ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΗ-ΕΠΑΝΔΡΩΜΕΝΟΥ ΙΠΤΑΜΕΝΟΥ ΟΧΗΜΑΤΟΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΓΕΩΡΓΙΑ ΑΚΡΙΒΕΙΑΣ

DRONE PROGRAMMING AND AN APPLICATION IN PRECISIONS AGRICULTURE

ΒΟΤΣΗΣ ΕΥΣΤΑΘΙΟΣ Α.Ε.Μ: 382

Επιβλέπων Καθηγητής: Δημήτριος Κατσαρός

Δεύτερο μέλος Επιτροπής: Θανάσης Κοράκης

ΒΟΛΟΣ 2015

Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου
Θεσσαλίας

Βότσης Χ. Ευστάθιος
Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών,
Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Για οποιαδήποτε χρήση του εμπεριεχόμενου υλικού επικοινωνήστε με τον συγγραφέα στην διεύθυνση: evotsis@inf.uth.gr

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά όλα τα μέλη της οικογένειας μου, που συνέβαλαν όλοι με τον τρόπο τους στο να πετύχω τους στόχους μου. Χωρίς αυτούς αδιαμφισβήτητα δε θα τα είχα καταφέρει. Ξεχωριστές ευχαριστίες στους γονείς μου και στον αδερφό μου για την υποστήριξη, συμπαράσταση και υπομονή που έδειξαν όλα αυτά τα χρόνια.

Επίσης θέλω να εκφράσω τις ευχαριστίες και την εκτίμηση μου σε όλο το τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας για την συνεργασία που είχα μαζί τους και για το έργο που έχουν επιτελέσει όλα αυτά τα χρόνια.

Συνάμα να εκφράσω τις ευχαριστίες μου στον κύριο Δημήτριο Κατσαρό που εμπνεύστηκε και δέχθηκε να συμμετάσχει στην εξεταστική επιτροπή αυτής της εργασίας, καθώς και στην κυρία Δασκαλοπούλου Ασπασία για την άψογη συνεργασία μας στο παρελθόν.

Τέλος , θέλω να ευχαριστήσω τους απόφοιτους Αλέξανδρο Σπαή, Παντελή Γιαζιτζή, Τάσο Σκούρτη, Φίλιππο Ζαμπούνη, Γεώργιο Τάτση καθώς και τους Γεώργιο Χατζή, Μανώλη Ζερβό, Νίκο Βασιλείου και Παναγιώτη Παζιάνα για την άψογη συνεργασία και παρέα μαζί τους όλα αυτά τα χρόνια στην πόλη του Βόλου.

Σας ευχαριστώ όλους
Βότσης Στάθης

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1: Εισαγωγή

1.1 Τι είναι τα DRONE-UAV, ποια η χρησιμότητα τους και εφαρμογές στον σύνγχρονο τρόπο ζωής

1.2 Αντικείμενο Εργασίας

Κεφάλαιο 2: Πλαίσιο, Μηχανικά & Ηλεκτρικά μέρη

2.1 Πλαίσιο

2.2 Μοτέρ(Brushless Motor) – Ρυθμιστές Ταχύτητας(Speed Controller)

2.3 Πλακέτα διανομής ρεύματος (Power Distribution Board) – Μπαταρία τροφοδοσίας

Κεφάλαιο 3: Μετάδοση Εικόνας & Εξαρτήματα

3.1 Camera

3.2 Video transmitter & receiver-Monitor

3.3 Antenna Tracker

Κεφάλαιο 4: Συστήματα Τηλεκατεύθυνσης & Πλοήγησης

4.1 Τηλεκατεύθυνση – Προσαρμογή UHF πομπού (433 mhz)

4.2 Αυτόματος Πιλότος

4.3 Σύστημα πλοήγησης εδάφους (Ground Control Station)

Κεφάλαιο 5: RLC Φίλτρα

5.1 Τι είναι τα RLC φίλτρα

5.2 Ποια η χρησιμότητα και τα οφέλη τους στο συγκεκριμένο project

Κεφάλαιο 6: Η Πρόοδος του Project σε Βίντεο

6.1 Δοκιμή πτητικής ικανότητας drone

6.2 Δοκιμή αυτόματης προσγείωσης drone

6.3 Καταγραφή αγροτικής περιοχής από το drone

6.4 Καταγραφή αγροτικής περιοχής με ακρίβεια τηλεμετρίας

6.5 Συντριβή σε κατοικημένη περιοχή

Κεφάλαιο 7: Βιβλιογραφία

Κεφάλαιο 1: Εισαγωγή

1.1 Τι είναι τα DRONE-UAV, ποια η χρησιμότητα τους και εφαρμογές στον σύνγχρονο τρόπο ζωής

Ένα μη επανδρωμένο εναέριο όχημα (UAV), κοινώς γνωστό ως **drone** το οποίο αναφέρεται και ως **Remotely Piloted Aircraft** (**RPA**) από τον Διεθνή Οργανισμό Πολιτικής Αεροπορίας (ICAO), είναι ένα αεροσκάφος χωρίς ανθρώπινη παρουσία επί του σκάφους. Υπάρχουν διάφορα είδη drones. Αρχικά τα UAS (μη-επανδρωμένα Air System), UAV (μη επανδρωμένα οχήματα αέρος) και τα RPAS (Remote Piloted Aircraft Systems). Η πτήση τους ελέγχεται είτε αυτόνομα από τους υπολογιστές επί του σκάφους ή από τον απομακρυσμένο έλεγχο ενός πιλότου στο έδαφος ή σε άλλο όχημα. Ιστορικά, τα UAV ξεκίνησαν ως ασύρματα τηλεκατευθυνόμενα πιλοτικά αεροσκάφη, αλλά η ραγδαία εξέλιξη της τεχνολογίας κατέστησε δυνατό τον αυτοματισμό πολλών συστημάτων τους.

Συνήθως αναπτύσσονται για στρατιωτικές και ειδικές εφαρμογές λειτουργίας, αλλά χρησιμοποιούνται επίσης σε ένα μικρό αλλά αυξανόμενο αριθμό μη στρατιωτικών εφαρμογών, όπως η αστυνόμευση, πυρόσβεση και η επιθεώρηση της ενέργειας αγωγών. Τα UAVs συχνά προτιμούνται για αποστολές στις οποίες είναι πάρα πολύ επικίνδυνη η ανθρώπινη παρουσία. Μπορούν να χρησιμοποιηθούν σε αποστολές έρευνας και διάσωσης, βοηθώντας στο να βρεθούν άνθρωποι οι οποίοι έχουν χαθεί, ή επιπλέουν στην θάλασσα, ψάχνοντας σε σημεία που για τον άνθρωπο είναι επικίνδυνα και δύσκολα προσβάσιμα.

Τέλος τα τελευταία χρόνια αρχίσανε να χρησιμοποιούνται και σε ερασιτεχνικό επίπεδο με εφαρμογές στην καθημερινή ζωή όπως αεροβιντεολήψεις-αεροφωτογραφίες καθώς και 3D σκανάρισμα χώρων και εγκαταστάσεων.

ΕΙΚΟΝΑ 1

UAV HEXACOPTER WITH 3D SCANNER ON BOARD

https://www.youtube.com/watch?v=Mz5z_xaB9i8

1.2 Αντικείμενο Εργασίας

Σκοπός αυτής της εργασίας ήταν ο προγραμματισμός και η κατασκευή ενός μη επανδρωμένου εξακοπτέρου (**hexacopter**), το οποίο έπειτα χρησιμοποιήθηκε , στην εναέρια παρακολούθηση αγροτικών εκτάσεων στην ευρύτερη περιοχή των Φαρσάλων. Στο τέλος της θα παρουσιασθεί σχετικό video όπου το εξακόπτερό μας απογειώνεται από κατοικημένη περιοχή και αφού φτάσει στο επιλεγμένο αγροτικό τεμάχιο καλύπτει περιμετρικά τις αποστάσεις που θέλουμε ώστε να βγάλουμε τα συμπεράσματα μας απλά με το πάτημα ενός κουμπιού.

Κεφάλαιο 2: Πλαίσιο, Μηχανικά & Ηλεκτρικά μέρη

2.1 Πλαίσιο

Το πλαίσιο που χρησιμοποιήθηκε για την υλοποίηση αυτού του project (τύπου hexa) είναι κατασκευασμένο από ανθρακονήματα (carbon) ώστε να ελαχιστοποιηθεί το βάρος στο μέγιστο δυνατό.

Αποτελείται από δύο κεντρικές πλάκες πάνω στις οποίες προσαρμόστηκαν έξι πόδια με γωνία ανάμεσά τους 60 μοίρες και διαμέτρου 80cm. Με αυτό τον τρόπο έχουμε έξι συμμετρικά τοποθετημένα πόδια πάνω στα οποία προσαρμόστηκαν τα ηλεκτρικά μοτέρ.

ΕΙΚΟΝΑ 2

CARBON FRAME HEXACOPTER 800mm

2.2 Μοτέρ(Brushless Motor) – Ρυθμιστές Ταχύτητας(Speed Controller)

Σύμφωνα με τους υπολογισμούς που γίνανε στην αρχή του project το τελικό βάρος του εξακοπτέρου υπολογίστηκε στα 3.5 κιλά. Για τον λόγο αυτό χρησιμοποιήθηκαν μοτέρ (brushless motor) άνωσης 2.2 κιλά (σε 100% throttle) έκαστο ώστε να είναι το εξακόπτερο ευέλικτο και γρήγορο. Τα συγκεκριμένα μοτέρ λειτουργούνε στα 22.2 Volt με μέση κατανάλωση 5.9A.

Σε συνδυασμό με τα ηλεκτρικά μοτέρ χρησιμοποιηθήκανε ρυθμιστές ταχύτητας έτσι ώστε να μπορούμε να δίνουμε τις κατάλληλες στροφές για να πετύχουμε σταθερή κίνηση στο εξακόπτερο μας & ανύψωση. Χρησιμοποιηθήκανε για ασφάλεια 40 A (speed controller) διότι τα μοτέρ σε 100% throttle καταναλώνουν πάνω από 20 A το καθένα.

ΕΙΚΟΝΑ 3

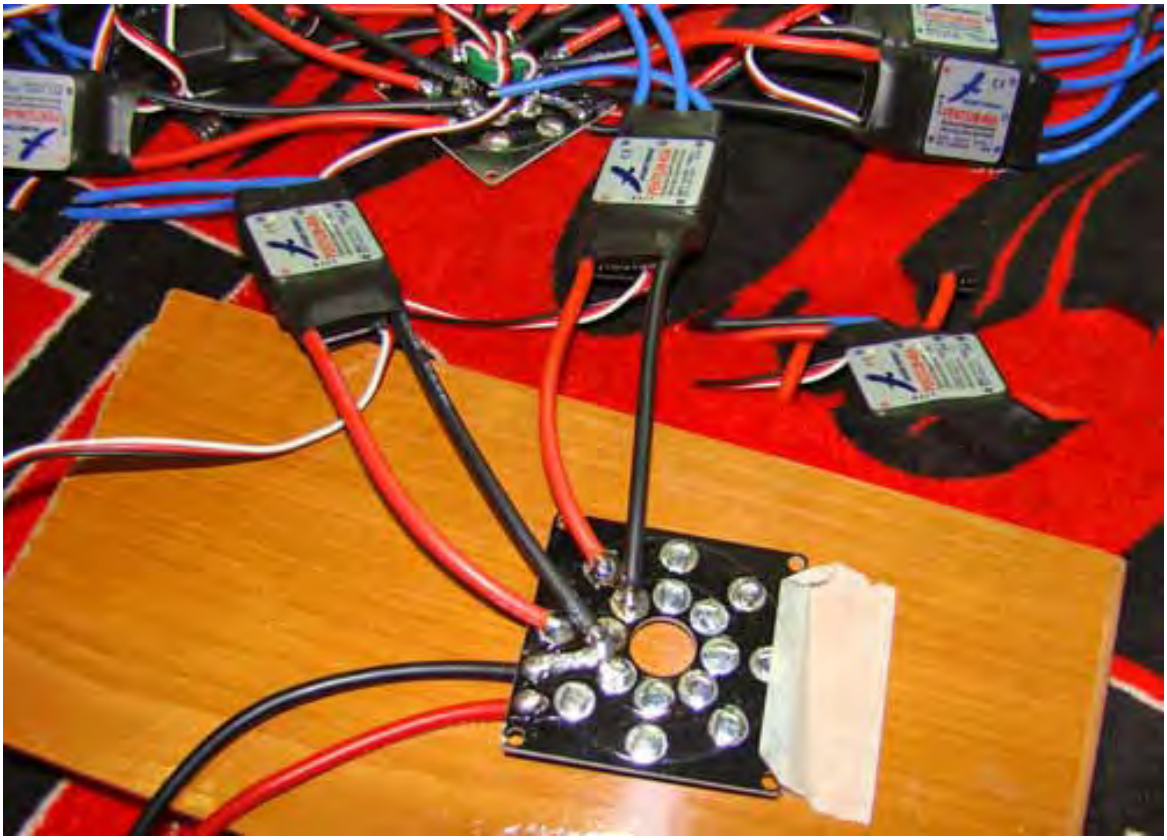
2.3 Πλακέτα διανομής ρεύματος (Power Distribution Board) – Μπαταρία τροφοδοσίας

Για την τροφοδοσία ρεύματος του κάθε ηλεκτρικού μοτέρ χρησιμοποιήθηκε πλακέτα διανομής ρεύματος και μία εξάσελη μπαταρία λιθίου 22.2 Volt 8000mah.

Η μπαταρία συνδέθηκε με τη χρήση βύσματος στην πλακέτα και όλοι οι ρυθμιστές ταχύτητας συνδεθήκανε απευθείας σε αυτή, έτσι ώστε να τροφοδοτούνται με ρεύμα τα ηλεκτρικά μοτέρ.

ΕΙΚΟΝΑ 4

CONNECTION DISTRIBUTION BOARD AND ESC



Κεφάλαιο 3: Μετάδοση Εικόνας & Εξαρτήματα

3.1 Camera (Goprohero 3 Black edition)

Για την μετάδοση εικόνας από το drone χρησιμοποιήθηκε κάμερα υψηλής ανάλυσης. Η Gopro 3 Black την δεδομένη στιγμή είναι από τις καλύτερες κάμερες της αγοράς με ποικίλες δυνατότητες. Συνδύαζει υψηλή ποιότητα video, απομακρυσμένο έλεγχό της με smartphone-tablet μέσω wifi και πάρα πολύ μικρό βάρος (76gr) το οποίο είναι απαραίτητο σε τέτοιου είδους εφαρμογές.

ΕΙΚΟΝΑ 5

GOPROHERO 3 BLACK EDITION

https://www.youtube.com/watch?v=3wbvpOIIBQA#t=18

3.2 Video transmitter & receiver-Monitor

Σε συνδυασμό με την Gopro χρησιμοποιήθηκε πομπός 9 καναλιών μετάδοσης video ισχύος 1.5 watt στα 1.2 ghz. Η gopro συνδέθηκε μέσω ειδικής θύρας με τον πομπό για να λαμβάνουμε στον δέκτη μας live εικόνα από το drone. Η εμβέλεια του συστήματος σε συνδυασμό με κεραία omnidirectional 9 dbi στον πομπό και κεραία τύπου yagi 13 dbi στον δέκτη φτάνει τα 20 χλμ. Τέλος ο δέκτης συνδέθηκε με monitor 7 inch.

ΕΙΚΟΝΑ 6

1.2 GHZ TX WITH GOPRO & LIPO 11.1 V

EIKONA 7

1.2 GHZ RX WITH TFT MONITOR AND YAGI ANTENNA

EIKONA 8

1.2 GHZ RX YAGI ANTENNA 13 DBI

3.3 Antenna Tracker

Σύστημα παρακολούθησης της κεραίας που ακολουθεί μια πηγή σήματος .

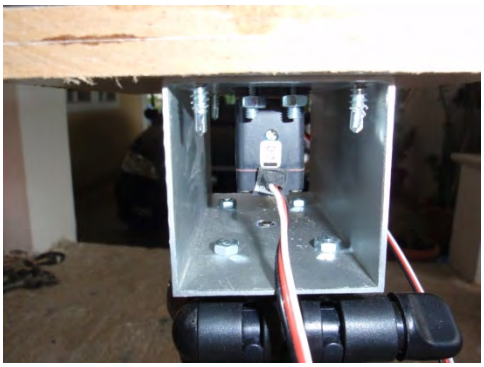
Στο συγκεκριμένο project η θέση του ιπτάμενου οχήματος μας, μεταδίδεται μέσω του πομπού της κάμερας μας από το κανάλι του ήχου και λαμβάνεται στη βάση μέσω του αντίστοιχου δέκτη, το οποίο επεξεργάζεται από έναν controller, ο οποίος οδηγεί δύο μηχανισμούς servo και στρέφει την κεραία μας στο σημείο που είναι το drone μας για καλύτερη λήψη.

Η χρησιμότητα του είναι απαραίτητη, εφόσων η κεραία στη βάση όπου γίνεται η λήψη της εικόνας είναι κατευθυντική.

Το συγκεκριμένο κομμάτι της εργασίας αποτελεί μέρος ειδικού θέματος που υλοποιήθηκε το εαρινό εξάμηνο 2011-2012 αλλά είναι σημαντικό κομμάτι ενός UAV και πρέπει να αναφερθεί.

EIKONA 9

EIKONA 10-11



https://www.youtube.com/watch?v=MiSJfS3BSQQ&list=UUjvHp-c_Pf3cezOFAKA9inQ

Κεφάλαιο 4: Συστήματα Τηλεκατεύθυνσης & Πλοήγησης

4.1 Τηλεκατεύθυνση – Προσαρμογή UHF πομπού (433 mhz)

Τηλεκατεύθυνση ελέγχου (συχνά με τα αρχικά RC) είναι η χρήση των ραδιοκυμάτων με σκοπό τον έλεγχο μια συσκευής από απόσταση. Η τηλεκατεύθυνση ελέγχου χρησιμοποιείται για τον χειροκίνητο έλεγχο οχημάτων με ραδιοφωνικό πομπό, π.χ. βιομηχανικό και στρατιωτικό εξοπλισμό.

Για την υλοποίηση του project αυτού χρησιμοποιήθηκε τηλεκατεύθυνση 9 καναλιών, με προσαρμογή πομπού UHF στα 433mhz, για την επίτευξη απομακρυσμένου ελέγχου έως 40 km.

ΕΙΚΟΝΑ 13

9 channel tx with 433 mhz module

4.2 Αυτόματος Πιλότος

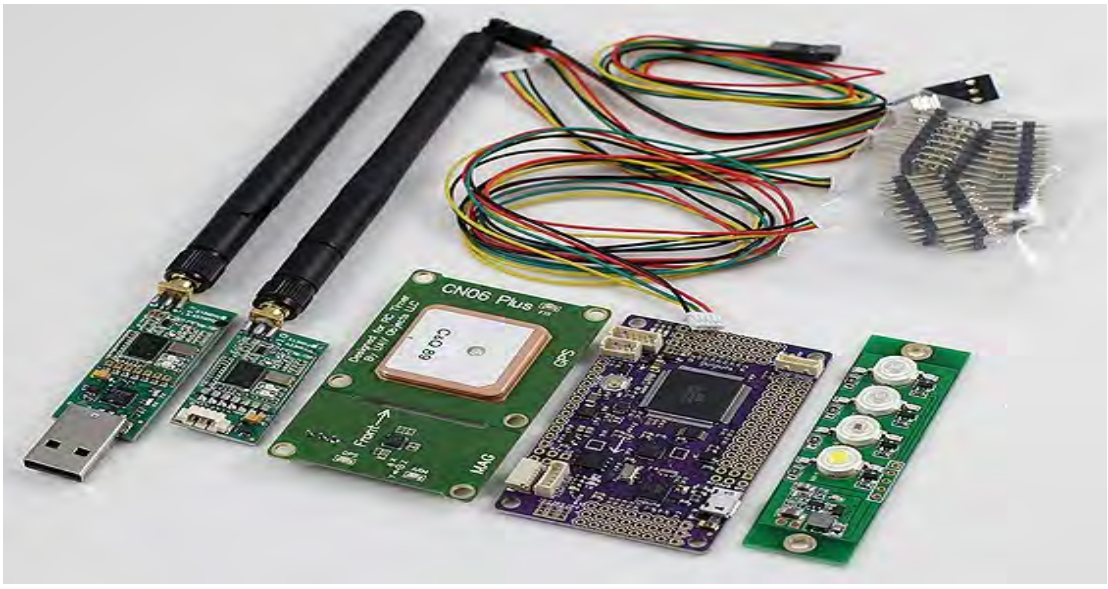
Το ArduCopter είναι μία open source πλατφόρμα μη επανδρωμένων εναέριων οχημάτων, που δημιουργήθηκαν από την κοινότητα DIY Drones βασισμένη στην πλατφόρμα Arduino.

Είναι μία μικρή all-in-one μονάδα που συνδυάζει την MC, ένα γυροσκόπιο, επιταχυνσιόμετρο, και ένα βαρομετρικό αλτίμετρο. Με την προαιρετική μονάδα GPS και Πυξίδας, βρίσκει και καταγράφει στην MC την ακριβή θέση του ιπτάμενου αντικειμένου. Επίσης μπορεί να μετρήσει το ύψος πτήσης και ως εκ τούτου μπορεί να χρησιμοποιηθεί για αυτόματο πιλότο και αυτόματη οδήγηση.

Με τον πρωτοποριακό σχεδιασμό All-in-One, περιέχει ελεγκτές, 3 αξόνων γυροσκόπιο και βαρόμετρο στο φως. Το ArduCopter περιέχει σύνθετο αλγόριθμο καθοδήγησης , αλγόριθμο ελέγχου και παρέχει εξαιρετική σταθερότητα και ευελιξία κατά τη διάρκεια της πτήσης.

ΕΙΚΟΝΑ 14

AUTOPILOT WITH TELEMETRY



RTH MODE CODE

```
/// -*- tab-width: 4; Mode: C++; c-basic-offset: 4; indent-tabs-mode: nil -*-
/*
* control_rtl.pde - init and run calls for RTL flight mode
*
* There are two parts to RTL, the high level decision making which controls which state we are in
* and the lower implementation of the waypoint or landing controllers within those states
*/
// rtl_init - initialise rtl controller
static bool rtl_init(bool ignore_checks)
{
if (GPS_ok() || ignore_checks) {
rtl_climb_start();
return true;
}else{
return false;
}
}
// rtl_run - runs the return-to-launch controller
// should be called at 100hz or more
static void rtl_run()
{
// check if we need to move to next state
if (rtl_state_complete) {
switch (rtl_state) {
case InitialClimb:
rtl_return_start();
break;
case ReturnHome:
rtl_loiterathome_start();
break;
case LoiterAtHome:
if (g.rtl_alt_final > 0 && !failsafe.radio) {
rtl_descent_start();
}else{
rtl_land_start();
}
break;
```

```
case FinalDescent:
// do nothing
break;
case Land:
// do nothing - rtl_land_run will take care of disarming motors
break;
}
}
// call the correct run function
switch (rtl_state) {
case InitialClimb:
rtl_climb_return_run();
break;
case ReturnHome:
rtl_climb_return_run();
break;
case LoiterAtHome:
rtl_loiterathome_run();
break;
case FinalDescent:
rtl_descent_run();
break;
case Land:
rtl_land_run();
break;
}
}
// rtl_climb_start - initialise climb to RTL altitude
static void rtl_climb_start()
{
rtl_state = InitialClimb;
rtl_state_complete = false;
// initialise waypoint and spline controller
wp_nav.wp_and_spline_init();
// get horizontal stopping point
Vector3f destination;
wp_nav.get_wp_stopping_point_xy(destination);
#if AC_RALLY == ENABLED
// rally_point.alt will be the altitude of the nearest rally point or the RTL_ALT. uses
```

```
absolute altitudes
Location rally_point = rally.calc_best_rally_or_home_location(current_loc, get_RTL_alt()+ahrs.get_home().alt);
rally_point.alt -= ahrs.get_home().alt; // convert to altitude above home
rally_point.alt = max(rally_point.alt, current_loc.alt); // ensure we do not descend before reaching home
destination.z = rally_point.alt;
#else
destination.z = get_RTL_alt();
#endif

// set the destination
wp_nav.set_wp_destination(destination);
wp_nav.set_fast_waypoint(true);

// hold current yaw during initial climb
set_auto_yaw_mode(AUTO_YAW_HOLD);
}

// rtl_return_start - initialise return to home
static void rtl_return_start()
{
rtl_state = ReturnHome;
rtl_state_complete = false;

// set target to above home/rally point
#if AC_RALLY == ENABLED
// rally_point will be the nearest rally point or home. uses absolute altitudes
Location rally_point = rally.calc_best_rally_or_home_location(current_loc, get_RTL_alt()+ahrs.get_home().alt);
rally_point.alt -= ahrs.get_home().alt; // convert to altitude above home
rally_point.alt = max(rally_point.alt, current_loc.alt); // ensure we do not descend before reaching home
Vector3f destination = pv_location_to_vector(rally_point);
#else
Vector3f destination = Vector3f(0,0,get_RTL_alt());
#endif

wp_nav.set_wp_destination(destination);

// initialise yaw to point home (maybe)
set_auto_yaw_mode(get_default_auto_yaw_mode(true));
}

// rtl_climb_return_run - implements the initial climb, return home and descent portions of RTL which all rely on the wp controller
// called by rtl_run at 100hz or more
static void rtl_climb_return_run()
```

```
{
// if not auto armed set throttle to zero and exit immediately
if(!ap.auto_armed) {
// reset attitude control targets
attitude_control.relax_bf_rate_controller();
attitude_control.set_yaw_target_to_current_heading();
attitude_control.set_throttle_out(0, false);
// To-Do: re-initialise wpnav targets
return;
}
// process pilot's yaw input
float target_yaw_rate = 0;
if (!failsafe.radio) {
// get pilot's desired yaw rate
target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
if (target_yaw_rate != 0) {
set_auto_yaw_mode(AUTO_YAW_HOLD);
}
}
// run waypoint controller
wp_nav.update_wpnav();
// call z-axis position controller (wpnav should have already updated it's alt target)
pos_control.update_z_controller();
// call attitude controller
if (auto_yaw_mode == AUTO_YAW_HOLD) {
// roll & pitch from waypoint controller, yaw rate from pilot
attitude_control.angle_ef_roll_pitch_rate_ef_yaw(wp_nav.get_roll(),
wp_nav.get_pitch(), target_yaw_rate);
}else{
// roll, pitch from waypoint controller, yaw heading from auto_heading()
attitude_control.angle_ef_roll_pitch_yaw(wp_nav.get_roll(), wp_nav.get_pitch(),
get_auto_heading(),true);
}
// check if we've completed this stage of RTL
rtl_state_complete = wp_nav.reached_wp_destination();
}
// rtl_return_start - initialise return to home
static void rtl_loiterathome_start()
{
rtl_state = LoiterAtHome;
```

```
rtl_state_complete = false;
rtl_loiter_start_time = millis();
// yaw back to initial take-off heading yaw unless pilot has already overridden yaw
if(get_default_auto_yaw_mode(true) != AUTO_YAW_HOLD) {
set_auto_yaw_mode(AUTO_YAW_RESETTOARMEDYAW);
} else {
set_auto_yaw_mode(AUTO_YAW_HOLD);
}
}
// rtl_climb_return_descent_run - implements the initial climb, return home and descent portions of RTL which all rely on the wp controller
// called by rtl_run at 100hz or more
static void rtl_loiterathome_run()
{
// if not auto armed set throttle to zero and exit immediately
if(!ap.auto_armed) {
// reset attitude control targets
attitude_control.relax_bf_rate_controller();
attitude_control.set_yaw_target_to_current_heading();
attitude_control.set_throttle_out(0, false);
// To-Do: re-initialise wpnav targets
return;
}
// process pilot's yaw input
float target_yaw_rate = 0;
if (!failsafe.radio) {
// get pilot's desired yaw rate
target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
if (target_yaw_rate != 0) {
set_auto_yaw_mode(AUTO_YAW_HOLD);
}
}
// run waypoint controller
wp_nav.update_wpnav();
// call z-axis position controller (wpnav should have already updated it's alt target)
pos_control.update_z_controller();
// call attitude controller
if (auto_yaw_mode == AUTO_YAW_HOLD) {
// roll & pitch from waypoint controller, yaw rate from pilot
attitude_control.angle_ef_roll_pitch_rate_ef_yaw(wp_nav.get_roll(),
```

```
wp_nav.get_pitch(), target_yaw_rate);
}else{
// roll, pitch from waypoint controller, yaw heading from auto_heading()
attitude_control.angle_ef_roll_pitch_yaw(wp_nav.get_roll(), wp_nav.get_pitch(),
get_auto_heading(),true);
}
// check if we've completed this stage of RTL
if ((millis() - rtl_loiter_start_time) >= (uint32_t)g.rtl_loiter_time.get()) {
if (auto_yaw_mode == AUTO_YAW_RESETTOARMEDYAW) {
// check if heading is within 2 degrees of heading when vehicle was armed
if (labs(wrap_180_cd(ahrs.yaw_sensor-initial_armed_bearing)) <= 200) {
rtl_state_complete = true;
}
} else {
// we have loitered long enough
rtl_state_complete = true;
}
}
}
// rtl_descent_start - initialise descent to final alt
static void rtl_descent_start()
{
rtl_state = FinalDescent;
rtl_state_complete = false;
// Set wp navigation target to above home
wp_nav.init_loiter_target(wp_nav.get_wp_destination());
// initialise altitude target to stopping point
pos_control.set_target_to_stopping_point_z();
// initialise yaw
set_auto_yaw_mode(AUTO_YAW_HOLD);
}
// rtl_descent_run - implements the final descent to the RTL_ALT
// called by rtl_run at 100hz or more
static void rtl_descent_run()
{
int16_t roll_control = 0, pitch_control = 0;
float target_yaw_rate = 0;
// if not auto armed set throttle to zero and exit immediately
if(!ap.auto_armed || !inertial_nav.position_ok()) {
attitude_control.relax_bf_rate_controller();
```

```
attitude_control.set_yaw_target_to_current_heading();
attitude_control.set_throttle_out(0, false);
// set target to current position
wp_nav.init_loiter_target();
return;
}
// process pilot's input
if (!failsafe.radio) {
if (g.land_repositioning) {
// apply SIMPLE mode transform to pilot inputs
update_simple_mode();
// process pilot's roll and pitch input
roll_control = g.rc_1.control_in;
pitch_control = g.rc_2.control_in;
}
// get pilot's desired yaw rate
target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
}
// process roll, pitch inputs
wp_nav.set_pilot_desired_acceleration(roll_control, pitch_control);
// run loiter controller
wp_nav.update_loiter(ekfGndSpdLimit, ekfNavVelGainScaler);
// call z-axis position controller
pos_control.set_alt_target_with_slew(g.rtl_alt_final, G_Dt);
pos_control.update_z_controller();
// roll & pitch from waypoint controller, yaw rate from pilot
attitude_control.angle_ef_roll_pitch_rate_ef_yaw(wp_nav.get_roll(),
wp_nav.get_pitch(), target_yaw_rate);
// check if we've reached within 20cm of final altitude
rtl_state_complete = fabs(g.rtl_alt_final - inertial_nav.get_altitude()) < 20.0f;
}
// rtl_loiterathome_start - initialise controllers to loiter over home
static void rtl_land_start()
{
rtl_state = Land;
rtl_state_complete = false;
// Set wp navigation target to above home
wp_nav.init_loiter_target(wp_nav.get_wp_destination());
// initialise altitude target to stopping point
```

```
pos_control.set_target_to_stopping_point_z();
// initialise yaw
set_auto_yaw_mode(AUTO_YAW_HOLD);
}
// rtl_returnhome_run - return home
// called by rtl_run at 100hz or more
static void rtl_land_run()
{
int16_t roll_control = 0, pitch_control = 0;
float target_yaw_rate = 0;
// if not auto armed set throttle to zero and exit immediately
if(!ap.auto_armed || ap.land_complete) {
attitude_control.relax_bf_rate_controller();
attitude_control.set_yaw_target_to_current_heading();
attitude_control.set_throttle_out(0, false);
// set target to current position
wp_nav.init_loiter_target();
#if LAND_REQUIRE_MIN_THROTTLE_TO_DISARM == ENABLED
// disarm when the landing detector says we've landed and throttle is at minimum
if (ap.land_complete && (ap.throttle_zero || failsafe.radio)) {
init_disarm_motors();
}
#else
// disarm when the landing detector says we've landed
if (ap.land_complete) {
init_disarm_motors();
}
#endif
// check if we've completed this stage of RTL
rtl_state_complete = ap.land_complete;
return;
}
// relax loiter target if we might be landed
if (land_complete_maybe()) {
wp_nav.loiter_soften_for_landing();
}
// process pilot's input
if (!failsafe.radio) {
if (g.land_repositioning) {
// apply SIMPLE mode transform to pilot inputs
```

```
update_simple_mode();
// process pilot's roll and pitch input
roll_control = g.rc_1.control_in;
pitch_control = g.rc_2.control_in;
}
// get pilot's desired yaw rate
target_yaw_rate = get_pilot_desired_yaw_rate(g.rc_4.control_in);
}
// process pilot's roll and pitch input
wp_nav.set_pilot_desired_acceleration(roll_control, pitch_control);
// run loiter controller
wp_nav.update_loiter(ekfGndSpdLimit, ekfNavVelGainScaler);
// call z-axis position controller
float cmb_rate = get_land_descent_speed();
pos_control.set_alt_target_from_climb_rate(cmb_rate, G_Dt, true);
pos_control.update_z_controller();
// record desired climb rate for logging
desired_climb_rate = cmb_rate;
// roll & pitch from waypoint controller, yaw rate from pilot
attitude_control.angle_ef_roll_pitch_rate_ef_yaw(wp_nav.get_roll(),
wp_nav.get_pitch(), target_yaw_rate);
// check if we've completed this stage of RTL
rtl_state_complete = ap.land_complete;
}
// get_RTL_alt - return altitude which vehicle should return home at
// altitude is in cm above home
static float get_RTL_alt()
{
// maximum of current altitude and rtl altitude
float rtl_alt = max(current_loc.alt, g.rtl_altitude);
#if AC_FENCE == ENABLED
// ensure not above fence altitude if alt fence is enabled
if ((fence.get_enabled_fences() & AC_FENCE_TYPE_ALT_MAX) != 0) {
rtl_alt = min(rtl_alt, fence.get_safe_alt()*100.0f);
}
#endif
return rtl_alt;
}
```

4.3 Σύστημα πλοήγησης εδάφους (Ground Control Station)

Ένας επίγειος σταθμός ελέγχου (GCS) είναι ένα κέντρο ελέγχου με βάση τη γη που παρέχει τις δυνατότητες για τον ανθρώπινο έλεγχο των μη επανδρωμένων οχημάτων στον αέρα ή στο χώρο. Το GCS θα μπορούσε να χρησιμοποιηθεί για τον έλεγχο μη επανδρωμένων εναέριων οχημάτων ή πυραύλων μέσα ή πάνω από την ατμόσφαιρα.

Ο σταθμός μας αποτελείται :

1. Τηλεχειριστήριο τηλεκατεύθυνσης 9 καναλιών.
2. Δέκτης 1.2 ghz για λήψη εικόνας σε οθόνη 7’ σε συνδυασμό με κεραία yagi 1.2 ghz (13 dbi).
3. Οθόνη 7’ για λήψη εικόνας από το drone.
4. Laptop για παρακολούθηση και χάραξη πορείας, μέσω συστήματος τηλεμετρίας.
5. Πομπός 2.4 ghz για μετάδοση εντολών από σύστημα τηλεμετρίας.
6. Γεννήτρια βενζίνης για παροχή ρεύματος στον επίγειο σταθμό.

Κεφάλαιο 5: RLC Φίλτρα

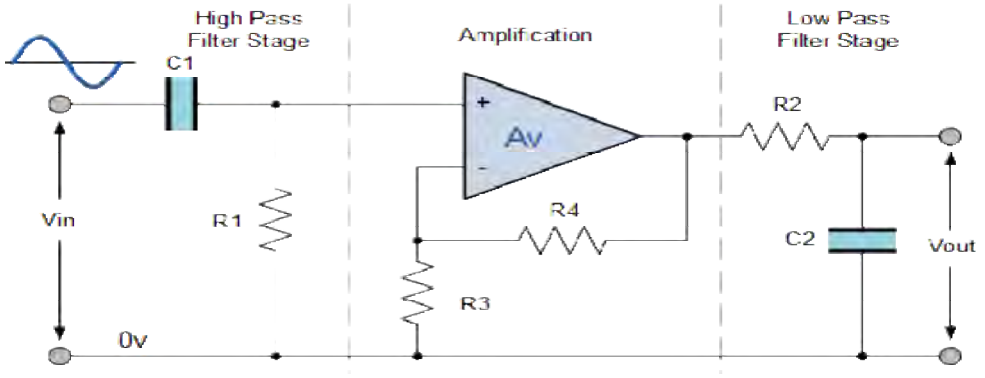
5.1 Τι είναι τα RLC φίλτρα

Ένα φίλτρο διέλευσης χαμηλών συχνοτήτων, είναι ένα φίλτρο που περνά σήματα με συχνότητα μικρότερη από μια ορισμένη συχνότητα αποκοπής και εξασθενεί τα σήματα με συχνότητες υψηλότερες από τη συχνότητα αποκοπής. Η ποσότητα της εξασθένησης για κάθε συχνότητα εξαρτάται από το σχεδιασμό του φίλτρου. Ένα χαμηλής διέλευσης φίλτρο είναι το αντίθετο ενός φίλτρου διέλευσης υψηλών συχνοτήτων. Ένα ζωνοπερατό φίλτρο είναι ένας συνδυασμός ενός low-pass(χαμηλοπερατό) και ένος high-pass φίλτρου(υψηλοπερατό).

Υπάρχουν φίλτρα διέλευσης συχνοτήτων σε πολλές διαφορετικές μορφές, συμπεριλαμβανομένων των ηλεκτρονικών κυκλωμάτων (όπως ένα φίλτρο που χρησιμοποιείται στον ήχο), τα φίλτρα anti-aliasing για σήματα μετατροπής από αναλογικό σε ψηφιακό σήμα, ψηφιακά φίλτρα για την εξομάλυνση συνόλου δεδομένων, ακουστικών εμποδίων , θόλωμα των εικόνων και ούτω καθεξής. Τα φίλτρα διέλευσης συχνοτήτων παρέχουν μια ομαλότερη μορφή ενός σήματος, αφαιρώντας τις μη επιθυμητές διακυμάνσεις.

ΕΙΚΟΝΑ 16

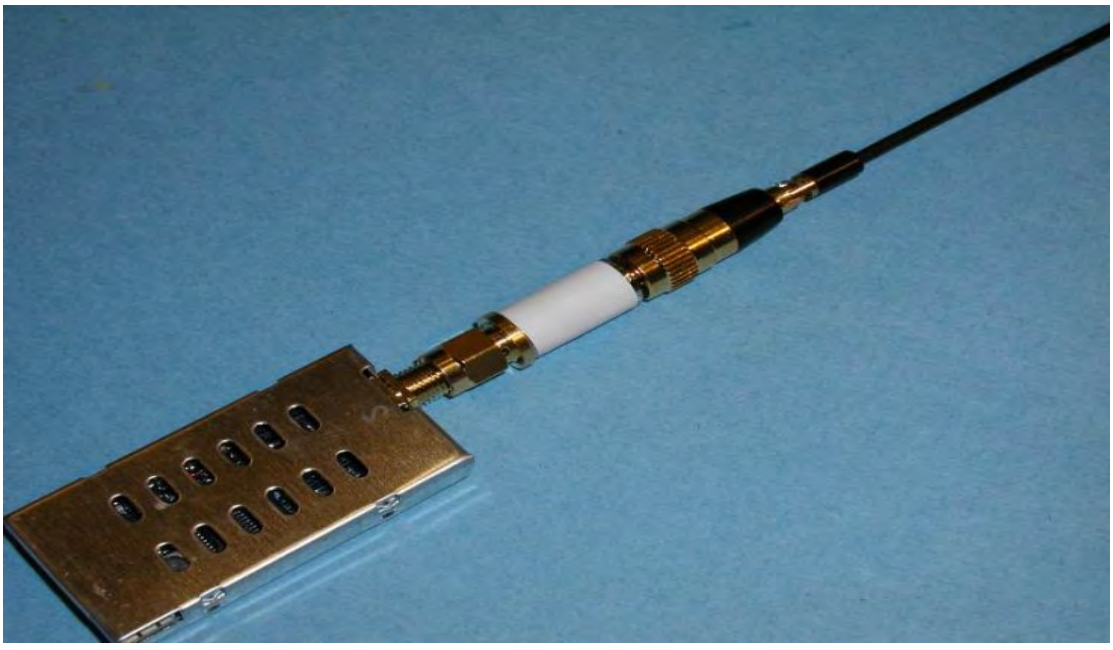
BAND-PASS FILTER



EIKONA 17

EIKONA 18

BAND-PASS FILTER ON 1.2 GHZ VTX



5.2 Ποια η χρησιμότητα και τα οφέλη τους στο συγκεκριμένο project

Λόγω ύπαρξης πολλών αναμεταδόσεων ανάμεσα στον επίγειο σταθμό (GCS) και του εξακοπτέρου, η χρησιμότητα φίλτρων διέλευσης συχνοτήτων είναι αναγκαία, καθώς μετά από δοκιμαστικές πτήσεις που γίνανε χωρίς αυτά, παρατηρήθηκε δραστική μείωση της εμβέλειας τηλεκατεύθυνσης, τηλεμετρίας και βιντεοαναμετάδασης μέχρι και 95 %.

Λόγω μικρού μεγέθους του drone, είναι αναπόφευκτη η τοποθέτηση δεκτών και πομπών σε μικρή απόσταση μεταξύ τους, (=<20cm) με αποτέλεσμα το ένα να παρεμβάλει στο άλλο.

Επιπλέον, παρόλο που χρησιμοποιηθήκανε διαφορετικές συχνότητες για κάθε πομπο-δέκτη, τα επίπεδα θορύβου ήταν πολύ υψηλά από τους πομπούς, καθώς η ισχύς τους ξεπερνούσε το 1 watt.

Στον σταθμό μας, τοποθετήθηκε 433 mhz Low-Pass Filter στον πομπό της τηλεκατεύθυνσης και ένα 2.4 ghz Band Pass Filter στον πομπό της τηλεμετρίας, καθώς παρατηρήθηκε μείωση της λήψης βίντεο από τον 1.2 ghz δέκτη.

Στο εξακόπτερο υπάρχουν, ένας πομπός 1.2 ghz για την μετάδοση της εικόνας και τρεις δέκτες. Ένας δέκτης για την τηλεκατεύθυνση, ένας της τηλεμετρίας και το gps module. Παρατηρήθηκε ότι, η χρησιμοποίηση αυτών χωρίς φίλτρο στον πομπό εικόνας, είχε δραστική μείωση της εμβέλειας και αδυναμία από τον αυτόματο πιλότο να λοκάρει την αρχική θέση απογείωσης λόγω παρεμβολής **(home point)**.

Για τον λόγο αυτό τοποθετήθηκε 1.2 ghz Band-Pass Filter στον πομπό.

Κεφάλαιο 6: Η Πρόοδος του Project se Βίντεο

6.1 Δοκιμή πτητικής ικανότητας drone

https://www.youtube.com/watch?v=QamXWB2e7vs&feature=youtu.be

6.2 Δομική αυτόματης προσγείωσης drone

https://www.youtube.com/watch?v=Lwjb_x0AstM

6.3 Καταγραφή αγροτικής περιοχής από το drone

http://youtu.be/hTcQeqGhqy4

6.4 Καταγραφή αγροτικής περιοχής με ακρίβεια τηλεμετρίας

https://www.youtube.com/watch?v=y0jekbmcvI8&feature=share

6.5 Συντριβή σε κατοικημένη περιοχή

https://www.youtube.com/watch?v=3dJeCNugRWM

ANTENNA TRACKER VIDEO:

https://www.youtube.com/watch?v=MiSJfS3BSQQ&list=UUjvHp-c_Pf3cezOFAKA9inQ

Κεφάλαιο 7: Βιβλιογραφία

1. http://ardupilot.com/
2. http://www.internetnow.gr/
3. http://electronicarc.com/catalogo/
4. http://www.electronics-tutorials.ws/filter/filter_7.html
5. http://www.foxtechfpv.com/
6. http://www.e-wireless.gr/
7. http://www.rangevideo.com/
8. http://hobbywireless.com/
9. http://www.hobbyking.com/hobbyking/store/index.asp
10. http://www.stockrc.com/
11. http://en.wikipedia.org/wiki/Main_Page