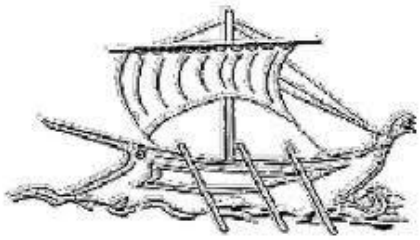
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ



Α.Τ.Ε.Ι. ΠΕΙΡΑΙΑ

ΜΕΛΕΤΗ ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΑΥΤΟΝΟΜΟΥ ΙΠΤΑΜΕΝΟΥ ΟΧΗΜΑΤΟΣ ΜΕ ΤΡΕΙΣ ΕΛΙΚΕΣ

Σπουδαστές: **ΖΟΥΤΗΣ ΔΗΜΗΤΡΙΟΣ, ΜΠΕΡΔΕΚΛΗΣ ΓΕΩΡΓΙΟΣ**

Επιβλέποντες Καθηγητές: **ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΙΟΣ,**

ΑΛΑΦΟΔΗΜΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

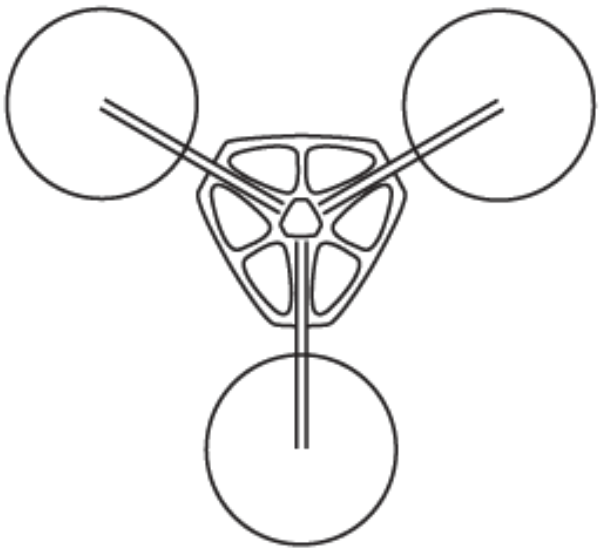
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΜΑΡΤΙΟΣ 2014

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ
Π. Ραλλη & Θηβών 250, 12244 Αιγάλαιω , Αθήνα – Ελλάδα
Τηλ. 210-5381488

Μελέτη, σχεδίαση και κατασκευή αυτόνομου ιπτάμενου οχήματος με τρεις έλικες



The greatest danger for most of us lies not in setting our aim too high and falling short; but in setting our aim too low, and achieving our mark.

Michelangelo

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τους γονείς μας για την συμπαράσταση τους καθ' όλη τη διάρκεια της εκπόνησης της εργασίας αυτής.

Τον καθηγητή μας Γρηγόρη Νικολάου για όλη τη βοήθεια και καθοδήγηση, αλλά και την εξύψωση πνεύματος στα αδιέξοδα μας.

Το φίλο μας Νικήτα Χρονά για τις πολύτιμες συμβουλές για την αποφυγή λαθών λόγω απειρίας και την πρόσβαση σε μηχανολογικό εξοπλισμό για την κατασκευή μερών του tricopter.

Επίσης, το Θάνο Ελευθεράκο για την παροχή συμβουλών και τη Μαίρη Τσετσέκου για την μεγάλη υποστήριξη και τη βοήθεια της στο σχεδιασμό.

Γιώργος, Δημήτρης

Περίληψη

Στο παρόν κείμενο παρουσιάζεται η πτυχιακή εργασία με τίτλο “Μελέτη, σχεδίαση και κατασκευή αυτόνομου ιπτάμενου οχήματος με τρεις έλικες”. Η εργασία πραγματεύεται τη συνολική διαδικασία της κατασκευής ενός tricopter εκ του μηδενός. Αναλύει την αρχική μελέτη, τα στάδια σχεδίασης και τη διαδικασία κατασκευής του.

Αρχικά, παρουσιάζονται εισαγωγικά στοιχεία: ιστορική αναδρομή στους προπάτορες των UAV, τα πεδία εφαρμογής μιας τέτοιας κατασκευής και η αρχή λειτουργίας ενός τέτοιου οχήματος.

Έπειτα, περιγράφεται ο τρόπος που προσεγγίζεται μαθηματικά ένα tricopter και αναλύονται ο ελεγκτής και οι αλγόριθμοι του. Περιγράφεται ο αυτόματος έλεγχος με τον ελεγκτή PID, καθώς και η πειραματική προσέγγιση αυτού και εξηγούνται οι αλγόριθμοι που χρησιμοποιούνται για τη μορφοποίηση των δεδομένων από τις εξόδους των αισθητηρίων.

Ακόμη, περιγράφεται η διαδικασία κατασκευής του tricopter και αναλύονται τα μέρη που το αποτελούν, καθώς και τα εργαλεία που χρησιμοποιήθηκαν για να έρθει σε πέρας η όλη διαδικασία. Επίσης, παρουσιάζεται και εξηγείται η λογική ροή του κώδικα που γράφτηκε για το tricopter.

Τέλος, εκτίθενται τα αποτελέσματα και τα συμπεράσματα που εξήχθησαν κατά την εκπόνηση της παρούσας εργασίας και παρουσιάζονται τρόποι μελλοντικής εξέλιξης και βελτίωσης.

Abstract

This text presents the thesis entitled "Design and construction of autonomous flying vehicle with three propellers". The paper discusses the overall process of building a tricopter from scratch. Analyzes the initial study, the design procedure and the construction process.

First, introductory elements are presented: history of the ancestors of UAVs, the fields of application of such a construction and the principles of operation of such a vehicle.

Then, the mathematical approach of a tricopter is described as is its control and algorithms. Also being described is the automation control with a PID controller, as well as its experimental approach and the algorithms used for the sensors' data manipulation are explained.

Moreover, the construction process of the tricopter is described and the component parts and tools used in the whole process and development are analyzed. Furthermore the flow chart and thought process of the code written is presented and explained.

Finally, the results and conclusions reached while working on this project are laid out and ways of future development and improvement are presented.

Περιεχόμενα

Κεφάλαιο 1 - Εισαγωγή

Εισαγωγή

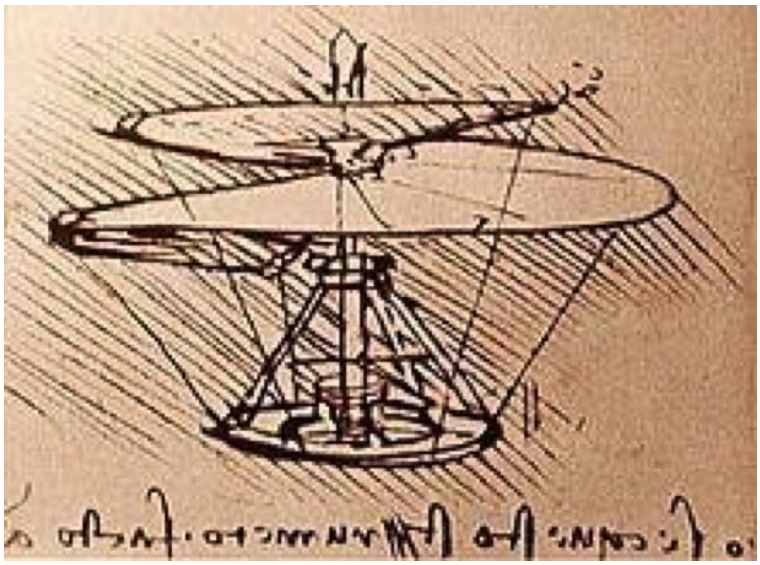
Ένα multirotor ή multicopter είναι ένα ελικοφόρο όχημα με περισσότερους από δύο ρότορες. Τα multirotors συχνά χρησιμοποιούν πτερύγια σταθερού βήματος, των οποίων ο ρότορας δεν μεταβάλλει την γωνία του καθώς τα πτερύγια περιστρέφονται.

Για τα multicopters με ρότορες ζυγού αριθμού, ο έλεγχος της κατεύθυνσης του οχήματος επιτυγχάνεται μεταβάλλοντας την σχετική ταχύτητα του κάθε ρότορα για να αλλάξει η ώθηση και η ροπή που παράγεται από τον καθένα ενώ για τα multicopters με μονού αριθμού ρότορες (πχ tricopters) χρειάζεται ένας τουλάχιστον από τους ρότορες να μπορεί να μεταβάλλει τη γωνία κλίσης του, για αλλαγή της κατεύθυνσης του οχήματος.

Συγκεκριμένα, ένα tricopter αποτελείται από τρεις ρότορες, εκ των οποίων ο ένας, συνήθως ο οπίσθιος δύναται να αλλάζει τη γωνία κλίσης του. Το σχήμα του μπορεί να είναι τύπου Υ ή και τύπου Τ, που όμως δε συνηθίζεται. Η αλλαγή κλίσης ενός εκ των μοτέρ, συνιστά το πιο δύσκολο σχεδιαστικό και μηχανολογικό στοιχείο της κατασκευής, καθώς είναι το μόνο κινητό κομμάτι της και απαιτεί μηχανολογική κατασκευή για να λειτουργεί ορθά.

Ιστορική αναδρομή

Οι πρώτες αναφορές στον όρο της κάθετης πτήσης προέρχονται από την Κίνα του 400 π. Χ.. Επρόκειτο όμως απλά για ένα παιδικό παιχνίδι που έφερε έλικα. Μόνο πολύ αργότερα, στις αρχές της δεκαετίας του 1480, ο Leonardo da Vinci δημιούργησε ένα σχέδιο μιας μηχανής που θα μπορούσε να περιγραφεί ως “εναέρια βίδα”. Το μόνο του ελάττωμα ήταν πως δεν προέβλεπε κάποιο μηχανισμό που να αντισταθμίζει τη ροπή στρέψης.



Εικόνα 1.1: Η «εναέρια βίδα» του Leonardo da Vinci

Από το 1754 έγιναν διάφορες προσπάθειες δημιουργίας οχημάτων κάθετης πτήσης, έως το 1861 όπου ο Γάλλος εφευρέτης Gustave de Ponton d'Amécourt χρησιμοποίησε για πρώτη φορά τη λέξη "ελικόπτερο" παρουσιάζοντας μια κατασκευή από αλουμίνιο τροφοδοτούμενη από μια ατμομηχανή. Η κατασκευή του γιορτάστηκε ως πρωτοποριακή χρήση του αλουμινίου, παρόλο που ποτέ δεν ανυψώθηκε.

Το 1878 κατασκευάζεται από τον Enrico Forlanini το πρώτο μη επανδρωμένο ελικόπτερο που απογειώθηκε. Έμεινε στον αέρα για είκοσι δευτερόλεπτα, σε ύψος δώδεκα μέτρων μετά την κάθετη απογείωσή του. Και το ελικόπτερο αυτό τροφοδοτούνταν από ατμομηχανή.

Το 1861, ο James Gordon Bennet, Jr. έδωσε στον Thomas Edison χίλια αμερικάνικα δολάρια για να πραγματοποιήσει πειράματα για την ανάπτυξη των πτήσεων. Ο Edison κατασκεύασε ένα ελικόπτερο με μηχανή εσωτερικής καύσης, το οποίο καταστράφηκε έπειτα από έκρηξη. Το Μάιο του 1905, ο Σλοβάκος ερευνητής Ján Bahýľ που υιοθέτησε τις έρευνες του Edison για τη χρήση μηχανών εσωτερικής καύσης στα ελικόπτερα, πραγματοποίησε πτήση με το δικό του σχέδιο που έφτασε σε ύψος τεσσάρων μέτρων και πέταξε για πάνω από χίλια πεντακόσια μέτρα.

Η πρώτη επανδρωμένη πτήση με το Gyroplane No.1

Η πρώτη επίσημα αναγνωρισμένη επανδρωμένη πτήση, καταγράφεται το Σεπτέμβριο του 1907, όταν το "Gyroplane No.1" σήκωσε τον πιλότο του εξήντα εκατοστά από το έδαφος για ένα λεπτό. Παρόλα αυτά, το ελικόπτερο ήταν εξαιρετικά ασταθές και χρειαζόταν έναν άνθρωπο σε κάθε του άκρη για να το κρατάνε σταθερό. Την ίδια χρονιά, ο Γάλλος εφευρέτης Paul Cornu σχεδίασε και κατασκεύασε ένα ελικόπτερο με δύο αντίστροφης περιστροφής έλικες που κινούνταν από έναν κινητήρα υγρών καυσίμων εικοσιτεσσάρων ίππων. Το Νοέμβριο του 1907, το ελικόπτερο αυτό υψώθηκε μαζί με τον Cornu 30 εκατοστά από το έδαφος για είκοσι δευτερόλεπτα. Παρόλο που η πτήση δεν ξεπερνά αυτή του "Gyroplane No.1", έχει καταγραφεί ως η πρώτη πραγματικά ελεύθερη πτήση με πιλότο. Πραγματοποιήθηκαν μερικές πτήσεις ακόμη με το ελικόπτερο αυτό που έφτασαν τα δύο μέτρα ύψος, αλλά αποδείχθηκε πως ήταν ασταθές και εγκαταλείφθηκε.

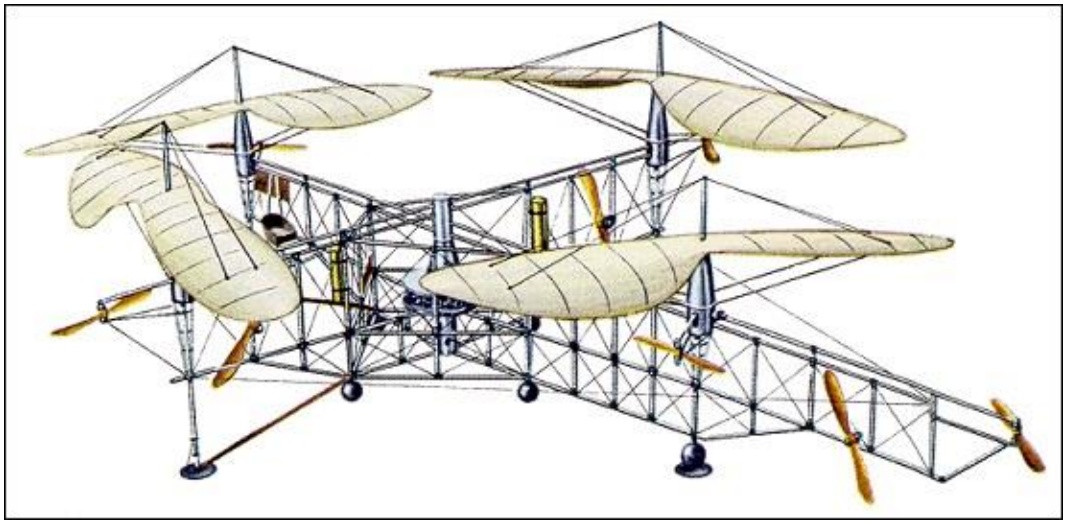
Εικόνα 1.2: Το "Gyroplane No.1" του Paul Cornu

Η μηχανή του Etienne Oemichen

Στη Γαλλία, ο Etienne Oemichen, ένας νεαρός μηχανικός στην Peugeot, ξεκίνησε πειράματα με περιστρεφόμενες πτέρυγες (έλικες) το 1920, κατασκευάζοντας ένα σύνολο από έξι διαφορετικές μηχανές. Η δεύτερη μηχανή του πέταξε χωρίς βοήθεια στις 11 Νοεμβρίου 1922. Το Oemichen No.2 είχε ένα σωληνοειδές πλαίσιο σχήματος "X" , με ένα ευρύ ρότορα δύο πτερυγίων στο άκρο του κάθε βραχίονα.

Για τον έλεγχο και την πλευρική κίνηση, χρησιμοποιήθηκαν οκτώ μικρές έλικες: πέντε οριζόντιες έλικες μεταβλητού και αναστρέψιμου βήματος για πλευρική σταθερότητα, μία άλλη έλικα στη μύτη για την αλλαγή κατεύθυνσης, και ένα άλλο ζεύγος από ωθητές για την προς τα εμπρός κίνηση. Μέχρι το 1923, το Oemichen No.2 ήταν σε θέση να παραμένει στον αέρα για αρκετά λεπτά και στις 14 Απριλίου του 1924, κατάφερε το πρώτο ρεκόρ απόστασης

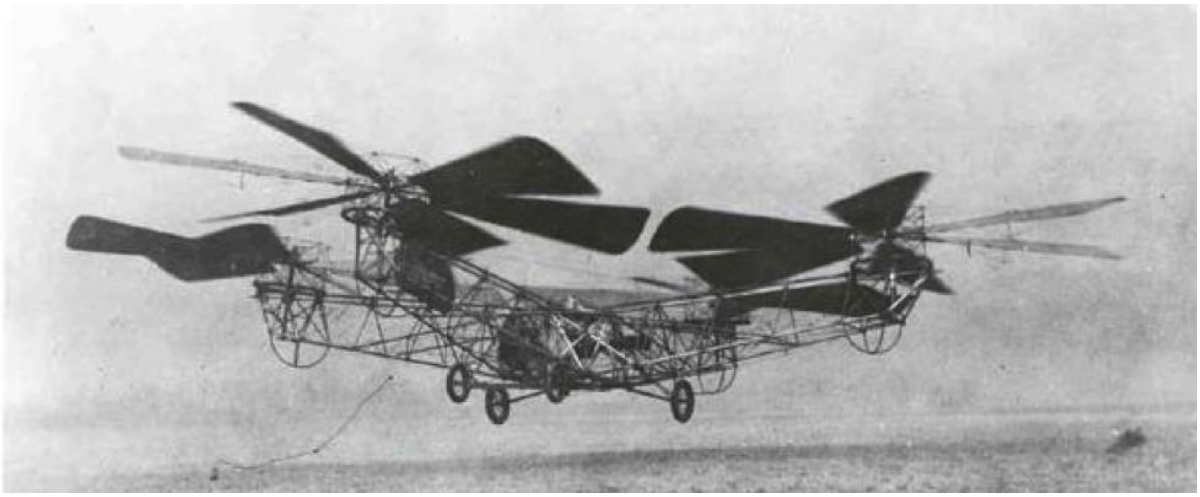
περιστροφικής πτέρυγας, τα 360 μέτρα. Στις 4 Μαΐου, ολοκλήρωσε την πρώτη πτήση ενός χιλιομέτρου για όχημα με περιστροφικές πτέρυγες σε 7 λεπτά 40 δευτερόλεπτα για να κερδίσει ένα βραβείο 90.000 φράγκων. Ο μέγιστος χρόνος πτήσης ήταν 14 λεπτά. Παρά το γεγονός ότι ήταν σε θέση να επιδείξει επαρκή ελεγξιμότητα και δύναμη σε αυτή την ιστορική πτήση, δεν ήταν μια πρακτική ιπτάμενη μηχανή. Αναγνωρίζοντας την έλλειψη πρακτικότητας της μηχανής, ο Oemichen επικεντρώθηκε σε μια σειρά από αεροσκάφη με ένα μόνο κύριο στροφείο και δύο ρότορες κατά της ροπής στρέψης, αλλά είχε μικρή επιτυχία.



Εικόνα 1.3: Το Oemichen No.2

Μια σημαντική συμβολή στη λειτουργία του ελικοπτέρου ήρθε να δώσει ο Αργεντινός Raúl Pateras-Pescara de Castelluccio που το 1920, ενώ εργαζόταν στην Ευρώπη, παρουσίασε την πρώτη επιτυχημένη εφαρμογή των πτερυγίων με ρυθμιζόμενη κλήση, με δύο ομοαξονικούς έλικες, αντίστροφα περιστρεφόμενους, που μπορούσαν να ρυθμίζουν την ροή αέρα σύμφωνα με το επιθυμητό. Ακόμη, οι έλικες του είχαν τη δυνατότητα να γείρουν μερικές μοίρες προς τα εμπρός, επιτρέποντας στο όχημα να κινηθεί προς αυτή την κατεύθυνση, χωρίς χρήση ξεχωριστού έλικα για το σκοπό αυτό. Ως τις αρχές του 1924, το ελικόπτερο No.1 του Pescara δοκιμάστηκε και κρίθηκε ως πολύ χαμηλής ισχύος για το βάρος του. Η Βρετανική κυβέρνηση χρηματοδότησε την περαιτέρω έρευνα του Pescara, το οποίο αποτέλεσε στο ελικόπτερο No.3, εφοδιασμένο με μια μηχανή απόδοσης 250 ίππων που μπορούσε να ίπταται για δέκα λεπτά.

Την ίδια εποχή στις ΗΠΑ, ο George de Bothezat κατασκεύασε ένα quadcopter, το "De Bothezat helicopter" για τη στρατιωτική αεροπορία των Ηνωμένων Πολιτειών, αλλά ο ίδιος ο στρατός ακύρωσε το πρόγραμμα το 1924, με αποτέλεσμα τη διάλυση του αεροσκάφους.



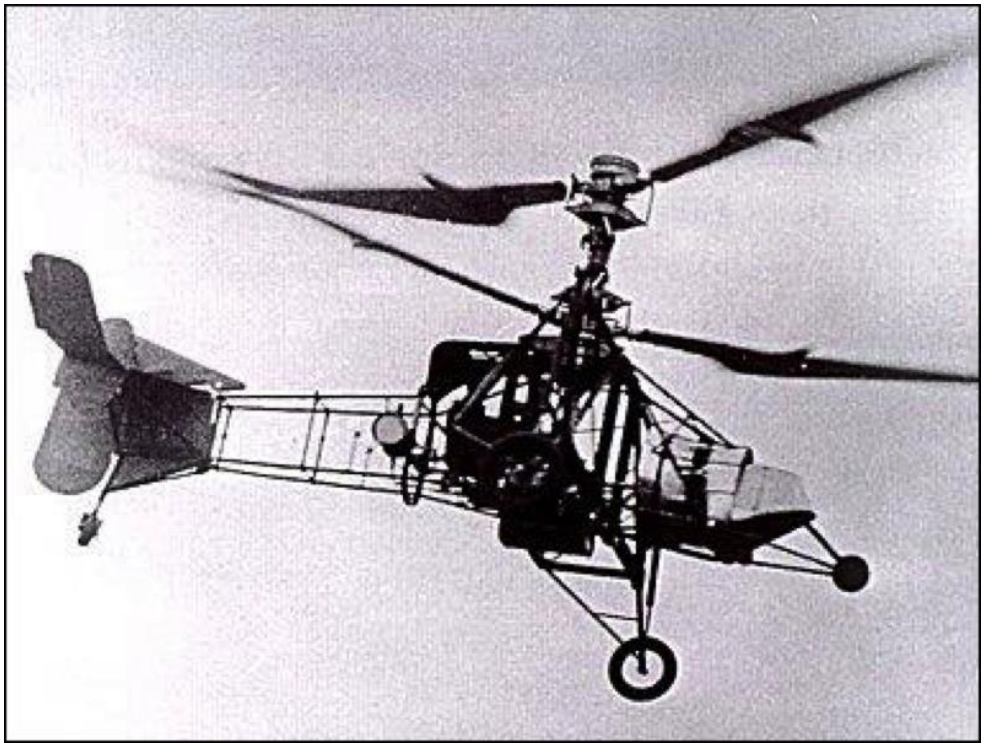
Εικόνα 1.4: Το "De Bothezat helicopter"

Το Αυτόγυρο: Πλησιάζοντας τη μορφή του σημερινού ελικοπτέρου.

Το 1923, ο Βρετανο-Ισπανός μηχανικός Juan de la Cierva, μετά από πολλές αποτυχημένες προσπάθειες τα προηγούμενα έτη, δημιούργησε το Αυτόγυρο, μια μηχανή που παρ' όλη τη χρήση τεσσάρων μεγάλων πτερυγίων για την ανύψωσή του, ο κατευθυντικός του έλεγχος ήταν όμοιος με αυτόν των αεροπλάνων.

Η παρουσίαση του μοντέλου αυτού στο Βρετανικό υπουργείο αεροπορίας στέφθηκε με επιτυχία, κάτι που ακολουθήθηκε από τη δημιουργία εταιρίας κατασκευής Αυτόγυρων από τον Juan de la Cierva και τον Σκοτσέζο βιομήχανο James Weir. Μερικά χρόνια αργότερα, το 1928, το μοντέλο C.8L4 της εταιρίας πραγματοποίησε πτήση από το Λονδίνο ως το Παρίσι και έπειτα συνέχισε την πορεία του πετώντας στο Βερολίνο, τις Βρυξέλλες και το Άμστερνταμ. Το C.8L4 ήταν το πρώτο αεροσκάφος περιστρεφόμενου έλικα που διέσχισε τη θάλασσα της Μάγχης.

Κατά τη διάρκεια του 1930 - 1936, οι διάσημοι μηχανικοί Louis Breguet και Rene Dorand, κατασκεύασαν ένα ελικόπτερο που έμοιαζε πολύ με τα σημερινά. Σε ένα στροφείο υπήρχαν 2 ζευγάρια λεπίδων και στην ουρά επίσης ένας ακόμα έλικας. Οι λεπίδες είχαν βήμα κυκλικό, ενώ το σχήμα τους ήταν πιο φαρδύ κοντά στο στροφείο και λεπτότερο προς την άκρη του.



Εικόνα 1.5: Το «Αυτόγυρο»

Στην ουρά του χρησιμοποιήθηκαν οριζόντια και κάθετα συστήματα ώστε να δοθεί μεγαλύτερη σταθερότητα στην κατασκευή. Για πρώτη φορά στην ιστορία, καταγράφηκαν πολλές πτήσεις, με πολλές επιτυχίες και με πρώτη από όλες την πτήση που διήρκεσε 62 λεπτά και διένυσε απόσταση 44 χιλιομέτρων. Η εξέλιξη του ελικοπτέρου όμως, δυστυχώς διακόπηκε από τον Δεύτερο Παγκόσμιο Πόλεμο.

Στις αρχές της δεκαετίας του 1940 στην Αμερική, ένας Ρώσος μηχανικός, ο Igor Sikorsky ανταγωνιζόταν τον W. Lawrence LaPage για την κατασκευή του πρώτου αμερικανικού στρατιωτικού ελικοπτέρου. Ο πρώτος κατασκεύασε ένα ελικόπτερο με έναν έλικα, το οποίο και θα ήταν ένα από τα πρώτα πρακτικά ελικόπτερα που χρησιμοποιούσαν έναν μόνο έλικα για την ανύψωσή τους. Έπειτα από πειραματισμούς στο σχεδιασμό και την κατασκευή, ο Sikorsky κατέληξε στην προσθήκη ενός μικρότερου έλικα στην ουρά του ελικοπτέρου για να αντισταθμίσει τη στροφορμή που δημιουργούσε ο κυρίως έλικας. Το ονόμασε VS-300. Με βάση αυτό, κατασκεύασε το R-4, το οποίο και ήταν το πρώτο μεγάλης κλίμακας ελικόπτερο που μπήκε σε μαζική παραγωγή. Ήταν το μόνο ελικόπτερο που χρησιμοποιήθηκε στην πλευρά των συμμάχων, κυρίως ως όχημα διάσωσης σε δύσβατες περιοχές.

Εικόνα 1.6: Το "R-4" του Igor Sikorsky

Ενώ οι LePage και Sikorsky κατασκεύαζαν ελικόπτερα για το στρατό, Η Bell Aircraft προσέλαβε τον Arthur Young για να βοηθήσει να κατασκευαστεί ένα ελικόπτερο χρησιμοποιώντας τον έλικα διπλής λεπίδας σχεδιασμένο από τον ίδιο, η οποία χρησιμοποιούσε μια σταθμισμένη μπάρα που τοποθετήθηκε σε γωνία 90 μοιρών σε σχέση με τον κυρίως έλικα. Το επόμενο ελικόπτερο που ονομάστηκε "model 30", έδειξε την απλότητα και την ευκολία χρήσης του σχεδιασμού. Το "model 30" αναπτύχθηκε στο "Bell 47", που έγινε το πρώτο ελικόπτερο με πιστοποίηση για πολιτική χρήση στις Ηνωμένες Πολιτείες. Το Bell 47 παράχθηκε σε αρκετές χώρες, και ήταν το πιο δημοφιλές μοντέλο ελικοπτέρου για σχεδόν 30 χρόνια.

Κατά τη διάρκεια της δεκαετίας του 1950, πολλά από τα ελικόπτερα άρχισαν να γίνονται γρηγορότερα και πιο ασφαλή. Ήταν κατασκευασμένα έτσι ώστε να έχουν ευκολότερο χειρισμό, να μεταφέρουν πάνω από 2 επιβάτες και να έχουν καλύτερη και σταθερότερη πτήση. Αυτή η χρονική στιγμή, είναι γνωστή για την μαζική παραγωγή ελικοπτέρων στην Αμερική αλλά και στην Ευρώπη.

Αξιοσημείωτο μοντέλο ελικοπτέρου είναι το Sikorsky S-70, γνωστό στο κοινό ως Blackhawk, με την παραγωγή του να ξεπερνά κάθε προηγούμενο και τον κινητήρα του να κατασκευάζεται ως και τον 21^{ο} αιώνα. Μετά από το S-70 Ακολούθησε το S-76, με παρόμοια κατασκευή, αλλά χαμηλότερου βάρους, με την κύρια χρήση του να είναι μεταφορικό μέσο για τραυματίες αλλά και άλλες δραστηριότητες, καθώς ήταν ικανό να εκτελεί γρήγορες και ακριβείς πτήσεις.

Εικόνα 1.7: Το “Sikorsky S-70” ή “Blackhawk”

Δύο ακόμη αξιοσημείωτα μοντέλα είναι της σειράς CH της εταιρίας Piasecki Corporation, που κυρίως κατασκεύαζε στρατιωτικά ελικόπτερα. Και πάλι στο κοινό έγιναν γνωστά με διαφορετικό όνομα, το “Chinook”. Το ενδιαφέρον επικεντρώνεται στο γεγονός πως για τη λειτουργία του χρησιμοποιεί δύο κυρίως έλικες, αντίστροφης περιστροφής, χωρίς να απαιτεί ουραίο έλικα αντιστάθμισης στροφορμής. Η χρήση του είναι κυρίως για μεταφορά προσωπικού και υλικού, καθώς χάρη στους δύο έλικες του, δύναται να ανυψώσει πολύ μεγάλο βάρος. Δευτερεύουσα χρήση του είναι οι επιβιβάσεις και αποβιβάσεις προσωπικού σε σημεία οπού είναι αδύνατη η προσγείωση, χάρη στην μεγάλη ευστάθειά του.

Εικόνα 1.8: Το "Chinook"

Στα επόμενα χρόνια η εξέλιξη των ελικοπτέρων ήταν ραγδαία, με την παράλληλη ανάπτυξη των ηλεκτρονικών και της μηχανολογίας να ωθούν τη δημιουργία ελικοπτέρων σε ανώτερα επίπεδα.

Το W.11 Air Horse της Cierva.

Τέλος, ένα ελικόπτερο του οποίου η κατασκευή έχει μεγάλο ενδιαφέρον είναι το W.11 Air Horse, καθώς είχε μορφή tricopter. Δημιουργήθηκε στα τέλη της δεκαετίας του 1940 και είναι το κοντινότερο σε σχεδιασμό μοντέλο ελικοπτέρου κανονικής κλίμακας στη δική μας κατασκευή.

Το W.11 Air Horse ήταν το μεγαλύτερο ελικόπτερο στον κόσμο, όταν πέταξε για πρώτη φορά στις 7 Δεκεμβρίου 1948. Ένας κινητήρας "Rolls-Royce Merlin 24" των 1642hp είχε τοποθετηθεί στην άτρακτο για να δώσει κίνηση σε τρεις μεγάλους ρότορες τριών λεπίδων τοποθετημένους σε ζυγοστάτες που προεξείχαν από την τετραγωνική άτρακτο. Δυο κάθετα πτερύγια ήταν τοποθετημένα στην πτέρυγα της ουράς στο πίσω μέρος της ατράκτου και υπήρχε χώρος για ένα πλήρωμα τριών ατόμων.

Εικόνα 1.9: Το "W.11 Ait Horse"

Ως επιβατικό αεροσκάφος το W.11 θα μπορούσε να σηκώσει 24 άτομα, αλλά είχαν προβλεφθεί και άλλες λειτουργίες που περιλάμβαναν: ασθενοφόρο αέρος, εναέριο γερανό και ψεκαστήρα καλλιεργειών. Τον Σεπτέμβριο του 1945 η Pest Control Ltd είχε συζητήσει την τελευταία δυνατότητα με την Cierva και ο σχεδιασμός του W.11 είχε τροποποιηθεί για να ανταποκριθεί σε αυτό το ρόλο. Ως εκ τούτου, η Cierva έλαβε σύμβαση ανάπτυξης για ένα W.11 τον Ιούλιο του 1946, που συντάχθηκε με την Προδιαγραφή E.19/46, και ένα δεύτερο παραγγέλθηκε στις αρχές του 1947. Η Cunliffe-Owen Aircraft Ltd υπέγραψε συμβάσεις για την κατασκευή των δύο αεροσκαφών στο αεροδρόμιο Southampton / Eastleigh υπό τον τεχνικό και οικονομικό έλεγχο της εταιρείας Cierva.

Με ωφέλιμο φορτίο 3048kg εντομοκτόνου το W.11 θα είχε κάνει ένα εντυπωσιακό ψεκαστήρα, και μετά την πρώτη πτήση τον Δεκέμβριο του 1948 οι επόμενες δοκιμές ήταν πολλά υποσχόμενες. Το Υπουργείο Αποικιών έκανε μια επιχορήγηση από 45.000 λίρες Αγγλίας για να βοηθήσει με το κόστος ανάπτυξης αφού ο τύπος του αεροσκάφους προσέφερε πολλές προοπτικές για χρήση στο ρόλο ψεκασμού (το συνολικό κόστος του Υπουργείου Εφοδιασμού για την ανάπτυξη του σκάφους υπολογίζονταν σε 350.000 λίρες Αγγλίας), αλλά πριν το δεύτερο W.11 πετάξει, το πρώτο συνετρίβη στις 13 Ιουνίου 1950, σκοτώνοντας τα τρία μέλη του πληρώματος της δοκιμαστικής πτήσης. Το δεύτερο W.11 ποτέ δεν πέταξε και οδηγήθηκε προς διάλυση το 1960. Η ονομασία W.11T διατέθηκε σε ένα σχέδιο για ένα πιο μεγάλο W.11 με δύο κινητήρες "Rolls-Royce Merlin 502" των 1642hp και το W.12 είχε προβλεφθεί για ανάπτυξη σε φορτηγό αεροσκάφος με κινητήρες Rolls-Royce Dart turbopropps. Κανένα από αυτά δεν υλοποιήθηκε αφού εκείνη τη χρονική περίοδο η Cierva, χωρίς να έχει χρησιμοποιήσει τον αριθμό W.13, συμμετείχε στην ανάπτυξη του W.14 που μετονομάστηκε σε "Skeeter" και κατασκευάστηκε από την Saunders-Roe Ltd.

Το Volocopter

Φτάνοντας στη σημερινή εποχή, το πρώτο επανδρωμένο multirotor όχημα κάνει την εμφάνισή του, κατασκευασμένο από μια ομάδα νέων μηχανικών γερμανικής καταγωγής. Το πρωτότυπο όχημα είναι ικανό να πραγματοποιήσει πτήση με ένα άτομο επάνω του. Ακούει στο όνομα volocopter VC1, διαθέτει 16 μοτέρ, έχει διαστάσεις 5 επί 5 μέτρα και το βάρος του ανέρχεται στα 80 κιλά. Μετά την επιτυχή μικρή του αιώρηση, οι κατασκευαστές του χρηματοδοτήθηκαν από το γερμανικό υπουργείο οικονομικών και τεχνολογίας με το ποσό των δύο εκατομμυρίων ευρώ για την ανάπτυξη του επόμενου μοντέλου, volocopter VC200. Το νέο όχημα έχει θέσεις για δύο άτομα, διαθέτει 18 μοτέρ, μπορεί να αναπτύξει ταχύτητα πτήσης πάνω από 100km/h και διάρκεια πτήσης πάνω από μία ώρα. Και τα δύο μοντέλα τροφοδοτούνται από ρεύμα, είναι συνεπώς φιλικά προς το περιβάλλον. Το νεότερο μοντέλο δε, έχει ήδη αναγνωριστεί με πρωτεία και θεωρείται από πολλούς ένας πολύ πιθανός τρόπος μετακίνησης στο μέλλον.

Εικόνα 1.10: Το Volocopter VC200

Πεδία εφαρμογής

Λόγω της ευκολίας τους τόσο την κατασκευή και τον έλεγχο, τα αεροσκάφη multirotor χρησιμοποιούνται συχνά στον μοντελισμό και τα project για τηλεκατευθυνόμενα αεροσκάφη, στα οποία τα ονόματα tricopter, quadcopter, hexacopter και octocopter συχνά χρησιμοποιούνται για να δηλώσουν ελικόπτερα με 3, 4, 6 και 8 ρότορες, αντίστοιχα.

Τα τηλεκατευθυνόμενα ή και αυτόνομα αεροσκάφη multirotor χρησιμοποιούνται όλο και περισσότερο τα τελευταία χρόνια. Πλεονεκτούν ως προς το χαμηλό κόστος κατασκευής τους, το μικρό μέγεθος τους, καθώς και τη μεγάλη ευστάθεια που μπορεί να επιτευχθεί, αρκετή για σταθερές λήψεις φωτογραφιών και βίντεο. Έτσι, επιλέγονται όλο και συχνότερα για τη δημιουργία αεροφωτογραφιών και βίντεο σε διάφορους χώρους και κτίρια.

Άλλες χρήσεις τους είναι ο τακτικός έλεγχος ζωνών και χώρων μεγάλης επικινδυνότητας, όπως για πυρασφάλεια σε δασικές εκτάσεις, καταμέτρηση της πυκνότητας βλαβερών για τον άνθρωπο ουσιών σε κάποιο χώρο. Επίσης αστυνομικές και άλλες αρχές χρησιμοποιούν multirotors για τη χαρτογράφηση και την τρισδιάστατη απεικόνιση τόπων εγκλήματος και άλλων χώρων ενδιαφέροντος.

Αεροφωτογράφιση:

Εικόνα 1.11: Tricopter Με ενσωματωμένη Camera

Τοποθετώντας στο multirotor μια φωτογραφική μηχανή υψηλής ανάλυσης, μπορούμε να καταφέρουμε λήψη αεροφωτογραφιών με τον ίδιο τρόπο που ένα αεροπλάνο ή ελικόπτερο μπορεί να καταφέρει, αλλά με σημαντικά μικρότερο κόστος αλλά και ρύπανση του περιβάλλοντος. Οι σημερινές φωτογραφικές μηχανές είναι ιδανικές γι' αυτό το σκοπό, καθώς το μικρό τους βάρος διευκολύνει την ανύψωση και πτήση του multirotor.

Επιτήρηση χώρων και τοπογραφικές εφαρμογές

Με τα Multicopters η εναέρια επιτήρηση ανοικτών, κυρίως, χώρων καθίσταται αρκετά εύκολη. Με περιοδική καταγραφή ή επιτήρηση πραγματικού χρόνου, για ασφάλεια της περιοχής, πυρασφάλεια κ.ο.κ.. ακόμα και δυνατότητα τρισδιάστατης αναπαράστασης του χώρου, ανάλογα με το βάθος υποστήριξης του λογισμικού, η χρήση των multicopters μπορεί σε λίγα χρόνια να γίνει καθημερινή υπόθεση.

Η τηλεπισκόπηση και η φωτογραμμετρία ήδη χρησιμοποιούνται από δημόσιες υπηρεσίες, όπως για παράδειγμα το κτηματολόγιο.

Εικόνα 1.12: Εναέρια επιτήρηση χώρου από tricopter

Έρευνα σε πανεπιστήμια

Πολλά είναι τα πανεπιστήμια που ασχολούνται με drones καθώς είναι ιδανικά ως αντικείμενα έρευνας για μηχανικούς. Ο συνολικός έλεγχός του γίνεται με ηλεκτρονικά μέσα και στηρίζεται σε μαθηματικά πρότυπα και μοντέλα, οπότε η υλοποίηση ενός τέτοιου έργου αποτελεί μια πολύ καλή εφαρμογή τέτοιου είδους γνώσεων.

Η πρόοδος που υπάρχει σε αυτόν τον τομέα τα τελευταία χρόνια είναι ραγδαία, καθώς υπάρχει μεγάλο ενδιαφέρον με αποτέλεσμα την εκθετική βελτίωση της ποιότητας και των δυνατοτήτων των κατασκευών. Έχουν ήδη δημιουργηθεί αλγόριθμοι αναγνώρισης και αποφυγής σταθερών ή κινούμενων εμποδίων, αυτόνομη απογείωση και προσγείωση καθώς και συνεργασία μεταξύ πολλών multirotors για τη μεταφορά αντικειμένων ή ακόμη και ένα παιχνίδι "τένις" μεταξύ τους ή τη στήριξη μιας κάθετης ράβδου.

Εικόνα 1.13: Παρουσίαση δυνατοτήτων quadcopter του Τεχνολογικού Ινστιτούτου Ζυρίχης

Μεταφορά αντικειμένων

Παρόλο που είναι κάτι μη πραγματοποιημένο ακόμη, παρουσιάζει μεγάλο ενδιαφέρον για το μέλλον. Η αμερικανική εταιρία Amazon έχει ανακοινώσει πως εντός του έτους 2015 θα αποστέλλει δέματα μικρού βάρους σε κοντινούς στις αποθήκες της προορισμούς εντός μισής ώρας από την πραγματοποίηση της παραγγελίας, με τη χρήση οκτακόπτερων UAV.

Εικόνα 1.14: Το Octacopter της Amazon

Κάτι σίγουρα πολύ ενδιαφέρον για να παρακολουθήσει κανείς την εξέλιξή του στο μέλλον.

Μοντελισμός ως χόμπι

Τα τελευταία χρόνια, με τη μείωση του κόστους κατασκευής ή αγοράς ενός multicopter, έχει διαδοθεί πολύ η ενασχόληση με αυτό από μοντελιστές ή φίλους της καινοτόμου τεχνολογίας.

Αρχή Λειτουργίας

Τα κύρια μέρη του tricopter

Το tricopter αποτελείται από τα εξής κύρια μέρη:

- Το σκελετό, σε σχήμα Υ, όπου επάνω του στηρίζονται όλα τα υπόλοιπα εξαρτήματα
- Τρεις κινητήρες, έναν σε κάθε άκρο του σκελετού, συνοδευόμενοι από τους αντίστοιχους έλικες.
- Ένα servo για τη μεταβολή της κλίσης του οπίσθιου κινητήρα
- Δύο μικροελεγκτές AVR σε ξεχωριστές πλατφόρμες arduino
- Ένα IMU - Inertial Measurement Unit, με επιταχυνσιόμετρο και γυροσκόπιο τριών αξόνων αντίστοιχα
- Μια πλακέτα και κεραία ασύρματης επικοινωνίας
- Μια μπαταρία

Χαρακτηριστικά και προδιαγραφές

Τα χαρακτηριστικά του tricopter εξαρτώνται από τις απαιτήσεις του χρήστη. Ένας μικροελεγκτής των 8 bit επαρκεί για την απλή λειτουργία του, έχουν αναπτυχθεί όμως και συστήματα στα οποία η επεξεργασία των δεδομένων, ο έλεγχος και όλοι οι απαραίτητοι υπολογισμοί πραγματοποιούνται σε επίγειους υπολογιστές μεγάλων ταχυτήτων. Ανεξαρτήτως αυτών όμως υπάρχουν προδιαγραφές που πρέπει να τηρούνται από τη σχεδίαση και την κατασκευή ενός tricopter για να μπορεί να ίπταται ικανοποιητικά, ειδικά στις περιπτώσεις όπου πρόκειται για πτήσεις σε εσωτερικούς χώρους.

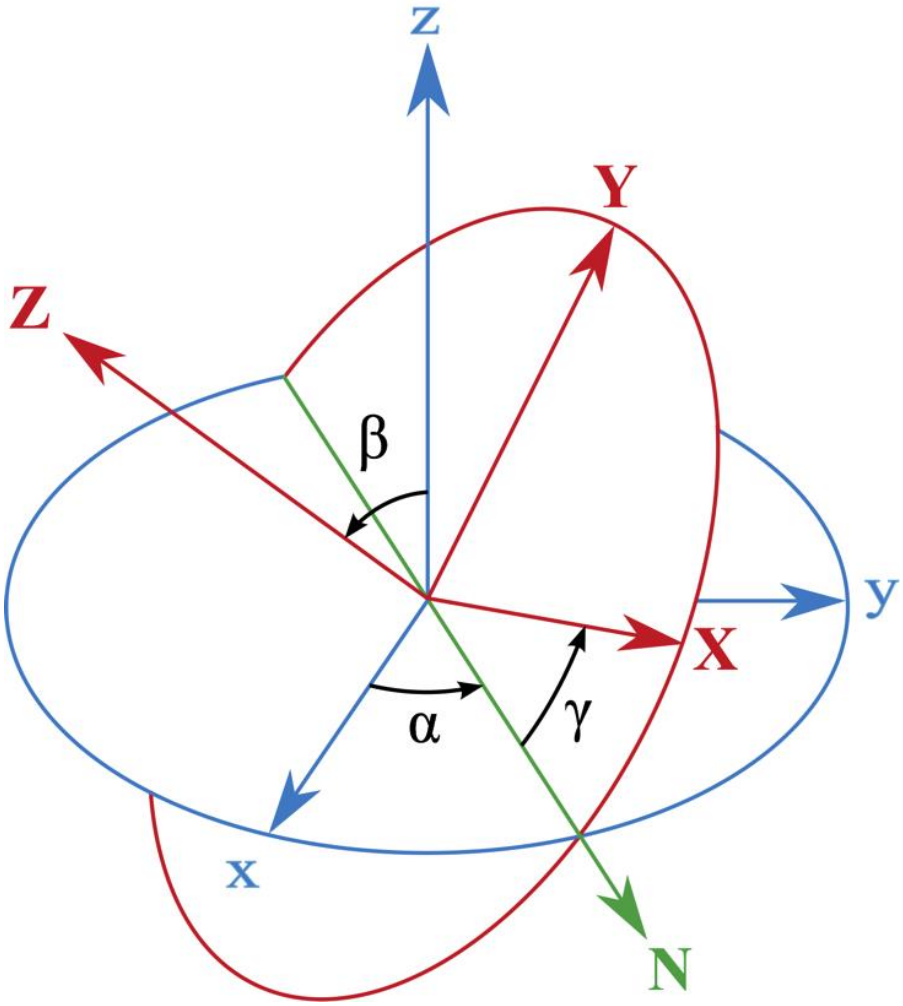
Οι προδιαγραφές αυτές είναι:

- Το μικρό του βάρος. Αν και είναι κάτι που επηρεάζεται από πολλούς παράγοντες, το βάρος του tricopter πρέπει να διατηρηθεί σε όσο το δυνατό χαμηλότερα επίπεδα, ώστε να έχει ομαλότερες και μεγαλύτερης διάρκειας πτήσεις. Το σύνηθες βάρος ενός tricopter είναι περίπου ένα κιλό. Η επίτευξη αυτού του στόχου, γίνεται κατασκευάζοντας το σκελετό του με τα ελαφρύτερα και όσο το δυνατόν λιγότερα υλικά, χωρίς να θέτουμε σε κίνδυνο όμως την σταθερότητά του σαν σύνολο.

- Η διάρκεια πτήσης του. Συνήθως δεν ξεπερνά τα είκοσι λεπτά, με το χρόνο αυτό να ποικίλει ανάλογα με το βάρος, τη συνολική χωρητικότητα της μπαταρίας και το είδος της πτήσης, που μπορεί να καταναλώσει τη μπαταρία πιο γρήγορα.
- Η ακτίνα πτήσης σε σχέση με το σταθμό βάσης. Αναλόγως λοιπόν το είδος και την ισχύ του πομπού και του δέκτη που θα επιλεχθεί, η ακτίνα πτήσης του tricopter μπορεί να μεταβληθεί δραματικά.
- Ο βαθμός αυτονομίας. Αυτός εξαρτάται από την ανάγκη που καλύπτει το tricopter στην κάθε εφαρμογή, αλλά και το διαθέσιμο χρηματικό ποσό για την κατασκευή. Μεγαλύτερος βαθμός αυτονομίας, συνεπάγεται σε περισσότερα, μεγαλύτερης ακρίβειας και ακριβότερα αισθητήρια, καθώς πιθανά και σε ακριβότερο σταθμό βάσης. Αντίθετα, αν ένας μικρός βαθμός αυτονομίας καλύπτει τις ανάγκες της εφαρμογής, μια απλή τηλεκατεύθυνση και ένα αρκετά ακριβές IMU, έχουν σημαντικά μικρότερο κόστος.

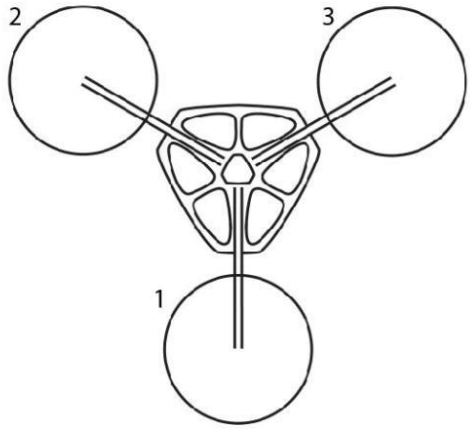
Βαθμοί ελευθερίας και τρόπος κίνησης

Για την περιγραφή των περιστροφικών κινήσεων του tricopter στηριζόμαστε στις γωνίες Euler. Για τις γωνίες Euler χρησιμοποιούμε τα γράμματα α, β και γ, που αντιστοιχούν στα yaw, roll και pitch. Η γωνία β (roll) αντιστοιχεί στον άξονα περιστροφής που επηρεάζουν τα M2 και M3. Η γωνία γ (pitch) στον άξονα που επηρεάζουν τα M2 και M3 συνδυαστικά και το M1. Τέλος, η γωνία α (yaw) αντιστοιχεί στον άξονα που επηρεάζει η ισχύς και των τριών μοτέρ καθώς και η κλίση του M1. Το σύστημα έχει σταθερό κέντρο το κέντρο του tricopter με μονάδα μέτρησης τις μοίρες.



Εικόνα 1.15: Σχεδιάγραμμα των γωνιών Euler

Έτσι, το κάθε μοτέρ επηρεάζει τη μεταβολή όλων των αξόνων, κάτι που κάνει το σύστημα αρκετά περίπλοκο. Συνολικά, το σύστημα μας έχει έξι βαθμούς ελευθερίας, επηρεαζόμενους μεταξύ τους σε πολλές περιπτώσεις. Για παράδειγμα, η μεταβολή της γωνίας *β* του άξονα X εξαρτάται από τη διαφορά ισχύος μεταξύ των *M2* και *M3*. Η μεταβολή αυτή όμως έχει ως αποτέλεσμα και τη μεταβολή της στροφορμής που εφαρμόζουν επάνω στο tricopter. Συνεπώς, μεταβολή της γωνίας *α* του άξονα Z. Αντιστοίχως συμβαίνει και με τη γωνία *γ* του άξονα Y. Τέλος, η γωνία α επηρεάζεται από την ισχύ και των τριών μοτέρ λόγω της στροφορμής τους. Η σταθεροποίηση του άξονα αυτού επιτυγχάνεται με τη δυνατότητα του *M1* να μεταβάλει την κλίση του. Το ύψος του tricopter ρυθμίζεται από την ισχύ των τριών μοτέρ.



Εικόνα 1.16: Η διάταξη των μοτέρ του tricopter

Ο έλεγχος του tricopter πραγματοποιείται μέσω τεσσάρων μεταβλητών:

- Throttle: η ισχύς των μοτέρ που μας επιτρέπει τον έλεγχο του ύψους του tricopter.
- Pitch: κατάλληλη κλίση του tricopter κατά τον άξονα Y, με αποτέλεσμα την κίνηση του προς τα εμπρός ή πίσω.
- Roll: κατάλληλη κλίση του tricopter κατά τον άξονα X, με αποτέλεσμα την κίνηση του προς τα δεξιά ή τα αριστερά.
- Yaw: μεταβολή της κλίσης του οπίσθιου μοτέρ, με αποτέλεσμα την περιστροφή του tricopter.

Φαίνεται έτσι πως πρέπει με τέσσερις μεταβλητές να ελέγξουμε έξι βαθμούς ελευθερίας. Αυτό κάνει το σύστημα μας under actuated. Η αλλαγή της ισχύος ενός μοτέρ μεταβάλλει περισσότερους από έναν βαθμό ελευθερίας. Πρόκειται λοιπόν για ένα πολύ δυναμικό σύστημα με πολύ μικρές, αντιτιθέμενες στην κίνηση του, δυνάμεις.

Οι δυνάμεις που αντιτίθενται στην κίνηση του tricopter προέρχονται από τη βαρύτητα, την αδράνεια και την αντίσταση του αέρα. Η βαρύτητα αντιτίθεται στην ανύψωση του tricopter

και είναι η μεγαλύτερη σε ισχύ από τις τρεις, με αποτέλεσμα να αποτελεί την κύρια κατανάλωση ενέργειας για την επιτυχή πτήση του tricopter. Η αδράνεια δρα σε όλες τις κατευθύνσεις που μπορεί να κινηθεί το tricopter, αλλά κυρίως στις οριζόντιες, παρόλο που οι ταχύτητες που αναπτύσσει ένα τέτοιο όχημα συνήθως δεν είναι μεγάλες. Τέλος, η αντίσταση του αέρα αποσβένει τις γραμμικές και περιστροφικές κινήσεις του tricopter, αλλά οι ριπές ανέμου μπορούν να το μετακινήσουν ή ακόμη και να το βγάλουν εκτός ισορροπίας.

Αλγόριθμος ελέγχου του Tricopter

Ο έλεγχος του tricopter πραγματοποιείται με χρήση ελεγκτών PID για την ευστάθεια του και με τις εισόδους του χρήστη που μεταβάλλουν τα setpoints των ελεγκτών με αποτέλεσμα την κίνηση του. Πιο συγκεκριμένα, οι γωνίες κλίσης του tricopter εισέρχονται στον αλγόριθμο των PID οι οποίοι φροντίζουν την ευστάθεια, διατηρώντας τις γωνίες κλίσης σε μηδενικές μοίρες ή διαφορετικά όπου είναι το ορισμένο από το χρήστη setpoint.

Ο αλγόριθμος PID για τα roll και pitch, λαμβάνει τα αντίστοιχα setpoints από το χειριστήριο, τα οποία έχουν ήδη μετατραπεί από ms παλμού σε μοίρες γωνιών. Η μεταβλητή αυτή, αφαιρείται από την πραγματική γωνία που μας δίνει το IMU, παράγοντας το σφάλμα. Μέσω αυτού, ο PID υπολογίζει την κατάλληλη έξοδο, που αποστέλλεται στα μοτέρ.

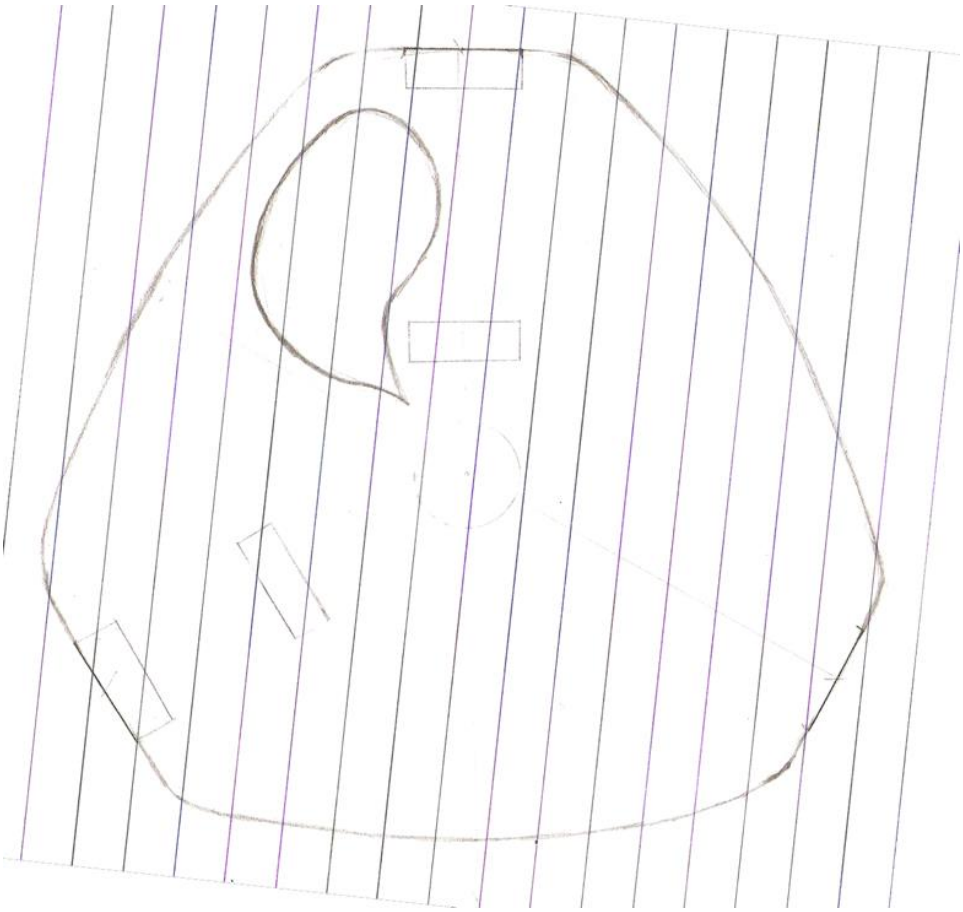
Ο αλγόριθμος PI για το yaw, λαμβάνει το δικό του setpoint από το χρήστη, που αυτή τη φορά η τιμή του διατηρείται σε ms παλμού, απλώς αφαιρώντας 1500 ms, τιμή που αντιστοιχεί στη μεσαία θέση του μοχλού. Αυτή η μεταβλητή αφαιρείται από τη γωνιακή ταχύτητα του άξονα *Z*, παράγοντας το σφάλμα. Από την τιμή αυτή, ο PI υπολογίζει την κατάλληλη έξοδο σε τιμή έτοιμη να σταλεί στο servo, μεταβάλλοντας την κλίση του πίσω μοτέρ M1, ρυθμίζοντας έτσι τη γωνιακή ταχύτητα.

Κεφάλαιο 2 - Μελέτη και Σχεδιασμός

Εισαγωγή

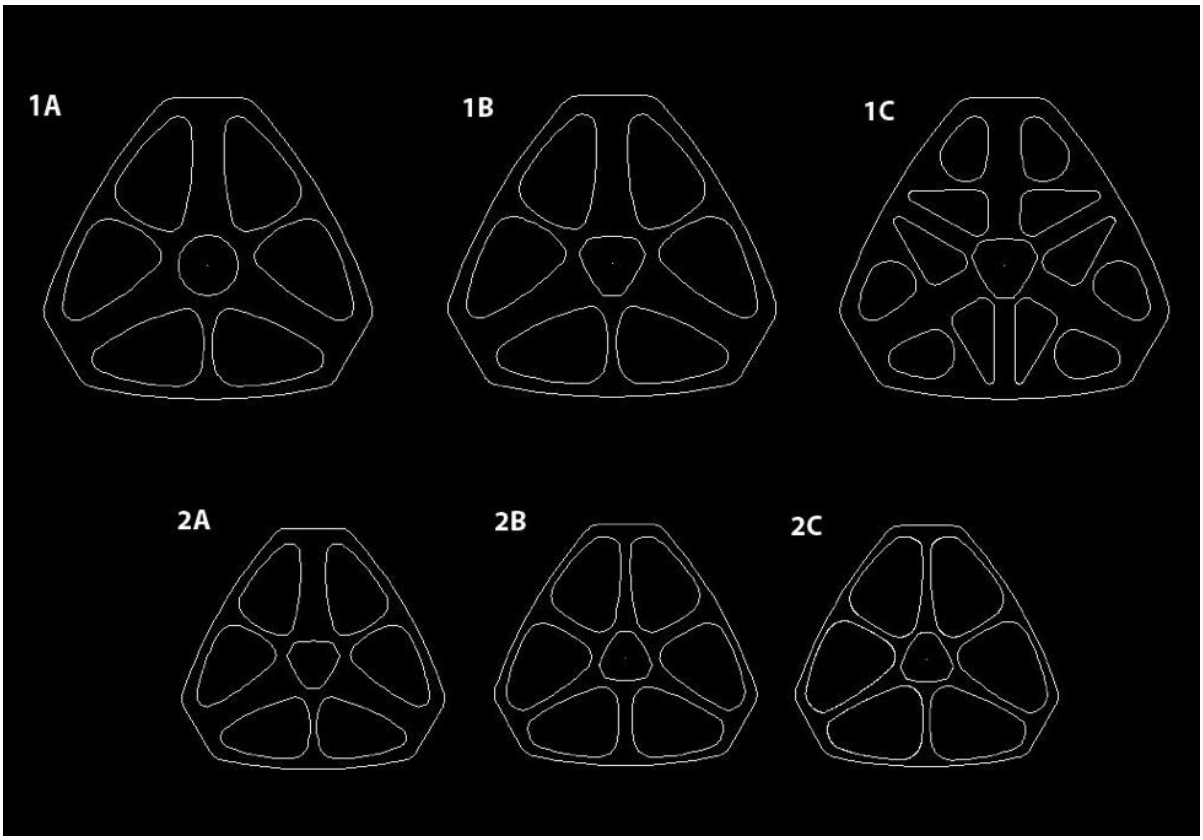
Βάσει των ανωτέρω, έγινε η μελέτη για την κατασκευή του tricopter. Μελετήθηκαν παράμετροι που θα επηρέαζαν τόσο την ίδια την υλική κατασκευή, όσο και τον απαραίτητο κώδικα για τη επιτυχή πτήση του οχήματος.

Αρχικά έγινε ένας πρώτος σχεδιασμός του σώματος του οχήματος και έπειτα η μεταφορά του στο AutoCAD, ώστε να μπορούν να κοπούν τα απαραίτητα κομμάτια σε φρέζα. Με την επιλογή του PCB σαν υλικό γι' αυτό το σκοπό, πέτυχαμε το κεντρικό κύτος του οχήματος να είναι ελαφρύ, χωρίς να θυσιάζουμε τη στιβαρότητα του. Ένα ακόμη πιθανό υλικό



Εικόνα 2.1: Αρχικό σχέδιο του σώματος του tricopter

προς χρήσει ήταν το plexiglass, το οποίο όμως για να έχει την ίδια στιβαρότητα, έπρεπε να είναι πιο παχύ και τελικά απορρίφθηκε. Έτσι τα τελικά υλικά που επιλέχθηκαν, κυριάρχησαν να είναι οι πλάκες PCB και το αλουμίνιο. Μετά από μερικά πιθανά σχέδια, επιλέχθηκε το πιο απλό και λειτουργικό.



Εικόνα 2.2: Διαφορετικές εκδοχές του σχεδίου του σώματος σε Autocad

Έπειτα από έρευνα επάνω στα μη επανδρωμένα ιπτάμενα οχήματα ανάλογων δυνατοτήτων, εκκίνησε η επιλογή των απαραίτητων υλικών. Τα κατάλληλα ηλεκτρονικά και μηχανικά μέρη έπρεπε να επιλεγούν πριν την κοπή των πρώτων υλικών του κύτους, ώστε να αποφευχθεί η ανάγκη για ανασχεδιασμό του οχήματος εξ αρχής. Τα μέρη αυτά επιλέχθηκαν με γνώμονα την αναγκαιότητα τους στην επιτυχή ολοκλήρωση της κατασκευής, το κόστος τους, την ευκολία χρήσης και προσαρμογής τους στις ανάγκες του τελικού προϊόντος, την ποιότητα κατασκευής και ορθής λειτουργίας χωρίς προβλήματα και σε κάποιες περιπτώσεις το βάρος και το μέγεθος τους. Ένα ακόμη σημαντικό κομμάτι στην επιλογή των υλικών κατέλαβε η συμβατότητα μεταξύ τους χωρίς τη μεσολάβηση ενδιάμεσων υλικών.

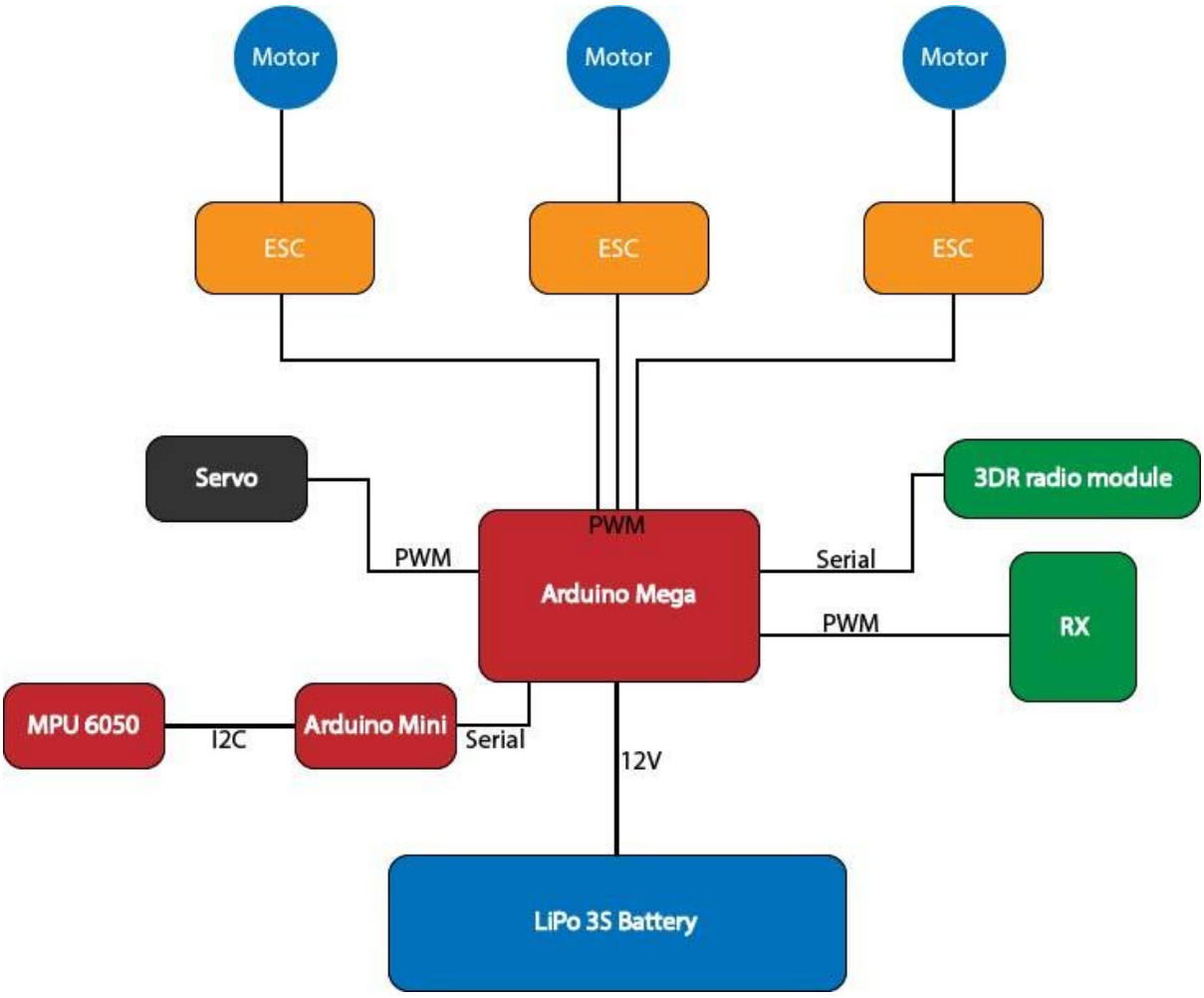
Παράλληλα το τελικό μέγεθος του οχήματος προσαρμόστηκε στις ανάγκες των υλικών που θα φιλοξενούσε και σχεδιάστηκε για να μας επιτρέπει τη διατήρηση του σε ένα μέγεθος όπου δε θα επιβάρυνε το κόστος ούτε θα προσέδιδε αχρείαστο βάρος. Επίσης θα ήταν σταθερό και με αντοχή σε πιθανές πτώσεις, ενώ θα προστάτευε τα ευαίσθητα υλικά όπως η μπαταρία και τα ηλεκτρονικά/ψηφιακά συστήματα, αλλά και θα επέτρεπε την εργασία επάνω του με

σχετική ευκολία. Τα πολλά επίπεδα που δημιουργήθηκαν βοήθησαν σημαντικά σε αυτό. Ολοκληρώθηκε έτσι ο σχεδιασμός του και προχώρησε η κοπή και χρήση του.

Το κομμάτι τις υλικής κατασκευής με τη μεγαλύτερη ανάγκη σε μελέτη, υπήρξε το οπίσθιο σύστημα μεταβολής της γωνίας του σχετικού μοτέρ. Καθώς επρόκειτο για το μοναδικό κινούμενο μέρος της κατασκευής, με αρκετά λεπτά χαρακτηριστικά και μεγάλο κίνδυνο καταστροφής σε περίπτωση πτώσης, έπρεπε να δεχθεί τη μεγαλύτερη προσοχή. Υπήρξαν αρκετές μεταβολές κατά τη διάρκεια των δοκιμών επάνω του, ειδικά ύστερα από την πρώτη πτώση του οχήματος, όπου η αρχική κατασκευή αδυνάτισε να αντέξει την κρούση.

Εκτενής μελέτη όμως πραγματοποιήθηκε και στο μη υλικό κομμάτι της κατασκευής. Έπειτα από την απαραίτητη έρευνα, επιλέχθηκαν το κατάλληλο λογισμικό που θα χρησιμοποιούνταν για τη δημιουργία των αναγκαίων προγραμμάτων. Οι πλατφόρμες των arduino και processing κυριάρχησαν λόγο ευκολίας χρήσης, cross-platform δυνατότητας και της μεγάλης ενεργής κοινότητας που έχουν αναπτύξει. Χρησιμοποιήθηκαν ως το τέλος της κατασκευής, παρόλο που υπήρξαν δοκιμές και άλλων επιλογών, όπως το AVR studio της εταιρίας Atmel για τον προγραμματισμό των μικροελεγκτών και το Visual Studio της εταιρίας Microsoft για τον προγραμματισμό της εφαρμογής τηλεμετρίας.

Δημιουργήθηκε, επίσης, ένα block διάγραμμα σύνδεσης των υλικών μεταξύ τους. Το διάγραμμα αυτό υπέστη μεταβολές με το χρόνο, όταν κάποια υλικά δέχθηκαν αντικατάσταση. Το τελικό, φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.3: Block διάγραμμα των υλικών του tricopter

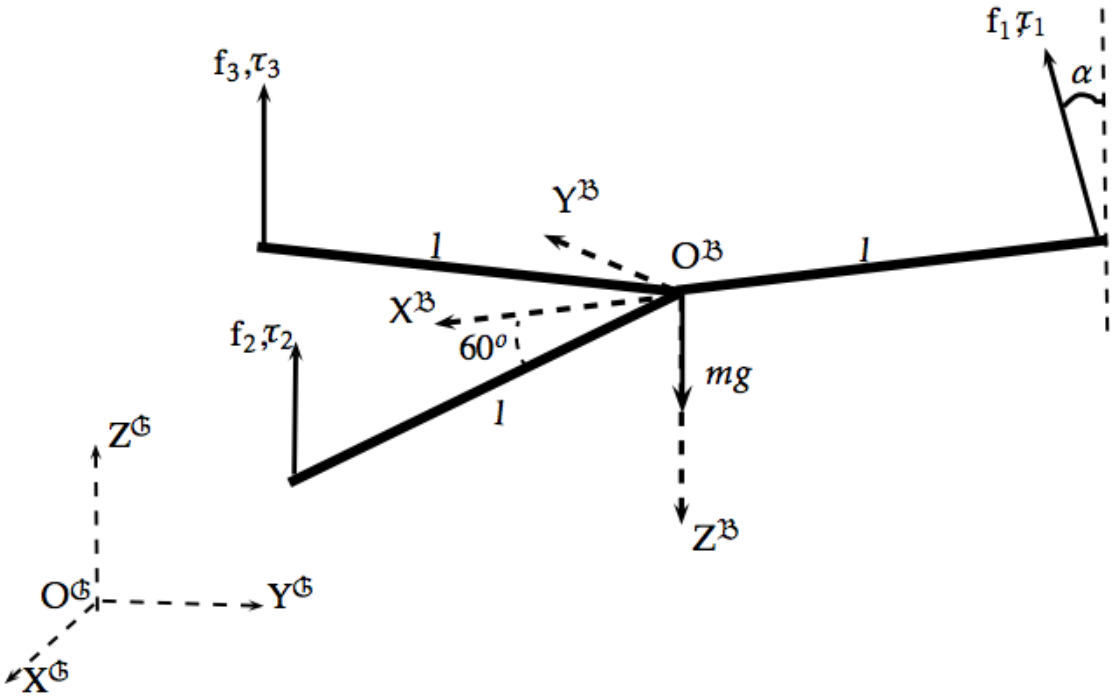
Μαθηματική Προσέγγιση

Έπειτα από την έρευνα που πραγματοποιήθηκε, προκύπτει η παρακάτω μαθηματική προσέγγιση του μοντέλου του οχήματος.

Συστήματα συντεταγμένων

Πριν την ανάλυση, τα συστήματα συντεταγμένων πρέπει να οριστούν ώστε να μπορούν να εκφραστούν τα σήματα εξόδου. Υπάρχουν τρία διαφορετικά συστήματα συντεταγμένων που πρέπει να καθοριστούν. Το τοπικό, που θεωρείται ως οι σταθερές συντεταγμένες του σώματος του tricopter και συμβολίζεται με το γράμμα *B* καθώς και το παγκόσμιο, που είναι το σύστημα συντεταγμένων της γης (περιβάλλον) και συμβολίζεται με το γράμμα *G*. Ένα τρίτο σύστημα συντεταγμένων θα μπορούσε επίσης να χρησιμοποιηθεί για να περιγράψει τις συντεταγμένες του οχήματος, το γεωγραφικό μήκος και πλάτος, τα οποία το GPS χρησιμοποιεί για προσδιορισμό θέσης του οχήματος. Αυτό το σύστημα συντεταγμένων συμβολίζεται με το γράμμα *E*.

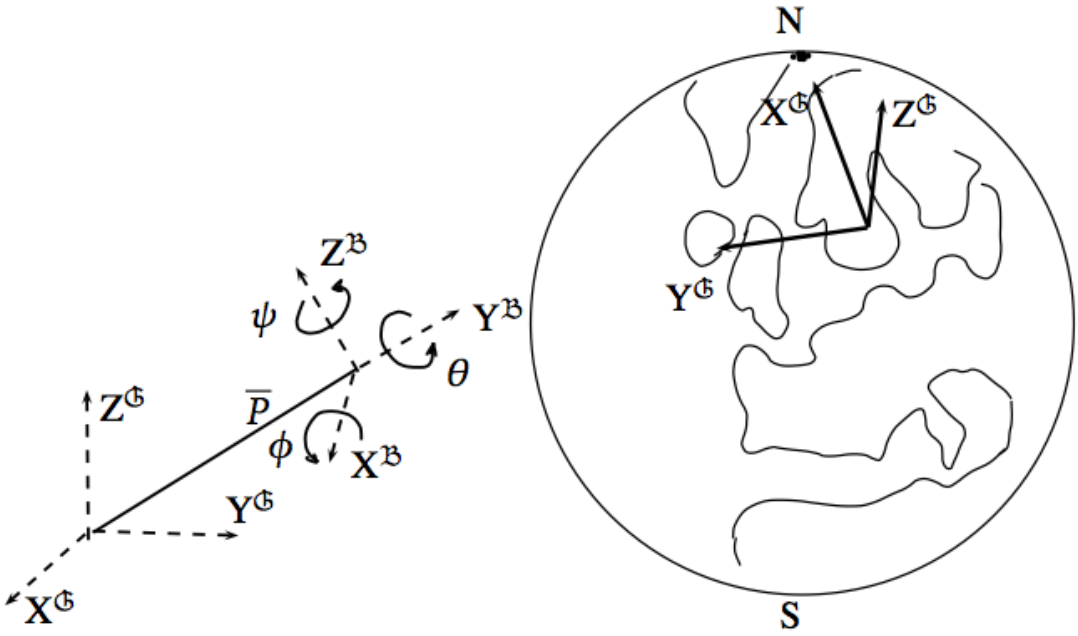
Το tricopter έχει τρία άκρα διαμορφωμένα σε σχήμα Υ, και ένα μοτέρ είναι τοποθετημένο στο τέλος του κάθε άκρου. Το σύστημα συντεταγμένων για το tricopter θα προσδιοριστεί όπως στο παρακάτω σχήμα.



Εικόνα 2.4: Το tricopter με τα συστήματα συντεταγμένων B και G

Ο άξονας X^B ορίζεται προς την κατεύθυνση ευθεία μπροστά από το tricopter, ενώ ο άξονας Y^B προς τα δεξιά του. Ο άξονας Z^B καθορίζεται ευθεία κάτω από το κέντρο μάζας του tricopter. Ο άξονας X^B μπορεί να θεωρηθεί ως επιθυμητή διεύθυνση προς τα εμπρός κατά τη διάρκεια μιας πτήσης. Δεδομένου ότι το οπίσθιο μοτέρ (*M1*) στρέφεται από ένα servo σε μια γωνία *α*, αυτή η γωνία πρέπει να οριστεί. Η γωνία *α* ορίζεται θετική όταν το στροφείο γέρνει προς τα δεξιά, όπως φαίνεται στο παραπάνω σχήμα.

Το σύστημα συντεταγμένων της γης έχει οριστεί ώστε ο άξονας X^G να δείχνει το Βορρά, ο άξονας Y^G είναι στραμμένος δυτικά και ο άξονας Z^G είναι στραμμένος προς τα επάνω, όπως εμφανίζεται στο παρακάτω σχήμα.



Εικόνα 2.5: Η σχέση μεταξύ του συστήματος συντεταγμένων G της γης και του συστήματος B του tricopter

Οι τρεις άξονες X^G, Y^G και Z^G ακολουθούν την επιφάνεια της γης. Αυτό δίνει ότι το διάνυσμα από μια θέση P στην επιφάνεια της γης ως το Βόρειο Πόλο, επικαλύπτεται με τον άξονα X^G ενώ το διάνυσμα που περνά από το σημείο P και είναι παράλληλο προς τον Ισημερινό, επικαλύπτεται με τον άξονα Y^G. Οι μηδενικές μοίρες του γεωγραφικού μήκους, λ_{lng}, ορίζονται στο Παρατηρητήριο του Greenwich και αυξάνουν θετικά ανατολικά του σημείου μηδέν. Αντίστοιχα οι μηδενικοί βαθμοί του γεωγραφικού πλάτους, λ_{lat}, ορίζονται στον Ισημερινό και αυξάνονται θετικά βόρεια αυτού. Οι συντεταγμένες γεωγραφικού μήκους και πλάτους χρησιμοποιούνται για να υπάρχει η δυνατότητα περιγραφής μια θέσης P στην επιφάνεια της γης, για την οποία θα ισχύει $P = (\lambda_{lat}, \lambda_{lng})^E = (x, y)^G$. Αυτό θεωρείται ως το τρίτο σύστημα συντεταγμένων και είναι ένας διαφορετικός τρόπος για να περιγραφεί η θέση του tricopter. Για να μπορέσει να χρησιμοποιηθεί το σύστημα συντεταγμένων E με το μοντέλο, δεδομένου ότι το GPS υποδεικνύει τη θέση του οχήματος στο σύστημα συντεταγμένων E, πρέπει να γίνει ένας μετασχηματισμός για τη μεταφορά στο σύστημα συντεταγμένων G.

Σήματα εισόδου και εξόδου

Για να είναι δυνατός ο έλεγχος του tricopter και να μπορεί να καθορίσει ένα δυναμικό πρότυπο αυτού, πρέπει να οριστούν τα σήματα εισόδου και εξόδου του συστήματος.

Το tricopter διαθέτει τρία μοτέρ με έλικες που δημιουργούν μια ροή αέρα που προκαλεί μια δύναμη ώθησης ώστε αυτό να μπορεί να πετάξει. Αυτή η δύναμη, *f*, μπορεί να υπολογιστεί κατά προσέγγιση ως ανάλογη με το τετράγωνο της γωνιακής ταχύτητας των περιστρεφόμενων ελικών. Δηλαδή $f = k_t\omega^2$. Η ροπή μπορεί επίσης να εκφραστεί με ίδιο τρόπο όπως η δύναμη ώθησης, $\tau = k_t\omega^2$. Δεδομένου ότι οι λεπίδες έχουν σταθερές γωνίες, η κατεύθυνση της ροής του αέρα εξαρτάται από την κατεύθυνση περιστροφής της λεπίδας. Ως εκ τούτου η δύναμη f_i και η ροπή τ_i μπορούν να εκφραστούν ως εξής:

$$f_i = k_i\omega_i|\omega_i|$$

$$\tau_i = k_i\omega_i|\omega_i|, \text{ για i = 1, 2, 3}$$

Αυτό δίνει ότι η γωνιακή ταχύτητα του μοτέρ, ω_i, είναι ένα σήμα εισόδου στο σύστημα και συμβολίζεται ως U_i, i = 1, 2, 3.

Το πρόβλημα που δημιουργήται με ένα tricopter, αλλά και γενικότερα με ένα multicopter που έχει μονό αριθμό μοτέρ είναι πώς το κάθε μοτέρ προκαλεί μια ροπή αντίδρασης που ισούται με την ροπή που παράγεται από την περιστροφή των ελίκων, αλλά με αντίθετη κατεύθυνση, $\tau_{rt} = -k_\tau\omega_i|\omega_i|$. Εάν το άθροισμα των ροπών αντίδρασης από τους έλικες είναι μηδέν, η ροπή που προκαλείται και εφαρμόζεται στο tricopter θα είναι μηδέν και αυτό σημαίνει ότι το όχημα δεν θα περιστρέφεται γύρω από τον άξονα Z^B. . Το συνολικό ποσό της ροπής αντίδρασης για n μοτέρ μπορεί να εκφραστεί ως $T_{rt} = \sum_{i=1}^{n} \tau_{rt,i}$. Αν ο αριθμός των μοτέρ n είναι μονός και κάθε ένας έχει την ίδια περίοδο περιστροφής με τους υπόλοιπους, $\omega_1 = \omega_2 = \omega_3$, πράγμα που συμβαίνει όταν το όχημα αιωρείται, η συνολική ροπή αντίδρασης, T_{rt}, θα είναι μη μηδενική τιμή. Αυτό συμβαίνει επειδή οι όροι δεν αλληλο-εξουδετερώνονται και σημαίνει ότι ένα multicopter όπως το tricopter θα περιστραφεί γύρω από το κέντρο μάζας του. Αυτό μπορεί να αντισταθμιστεί με την κλίση του ενός εκ των μοτέρ, που στην περίπτωση αυτή, θα είναι το οπίσθιο, όπως φαίνεται στο σχήμα των συστημάτων των συντεταγμένων *B* και *G*.

Σύμφωνα με τα παραπάνω μπορεί να υπολογιστεί αν η γωνία α (δηλαδή η κλίση του πίσω μοτέρ) είναι σταθερή για αιώρηση ή εξαρτάται από την μεταβολή της ροπής που ασκούν στο όχημα τα μοτέρ.

Ισχύει ότι η ροπή ενός μοτέρ είναι $\tau=k_\tau*\omega2$ και η ώση του είναι $F=k_t*\omega2$. Καθώς το πίσω μοτέρ έχει κλίση γωνίας α, η δύναμη που αντιτίθεται στη δύναμη της ροπής είναι $F_s=F*\sin\alpha$. Η δύναμη της ροπής F_τ ισούται με την ροπή επί την απόσταση του μοτέρ από το

κέντρο του tricopter, δηλαδή $F_\tau = \tau * l$, οπότε εξισώνοντας $F_\tau = F_s$ καταλήγουμε στο $\sin \alpha = k_\tau / k_t * l$, δηλαδή πως η γωνία α είναι ανεξάρτητη της γωνιακής ταχύτητας των μοτέρ.

Καθώς δεν μπορούν να υπολογιστούν οι σταθερές k_τ και k_t για να μετρηθεί θεωρητικά η γωνία α, με πειραματικό τρόπο να βρέθηκε σε ποια γωνία το tricopter ευσταθεί όσον αφορά την κατεύθυνση του.

Η γωνιακή ταχύτητα και η γωνία κλίσης θα δώσουν στο tricopter τέσσερα σήματα για τον έλεγχο της κίνησης: τις ταχύτητες των τριών μοτέρ και τη γωνία κλίσης του πίσω μοτέρ. Αυτά τα σήματα εισόδου στο σύστημα θα επηρεάσουν τον προσανατολισμό και τη μετακίνηση του οχήματος. Θεωρώντας κατά προσέγγιση το tricopter ως ένα άκαμπτο σώμα με 6 DOF, οι καταστάσεις του συστήματος που περιγράφουν την κίνηση μπορούν να οριστούν ως: $(x, y, z)^G$, $(u, v, w)^G$, $(\varphi, \theta, \psi)^G$, $(p, q, r)^B$. Αυτά είναι η θέση και η ταχύτητα του tricopter στο σύστημα συντεταγμένων G, η περιστροφή των γωνιών στο σύστημα G και η περιστροφή γύρω από τους άξονες στο σύστημα B, αντίστοιχα.

Τα σήματα εξόδου περιγράφουν την κατάσταση του οχήματος όπως αυτή μετράται από τα σήματα εισόδου. Χρησιμοποιώντας τις μετρήσεις των διαθέσιμων αισθητηρίων του οχήματος, έχουμε ότι: ρυθμός της περιστροφής, $(p, q, r)^B$, μπορεί να μετρηθεί με τα γυροσκόπια. Ο προσανατολισμός του οχήματος, ψ^G, μπορεί να προσδιοριστεί από το μαγνητόμετρο και φανερώνει το πώς το tricopter είναι προσανατολισμένο σε σχέση με το μαγνητικό πεδίο της γης, το οποίο θεωρείται παράλληλο με τον άξονα X^G. Το GPS υποδεικνύει τη γεωγραφική θέση του οχήματος $(\lambda_{lat}, \lambda_{lng})^E$ και το υψόμετρο z^G μπορεί να μετρηθεί είτε με ένα βαρόμετρο, είτε με έναν αισθητήρα σόναρ. Τέλος, μέσω των στιγμάτων του GPS μπορούμε να υπολογίσουμε την απόλυτη ταχύτητα του οχήματος, $|V|^G$.

Άρα τα σήματα εξόδου είναι τα $(\lambda_{lat}, \lambda_{lng})^E$, z^G, $|V|^G$, ψ^G, $(p, q, r)^B$.

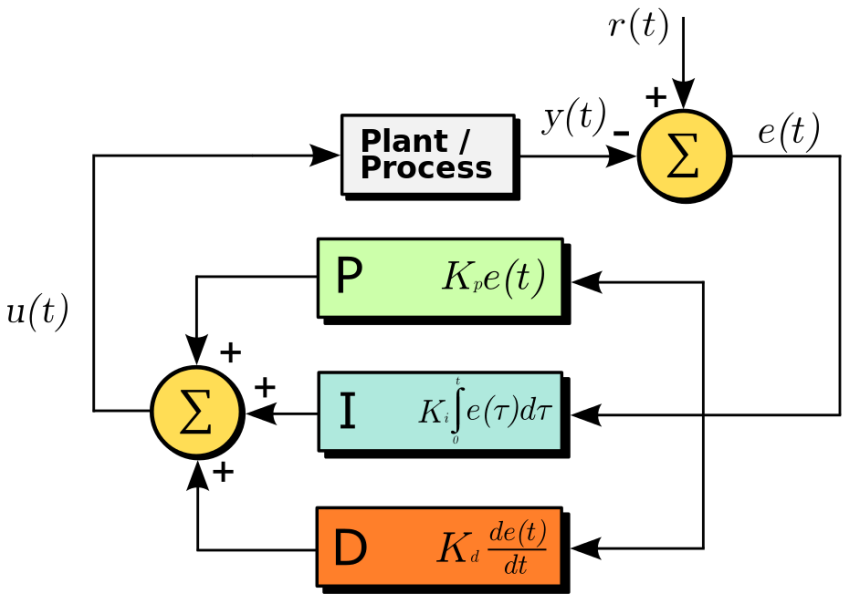
Στην περίπτωση αυτής της πτυχιακής εργασίας, το tricopter δεν διαθέτει μαγνητόμετρο στην τελική του μορφή, όπως ούτε και αισθητήρα GPS. Ουσιαστικά από τις εξόδους του συστήματός μας μπορούμε να μετρήσουμε μονάχα τον ρυθμό περιστροφής $(p, q, r)^B$ με τα γυροσκόπια και τη στατική κλίση με τα επιταχυνσιόμετρα. Συνεπώς, με τα διαθέσιμα αισθητήρια ο ελεγκτής μπορεί να ελέγξει μόνο την ευστάθειά του οχήματος εν πτήση.

Ελεγκτής PID

Για τον έλεγχο του tricopter χρησιμοποιούνται 2 PID ελεγκτές για την κλίση εμπρός/πίσω και πλάγια, και ένας ελεγκτής PI για την κατεύθυνσή του. Οι 2 PID ελεγκτές ελέγχουν την γωνία του tricopter στους οριζόντιους άξονες, ενώ ο PI ελεγκτής ελέγχει την γωνιακή ταχύτητα στον κάθετο άξονα.

Ο ελεγκτής PID είναι από τους πιο απλούς και διαδεδομένους ελεγκτές, είναι από τους πρώτους ελεγκτές που διδάσκονται στα πανεπιστήμια και συνήθως καλύπτει με την απόδοση του τις περισσότερες εφαρμογές, είναι αρκετά όμως δύστροπος στον συντονισμό. Ειδικά σε εφαρμογή σε multicopters αν και έχουν βγει αρκετές μεθοδολογίες που επιτρέπουν εύκολο συντονισμό -όπως η Ziegler-Nichols και η Cohen-Coon- οι οποίες όμως είναι δύσκολο να εφαρμοστούν στο tricopter και τα αποτελέσματα τους ίσως απέχουν πολύ από το ιδανικό, καθώς μεταξύ του τροποποιημένου συστήματος για την εφαρμογή της μεθοδολογίας και το πραγματικό σύστημα, υπάρχουν μεγάλες διαφορές. Αυτό συμβαίνει διότι η διαδικασία των μεθοδολογιών αυτών δε μπορεί να πραγματοποιηθεί σε ένα tricopter ελεύθερο στον αέρα. Έτσι, η βάση που θα το κρατά επαρκώς σταθερό, αφήνοντας μόνο τον έναν βαθμό ελευθερίας ελεύθερο, παραλλάσει πολύ το σύστημα.

Ο ελεγκτής PID στην ουσία αποτελείται από τρία διαφορετικά στοιχεία τα οποία συνδυάζονται για να δημιουργηθεί ο ελεγκτής. Τα στοιχεία αυτά είναι τα P, I και D και ανάλογα με την επιθυμητή προς έλεγχο εφαρμογή και τις απαιτήσεις που υπάρχουν, συνδυάζονται διαφορετικά, δηλαδή μπορεί να υπάρχει μόνο το στοιχείο P και αντίστοιχα ελεγκτής P, τα στοιχεία PI άρα ελεγκτής PI, αντίστοιχα ελεγκτής PD και τέλος ελεγκτής PID.



Εικόνα 2.6: Η σχέση μεταξύ του συστήματος συντεταγμένων G της γης και του συστήματος B του tricopter

Όπως φαίνεται στο διάγραμμα της παραπάνω εικόνας, ο ελεγκτής δέχεται σαν είσοδο το σφάλμα που απορρέει από τη διαφορά του επιθυμητού set point που δίνει ο χρήστης *r(t)* και της απόκρισης του συστήματος *y(t)*. Έπειτα ο PID υπολογίζει, σύμφωνα με τις παραμέτρους που έχει δώσει ο χρήστης, το αποτέλεσμα *u(t)* το οποίο ανατροφοδοτείται στο σύστημα για να διορθωθεί το σφάλμα. Η διαδικασία επαναλαμβάνεται, καθώς ο ελεγκτής PID πρόκειται για ελεγκτή κλειστού βρόγχου.

Μια απλή μαθηματική παρουσίαση ενός ελεγκτή PID είναι:

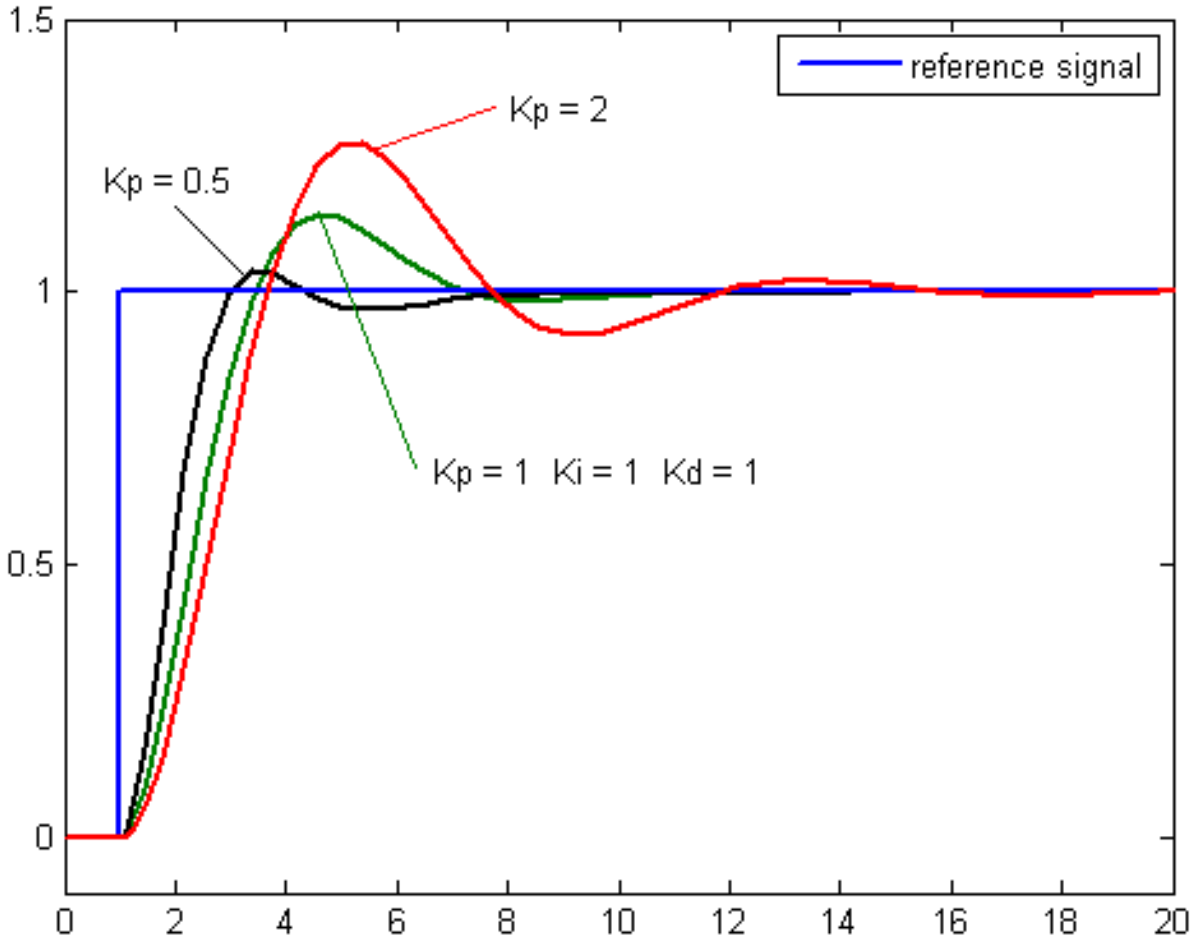
$$e(t) = r(t) - y(t)$$

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t)$$

Αναλυτικά, τα τρία στοιχεία του ελεγκτή PID έχουν ως εξής:

Στοιχείο P

$$P = K_p e(t)$$

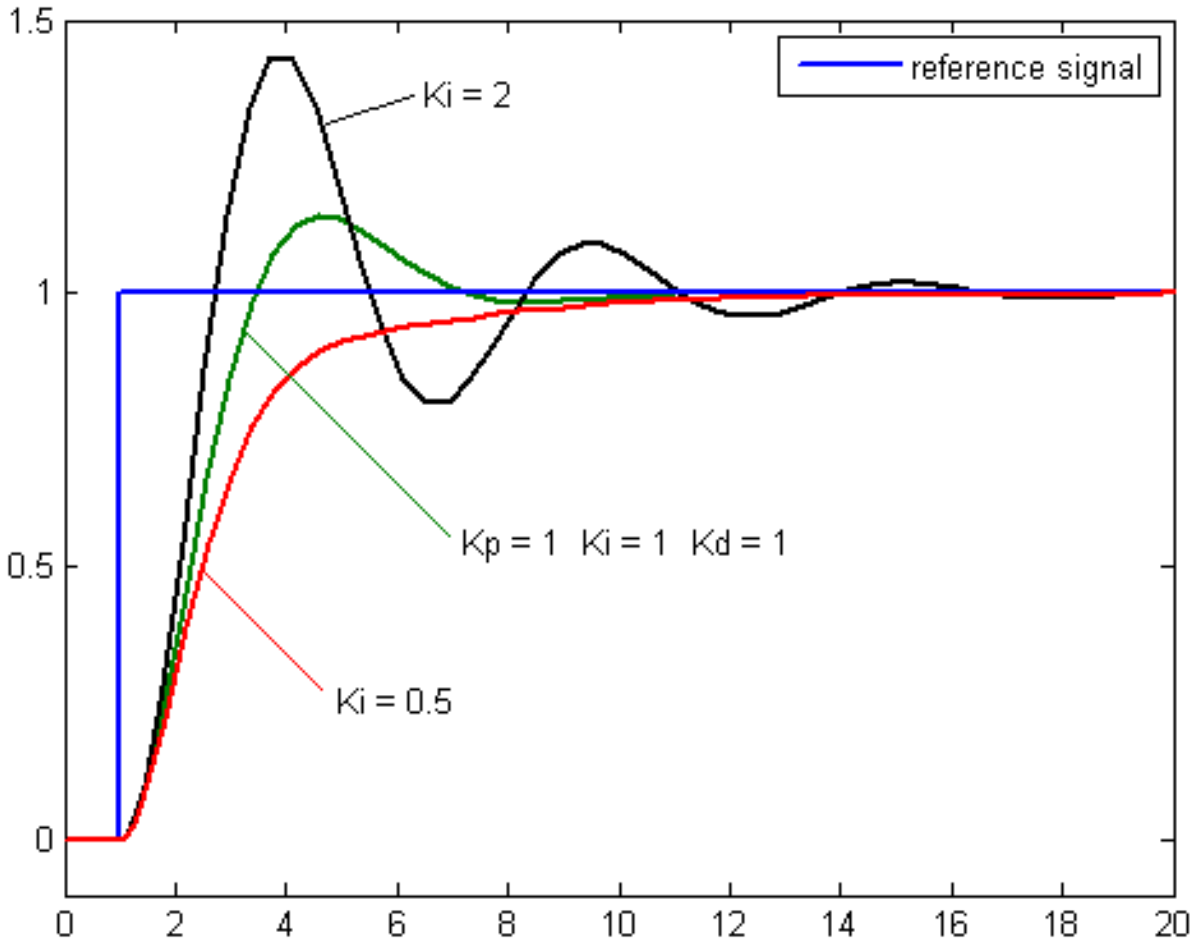


Εικόνα 2.7: Η συμπεριφορά ενός συστήματος όταν οι τιμές των Ki και Kd παραμένου σταθερές και μόνο η τιμή του Kp μεταβάλλεται.

Η ονομασία του στοιχείου αυτού προέρχεται από τον όρο *Proportional* (αναλογικό). Ακόμη, ονομάζεται και κέρδος (*gain*) και είναι ανάλογο του σφάλματος που έχει το σύστημά μας. Το πολλαπλασιάζουμε με το σφάλμα του συστήματος για να διορθωθεί. Εάν η τιμή του είναι μικρότερη της ιδανικής για το σύστημα, δε θα υπάρχει επαρκείς διόρθωση του σφάλματος. Αντιθέτως, εάν είναι μεγαλύτερη, θα υπάρχει αφενός πιο άμεση απόκριση, αλλά μπορεί να οδηγήσει σε ταλάντωση και εν συνεχεία αστάθεια.

Στοιχείο I

$$I = K_i \int_0^t e(t)dt$$

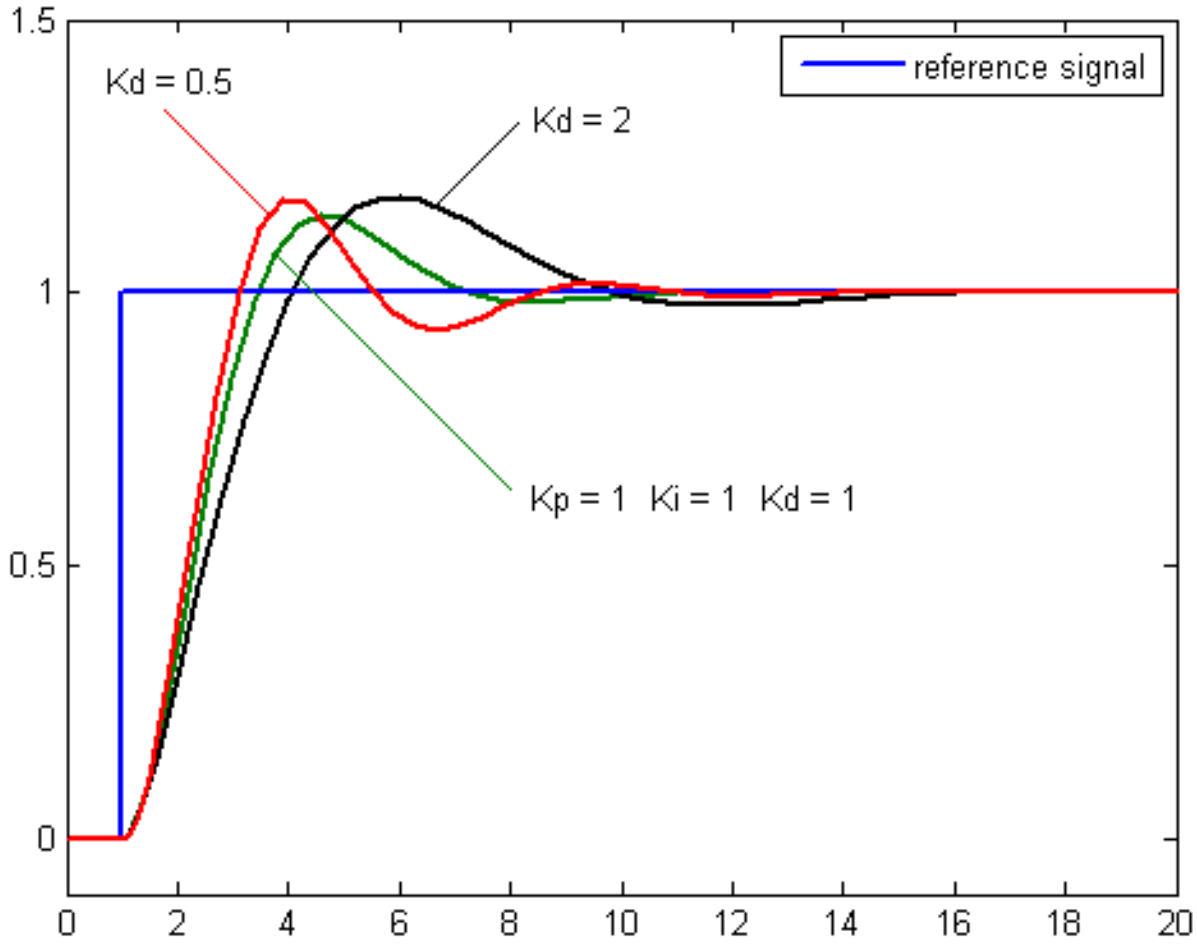


Εικόνα 2.8: Η συμπεριφορά ενός συστήματος όταν οι τιμές των Kp και Kd παραμένου σταθερές και μόνο η τιμή του Ki μεταβάλλεται.

Η ονομασία του στοιχείου αυτού προέρχεται από τον όρο *Integral* (ολοκληρωτικό). Χρησιμοποιείται για να ολοκληρώσει το σφάλμα, ουσιαστικά προσθέτοντας διαρκώς το μόνιμο σφάλμα που δημιουργεί το στοιχείο P για να το διορθώσει. Εάν το στοιχείο αυτό είναι μεγαλύτερο του ιδανικού μπορεί να δημιουργήσει ταλάντωση στο σύστημα, ενώ εάν είναι πολύ μικρό να καθυστερήσει τη διόρθωση του σφάλματος.

Στοιχείο D

$$D = K_d \frac{d}{dt} e(t)$$



Εικόνα 2.9: Η συμπεριφορά ενός συστήματος όταν οι τιμές των Kp και Ki παραμένου σταθερές και μόνο η τιμή του Kd μεταβάλλεται.

Η ονομασία του στοιχείου αυτού προέρχεται από τον όρο *Derivative* (παραγωγικό) και πρόκειται ουσιαστικά για τη διαφορά μεταξύ του σφάλματος της χρονικής στιγμής *t-1* και του σφάλματος της χρονικής στιγμής *t*. Ο ρόλος του στοιχείου αυτού είναι η μείωση ή και εκμηδένιση των απότομων μεταβολών στο σύστημα που θα μπορούσαν να οδηγήσουν σε ταλαντώσεις και αστάθεια. Ο συντονισμός του στοιχείου αυτού απαιτεί προσοχή ώστε να μην οδηγεί το ίδιο σε υπέρβαση ή αστάθεια. Μπορεί δηλαδή ένα σύστημα με καλή συμπεριφορά σε μικρές μεταβολές, να γίνει ασταθές έπειτα από συνεχώς αυξανόμενες ταλαντώσεις λόγω του στοιχείου *D*.

Πειραματική ρύθμιση ελεγκτή PID

Καθώς η ρύθμιση της ευστάθειας του οχήματος σε πραγματικές συνθήκες δε μπορεί να υλοποιηθεί απευθείας, έπρεπε να υπάρξει μια πιο πειραματική προσέγγιση σε ιδανικές συνθήκες. Η πιο εύκολη και ασφαλής προσέγγιση είναι η δημιουργία μιας "τραμπάλας", δηλαδή, η όσο το δυνατόν εκμηδένιση της κίνησης σε δύο εκ των αξόνων, ώστε να υπάρχει η δυνατότητα απομόνωσης του τρίτου άξονα, για ευκολότερο έλεγχο του. Με αυτό τον τρόπο καθίσταται δυνατός ο πειραματισμός των δυνατοτήτων και ο συντονισμός των παραμέτρων του ελεγκτή PID.

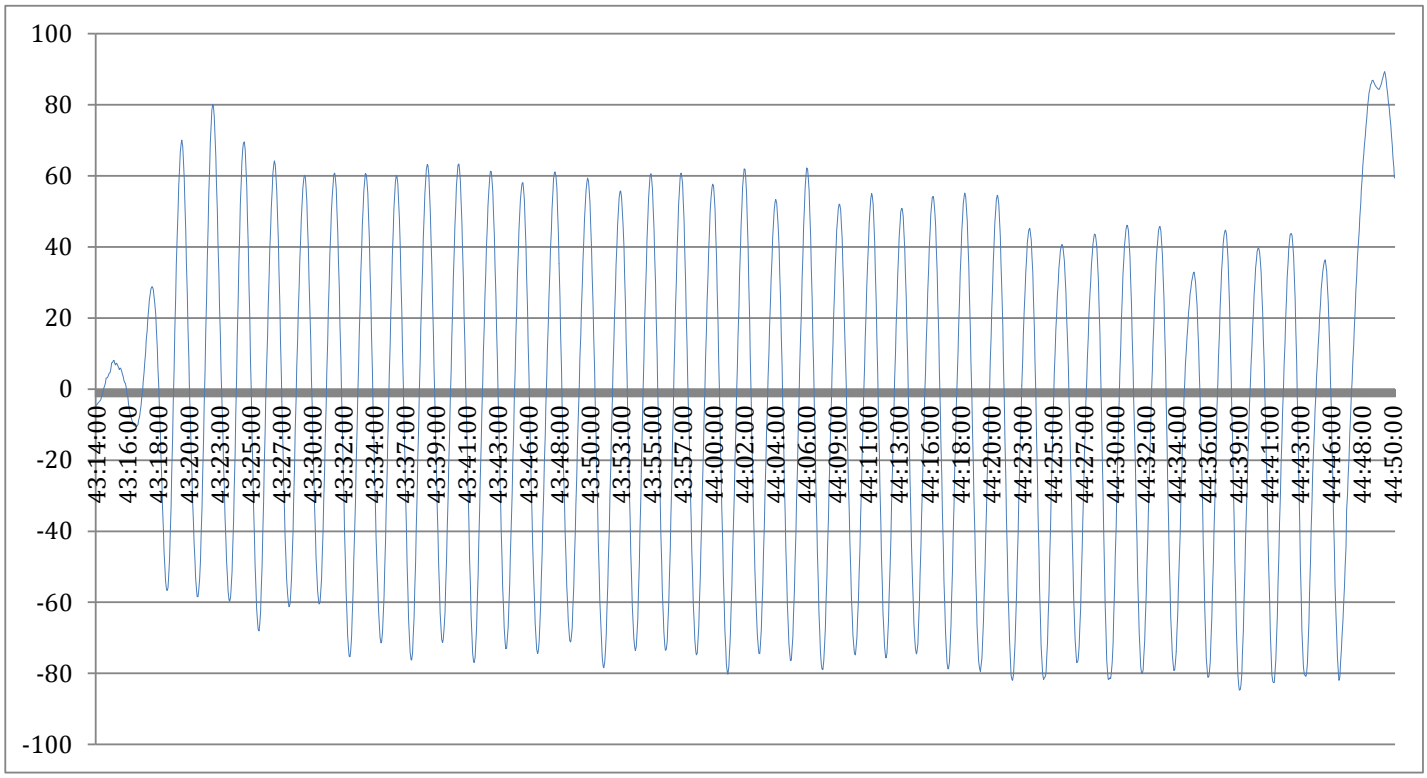
Η διαδικασία αυτή βοήθα σε μεγάλο βαθμό στην περαιτέρω κατανόηση της συμπεριφοράς του συστήματος ανάλογα με τις παραμέτρους του PID, ώστε να είναι ευκολότερος ο πιο ακριβής συντονισμός του.

Η κατασκευή αποτελείται από δύο σταθερά σημεία στο περιβάλλον από τα οποία δένεται το tricopter με τέτοιο τρόπο ώστε ο μόνος ελεύθερος άξονας να είναι ο *X* (*roll*). Στη συνέχεια, μηδενίζονται οι παράμετροι όλων των ελεγκτών του συστήματος. Από αυτό το σημείο, ακολουθείται η διαδικασία της μεθοδολογίας Ziegler-Nichols. Δηλαδή αυξάνεται ό όρος *P* του ελεγκτή του άξονα *X* σταδιακά ωσότου το σύστημα να κάνει σταθερή ταλάντωση. Ο όρος του κέρδους σε αυτή την κατάσταση ονομάζεται κρίσιμο κέρδος K_{cr}. Χρησιμοποιώντας το K_{cr} και την περίοδο ταλάντωσης του συστήματος T_0, βάση του παρακάτω πίνακα, η μέθοδος αυτή δίνει τις τιμές των όρων του ελεγκτή κοντά στο βέλτιστο για το παρόν σύστημα.

Πίνακας 2.1: Πίνακας εύρεσης τιμών PID με τη μέθοδο Ziegler-Nichols

	K_p	K_i	K_d
P	$0.5\ K_{cr}$		
PI	$0.5\ K_{cr}$	$1.2\ K_p / T_0$	
PID	$0.6\ K_{cr}$	$2\ K_p / T_0$	$0.125\ K_p\ T_0$

Από το πείραμα που πραγματοποιήθηκε στο όχημα της εργασίας αυτής, προέκυψε η παρακάτω κυμματομορφή:



Εικόνα 2.10: Η περιοδική ταλάντωση του άξονα X του tricopter κατά τη διαδηκασία της μεθοδολογίας Ziegler-Nichols

Από ανάλυση των δεδομένων, προκύπτουν οι τιμές των δύο μεταβλητών $K_{cr} = 0.67$ και $T_0 = 2.41$ sec. Επομένως, εφαρμόζοντας τη μεθοδολογία Ziegler-Nichols, για ελεγκτή τύπου PID, οι τιμές των παραμέτρων θα είναι: $K_p = 0.4$, $K_i = 1.9$, $K_d = 0.12$.

Αλγόριθμος DCM

Η στροφή ενός συστήματος συντεταγμένων από μια θέση σε μια άλλη είναι δυνατόν να περιγραφεί από ένα πίνακα, που ονομάζεται Direction Cosine Matrix (DCM). Για να περιγραφεί ο πίνακας DCM, απαιτείται ένα συγκεκριμένο μαθηματικό μοντέλο, σύμφωνα με το οποίο η μετάβαση από την αρχική θέση των διανυσμάτων βάσης στη νέα θέση μπορεί να επιτευχθεί με τη βοήθεια τριών διαδοχικών στροφών γύρω από τους άξονες *X, Y, Z* κατά αντίστοιχες γωνίες *φ, θ, ψ*. Οι στροφές αυτές περιγράφονται από τους στοιχειώδεις ορθογώνιους πίνακες στροφής *R1(φ), R2(θ)* και *R3(ψ)*.

Τηρώντας τη σειρά στροφών γύρω από τους άξονες Z, Y, X ο συνολικός πίνακας στροφής είναι:

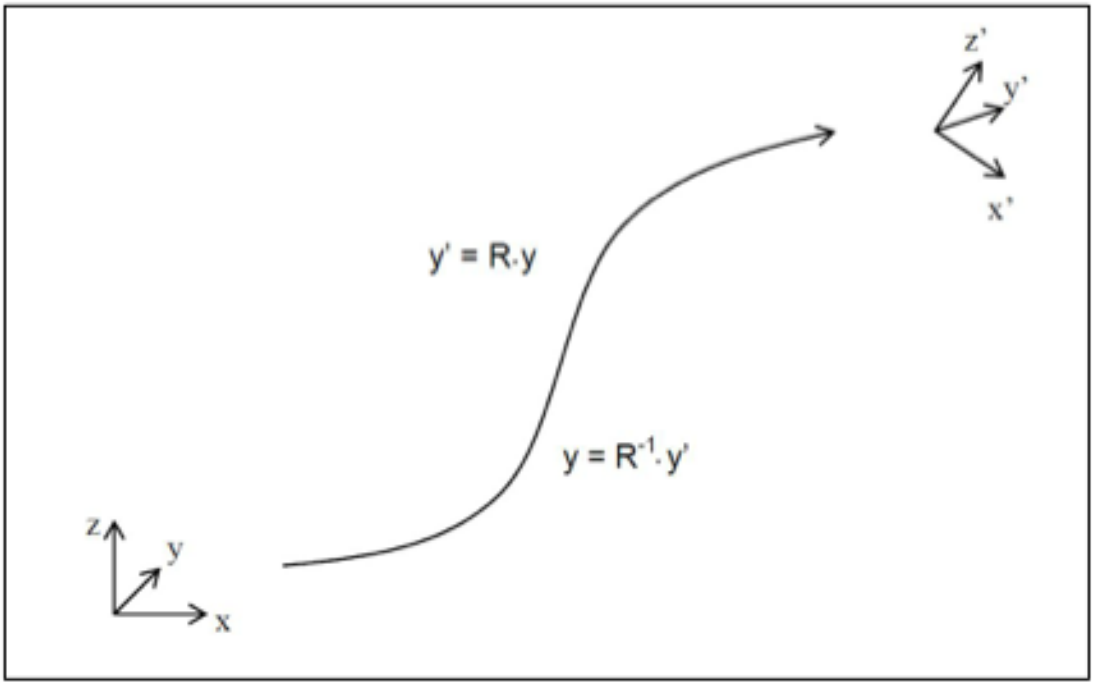
$$R=R_1(\varphi)*R_2(\theta)*R_3(\psi)$$

ή αναλυτικά

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Στην παραπάνω σχέση, οι πίνακες εμφανίζονται στο γινόμενο από τα δεξιά προς τα αριστερά σύμφωνα με τη σειρά που εφαρμόστηκαν οι στροφές γύρω από τους άξονες.

Για το μετασχηματισμό ενός διανύσματος y από το ένα σύστημα συντεταγμένων στο είναι y'=R·y και αντίστροφα y=R^{-1}·y', ενώ ισχύει η ιδιότητα ότι $R^{-1}=R^T$.



Εικόνα 2.11: Μετασχηματισμός διανύσματος μεταξύ δύο συστημάτων συντεταγμένων

Ο πίνακας DCM για την αναπαράσταση του τρισδιάστατου προσανατολισμού μπορεί να εκφραστεί ως εξής:

$$DCM = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

και πιο αναλυτικά, συναρτήσει των γωνιών στροφής:

$$DCM = \begin{bmatrix} \theta_c\psi_c & -\varphi_c\psi_s + \varphi_s\theta_s\psi_c & \varphi_s\psi_s + \varphi_c\theta_s\psi_c \\ \theta_c\psi_s & \varphi_c\psi_c + \varphi_s\theta_s\psi_s & -\varphi_s\psi_c + \varphi_c\theta_s\psi_s \\ -\theta_s & \varphi_s\theta_c & \varphi_c\theta_c \end{bmatrix}$$

Οι δείκτες *s* και *c* αντιστοιχούν στα ημίτονο και συνημίτονο.

Αντίστροφα, είναι δυνατό να υπολογιστούν οι γωνίες στροφής από τον πίνακα DCM, με τις παρακάτω σχέσεις:

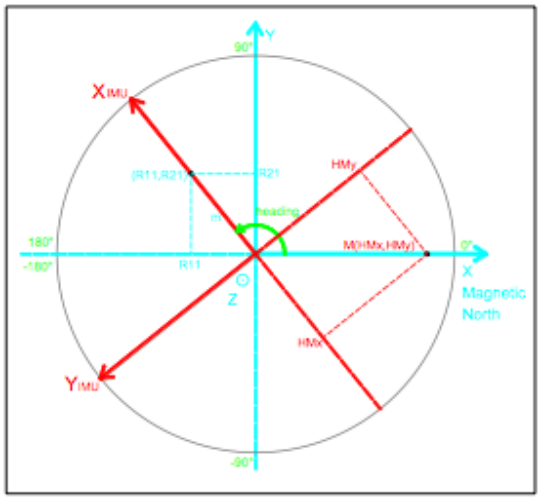
$$\varphi = \tan^{-1}\left(\frac{R_{32}}{R_{33}}\right)$$

$$\theta = \sin^{-1} -R_{31} = \tan^{-1}\left(\frac{R_{31}}{\sqrt{{R_{32}}^2 + {R_{33}}^2}}\right)$$

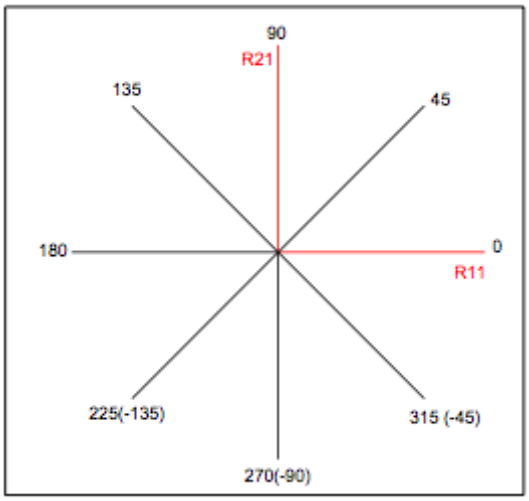
$$\psi = \tan^{-1}\left(\frac{R_{21}}{R_{11}}\right)$$

Είναι προφανές ότι δεν ορίζονται οι γωνίες φ και ψ, στις περιπτώσεις που είναι R33=0 και R11=0 αντίστοιχα. Πρακτικά αυτό συμβαίνει, όταν η γωνία θ ή αντίστοιχη γωνία πλοήγησης pitch γίνεται ίση με ±90 μοίρες, που σημαίνει ότι ο άξονας X της αδρανειακής μονάδας μέτρησης συμπίπτει με τη κατακόρυφο ή ισοδύναμα με τον άξονα Z του επίγειου συστήματος αναφοράς.

Η σύνδεση του συστήματος συντεταγμένων της αδρανειακής μονάδας μέτρησης και του επίγειου συστήματος αναφοράς πραγματοποιείται με τη βοήθεια της γωνίας heading, που σχηματίζεται μεταξύ της προβολής του άξονα X της αδρανειακής μονάδας μέτρησης στο οριζόντιο επίπεδο και του μαγνητικού βορρά. Ο υπολογισμός της γωνίας heading πραγματοποιείται με τη βοήθεια των οριζόντιων προβολών HMx και HMy των συνιστωσών της έντασης του μαγνητικού πεδίου, όπως αυτές μετρούνται στους άξονες της αδρανειακής μονάδας μέτρησης. Από τη γωνία heading υπολογίζονται οι συνιστώσες R11 και R21, οι οποίες και αποτελούν και τα στοιχεία του πίνακα στροφής, όπως φαίνεται στις παρακάτω εικόνες.



Εικόνα 2.12: Σύνδεση των συστημάτων συντεταγμένων του αδρανιακής μονάδας μέτρησης και του επίγειου συστήματος αναφοράς στο οριζόντιο επίπεδο.

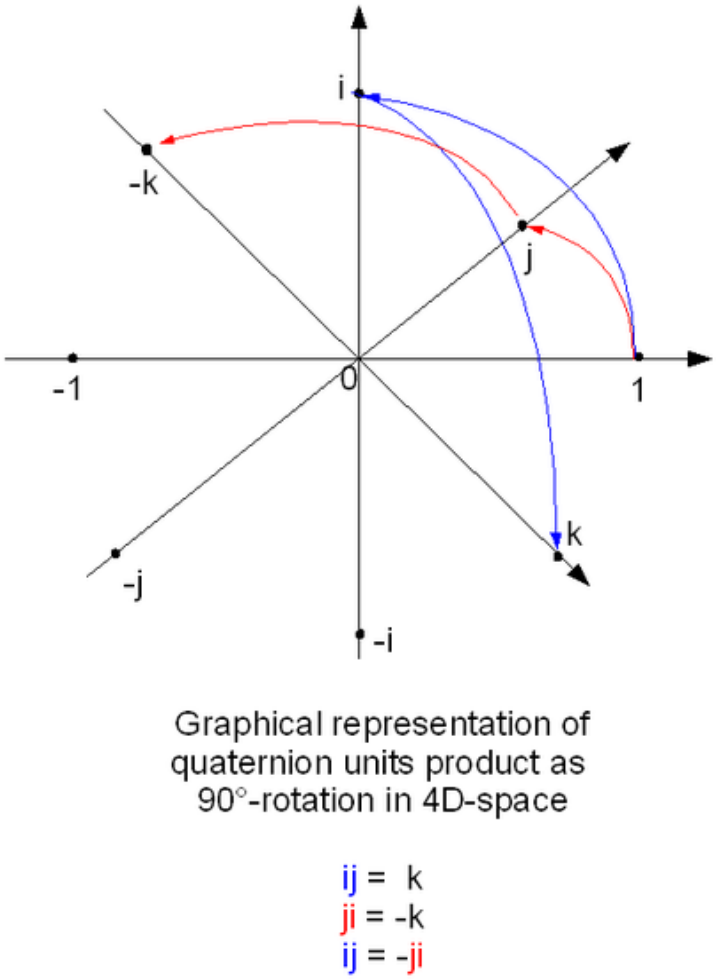


Εικόνα 2.13: Οι τιμές των στοιχείων R11 και R21 του πίνακα DCM για διαφορετικές γωνίες προσανατολισμού (σε μοίρες)

Quaternions

Τα quaternions είναι ένα σύστημα τεσσάρων αριθμών που επεκτείνουν το σύνολο των σύνθετων αριθμών. Ανακαλύφθηκαν από τον Ιρλανδό μαθηματικό William Rowan Hamilton το 1843 και χρησιμοποιούνται σε πεδία των θεωρητικών και εφαρμοσμένων επιστημών για την περιγραφή της τρισδιάστατης περιστροφής ενός αντικειμένου στο χώρο. Το σύνολο των τεσσάρων αυτών αριθμών αποτελείται από τρεις φανταστικούς και έναν πραγματικό αριθμό. Χαρακτηριστικό των φανταστικών αριθμών είναι πως ο πολλαπλασιασμός τους δε διαθέτει την αντιμεταθετική ιδιότητα, καθώς και ότι περιγράφονται από την παρακάτω εξίσωση:

$$i^2 = j^2 = k^2 = ijk = -1$$



Εικόνα 2.14: Γραφική αναπαράσταση των προϊόντων των μονάδων των quaternion ως περιστροφή 90 μοιρών στον τετραδιάστατο χώρο

Υλοποίηση στην παρούσα εργασία

Στην παρούσα εργασία, η βιβλιοθήκη που χρησιμοποιήται για την εξαγωγή των τιμών των γωνιών από τις εξόδους των αισθητηρίων κάνει χρήση του φίλτρου DCM του Robert Mahony σε μορφή quaternions όπως υλοποιήθηκε από τον Sebastian Madgwick.

Το φίλτρο Kalman δε χρησιμοποιήθηκε, καθώς η απαιτούμενη επεξεργαστική ισχύς για τους πολυσύνθετους υπολογισμούς του το καθιστούν ασύμφορο για μικροελεγκτές χαμηλών δυνατοτήτων, όπως αυτοί που χρησιμοποιούνται στα UAVs, παρ' όλο που η ενδεχομένως μεγαλύτερη ακρίβεια του μπορεί να ήταν επιθυμητή.

Ακόμη, το Complementary filter δε χρησιμοποιήθηκε παρόλη την ευκολία και ταχύτητα του, καθώς η ακρίβεια που μπορεί να προσφέρει κρίθηκε ανεπαρκής. Επίσης, το Complementary filter εκφράζεται σε γωνίες Euler, οι οποίες διατρέχουν τον κίνδυνο εμφάνισης gimbal lock. Γι' αυτό η χρήση του φίλτρου DCM σε μορφή quaternions κρίθηκε ως η βέλτιστη. Ακόμη, η μέθοδος γωνιών Euler μπορεί κάποιες φορές να δημιουργήσει ασάφεια ανάλογα με το χρησιμοποιούμενο αισθητήριο ως προς τη σειρά με την οποία υπολογίζονται οι άξονες. Έτσι, μπορεί να υπάρχει ασυμβατότητα μεταξύ δύο κομματιών κώδικα ή βιβλιοθηκών, κάτι που δε συμβαίνει με τα quaternions καθώς αντιπροσωπεύουν μόνο μια περιστροφή ενός καθορισμένου άξονα.

Ο τρόπος με τον οποίο γίνεται η εξαγωγή των τιμών των γωνιών από το IMU φαίνεται στο παρακάτω κομμάτι κώδικα:

```
/*
Όπου:
gx,gy,gz: Οι raw τιμές των γυροσκοπίων
ax,ay,az: Οι τιμές των επιταχυνσιομέτρων σε ακτίνια
recipNorm: προσωρινή τιμή για κανονικοποιήσεις
halfvx,halfvy,halfvz: Εκτιμώμενη κατεύθυνση της βαρύτητας
halfex,halfey,halfez: Σφάλματα μεταξύ της εκτιμώμενης και της μετρήσιμης κατεύθυνσης της βαρύτητας
sampleFreq: Ο ρυθμός δειγματοληψίας
invSqrt: συνάρτηση για 4 φορές γληγορότερη μέτρηση της αντίστροφης τετραγωνικής ρίζας (1 / Sqrt)
*/


void MahonyAHRSupdateIMU(float gx, float gy, float gz, float ax, float ay, float az) {
  float recipNorm;
  float halfvx, halfvy, halfvz;
  float halfex, halfey, halfez;
  float qa, qb, qc;

  // Κανονικοποίηση της μέτρησης από τα επιταχυνσιόμετρα
  recipNorm = invSqrt(ax * ax + ay * ay + az * az);
  ax *= recipNorm;
  ay *= recipNorm;
  az *= recipNorm;

  // Εκτιμώμενη κατεύθυνση της βαρύτητας
  halfvx = q1 * q3 - q0 * q2;
  halfvy = q0 * q1 + q2 * q3;
  halfvz = q0 * q0 - 0.5f + q3 * q3;
```

```
            // Το σφάλμα είναι το άθροισμα των γινομένων μεταξύ της εκτιμώμενης και της μετρήσιμης κατεύθυνσης της βαρύτητας
            halfex = (ay * halfvz - az * halfvy);
            halfey = (az * halfvx - ax * halfvz);
            halfez = (ax * halfvy - ay * halfvx);

            // Υπολογισμός και εφαρμογή ανάδρασης ολοκλήρωσης
            integralFBx += twoKi * halfex * (1.0f / sampleFreq);
            integralFBy += twoKi * halfey * (1.0f / sampleFreq);
            integralFBz += twoKi * halfez * (1.0f / sampleFreq);
            gx += integralFBx;
            gy += integralFBy;
            gz += integralFBz;

            // Εφαρμογή αναλογικής ανάδρασης
            gx += twoKp * halfex;
            gy += twoKp * halfey;
            gz += twoKp * halfez;

            // Ρυθμός ολοκλήρωσης της μεταβολής των quaternion
            gx *= (0.5f * (1.0f / sampleFreq));
            gy *= (0.5f * (1.0f / sampleFreq));
            gz *= (0.5f * (1.0f / sampleFreq));
            qa = q0;
            qb = q1;
            qc = q2;
            q0 += (-qb * gx - qc * gy - q3 * gz);
            q1 += (qa * gx + qc * gz - q3 * gy);
            q2 += (qa * gy - qb * gz + q3 * gx);
            q3 += (qa * gz + qb * gy - qc * gx);

            // Κανονικοποίηση των quaternion
            recipNorm = invSqrt(q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3);
            q0 *= recipNorm;
            q1 *= recipNorm;
            q2 *= recipNorm;
            q3 *= recipNorm;
        }

        // Fast inverse square-root
        float invSqrt(float x) {
            float halfx = 0.5f * x;
            float y = x;
            long i = *(long*)&y;
            i = 0x5f3759df - (i>>1);
            y = *(float*)&i;
            y = y * (1.5f - (halfx * y * y));
            return y;
        }
```

```
// Η συνάρτηση που καλούμε από το κυρίως πρόγραμμα για λήψη των γωνιών από το IMU
void FreeIMU::getYawPitchRoll(float * ypr) {
  getYawPitchRollRad(ypr);
  arr3_rad_to_deg(ypr);
}

// Εξαγωγή των τιμών των γωνιών yaw, pitch, roll σε ακτίνια από τα Quaternion
void FreeIMU::getYawPitchRollRad(float * ypr) {
  float q4; // quaternion
  float gx, gy, gz; // estimated gravity direction
  getQ(q);

  gx = 2 * (q1*q3 - q0*q2);
  gy = 2 * (q0*q1 + q2*q3);
  gz = q0*q0 - q1*q1 - q2*q2 + q3*q3;

  ypr[0] = atan2(2 * q1 * q2 - 2 * q0 * q3, 2 * q0*q0 + 2 * q1 * q1 - 1);
  ypr[1] = atan(gx / sqrt(gy*gy + gz*gz));
  ypr[2] = atan(gy / sqrt(gx*gx + gz*gz));
}

// Μετατροπή από ακτίνια σε μοίρες
void arr3_rad_to_deg(float * arr) {
  arr[0] *= 180/M_PI;
  arr[1] *= 180/M_PI;
  arr[2] *= 180/M_PI;
}
```

Κεφάλαιο 3 - Κατασκευή και υλοποίηση

Μικροελεγκτές - Arduino Mega 1280 και Arduino Pro Mini

Εικόνα 3.1: Οι Arduino Mega και Pro Mini

Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring, η οποία είναι μια διαφοροποιημένη έκδοση της C++.

Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες· το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

Υλικό

Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για τη διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό αντηχητή σε κάποιες παραλλαγές). Ο

μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής

Γενικά όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων RS-232 και TTL. Οι πλακέτες Arduino που κυκλοφορούν σήμερα στην αγορά, προγραμματίζονται μέσω USB, εφαρμόζοντας ένα chip προσαρμογέα USB-to-serial όπως το FTDI FT232. Κάποιες παραλλαγές, όπως το Arduino mini και το ανεπίσημο Boarduino, χρησιμοποιούν προσαρμογέα USB-to-serial σε μορφή πλακέτας ή καλωδίου.

Η πλακέτα του Arduino έχει εκτεθειμένες τις περισσότερες επαφές εισόδου/εξόδου για χρήση με άλλα κυκλώματα. Το Diecimila, για παράδειγμα, παρέχει 14 ψηφιακές επαφές εισόδου/εξόδου, από τις οποίες οι 6 μπορούν να παράγουν σήματα PWM, και 6 αναλογικές εισόδους. Αυτές οι επαφές είναι διαθέσιμες στην κορυφή της πλακέτας μέσω θηλυκών συνδέσεων μεγέθους 0,1 ιντσών. Διάφορες plug-in πλακέτες εφαρμογών γνωστές σαν “shields” είναι, επίσης, διαθέσιμες στο εμπόριο.

Οι συμβατές με το Arduino πλακέτες Barebones και Boarduino διαθέτουν αρσενικές επαφές στην κάτω πλευρά της πλακέτας για να μπορούν να συνδεθούν με πλακέτες που δεν χρειάζονται συγκολλήσεις.

Χρήση στο όχημά μας

Στο όχημα της παρούσας εργασίας χρησιμοποιούνται δύο μικροελεγκτές Arduino. Ο πρώτος είναι ένας *Arduino Pro Mini* ο οποίος μέσω του πρωτοκόλλου I^2C δέχεται τις τιμές των αισθητηρίων και με χρήση φίλτρου βασισμένου σε DCM, υλοποιημένου με quaternions, μετατρέπει αυτές τις τιμές στις γωνίες κλίσης των 2 οριζόντιων αξόνων του tricopter και τις αποστέλλει μέσω σειριακής θύρας στον arduino mega. Ο δεύτερος είναι ένας Arduino Mega 1280, που δεχόμενος τις γωνίες κλίσης, εφαρμόζει κώδικα για τρεις ελεγκτές PID, έναν για κάθε άξονα και στέλνει τις κατάλληλες τιμές στους τρεις brushless κινητήρες του οχήματος, καθώς και τον σερβοκινητήρα που καθορίζει την γωνία κλίσης του M1. Επίσης είναι υπεύθυνος για την αναγνώριση των εντολών από την τηλεκατεύθυνση καθώς και για την αποστολή της τηλεμετρίας στον υπολογιστή μέσω της σειριακής θύρας.

Ο λόγος που χρησιμοποιήθηκαν δύο μικροελεγκτές είναι η έλλειψη ειδικών υποδοχών στον arduino mega για την ανάγνωση των τιμών των αισθητηρίων καθώς οι συγκεκριμένοι υποδοχείς είναι υπεύθυνοι για τα hardware interrupts που χρησιμοποιούνται για την ανάγνωση των τιμών από την τηλεκατεύθυνση.

Αισθητήρια

MPU-6050 και Razor 9DOF

Εικόνα 3.2: Το MPU-6050 της εταρίας Drotek

Εικόνα 3.3: Το Razor 9DOF της εταιρίας Sparkfun

Η πλακέτα που έχει χρησιμοποιηθεί στην τελική εκδοχή του οχήματος είναι της εταιρίας Drotek και χρησιμοποιεί το chip MPU-6050. Πρόκειται για μια πλακέτα η οποία ενσωματώνει ψηφιακό γυροσκόπιο τριών αξόνων, ψηφιακό επιταχυνσιόμετρο τριών αξόνων καθώς και μικροελεγκτή, όλα αυτά σε ένα chip κλειστής αρχιτεκτονικής. Ο ενσωματωμένος μικροελεγκτής λαμβάνει τις μετρήσεις από τα αισθητήρια και εφαρμόζει ψηφιακά φίλτρο DCM για να δώσει την ακριβή γωνία κλίσης σε κάθε άξονα. Το αισθητήριο επικοινωνεί με τον δεύτερο μικροελεγκτή του tricopter με το πρωτόκολλο I2C στέλνοντας τιμές τις οποίες αναλύει ο μικροελεγκτής και μας δίνει τις γωνίες κλίσης των αξόνων X και Y, καθώς και την τιμή του γυροσκοπίου για τον άξονα Z.

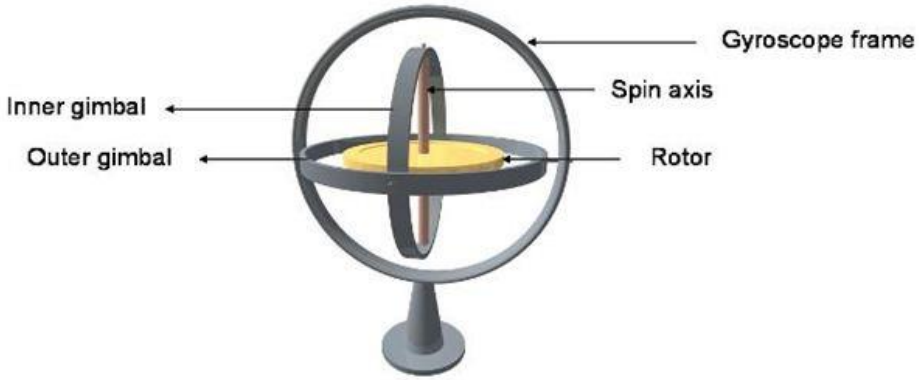
Στα αρχικά στάδια κατασκευής είχε χρησιμοποιηθεί η πλακέτα Razor 9DOF από την εταιρία Sparkfun. Η συγκεκριμένη πλακέτα περιλάμβανε ξεχωριστά επιταχυνσιόμετρο τριών αξόνων, γυροσκόπιο δύο αξόνων για την εμπρός/πίσω και πλάγια κλίση, γυροσκόπιο ενός

άξονα για την οριζόντια περιστροφή καθώς και μαγνητόμετρο τριών αξόνων. Ακόμη, περιείχε έναν μικροελεγκτή ATMEGA 328 ο οποίος λάμβανε τις τιμές από όλα τα αισθητήρια και με φίλτρο DCM υπολόγιζε τις γωνίες των αξόνων X και Y. Έπειτα, έστελνε μέσω της σειριακής θύρας τις τιμές των γωνιών των δύο αυτών αξόνων και το γυροσκόπιο για την οριζόντια περιστροφή, καθώς και την μαγνητική κατεύθυνση του οχήματος. Η πλακέτα Razor 9DOF χρειάστηκε να αντικατασταθεί, όταν μετά από πτώση του οχήματος υπέστη σοβαρή βλάβη που την κατέστησε μη λειτουργική.

Γυροσκόπια

Τα γυροσκόπια είναι συσκευές μέτρησης της γωνιακής ταχύτητας. Η συνήθης μονάδα είναι οι μοίρες ανά δευτερόλεπτο.

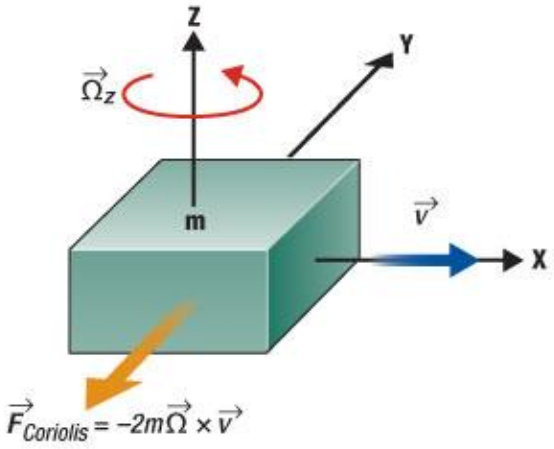
Ιστορικά, τα γυροσκόπια ξεκίνησαν σαν παιχνίδι, αλλά μέσα στην πορεία των χρόνων επιστήμονες άρχισαν να ασχολούνται με αυτό το αντικείμενο, ανακαλύπτοντας τις δυνατότητες που προσέφερε, εξελίσσοντάς το σε εργαλείο ναυσιπλοΐας, αεροπορίας και γενικά πολιτικής και στρατιωτικής χρήσης για διατήρηση πορείας και ευστάθειας.



Εικόνα 3.4: Γυροσκόπιο ελεύθερης περιστροφής

Εξέλιξη και σύγχρονη μορφή των γυροσκοπίων

Τα όργανα που χρησιμοποιούσαν αρχικά γυροσκόπια ήταν ογκώδη και βαριά. Οι βιομηχανίες όμως, σιγά σιγά κατασκεύαζαν όλο και μικρότερες συσκευές που ζύγιζαν μόλις λίγα γραμμάρια. Στη σημερινή εποχή τα γυροσκόπια συναντιόνται σε μορφή MEMS – Microelectromechanical Systems. Υπάρχει πληθώρα εταιριών που κατασκευάζουν και διανέμουν πλακέτες με MEMS, άλλες μόνο με γυροσκόπια και άλλες τύπου IMU ή AHRS, περιλαμβάνοντας επιταχυνσιόμετρα και μαγνητόμετρα, καθώς και on-board μικροελεγκτές. Στις πλακέτες αυτές μπορεί να συναντήσει κανείς MEMS με γυροσκόπια ενός, δύο ή και τριών αξόνων, με τα τελευταία να κερδίζουν έδαφος προτίμησης στην αγορά, λόγω της προσφοράς συνολικά πιο ολοκληρωμένων δυνατοτήτων σε πολύ χαμηλό βάρος αλλά και κόστος.



Εικόνα 3.5: Γυροσκόπιο MEMS

Τα χαρακτηριστικά των γυροσκοπίων

Σημαντικό χαρακτηριστικό ενός γυροσκοπίου και κριτήριο επιλογής του για την εφαρμογή για την οποία προορίζεται, είναι οι επιλογές που προσφέρει όσων αφορά την ανάλυση του. Αυτή εκφράζεται σε μοίρες ή ακτίνια ανά δευτερόλεπτο, που αντιστοιχεί στο πόσο γρήγορη μεταβολή μπορεί το αισθητήριο να αντιληφθεί. Συνήθως, για χρήση σε multicopters, οι προτιμώμενες τιμές είναι μεταξύ ±100˚/s ως ±300˚/s. Το MPU-6050 που χρησιμοποιείται στο tricopter έχει τέσσερις ρυθμίσεις ευαισθησίας που μπορεί να επιλέξει ο χρήστης: ±250˚/s, ±500˚/s, ±1000˚/s και ±2000˚/s. Το όχημα της παρούσας εργασίας χρησιμοποιεί την πρώτη επιλογή ευαισθησίας του αισθητηρίου.

Τα μειονεκτήματα των γυροσκοπίων

Το κύριο μειονέκτημα των γυροσκοπίων, είναι το φαινόμενο της ολίσθησης ή drift. Το φαινόμενο αυτό, αποτρέπει την αποκλειστική χρήση γυροσκοπίων για την ευστάθεια του οχήματος, καθώς μετά από κάποιο χρονικό διάστημα, η απόκλιση των μετρήσεων θα μεταβάλλει το αρχικά θεωρημένο σημείο μηδέν, με αποτέλεσμα να μην υπάρχει το απαραίτητο set point για τη διατήρηση της ευστάθειας. Αυτό φαίνεται σύμφωνα με εργαστηριακές μετρήσεις που πραγματοποιούνται και ενώ η τιμή εξόδου του αισθητηρίου θα έπρεπε να παραμένει μηδενική κατά την κατάσταση ακινησίας, παρατηρείται αργή, σταδιακή μεταβολή στο χρόνο. Συμπληρώνοντας τις μετρήσεις με αυτές ενός επιταχυνσιόμετρου και τα κατάλληλα φίλτρα, μπορούμε να μειώσουμε κατά πολύ αυτή την ολίσθηση.

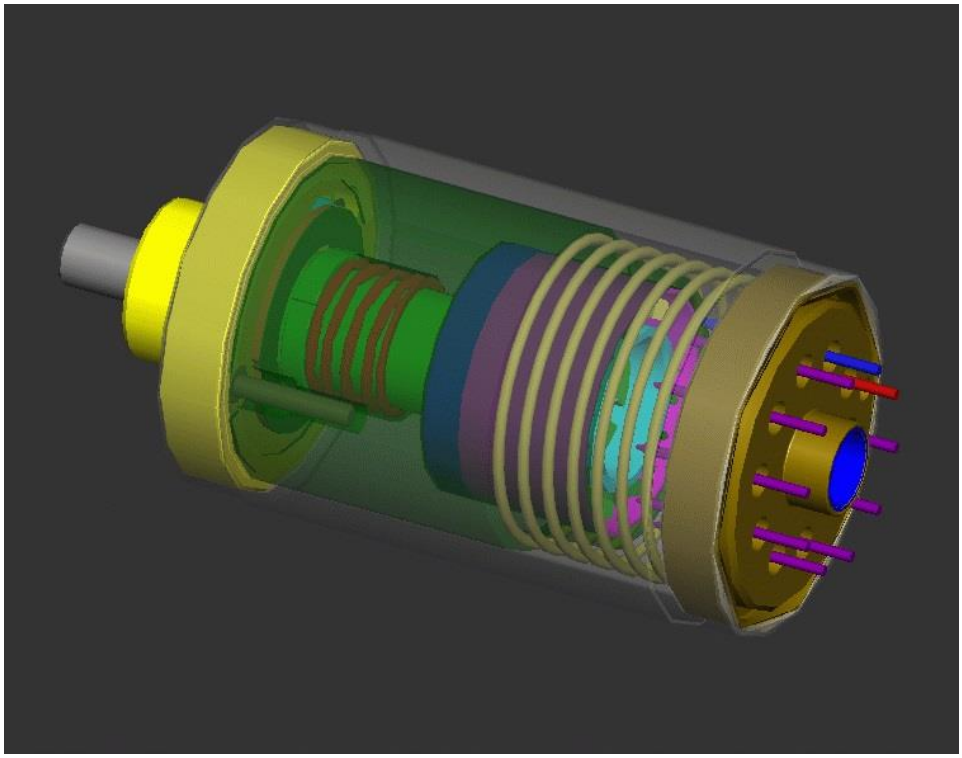
Ένα ακόμη μειονέκτημα που παρουσιάζουν τα γυροσκόπια είναι η ευαισθησία τους στις δονήσεις. Κάτι τέτοιο μπορεί να αποτελέσει πρόβλημα σε ένα tricopter, καθώς τα τρία μοτέρ,

συνυπολογίζοντας την αλλαγή κλίσης του ενός, μπορούν να δημιουργήσουν δονήσεις στο σώμα του οχήματος. Για μείωση του θορύβου αυτού, το αισθητήριο MPU-6050 έχει τοποθετηθεί ανάμεσα σε μαλακό αντικραδασμικό υλικό, που το προστατεύει από μεγάλο μέρος των δονήσεων.

Επιταχυνσιόμετρα

Το Επιταχυνσιόμετρο είναι ένα αισθητήριο που μετρά την επιτάχυνση, δηλαδή την μεταβολή της ταχύτητας στον χρόνο και είναι διανυσματική τιμή (έχει κατεύθυνση και μέγεθος).

Τα Επιταχυνσιόμετρα μετρούν σε μονάδες g, ένα g ισούται με την επιτάχυνση της βαρύτητας δηλαδή 9.81m/s². Τα επιταχυνσιόμετρα έχουν εξελιχθεί από απλούς σωλήνες με μια φούσκα αέρα σε ολοκληρωμένα κυκλώματα. Τα επιταχυνσιόμετρα μπορούν να μετρήσουν: δονήσεις, κρούσεις, κλίση και τη κίνηση ενός αντικειμένου.



Εικόνα 3.6: Αναλογικό επιταχυνσιόμετρο σχεδιασμένο από στην Sandia National Laboratiories

Τύποι Επιταχυνσιόμετρων:

Υπάρχουν διάφοροι τύποι επιταχυνσιομέτρων. Αυτό που διαφοροποιεί τα είδη είναι το αισθητήριο στοιχείο και οι αρχές λειτουργίας τους .

Τα χωρητικά επιταχυνσιόμετρα (capacitive) μετρούν μια αλλαγή στην ηλεκτρική χωρητικότητα , σε σχέση με την επιτάχυνση. Το επιταχυνσιόμετρο αυτό ανιχνεύει την μεταβολή της χωρητικότητας μεταξύ μιας στατικής κατάστασης και τη δυναμική θέση.

Τα Πιεζοηλεκτρικά επιταχυνσιόμετρα χρησιμοποιούν υλικά όπως κρυστάλλους , τα οποία παράγουν ηλεκτρική τάση από μια εφαρμοζόμενη δύναμη. Αυτό είναι γνωστό ως πιεζοηλεκτρικό φαινόμενο.

Τα Πιεζοαντιστασιακά επιαχυνσιόμετρα (strain gauge) λειτουργούν μετρώντας τη διαφορά της ηλεκτρικής αντίστασης ενός υλικού όταν αυτό παραμορφώνεται ελαφρώς όταν ασκείται κάποια δύναμη.

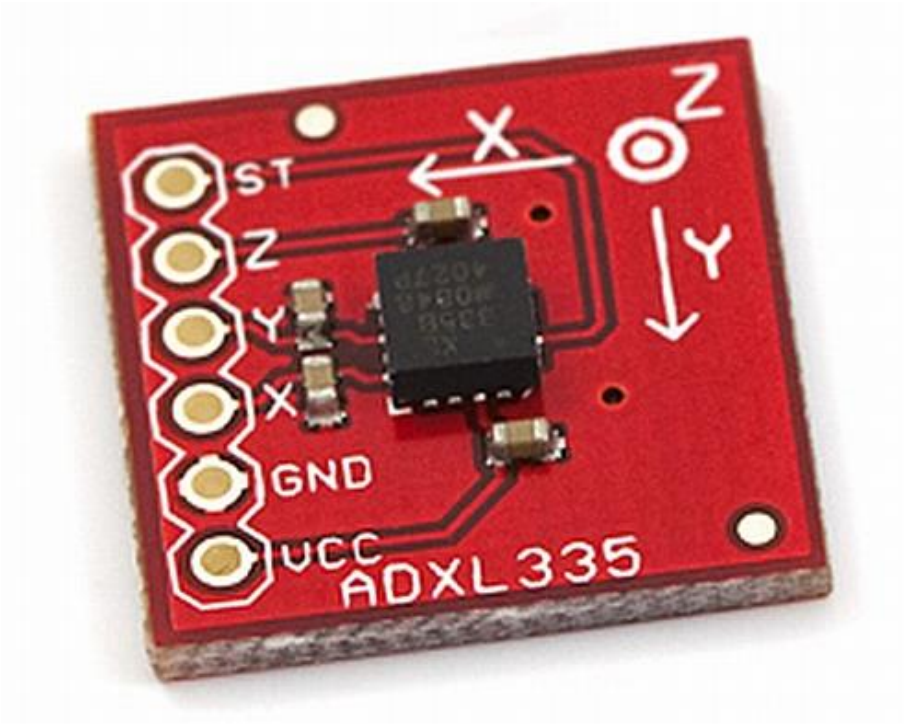
Τα επιταχυνσιόμετρα Hall Effect μετρούν αλλαγές τάσης προερχόμενες από την αλλαγή του μαγνητικού πεδίου γύρω από το επιταχυνσιόμετρο.

τα Μαγνητοαντιστασιακά επιταχυνσιόμετρα λειτουργούν μετρώντας αλλαγές στην αντίσταση λόγω μαγνητικού πεδίου. Είναι αρκετά όμοια σε δομή και λειτουργία με τα επιταχυνσιόμετρα Hall Effect, μόνο που μετρούν αλλαγές στην αντίσταση αντί για τάση.

Τα επιταχυνσιόμετρα μεταφοράς θερμότητας, μετρούν αλλαγές στην μεταφορά θερμοκρασίας λόγω της επιτάχυνσης. Μια πηγή θερμότητας βρίσκεται στο κέντρο μιας κοιλότητας και θερμοαντιστάσεις είναι μοιρασμένες στα τοιχώματα. Σε κατάσταση μηδενικής επιτάχυνσης η διαβάθμιση της θερμότητας θα είναι συμμετρική. Με οποιαδήποτε επιτάχυνση η διαβάθμιση της θερμότητας γίνεται ασύμμετρη λόγω της μεταγωγικής μεταφοράς θερμότητας.

Επιταχυνσιόμετρα βασισμένα σε MEMS (Micro-Electro Mechanical System):

Η τεχνολογία MEMS βασίζεται σε μια σειρά από εργαλεία και μεθοδολογίες, οι οποίες χρησιμοποιούνται για να σχηματίσουν μικρές δομές με διαστάσεις στην κλίμακα μικρομέτρου (ένα εκατομμυριοστό του μέτρου). Αυτή η τεχνολογία χρησιμοποιείται πλέον για την κατασκευή υπερσύγχρονων επιταχυνσιομέτρων.



Εικόνα 3.7: Το τύπου MEMS επιταχυνσιόμετρο τριών αξόνων ADXL335 της εταιρίας Sparkfun

Χρήσεις των επιταχυνσιομέτρων:

Από τη βιομηχανία ως την εκπαίδευση, τα επιταχυνσιόμετρα έχουν πολλές εφαρμογές. Αυτές οι εφαρμογές κυμαίνονται από την ενεργοποίηση του αερόσακου ως τη παρακολούθηση των πυρηνικών αντιδραστήρων. Υπάρχει μεγάλος αριθμός από πρακτικές εφαρμογές για επιταχυνσιόμετρα. Επιταχυνσιόμετρα χρησιμοποιούνται για τη μέτρηση της στατικής επιτάχυνσης (βαρύτητα), τη κλίση ενός αντικειμένου, τη δυναμική επιτάχυνση, χτυπήματα σε ένα αντικείμενο, την ταχύτητα, τον προσανατολισμό και τη δόνηση ενός αντικειμένου. Τα επιταχυνσιόμετρα γίνονται όλο και περισσότερο πανταχού παρόντα. Βρίσκονται πλέον σε κινητά τηλέφωνα, υπολογιστές, ακόμα και σε και πλυντήρια.

Επιλέγοντας επιταχυνσιόμετρο:

Όταν επιλέγει κάποιος επιταχυνσιόμετρο για κάποια εφαρμογή, οι πρώτοι παράγοντες που πρέπει να λάβει υπόψιν του είναι:

1. Δυναμικό εύρος: είναι το +/- μέγιστο εύρος επιτάχυνσης που μπορεί να μετρήσει πριν παραμορφώσει ή κλειδώσει το παραγόμενο σήμα. Το δυναμικό εύρος συνήθως μετριέται σε g.

2. Ευαισθησία: Η ευαισθησία είναι ο συντελεστής κλίμακας ενός αισθητήρα ή συστήματος, το οποίο μετράται από την άποψη της αλλαγής στο σήμα εξόδου ανά αλλαγή στην μετρήσιμη είσοδο. Η ευαισθησία αναφέρεται στην ικανότητα του επιταχυνσιόμετρου για την ανίχνευση κίνησης. Η ευαισθησία του επιταχυνσιόμετρου συνήθως καθορίζεται σε millivolt ανά g (mV / g).

3. Απόκριση συχνότητας: Απόκριση συχνότητας είναι το φάσμα συχνοτήτων για το οποίο ο αισθητήρας θα ανιχνεύσει κίνηση και θα αναφέρει μια πραγματική έξοδο. Η απόκριση συχνότητας συνήθως ορίζεται ως ένα εύρος που μετριέται σε Hertz (Hz).

4. Ευαίσθητος άξονας: Τα επιταχυνσιόμετρα έχουν σχεδιαστεί για την ανίχνευση επιτάχυνσης σε σχέση με έναν άξονα. Επιταχυνσιόμετρα μονού άξονα μπορούν να ανιχνεύσουν μόνο εισόδους στο ίδιο επίπεδο. Επιταχυνσιόμετρα τριών αξόνων μπορούν να ανιχνεύσουν τις εισόδους σε οποιοδήποτε επίπεδο και είναι απαραίτητα για τις περισσότερες εφαρμογές.

5. Μέγεθος και Μάζα: Το μέγεθος και η μάζα ενός επιταχυνσιόμετρου μπορεί να αλλάξει τα χαρακτηριστικά του αντικειμένου που εξετάζεται. Η μάζα από τα επιταχυνσιόμετρα θα πρέπει να είναι σημαντικά μικρότερη από τη μάζα του συστήματος που μετράται.

Το μειονέκτημα που παρουσιάζουν τα επιταχυνσιόμετρα, όταν βρίσκουν εφαρμογή σε ιπτάμενα οχήματα, όπως το tricopter της παρούσας εργασίας, είναι πως κατά την οριζόντια μετακίνηση του οχήματος, το επιταχυνσιόμετρο του ανάλογου άξονα θα παρουσιάσει μη μηδενικές τιμές, όπως ακριβώς και όταν υπάρξει κλίση προς την ίδια μεριά. Το ίδιο το επιταχυνσιόμετρο σαν αισθητήριο δε μπορεί να γνωρίζει τι από τα δύο συμβαίνει, η χρήση των γυροσκοπίων βοηθά σε αυτό, κι έτσι το ένα αισθητήριο διορθώνει τις αδυναμίες του άλλου.

Κινητήρες

Εικόνα 3.8: Οι brushless κινητήρες
KDA20-22L

Το όχημα διαθέτει τρεις τριφασικούς κινητήρες τύπου DC Brushless outrunner οι οποίοι δίνουν κίνηση σε τρεις έλικες. Ο κινητήρας τύπου DC Brushless είναι Σύγχρονος Κινητήρας Μόνιμου Μαγνήτη, και η οδήγησή του γίνεται ηλεκτρονικά με τη χρήση τριφασικού αντιστροφέα ισχύος (ηλεκτρονικός ελεγκτής ταχύτητας ή ESC). Αυτοί οι κινητήρες έχουν την καλύτερη απόδοση καθώς ελαχιστοποιούν τις απώλειες χάριν στην έλλειψη της τριβής· τα γνωστά "καρβουνάκια" που έχουν οι κλασικοί DC κινητήρες, δεν υπάρχουν στους DC Brushless κινητήρες.

Η επιλογή των συγκεκριμένων κινητήρων βασίστηκε σε συγκριτικές δοκιμές μεταξύ διαφόρων κινητήρων στις οποίες ξεχώρισαν οι επιλεχθέντες.

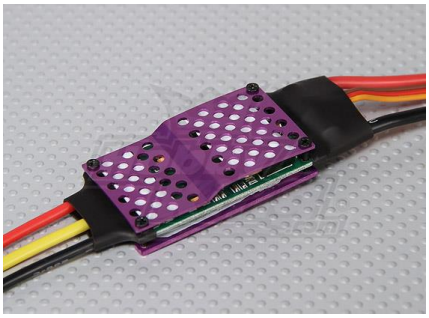
	EPP1045 Prop	Kv	Source	Price	Wt	Max T	Watts/oz	Watts/oz	WQF	Vib-O	Vib-R
Rank				$	(oz)	(oz)	@maxT	@16 oz - Wt		(Low Freq)	(Hi Freq)
1	KDA20-22L	924	Hobby City	17	2.5	32	5.0	4.0	62%	1.9%	2.6%
2	DT-750	750	Hobby City	11	3.3	31	4.4	4.3	70%	3.1%	2.7%
3	Turnigy 2213/22	924	Hobby City	19	2.6	30	4.7	4.2	60%	2.9%	
4	Turnigy 2217/20	860	Hobby City	20	3.0	31	4.7	4.5	58%	3.3%	2.7%
5	DT-700	700	Hobby City	11	3.3	29	4.3	4.5	70%		4.8%
6	KA20-20L	1050	Planeinsanerc	25	2.7	36	5.4	4.3	57%	2.8%	
7	HC2812-0650	650	Maxx Products	52	2.7	20	3.5	3.9	62%	2.3%	4.3%
8	2210N	1000	Hobby City	6	1.9	32	5.2	4.3	61%	4.8%	
9	2410-09 Red Base	840	QuadroUFO	10	2.6	23	4.0	4.0	57%	4.6%	
10	2410-09 Gold Base	840	Nitro Planes	10	2.6	23	4.3	4.2	56%	3.8%	7.9%
11	2410-09 Open Base	840	Hobby City	6	2.5	23	4.4	4.2	54%	8.5%	6.3%
12	2410-09Y Blue Base	1100	RobotBirds	7	2.6	20	4.8	4.8	45%	3.8%	6.0%
13	2409-18 Open Base	1000	Hobby City	10	2.5	29	5.5	5.5	37%	4.3%	
14	C2409-1200	1200	Hobby City	6	2.4	32	6.4	5.1	56%		6.5%
15	FC2822	1200	Hobby City	6	1.9	28	6.8	5.8	55%		4.1%
16	D2730-1300	1300	Hobby City	8	1.3	17.6	6.5	6.1	45%		5.1%

Εικόνα 3.9: Αναλυτικός πίνακας των δυνατοτήτων διαφόρων brushless κινητήρων

Ηλεκτρονικός Ελεγκτής Ταχύτητας

Για κάθε κινητήρα του οχήματός μας χρειαζόμαστε ένα κύκλωμα που θα μετατρέπει τα 12 Volt συνεχούς ρεύματος σε τριφασικό εναλλασσόμενο κατάλληλης συχνότητας για να γυρίσουν τα μοτέρ στις στροφές που θέλουμε. Αυτά τα κυκλώματα δέχονται έναν τετραγωνικό παλμό τύπου servo από τον μικροελεγκτή μας, και στην έξοδό τους βγάζουν ακριβώς την κατάλληλη τριφασική τάση που απαιτείται ανά περίπτωση. Στην περίπτωσή μας χρησιμοποιούμε 3 ελεγκτές, αντοχής μέχρι 20Α, έναν για κάθε brushless κινητήρα.

Στην πρώτη φάση σχεδιασμού του Tricopter επιλέχθηκαν ESC αντοχής μέχρι 40Α, για μέγιστη αντοχή στον χρόνο, αλλά τελικώς ήταν ασύμφορα λόγω μεγέθους και βάρους.



Εικόνα 3.10: Τα πρώτα ESC του Tricopter

Εικόνα 3.11: Τα τρέχοντα ESC του tricopter

Servo

Εικόνα 3.12: Σερβοκινητήρας πολύ μικρού μεγέθους.

Ένας σερβοκινητήρας ελέγχει με ακρίβεια την κλίση του πίσω μοτέρ για την αντιστάθμιση της ροπής στρέψης αλλά και για την αλλαγή κατεύθυνσης του οχήματος.

Οι σερβοκινητήρες είναι ουσιαστικά σερβομηχανισμοί με κλειστό κύκλωμα ανάδρασης ώστε να διατηρούν με ακρίβεια την επιθυμητή γωνία. Διαθέτουν μειωτήρα στροφών έτσι ώστε να παρέχουν αρκετά μεγάλη ροπή για να διατηρούν την γωνία ακόμα και όταν ασκούνται μεγάλες σχετικά δυνάμεις.

Pulse Width Modulation

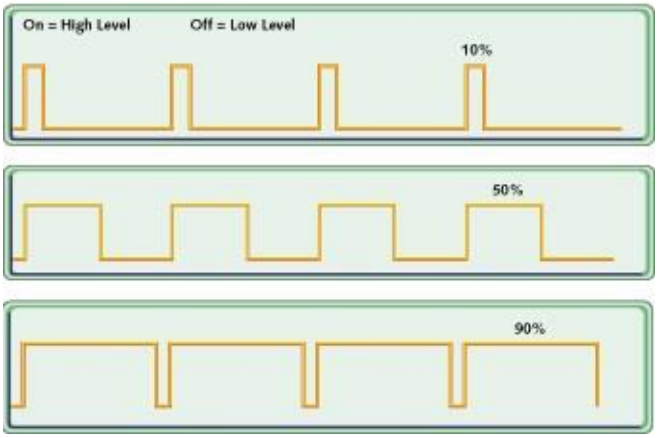
Η διαμόρφωση εύρους παλμού (PWM) είναι μια τεχνική για τον έλεγχο των αναλογικών κυκλωμάτων με τις ψηφιακές εξόδους ενός μικροελεγκτή. Το PWM χρησιμοποιείται σε μια ευρεία ποικιλία εφαρμογών, που κυμαίνονται από τις μετρήσεις και τις επικοινωνίες έως και για τον έλεγχο ισχύος.

Ελέγχοντας αναλογικά κυκλώματα με ψηφιακό τρόπο, το κόστος και η κατανάλωση ενέργειας μπορεί να μειωθεί δραστικά. Επίσης, πολλοί μικροελεγκτές περιλαμβάνουν ήδη on-chip ελεγκτές PWM, καθιστώντας εύκολη την εφαρμογή του.

Με λίγα λόγια, PWM είναι ένας τρόπος για την ψηφιακή κωδικοποίηση του επίπεδου ενός αναλογικού σήματος. Μέσω της χρήσης μετρητών υψηλής ανάλυσης, ο κύκλος λειτουργίας ενός τετραγωνικού παλμού είναι διαμορφωμένος για να κωδικοποιήσει ένα συγκεκριμένο επίπεδο αναλογικού σήματος. Το σήμα PWM εξακολουθεί να είναι ψηφιακό, διότι, σε κάθε δεδομένη χρονική στιγμή, το DC σήμα είναι είτε "1" (on) ή "0" (off).

Η πηγή τάσης ή ρεύματος παρέχεται στο αναλογικό φορτίο μέσω μιας επαναλαμβανόμενης σειράς από παλμούς "0" και "1".

Στην παρακάτω εικόνα υπάρχουν τρεις διαφορετικοί παλμοί PWM. Στο πρώτο κομμάτι της εικόνας βλέπουμε ένα παλμό με 10% duty cycle, δηλαδή ο παλμός είναι "1" για το 10% του συνολικού χρόνο του παλμού. Αντίστοιχα στα επόμενα 2 κομμάτια βλέπουμε παλμούς με 50% και 90% duty cycle



Εικόνα 3.13: Διάφορα duty cycles παλμού PWM

Η Συχνότητα του παλμού διαφέρει κατά περίπτωση, ανάλογα την εφαρμογή. Για την ένταση ακτινοβολίας μιας λάμπας πυρακτώσεως μια συχνότητα 10Hz είναι αρκετή. Τα περισσότερα φορτία χρειάζονται αρκετά μεγαλύτερες συχνότητες για τον PWM παλμό.

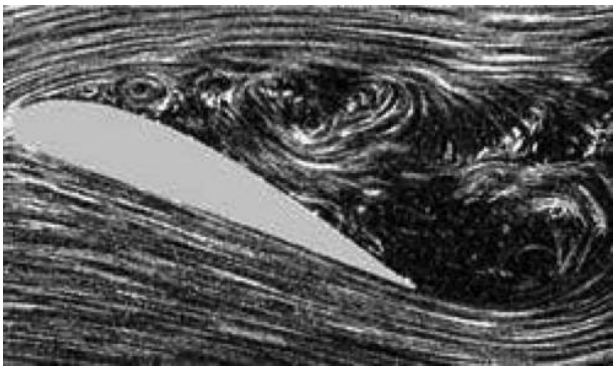
Τα servo, μοτέρ που χρησιμοποιούνται στον μοντελισμό, καθώς και τα ESC των bushless μοτέρ χρειάζονται συγκεκριμένου εύρους τιμών PWM για να λειτουργήσουν. Η συχνότητα πρέπει να είναι 50Hz, δηλαδή να έχει περίοδο 20ms και duty cycle 5-10%. Στο 5% duty cycle, βρίσκονται τέρμα αριστερά και στο 10% βρίσκονται τέρμα δεξιά, δηλαδή με παλμούς 1ms και 2ms αντίστοιχα. Ο μεγάλος κενός χρόνος στο σύνολο της περιόδου του παλμού επιτρέπει στο να υπάρξει πολύ εύκολα πολυπλεξία μέχρι και 9 σημάτων για servo σε ένα κανάλι επικοινωνίας, επιτρέποντας εξοικονόμηση χρημάτων. Σήματα που προκύπτουν από πολυπλεξία σημάτων servo ονομάζονται PPM (Pulse Position Modulation).

Έλικες

Στο όχημά μας χρησιμοποιούμε έλικες που έχουν 10 ίντσες μήκος, και κλίση 4,5 ιντσών ανά περιστροφή, 2 δεξιόστροφους και έναν αριστερόστροφο για μείωση της συνολικής ροπής στρέψης στο ⅓ σε σχέση με την περίπτωση όλοι οι έλικες να ήταν της ίδιας κατεύθυνσης περιστροφής.

Εικόνα 3.14: Οι έλικες του tricopter

Αρχικά, είχαν τοποθετηθεί στο όχημά έλικες με 8 ίντσες μήκος και κλίση 6 ιντσών ανά περιστροφή. Δυστυχώς τέτοιου είδους έλικες με μεγάλη κλίση σε σχέση με το μήκος τους δεν ταιριάζουν σε ελικόπτερα καθώς εξαιτίας αυτής δημιουργούνται δίνες αέρα που καταναλώνουν ουσιαστικά την περισσότερη ενέργεια από τον κινητήρα. Το αποτέλεσμα αυτού είναι να μην υπάρχει αρκετή ώθηση για να μπορεί να αντισταθμίσει τη δύναμη του βάρους, και έτσι το όχημα χάνει απότομα ύψος χωρίς κατά τα άλλα κάποιον προφανή λόγο για τον παρατηρητή/πιλότο. Ο σχετικός όρος στην αεροπλοΐα είναι το “stalling”.



Εικόνα 3.15: Τα δινορεύματα γύρω από τους έλικες

Τηλεκατεύθυνση

Εικόνα 3.16: Το χειρηστήριο του συστήματος τηλεκατεύθυνσης

Εικόνα 3.17: Ο δέκτης του συστήματος τηλεκατεύθυνσης

Ένα σύστημα τηλεκατεύθυνσης αποτελείται από έναν πομπό και έναν δέκτη, και είναι υπεύθυνο για την μεταφορά των εντολών του χειριστή στο όχημα. Έχει οκτώ "αναλογικά" κανάλια, δηλαδή μπορεί να στείλει μέχρι οκτώ τετραγωνικούς παλμούς σε διάφορα μοτέρ, σερβοκινητήρες κλπ. Για την κάλυψη των αναγκών του χειρισμού του οχήματος χρησιμοποιούνται τα πέντε από τα οκτώ διαθέσιμα κανάλια, ένα για να ελέγξουμε την ισχύ των κινητήρων, ένα για την κατεύθυνση, δύο για την κλίση των οριζόντιων αξόνων, και τέλος στέλνουμε και την κατάσταση ενός διακόπτη, με τον οποίο μπορούμε ανά πάσα στιγμή να απενεργοποιήσουμε τους κινητήρες σε περίπτωση ανάγκης.

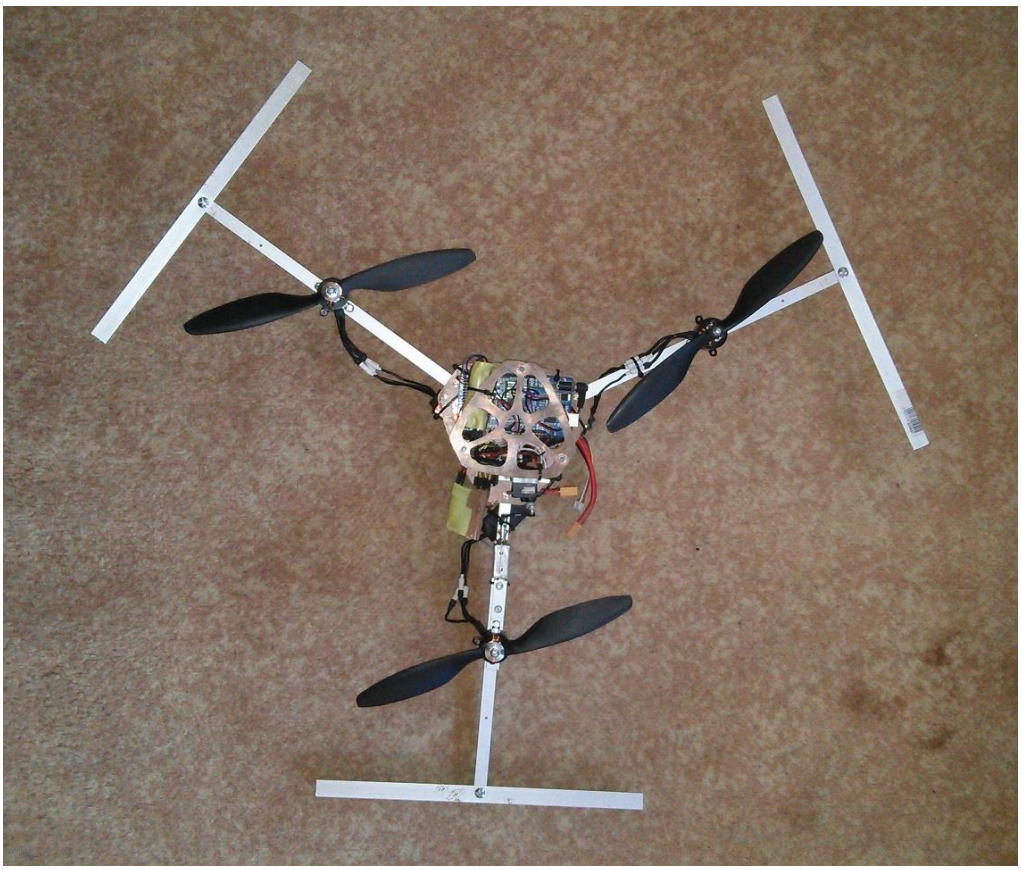
Υπάρχουν 2 τρόποι επικοινωνίας του πομπού της τηλεκατεύθυνσης με τον δέκτη, το PCM (Pulse code modulation), και το PPM (pulse position modulation).

Στο PCM η τιμή της κάθε μεταβλητής στέλνεται αυτούσια ως ψηφιακά κωδικοποιημένος αριθμός. Κάθε κατασκευαστής χρησιμοποιεί τη δική του κωδικοποίηση, με

διαφορετικό βαθμό ακρίβειας, πχ 9bit για τιμές 0-511, 10bit για τιμές μεταξύ 0-1023 ή ακόμα και 11bit για τιμές 0-2047. Το PCM συμπεριλαμβάνει ένα άθροισμα ελέγχου στο τέλος για να επαληθευτεί ότι έχουν ληφθεί όλες οι τιμές σωστά. Σε περίπτωση που υπάρχει λάθος στην μετάδοση, υπάρχει δικλείδα ασφαλείας ώστε να δίνει μια προκαθορισμένη τιμή, ή την τελευταία έγκυρη τιμή.

Στο PPM οι τιμές των μεταβλητών στέλνονται ως μεταβλητού μήκους παλμοί στη σειρά. Είναι αυτό που χρησιμοποιήθηκε στην παρούσα πτυχιακή καθώς η διαφορά κόστους ήταν σημαντική, χωρίς να υπάρχει μεγάλη διαφορά στην απόδοση, καθώς και το γεγονός ότι λόγω του χαμηλού κόστους, το συγκεκριμένο σετ τηλεκατεύθυνσης ήταν πολυδοκιμασμένο και αξιόπιστο.

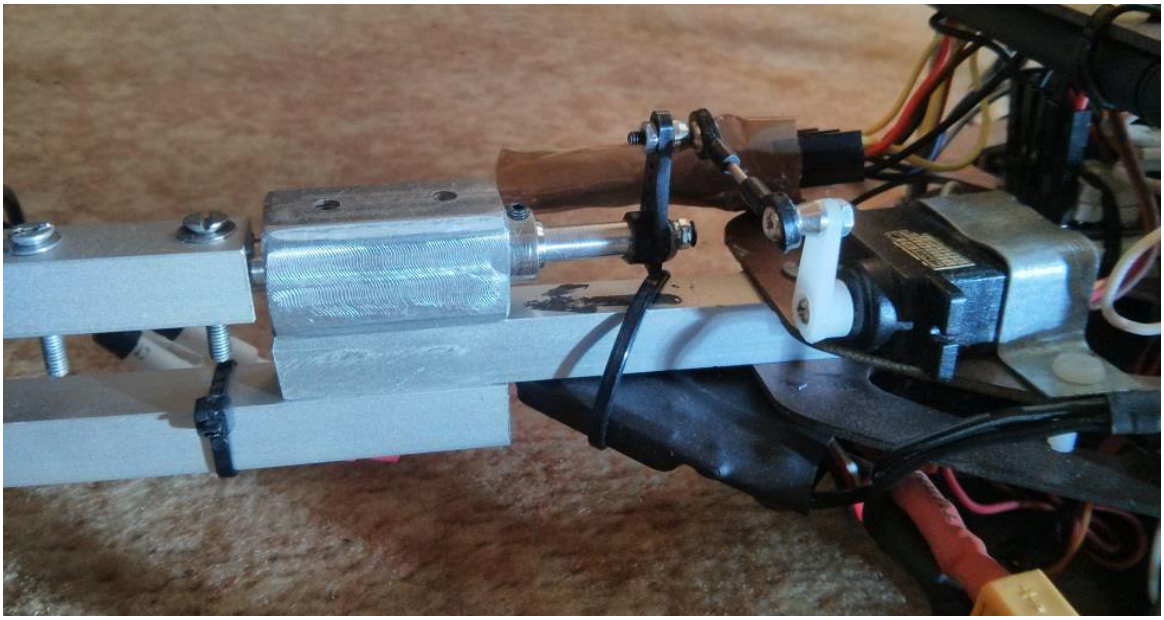
Σκελετός



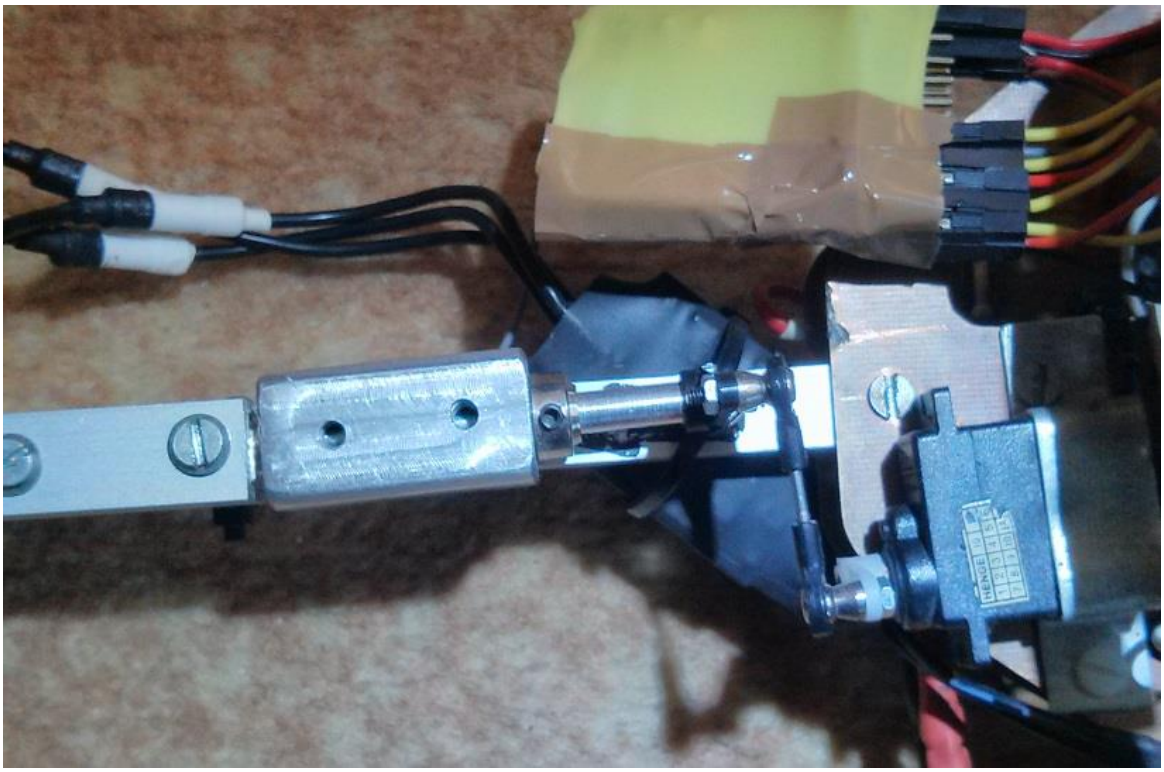
Εικόνα 3.18: Κάτοψη της τελικής μορφής του tricopter

Ο σκελετός του οχήματος είναι κατασκευασμένος από αλουμίνιο και φύλλα fiberglass επικαλυμμένα με χαλκό (PCB). Τα φύλλα είναι τοποθετημένα σε στρώσεις έτσι ώστε να μπουν ανάμεσα όλα τα ηλεκτρονικά εξαρτήματα του οχήματος και έχουν ανοιχθεί μεγάλες τρύπες σε αυτά για μείωση του βάρους αλλά και για τη βέλτιστη κατανομή των καλωδίων ανάμεσα στις διάφορες στρώσεις. Τα άκρα του οχήματος είναι φτιαγμένα από αλουμίνιο για μέγιστη αντοχή και ελαχιστοποίηση βάρους. Στις άκρες αυτών βρίσκονται τα τρία μοτέρ με τους έλικες. Στο πίσω άκρο για την αντιστάθμιση της ροπής στρέψης αλλά και για την αλλαγή κατεύθυνσης του οχήματος υπάρχει μηχανισμός με ρουλεμάν για την κλίση του αντίστοιχου μοτέρ.

Ο μηχανισμός αλλαγής κατεύθυνσης έχει κατασκευαστεί σε μηχανουργείο για μέγιστη αντοχή και ακρίβεια για το σημαντικό αυτό κομμάτι της κατασκευής. Αποτελείται από δύο ρουλεμάν χωνεμένα σε βάση αλουμινίου στα οποία συγκρατείται ο άξονας επί του οποίου στηρίζεται το οπίσθιο μοτέρ.



Εικόνα 3.19: Πλάγια όψη του μηχανισμού στρέψης του M1



Εικόνα 3.20: Κάτοψη του μηχανισμού στρέψης του M1

Μπαταρία και φορτιστής

Εικόνα 3.21: Μπαταρία τύπου LiPo τριών κελιών, τάσης 12V και χωριτικότητας 3000mAh

Εικόνα 3.22: Φορτιστής ειδικός για μπαταρίες μοντελισμού

Οι μπαταρίες LiPo είναι ένας σχετικά νέος τύπος μπαταρίας. Τα πολλά τους πλεονεκτήματα τις καθιέρωσαν και στο χώρο του τηλεκατευθυνόμενου μοντελισμού. Για τη

σωστή επιλογή του κατάλληλου τύπου αλλά και τη σωστή και ασφαλή χρήση τους απαιτούνται κάποιες γνώσεις. Η λάθος και απρόσεκτη χρήση τους μπορεί να αποβεί πολύ επικίνδυνη.

Οι μπαταρίες LiPo αποτελούνται από στοιχεία συνδεδεμένα σε σειρά ώστε να δώσουν την επιθυμητή τάση (βολτ, V). Το κάθε στοιχείο έχει ονομαστική τιμή τάσης 3,7V. Σειρές στοιχείων μπορούν να δώσουν πολλαπλάσιες τιμές τάσης του 3,7. Έτσι υπάρχουν μπαταρίες με 1cell και 3.7V, 2cell και 7.4V, 3cell και 11.1V, 4cell και 14.8V κοκ. Ένα πλήρως φορτισμένο στοιχείο LiPo έχει μέγιστη τάση 4.2V. Οπότε αντίστοιχα μία 3cell μπαταρία πλήρως φορτισμένη, θα έχει τάση (3x4,2V) 12.6V.

Προσοχή πρέπει να δοθεί κατά την εκφόρτωση της μπαταρίας, ώστε κανένα στοιχείο της να μην πέσει κάτω από τα 3V. Εάν συμβεί αυτό η μπαταρία ενδέχεται να αχρηστευθεί. Συνιστάται να χρησιμοποιούνται ρυθμιστές ταχύτητας (ESC) κατάλληλοι για τις LiPo. Αυτοί όταν η τάση πέσει κάτω από την καθορισμένη τάση, σταδιακά μειώνουν την ταχύτητα περιστροφής του μοτέρ ή σταματούν τελείως την παροχή, ανάλογα με τον τρόπο που έχουν ρυθμιστεί να λειτουργούν. Υπάρχουν ακόμη κατάλληλοι βομβητές που όταν η τάση πέσει κάτω από ένα όριο, ειδοποιούν εκπέμποντας έναν χαρακτηριστικό ήχο.

Εκτός από την τάση τους οι μπαταρίες LiPo χαρακτηρίζονται και από το ρεύμα που μπορούν να δώσουν. Αυτό μετριέται σε "C". Έτσι έχουμε μπαταρίες 15C, 20C, 25C κοκ. Για να βρεθεί πόσο ρεύμα μπορεί να αποδώσει μια μπαταρία πολλαπλασιάζουμε τον αριθμό C με τα Ampere που έχει αποθηκευμένα η μπαταρία. Έτσι μια LiPo 5000mAh και 30C μπορεί να δώσει 5x30=150A. Καθώς η μπαταρία όμως εκφορτίζεται, το ρεύμα που έχει αποθηκευμένο μειώνεται. Έτσι όταν είναι στο 50% της φόρτισης της η μπαταρία του παραδείγματος θα έχει φορτίο μόνο 2500mAh. Οπότε τότε θα μπορεί να δώσει μόνο 2,5x30=75A.

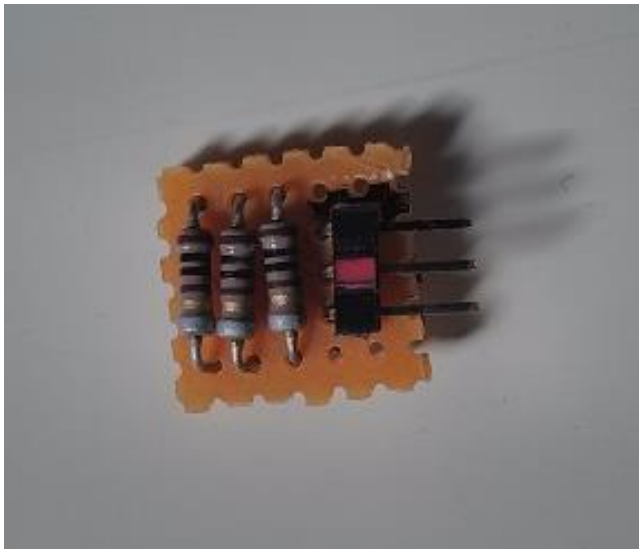
Για τη φόρτιση των LiPo υπάρχουν ειδικοί φορτιστές. Αυτοί φροντίζουν εκτός των άλλων, τα στοιχεία της μπαταρίας να έχουν το ίδιο φορτίο. Κάνουν δηλαδή το λεγόμενο *balancing*. Καλό είναι οι μπαταρίες LiPo όταν δεν πρόκειται να χρησιμοποιηθούν άμεσα να φορτίζονται μέχρι το 80% της χωρητικότητας τους. Πολλοί φορτιστές έχουν για αυτό κατάλληλο πρόγραμμα, το *storage*. Ακόμη μια τιμή που χαρακτηρίζει τις LiPo είναι το ρεύμα φόρτισης. Αυτό εκφράζεται επίσης με C και υπολογίζεται με το ίδιο τρόπο με το ρεύμα εκφόρτισης. Έτσι ρυθμίζουμε τον φορτιστή ανάλογα.

Οι LiPo ίσως να ξεγελούν με το μέγεθός τους, αλλά έχουν αποθηκευμένα μέσα τους μεγάλα ποσά ενέργειας. Έτσι εάν βραχυκυκλώσουν ή ξεπεραστούν τα όρια τους τότε μπορεί

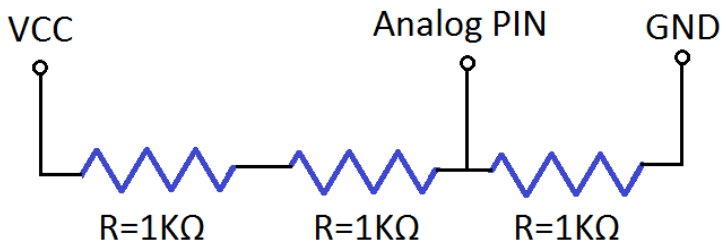
να γίνουν πολύ επικίνδυνες. Πάντα πρέπει να μεταχειρίζονται με ιδιαίτερη προσοχή και σύμφωνα τις οδηγίες του κατασκευαστή τους.

Διαιρέτης τάσης για μέτρηση μπαταρίας

Για να γνωρίζουμε την τάση της μπαταρίας του tricopter σε πραγματικό χρόνο, κατασκευάστηκε και χρησιμοποιείται ένας διαιρέτης τάσης. Ο διαρέτης τάσης κρίνεται αναγκαίος καθώς στις αναλογικές εισόδους του Arduino μπορούν να μετρηθούν ως 5V, ενώ η μπαταρία του tricopter λειτουργεί μεταξύ 9V και 12.6V. Η τιμή αυτή διαιρείται δια του 3 και έτσι, μέσω μιας αναλογικής εισόδου του arduino, γίνεται μέτρηση της τάσης της μπαταρίας. Η τιμή αυτή, αποστέλλεται μέσω της τηλεμετρίας, για εμφάνιση στην οθόνη του υπολογιστή, ώστε ο χρήστης να πληροφορείται ανά πάσα στιγμή για την κατάσταση της μπαταρίας.



Εικόνα 3.23: Ο διαιρέτης τάσης για μέτρηση της μπαταρίας



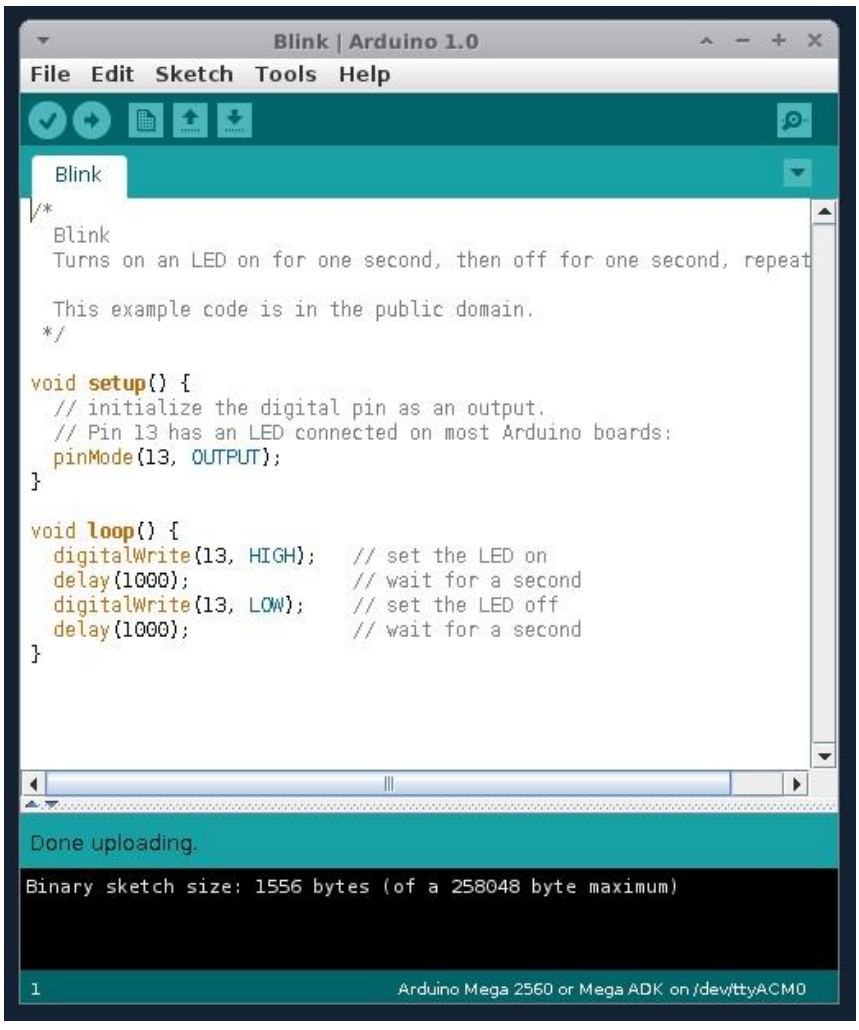
Εικόνα 3.24: Κύκλωμα διαιρέτη τάσης

Λογισμικό

Arduino

Το IDE του Arduino είναι γραμμένο σε Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστή και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα.

Το περιβάλλον ανάπτυξης είναι βασισμένο στο Processing, ένα περιβάλλον ανάπτυξης



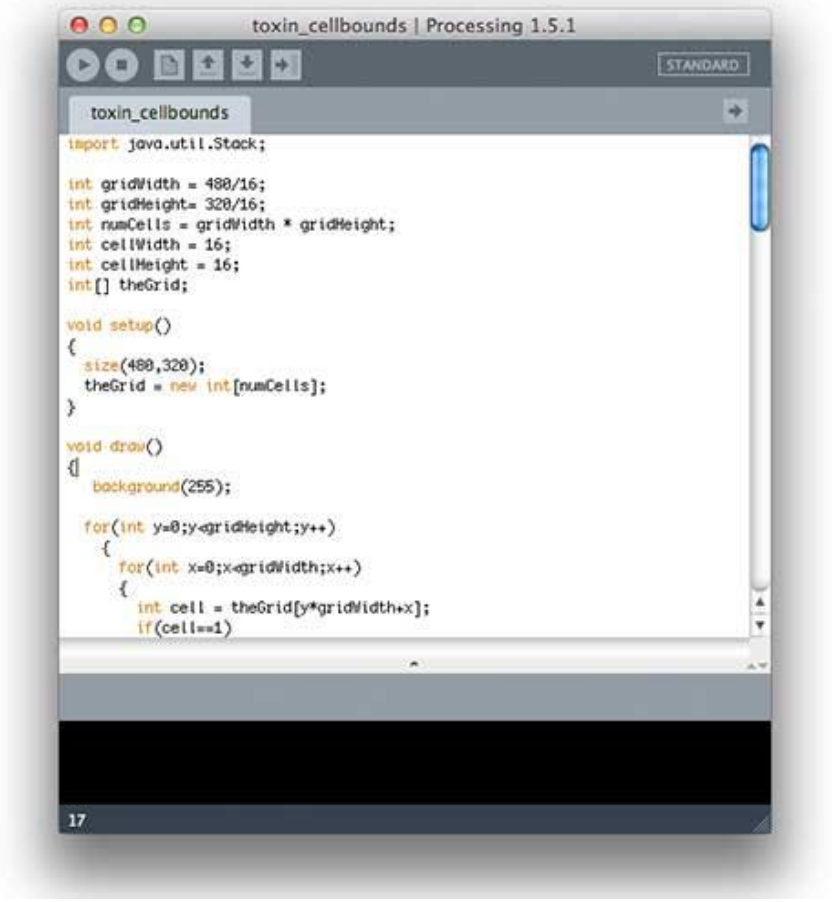
Εικόνα 3.25: Το περιβάλλον ανάπτυξης της πλατφόρμας Arduino

σχεδιασμένο να εισαγάγει στον προγραμματισμό καλλιτέχνες μη εξοικειωμένους με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring,

μια γλώσσα που μοιάζει με την C η οποία παρέχει παρόμοια λειτουργικότητα για μια πιο περιορισμένης σχεδίασης πλακέτα, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στο Processing.

Processing

Το Processing είναι μια γλώσσα προγραμματισμού ανοικτού κώδικα και παράλληλα ένα προγραμματιστικό περιβάλλον για ανθρώπους που θέλουν να προγραμματίσουν εικόνες, animation και ήχο. Το 2001 δύο απόφοιτοι του πανεπιστημίου MIT, Benjamin Fry και Casey Reas, ξεκίνησαν την ανάπτυξη της γλώσσας Processing πάνω σε Java. Παρόλο που η γλώσσα αναπτύχθηκε στη Java, το συντακτικό της είναι απλουστευμένο και το προγραμματιστικό της

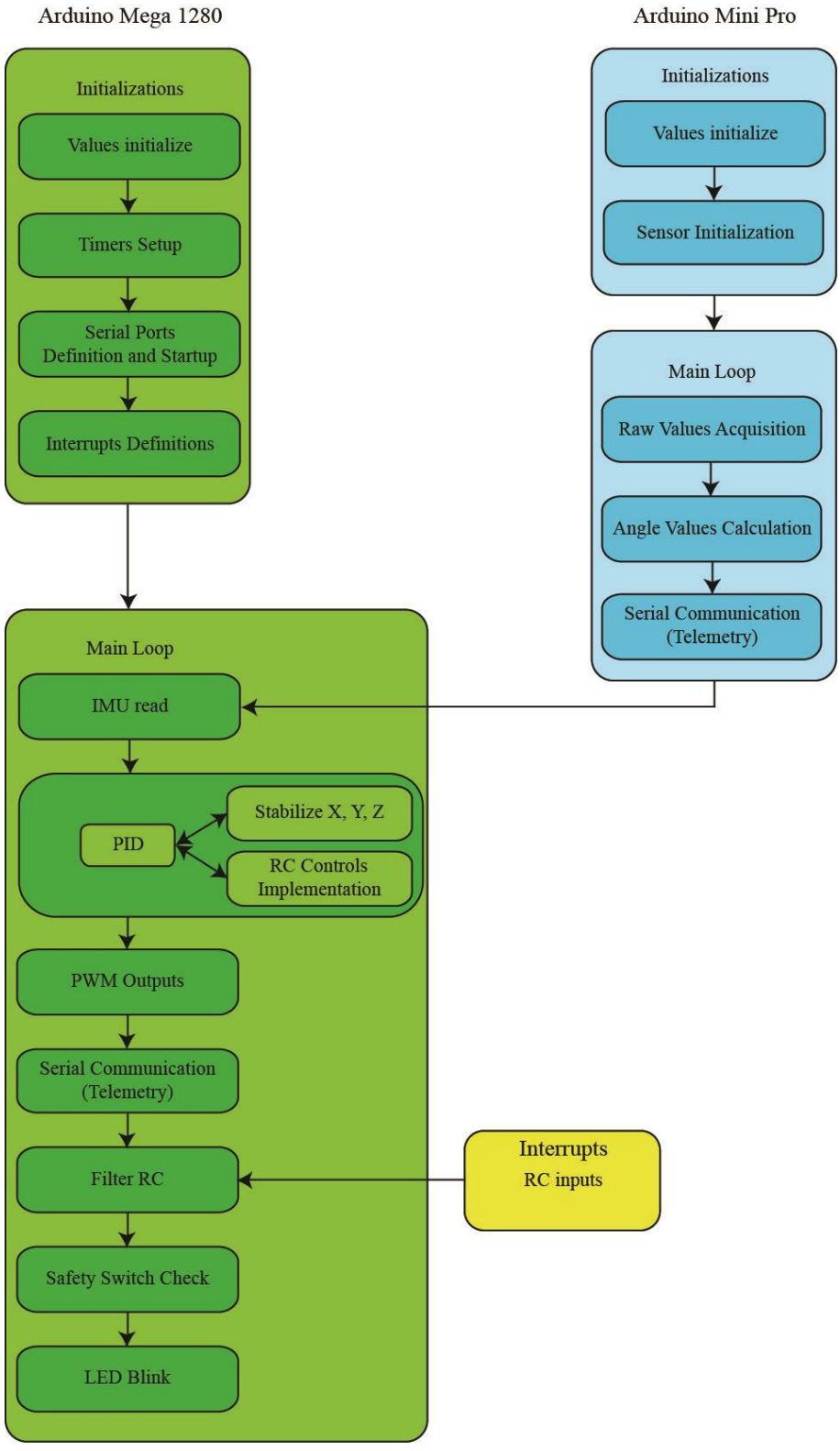


Εικόνα 3.26: Το περιβάλλον ανάπτυξης της πλατφόρμας Processing

μοντέλο βασίζεται στα γραφικά. Απώτερος σκοπός των δύο δημιουργών είναι η εκμάθηση προγραμματισμού από αρχάριους χρήστες μέσω ενός οπτικού πλαισίου καθώς και η παροχή ενός επαγγελματικού εργαλείου παραγωγής πολυμεσικών εφαρμογών.

Στην παρούσα εργασία χρησιμοποιήθηκε για να δημιουργηθεί μια εφαρμογή η οποία επιτρέπει την ύπαρξη τηλεμετρίας από το όχημά εν ώρα πτήσης και ταυτόχρονα δίνει τη δυνατότητα πιέζοντας τα κατάλληλα πλήκτρα από τον υπολογιστή να μεταβάλλονται οι τιμές στους τρεις PID ελεγκτές μας για ρύθμιση σε πραγματικό χρόνο.

Κεφάλαιο 4 – Κώδικας



Εικόνα 3.1: Block diagram του κώδικα

Η λειτουργία του προγράμματος στον Arduino Mega

Αρχικοποιήσεις

Το πρόγραμμα ξεκινά καλώντας τις απαραίτητες βιβλιοθήκες και αρχικοποιώντας τις χρησιμοποιούμενες μεταβλητές κατά το σύνολο της διαδικασίας. Εν συνεχεία δηλώνονται τα interrupts που απαιτούνται για τη λήψη των τιμών της τηλεκατεύθυνσης. Επίσης, αρχικοποιούνται οι timer3 και timer4 του μικροελεγκτή ATmega1280, που χρησιμοποιεί ο Arduino Mega, με χρήση καθαρής C για την αλλαγή των τιμών των καταχωρητών. Με αυτόν τον τρόπο επιτυγχάνεται, με την χρήση των hardware timers του μικροελεγκτή, να μην υπάρχουν διαφοροποιήσεις στους servo PWM παλμούς εξόδου από τα διάφορα interrupts του υπόλοιπου κώδικα. Οι διαφοροποιήσεις αυτές που υπήρχαν στα πρώτα στάδια της πτυχιακής εργασίας δημιουργούσαν μεγάλη αστάθεια, καθώς η λειτουργία της βιβλιοθήκης servo διακοπτόταν από τα hardware interrupts δίνοντας ενίοτε στην έξοδο προς τα μοτέρ λίγο μεγαλύτερες τιμές χωρίς κάποιον προφανή λόγο.

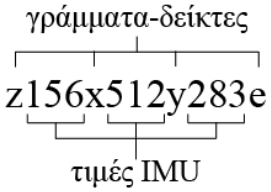
```
TCCR3A&= 0b11111100;  TCCR3B&= 0b11110111;  TCCR3B|= 0b00010000;
TCCR3B&= 0b11111010;  TCCR3B|= 0b00000010;
TCCR3A&= 0b10101011;  TCCR3A|= 0b10101000;
ICR3 = 20000;

TCCR4A&= 0b11111100;  TCCR4B&= 0b11110111;  TCCR4B|= 0b00010000;
TCCR4B&= 0b11111010;  TCCR4B|= 0b00000010;
TCCR4A&= 0b10101011;  TCCR4A|= 0b10101000;
ICR4 = 20000;
```

Κυρίως Βρόγχος

Μετά το πέρας των αρχικοποιήσεων εκκινεί ο *Κυρίως Βρόγχος* του προγράμματος. Πρόκειται για έναν ατέρμονο βρόγχο μέσα στον οποίο καλούνται όλες οι απαραίτητες συναρτήσεις για τη λειτουργία του προγράμματος. Παρακάτω περιγράφονται με τη σειρά οι λειτουργίες που επιτελούνται κατά την εκτέλεση.

Ανάγνωση τιμών από το IMU

Οι τιμές των γωνιών των *yaw*, *pitch* και *roll* μεταφέρονται στον Arduino Mega μέσω σειριακής θύρας. Για να ξεχωρίσουν οι τιμές μέσα από το συνολικό string που αποστέλλεται από τον Arduino Pro Mini, το string είναι της μορφής



Οι τιμές προτού σταλούν πολλαπλασιάζονται με τον αριθμό 100 και μετατρέπονται σε ακέραιους για την αποφυγή αποστολής υποδιαστολής. Μετά την αποστολή, διαιρούνται με το 100 και προκύπτουν οι τιμές στην αρχική τους μορφή. Έπειτα, σύμφωνα με το γράμμα-δείκτη που προπορεύεται της κάθε τιμής, γίνεται ο διαχωρισμός, η κάθε τιμή μετατρέπεται από string σε float και εισάγεται στην κατάλληλη μεταβλητή για να χρησιμοποιηθεί μέσα στο πρόγραμμα. Το τελικό γράμμα-δείκτης *e* δηλώνει το πέρας του string, ώστε το πρόγραμμα να αναγνωρίσει τις τιμές σαν πακέτο της στιγμής εκείνης.

```
while (Serial3.available() > 0){


 if (Serial3.peek()=='z'){              //an to string ksekinaei
     peekchr=1;                         //tote arxizoume na katagrafoume
 }  else if (peekchr==0) {              //alliws an den exoume arxise ina katagrafoume
     Serial3.read();                    //diavase enan xaraktira gia na proxwrisei to buffer
 }


 if ((Serial3.available() > 0)&&(peekchr==1))      //an yparxei xaraktiras stin cache kai
                                                   exoume ksekinisei na katagrafoume
 {
  recieved_char = Serial3.read();

  if ((recieved_char == 'x')||(recieved_char == 'y')||(recieved_char == 'z')||(recieved_char
== 'e'))
  {

   if ((pointer != ' ')&&(recieved_char == 'z'))      //ayto simainei oti ksanairthe to string ap
                                                      otin arxi xwris na exei teleiwsei,
                                                      midenizoume ola ta temp kai pame apo tin
                                                      arxi
   {  xtemp = "";
      ytemp = "";
      gyroZtemp = "";
```

```
    }

    pointer = recieved_char;

    if (xtemp!="") {
        x=(str2flt(xtemp))/100;
        xtemp = "";
        }

   if (ytemp!="") {
        y=(str2flt(ytemp))/100;
        ytemp = "";

     }



    if (gyroZtemp!="") {
        gyroZ=(str2flt(gyroZtemp));
        gyroZtemp = "";
   }

   if (recieved_char == 'e'){
        pointer=' ';
        peekchr=0;
        PID();
   }
  }

  else
  {
    switch (pointer){
      case 'x':
        xtemp+=recieved_char;
        break;
      case 'y':
        ytemp+=recieved_char;
        break;
      case 'z':
        gyroZtemp+=recieved_char;
        break;
       }

   }

  }

 }

}
```

```
float str2flt (String readstring)
{
   char carray[readstring.length() + 1];              //determine size of the array
   readstring.toCharArray(carray, sizeof(carray)); //put readStringinto an array
   float n = atof(carray);                            //convert the array into a Float
   return n;
}
```

Έλεγχος από τους PID

Αμέσως μετά την ανάγνωση τιμών του IMU καλείται η συνάρτηση του ελέγχου, μέσω των δύο PID για τα *roll* και *pitch* και του ενός PI για το *yaw*. Εδώ αρχικά γίνεται ένας έλεγχος προτού γίνουν οι υπολογισμοί των PID. Προϋποθέσεις του ελέγχου ώστε να επιτραπούν οι υπολογισμοί είναι η θέση του διακόπτη ασφαλείας να είναι στην κατάλληλη θέση, όπως εξηγείται παρακάτω, καθώς και η ισχύς των μοτέρ να είναι πάνω από κάποιο επίπεδο μετά το οποίο, όπως έχει βρεθεί πειραματικά, το tricopter αρχίζει να σηκώνεται από το έδαφος. Το τελευταίο εξυπηρετεί διότι το integral των PID δεν αυξάνει από τυχών μικροτιμές γωνιών όσο το όχημα βρίσκεται στο έδαφος, κάτι που μπορεί να δημιουργήσει αστάθεια από το πρώτο δευτερόλεπτο πτήσης.

Όταν τα παραπάνω κριτήρια ισχύουν, τότε εκκινεί το κομμάτι του κώδικα στο οποίο πραγματοποιείται ο έλεγχος των PID. Διαφορετικά, μηδενίζονται οι τιμές των integral σε περίπτωση που έχει γίνει ήδη πτήση ώστε να επανέλθουν οι PID σε αρχικό στάδιο, όπως και οι τιμές που καταλήγουν στις μεταβλητές που αποστέλλονται στα μοτέρ για μέγιστη ασφάλεια.

Ο κώδικας του ελεγκτή PI για το *yaw* πραγματοποιείται χωρίς τον έλεγχο κριτηρίων διότι η ροπή στρέψης επηρεάζει το όχημα εξ αρχής.

```
void PID() {

  temp_time_holder = micros() - temp_time;
  temp_time = micros();

  if ((thro_setpoint > 1400)&&(RC_switch_value>1500))
    PID_on = 1;
  if ((thro_setpoint <1400)||(RC_switch_value<1500))
    PID_on = 0;
```

```
  if (PID_on==1) {
    // ------ X PID ------
    error_x= - x - roll_setpoint;

    integral_x = integral_x + (error_x * prog_period);
    integral_x = constrain(integral_x, - I_limit_x, I_limit_x);      //oriothetoume to
                                                                     integral_x
    derivative_x = Kd_x * (x-last_x) / prog_period;
    last_x = x;
    output_x = (Kp_x * error_x) + (Ki_x * integral_x) + derivative_x;

    // ------ Y PID ------
    error_y = - y + pitch_setpoint;

    integral_y = integral_y + (error_y * prog_period);
    integral_y = constrain(integral_y, - I_limit_y, I_limit_y);      //oriothetoume to
                                                                     integral_y
    derivative_y = Kd_y * (y - last_y) / prog_period;
    last_y = y;
    output_y = (Kp_y * error_y) + (Ki_y * integral_y) + derivative_y;

    valESCright = output_x - output_y * 0.9 / 2; //to 0.9 einai epeidi, logw tis klisis tou pisw
moter, i anypswtiki dynami tou eiani mikroteri apo oti an itan teleiws orthio, opote etsi
mikrainoume ti dynami pou paei sta mprostina moter.
    valESCleft = - output_x - output_y * 0.9 / 2;
    valESCyaw = output_y;
  }

else
  {
    integral_x = 0;
    integral_y = 0;
    valESCright = 0;
    valESCleft = 0;
    valESCyaw = 0;
  }


  // ------ Z PI ------

  error_z = yaw_setpoint + gyroZ - 1500;
  integral_z = integral_z + (error_z * prog_period);
  integral_z = constrain(integral_z, - I_limit_z, I_limit_z);      //oriothetoume to
integral_z
  output_z = constrain(1500 + error_z * Kp_z + (Ki_z * integral_z), 1300, 1700);

  valservo = output_z;

  if (continue_program==1) send_PWM();
}
```

Αφού οι υπολογισμοί των ελεγκτών τελειώσουν, καλείται η συνάρτηση *send_PWM* μέσα στην οποία αποστέλλονται οι κατάλληλες τιμές στους ελεγκτές των μοτέρ και το σερβοκινητήρα.

Αποστολή σημάτων PWM

Σε αυτή τη συνάρτηση αρχικά, οι τιμές που έχουν υπολογιστεί από τους ελεγκτές PID προστίθενται με το thro_setpoint, τη μεταβλητή στην οποία είναι αποθηκευμένη η τιμή της ισχύος των κινητήρων όπως αυτή αποστέλλεται από το χρήστη μέσω του τηλεχειρισμού. Εν συνεχεία, περιορίζονται σε μέγιστες και ελάχιστες τιμές, όπως τις αναγνωρίζουν οι ελεγκτές των μοτέρ και τέλος, αποστέλλονται στα κατάλληλα pins του μικροελεγκτή.

```
void send_PWM() {

  valESCyaw_final = valESCyaw + thro_setpoint;
  valESCright_final = valESCright + thro_setpoint;
  valESCleft_final = valESCleft + thro_setpoint;

if (PID_on){
  valESCyaw_final = constrain (valESCyaw_final, 1050, 1900);
  valESCright_final = constrain (valESCright_final, 1050, 1900);
  valESCleft_final = constrain (valESCleft_final, 1050, 1900);
 }

  OCR3A = valESCright_final;
  OCR4A = valESCyaw_final;
  OCR4B = valESCleft_final;
  OCR4C = valservo;

 }
```

Σειριακή επικοινωνία

Η επόμενη συνάρτηση που καλεί ο Κυρίως Βρόγχος είναι αυτή της σειριακής επικοινωνίας με τον υπολογιστή. Μέσω αυτής ο χρήστης μπορεί να μεταβάλλει τις παραμέτρους των ελεγκτών PID, καθώς και να προβάλλει μέσω του GUI, που έχει δημιουργηθεί για το σκοπό, αυτό όλα τα στοιχεία τηλεμετρίας που αποστέλλει ο μικροελεγκτής του οχήματος.

Λόγω του μεγάλου όγκου δεδομένων που αποστέλλει ο μικροελεγκτής, εμφανίστηκε μεγάλο πρόβλημα καθυστέρησης της εκτέλεσης του προγράμματος. Η σειριακή επικοινωνία είναι αρκετά αργή σε σχέση με τις υπόλοιπες διεργασίες του μικροελεγκτή, οπότε ο μικροελεγκτής απλώς περιμένει να τελειώσει την μετάδοση για να συνεχίσει. Για να λυθεί αυτό, επιλέχθηκε η αποστολή μιας μόνο μεταβλητής σε κάθε κύκλο προγράμματος. Έτσι ένας μετρητής υποδεικνύει σε κάθε κύκλο ποια από τις μεταβλητές έχει σειρά αποστολής. Τα αποτέλεσμα ήταν τα επιθυμητά, χωρίς να έχουμε απώλειες στην απεικόνιση των δεδομένων στο GUI.

```
void serial_COMM() {

if (Serial2.available() > 0){
  char command = Serial2.read();
 switch (command) {
   case '1':  Kp_x+=0.01;              break;
   case 'q':  Kp_x=Kp_x-0.01;          break;
   case '2':  Ki_x+=0.01;              break;
   case 'w':  Ki_x=Ki_x-0.01;          break;
   case '3':  Kd_x+=0.1;               break;
   case 'e':  Kd_x+=-0.1;              break;
   case '4':  I_limit_x+=1;            break;
   case 'r':  I_limit_x=I_limit_x-1;   break;
   case '5':  Kp_y+=0.01;              break;
   case 't':  Kp_y=Kp_y-0.01;          break;
   case '6':  Ki_y+=0.01;              break;
   case 'y':  Ki_y=Ki_y-0.01;          break;
   case '7':  Kd_y+=0.1;               break;
   case 'u':  Kd_y=Kd_y-0.1;           break;
   case '8':  I_limit_y+=1;            break;
   case 'i':  I_limit_y=I_limit_y-1;   break;
   case 's':  Kp_z+=0.01;              break;
   case 'x':  Kp_z=Kp_z-0.01;          break;
   case 'd':  Ki_z+=0.01;              break;
   case 'c':  Ki_z=Ki_z-0.01;          break;
   case 'f':  I_limit_z+=1;            break;
   case 'v':  I_limit_z=I_limit_z-1;   break;
 }
}

// Angles
nums[0]=(x);
nums[1]=(y);

// Gyros
nums[2]=(gyroX);
nums[3]=(gyroY);
```

```
nums[4]=(gyroZ);

//Errors
nums[5]=(error_x);
nums[6]=(error_y);
nums[7]=(error_z);

//Setpoints
nums[8]=(roll_setpoint);
nums[9]=(pitch_setpoint);
nums[10]=(yaw_setpoint);
nums[11]=(RC_switch_value);

//Outputs
nums[12]=(valESCright_final);
nums[13]=(valESCleft_final);
nums[14]=(valESCyaw_final);
nums[15]=(valservo);

// PID of X
nums[16]=(Kp_x);
nums[17]=(Ki_x);
nums[18]=(Kd_x*10);
nums[19]=(I_limit_x);
nums[20]=(integral_x);

// PID of Y
nums[21]=(Kp_y);
nums[22]=(Ki_y);
nums[23]=(Kd_y*10);
nums[24]=(I_limit_y);
nums[25]=(integral_y);

// PID of Z
nums[26]=(Kp_z);
nums[27]=(Ki_z);
nums[28]=(Kd_z*10);
nums[29]=(I_limit_z);
nums[30]=(integral_z);

nums[31]=(thro_setpoint);
nums[32]=(temp_time_holder);
nums[33]=(battery_mes);
nums[34]=(batt_time);


Serial2.print(nums[serial_send_counter]);Serial2.print(",");
serial_send_counter++;

if (serial_send_counter>34) {
```

```
   serial_send_counter=0;
   Serial2.println ();
}

}
```

Ανάγνωση σημάτων και interrupts από την τηλεκατεύθυνση

Η ανάγνωση των σημάτων που αποστέλλονται από την τηλεκατεύθυνση γίνεται αποκλειστικά μέσω hardware interrupts. Συγκεκριμένα γίνονται δύο interrupts για το κάθε ένα από τα πέντε σήματα, ένα για την εκκίνηση του παλμού και ένα για το τέλος του. Με έναν χρονιστή μετράται ο χρόνος μεταξύ των δύο στιγμών και έτσι απορρέει το μήκος του παλμού. Εν συνεχεία, όταν κληθεί η ανάλογη συνάρτηση από τον Κυρίως Βρόγχο, οι τιμές αυτές περνούν από βαθυπερατά φίλτρα δύο βαθμών για απόσβεση του θορύβου και εξομάλυνση της κυμματομορφής που δημιουργούν.

```
 void rc_yaw_interupt() {          //------------------------------YAW Interrupt-------------------
---
 if(int0)
   count0=micros();                // we got a positive edge
 else
   yaw_pulse=micros()-count0;      // Negative edge: get pulsewidth
}


 void rc_thro_interupt() {         //------------------------------THROTTLE Interrupt---------
----
 if(int1)
   count1=micros();                // we got a positive edge
 else
  {
  thro_pulse=micros()-count1;      // Negative edge: get pulsewidth
  if (thro_pulse<1150) thro_pulse=1050;
  else thro_pulse=map (thro_pulse, 1150, 2000, 1400, 1650);
  }
 }


 void rc_pitch_interupt() {        //------------------------------PITCH Interrupt---------------
-----
 if(int2)
   count2=micros();                // we got a positive edge
 else
```

```
  pitch_pulse=micros()-count2;       // Negative edge: get pulsewidth
 }

 void rc_roll_interupt() {           //-------------------------------ROLL Interrupt-----------------
---
 if(int3)
   count3=micros();                 // we got a positive edge
 else
   roll_pulse=micros()-count3;      // Negative edge: get pulsewidth
 }

 void rc_sw_interupt()  {            //-------------------------------SWITCH Interrupt-------------
----
 if(int4)
   count4=micros();                 // we got a positive edge
 else
  RC_switch_value=micros()-count4; // Negative edge: get pulsewidth
 }



 void filter_RC (){                  //-----------------------------------FILTER RC----------------
----

 if (yaw_pulse>0){                   //an exoume lavei enan palmo sto pin tou yaw (pin 21)
   tmp_time_y[jy]=yaw_pulse;         //na valei tin timi stin jy thesi tou pinaka
   jy++;                             //opou 0<=jy<=1 oi 2 theseis tou pinaka
   if (jy>1) jy=0;
   yaw_setpoint=0;
   for (ky=0; ky<2; ky++)  yaw_setpoint+=tmp_time_y[ky];
   yaw_setpoint=yaw_setpoint/2;
   if (( yaw_setpoint < 1550) &&  ( yaw_setpoint > 1450))   yaw_setpoint= 1500;
   yaw_pulse=0;
 }


 if (thro_pulse>0){                  //throttle (pin 20)
   tmp_time_t[jt]=thro_pulse;
   jt++;
   if (jt>1) jt=0;
   thro_setpoint=0;
   for (kt=0; kt<2; kt++)  thro_setpoint+=tmp_time_t[kt];
   thro_setpoint= thro_setpoint/2;
   thro_pulse=0;
 }


 if (roll_pulse>0){                  //roll (pin 19)
   tmp_time_r[jr]=roll_pulse;
   jr++;
```

```
  if (jr>1) jr=0;
  roll_setpoint=0;
  for (kr=0; kr<2; kr++)  roll_setpoint+=tmp_time_r[kr];
  roll_setpoint = (roll_setpoint / 2) + 50;
  if (( roll_setpoint < 1550) &&  ( roll_setpoint > 1450)) roll_setpoint= 1500;
  roll_setpoint=map (roll_setpoint, 1000, 2000, -20, 20);
  roll_pulse=0;
 }


 if (pitch_pulse>0) {                    //pitch (pin 18)
  tmp_time_p[jp]=pitch_pulse;
  jp++;
  if (jp>1) jp=0;
  pitch_setpoint=0;
  for (kp=0; kp<2; kp++)  pitch_setpoint+=tmp_time_p[kp];
  pitch_setpoint = pitch_setpoint / 2;
  if (( pitch_setpoint < 1550) &&  ( pitch_setpoint > 1450)) pitch_setpoint= 1500;
  pitch_setpoint= map (pitch_setpoint,1000, 2000, -20, 20);
  pitch_pulse=0;
 }

}
```

Έλεγχος διακόπτη ασφαλείας

Το τελευταίο μέρος κώδικα που εκτελείται στον Κυρίως Βρόγχο είναι ο έλεγχος που πραγματοποιείται στο πέμπτο σήμα που αποστέλλει η τηλεκατεύθυνση. Πρόκειται για ένα διακόπτη, που ανάλογα με τη θέση του, οπότε και τον παλμό που αποστέλλει, κρίνεται η εκτέλεση ορισμένων συναρτήσεων. Αυτό συμβαίνει καθαρά για λόγους ασφαλείας και επί της ουσίας η χρήση του διακόπτη είναι η εκκίνηση ή μη των μοτέρ. Έτσι, ο χρήστης μπορεί ανά πάσα στιγμή κρίνει απαραίτητο να διακόψει τη λειτουργία του οχήματος. Κάτι τέτοιο βέβαια μπορεί να προκαλέσει πτώση του tricopter εν ώρα πτήσης, αλλά ίσως να είναι απαραίτητο για την αποφυγή τραυματισμού από τους έλικες.

```
 if (RC_switch_value==0) {

    OCR3A=motor_off_val;
    OCR4A=motor_off_val;
    OCR4B=motor_off_val;
    OCR4C=servo_off_val;
 }
 else if (RC_switch_value<1500) {      //if controller on and rc switch is off
```

```
    OCR3A=motor_off_val;
    OCR4A=motor_off_val;
    OCR4B=motor_off_val;
    OCR4C=servo_off_val;

    continue_program=0;
    PID_on=0;
    unsigned int tmp_time_y[]={1500,1500};   // pinakas gia LP filtro stin eisodo tou RC YAW
    unsigned int tmp_time_t[]={1030,1030};    // pinakas gia LP filtro stin eisodo tou RC THR
    unsigned int tmp_time_p[]={1500,1500};// pinakas g LP filtro stin eisodo tou RC PITCH
    unsigned int tmp_time_r[]={1500,1500};  // pinakas gia LP filtro stin eisodo tou RC ROLL
  }
else if (RC_switch_value < 2300) continue_program=1;
```

Κεφάλαιο 5 - Αποτελέσματα, συμπεράσματα

Αποτελέσματα - Συμπεράσματα

Η καλή θεωρητική μελέτη των διαφόρων αντικειμένων με τα οποία η εργασία συσχετίζεται μπορεί να βοηθήσει εξαιρετικά στην προσπέλαση εμποδίων, όμως δε μπορεί να προετοιμάσει πλήρως κάποιον για όλα τα αναπάντεχα εμπόδια που θα συναντήσει.

Η εις βάθος κατανόηση των μικροελεγκτών εξυπηρετεί κατά πολύ στην υλοποίηση και βελτιστοποίηση του κώδικα, ειδικά σε ένα project όπου οι χρόνοι και χρονισμοί έχουν μεγάλη σημασία για τη βέλτιστη λειτουργία του συνόλου του συστήματος.

Έπειτα από πολλούς πειραματισμούς είναι σίγουρα εμφανές πως το τελικό πραγματικό μοντέλο του συστήματος απέχει πολύ από αυτό των αρχικών πειραμάτων. Οι ασκούμενες εξωτερικές δυνάμεις που αναπόφευκτα επιδρούν πάνω στο όχημα στις πειραματικές διατάξεις επηρεάζουν τόσο το σύστημα ώστε η διαφορά πειραματικών παραμέτρων PID ή ευαισθησίας χειρισμού από την τηλεκατεύθυνση να είναι αισθητά μεγάλη σε σχέση με τις απαιτούμενες στο πραγματικό, τελικό μοντέλο.

Παρόλα αυτά, οι πειραματισμοί αυτοί επέτρεψαν την ασφαλή εξερεύνηση και μελέτη των αντικειμένων γύρω από την κατασκευή του οχήματος, κάτι που συνέβαλλε σημαντικά στην κατανόηση τους, ώστε αργότερα να μπορούν να επιλυθούν προβλήματα στο πραγματικό μοντέλο. Επί παραδείγματι, η μέθοδος Ziegler-Nichols που εφαρμόστηκε, βοήθησε στην κατανόηση του τρόπου που η κάθε παράμετρος του PID επηρεάζει το tricopter.

Η τηλεμετρία είναι ένα αναντικατάστατο εργαλείο που προσφέρει επίβλεψη και έλεγχο σε πραγματικό χρόνο. Ακόμη, δίνοντας στο χρήστη τη δυνατότητα να μεταβάλλει τις παραμέτρους του PID, ακόμη και χωρίς τη διακοπή πτήσης ή κάποιας άλλης δοκιμής, εξοικονομεί πάρα πολύ, πολύτιμο χρόνο. Ακόμη, η αποθήκευση των δεδομένων της τηλεμετρίας για μελέτη σε μεταγενέστερο χρόνο, βοηθά στη μελέτη τους σε χρόνο βολικό για το χρήστη, για τον εντοπισμό σφαλμάτων και λαθών και την καλύτερη κατανόηση της κατάστασης.

Το *fine tuning*, είναι απαραίτητο για την βέλτιστη πτήση ενός τέτοιου οχήματος, αλλά αρκετά δύσκολο στην επίτευξη του. Ακόμη, ο χειρισμός ενός τέτοιου οχήματος είναι ιδιαίτερα

δύσκολος για τον αρχάριο, μη μυημένο χρήστη. Η εξάσκηση σε εξομοιωτές πτήσης ιπτάμενων τηλεκατευθυνόμενων οχημάτων είναι πολύ βοηθητική και χρήσιμη σε αυτές τις περιπτώσεις.

Καθώς πρόκειται για μία εύθραυστη κατασκευή, κάποιου είδους προστατευτικά για το όχημα προτείνονται, κυρίως περιμετρικά των μοτέρ, καθώς οι έλικες είναι εξαιρετικά εύκολο να καταστραφούν σε μια πτώση, καθώς και στο κάτω μέρος του οχήματος όπου βρίσκεται η μπαταρία, η οποία, όπως αναφέρεται σε προηγούμενο κεφάλαιο, μπορεί να γίνει αρκετά επικίνδυνη σε περιπτώσεις κρούσης, προκαλώντας ως και ολοκληρωτική καταστροφή στο όχημα. Τα προστατευτικά αυτά πρέπει να είναι όσο το δυνατό πιο ελαφριά και ανθεκτικά.

Διάφορα μικρά ατυχήματα είναι σχεδόν αδύνατο να αποφευχθούν, οπότε πρέπει και όλα τα ηλεκτρονικά εξαρτήματα να είναι καλά προστατευμένα και μονωμένα. Κατά τη διάρκεια της ανάπτυξης του tricopter λόγω κάποιων πτώσεων, δημιουργήθηκαν βραχυκυκλώματα που κατέστρεψαν αρχικά το πρώτο IMU (Sparkfun 9DOF Razor), υποχρεώνοντας αγορά ενός νέου IMU, και σε εκ νέου σχεδίαση των ηλεκτρονικών κυκλωμάτων του tricopter. Σε μεταγενέστερη πτώση, καταστράφηκε το ESC (electronic speed controller) του ενός μοτέρ, δημιουργώντας καθυστέρηση κάποιων μηνών μέχρι την παραλαβή των νέων και επαναρρύθμιση του tricopter.

Τα ποιοτικά υλικά, είτε πρόκειται για κομμάτια της κατασκευής είτε για ηλεκτρονικά μέρη, έχουν εμφανή διαφορά με τα μέτριας ποιότητας, βοηθούν το έργο να προχωρήσει ταχύτερα και τυγχάνουν μικρότερου αριθμού βλαβών. Για παράδειγμα οι πρώτοι φθηνοί σερβοκινητήρες δεν άντεξαν τις μεγάλες δυνάμεις που αναπτύχθηκαν σε διάφορες πτώσεις, οπότε αντικαταστάθηκαν από ένα servo, με μεταλλικά γρανάζια, που αντέχει αρκετά παραπάνω.

Στο πρώτο στάδιο της κατασκευής, έγινε χρήση σωλήνων από ανθρακονήματα διαμέτρου 11mm για τα άκρα του tricopter. Μετά από αρκετές δοκιμές, έγινε σαφές πως οι κυλινδρικοί σωλήνες μειονεκτούν έναντι των τετράγωνων όσον αφορά τη στήριξή τους στο σώμα του tricopter καθώς και τη στήριξη των μοτέρ στις άκρες τους. Αντικαταστάθηκαν λοιπόν από τετράγωνους σωλήνες αλουμινίου, που είχαν το επιπλέον προτέρημα της μεγαλύτερης αντοχής στις πτώσεις προσθέτοντας ελάχιστο επιπλέον βάρος. Αργότερα, καθώς υπήρχε ανάγκη για μεγαλύτερα άκρα, για επιπλέον προστασία στα μοτέρ και τους έλικες, αντικαταστάθηκαν με σωλήνες αλουμινίου σχήματος Π. Αυτοί, χωρίς να υστερούν στη

στιβαρότητα, επιτρέπουν την ευκολότερη πρόσβαση στις βίδες που στηρίζουν τα μοτέρ, καθώς και τη χώνευση των καλωδίων των μοτέρ.

Στόχος της παρούσας εργασίας και μελλοντικές αναβαθμίσεις

Το τελικό αποτέλεσμα της εργασίας αυτής εκπλήρωσε τους κύριους στόχους της. Την κατασκευή ενός μη επανδρωμένου ιπτάμενου οχήματος ικανού να αιωρηθεί χωρίς ανθρώπινη επέμβαση.

Οι λόγοι για τους οποίους επιλέχθηκε το συγκεκριμένο είδος μη επανδρωμένου ιπτάμενου οχήματος για την εκπόνηση της πτυχιακής μας εργασίας είναι αφενός μεν ο βαθμός δυσκολίας που παρουσιάζει σε σχέση με τα άλλη είδη, κάτι που μας ώθησε να αναπτύξουμε τις γνώσεις μας γύρω από πληθώρα αντικειμένων ώστε να ανταπεξέλθουμε στην κάθε ανάγκη της μελέτης μας. Αφετέρου είναι το λιγότερο επιλεγμένο είδος προς κατασκευή, κάτι που μας επέτρεπε να μην ακολουθήσουμε την πεπατημένη οδό και πιθανά να δώσουμε με τη σειρά μας κάτι στο χώρο των αυτομάτων συστημάτων και των μη επανδρωμένων οχημάτων.

Μελλοντική ανάπτυξη

Το εύρος των δυνατών αναβαθμίσεων στο tricopter είναι μεγάλο.

Αρχικά, ένας δέκτης GPS μπορεί να δίνει το γεωγραφικό στίγμα του οχήματος στο χάρτη καθώς και να υποβοηθήσει στην ευστάθεια του. Η γνώση του στίγματος του οχήματος σε πραγματικό χρόνο, μας δίνει επίσης τη δυνατότητα ανάπτυξης εφαρμογών και είναι το πρώτο βήμα για την πραγματοποίηση των χρήσεων του οχήματος, όπως αυτές αναφέρθηκαν παραπάνω.

Η συνηθέστερη αναβάθμιση ενός τέτοιου οχήματος, είναι η τοποθέτηση βιντεοκάμερας επάνω στο σώμα του. Αυτή θα δίνει στο χρήστη τη δυνατότητα πτήσης του οχήματος εκτός του οπτικού του πεδίου, καθώς και καταγραφή εικόνων και βίντεο από το χώρο στον οποίο βρίσκεται ή αποστέλλεται το όχημα.

Μια ακόμη αναβάθμιση είναι η τοποθέτηση αισθητηρίων όπως υπερήχων ή laser, με τα οποία μπορεί να γίνει καταμέτρηση των αποστάσεων περιμετρικά του οχήματος, και με κατάλληλη εφαρμογή, να γίνεται χαρτογράφηση του χώρου στον οποίο ίπταται το όχημα.

Τέλος, αναλόγως με την επιθυμητή χρήση του οχήματος από το χρήστη, ανάλογα υλικά και αισθητήρια μπορούν να χρησιμοποιηθούν, δίνοντας μεγάλη δυνατότητα εξατομίκευσης και κάλυψης αναγκών από το βασικό μοντέλο της κατασκευής μας.

Βιβλιογραφία

William Premerlani and Paul Bizard ‘Direction Cosine Matrix IMU: Theory’, 2009

Karl-Johan Barsk ‘Model Predictive Control of a Tricopter’, Linköpings universitet, Department of Electrical Engineering, 2012

Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim and Tarek Hamel ‘A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV’, 2008

Sebastian O.H. Madgwick ‘An efficient orientation filter for inertial and inertial/magnetic sensor arrays’, 2010

Φώτης Πατώνης, ‘Συνδυασμός των τεχνολογιών χαμηλού κόστους Αδρανειακών Μονάδων Μέτρησης και του Παγκόσμιου Δορυφορικού Συστήματος Πλοήγησης σε Εφαρμογές Φωτογραμμετρίας’, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2012

Nguyen Ho Quoc Phuong, Hee-Jun Kang, Young-Soo Suh, Young-Sik Ro ‘A DCM Based Orientation Estimation Algorithm with an Inertial Measurement Unit and a Magnetic Compass’, Journal of Universal Computer Science, vol. 15, no. 4 (2009)

Starlino Electronics ‘DCM Tutorial – An introduction to orientation kinematics (rev 0.1)’, 2011

SENSR Company ‘Practical guide to Accelerometers’

Κωνσταντίνος Αλέξης ‘Έλεγχος Συνεργαζόμενων Ρομποτικών Οχημάτων’, Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών, 2011

http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/

http://www.aeromodelistis.com/smfgr/index.php?PHPSESSID=bbfkkak2usn16hjj6mcl8e2ph1&topic=4437.0

http://en.wikipedia.org/wiki/Multicopter

http://en.wikipedia.org/wiki/Cierva_Air_Horse

http://www.aviastar.org/helicopters_eng/west_w11.php

http://www.varesano.net/projects/hardware/FreeIMU

http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf

http://www.arduino.cc/

http://playground.arduino.cc/

http://forum.arduino.cc/

http://processing.org/

http://www.rcgroups.com/forums/showthread.php?t=1006721

https://code.google.com/p/sf9domahrs/

http://en.wikipedia.org/wiki/Attitude_and_Heading_Reference_Systems

http://www.starlino.com/imu_guide.html

http://en.wikipedia.org/wiki/PID_controller

http://extremeelectronics.co.in/avr-tutorials/avr-timers-an-introduction/

http://mil.ufl.edu/5666/handouts/ATMPWM.pdf

http://www.ecalc.ch/xcoptercalc.htm?ecalc&lang=en

Παράρτημα

Πηγαίος κώδικας στο Processing

```
import processing.serial.*;

int value_1 = 0;
int value_2 = 1;
String text_value="Gwnies";


Serial myPort;
PFont fontA;
String gg;
String filename;
byte start=0;
int battVal=75;
long serialStart=0;
float[] nums=new float[42];
int[] values1;
int[] values2;
int[] values3;
int[] values4;
int[] valuesOutputX;
int[] valuesErrorX;
String outputString;
String[] outputStringArr;



void setup()
{
  filename=sketchPath("")+"CSV/"+nf(year(),4)+nf(month(),2)+nf(day(),2)+"_"
+nf(hour(),2)+"."+nf(minute(),2)+"."+nf(second(),2)+".txt";
  println("Creating file: "+filename);
  outputString = "x,y,X,Y,Z,Error X,Error Y,Error Z,Setpoint X,Setpoint
Y,Setpoint Z,SW1,Output Right,Output Left,Output Yaw,Output
Servo,Kp_X,Ki_X,Kd_X,I Limit_X,Integral_X,Kp_Y,Ki_Y,Kd_Y,I
Limit_Y,Integral_Y,Kp_Z,Ki_Z,Kd_Z,I
Limit_Z,Integral_Z,Throttle,Period,Battery,PID ON";
  outputStringArr = split(outputString,",");
  saveData(filename,"time      "+join(outputStringArr, "      "),false);

  println(Serial.list());
  println("Hello!");
  size(1200, 750);
  smooth();
  fontA = loadFont("Arial48.vlw");
  textAlign(LEFT);
  textFont(fontA, 32);
  background(0);
  String portName = Serial.list()[1];
  myPort = new Serial(this, portName, 38400);
  myPort.bufferUntil(10);
  values1 = new int[width];
  values2 = new int[width];
  values3 = new int[width];
  values4 = new int[width];
  println("setup end");
```

```
}


void draw()
{

  if(nums[33]>15) background(0);
  else if (second()%2==0) background(#FF381d);
  else background(0);


  stroke(100);
  fill(255);
  textFont(fontA, 20);
  text(text_value,width-80,225);
  text("X", width-150, 149);
  line(0, 150, width, 150);
  text("Y", width-150, 299);
  line(0, 300, width, 300);


  if ((start==1)){
nums = float(split(gg, ','));
if (nums.length < 42) { nums=expand(nums,42); }


textFont(fontA, 24);

text("Angles:",5,460);
text("X:"+nums[0],140,460);
text("Y:"+nums[1],270,460);
text("Gyro Z:"+nums[4],359,460);

text("Errors:",5,490);
text("X:"+nums[5],140,490);
text("Y:"+nums[6],270,490);
text("Z:"+nums[7],418,490);

text("Setpoints:",5,520);
text("X:"+int(nums[8]),140,520);
text("Y:"+int(nums[9]),270,520);
text("Z:"+int(nums[10]),418,520);
text("Sw#1:"+nums[11],540,520);
text("Throtle:"+nums[31],720,520);

text("Ouputs:",5,550);
text("R:"+int(nums[12]),140,550);
text("L:"+int(nums[13]),270,550);
text("Y:"+int(nums[14]),418,550);
text("Serv:"+int(nums[15]),540,550);

noFill();
strokeWeight(3);

rect(450,740,740,-140);
rect(450,740,740,-105);
rect(450,740,740,-70);
rect(450,740,740,-35);

rect(450,740,620,-140);
rect(450,740,490,-140);
```

```
rect(450,740,375,-140);
rect(450,740,250,-140);
rect(450,740,130,-140);
strokeWeight(1);


text("PID",470,625);
text("Kp",590,625);
text("Ki",710,625);
text("Kd",835,625);
text("I limit",950,625);
text("Integral",1080,625);

text("X angle",470,660);
text(nums[16],590,660);
text(nums[17],710,660);
text(nums[18]/10,835,660);
text(nums[19],950,660);
text(nums[20],1080,660);

text("Y angle",470,695);
text(nums[21],590,695);
text(nums[22],710,695);
text(nums[23]/10,835,695);
text(nums[24],950,695);
text(nums[25],1080,695);

text("Z gyro",470,730);
text(nums[26],590,730);
text(nums[27],710,730);
text(nums[28]/10,835,730);
text(nums[29],950,730);
text(nums[30],1080,730);

// Throtle nums[31]
text("Period:"+nums[32],5,650);
text("Battery:"+ map(round( nums[33] * 100.0f ) / 100.0f, 0, 100, 8.9,
12.4),5,720);
text("PID ON:"+nums[34],5,680);

  for (int i=0; i<width-1; i++)
{    values1[i] = values1[i+1];
     values2[i] = values2[i+1];
}


if (text_value=="Outputs"){
  values1[width-1] = int(nums[value_1]-nums[31]); //diladi apo tin timi
eksodou na vgalei to throttle gia na doume mono tin eksodo tou PID
  values2[width-1] = int(nums[value_2]-nums[31]);
}else{
  values1[width-1] = int(nums[value_1]);
  values2[width-1] = int(nums[value_2]);
}


  for (int x=1; x<width; x++) {
    stroke(color(0, 255, 255));
    line(width-x,   150-values1[x-1], width-1-x, 150-values1[x]);
    stroke(color(0, 255, 0));
    line(width-x,   300-values2[x-1], width-1-x, 300-values2[x]);
```

```
  }

    stroke(color(255,255,255));
}

  logo();
//  rotation();
  battery_gauge();


}




void serialEvent(Serial myPort) {
    gg = (myPort.readString());
    outputStringArr = split(trim(gg),",");
    outputString=hour()+":"+minute()+":"+second()+"
        "+join(outputStringArr, "      ");
    saveData(filename,outputString,true);
    start=1;
}

void keyTyped() {
  myPort.write(key);
}

void keyPressed() {
if (key == CODED) {
  if (keyCode == KeyEvent.VK_F1) {
    value_1 = 0;
    value_2 = 1;
    text_value="Gwnies";
  } else if (keyCode == KeyEvent.VK_F2){
    value_1 = 2;
    value_2 = 3;
    text_value="Gyro";
  }else if (keyCode == KeyEvent.VK_F3){
    value_1 = 12;
    value_2 = 13;
    text_value="Outputs";
  }
}
}


void battery_gauge() {
  pushMatrix();
  translate (300, 720);
    fill(0);
    strokeWeight(4);
    rect(-130,20,260,-23);
    arc (0,0,260,260,PI,2*PI);
    strokeWeight(1);
    fill(255);

for (int angle=0;angle>-181;angle+=-18)
```

```
    line(130*cos(angle*PI/180),130*sin(angle*PI/180),120*cos(angle*PI/180),
120*sin(angle*PI/180));
    stroke(#FF381d);
    strokeWeight(3);
    line(126*cos(-153*PI/180),126*sin(-153*PI/180),110*cos(-
153*PI/180),110*sin(-153*PI/180));
    stroke(255);
    strokeWeight(1);
    fill(255);
    text ("50%", -18, -100);
    text ("BATTERY", -30, -40);
    text ("0%", -110, 5);
    text ("100%", 70, 5);
  battVal=int(map(nums[33]+50,0,100,0,180));
  rotate(radians(battVal));
  stroke(#0030FF);
  fill(#0030FF);
  triangle(10, 0,-10, 0,  0, 110);
  triangle(10, 0,-10, 0,  0, -10);

  fill(0);
  ellipse (0,0,3,3);
  fill(255);

  popMatrix();
}



void logo() {
  noFill();
  translate(width-75, 20);
  fill(#c5681d);
  roundedRect(140, 30, 3 );
  translate(75-width, -20);
  fill(#eeeeee);
  textFont(fontA, 18);
  text("DART_tricopter", width-136, 27);

  noFill();
  translate(115, 20);
  fill(#c5681d);
  roundedRect(215, 30, 3 );
  translate(-115, -20);
  fill(#eeeeee);
  textFont(fontA, 18);
  text(nf(year(),4)+"/"+nf(month(),2)+"/"+nf(day(),2)+"
"+nf(hour(),2)+":"+nf(minute(),2)+":"+nf(second(),2), 20, 27);



}

void roundedRect(float w, float h, float r)
{
    beginShape();
    vertex(  w/2 - r, -h/2 );
    bezierVertex( w/2 - r , -h/2, w/2, -h/2, w/2 , -h/2 + r);
    vertex(  w/2 , h/2 - r );
    bezierVertex( w/2, h/2 , w/2 - r , h/2 , w/2 - r, h/2);
    vertex( -w/2 + r,  h/2 );
```

```
    bezierVertex( - w/2, h/2, -w/2, h/2 - r, -w/2 , h/2 - r );
    vertex( -w/2 , -h/2 + r );
    bezierVertex( -w/2, -h/2, -w/2 + r, -h/2, -w/2 + r, -h/2 );
    endShape(CLOSE);
}


void saveData(String fileName, String newData, boolean appendData){
  BufferedWriter bw = null;
  try {
    FileWriter fw = new FileWriter(fileName, appendData);
    bw = new BufferedWriter(fw);
    bw.write(newData + System.getProperty("line.separator"));
  } catch (IOException e) {
  } finally {
    if (bw != null){
      try {
        bw.close();
      } catch (IOException e) {}
    }
  }
}
```

Πηγαίος κώδικας στο Arduino Mega:

```
#define RC_YAW 2 // pin 21   //arduino pins attached to the reciever
#define RC_THRO 3 // pin 20
#define RC_PITCH 4 // pin 19
#define RC_ROLL 5 // pin 18
#define RC_SWITCH 0 // pin 2
#define LED1 51
#define LED2 53
                        // ta pins me ta interrupts, gia na vlepoume grigora an einai 1 i 0
#define int0 (PIND & 0b00000001) // Faster than digitalRead
#define int1 (PIND & 0b00000010)
#define int2 (PIND & 0b00000100)
#define int3 (PIND & 0b00001000)
#define int4 (PINE & 0b00010000)

long count0, count1, count2, count3, count4;  // prosorines times gia micros() sta positive edge (read_RC)
int yaw_pulse=0, thro_pulse=0, roll_pulse=0, pitch_pulse=0; // prosorines times gia mikos palmou se microsecond, pros filter_RC

float prog_period=0.1;  //o xronos enos kyklou tou programmatos, mpainei stous PID

int thro_setpoint=1010;
float roll_setpoint=0, pitch_setpoint=0, yaw_setpoint=0;

float error_x=0, error_y=0, error_z=0;                    //ta sfalamata gia ton PID
float previous_error_x=0, previous_error_y=0, previous_error_z=0;
float integral_x=0.01, integral_y=0.01, integral_z=0;
float derivative_x, derivative_y, derivative_z;


float Kp_x=6.5, Kp_y=6.5, Kp_z=0.2;
float Ki_x=0.4, Ki_y=0.5, Ki_z=0.1;
float Kd_x=-11, Kd_y=-11, Kd_z=0;
float output_x, output_y, output_z;
float I_limit_x=270, I_limit_y=270, I_limit_z=2000; // orio gia to integral


int continue_program=0;
int PID_on=0;
int PID_angle_on=0;

int motor_off_val=1000;

int servo_off_val=1500;
int valservo=1500;
int valESCyaw=0;
int valESCright=0;
int valESCleft=0;
int valESCyaw_final=0;
```

```
int valESCright_final=0;
int valESCleft_final=0;

long temp_time=0;
long temp_time_holder=0;

float nums[42];
int battery_mes=0;
int batt_time=0;

char recieved_char; // times gia serial me IMU
byte peekchr=0;
char pointer;

float x, y, gyroZ, gyroX, gyroY; //oi gwnies mas apo to imu
float gyroXtable[]={0,0,0,0,0,0,0,0,0,0};

float last_x, last_y; // times gia PID

String xtemp, ytemp, gyroZtemp;

unsigned int RC_switch_value = 0;                    //i timi tou diakopti

unsigned int tmp_time_y[]={1500,1500};     // pinakas gia LP filtro stin eisodo tou RC YAW
unsigned int tmp_time_t[]={1030,1030};     // pinakas gia LP filtro stin eisodo tou RC THR
unsigned int tmp_time_p[]={1500,1500};     // pinakas gia LP filtro stin eisodo tou RC PITCH
unsigned int tmp_time_r[]={1500,1500};     // pinakas gia LP filtro stin eisodo tou RC ROLL

int valESCyaw_table[]={1030,1030,1030,1030,1030};
int valESCright_table[]={1030,1030,1030,1030,1030};
int valESCleft_table[]={1030,1030,1030,1030,1030};

int jy=0, ky=0;  // counter LP filtrou RC YAW
int jt=0, kt=0;  // counter LP filtrou RC THR
int jp=0, kp=0;  // counter LP filtrou RC PITCH
int jr=0, kr=0;  // counter LP filtrou RC ROLL

long LEDtime;
byte LEDi;

byte serial_send_counter=0;

void setup() {

  pinMode (5, OUTPUT);
  pinMode (6, OUTPUT);
  pinMode (7, OUTPUT);
  pinMode (8, OUTPUT);
```

```
  TCCR3A&= 0b11111100;  TCCR3B&= 0b11110111;  TCCR3B|= 0b00010000;   //ayta setaroun ta WGMn4 mexri WGMn0 stis times 1 0 0 0 gia to mode PWM, Phase and Frequency Correct, me xrisi tou ICRn
  TCCR3B&= 0b11111010;  TCCR3B|= 0b00000010;                  //ayta setaroun ta CSn2 mexri CSn0 stis times 0 1 0 gia prescaler 8
  TCCR3A&= 0b10101011;  TCCR3A|= 0b10101000;                  //ayta setaroun ta COMnA1 mexri COMnC0 stis times 0 1 0 1 0 1 gia na rythmisoume tin leitourgia tou PWM (na to kanei 0 otan synantaei to OCRn anevainontas kai 1 katevainontas)
  ICR3 = 20000;                                               //50Hz gia to prescaler 8


  TCCR4A&= 0b11111100;  TCCR4B&= 0b11110111;  TCCR4B|= 0b00010000;   //ayta setaroun ta WGMn4 mexri WGMn0 stis times 1 0 0 0 gia to mode PWM, Phase and Frequency Correct, me xrisi tou ICRn
  TCCR4B&= 0b11111010;  TCCR4B|= 0b00000010;                  //ayta setaroun ta CSn2 mexri CSn0 stis times 0 1 0 gia prescaler 8
  TCCR4A&= 0b10101011;  TCCR4A|= 0b10101000;                  //ayta setaroun ta COMnA1 mexri COMnC0 stis times 0 1 0 1 0 1 gia na rythmisoume tin leitourgia tou PWM (na to kanei 0 otan synantaei to OCRn anevainontas kai 1 katevainontas)
  ICR4 = 20000;                                               //50Hz gia to prescaler 8


  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);


  Serial3.begin(115200);
  Serial2.begin(38400);


  attachInterrupt(RC_YAW, rc_yaw_interupt, CHANGE); // attach a PinChange Interrupt
  attachInterrupt(RC_THRO, rc_thro_interupt, CHANGE); // attach a PinChange Interrupt
  attachInterrupt(RC_PITCH, rc_pitch_interupt, CHANGE); // attach a PinChange Interrupt
  attachInterrupt(RC_ROLL, rc_roll_interupt, CHANGE); // attach a PinChange Interrupt
  attachInterrupt(RC_SWITCH, rc_sw_interupt, CHANGE); // attach a PinChange Interrupt


}




void loop() {

  if (RC_switch_value==0) {

      OCR3A=motor_off_val;
      OCR4A=motor_off_val;
      OCR4B=motor_off_val;
```

```
    OCR4C=servo_off_val;
 }
 else if (RC_switch_value<1500) {      //if controller on and rc switch is off

    OCR3A=motor_off_val;
    OCR4A=motor_off_val;
    OCR4B=motor_off_val;
    OCR4C=servo_off_val;

    continue_program=0;
    PID_on=0;
    unsigned int tmp_time_y[]={1500,1500};       // pinakas gia LP filtro stin eisodo tou RC
YAW
    unsigned int tmp_time_t[]={1030,1030};       // pinakas gia LP filtro stin eisodo tou RC
THR
    unsigned int tmp_time_p[]={1500,1500};       // pinakas gia LP filtro stin eisodo tou RC
PITCH
    unsigned int tmp_time_r[]={1500,1500};       // pinakas gia LP filtro stin eisodo tou RC
ROLL
   }
  else if (RC_switch_value < 2300) continue_program=1;


 if (millis() - LEDtime > 100  ){
    LEDi++;
    if (battery_mes > 15) {
      if (LEDi == 5) {       digitalWrite(LED1, HIGH);     }
      if (LEDi == 7) {       digitalWrite(LED1, LOW);      }
      if (LEDi == 8) {       digitalWrite(LED2, HIGH);     }
      if (LEDi == 10) { digitalWrite(LED2, LOW);   LEDi = 0; }
    }
    else {
      if (LEDi % 2 == 0) {digitalWrite(LED1, HIGH); digitalWrite(LED2, HIGH);}
      if (LEDi % 2 == 1) {digitalWrite(LED1, LOW); digitalWrite(LED2, LOW);}

      if (LEDi == 10) {LEDi = 0;}
    }

    LEDtime = millis();
    batt_time++;
 }

 if (batt_time == 10) {
   battery_mes = analogRead(1);
   battery_mes = map (battery_mes, 600, 850, 0, 100);
   batt_time = 0;
 }

 read_IMU();
 serial_COMM ();
```

```
  filter_RC();

}



void PID() {

  temp_time_holder=micros()-temp_time;
  temp_time=micros();

  if ((thro_setpoint > 1400)&&(RC_switch_value>1500))
    PID_on=1;
  if ((thro_setpoint <1400)||(RC_switch_value<1500))
    PID_on=0;


  if (PID_on==1) {
    // ------ X PID ------
    error_x= - x - roll_setpoint;

    integral_x=integral_x+(error_x*prog_period);
    integral_x = constrain(integral_x, - I_limit_x, I_limit_x); //oriothetoume to integral_x gia na
min ksefeugei i timi tou
    derivative_x=Kd_x*(x-last_x)/prog_period;
    last_x=x;
    output_x=(Kp_x*error_x)+(Ki_x*integral_x)+derivative_x;

    // ------ Y PID ------
    error_y = - y + pitch_setpoint;

    integral_y = integral_y + (error_y * prog_period);
    integral_y = constrain(integral_y, - I_limit_y, I_limit_y); //oriothetoume to integral_y gia na
min ksefeugei i timi tou
    derivative_y = Kd_y * (y - last_y) / prog_period;
    last_y = y;
    output_y = (Kp_y * error_y) + (Ki_y * integral_y) + derivative_y;

    valESCright = output_x - output_y * 0.9 / 2; //to 0.9 einai epeidi, logw tis klisis tou pisw
moter, i anypswtiki dynami tou eiani mikroteri apo oti an itan teleiws orthio, opote etsi
mikrainoume ti dynami pou paei sta mprostina moter.
    valESCleft = - output_x - output_y * 0.9 / 2;
    valESCyaw = output_y;
  }
  else

  {
    integral_x = 0;
    integral_y = 0;
```

```
   valESCright = 0;
   valESCleft = 0;
   valESCyaw = 0;

  }


  // ------ Z PI ------

  error_z = yaw_setpoint + gyroZ - 1500;
  integral_z = integral_z + (error_z * prog_period);
  integral_z = constrain(integral_z, - I_limit_z, I_limit_z); //oriothetoume to integral_z gia na
min ksefeugei i timi tou kai exoume drift
  output_z = constrain(1500 + error_z * Kp_z + (Ki_z * integral_z), 1300, 1700);

  valservo = output_z;

  if (continue_program==1) send_PWM();
}



void read_IMU() {


while (Serial3.available() > 0){



  if (Serial3.peek()=='z'){  //an to string ksekinaei
       peekchr=1;          //tote arxizoume na katagrafoume
  }  else if (peekchr==0) {  //alliws an den exoume arxise ina katagrafoume
       Serial3.read();      //diavase enan xaraktira gia na proxwrisei to buffer
  }


  if ((Serial3.available() > 0)&&(peekchr==1)) //an yparxei xaraktiras stin cache kai exoume
ksekinisei na katagrafoume
  {
    recieved_char = Serial3.read();

   if ((recieved_char == 'x')||(recieved_char == 'y')||(recieved_char == 'z')||(recieved_char ==
'e'))
   {

     if ((pointer != ' ')&&(recieved_char == 'z')) //ayto simainei oti ksanairthe to string ap otin
arxi xwris na exei teleiwsei, midenizoume ola ta temp kai pame apo tin arxi
     {   xtemp = "";
         ytemp = "";
         gyroZtemp = "";
```

```
    }

    pointer = recieved_char;

    if (xtemp!="") {
        x=(str2flt(xtemp))/100;
        xtemp = "";
        }

   if (ytemp!="") {
        y=(str2flt(ytemp))/100;
        ytemp = "";

     }



    if (gyroZtemp!="") {
        gyroZ=(str2flt(gyroZtemp));
        gyroZtemp = "";
   }

   if (recieved_char == 'e'){
        pointer=' ';
        peekchr=0;
        PID();
   }
 }

 else
 {
   switch (pointer){
     case 'x':
       xtemp+=recieved_char;
       break;
     case 'y':
       ytemp+=recieved_char;
       break;
     case 'z':
       gyroZtemp+=recieved_char;
       break;
      }

  }

 }

}
```

```
float str2flt (String readstring)
{
   char carray[readstring.length() + 1];           //determine size of the array
   readstring.toCharArray(carray, sizeof(carray)); //put readStringinto an array
   float n = atof(carray);                         //convert the array into a Float
   return n;
}



 void rc_yaw_interupt()  {     //---------------------------------YAW Interrupt----------------------
-----
 if(int0)
   count0=micros(); // we got a positive edge
 else
   yaw_pulse=micros()-count0; // Negative edge: get pulsewidth
}


 void rc_thro_interupt()  {    //---------------------------------THROTTLE Interrupt---------------
------------
 if(int1)
   count1=micros(); // we got a positive edge
 else
  {
  thro_pulse=micros()-count1; // Negative edge: get pulsewidth
  if (thro_pulse<1150) thro_pulse=1050;
  else thro_pulse=map (thro_pulse, 1150, 2000, 1400, 1650);
 }
}


 void rc_pitch_interupt()  {    //---------------------------------PITCH Interrupt--------------------
------
 if(int2)
   count2=micros(); // we got a positive edge
 else
   pitch_pulse=micros()-count2; // Negative edge: get pulsewidth
}

 void rc_roll_interupt()  {     //---------------------------------ROLL Interrupt----------------------
-----
 if(int3)
   count3=micros(); // we got a positive edge
 else
   roll_pulse=micros()-count3; // Negative edge: get pulsewidth
}
```

```
void rc_sw_interupt()   {     //----------------------------------SWITCH Interrupt--------------------
-------
 if(int4)
   count4=micros(); // we got a positive edge
 else
  RC_switch_value=micros()-count4; // Negative edge: get pulsewidth
}
```

```
 void filter_RC () {        //----------------------------------FILTER RC--------------------------

 if (yaw_pulse>0) {             //an exoume lavei enan palmo sto pin tou yaw (pin 21)
   tmp_time_y[jy]=yaw_pulse;    //na valei tin timi stin jy thesi tou pinaka
   jy++;                        //opou 0<=jy<=1 oi 2 theseis tou pinaka
   if (jy>1) jy=0;
   yaw_setpoint=0;
   for (ky=0; ky<2; ky++)  yaw_setpoint+=tmp_time_y[ky];
   yaw_setpoint=yaw_setpoint/2;
   if (( yaw_setpoint < 1550) &&  ( yaw_setpoint > 1450))   yaw_setpoint= 1500;
   yaw_pulse=0;
 }


 if (thro_pulse>0) {           //throttle (pin 20)
   tmp_time_t[jt]=thro_pulse;
   jt++;
   if (jt>1) jt=0;
   thro_setpoint=0;
   for (kt=0; kt<2; kt++)  thro_setpoint+=tmp_time_t[kt];
   thro_setpoint= thro_setpoint/2;
   thro_pulse=0;
 }


 if (roll_pulse>0) {           //roll (pin 19)
   tmp_time_r[jr]=roll_pulse;
   jr++;
   if (jr>1) jr=0;
   roll_setpoint=0;
   for (kr=0; kr<2; kr++)  roll_setpoint+=tmp_time_r[kr];
   roll_setpoint = (roll_setpoint / 2) + 50;
   if (( roll_setpoint < 1550) &&  ( roll_setpoint > 1450)) roll_setpoint= 1500;
   roll_setpoint=map (roll_setpoint, 1000, 2000, -20, 20);
   roll_pulse=0;
 }
```

```
 if (pitch_pulse>0) {          //pitch (pin 18)
   tmp_time_p[jp]=pitch_pulse;
   jp++;
   if (jp>1) jp=0;
   pitch_setpoint=0;
   for (kp=0; kp<2; kp++)  pitch_setpoint+=tmp_time_p[kp];
   pitch_setpoint = pitch_setpoint / 2;
   if (( pitch_setpoint < 1550) &&  ( pitch_setpoint > 1450)) pitch_setpoint= 1500;
   pitch_setpoint= map (pitch_setpoint, 1000, 2000, -20, 20);
   pitch_pulse=0;
 }

}



void send_PWM() {

 valESCyaw_final = valESCyaw+thro_setpoint;
 valESCright_final = valESCright+thro_setpoint;
 valESCleft_final = valESCleft+thro_setpoint;

 valESCyaw_final = constrain (valESCyaw_final, 1050, 1900);
 valESCright_final = constrain (valESCright_final, 1050, 1900);
 valESCleft_final = constrain (valESCleft_final, 1050, 1900);

 OCR3A = valESCright_final;
 OCR4A = valESCyaw_final;
 OCR4B = valESCleft_final;
 OCR4C = valservo;
}



 void serial_COMM() {

if (Serial2.available() > 0){
  char command = Serial2.read();
 switch (command) {
   case '1':   Kp_x+=0.01;            break;
   case 'q':   Kp_x=Kp_x-0.01;        break;
   case '2':   Ki_x+=0.01;            break;
   case 'w':   Ki_x=Ki_x-0.01;        break;
   case '3':   Kd_x+=0.1;             break;
   case 'e':   Kd_x+=-0.1;            break;
   case '4':   I_limit_x+=1;          break;
   case 'r':   I_limit_x=I_limit_x-1;  break;
```

```
    case '5':   Kp_y+=0.01;           break;
    case 't':   Kp_y=Kp_y-0.01;       break;
    case '6':   Ki_y+=0.01;           break;
    case 'y':   Ki_y=Ki_y-0.01;       break;
    case '7':   Kd_y+=0.1;            break;
    case 'u':   Kd_y=Kd_y-0.1;        break;
    case '8':   I_limit_y+=1;         break;
    case 'i':   I_limit_y=I_limit_y-1;  break;
    case 's':   Kp_z+=0.01;           break;
    case 'x':   Kp_z=Kp_z-0.01;       break;
    case 'd':   Ki_z+=0.01;           break;
    case 'c':   Ki_z=Ki_z-0.01;       break;
    case 'f':   I_limit_z+=1;         break;
    case 'v':   I_limit_z=I_limit_z-1;  break;
  }
}

// Angles
nums[0]=(x);
nums[1]=(y);

// Gyros
nums[2]=(gyroX);
nums[3]=(gyroY);
nums[4]=(gyroZ);

//Errors
nums[5]=(error_x);
nums[6]=(error_y);
nums[7]=(error_z);

//Setpoints
nums[8]=(roll_setpoint);
nums[9]=(pitch_setpoint);
nums[10]=(yaw_setpoint);
nums[11]=(RC_switch_value);

//Outputs
nums[12]=(valESCright_final);
nums[13]=(valESCleft_final);
nums[14]=(valESCyaw_final);
nums[15]=(valservo);

// PID of X
nums[16]=(Kp_x);
nums[17]=(Ki_x);
nums[18]=(Kd_x*10);
nums[19]=(I_limit_x);
nums[20]=(integral_x);
```

```
// PID of Y
nums[21]=(Kp_y);
nums[22]=(Ki_y);
nums[23]=(Kd_y*10);
nums[24]=(I_limit_y);
nums[25]=(integral_y);

// PID of Z
nums[26]=(Kp_z);
nums[27]=(Ki_z);
nums[28]=(Kd_z*10);
nums[29]=(I_limit_z);
nums[30]=(integral_z);

nums[31]=(thro_setpoint);

nums[32]=(temp_time_holder);
nums[33]=(battery_mes);
nums[34]=(batt_time);


Serial2.print(nums[serial_send_counter]);Serial2.print(",");
serial_send_counter++;

if (serial_send_counter>34) {
   serial_send_counter=0;
   Serial2.println ();
}

}
```

Πηγαίος κώδικας στο Arduino Pro Mini:

```
#include <I2Cdev.h>
#include <MPU60X0.h>
#include <EEPROM.h>

//#define DEBUG
#include "DebugUtils.h"
//#include "CommunicationUtils.h"
#include "FreeIMU.h"
#include <Wire.h>
#include <SPI.h>



int raw_values[9];
float ypr[3]; // yaw pitch roll
```

```
FreeIMU TRI_IMU = FreeIMU();


long time;
long time2;
long initZ=0;
int initZi=0;
int LEDi;


void setup() {
  Serial.begin(115200);
  Wire.begin();
  TRI_IMU.init(true);

  // LED
  pinMode(13, OUTPUT);


  time=millis();
  time2=millis();


  while (millis()-time <30000){
        TRI_IMU.getRawValues(raw_values);
        TRI_IMU.getYawPitchRoll(ypr);
//        Serial.print("x");
        Serial.print(int(ypr[2]));
        Serial.print(",");
        Serial.print(int(ypr[1]));
        Serial.print(",");
        Serial.print(int(raw_values[5])/5);
        Serial.println(",");
        initZ+=int(raw_values[5]/5);
        initZi++;

if (millis() - time2 > 50  ){
      LEDi++;
      if (LEDi == 1) {
        digitalWrite(13, HIGH);
      }
      if (LEDi == 2) {
        digitalWrite(13, LOW);
      LEDi = 0;
      }
      time2 = millis();
   }
```

```
  }

initZ=initZ/initZi;
}


void loop() {


      TRI_IMU.getRawValues(raw_values);
      TRI_IMU.getYawPitchRoll(ypr);
      Serial.print("z");
      Serial.print((raw_values[5]/5)-initZ);
      Serial.print("x");
      Serial.print(int(ypr[2]*100));
      Serial.print("y");
      Serial.print(int(ypr[1]*100));
      Serial.println("e");

if (millis() - time2 > 500  ){
     LEDi++;
     if (LEDi == 1) {
       digitalWrite(13, HIGH);
     }
     if (LEDi == 2) {
       digitalWrite(13, LOW);
     LEDi = 0;
     }
     time2 = millis();
  }

  }
```