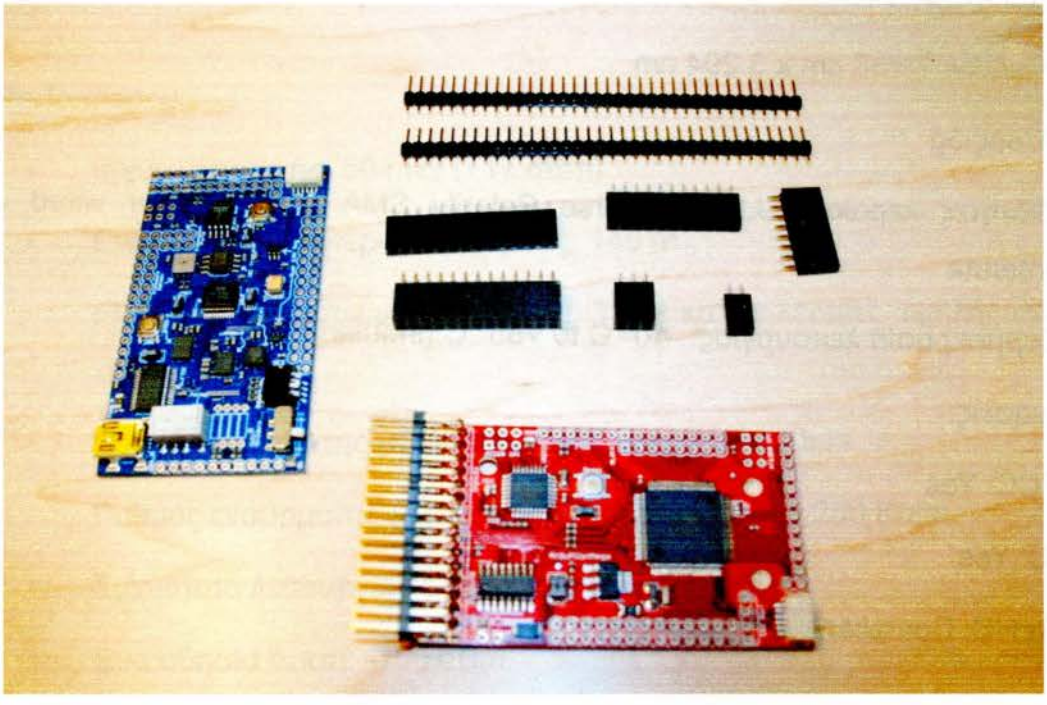
- FCC: Yes
- IC: Yes
- RoHS: Compliant

Κεφάλαιο 4

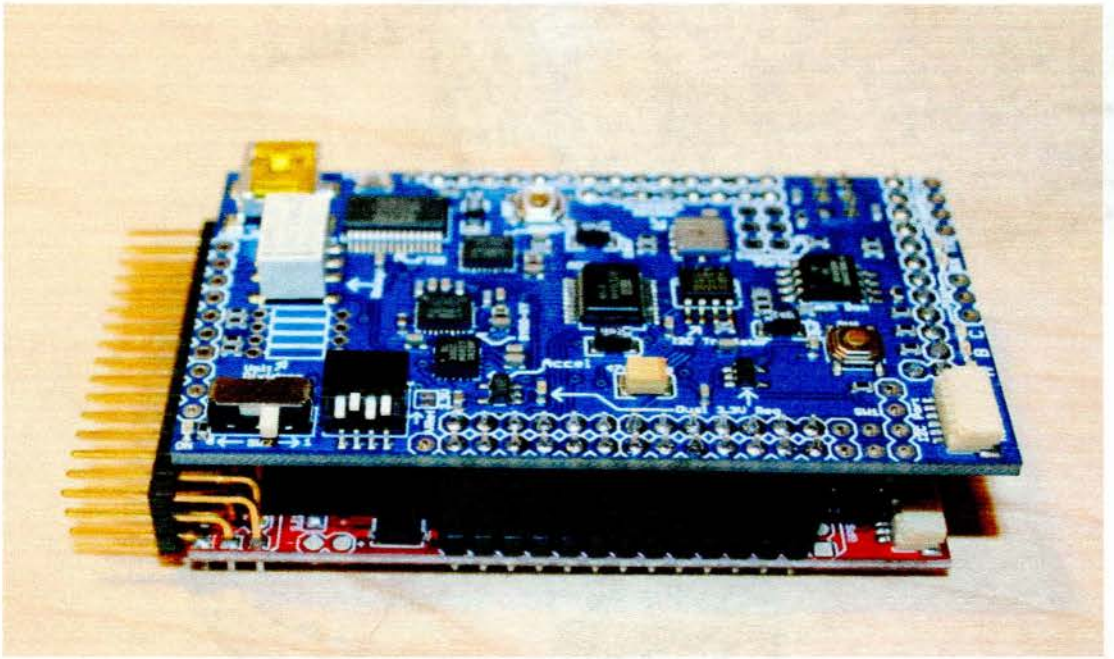
Κατασκευή και συνδεσμολογία υλικών

Σύνδεση Ardupilot με μονάδα IMU (4.1)

Η σύνδεση μεταξύ της μονάδας μικρο ελεγκτή και μονάδας IMU ,επιτυγχάνετε με τη χρήση ειδικών συνδέσμων που θα κολληθούν πάνω στις πλακέτες του μικρο ελεγκτή και μονάδος IMU

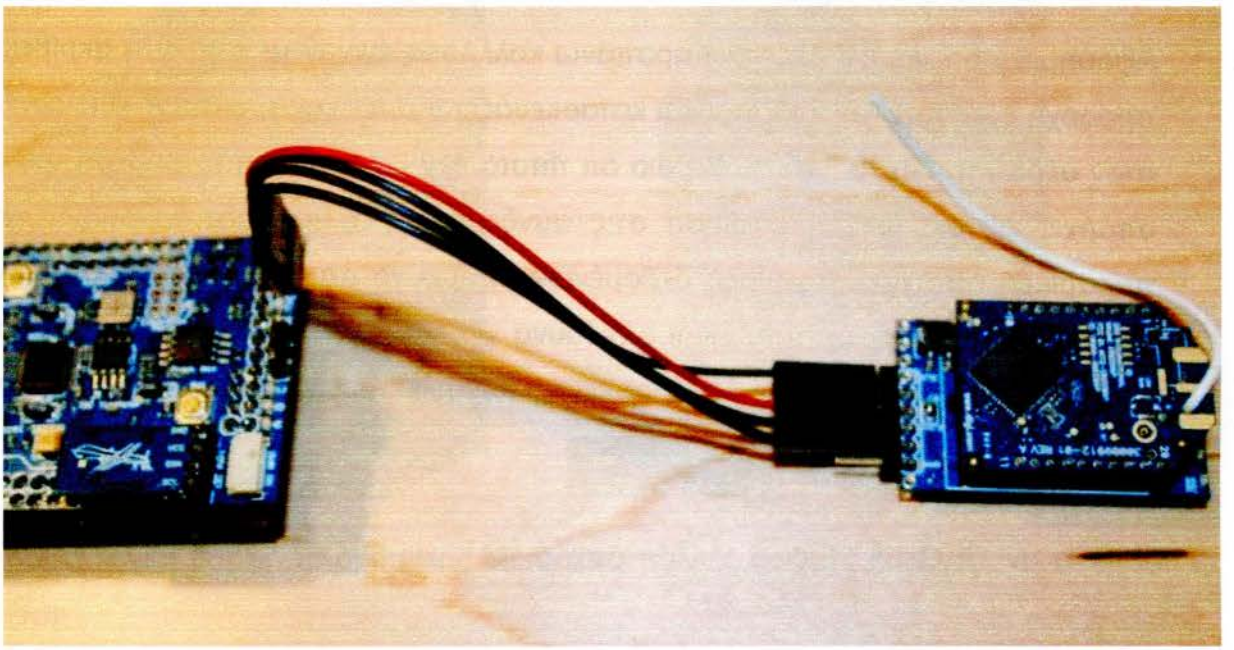


Με αποτέλεσμα το τελικό κύκλωμα να είναι έτσι:



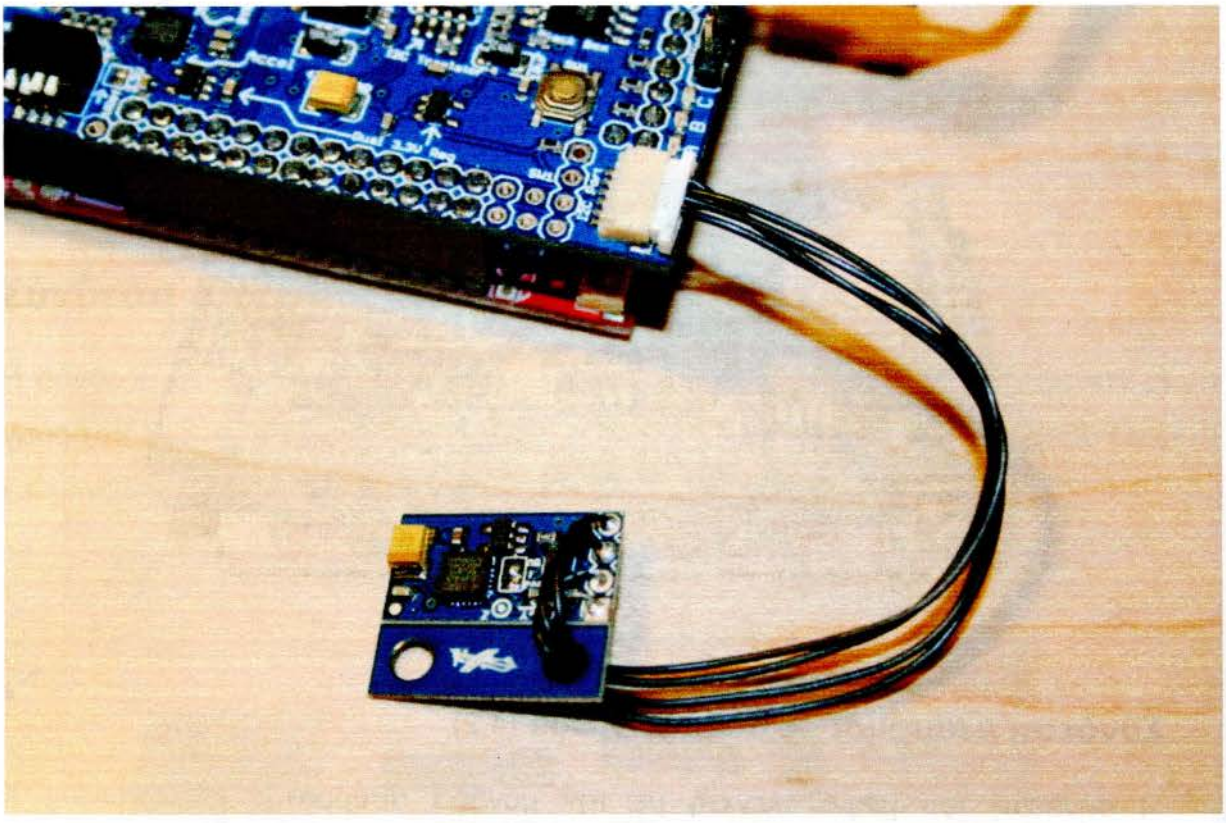
Σύνδεση Ardupilot με μονάδα XBee (4.2)

Η σύνδεση του μικρο ελεγκτή με την μονάδα ασύρματης επικοινωνίας XBee επιτυγχάνεται με χρήση καλωδίου , και πάνω στις πλακέτες χρήση ακιδοσειρών.



Σύνδεση Ardupilot με το Αισθητήριο πυξίδας (4.3)

Η σύνδεση του αισθητήριου της πυξίδας με το μικρο ελεγκτή επιτυγχάνεται με χρήση τετραπλού καλωδίου κολλημένο πάνω στις πλακέτες

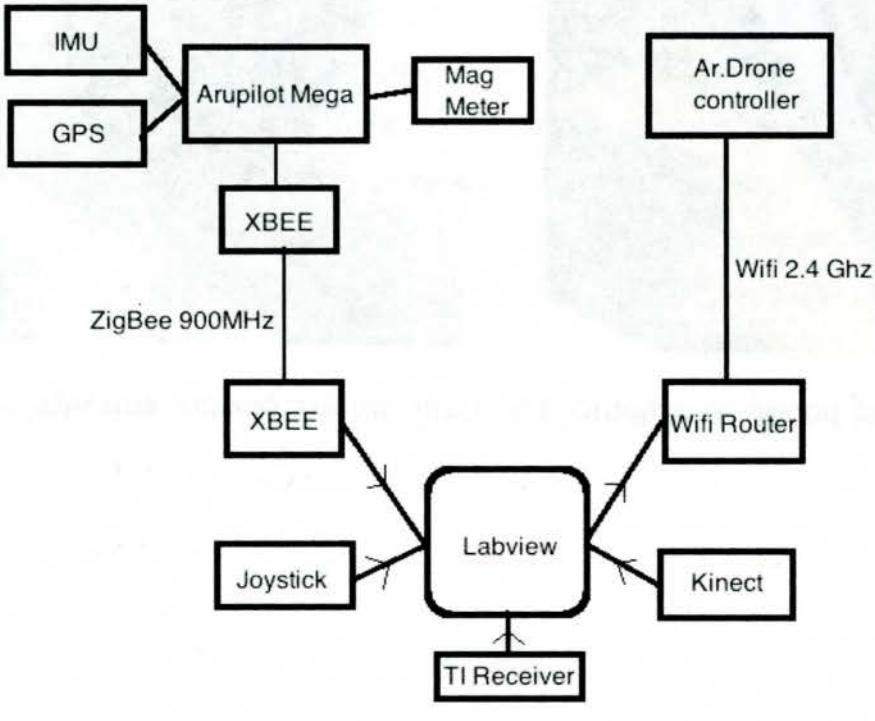


Τελική μορφή οχήματος (4.4)

Αξιοσημείωτο είναι ότι όλες οι παραπάνω κολλήσεις έγιναν με απόλυτη ακρίβεια και προσοχή διότι όταν σχεδιάζεται και κατασκευάζεται ένα όχημα το οποίο θα λειτουργεί στον αέρα , πρέπει να 100% βέβαιο ότι τίποτα δεν θα φύγει από την θέση του ή ότι υπάρχει κάποια ψυχρή κόλληση στις συνδεσμολογίες των εξαρτημάτων , για η παραμικρή αστοχία μεταφοράς δεδομένων μπορεί να αποβεί μοιραία για το όχημά εφόσον τα εξαρτήματα αυτά είναι υπεύθυνα για τη διατήρηση σωστής λειτουργίας του οχήματος. Η τελική μορφή κατασκευής των ηλεκτρονικών εξαρτημάτων φαίνεται στις εικόνες που ακολουθούν:

Και η τελική μορφή το οχήματος Μέτα από πολλές δοκιμές κατέληξε να είναι κάπως έτσι:

και στην επόμενή εικόνα φαίνεται το σχεδιάγραμμα συνδεσμολογίας όλων των παραπάνω υλικών :



Επίγειος σταθμός ελέγχου (Ground Control Station (GCS))

Αφού κατά πρώτη προσέγγιση καταλήξαμε στον εξοπλισμό , ήρθε η στιγμή να σχεδιαστεί μια εφαρμογή η οποία επεξεργάζεται όλα αυτά τα δεδομένα που συλλέγονται απο τα αισθητήρια και να κάνει τους κατάλληλους υπολογισμούς και να προβεί στις απαραίτητες ενέργειες για την ολοκλήρωση του στόχου που είναι ο υπολογισμός και αυτόματη πλοήγηση του οχήματος από το ένα σημείο στο άλλο με την ελάχιστη δυνατή διαδρομή .

(5.1) **Προδιαγραφές για τον σχεδιασμό λογισμικού Επίγειου Σταθμού Εδάφους**

Οι **Προδιαγραφές** που πρέπει να τηρεί το λογισμικό που θα σχεδιαστεί είναι οι εξής:

1. Να κάνει τη συλλογή δεδομένων από το όχημα και να τα απεικονίζει με το κατάλληλο τρόπο ώστε να είναι εύκολα αντιληπτές από τον χρήστη .
2. Να περιέχει ένα διαδραστικό και παράλληλα μεγάλης ακρίβειας ψηφιακό χάρτη όπου να μπορούν να είναι γνωστές οι συντεταγμένες σε ένα σημείο που επέλεξε ο χρήστης , όπως επίσης και το σημείο που βρίσκεται το όχημα σε πραγματικό χρόνο.
3. Να υπολογίζει τη διαδρομή την οποία πρέπει να ακολουθήσει το όχημα ώστε να μεταφερθεί από το σημείο που βρίσκεται σε ένα επιθυμητό σημείο.
4. Να εκτελεί τη πλοήγηση του οχήματος , και να διορθώνει τυχόν σφάλματα που μπορούν να παρουσιαστούν κατά τη διάρκεια της πλοήγησης.
5. Να προειδοποιεί τον χρήστη σε περίπτωση που πρέπει να παρέμβει ώστε να αποφευχθεί η αποτυχία εκπλήρωσης της αποστολής λόγω απροσπέλαστου σφάλματος
6. Να έχει την δυνατότητα να εκτελεστεί σε οποιοδήποτε ηλεκτρονικό υπολογιστή και λειτουργικό σύστημα.

7. Να υποστηρίζει συνεργασία με άλλες εφαρμογές και ανταλλαγή δεδομένων είτε αυτή είναι τοπικά ή απομακρυσμένα .
8. Να έχει εύκολο χειρισμό και να είναι ευκατανόητο από ένα μη εξειδικευμένο χρήστη.
9. Να είναι αξιόπιστο και σταθερό στη λειτουργία του.
10. Να έχει δυνατότητα αναβάθμισης.
11. Να είναι παραμετροποιήσιμο ώστε να μπορεί να δουλέψει σε διαφορετικούς τύπους οχημάτων , ασύρματης η ενσύρματης επικοινωνίας και υλικού (Hardware).

(5.2) **Υλοποίηση εφαρμογής επίγειου σταθμόυ ελέγχου** (Ground Control Station (GCS)) **σε προγραμματιστικό περιβάλλον Labview**

Η εφαρμογή επίγειου σταθμού ελέγχου σχεδιάστηκε σε περιβάλλον προγραμματισμού Labview και χωρίζετε σε δύο υπομέρη τμήματα που δουλεύουν ανεξάρτητα μεταξύ τους αλλά μοιράζονται τις πληροφορίες οι οποίες είναι απαραίτητες για την επίτευξη του στόχου. Τα δύο αυτά τμήματα είναι :

1. **Εφαρμογή ελέγχου του οχήματος.**

2. **Εφαρμογή συλλογής δεδομένων , απεικόνισης, υπολογισμού διαδρομής, και πλοήγησης του οχήματος.**

(5.4) **Περιβάλλον προγραμματισμού Labview**

Το Labview είναι μια γλώσσα προγραμματισμού τέταρτης γενιάς εξειδικευμένη για μηχανικούς με έτοιμα εργαλεία (βιβλιοθήκες) που βοηθούν στις εξής λειτουργίες : συλλογή δεδομένων, την ανάλυση δεδομένων, την προσομοίωση και τον έλεγχο οργάνων και μετρήσεων μέσω υπολογιστή.

Στηρίζεται στον γραφικό προγραμματισμό μέσω αντικειμένων και αποτελεί ένα καλό παράδειγμα του «αντικειμενοστραφή προγραμματισμού» (object oriented programming) όπου σε αντίθεση με τον λεγόμενο «προγραμματισμό διαδικασιών», όπου ο προγραμματιστής γράφει κώδικα εντολών που εκτελούνται με γραμμική διαδοχή ,στο γραφικό περιβάλλον LabView ο προγραμματιστής δεν γράφει κώδικα, αλλά χρησιμοποιεί γραφικά αντικείμενα, όπως κουμπιά, δείκτες, οθόνες ή τετραγωνίδια που παριστάνουν συναρτήσεις ή εκτελούν συγκεκριμένες λειτουργίες με τη μορφή υπορουτινών. Αυτά τα εικονίδια έχουν εισόδους και εξόδους και επιδέχονται προγραμματισμό των ιδιοτήτων τους.

Το όνομα LabView είναι το ακρωνύμιο των λέξεων «Laboratory Virtual Instrument Engineering Workbench» (Σχεδιαστήριο για την Κατασκευή Εργαστηριακών Εικονικών Οργάνων) και αναπτύχθηκε κατά το τέλος της δεκαετίας του 80 από την εταιρία National Instruments για το λειτουργικό Mac και στη συνέχεια εξελίχθηκε και μετατράπηκε έτσι ώστε να λειτουργεί με τα περισσότερο γνωστά λειτουργικά συστήματα Mac ,Windows και Linux . Η σχεδιάστρια εταιρία National Instruments ειδικεύεται σε συστήματα συλλογής δεδομένων, σε αισθητήρες, αυτοματισμούς ,σε λογισμικό μετρήσεων και ελέγχου. Η γραφική γλώσσα που χρησιμοποιεί το LabView για τον προγραμματισμό και τη δημιουργία εικονικών οργάνων ονομάζεται γλώσσα G.

(5.6) **Περιγραφή εφαρμογής ελέγχου του οχήματος (Ar.Drone Control Center)**

Η εφαρμογή ελέγχου οχήματος (Ar.Drone control) βασίζεται πάνω στο SDK(Software Development Kit) που μας παρέχει ο κατασκευαστής του Ar.Drone και συμμορφώνεται σύμφωνα με τα πρωτόκολλα επικοινωνίας που έχει θέσει ο κατασκευαστής Parrot .

Όπως έχει αναφερθεί σε προηγούμενες παραγράφους το όχημα Ar.Drone επικοινωνεί με τον υπολογιστή διαμέσου του ασύρματου πρωτοκόλλου WiFi IEEE802.11 και ανταλλάσσει πακέτα πρωτοκόλλου UTP. Οπότε έπρεπε να επιλέξουμε μία γλώσσα προγραμματισμού η οποία μας δίνει τη δυνατότητα να χρησιμοποιήσουμε την τεχνολογία Internet Socket ή Network Socket .Σύμφωνα και

με αυτό το παράγοντα κρίθηκε ότι το Labview είναι άκρως κατάλληλο για το σχεδιασμό της εφαρμογής αυτής .

Οι διαδικασίες που πρέπει να εκτελεί η **εφαρμογή ελέγχου** του οχήματος είναι οι εξής :

1. Ο πλήρης έλεγχος της συμπεριφοράς του οχήματος .

2. Να έχει ανάδραση από το όχημα στο αν έλαβε και εκτέλεσε τις εντολές που στάλθηκαν.
3. Να παίρνει πληροφορίες για τη πραγματική πλοήγησή του και την κατάστασή του σε πραγματικό χρόνο.

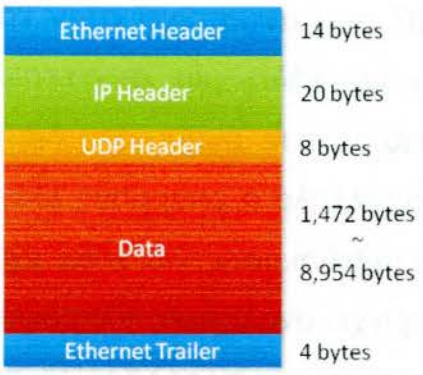
3. Να έχει σαν είσοδο την επιθυμητή κατάσταση του οχήματος από διαφορετικού τύπου χειριστήρια και συσκευές ή κάποιο άλλο λογισμικό χωρίς να περιορίζεται σε κάποιο συγκεκριμένο προϊόν. Δηλαδή να είναι ανοιχτό και παραμετροποιήσιμο σε οποιαδήποτε αλλαγή είναι επιθυμητή να γίνει ώστε να του δοθεί σαν είσοδος η επιθυμητή κατάσταση του οχήματος.

(5.7) **Πρωτόκολλο UDP**

Πριν αναφερθεί περιληπτικά η διαδικασία επικοινωνίας με το όχημα Ar.Drone πρέπει να περιγραφεί κάποιες βασικές πληροφορίες για το πρωτόκολο UDP και τη χρήση του σήμερα .

Το πρωτόκολλο User Datagram Protocol (UDP) είναι ένα από τα βασικά πρωτόκολλα που χρησιμοποιούνται στο Διαδίκτυο. Μία εναλλακτική ονομασία του πρωτοκόλλου είναι Universal Datagram Protocol. Διάφορα προγράμματα χρησιμοποιούν το πρωτόκολλο UDP για την αποστολή σύντομων μηνυμάτων (γνωστών και ως datagrams) από τον έναν υπολογιστή στον άλλον μέσα σε ένα δίκτυο υπολογιστών. Ένα από τα κύρια χαρακτηριστικά του UDP είναι ότι δεν εγγυάται αξιόπιστη επικοινωνία αλλά γρήγορη επικοινωνία. Τα πακέτα UDP που αποστέλλονται από έναν υπολογιστή μπορεί να φτάσουν στον παραλήπτη με λάθος

σειρά, διπλά, ή να μην φτάσουν καθόλου εάν το δίκτυο ειναι φορτωμένο. Αντιθέτως, το πρωτόκολλο TCP διαθέτει όλους τους απαραίτητους μηχανισμούς ελέγχου και επιβολής της αξιοπιστίας και συνεπώς μπορεί να εγγυηθεί την αξιόπιστη επικοινωνία μεταξύ των υπολογιστών. Η έλλειψη των μηχανισμών αυτών από το πρωτόκολλο UDP το καθιστά αρκετά πιο γρήγορο και αποτελεσματικό, τουλάχιστον για τις εφαρμογές εκείνες που δεν απαιτούν αξιόπιστη επικοινωνία αλλά μεγάλη ταχύτητα.



Οι εφαρμογές audio και video streaming χρησιμοποιούν κατά κόρον τα πακέτα UDP. Για τις εφαρμογές αυτές είναι πολύ σημαντικό τα πακέτα να παραδοθούν στον παραλήπτη σε σύντομο χρονικό διάστημα ώστε να μην υπάρχει διακοπή στην ροή του ήχου ή της εικόνας. Κατά συνέπεια προτιμάται το πρωτόκολλο UDP διότι είναι αρκετά γρήγορο, παρόλο που υπάρχει η πιθανότητα μερικά πακέτα UDP να χαθούν. Στην περίπτωση που χαθεί κάποιο πακέτο, οι εφαρμογές αυτές διαθέτουν ειδικούς μηχανισμούς διόρθωσης και παρεμβολής ώστε ο τελικός χρήστης να μην παρατηρεί καμία αλλοίωση ή διακοπή στην ροή του ήχου και της εικόνας λόγω του χαμένου πακέτου. Σε αντίθεση με το πρωτόκολλο TCP, το UDP υποστηρίζει broadcasting, δηλαδή την αποστολή ενός πακέτου σε όλους τους υπολογιστές ενός δικτύου, και multicasting, δηλαδή την αποστολή ενός πακέτου σε κάποιους συγκεκριμένους υπολογιστές ενός δικτύου. Η τελευταία δυνατότητα χρησιμοποιείται πολύ συχνά στις εφαρμογές audio και video streaming ούτως ώστε μία ροή ήχου ή εικόνας να μεταδίδεται ταυτόχρονα σε πολλούς συνδρομητές.

(5.7) **Εφαρμογές του πρωτοκόλου UDP**

Μερικές σημαντικές εφαρμογές που χρησιμοποιούν πακέτα UDP είναι οι εξής: Domain Name System (DNS), IPTV, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) και τα παιχνίδια που παίζονται ζωντανά μέσω του Διαδικτύου.

(5.8) Αναλυτική περιγραφή επικοινωνίας και έλεγχος του οχήματος

Η εφαρμογή ελέγχου οχήματος όπως εχει αναφερθεί βασίστηκε στο SDK(Software Development Kit) και η κύρια διεργασία της είναι να συνθέσει ένα πακέτο με δεδομένα που αντιπροσωπεύουν μια συγκεκριμένη συμπεριφορά του οχήματος σε μια συγκεκριμένη χρονική στιγμή έπειτα να ανοίξει μία πόρτα (Socket) και από αυτή να στέλνει το πακέτο που δημιούργησε σε μια συγκεκριμένη διεύθυνση IP που αντιστοιχεί στην κάρτα δικτύου του οχήματος .Ταυτόχρονα φροντίζει να βάλει μία συγκεκριμένη υπογραφή στο πακέτο που στέλνει ώστε να γνωρίζει το όχημα την διαδοχή των πακέτων που στέλνει η εφαρμογή αυτή. Παράλληλα η εφαρμογή ελέγχου του οχήματος Ar.Drone περιμένει ένα πακέτο επιβεβαίωσης για κάθε εντολή που έστειλε καθώς και ένα πακέτο στο οποίο περιέχει αναλυτικές πληροφορίες για την κατάσταση του οχήματος .Για να πάρουμε τις πληροφορίες αυτές ανοίγει μια διαφορετική πόρτα (socket) από αυτή που είχε ανοίξει για να στέλει τα δεδομένα στο όχημα, ώστε να δεχθεί τα πακέτα που στέλνονται από το Ar.Drone ,έπειτα τα αναλύει και επεξεργάζεται τη χρήσιμη πληροφορία.

(5.9) **Σύνθεση πακέτων εντολών και αποστολή στο Ar.Drone**

Οι εντολές που δέχεται το όχημα Ar.Drone είναι τύπου AT ,έχουν συγκεκριμένη μορφή και σύνταξη η οποία περιγράφεται στο (SDK) του Ar.Drone και θα δωθεί η περιγραφή τους στις παρακάτω παραγράφους.

Το πακέτο που στέλνεται περιέχει συμβολοσειρές (String) που κωδικοποιούνται ως 8-bit ASCII χαρακτήρες, με τον χαρακτήρα line feed <LF> στο τέλος τους, ως οριοθέτηση και αλλαγή γραμμής.

Μια εντολή αποτελείται από τρεις χαρακτήρες "AT*" (δηλαδή τρεις λέξεις των 8-bit με τιμες $41_{(16)}$,$54_{(16)}$,$2a_{(16)}$) ακολουθείται από το όνομα της εντολής, το σύμβολο της ισότητας, και τον αύξων αριθμό, προαιρετικά ακολουθεί, μια λίστα διαχωρισμένη με κόμματα περιέχοντας τις παραμέτρους της εντολής.

Ένα ενιαίο πακέτο UDP μπορεί να περιέχει μία ή περισσότερες εντολές, που χωρίζονται από νέα γραμμή με τον χαρακτήρα (byte $0A_{(16)}$). Μια εντολή AT πρέπει να παραμένει σε ένα ενιαίο πακέτο UDP. Η Διάσπαση της AT εντολή σε δύο ή περισσότερα πακέτα UDP δεν είναι αναγνωρίσιμη.

Παράδειγμα :

AT*PCMD=21625,1,0,0,0,0<LF>AT*REF=21626,290717696<LF>

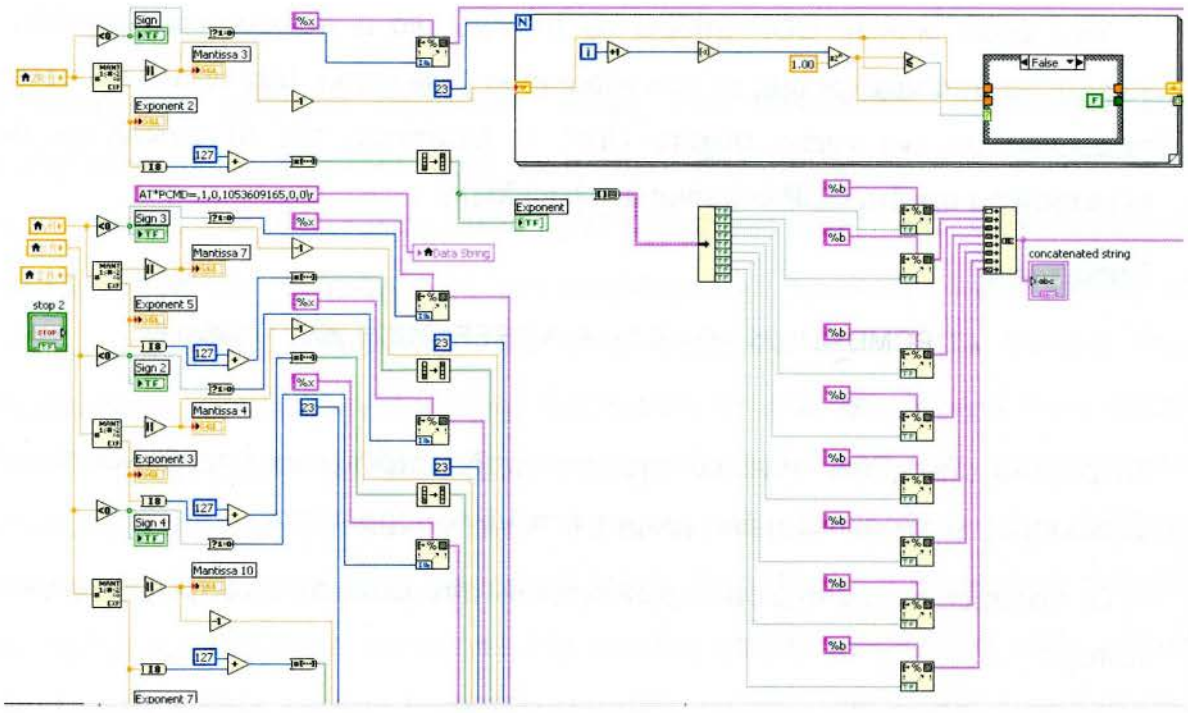
Το μέγιστο μήκος του συνόλου της εντολής δεν πρέπει να υπερβαίνει τους 1024 χαρακτήρες ,διαφορετικά η όλη γραμμή εντολών απορρίπτεται.

Οι παράμετροι των περισσότερων εντολών αποτελούνται από τρεις διαφορετικούς τύπους

- Ένας προσυμασμένος ακέραιος, τοποθετημένος στη συμβολοσειρά (String) εντολών με δεκαδική αναπαράσταση (παράδειγμα: αύξων αριθμός)
- Η τιμή μιας συμβολοσειράς τοποθετείτε μεταξύ διπλών εισαγωγικών (π.χ.: οι παράμετροι της εντολής AT * CONFIG)
- Μιας ενιαίας ακρίβειας IEEE-754 floating-point τιμή. Όσες δε τιμές δεν περιέχονται μέσα στην εντολή θα πρέπει να δημιουργηθούν και να τοποθετηθούν μέσα στη συμβολοσειρά (String) σαν 32 bit λέξη.

Παράδειγμα: Ο αριθμός "−0.8" μετατρέπεται σε λέξη των 32 bit η οποία είναι "$BF4CCCCD_{(16)}$", σύμφωνα μορφή IEEE-754.

(5.10) Αλγόριθμος μετατροπής ακέραιων σε IEEE-754 floating-point



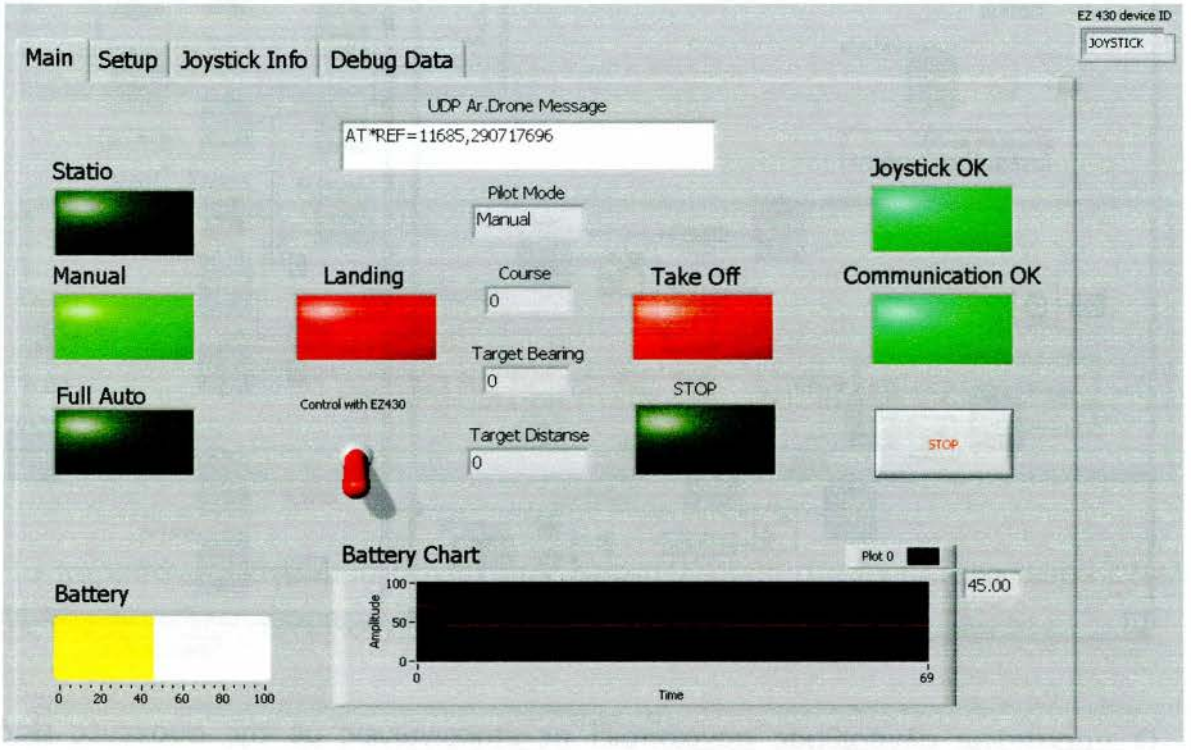
Στον παραπάνω αλγόριθμο υπολογίζει την μετατροπή ακεραίων σε IEEE-754 floating-point.

Παρακάτω δίνεται ένας πίνακας με το σύνολο των εντολών που αναγνωρίζονται απο το όχημα Ar.Drone.

AT command	Callback[1]	Arguments[2]	Description
AT*REF	***at_rc_ref_exe***	input	Takeoff/Landing/Emergency stop command
AT*PCMD	***at_pcmd_exe***	roll,pitch,gaz,yaw	Move the drone
AT*FTRIM		-	Sets the reference for the horizontal plane
AT*MTRIM		-	Manually set an offset in the commands
AT*MISC	***at_misc_exe***	m1,m2,m3,m4	Send Misc data (i.e. undocumented drone parameters for internal usage)

Περισσότερες και πιο λεπτομερής περιγραφές για τις παραπάνω εντολές μπορούν βρεθούν στο παράρτημα Ar.Drone SDK.

(5.11) Αλγόριθμος σύνθεσης πακέτου εντολών και αποστολή σε πακέτα UDP



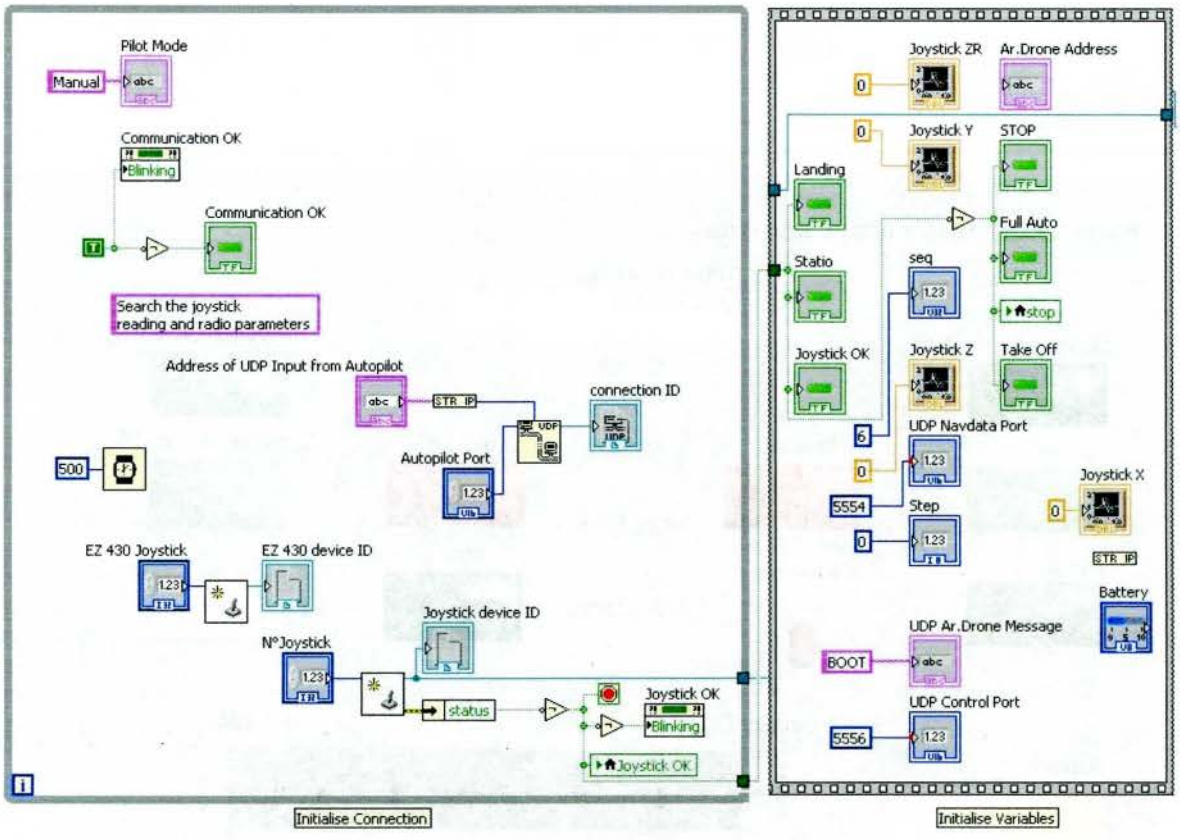
Η παραπάνω εικόνα απεικονίζει την κεντρική οθόνη της εφαρμογής ελέγχου οπού επιδεικνύει τις βασικές καταστάσεις του οχήματος

δηλαδή:

1. Hover (αιώρηση)
2. Χειροκίνητος χειρισμός
3. Αυτόματη πλοήγηση

Επίσης πληροφορεί τον χρήστη για τη κατάσταση της μπαταρίας , και την κατάσταση επικοινωνίας.

(5.12) **Block διάγραμμα του αλγορίθμου έλεγχου οχήματος**

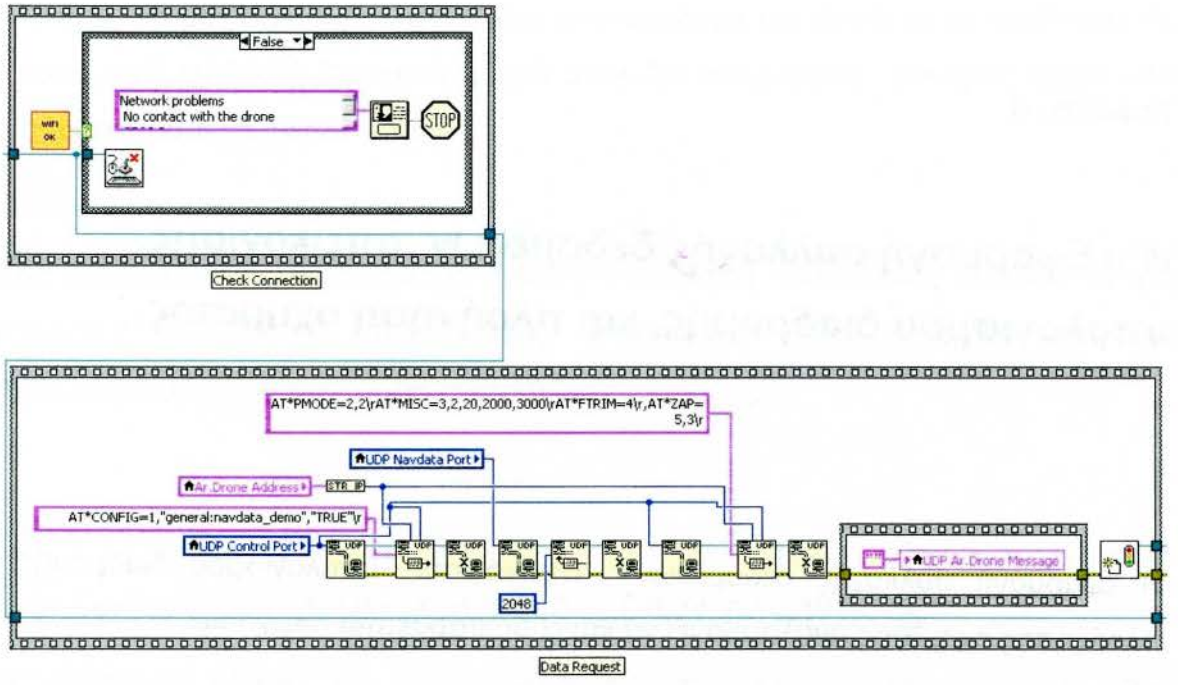


Ο παραπάνω αλγόριθμος αρχικοποιεί τις επικοινωνείς με της συσκευές εισόδου (Joystick , Chronos ez430, Kinect Port) , όπως και επίσης την επικοινωνία με το όχημα.

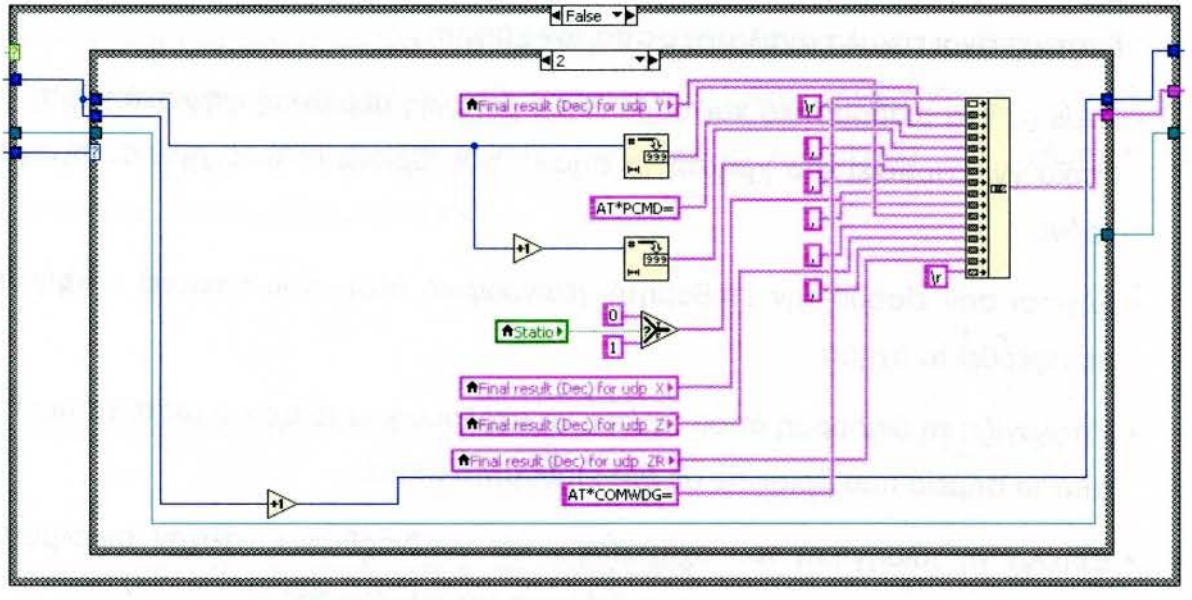
Συσκευές εισόδου χειροκίνητου χειρισμού

Chronos EZ430

Cyborg X Joystick



Το παραπάνω διάγραμμα δείχνει πως αποστέλλονται στο όχημα μια σειρά εντολών με την κατάλληλη αλληλουχία ώστε να εκκινηθεί ο χειρισμός του οχήματος.



Το παραπάνω διάγραμμα δείχνει πως συνθέτεται το πακέτο με τις πληροφορίες χειρισμού του οχήματος το οποίο προωθείται σε κάποιο άλλο σημείο του κώδικα και συγχωνεύεται σε ένα μεγαλύτερο πακέτο πληροφοριών που περιέχει έναν αύξων αριθμό που επισφραγίζει την αλληλουχία του κάθε πακέτου

Κεφάλαιο 6

(6.1) Εφαρμογή συλλογής δεδομένων ,απεικόνισης, υπολογισμού διαδρομής, και πλοήγηση οχήματος.

Η εφαρμογή συλλογής δεδομένων , απεικόνισης, υπολογισμού διαδρομής, και πλοήγησης οχήματος αναλαμβάνει να κάνει τις παρακάτω διεργασίες :

- Εκτελεί τη συλλογή δεδομένων από το όχημα και τα απεικονίζει με τέτοιο τρόπο ώστε να είναι εύκολα αντιληπτές από τον χρήστη .
- Περιέχει ένα διαδραστικό και παράλληλα μεγάλης ακριβείας ψηφιακό χάρτη με τον οποίο γνωστοποιεί στο χρήστη το σημείο που βρίσκεται το όχημα σε πραγματικό χρόνο.
- Δέχεται σαν είσοδο την επιθυμητή γεωγραφική θέση που επιλέγει ο χρήστης να μεταφερθεί το όχημα.
- Υπολογίζει τη διαδρομή όπου πρέπει να ακολουθήσει το όχημα ώστε να μεταφερθεί από το σημείο που βρίσκετε σε ένα επιθυμητό σημείο.
- Εκτελεί τη πλοήγηση του οχήματος , και να διορθώνει τυχόν σφάλματα που μπορούν να προκύπτουν κατά τη διάρκεια της πλοήγησης.
- Προειδοποιεί τον χρήστη σε περίπτωση που πρέπει να παρέμβει ώστε να αποφευχθεί η αποτυχία εκπλήρωσης της αποστολής λόγω απροσπέλαστου σφάλματος.
- Δίνει στην εφαρμογή ελέγχου του οχήματος οδηγίες για το πως πρέπει να κινηθεί το όχημα ώστε να ολοκληρώσει το στόχο του.

- Μπορεί να μοιράσει τις πληροφορίες που συλλέγει και εξάγει σε οποιαδήποτε άλλη εφαρμογή, είτε αυτή βρίσκεται τοπικά στον ίδιο υπολογιστή , είτε αυτή τρέχει κάπου απομακρυσμένα, μέσω δικτύου.

Στις παρακάτω παραγράφους θα δοθεί η περιγραφή για το πως ακριβώς το τμήμα συλλογής δεδομένων , απεικόνισης, υπολογισμού διαδρομής, και πλοήγησης οχήματος εκτελεί τις παραπάνω διεργασίες.

(6.2) **Συλλογή και απεικόνιση δεδομένων πλοήγησης**

Η εφαρμογή δέχεται με πρωτόκολλο σειριακής επικοινωνίας από τη μονάδα ασύρματης επικοινωνίας XBEE ένα πακέτο με τις πληροφορίες από τα αισθητήρια , που είναι συνδεδεμένα στο μικρο ελεγκτή APM ,όπου εκεί συνθέτονται με ένα συγκεκριμένο πρωτόκολλο επικοινωνίας .Έπειτα η εφαρμογή αναλύει το πακέτο αυτό ώστε να μπορεί να ξεχωρίσει τις διάφορες πληροφορίες, και τα απεικονίζει στο γραφικό περιβάλλον του χρήστη . Οι πληροφορίες όπου απεικονίζονται στην πρώτη καρτέλα του γραφικού περιβάλλοντος που είναι η τηλεμετρία του οχήματος και αποτελείται απο τις εξής μετρήσεις :

1. Υψόμετρο από τον αισθητήρα πίεσης

2. Υψόμετρο από τον υπολογισμό του δέκτη GPS

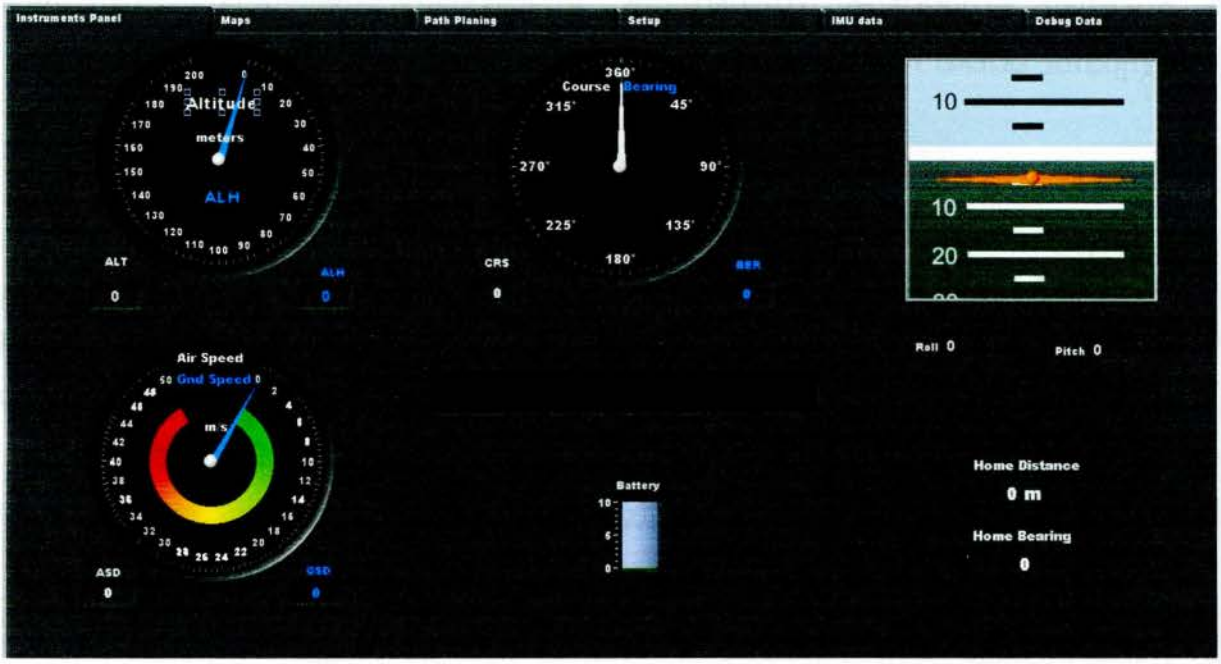
3. Πορεία του οχήματος από το αισθητήριο μαγνητικού βορρά

4. Πορεία του οχήματος από τον υπολογισμό του δέκτη GPS

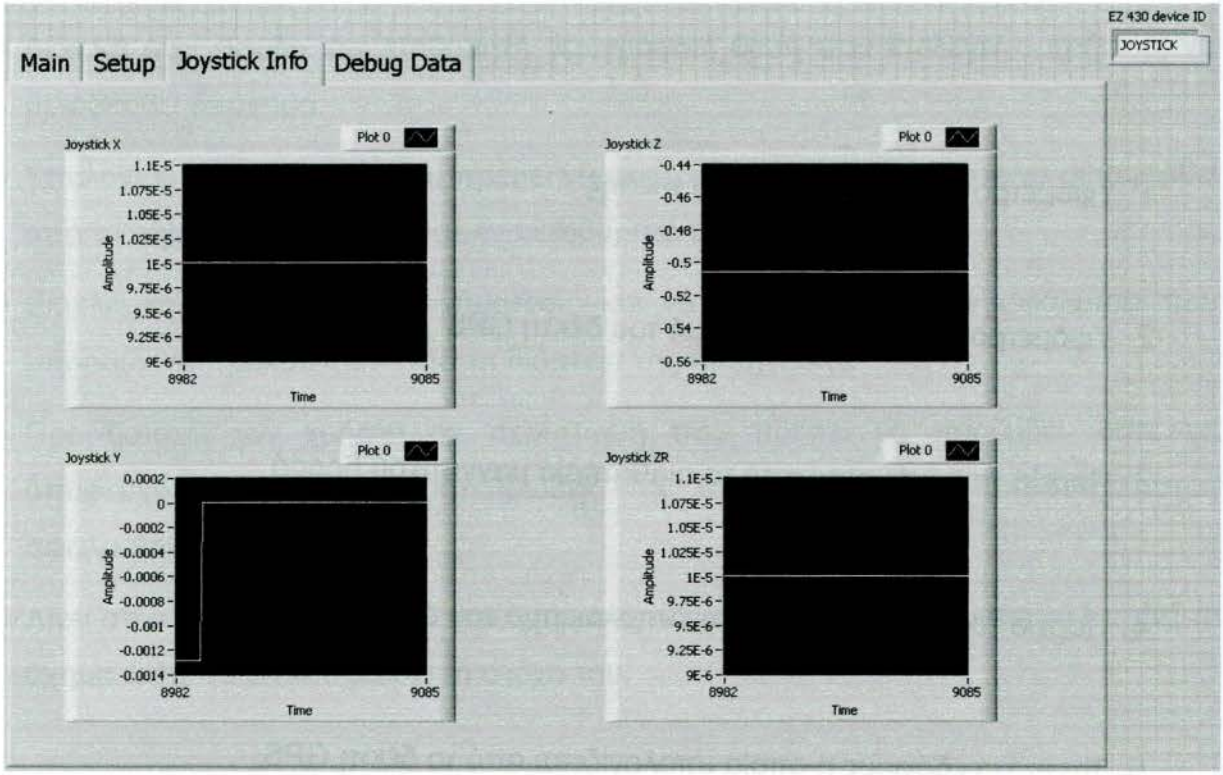
5. Ταχύτητα εδάφους η οποία υπολογίζεται από το δέκτη GPS

6. Η στάση (attitude) του οχήματος δηλαδή οι γωνίες Pitch και Roll.

Όλες οι παραπάνω ενδείξεις απεικονίζονται και γραφικά αλλά και σε ψηφιακές τιμές .Στην παρακάτω εικόνα απεικονίζονται τα ενδεικτικά όργανα τηλεμετρίας από το όχημά



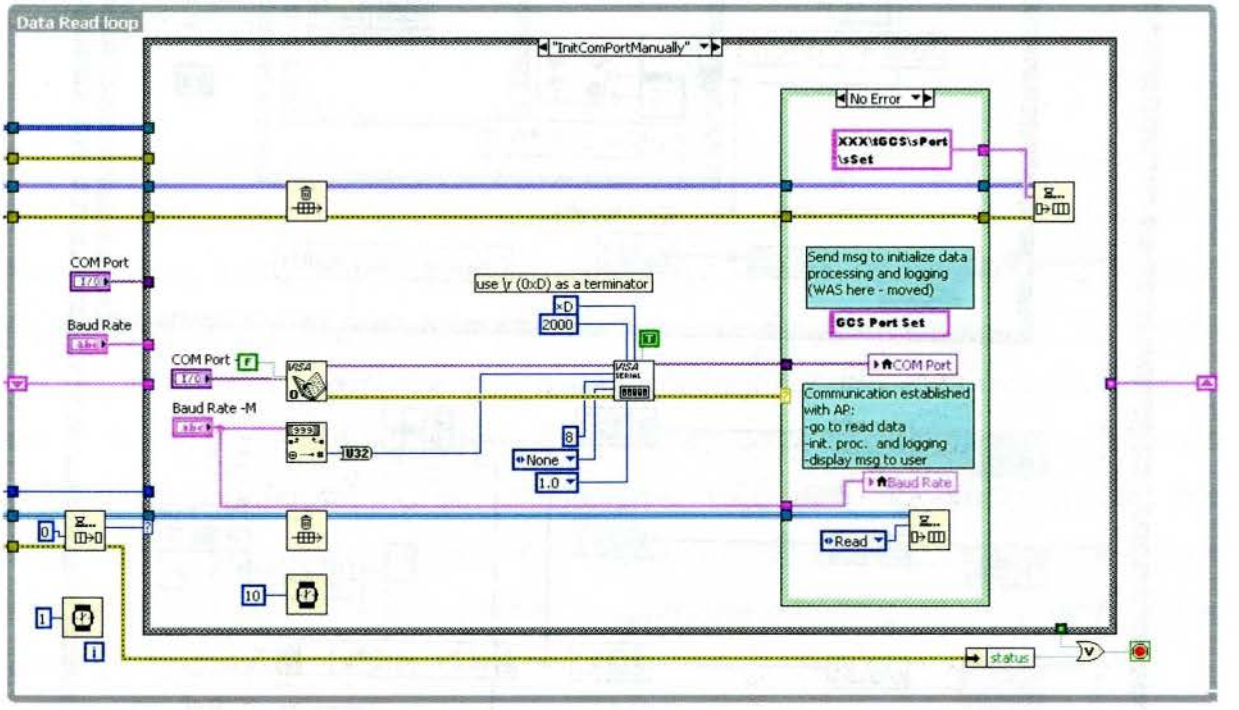
Ενώ στη παρακάτω εικόνα εμφανίζονται πληροφορίες από τις συσκευές εισόδου.



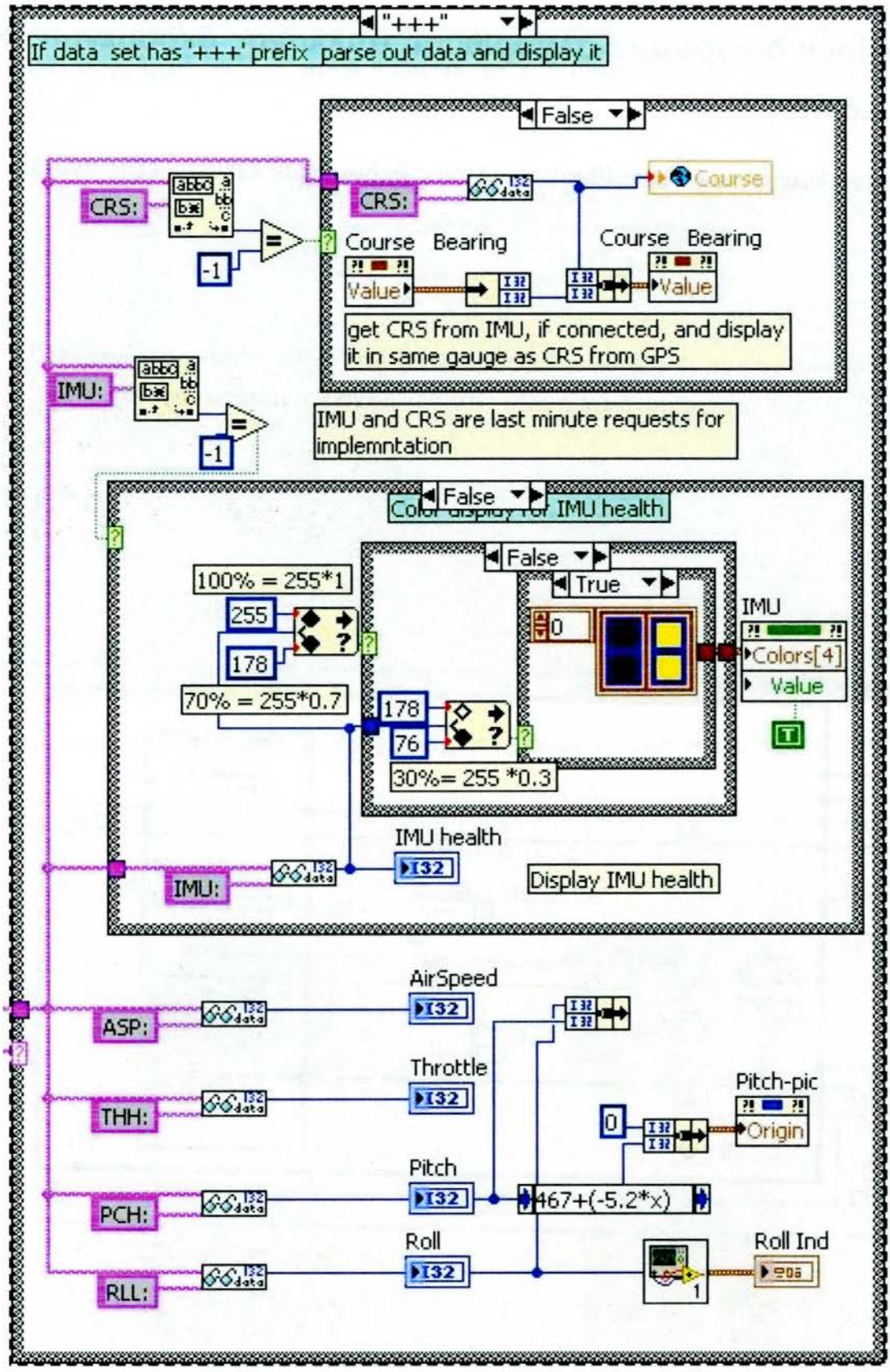
(6.2) Block διάγραμμα αλγορίθμου συλλογής δεδομένων και απεικόνιση

Η αλληλουχία του αλγορίθμου συλλογής δεδομένων έχει την εξής σειρά:

1. Την αρχικοποίηση τη σειριακής πόρτας και επικοινωνία με το μικρο ελεγκτή όπως φαίνεται στην παρακάτω εικόνα οπού αποτελεί ένα κομμάτι του block διαγράμματος.



2. Τη συλλογή δεδομένων και των διαχωρισμό των πληροφοριών. Όπως φαίνεται στο παρακάτω μπλοκ διάγραμμα.



(6.3) Διαδραστικός χάρτης για την ένδειξη θέσης του οχήματος και την εύρεση συντεταγμένων στα επιθυμητά σημεία που εισάγει ο χρήστης πάνω στο χάρτη.

Ο ψηφιακός χάρτης που επιλέχτηκε να συγχωνευθεί στην εφαρμογή είναι της εταιρίας Google και βασίζεται στη βάση δεδομένων της εφαρμογής Google Earth όπου διατίθεται δωρεάν.

(6.3.1) Εφαρμογη Google Earth

Το GoogleEarth είναι ένας εικονικός παγκόσμιος χάρτης με γεωγραφικές πληροφορίες . Χαρτογραφεί και παρατηρεί τη γη από δορυφορικές φωτογραφίες ταξινομώντας αυτές σύμφωνα με την θέση τους και την υψομετρική τους διαφορά δημιουργώντας ένα τρισδιάστατο μοντέλο της γης που μπορεί κανείς να περιηγηθεί εύκολα ,γρήγορα και ανέξοδα γιατί ή βάση δεδομένων του είναι ανοιχτή προς το κοινό . Το πιο σημαντικό όμως είναι πως οι φωτογραφίες από τις οποίες αποτελείται ο χάρτης, είναι σημαδεμένες σε κάθε σημείο τους από τις συντεταγμένες που αντιστοιχούν πάνω στη γη με αποτέλεσμα αν κάποιος χρήστης επιλέξει ένα σημείο πάνω στο χάρτη να του γίνεται γνωστό το γεωγραφικό πλάτος και γεωγραφικό μήκος του σημείου αυτού.

Το Google Earth σαν εφαρμογή, είναι ίσως ένας από τους πιο πληρέστερος ψηφιακούς χάρτες, αλλά μια τέτοια εφαρμογή θα μας ήταν τελείως άχρηστη αν δεν μπορούσε να συγχωνευθεί στη εφαρμογή που σχεδιάζεται και να ανταλλάσσει δεδομένα με αυτή .Ο τρόπος που έγινε αυτή η συγχώνευση θα περιγραφεί στις παρακάτω παραγράφους.

(6.3.2) **Google Earth API** (Application programming interface)

Η συγχώνευση του χάρτη στη εφαρμογή που σχεδιάστηκε για τη πλοήγηση του οχήματος έγινε χρησιμοποιώντας Google Earth API (Application programming interface)

Το Google Earth plug-in και JavaScript API επιτρέπει να συγχωνευθεί Google Earth, σε ιστοσελίδες. Χρησιμοποιώντας το API είναι εφικτό να σχεδιάστουν γραμμές και δείκτες, πάνω από τις εικόνες εδάφους του χάρτη,να προσθέθουν 3D μοντέλα, ή αρχεία KML, τα οποία επιτρέπουν την ανταλλαγή δεδομένων μεταξύ δύο εφαρμογών.

(6.3.3) **Ανταλλαγή δεδομένων μεταξύ εφαρμογής πλοήγησης και εφαρμογής Google Earth**

Για να γίνει δυνατή η ανταλλαγή δεδομένων του Google Earth και της εφαρμογής πλοήγησης πρέπει να δημιουργηθεί ένα αρχείο με μορφοποίηση KML

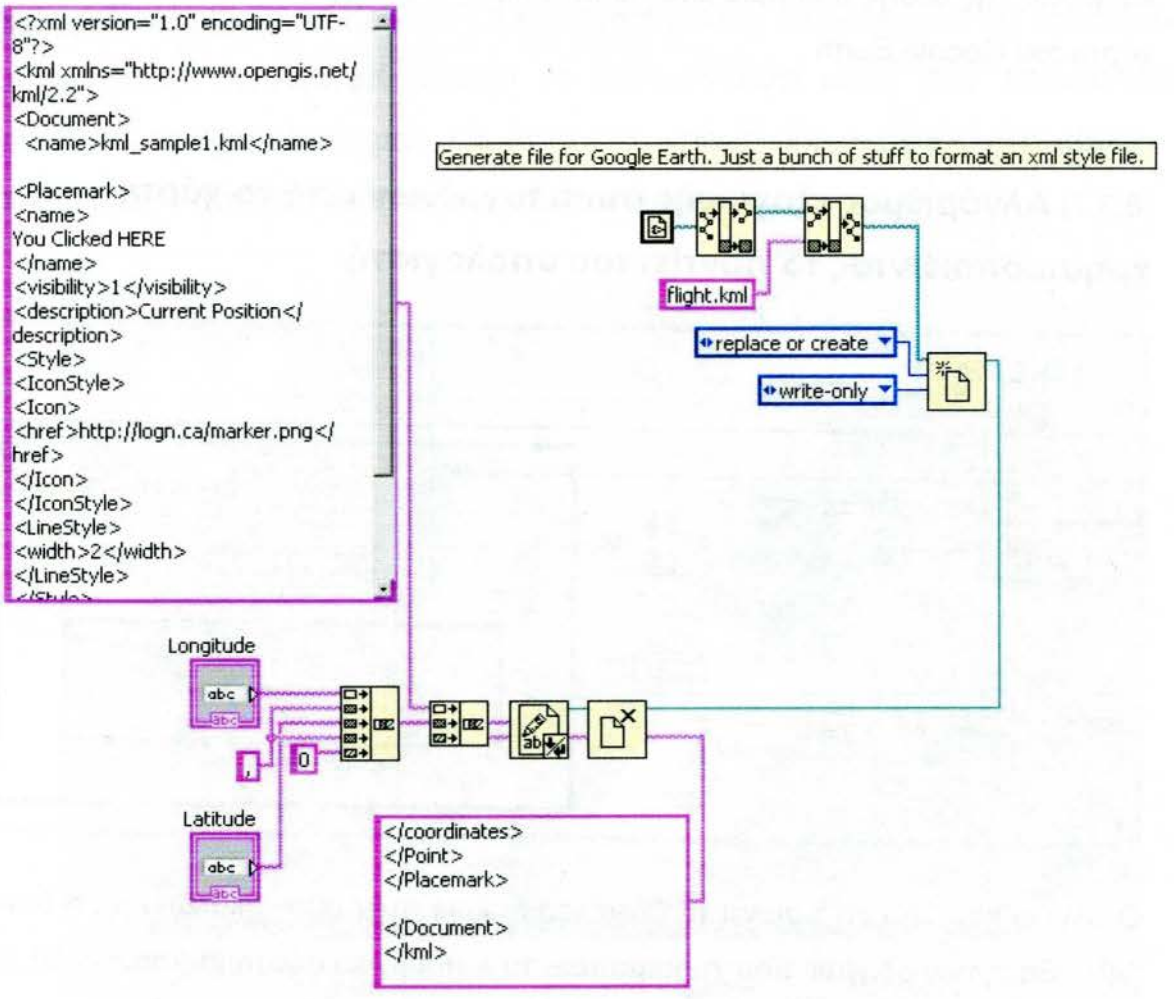
(6.3.4) **Αρχεία KML**

Τα KML αρχεία χρησιμοποιούνται για να εμφανίσουν γεωγραφικά δεδομένα σε ένα φιλομετρητή όπως είναι το Google Earth , Goggle maps. Τα KML αρχεία χρησιμοποιούν δομή ετικέτας με στοιχεία και γνωρίσματα βασισμένα σε δομή XML.

Η εφαρμογή πλοήγησης που σχεδιάστηκε, αφού ξεχωρίσει και απεικονίσει τις πληροφορίες πλοήγησης συνθέτει ένα καινούριο πακέτο το οποίο περιέχει

πληροφορίες που πρέπει να ανταλλαχθούν με την εφαρμογή Google Earth και το αποθηκεύει σαν αρχείο KML σε συγκεκριμένο μονοπάτι του δίσκου που .Το Google Earth αναλαμβάνει να διαβάσει αυτό το αρχείο και να απεικονίσει με τη σειρά του τη τρισδιάστατη θέση του οχήματος (γεωγραφικό πλάτος - γεωγραφικό μήκος-γεωγραφικό ύψος από την επιφάνεια της θάλασσας) , καθώς και τα επιλεγμένα σημεία που έχουμε σαν θέση στόχου (Way Points).

(6.3.5) **Block διάγραμμα αλγορίθμου ανταλλαγής δεδομένων διαμέσου του αρχείου KML**

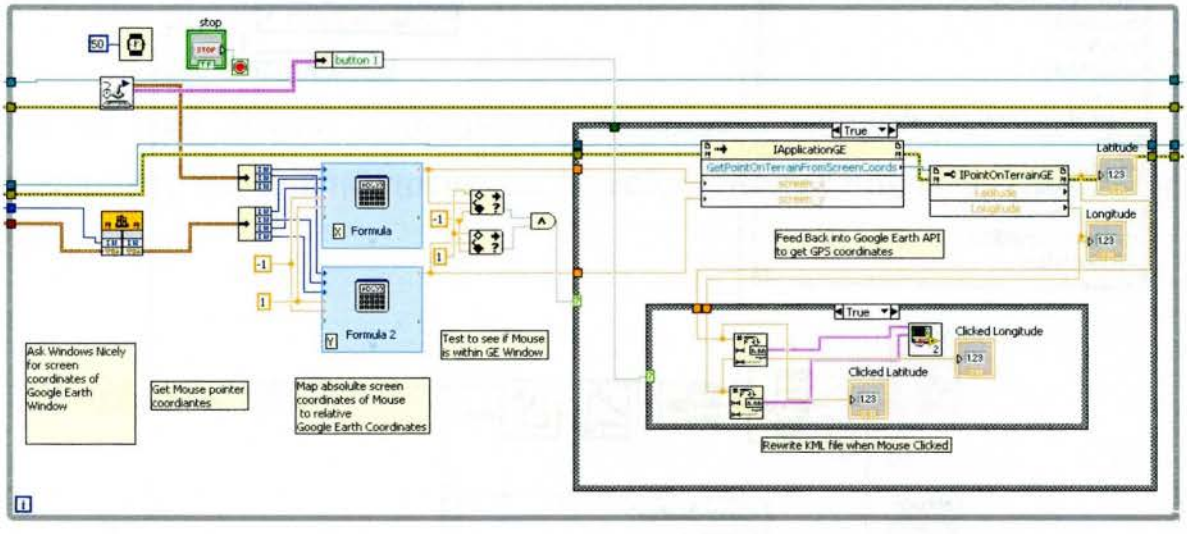


(6.3.6) Εξαγωγή συντεταγμένων ενός σημείου πάνω στο χάρτη

Για να υπολογιστεί η διαδρομή που πρέπει να ακολουθήσει το όχημα ώστε να μετακινηθεί από τη γεωγραφική θέση που βρίσκεται σε μια άλλη γεωγραφική θέση , και αν θεωρηθεί ότι η θέση που βρίσκεται το όχημα τη συγκεκριμένη στιγμή είναι γνωστή από το δέκτη GPS , αυτό που μένει άγνωστο είναι η θέση που επιθυμεί ο χρήστης να μετακινηθεί το όχημα. Οπότε ο χρήστης θα πρέπει να γνωρίζει τις ακριβές συντεταγμένες του σημείου αυτού , κάτι που είναι λίγο δύσχρηστο.

Οπότε ή λύση δόθηκε απο έναν αλγόριθμο που υπολογίσει το γεωγραφικό πλάτος και μήκος της θέσης που βρίσκεται το mouse, κάθε στιγμή που βρίσκεται πάνω στο χάρτη του Google Earth.

(6.3.7) Αλγόριθμος εξαγωγής συντεταγμένων από το χάρτη χρησιμοποιώντας το ποντίκι του υπολογιστή



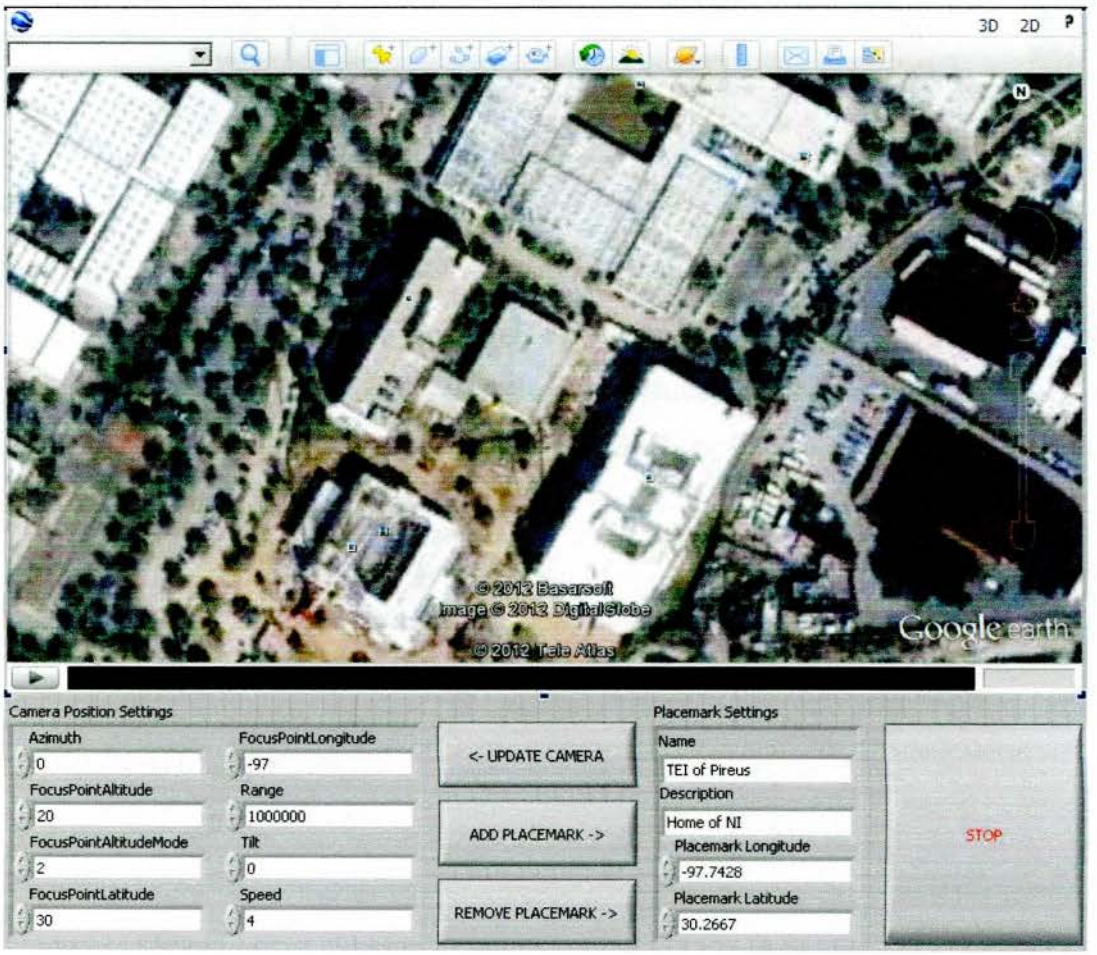
Ο αλγόριθμος αρχικά παίρνει τη θέση του mouse στην οθόνη καλώντας τη δυναμική βιβλιοθήκη των οδηγών που χρησιμοποιεί το λειτουργικό σύστημα (mouse.dll), έπειτα παίρνει την θέση του χάρτη στην οθόνη από το API του Google earth script και τριγωνομετρικά υπολογίζει τις συντεταγμένες της Γης που αντιστοιχούν στο σημείο που βρίσκετε το mouse πάνω στο χάρτη.

(6.3.8) Συγχώνευση του χάρτη στο γραφικό περιβάλλον της εφαρμογής

Για να συγχωνευθεί ο χάρτης στο γραφικό περιβάλλον της εφαρμογής χρησιμοποιήθηκε μια βιβλιοθήκη που περιέχεται στο Labview και δημιουργεί έναν φυλλομετρητή (web browser) μέσα στο γραφικό περιβάλλον του LABVIEW μέσα σε αυτόν απεικονίζεται ο χάρτης Google Earth μέσω της τεχνολογίας ActiveX.

(6.3.9) Τεχνολογία ActiveX

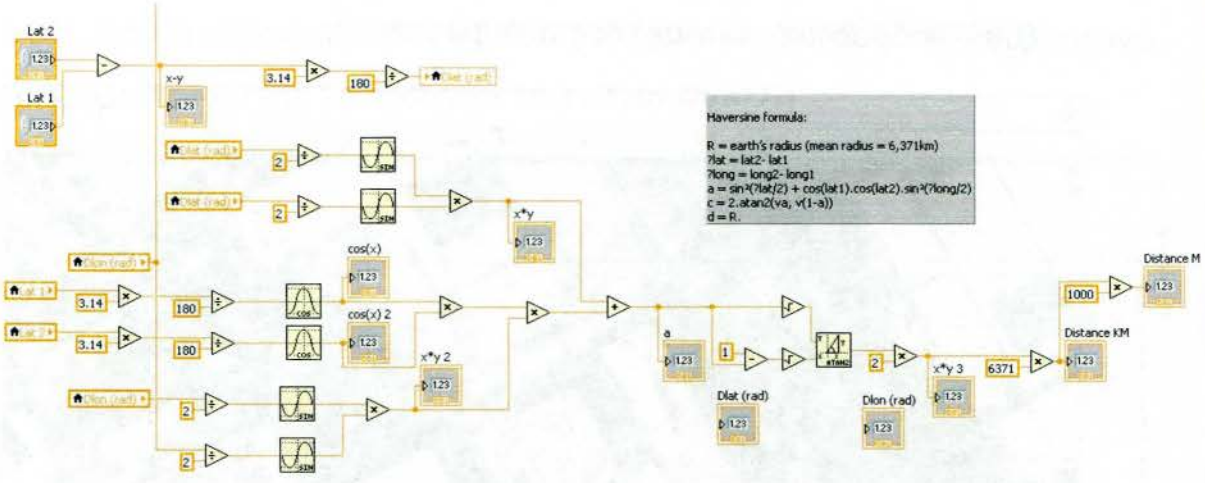
Το ActiveX είναι ένα πρόγραμμα της Microsoft που χρησιμοποιείται για να καθοδηγήσει ορισμένες τεχνολογίες στο πώς να συμπεριλάβουν τις COM (component object model) και OLE (object linking και embedding) λειτουργίες-συναρτήσεις. Το ActiveX μπορεί να ενσωματωθεί μέσα στις περισσότερες γλώσσες προγραμματισμού και επιτρέπει στους δημιουργούς ιστοχώρων να αναπτύξουν διαδραστικές ιστοσελίδες πολυμέσων.



(6.4) Υπολογισμός διαδρομής

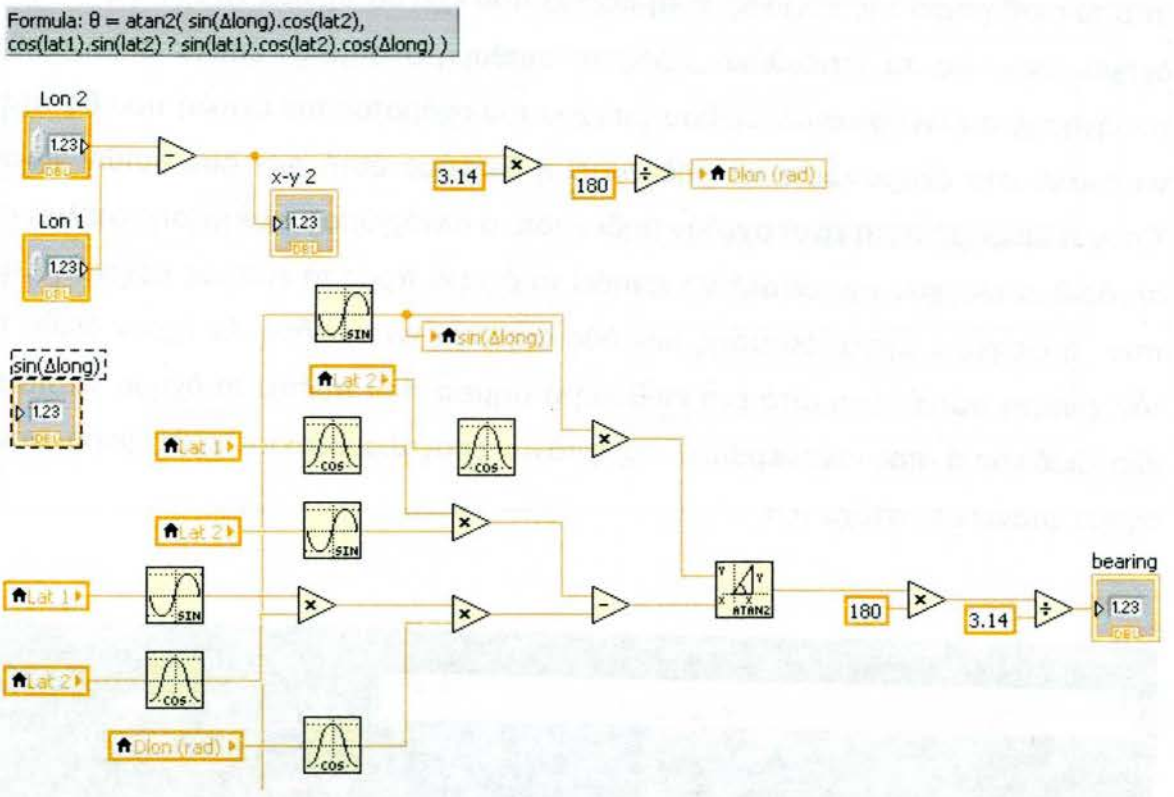
Για τον υπολογισμό της διαδρομής που πρέπει να ακολουθήσει το όχημα ώστε και να μεταφερθεί από το σημείο που βρίσκεται στο σημείο που έχει στόχο να πάει, αυτό που χρειάζεται είναι να εφαρμοστεί ο μαθηματικός τύπος Harvesine που περιγράφηκε σε προηγούμενο κεφάλαιο. Εισάγοντας στο μαθηματικό τύπο τις μετρήσεις από τα αισθητήρια που έχει συλλέξει ο μικρο ελεγκτής και τις συντεταγμένες του επιθυμητού σημείου που έχει ορίσει ο χρήστης ,το αποτέλεσμα του τύπου αυτού είναι η απόσταση από το σημείο που βρίσκεται το όχημα μέχρι το τελικό σημείο που πρέπει το όχημα να πάει.

(6.4.1) Αλγόριθμος εφαρμογής του μαθηματικού τύπου Harvesine στο προγραμματιστικό



(6.4.2) **Αλγόριθμος εφαρμογής του μαθηματικού τύπου υπολογισμού διόπτευσης**

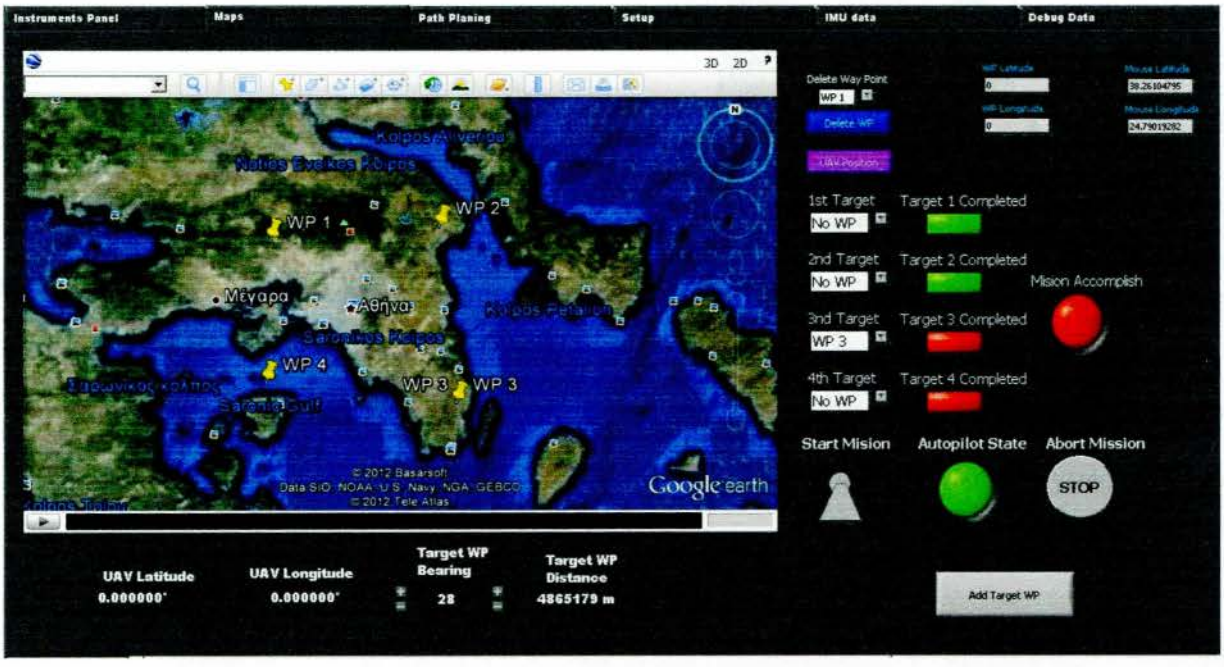
Η εφαρμογή που παρουσιάζεται στο παρακάτω διάγραμμα υπολογίζει τη διόπτευση του στόχου κάθε στιγμή, με το μαθηματικό τύπο υπολογισμού διόπτευσης που αναφέρθηκε σε προηγούμενο κεφάλαιο.



Η τιμή της διόπτευσης που αντιστοιχεί στη πορεία που πρέπει να διατηρήσει το όχημα ώστε να φτάσει στο στόχο, μεταβάλλεται συνεχώς καθώς αλλάζει η θέση του οχήματος ως προς το στόχο ώστε να διατηρηθεί η ορθοδρομική διαδρομή που έχει υπολογιστεί από το σύστημά.

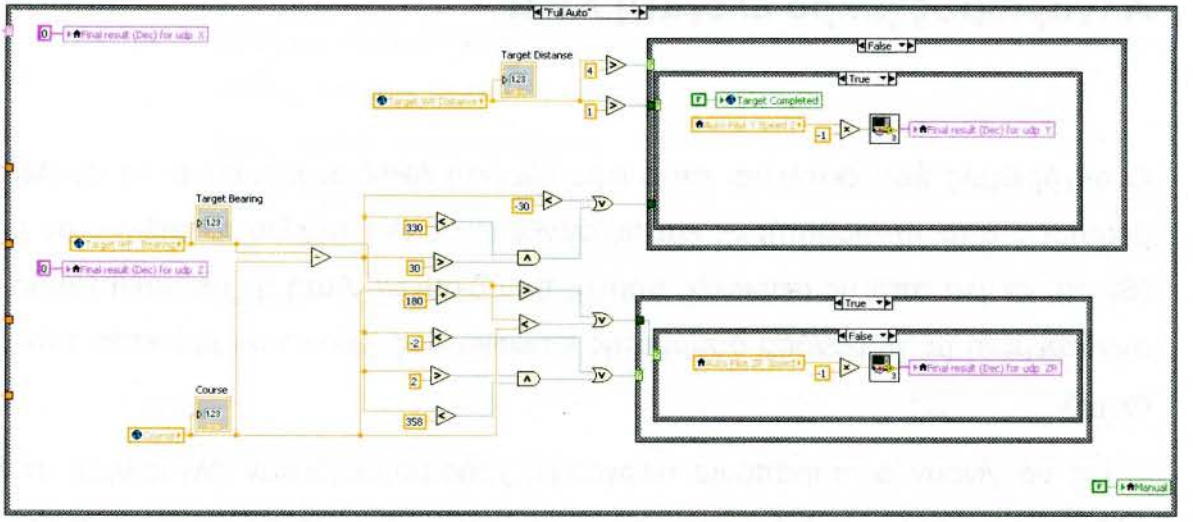
(6.4.3) **Πλοήγηση του οχήματος**

Η πλοήγηση του οχήματος γίνεται ως εξής , αφού είναι γνωστή η γεωγραφική θέση του οχήματος , γνωστή η ορθοδρομική απόσταση των δυο σημείων, και γνωστή η διόπτευση του στόχου τότε ο αλγόριθμος πλοήγησης συγκρίνει αρχικά τη μέτρηση από το αισθητήριο της πυξίδας με τη πορεία που έχει θα πρέπει να έχει σαν στόχο το όχημα όπου θα το κατευθύνει προς το επιθυμητό σημείο, έπειτα ο αλγόριθμος πλοήγησης στέλνει στον αλγόριθμο ελέγχου του οχήματος την εντολή που θα πρέπει να σταλεί στο όχημα ώστε να μηδενιστεί η διαφορά αυτή των δύο αυτών γωνιών. Όταν η διαφορά αυτή είναι σχεδόν μηδέν τότε ο αλγόριθμος πλοήγησης στέλνει στον αλγόριθμο ελέγχου την εντολή να κινηθεί το όχημα προς τα εμπρός μέχρι τη στιγμή που η διαφορά της απόστασης των δύο σημείων γίνει μηδέν. Αν έχουν δωθεί από τον χρήστη παραπάνω από ένα επιθυμητά σημεία που πρέπει το όχημα να πάει , η ίδια διαδικασία που αναφέρθηκε παραπάνω επαναλαμβάνεται κάθε φορά που το όχημα φτάνει στο στόχο του.



Στη παραπάνω εικόνα παρουσιάζεται η γραφική διεπαφή χρήστη (GUI graphical user interface) όπου δίνει τη δυνατότητα, επιλογής δρομολογίου και την ανατροφοδότηση από τη εξέλιξη τς αποστολής

(6.4.4) **Αλγόριθμος πλοήγησης οχήματος**



Κεφάλαιο 7

Αλγόριθμος μικρο ελεγκτή APM

Ο αλγόριθμος που εκτελείται στο μικρο ελεγκτή APM αναλαμβάνει να συλλέξει τις μετρήσεις από τα αισθητήρια, και τις συνθέτει σε ένα πακέτο το στέλνει σε μορφή (String) σε μια από τις σειριακές πόρτες που διαθέτει. Αυτή η σειριακή πόρτα είναι συνδεδεμένη με τη μονάδα ασύρματης επικοινωνίας XBee που βρίσκεται πάνω στο όχημά .

Για να γίνουν οι παραπάνω διεργασίες χρησιμοποιήθηκαν αλγόριθμοι ανοικτού κώδικα γραμμένοι από διάφορους συγγραφείς που συμμετέχουν σε ένα ανοικτό έργο κατασκευής UAV με την ονομασία Arduplane, Arducopter και βασίζεται στην πλατφόρμα μικρο εκλεκτή Ardupilot Mega που χρησιμοποιήθηκε και στο σύστημα που υλοποιήθηκε.

(7.1) Αλγόριθμος μικρο ελεγκτή συλλογής μετρήσεων από το αισθητήριο της πυξίδας

```
// -*- tab-width: 4; Mode: C++; c-basic-offset: 3; indent-tabs-mode: t -*-
/*
        APM_Compass.cpp - Arduino Library for HMC5843 I2C Magnetometer
        Code by Jordi MuÒoz and Jose Julio. DIYDrones.com

        This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
```

```
   version 2.1 of the License, or (at your option) any later version.

        Sensor is conected to I2C port
        Sensor is initialized in Continuos mode (10Hz)

        Variables:
            Heading : Magnetic heading
            Heading_X : Magnetic heading X component
            Heading_Y : Magnetic heading Y component
            Mag_X : Raw X axis magnetometer data
            Mag_Y : Raw Y axis magnetometer data
            Mag_Z : Raw Z axis magnetometer data
            lastUpdate : the time of the last successful reading

        Methods:
            Init() : Initialization of I2C and sensor
            Read() : Read Sensor data
            Calculate(float roll, float pitch) : Calculate tilt adjusted heading
            SetOrientation(const Matrix3f &rotationMatrix) : Set orientation of compass
            SetOffsets(int x, int y, int z) : Set adjustments for HardIron disturbances
            SetDeclination(float radians) : Set heading adjustment between true north and magnetic north

        To do : code optimization
        Mount position : UPDATED
      Big capacitor pointing backward, connector forward

*/
extern "C" {
  // AVR LibC Includes
  #include <math.h>
  #include "WConstants.h"
}

#include <Wire.h>
#include "APM_Compass.h"

#define CompassAddress      0x1E
```

```
#define ConfigRegA          0x00
#define ConfigRegB          0x01
#define MagGain             0x20
#define PositiveBiasConfig  0x11
#define NegativeBiasConfig  0x12
#define NormalOperation     0x10
#define ModeRegister        0x02
#define ContinuousConversion 0x00
#define SingleConversion    0x01

// Constructors ////////////////////////////////////////////////////////////
APM_Compass_Class::APM_Compass_Class() : orientation(0), declination(0.0)
{
  // mag x y z offset initialisation
  offset[0] = 0;
  offset[1] = 0;
  offset[2] = 0;

  // initialise orientation matrix
  orientationMatrix = ROTATION_NONE;
}

// Public Methods //////////////////////////////////////////////////////////
bool APM_Compass_Class::Init(int initialiseWireLib)
{
  unsigned long currentTime = millis();  // record current time
  int numAttempts = 0;
  int success = 0;

  if( initialiseWireLib != 0 )
     Wire.begin();

  delay(10);

  // calibration initialisation
  calibration[0] = 1.0;
  calibration[1] = 1.0;
```

```
calibration[2] = 1.0;


while( success == 0 && numAttempts < 5 )
{
  // record number of attempts at initialisation
        numAttempts++;

        // force positiveBias (compass should return 715 for all channels)
        Wire.beginTransmission(CompassAddress);
        Wire.send(ConfigRegA);
        Wire.send(PositiveBiasConfig);
        if (0 != Wire.endTransmission())
                continue;                       // compass not responding on the bus
        delay(50);

        // set gains
        Wire.beginTransmission(CompassAddress);
        Wire.send(ConfigRegB);
        Wire.send(MagGain);
        Wire.endTransmission();
        delay(10);

        Wire.beginTransmission(CompassAddress);
        Wire.send(ModeRegister);
        Wire.send(SingleConversion);
        Wire.endTransmission();
        delay(10);

        // read values from the compass
        Read();
        delay(10);

        // calibrate
        if( abs(Mag_X) > 500 && abs(Mag_X) < 1000 && abs(Mag_Y) > 500 && abs(Mag_Y) < 1000 && abs(Mag_Z) >
500 && abs(Mag_Z) < 1000)
        {
                calibration[0] = fabs(715.0 / Mag_X);
                calibration[1] = fabs(715.0 / Mag_Y);
```

```
                calibration[2] = fabs(715.0 / Mag_Z);

                // mark success
                success = 1;
            }

            // leave test mode
            Wire.beginTransmission(CompassAddress);
            Wire.send(ConfigRegA);
            Wire.send(NormalOperation);
            Wire.endTransmission();
            delay(50);

            Wire.beginTransmission(CompassAddress);
            Wire.send(ModeRegister);
            Wire.send(ContinuousConversion);      // Set continuous mode (default to 10Hz)
            Wire.endTransmission();               // End transmission
            delay(50);
  }
  return(success);
}

// Read Sensor data
void APM_Compass_Class::Read()
{
  int i = 0;
  byte buff[6];

  Wire.beginTransmission(CompassAddress);
  Wire.send(0x03);      //sends address to read from
  Wire.endTransmission(); //end transmission

  //Wire.beginTransmission(CompassAddress);
  Wire.requestFrom(CompassAddress, 6);   // request 6 bytes from device
  while(Wire.available())
  {
    buff[i] = Wire.receive();  // receive one byte
```

```
    i++;
  }
  Wire.endTransmission(); //end transmission

  if (i==6)  // All bytes received?
  {
    // MSB byte first, then LSB, X,Y,Z
    Mag_X = -((((int)buff[0]) << 8) | buff[1]) * calibration[0];    // X axis
    Mag_Y = ((((int)buff[2]) << 8) | buff[3]) * calibration[1];    // Y axis
    Mag_Z = -((((int)buff[4]) << 8) | buff[5]) * calibration[2];    // Z axis
    lastUpdate = millis();  // record time of update
  }
}

void APM_Compass_Class::Calculate(float roll, float pitch)
{
  float Head_X;
  float Head_Y;
  float cos_roll;
  float sin_roll;
  float cos_pitch;
  float sin_pitch;
  Vector3f rotMagVec;

  cos_roll = cos(roll);  // Optimizacion, se puede sacar esto de la matriz DCM?
  sin_roll = sin(roll);
  cos_pitch = cos(pitch);
  sin_pitch = sin(pitch);

  // rotate the magnetometer values depending upon orientation
  if( orientation == 0 )
     rotMagVec = Vector3f(Mag_X+offset[0],Mag_Y+offset[1],Mag_Z+offset[2]);
  else
     rotMagVec = orientationMatrix*Vector3f(Mag_X+offset[0],Mag_Y+offset[1],Mag_Z+offset[2]);

  // Tilt compensated Magnetic field X component:
  Head_X = rotMagVec.x*cos_pitch+rotMagVec.y*sin_roll*sin_pitch+rotMagVec.z*cos_roll*sin_pitch;
```

```
    // Tilt compensated Magnetic field Y component:
    Head_Y = rotMagVec.y*cos_roll-rotMagVec.z*sin_roll;
    // Magnetic Heading
    Heading = atan2(-Head_Y,Head_X);

    // Declination correction (if supplied)
    if( declination != 0.0 )
    {
        Heading = Heading + declination;
        if (Heading > M_PI)    // Angle normalization (-180 deg, 180 deg)
            Heading -= (2.0 * M_PI);
        else if (Heading < -M_PI)
            Heading += (2.0 * M_PI);
    }

    // Optimization for external DCM use. Calculate normalized components
    Heading_X = cos(Heading);
    Heading_Y = sin(Heading);
}

void APM_Compass_Class::SetOrientation(const Matrix3f &rotationMatrix)
{
    orientationMatrix = rotationMatrix;
            if( orientationMatrix == ROTATION_NONE )
                orientation = 0;
            else
                orientation = 1;
}

void APM_Compass_Class::SetOffsets(int x, int y, int z)
{
    offset[0] = x;
            offset[1] = y;
            offset[2] = z;
}

void APM_Compass_Class::SetDeclination(float radians)
```

```
{
    declination = radians;
}



// Constructors ////////////////////////////////////////////////////////////
APM_Compass_HIL_Class::APM_Compass_HIL_Class() : orientation(0), declination(0.0)
{
  // mag x y z offset initialisation
  offset[0] = 0;
  offset[1] = 0;
  offset[2] = 0;

  // initialise orientation matrix
  orientationMatrix = ROTATION_NONE;
}

// Public Methods //////////////////////////////////////////////////////////
booi APM_Compass_HIL_Class::Init(int initialiseWireLib)
{
  unsigned long currentTime = millis();  // record current time
  int numAttempts = 0;
  int success = 0;

  // calibration initialisation
  calibration[0] = 1.0;
  calibration[1] = 1.0;
  calibration[2] = 1.0;

  while( success == 0 && numAttempts < 5 )
  {
    // record number of attempts at initialisation
            numAttempts++;

            // read values from the compass
            Read();
            delay(10);
```

```
            // calibrate
            if( abs(Mag_X) > 500 && abs(Mag_X) < 1000 && abs(Mag_Y) > 500 && abs(Mag_Y) < 1000 && abs(Mag_Z) >
500 && abs(Mag_Z) < 1000)
            {
                calibration[0] = fabs(715.0 / Mag_X);
                calibration[1] = fabs(715.0 / Mag_Y);
                calibration[2] = fabs(715.0 / Mag_Z);

                // mark success
                success = 1;
            }
 }
 return(success);
}

// Read Sensor data
void APM_Compass_HIL_Class::Read()
{
   // values set by setHIL function
}

void APM_Compass_HIL_Class::Calculate(float roll, float pitch)
{
 float Head_X;
 float Head_Y;
 float cos_roll;
 float sin_roll;
 float cos_pitch;
 float sin_pitch;
 Vector3f rotMagVec;

 cos_roll = cos(roll);  // Optimizacion, se puede sacar esto de la matriz DCM?
 sin_roll = sin(roll);
 cos_pitch = cos(pitch);
 sin_pitch = sin(pitch);

 // rotate the magnetometer values depending upon orientation
```

```
 if( orientation == 0 )
    rotMagVec = Vector3f(Mag_X+offset[0],Mag_Y+offset[1],Mag_Z+offset[2]);
 else
    rotMagVec = orientationMatrix*Vector3f(Mag_X+offset[0],Mag_Y+offset[1],Mag_Z+offset[2]);

 // Tilt compensated Magnetic field X component:
 Head_X = rotMagVec.x*cos_pitch+rotMagVec.y*sin_roll*sin_pitch+rotMagVec.z*cos_roll*sin_pitch;
 // Tilt compensated Magnetic field Y component:
 Head_Y = rotMagVec.y*cos_roll-rotMagVec.z*sin_roll;
 // Magnetic Heading
 Heading = atan2(-Head_Y,Head_X);

 // Declination correction (if supplied)
 if( declination != 0.0 )
 {
    Heading = Heading + declination;
    if (Heading > M_PI)    // Angle normalization (-180 deg, 180 deg)
       Heading -= (2.0 * M_PI);
    else if (Heading < -M_PI)
       Heading += (2.0 * M_PI);
 }

 // Optimization for external DCM use. Calculate normalized components
 Heading_X = cos(Heading);
 Heading_Y = sin(Heading);
}

void APM_Compass_HIL_Class::SetOrientation(const Matrix3f &rotationMatrix)
{
   orientationMatrix = rotationMatrix;
            if( orientationMatrix == ROTATION_NONE )
               orientation = 0;
            else
               orientation = 1;
}

void APM_Compass_HIL_Class::SetOffsets(int x, int y, int z)
```

```
{
    offset[0] = x;
                offset[1] = y;
                offset[2] = z;
}

void APM_Compass_HIL_Class::SetDeclination(float radians)
{
    declination = radians;
}

void APM_Compass_HIL_Class::setHIL(float _Mag_X, float _Mag_Y, float _Mag_Z)
{
    // TODO: map floats to raw
    Mag_X = _Mag_X;
    Mag_Y = _Mag_Y;
    Mag_Z = _Mag_Z;
}
```

(7.2) Αλγόριθμος μικρο ελεγκτή συλλογής μετρήσεων από το δέκτη GPS

```
// -*- tab-width: 4; Mode: C++; c-basic-offset: 4; indent-tabs-mode: t -*-
//
// u-blox UBX GPS driver for ArduPilot and ArduPilotMega.
//          Code by Michael Smith, Jordi Munoz and Jose Julio, DIYDrones.com
//
//          This library is free software; you can redistribute it and / or
//          modify it under the terms of the GNU Lesser General Public
//          License as published by the Free Software Foundation; either
//          version 2.1 of the License, or (at your option) any later version.
//

#include "AP_GPS_UBLOX.h"
```

```
#include <stdint.h>

// Constructors ////////////////////////////////////////////////////////////////

AP_GPS_UBLOX::AP_GPS_UBLOX(Stream *s) : GPS(s)
{
}

// Public Methods //////////////////////////////////////////////////////////////

void
AP_GPS_UBLOX::init(void)
{
        // XXX it might make sense to send some CFG_MSG,CFG_NMEA messages to get the
        // right reporting configuration.

        _port->flush();
}

// Process bytes available from the stream
//
// The stream is assumed to contain only messages we recognise.  If it
// contains other messages, and those messages contain the preamble
// bytes, it is possible for this code to fail to synchronise to the
// stream immediately.  Without buffering the entire message and
// re-processing it from the top, this is unavoidable. The parser
// attempts to avoid this when possible.
//
bool
AP_GPS_UBLOX::read(void)
{
        uint8_t         data;
        int             numc;
        bool            parsed = false;

        numc = _port->available();
        for (int i = 0; i < numc; i++){     // Process bytes received
```

```
// read the next byte
data = _port->read();

switch(_step){

        // Message preamble detection
        //
        // If we fail to match any of the expected bytes, we reset
        // the state machine and re-consider the failed byte as
        // the first byte of the preamble.  This improves our
        // chances of recovering from a mismatch and makes it less
        // likely that we will be fooled by the preamble appearing
        // as data in some other message.
        //
case 1:
        if (PREAMBLE2 == data) {
                _step++;
                break;
        }
        _step = 0;
        // FALLTHROUGH
case 0:
        if(PREAMBLE1 == data)
                _step++;
        break;

        // Message header processing
        //
        // We sniff the class and message ID to decide whether we
        // are going to gather the message bytes or just discard
        // them.
        //
        // We always collect the length so that we can avoid being
        // fooled by preamble bytes in messages.
        //
case 2:
```

```
                _step++;
                if (CLASS_NAV == data) {
                        _gather = true;                         // class is interesting,
maybe gather
                        _ck_b = _ck_a = data;                   // reset the checksum
accumulators
                } else {
                        _gather = false;                        // class is not interesting, discard
                }
                break;
        case 3:
                _step++;
                _ck_b += (_ck_a += data);                       // checksum byte
                _msg_id = data;
                if (_gather) {                                  // if class was
interesting
                        switch(data) {
                        case MSG_POSLLH:                        // message is
interesting
                                _expect = sizeof(ubx_nav_posllh);
                                break;
                        case MSG_STATUS:
                                _expect = sizeof(ubx_nav_status);
                                break;
                        case MSG_SOL:
                                _expect = sizeof(ubx_nav_solution);
                                break;
                        case MSG_VELNED:
                                _expect = sizeof(ubx_nav_velned);
                                break;
                        default:
                                _gather = false;                // message is not interesting
                        }
                }
                break;
        case 4:
                _step++;
                _ck_b += (_ck_a += data);                       // checksum byte
                _payload_length = data;                         // payload length low byte
```

```
                break;
        case 5:
                _step++;
                _ck_b += (_ck_a += data);                       // checksum byte
                _payload_length += (uint16_t)data;      // payload length high byte
                _payload_counter = 0;                                   // prepare to receive payload
                if (_payload_length != _expect)
                        _gather = false;
                break;

                // Receive message data
                //
        case 6:
                _ck_b += (_ck_a += data);                       // checksum byte
                if (_gather)                                                    // gather data if
requested
                        _buffer.bytes[_payload_counter] = data;
                if (++_payload_counter == _payload_length)
                        _step++;
                break;

                // Checksum and message processing
                //
        case 7:
                _step++;
                if (_ck_a != data)
                        _step = 0;                                              // bad checksum
                break;
        case 8:
                _step = 0;
                if (_ck_b != data)
                        break;                                                  // bad
checksum

                if (_gather) {
                        parsed = _parse_gps();                  // Parse the new GPS packet
                }
        }
```

```
        }
        return parsed;
}

// Private Methods ////////////////////////////////////////////////////////////

bool
AP_GPS_UBLOX::_parse_gps(void)
{
        switch (_msg_id) {
        case MSG_POSLLH:
                time                    = _buffer.posllh.time;
                longitude = _buffer.posllh.longitude;
                latitude    = _buffer.posllh.latitude;
                altitude    = _buffer.posllh.altitude_msl / 10;
                break;
        case MSG_STATUS:
                fix                             = (_buffer.status.fix_status & NAV_STATUS_FIX_VALID) &&
(_buffer.status.fix_type == FIX_3D);
                break;
        case MSG_SOL:
                fix                             = (_buffer.solution.fix_status & NAV_STATUS_FIX_VALID) &&
(_buffer.solution.fix_type == FIX_3D);
                num_sats = _buffer.solution.satellites;
                break;
        case MSG_VELNED:
                speed_3d = _buffer.velned.speed_3d;                                     // cm/s
                ground_speed = _buffer.velned.speed_2d;                                 // cm/s
                ground_course = _buffer.velned.heading_2d / 1000;       // Heading 2D deg * 100000 rescaled to deg
* 100
                break;
        default:
                return false;
        }
        return true;
}
```

Αλγόριθμος μικροελεγκτή συλλογής μετρήσεων από τα επιταχυνσιόμετρα , γυροσκόπια και υπολογισμός γονιών Pitch, Roll

```
APM_DCM_FW.cpp - DCM AHRS Library, fixed wing version, for Ardupilot Mega
        Code by Doug Weibel, Jordi MuÒoz and Jose Julio. DIYDrones.com

This library works with the ArduPilot Mega and "Oilpan"

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

  Methods:
                update_DCM(_G_Dt)       : Updates the AHRS by integrating the rotation matrix over time
_Dt using the IMU object data
                get_gyro()                      : Returns gyro vector corrected for bias
                get_accel()             : Returns accelerometer vector
                get_dcm_matrix()  : Returns dcm matrix


clude <AP_DCM.h>

efine OUTPUTMODE 1                      // This is just used for debugging, remove later
efine ToRad(x) (x*0.01745329252)  // *pi/180
efine ToDeg(x) (x*57.2957795131)  // *180/pi

define Kp_ROLLPITCH 0.05967             // .0014 * 418/9.81 Pitch&Roll Drift Correction Proportional Gain
define Ki_ROLLPITCH 0.00001278          // 0.0000003 * 418/9.81 Pitch&Roll Drift Correction Integrator Gain
define Ki_ROLLPITCH 0.0                 // 0.0000003 * 418/9.81 Pitch&Roll Drift Correction Integrator Gain
```

```
'#define Kp_YAW 0.8                          // Yaw Drift Correction Porportional Gain
'#define Ki_YAW 0.00004                      // Yaw Drift CorrectionIntegrator Gain

#define SPEEDFILT 300                       // centimeters/second
#define ADC_CONSTRAINT 900


oid
AP_DCM::set_compass(Compass *compass)

    _compass = compass;


**********************************************/
oid
AP_DCM::update_DCM(float _G_Dt)

    _imu->update();
    _gyro_vector    = _imu->get_gyro();         // Get current values for IMU sensors
    _accel_vector   = _imu->get_accel();            // Get current values for IMU sensors

    matrix_update(_G_Dt);       // Integrate the DCM matrix

    //if(_toggle){
        normalize();                    // Normalize the DCM matrix
    //}else{
        drift_correction();         // Perform drift correction
    //}
    //_toggle = !_toggle;

    euler_angles();                 // Calculate pitch, roll, yaw for stabilization and navigation


**********************************************/

  //For Debugging
```

```
id

ntm(const char *l, Matrix3f &m)

    Serial.println(" "); Serial.println(l);

    Serial.print(m.a.x, 12); Serial.print(" "); Serial.print(m.a.y, 12); Serial.print(" "); Serial.println(m.a.z, 12);

    Serial.print(m.b.x, 12); Serial.print(" "); Serial.print(m.b.y, 12); Serial.print(" "); Serial.println(m.b.z, 12);

    Serial.print(m.c.x, 12); Serial.print(" "); Serial.print(m.c.y, 12); Serial.print(" "); Serial.println(m.c.z, 12);

    Serial.print(*(uint32_t *)&(m.a.x), HEX); Serial.print(" "); Serial.print(*(uint32_t *)&(m.a.y), HEX); Serial.print(" ");
erial.println(*(uint32_t *)&(m.a.z), HEX);

    Serial.print(*(uint32_t *)&(m.b.x), HEX); Serial.print(" "); Serial.print(*(uint32_t *)&(m.b.y), HEX); Serial.print(" ");
erial.println(*(uint32_t *)&(m.b.z), HEX);

    Serial.print(*(uint32_t *)&(m.c.x), HEX); Serial.print(" "); Serial.print(*(uint32_t *)&(m.c.y), HEX); Serial.print(" ");
erial.println(*(uint32_t *)&(m.c.z), HEX);




**********************************************/

oid

P_DCM::matrix_update(float _G_Dt)


    Matrix3f    update_matrix;

    Matrix3f    temp_matrix;


    //Record when you saturate any of the gyros.

    if( (fabs(_gyro_vector.x) >= radians(300)) ||

            (fabs(_gyro_vector.y) >= radians(300)) ||

            (fabs(_gyro_vector.z) >= radians(300))){

            gyro_sat_count++;

    }


    _omega_integ_corr   = _gyro_vector              + _omega_I;             // Used for _centripetal correction
heoretically better than _omega)

    _omega                          = _omega_integ_corr + _omega_P;         // Equation 16, adding
roportional and integral correction terms


    if(_centripetal){

            accel_adjust();                         // Remove _centripetal acceleration.

    }


#if OUTPUTMODE == 1
```

```
    float tmp = _G_Dt * _omega.x;
    update_matrix.b.z = -tmp;                   // -delta Theta x
    update_matrix.c.y =  tmp;                   // delta Theta x

    tmp = _G_Dt * _omega.y;
    update_matrix.c.x = -tmp;                   // -delta Theta y
    update_matrix.a.z =  tmp;                   // delta Theta y

    tmp = _G_Dt * _omega.z;
    update_matrix.b.x =  tmp;                   // delta Theta z
    update_matrix.a.y = -tmp;                   // -delta Theta z

    update_matrix.a.x = 0;
    update_matrix.b.y = 0;
    update_matrix.c.z = 0;
else                                                    // Uncorrected data (no drift
rrection)
    update_matrix.a.x = 0;
    update_matrix.a.y = -_G_Dt * _gyro_vector.z;
    update_matrix.a.z =  _G_Dt * _gyro_vector.y;
    update_matrix.b.x =  _G_Dt * _gyro_vector.z;
    update_matrix.b.y = 0;
    update_matrix.b.z = -_G_Dt * _gyro_vector.x;
    update_matrix.c.x = -_G_Dt * _gyro_vector.y;
    update_matrix.c.y =  _G_Dt * _gyro_vector.x;
    update_matrix.c.z = 0;
endif

    temp_matrix = _dcm_matrix * update_matrix;
    _dcm_matrix = _dcm_matrix + temp_matrix;        // Equation 17

**********************************************/
id
P_DCM::accel_adjust(void)

    Vector3f veloc, temp;
```

```
    veloc.x = _gps->ground_speed / 100;                    // We are working with acceleration in m/s^2 units

    // We are working with a modified version of equation 26 as our IMU object reports acceleration in the positive axis direction as
itive

    //_accel_vector -= _omega_integ_corr % _veloc;                    // Equation 26  This line is giving the compiler a problem so we
ak it up below
    temp.x = 0;
    temp.y = _omega_integ_corr.z * veloc.x;                    // only computing the non-zero terms
    temp.z = -1.0f * _omega_integ_corr.y * veloc.x;        // After looking at the compiler issue lets remove _veloc and simlify

    _accel_vector -= temp;
```

```
**********************************************
ection Cosine Matrix IMU: Theory
liam Premerlani and Paul Bizard

merical errors will gradually reduce the orthogonality conditions expressed by equation 5
approximations rather than identities. In effect, the axes in the two frames of reference no
ger describe a rigid body. Fortunately, numerical error accumulates very slowly, so it is a
ple matter to stay ahead of it.
 call the process of enforcing the orthogonality conditions "renormalization".

d
_DCM::normalize(void)

    float error = 0;
    Vector3f     temporary[3];

    int problem = 0;

    error = _dcm_matrix.a * _dcm_matrix.b;                                                    // eq.18

    temporary[0] = _dcm_matrix.b;
    temporary[1] = _dcm_matrix.a;
    temporary[0] = _dcm_matrix.a - (temporary[0] * (0.5f * error));                // eq.19
```

```
        temporary[1] = _dcm_matrix.b - (temporary[1] * (0.5f * error));                 // eq.19

        temporary[2] = temporary[0] % temporary[1];                                                                     // c= a x b //
q.20

        _dcm_matrix.a = renorm(temporary[0], problem);
        _dcm_matrix.b = renorm(temporary[1], problem);
        _dcm_matrix.c = renorm(temporary[2], problem);

        if (problem == 1) {             // Our solution is blowing up and we will force back to initial condition.   Hope we are not upside
own!
                _dcm_matrix.a.x = 1.0f;
                _dcm_matrix.a.y = 0.0f;
                _dcm_matrix.a.z = 0.0f;
                _dcm_matrix.b.x = 0.0f;
                _dcm_matrix.b.y = 1.0f;
                _dcm_matrix.b.z = 0.0f;
                _dcm_matrix.c.x = 0.0f;
                _dcm_matrix.c.y = 0.0f;
                _dcm_matrix.c.z = 1.0f;
        }


**************************************************/
Vector3f
AP_DCM::renorm(Vector3f const &a, int &problem)

        float           renorm_val;

        renorm_val = a * a;

        if (renorm_val < 1.5625f && renorm_val > 0.64f) {                       // Check if we are OK to use Taylor expansion
                renorm_val = 0.5 * (3 - renorm_val);                                                    // eq.21
        } else if (renorm_val < 100.0f && renorm_val > 0.01f) {
                renorm_val = 1.0 / sqrt(renorm_val);
                renorm_sqrt_count++;
        } else {
                problem = 1;
```

```
            renorm_blowup_count++;
    }

    return(a * renorm_val);



**********************************************/
id
'_DCM::drift_correction(void)

    //Compensation the Roll, Pitch and Yaw drift.
    //float mag_heading_x;
    //float mag_heading_y;
    float error_course;
    float accel_magnitude;
    float accel_weight;
    float integrator_magnitude;
    //static float scaled_omega_P[3];
    //static float scaled_omega_I[3];
    static bool in_motion = false;
    Matrix3f rot_mat;

    //*****Roll and Pitch***************

    // Calculate the magnitude of the accelerometer vector
    accel_magnitude = _accel_vector.length() / 9.80665f;

    // Dynamic weighting of accelerometer info (reliability filter)
    // Weight for accelerometer info (<0.5G = 0.0, 1G = 1.0 , >1.5G = 0.0)
    accel_weight = constrain(1 - 3 * fabs(1 - accel_magnitude), 0, 1);          // upped to (<0.66G = 0.0, 1G = 1.0 , >1.33G = 0.0)

    //          We monitor the amount that the accelerometer based drift correction is deweighted for performance reporting
    _health = constrain(_health+(0.02 * (accel_weight - .5)), 0, 1);

    // adjust the ground of reference
    _error_roll_pitch =  _dcm_matrix.c % _accel_vector;                         // Equation 27     *** sign changed from prev
plementation???
```

```
// error_roll_pitch are in Accel m/s^2 units
// Limit max error_roll_pitch to limit max omega_P and omega_I
_error_roll_pitch.x = constrain(_error_roll_pitch.x, -1.17f, 1.17f);
_error_roll_pitch.y = constrain(_error_roll_pitch.y, -1.17f, 1.17f);
_error_roll_pitch.z = constrain(_error_roll_pitch.z, -1.17f, 1.17f);

_omega_P =  _error_roll_pitch * (_kp_roll_pitch * accel_weight);
_omega_I += _error_roll_pitch * (_ki_roll_pitch * accel_weight);


//*****YAW***************

if (_compass) {
        // We make the gyro YAW drift correction based on compass magnetic heading
error_course = (_dcm_matrix.a.x * _compass->heading_y) - (_dcm_matrix.b.x * _compass->heading_x);
Equation 23, Calculating YAW error

} else {

        // Use GPS Ground course to correct yaw gyro drift
        if (_gps->ground_speed >= SPEEDFILT) {

                _course_over_ground_x = cos(ToRad(_gps->ground_course/100.0));
                _course_over_ground_y = sin(ToRad(_gps->ground_course/100.0));
                if(in_motion) {
                        error_course = (_dcm_matrix.a.x * _course_over_ground_y) - (_dcm_matrix.b.x *
_course_over_ground_x);      // Equation 23, Calculating YAW error
                } else {
                        float cos_psi_err, sin_psi_err;
                        // This is the case for when we first start moving and reset the DCM so that yaw matches the
ps ground course

                        // This is just to get a reasonable estimate faster
                        yaw = atan2(_dcm_matrix.b.x, _dcm_matrix.a.x);
                        cos_psi_err = cos(ToRad(_gps->ground_course/100.0) - yaw);
                        sin_psi_err = sin(ToRad(_gps->ground_course/100.0) - yaw);
                        // Rxx = cos psi err, Rxy = - sin psi err, Rxz = 0
                        // Ryx = sin psi err, Ryy = cos psi err,   Ryz = 0
                        // Rzx = Rzy = 0, Rzz = 1
```

```
                rot_mat.a.x = cos_psi_err;
                rot_mat.a.y = -sin_psi_err;
                rot_mat.b.x = sin_psi_err;
                rot_mat.b.y = cos_psi_err;
                rot_mat.a.z = 0;
                rot_mat.b.z = 0;
                rot_mat.c.x = 0;
                rot_mat.c.y = 0;
                rot_mat.c.z = 1.0;

                _dcm_matrix = rot_mat * _dcm_matrix;
                in_motion =  true;
                error_course = 0;
            }

        } else {
            error_course = 0;
            in_motion = false;
        }
    }

    _error_yaw = _dcm_matrix.c * error_course;          // Equation 24, Applys the yaw correction to the XYZ rotation of the aircraft,
epeding the position.

    _omega_P += _error_yaw * _kp_yaw;                   // Adding yaw correction to proportional correction vector.
    _omega_I += _error_yaw * _ki_yaw;                   // adding yaw correction to integrator correction vector.

    //          Here we will place a limit on the integrator so that the integrator cannot ever exceed half the saturation limit of the
yros
    integrator_magnitude = _omega_I.length();
    if (integrator_magnitude > radians(300)) {
        _omega_I *= (0.5f * radians(300) / integrator_magnitude);       // Why do we have this discontinuous?  EG,
hy the 0.5?
    }
    //Serial.print("*");
```

```
*************************************************/

DCM::euler_angles(void)

    #if (OUTPUTMODE == 2)                                   // Only accelerometer info (debugging purposes)
    roll                = atan2(_accel_vector.y, -_accel_vector.z);         // atan2(acc_y, acc_z)
    pitch               = asin((_accel_vector.x) / (double)9.81); // asin(acc_x)
    yaw                         = 0;
    #else
    pitch               = -asin(_dcm_matrix.c.x);
    roll                = atan2(_dcm_matrix.c.y, _dcm_matrix.c.z);
    yaw                 = atan2(_dcm_matrix.b.x, _dcm_matrix.a.x);
    #endif

    roll_sensor  = degrees(roll)  * 100;
    pitch_sensor        = degrees(pitch) * 100;
    yaw_sensor                  = degrees(yaw)   * 100;

    if (yaw_sensor < 0)
                yaw_sensor += 36000;


*************************************************/

t
_DCM::get_health(void)

    return _health;
```

Βιβλιογραφία

(8.1) Βιβλία και επιστημονικά άρθρα

1. Astronomical methods in aerial navigation by k.hiling beij burau of standards
2. Global Positioning Systems, Inertial Navigation, and Integration By: Grewal, Mohinder S.; Weill, Lawrence R.; Andrews, Angus P. © 2007 John Wiley & Sons

3. Aerial Navigation and Navigating Instruments by H. N.EATON Bureau of Standards

4. The unmanned aerial vehicles (UAV) market 2009–2019 (2009) Visiongain

5. Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development by Peter Lubbers, Brian Albers and Frank Salim Apress © 2010 (305 pages) Citation ISBN:9781430227908

6. Autonomous Flying Robots Unmanned Aerial Vehicles and Micro Aerial Vehicles Kenzo Nonami ,Farid Kendoul ,Satoshi Suzuki Wei Wang ,Daisuke Nakazawa

7. Dickerson L (2007) UAV on the rise. Aviat Week Space Technol, Aerospace Source Book 2007 166(3)

8. LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control 2nd McGraw-Hill School Education Group ©1997 0ISBN:007032915X

9. GPS theory, algorithms, and applications Xu, Guochang, 1953- ISBN :9783540727149 Springer

10. The Cosine-Haversine Formula C. C. Robusto ,The American Mathematical Monthly
11. Vol. 64, No. 1 (Jan., 1957), pp. 38-40Published by: Mathematical Association of America

12. The unmanned aerial vehicles (UAV) market 2009–2019 (2009) Visiongain

13. Celestial Navigation in the GPS Age by John Karl (Apr 1, 2007)

14. Realization of ActiveX event callback and data transfer in LabVIEW

15. Geological and geophysical modeling on virtual globes using KML, COLLADA, and Javascript Declan G. De Paor Steven J. Whitmeyer Department of Physics, Old Dominion University, 4600 Elkhorn Avenue, Norfolk, VA 23529, USA Department of Geology & Environmental Science, James Madison University, 395S. High Street, MSC 6903, Harrisonburg, VA 22807, USA

16. Simultaneous localization and mapping with the AR.Drone Nick Dijkshoorn February 15, 2012

17. The Navigation and Control technology inside the AR.Drone micro UAV Pierre-Jean Bristeau , François Callou ,David Vissière ,Nicolas Petit

18. Using Mobile Devices for Robotic Controllers: Examples and Some Initial Concepts for Experimentation by Rosemarie E. Yagoda and Susan G. Hill

19. AR.Drone Developer Guide 17, 2011 Revision SDK 1.7

20. A Novel Approach to Loxodrome (Rhumb Line), Orthodrome (Great Circle) and Geodesic Line in ECDIS and Navigation in General A. Weintrit & P. Kopacz Gdynia Maritime University, Gdynia, Poland

21. UDP Lite for Real Time Multimedia Applications Lars-Åke Larzon*, Mikael Degermark*†, Stephen Pink*† Extended Enterprise Laboratory HP Laboratories Bristol HPL-IRI-1999-001 April, 1999
22. GPS satellite signal tracking method for GPS receivers Hitoshi Ando

(8.2) Ιστότοποι

1. http://en.wikipedia.org/wiki/Haversine_formula

2. http://en.wikipedia.org/wiki/Great-circle_distance

3. Calculate distance, bearing and more between Latitude/Longitude points http://www.movable-type.co.uk/scripts/latlong.html

4. http://en.wikipedia.org/wiki/Orthodrome

5. http://en.wikipedia.org/wiki/Bearing_(navigation)

6. http://en.wikipedia.org/wiki/Haversine

7. http://en.wikipedia.org/wiki/Haversine_formula

8. http://en.wikipedia.org/wiki/Gps

9. http://en.wikipedia.org/wiki/GPS_(satellite)

10. http://www.wireshark.org/

11. UDP Communication in LabVIEW http://zone.ni.com/devzone/cda/tut/p/id/4950

12. http://diydrones.com/

13. http://en.wikipedia.org/wiki/Latitude

14. http://en.wikipedia.org/wiki/Longtitude

15. http://en.wikipedia.org/wiki/User_Datagram_Protocol

16. http://www.math.utah.edu/~beebe/software/ieee/

17. http://en.wikipedia.org/wiki/IEEE_754-2008

18. Global Variables
http://zone.ni.com/reference/en-XX/help/371361H-01/lvconcepts/glob_variables/

19. ActiveX http://zone.ni.com/devzone/cda/tut/p/id/2983

(8.3) Πήγες αλγορίθμων

1. http://code.google.com/p/arducopter/

2. http://code.google.com/p/ardupilot-mega/wiki/home

3. http://arduino.cc/en/Tutorial/HomePage