ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ, ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Τηλεχειρισμός Ιπτάμενου Drone από προσωπικό υπολογιστή, μέσω WiFi

Πτυχιακή Εργασία

Κασσίδης Αιμιλιανός(4136)

Επιβλέπων:Δρ. Σπυρίδων Α. Καζαρλής, Καθηγητής

ΣΕΡΡΕΣ, ΙΟΥΝΙΟΣ 2023

Περίληψη

Η συγκεκριμένη πτυχιακή εργασία αναφέρεται στην κατασκευή ενός μη επανδρωμένου εναέριου οχήματος (UAV/Drone), τεσσάρων ελίκων (quadcopter), καθώς και στην ενσωμάτωση του καταλλήλως προγραμματισμένου μικροελεγκτή NodeMCU. Με την ενσωμάτωση αυτή επιτυγχάνεται ο ασύρματος τηλεχειρισμός του drone από προσωπικό υπολογιστή, μέσω Wi-Fi. Το quad (για συντομία του quadcopter) είναι ένας τύπος drone, που χαρακτηριστικό του είναι οι τέσσερις του έλικες. Το συγκεκριμένο μοντέλο είναι πιο εύχρηστο, λόγω του μικρού αριθμού ελίκων σε σχέση με άλλα μοντέλα, που μπορεί να έχουν έξι ή και οκτώ (hexacopter και octocopter οι ονομασίες τους αντίστοιχα). Το υλικό για την κατασκευή ενός drone δεν είναι κάτι το εξεζητημένο. Η απλότητα τους όμως, μόνο προνόμια έχει φέρει, καθώς είναι αρκετά πιο εύκολη η επιδιόρθωση κάποιας βλάβης αλλά και η δυνατότητα αναβάθμισής τους. Στην περίπτωση της αναβάθμισης έρχεται να παίξει το ρόλο του ο μικροελεγκτής NodeMCU .Ο NodeMCU είναι μια ανοιχτού λογισμικού πλακέτα, που σε συνδυασμό με το μικροτσίπ ESP8266, εξειδικεύεται σε Internet of Things (IoT) εφαρμογές. Αυτή η προγραμματιζόμενη πλακέτα επιτρέπει την ασύρματη επικοινωνία με το quad και έτσι ο χρήστης μπορεί να κατευθύνει το drone από τον ηλεκτρονικό του υπολογιστή.

Λέξεις Κλειδιά: μη επανδρωμένα οχήματα (UAV), quad, drone, NodeMCU, μικροελεγκτής.

Περιεχόμενα

Πίνακας εικόνων

Εισαγωγή

Τα τελευταία χρόνια είναι φανερή η ραγδαία εξέλιξη της τεχνολογίας σε πολλούς τομείς. Ένας από αυτούς είναι και των μη επανδρωμένων εναέριων οχημάτων ή αλλιώς γνωστά και ως drone. Η χρήση τέτοιων οχημάτων αρχίζει όλο και περισσότερο να αυξάνεται, καθώς αποτελεί προσιτή τεχνολογία σε πολλούς κλάδους. Καινούργιες εφαρμογές αναπτύσσονται καθημερινά έχοντας πρωταρχικό στόχο τη βελτίωση, τόσο της ασφάλειας όσο και της αποτελεσματικότητας, στην υλοποίηση διαφόρων εργασιών. Η τεχνολογία των εναέριων οχημάτων είναι αδιαμφισβήτητα μια από τις καλύτερες λύσεις σε ενέργειες, όπου η πρόσβαση του ανθρώπινου παράγοντα καθίσταται επικίνδυνη ή και πολλές φορές αδύνατη. Επιπλέον, ένα ακόμα χαρακτηριστικό είναι η ευελιξία παραμετροποίησης τους. Η προσθήκη αισθητήρων, πομποδεκτών και άλλων συστημάτων, μετατρέπει ένα drone από συλλέκτη δεδομένων στον αγροτικό τομέα μέχρι και πολεμική μηχανή στο στρατό. Τα στοιχεία που αποτελούν ένα τέτοιο drone είναι αρκετά απλά άρα μπορούν και παραμετροποιούνται πιο εύκολα. Ένα quadocopter έχει μια μονάδα ελέγχου, τέσσερις έλικες με τους κινητήρες τους, καθώς και τέσσερις ελεγκτές (ESC) για τη παροχή ισχύος από τη μονάδα ελέγχου στα τέσσερα μοτέρ.

Η παρούσα εργασία στηρίζεται σε μία προσπάθεια να εισάγει τον αναγνώστη στη δομή ενός quad και στον τρόπο με τον οποίο μπορεί ένας μικροελεγκτής να καθοδηγήσει ένα τέτοιο όχημα. Μετά την ανάγνωση της εργασίας ο αναγνώστης θα μπορεί, έχοντας την αναγκαία πλακέτα και κάνοντας κάποιες μικρές καλωδιακές συνδέσεις, να τηλεκατευθύνει το quad του, από τον ηλεκτρονικό του υπολογιστή, μέσω WiFi σύνδεσης. Με τον απομακρυσμένο έλεγχο ενός quad ο χειριστής μπορεί να μην είναι παρών στο χώρο και παρ' όλα αυτά να έχει τον πλήρη έλεγχο του quad. Με αυτόν τον τρόπο καταφέρνει να έχει πρόσβαση σε μέρη που η φυσική του παρουσία είναι δύσκολη ή επικίνδυνη. Η απλότητα της εφαρμογής κάνει την εγκατάσταση της πολύ εύκολη.

Στο πρώτο Κεφάλαιο θα αναφερθούν τα μη επανδρωμένα εναέρια οχήματα. Ακόμη, γίνεται μια εισαγωγή στα drone, αναλύονται κάποιες κατηγορίες τους αλλά και εφαρμογές που μπορούν να έχουν στη καθημερινότητα. Στο Κεφάλαιο 2 αναφέρονται διάφοροι τρόποι τηλεκατεύθυνσης, κάποιοι αρκετά διαδεδομένοι και κάποιοι που βρίσκονται σε πειραματικό στάδιο. Στη συνέχεια, θα γίνει μια αναφορά στο υλικό αλλά και το λογισμικό που χρησιμοποιήθηκε για την επίτευξη της εργασίας. Στο τέταρτο κεφάλαιο, αναλύεται ο τρόπος με τον οποίο τα δύο κομμάτια, υλικό και λογισμικό, συγκροτήθηκαν και αλληλεπίδρασαν για να στηθεί η εργασία. Στο τελευταίο, κεφάλαιο γίνεται λόγος για τα βασικά προβλήματα που προέκυψαν, ο τρόπος επίλυσης τους καθώς και πρακτικές βελτίωσης των μεθόδων που χρησιμοποιήθηκαν για μελλοντική συνέχεια της εργασίας.

Κεφάλαιο 1: Μη επανδρωμένα εναέρια οχήματα

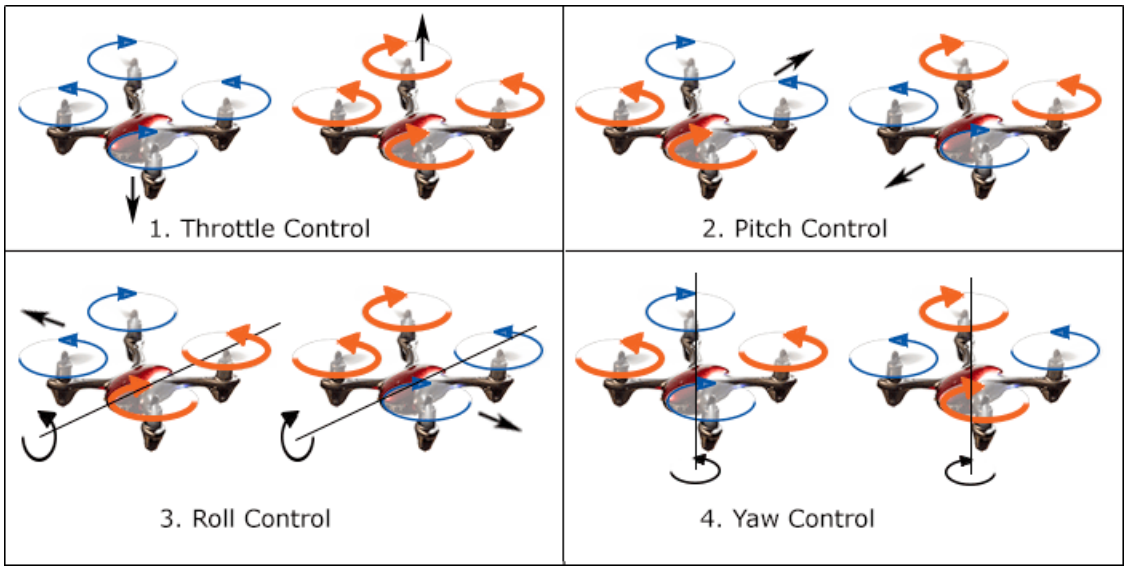
Ως μη επανδρωμένα εναέρια οχήματα UAV (Unmanned Aerial Vehicles) ή UAS (Unmanned Aerial Systems), ονομάζουμε όλα τα ιπτάμενα οχήματα που δεν έχουν χειριστή στην άτρακτό τους, αλλά πραγματοποιούν πτήσεις, είτε αυτόνομα είτε μέσω τηλεκατεύθυνσης. Οι προαναφερθείσες ονομασίες αναφέρονται στους ορισμούς που κατά καιρούς έχουν δοθεί για την περιγραφή αυτών των οχημάτων. Ο όρος UAV περιγράφει το αεροσκάφος χωρίς χειριστή . Ο όρος UAS περιλαμβάνει όλες τις συσκευές, το προσωπικό και τις διαδικασίες, οι οποίες χρησιμοποιούνται, προκειμένου το μη επανδρωμένο αεροσκάφος να θεωρείται ως ολοκληρωμένο σύστημα. Τέλος, ο όρος RPAS (Remotely Piloted Aircraft Systems) καθιερώθηκε σύμφωνα με την ισχύουσα νομοθεσία και με την ανάγκη, όλες οι πτήσεις μη επανδρωμένων αεροσκαφών, να έχουν τουλάχιστον έναν επιβλέποντα πιλότο στο έδαφος. Τα μη επανδρωμένα ιπτάμενα οχήματα συνήθως έχουν τη μορφή μικρού αεροπλάνου ή ελικοπτέρου, με έναν ή περισσότερους κινητήρες και έλικες συντονισμένους για πλήρως ελεγχόμενη πτήση από ειδικό πρόγραμμα ή χειριστήριο εδάφους.[1]

1.1 Κατηγορίες μη επανδρωμένων εναέριων οχημάτων

Η κατηγοριοποίηση των UAV έχει γίνει βάση κάποιων χαρακτηριστικών, όπως ο τρόπος παραγωγής της άντωσης, το μέγεθος και ο αριθμός των κινητήρων που φέρουν. Το υποκεφάλαιο αυτό θα αναφερθεί σε αυτές ακριβώς τις κατηγορίες.

UAV Κινητής πτέρυγας (ελικόπτερα, πολύ-κόπτερα)

Τα μη επανδρωμένα οχήματα με ρότορες ή αλλιώς drones είναι ο πιο δημοφιλής τύπος UAV . Χρησιμοποιώντας τους περιστρεφόμενους έλικες τους καταφέρνουν να ανυψωθούν και να κινηθούν στο χώρο. Η πτήση τους είναι παρόμοια με αυτή των ελικοπτέρων. Ο πιλότος μπορεί να ελέγξει ξεχωριστά τις τέσσερις κινήσεις του drone. Την ρύθμιση του υψομέτρου ή γκάζι (throttle), την κίνηση μπρός-πίσω (pitch), την κύλιση δεξιά-αριστερά γύρω από τον οριζόντιο άξονα του quad (roll) και την περιστροφή δεξιά-αριστερά γύρω από τον κατακόρυφο άξονα του drone (yaw). Έχοντας τον πλήρη έλεγχο για το που βρίσκεται, ο χρήστης μπορεί ανά πάσα στιγμή να θέσει το drone σε οποιαδήποτε θέση στο χώρο με ακρίβεια.



Εικόνα 1: Έλεγχος τετρακοπτέρου. Μπλε-κανονική ταχύτητα, πορτοκαλί-αυξημένη ταχύτητα.

Τα μη επανδρωμένα οχήματα αυτού του τύπου μπορούν να κατηγοριοποιηθούν με διάφορους τρόπους, ωστόσο θα δούμε δύο από αυτούς. Κατηγοριοποίηση με βάση το μέγεθος τους και με βάση τον αριθμό των κινητήρων που διαθέτουν.

- **UAV με βάση το μέγεθος**

Ο βασικότερος τρόπος κατηγοριοποίησης των μη επανδρωμένων εναέριων οχημάτων είναι με βάση το μέγεθος. Το μέγεθος παίζει σημαντικό ρόλο, καθώς είναι από τα κύρια χαρακτηριστικά που τα καθιστούν κατάλληλα για να ανταπεξέλθουν στις διάφορες συνθήκες και εργασίες [2].

Nano drone: Θεωρούνται τα μικρότερα UAV στην αγορά. Το μέγεθος τους κυμαίνεται από 5-100 χιλιοστά και είναι η καλύτερη επιλογή για να χρησιμοποιηθούν για εκπαιδευτικό σκοπό και ψυχαγωγία.

Mini drone: Από 100-350 χιλιοστά. Ένα σαφώς μεγαλύτερο όχημα το οποίο χειρίζεται επιπλέον εξοπλισμό όπως μικρές κάμερες.

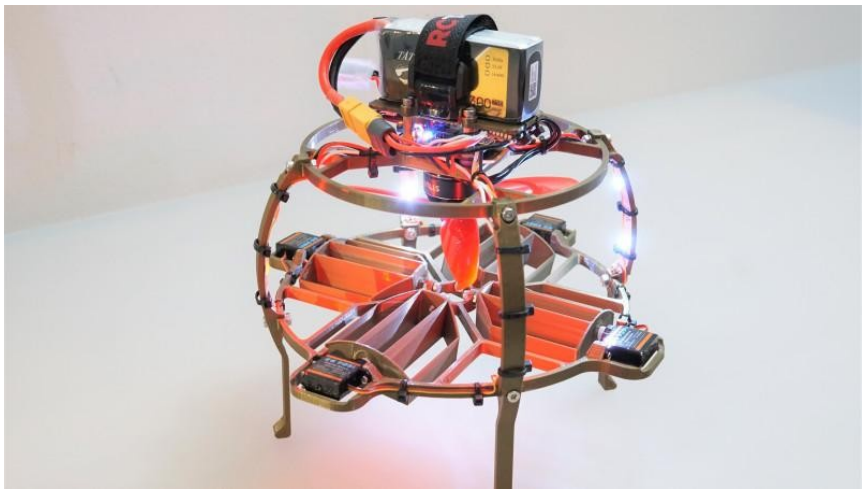
Drone μεσαίου μεγέθους: Είναι έως και δύο φορές μεγαλύτερα από τα mini drones, το μέγεθος τους κυμαίνεται από 350-600 χιλιοστά. Πλέον, το UAV έχει την ικανότητα υποστήριξης μεγαλύτερης μπαταρίας, διαθέτει μεγαλύτερη εμβέλεια από τον πομπό καθώς και μεγαλύτερες κάμερες.

Μεγάλου μεγέθους Drone: Μεγαλύτερα από 600 χιλιοστά και εξαιρετικά δυνατά εναέρια οχήματα που χρησιμοποιούνται κυρίως σε εργασίες ανύψωσης μεγάλων φορτίων.

- **UAV με βάση τον αριθμό των κινητήρων**

Ο αριθμός των κινητήρων είναι ένα ακόμα πολύ σημαντικό χαρακτηριστικό στα UAV. Τόσο η ισχύς όσο και ο χρόνος χρήσης έχουν άμεση συσχέτιση με το πλήθος των ελίκων πάνω στο UAV, κριτήρια αρκετά σημαντικά για την αγορά ενός drone [2].

Μονοκόπτερα (Monocopters): Στην κατηγορία των μονοκόπτερων drone μπορεί να παρατηρηθεί μια τεράστια γκάμα διαφορετικών μεγεθών, σχημάτων και διαστάσεων. Υπάρχουν μονοκόπτερα που περιστρέφονται γύρω από τον εαυτό τους παρομοιάζοντας την μορφή ενός φύλλου που πέφτει από το δέντρο, μέχρι και drone σε σχήμα μπάλας. Αν και έχουν μόνο έναν ρότορα, τα μονοκόπτερα έχουν απασχολήσει μεγάλο μέρος της κοινότητας.

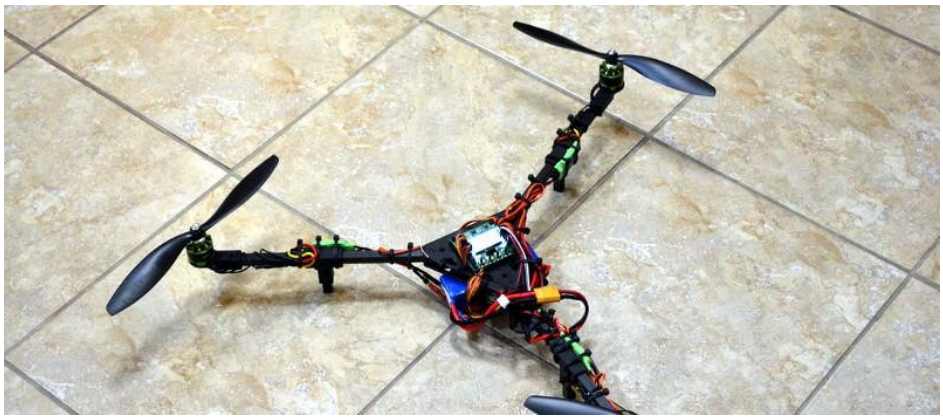


Εικόνα 2: Single rotor Ball-Drone. Μονοκόπτερο drone στιλ μπάλας

Δικόπτερα (Dual-Rotor drone): Όπως και με τα μονοκόπτερα έτσι και στα drone με δύο έλικες μπορούμε να βρούμε πολλές διαφορετικές κατασκευές. Με μόνο δύο έλικες και με μία προηγμένη τεχνολογία σταθεροποίησης που του επιτρέπει να πετάει για διπλάσιο χρόνο από τα συνηθισμένα μοντέλα, η Flybotix έχουν κατασκευάσει το παρακάτω δικόπτερο

Εικόνα 3: Dual-propeller drone by Flybotix. Το πρωτοποριακό δικόπτερο της Flybotix

Τρικόπτερα (Tricopters): Στους τρεις έλικες πλέον μπορεί να παρατηρηθεί μια πιο οικεία εικόνα πλησιάζοντας σε κάτι πιο συνηθισμένο για τα μάτια ενός απλού παρατηρητή.



Εικόνα 4: Tri Rotor Drone. Τρικόπτερο drone

Τετρακόπτερα (Quadcopters): Ο πιο συνηθισμένος τύπος drone είναι αυτός με τους τέσσερις ρότορες. Αυτή τους η αρχιτεκτονική τα καθιστά αρκετά σταθερά , ασφαλή και αποτελούν μια πολύ καλή επιλογή και για χόμπι αλλά και για επαγγελματικό σκοπό.

Εικόνα 5: Quadcopter DJI Phantom.
Τετρακόπτερο DJI Phantom

Εξακόπτερα (Hexacopters): Με τους έξι ρότορες είναι αρκετά πιο σταθερά από τα τετρακόπτερα. Λόγω του μεγέθους τους διαθέτουν και πιο πολλούς πόρους , έτσι έχουν μεγαλύτερη ταχύτητα , διάρκεια , μέχρι και εμβέλεια. Επιπλέον ο μεγάλος αριθμός κινητήρων τα καθιστά πιο ανεκτικά σε σφάλματα καθώς στην περίπτωση που χάσουν έναν ρότορα οι υπόλοιποι πέντε μπορούν να ισορροπήσουν το drone.

Εικόνα 6: Hexacopter Hexo+ Drone. Εξακόπτερο Hexo+

Οκτακόπτερα (Octocopter): Αποτελώντας μια από τις μεγαλύτερες κατηγορίες πολυκοπτέρων τα οκτακόπτερα, με οκτώ ρότορες, είναι από τα πιο δυνατά drone . Οι ισχυροί του κινητήρες μπορούν να συγκρατήσουν πολύ μεγάλα φορτία με ευκολία .

Εικόνα 7: DJI S1000. Οκτακόπτερο DJI S1000

UAV σταθερής πτέρυγας

Λειτουργούν παρόμοια με τα κοινά αεροπλάνα και πολλές φορές έχουν και την ίδια εμφάνιση. Χρησιμοποιώντας τα μακριά τους πτερύγια, επωφελούνται από την διαφορά πίεσης του αέρα για να σηκωθούν από το έδαφος και να παραμένουν στον αέρα για μεγάλο χρονικό διάστημα. Αυτή η τεχνική, τους επιτρέπει να εξοικονομούν μεγάλες ποσότητες ενέργειας και τα καθιστά δημοφιλή σε εργασίες που απαιτούν κάλυψη μεγάλων αποστάσεων [3].

Εικόνα 8: UAV από το ΑΠΘ

UAV κάθετης απογείωσης και προσγείωσης (VTOL)

Ένας συνδυασμός των δύο προηγούμενων κατηγοριών, τα VTOL είναι υβριδικά UAV τα οποία περιλαμβάνουν τόσο σταθερά πτερύγια, όσο και έλικες που χρησιμοποιούνται κατά τη διάρκεια της πτήσης. Οι έλικες τους επιτρέπουν την κάθετη απογείωση, ενώ με τα φτερά τους μπορούν να καλύψουν μεγάλες αποστάσεις με μικρή δαπάνη ενέργειας [2].

Εικόνα 9: SV1 Vanguard-Professional LR VTOL drone. Drone κάθετης απογείωσης και προσγείωσης από την sunbirds

1.2 Εφαρμογές μη επανδρωμένων εναέριων οχημάτων

Το γεγονός ότι μπορούν να κινηθούν τρισδιάστατα στον χώρο και ο πιλότος μπορεί να έχει τον έλεγχο από απόσταση, άλλα και πολυάριθμα άλλα χαρακτηριστικά, θέτουν τα μη επανδρωμένα εναέρια οχήματα ως εξαιρετική επιλογή για πολλές εφαρμογές σε πολλούς κλάδους .

- **Στρατιωτικές εφαρμογές**

Η πρώτη καταγεγραμμένη στρατιωτική εφαρμογή των UAV είναι πριν από περίπου 100 χρόνια όταν η Βενετία βομβαρδίστηκε από την Αυστρία το 1849 [4]

- **Κυβερνητικές εφαρμογές**

Αποστολές που μια κυβέρνηση κρίνει πως είναι επικίνδυνες για την σωματική ακεραιότητα ενός πληρώματος, είναι δυνατό να τις αναθέσει σε drone. Τέτοιες αποστολές μπορεί να είναι , πολύωρες περιπολίες συνόρων, επιχειρήσεις διάσωσης σε περιοχές με επικίνδυνα υλικά , και πολλές άλλες.

- **Χόμπι**

Όταν ένα τέτοιο όχημα είναι διαθέσιμο προς όλους δεν μπορεί να αποκοπεί και το κομμάτι της διασκέδασης. Τα τελευταία χρόνια η κοινότητα των αερομοντελιστών έχει μεγάλη αύξηση, με αποτέλεσμα την διάδοση αυτών των οχημάτων. Αυτή η αύξηση ενδιαφέροντος έχει φέρει πολλά θετικά στον χώρο των drone enthusiasts.

- **Επαγγελματικές εφαρμογές**

Όσο περνάνε τα χρόνια πάρα πολλές εταιρείες ανακαλύπτουν τα οφέλη των drone και τα ενσωματώνουν στους εργασιακούς τους χώρους. Η πιο κοινή ενσωμάτωσή τους στον επαγγελματικό χώρο είναι στην λήψη εικόνων ή βίντεο. Η αεροφωτογράφιση και η αεροβιντεοσκόπηση ίσως είναι η πιο βασική λειτουργία τους. Εφαρμογές αυτών των λειτουργιών βλέπουμε στα διαφημιστικά σποτ, στις ταινίες , στον αθλητισμό, στην τρισδιάστατη χαρτογράφηση και αλλού.

- **Παράδοση προϊόντων**

Πολλές γνωστές εταιρίες έχουν εντάξει τα UAV στην μεταφορά προϊόντων τους με χαρακτηριστικό παράδειγμα την Amazon. Με αυτόν τον τρόπο, ελπιδοφόρα μας δείχνουν την εξέλιξη και την συνεισφορά που μπορούν να έχουν τέτοιου είδους οχήματα. Στο μέλλον μπορούμε να δούμε σίτιση ανθρώπων σε απομακρυσμένες περιοχές ή και αποστολή φαρμάκων.

- **Αποστολές έρευνας και διάσωσης**

Φοιτητές από το Ιράν κατασκεύασαν drone που μεταφέρει σωσίβια για να σώσει άτομα από πιθανό πνιγμό[5]. Ένα ακόμα project διάσωσης έχει αναπτύξει το Delf University στην Ολλανδία, όπου το drone μεταφέρει ειδικό εξοπλισμό για πρώτες βοήθειες [6].

Κεφάλαιο 2: Απομακρυσμένος Έλεγχος

Μιας και τα οχήματα είναι μη επανδρωμένα, ο έλεγχος από απόσταση είναι ο κύριος τρόπος κατεύθυνσης τους. Λόγω του πλήθους των εφαρμογών που μπορούν να έχουν τέτοια οχήματα έχουν αναπτυχθεί ραγδαία και οι τεχνικές τηλεκατεύθυνσης τους. Μεγάλες βιομηχανίες, όπως η MOTOROLA solutions, μέχρι και απλοί χομπίστες έχουν αναπτύξει τρόπους κατεύθυνσης και χειρισμού τέτοιων οχημάτων. Κάποιες από αυτές τις τεχνικές είναι οι επακόλουθες.

2.1 Τεχνικές απομακρυσμένου ελέγχου

- **Live Streaming από ηλεκτρονική συσκευή**

Ένας από τους πιο διαδεδομένους τρόπους ελέγχου ενός drone είναι μέσω κάποιας ηλεκτρονικής συσκευής, όπως smartphone ή tablet. Αρκετές εταιρείες, όπως η DJI, έχουν δημιουργήσει εφαρμογές που επιτρέπουν τους χρήστες να ελέγχουν το drone τους από το κινητό τους τηλέφωνο. Μέσω WiFi ή Bluetooth, το smartphone αποτελεί πομπός του σήματος και το drone ο δέκτης, έτσι ο χρήστης μπορεί και το τηλεκατευθύνει πολύ εύκολα από την ηλεκτρονική του συσκευή.

Εικόνα 10: Οθόνη smartphone, ενώ γίνεται η τηλεκατεύθυνσης του drone

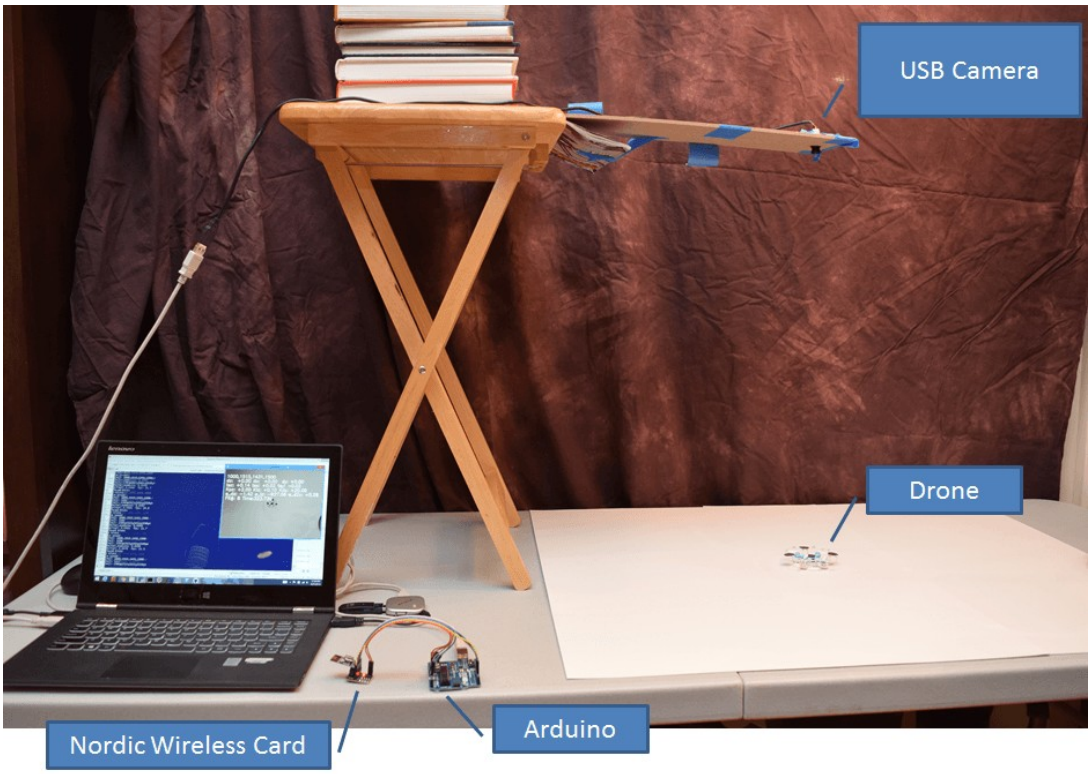
- **Transmitter & Receiver**

Ίσως ο πιο συνηθισμένος τρόπος χειρισμού μη επανδρωμένου οχήματος είναι με την χρήση τηλεχειριστηρίου εκπομπής. Οι συσκευές αυτές εκπέμπουν συχνότητες από 27MHz μέχρι και 5.8 GHz με συγκεκριμένα πρωτόκολλα κωδικοποίησης σήματος[7]. Τα πρωτόκολλα αυτά χωρίζονται σε δύο κατηγορίες. TX Protocols, τα οποία σχετίζονται με την επικοινωνία ανάμεσα στον πομπό και δέκτη. RX Protocols που σχετίζονται με την επικοινωνία ανάμεσα στον δέκτη και τον Flight Controller[8]. Οι λαβές του τηλεχειριστηρίου προσφέρουν ασύγκριτη σταθερότητα και ακρίβεια στον χειρισμό. Επιπλέον, υπάρχουν συσκευές που υποστηρίζουν σύνδεση του smartphone σαν οθόνη, στην περίπτωση που το drone διαθέτει κάμερα.

Εικόνα 11: DJI phantom remote controller. Τηλεχειριστήριο της DJI

- **Από τον υπολογιστή μέσω κάμερας**

Σε μια μεγάλη κοινότητα που υποστηρίζουν και εξελίσσουν τα drones, υπάρχουν πολλοί drone enthusiasts που έχουν βρει νέους τρόπους χειρισμού αυτών των συσκευών. Έχοντας λοιπόν, έναν arduino με μια wireless κάρτα, μια κάμερα και ένα drone, έχουν καταφέρει να φτιάξουν ένα πρόγραμμα τηλεχειρισμού από τον υπολογιστή μέσω δικτύου. [9]



Εικόνα 12: Fly a mini-drone with your computer! Πρότζεκτ τηλεκατεύθυνσης drone από τον ηλεκτρονικό υπολογιστή

Κεφάλαιο 3: Υλικό και λογισμικό που χρησιμοποιήθηκε

Για την επίτευξη της παρούσας πτυχιακής εργασίας ήταν αναγκαία η χρήση συγκεκριμένου υλικού αλλά και λογισμικού έτσι ώστε να επιτευχθούν τα επιθυμητά αποτελέσματα. Στο κεφάλαιο αυτό θα γίνει παρουσίαση των πόρων που χρησιμοποιήθηκαν, αλλά και ανάλυση των λόγων που επιλέχθηκαν οι συγκεκριμένοι για να αποτελέσουν την εργασία.

Πιο συγκεκριμένα, θα αναφερθούν αναλυτικά τα εξαρτήματα που αποτελούν το quad. Θα εξεταστεί το λογισμικό για τον προγραμματισμό του drone. Θα γίνει ανάλυση σε βάθος του μικροελεγκτή που χρησιμοποιήθηκε για την επικοινωνία μεταξύ υπολογιστή και drone. Και τέλος υπάρχει το λογισμικό προγραμματισμού του ελεγκτή και το λογισμικό για την δημιουργία του προγράμματος επικοινωνίας μέσω προσωπικού υπολογιστή με το drone.

3.1 Drone (υλικό)

Το drone που χρησιμοποιήθηκε στη συγκεκριμένη πτυχιακή έχει τα εξής χαρακτηριστικά.

Πλαίσιο (Frame): F450 Drone frame.
Ένα πλαίσιο 450 χιλιοστών (mm) για τη κατασκευή ενός drone. Διαθέτει πλακέτα κατανομής ρεύματος (PDB) και 4 πόδια για στήριξη.

Εικόνα 13: F450 Drone Frame. Πλαίσιο F450 drone

Κινητήρες (Motors): Mini Brushless Motor MT2204 2300KV.

Οι κινητήρες στα drones είναι δύο ειδών. Υπάρχουν οι brushed κινητήρες, για τα micro ή nano drones και οι brushless, που είναι πιο μεγάλοι σε μέγεθος και ισχύ για τα μεγαλύτερα UAV [10]. Ένα ακόμα χαρακτηριστικό τους, είναι ο αναγραφόμενος αριθμός KV. Αυτός ο αριθμός προσδιορίζει τις στροφές ανά λεπτό / volt. Στην περίπτωση της εικόνας υπάρχει ένας κινητήρας MT2204 με αριθμό KV2300. Αυτό σημαίνει ότι αν θέσουμε τάση του ενός Volt στο μοτέρ, αυτό θα περιστρέφεται με 2300 στροφές το λεπτό.

Διάμετρος: 28mm.

Βάρος: 30g.

Συνολικό ύψος: 36mm.

Μέγιστη ισχύς: 150W.

Εικόνα 14: Κινητήρας MT2204 με αριθμό KV2300

Ελεγκτές Ταχύτητας (ESC): XXD HW30A Brushless Motor ESC.

Οι ελεγκτές ταχύτητας (ESC) όπως υποδηλώνει και το όνομα τους είναι αυτοί που ελέγχουν την ταχύτητα περιστροφής του κάθε κινητήρα έτσι ώστε το UAV να κινείται σύμφωνα με τις οδηγίες που λαμβάνει από τον flight controller [11].

Βάρος:25g.

Διαστάσεις: 57mm X 25mm X 8mm.

Συνεχές ρεύμα: 30A.

Ισχύς εισόδου: 6V - 12V.

Έξοδος BEC (Battery Eliminator Circuit): 5V/3A. (μπορεί να χρησιμοποιηθεί ως πηγή για άλλα περιφερειακά)

Εικόνα 15: XXD HW30A ESC. Ηλεκτρικός ελεγκτής ταχύτητας.

Ελεγκτής Πτήσης (Flight Controller): Sp Racing F3 Flight Controller board deluxe.

Ο ελεγκτή πτήσης (FC) μπορεί να τον παρομοιαστεί με την μητρική κάρτα ενός ηλεκτρονικού υπολογιστή. Κουμπώνοντας όλα τα εξαρτήματα (ESC, receiver, κάμερα) πάνω σε αυτόν και με ένα σύνολο από κυκλώματα και αισθητήρες, που διαθέτει ο ίδιος, είναι αυτός που "αισθάνεται" τον χώρο, ελέγχει το drone και επικοινωνεί με τον χρήστη [12].

Διαστάσεις: 36mm X 36mm.

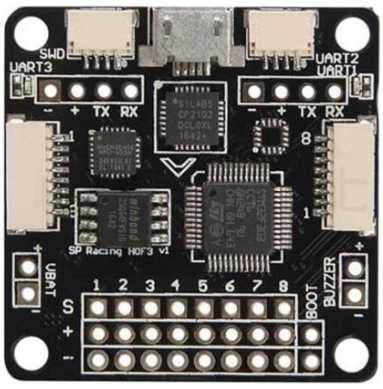
Βάρος: 5g.

Επεξεργαστής: STM32F3.

Αισθητήρες: MPU6050 accelerometer/gyro,MS5611 barometer,QMC5883 compass.

Sockets:

- 4x 4pin JST-SH sockets (I2C, SWD, 2xUART)
- 2x 8pin JST-SH sockets (PPM, PWM, SERIAL RX, GPIO, ADC, 3V, 5V, GND)
- 8x 3pin though-holes for pin headers for ESC/Servo connections.
- 2x 4pin though-holes for pin headers for 2x serial ports.
- 2x 2pin though-holes for pin headers for battery voltage and buzzer.



Εικόνα 16: Sp Racing F3 Flight Controller board deluxe.
Ελεγκτής πτήσης SP Racing F3.

Μπαταρία (Battery): LYNYOUNG 11.1V 1500mAh 30C Lipo Battery.

Ο πιο δημοφιλής τύπος μπαταρίας στην κοινότητα των UAV είναι οι μπαταρίες λιθίου πολυμερών ή Lithium Polymer (LiPo) batteries. Οι LiPo μπαταρίες είναι αρκετά ελαφριές, έχουν μεγάλη χωρητικότητα και διατηρούν σταθερό ρεύμα, χαρακτηριστικά ικανά να τις θέσουν ως νούμερο ένα επιλογή [13]. Ο δείκτης C που διακρίνεται παρακάτω υποδηλώνει την ικανότητα της μπαταρίας να παρέχει ρεύμα σε σχέση με την χωρητικότητά της. Στην περίπτωση που η μπαταρία έχει χωρητικότητα = 1500 mAh και C = 30 σημαίνει ότι μπορεί να παρέχει 1,5A * 30 = 45A ρεύμα.

Χωρητικότητα: 1500mAh.

Ποσοστό συνεχούς εκφόρτισης: 30C

Τάση: 11.1V

Βύσμα: XT60

Εικόνα 17: LYNYOUNG 11.1V 1500mAh 30C LiPo Battery. Μπαταρία LiPo 1500mAh

Τηλεκατεύθυνση (Transmitter): Newest Flysky FS-i6 FS I6 2.4G 6ch RC Transmitter Controller w/ FS-IA6B Receiver.

Αποτελείται από δύο μέρη, τον πομπό που τον έχει στην κατοχή του ο χρήστης και τον δέκτη που βρίσκεται πάνω στο drone. Με εύρος συχνότητας κοντά στα 2.4GHz και έξι διαθέσιμα κανάλια είναι μία πολύ καλή επιλογή για τον τηλεχειρισμό ενός quad.

Εύρος ραδιοσυχνότητας: 2.405 - 2.475GHz

Εύρος ζώνης (Bandwidth): 500KHz

Κανάλια: 6.

Εικόνα 18: Flysky FS-i6 Transmitter Controller.
Τηλεχειριστήριο Flysky FS-i6..

3.2 Drone (λογισμικό)

Κατασκευάζοντας ένα quad από την αρχή, πέρα από το υλικό πρέπει να δοθεί ιδιαίτερη σημασία και σε κάποια βασικά χαρακτηριστικά του λογισμικού του. Το βασικότερο στοιχείο σε ένα drone είναι ο Flight Controller. Πάνω του κουμπώνουν όλα τα υπόλοιπα στοιχεία και είναι αυτός που έχει την ευθύνη να τα συντονίσει έτσι ώστε όλο το σύστημα να δουλεύει εναρμονισμένα. Για τον σκοπό αυτό, χρησιμοποιήθηκε το Cleanflight. Το Cleanflight είναι ένα ανοιχτού λογισμικού πρόγραμμα, ανάμεσα σε πολλά άλλα, το οποίο επιτρέπει τον προγραμματισμό του Flight Controller πάνω στο drone. Επιπλέον προσφέρει στον χρήστη την ικανότητα να παρακολουθήσει διάφορα χαρακτηριστικά του UAV(ισχύ κινητήρων, βαρομετρικά κ.α.) ή και να προσθέσει διάφορες λειτουργίες σε αυτό(Arm, Angle, Beeper κ.α.). Θα εξεταστεί πιο αναλυτικά στο επόμενο κεφάλαιο. Μερικά ακόμα τέτοια προγράμματα είναι το Betaflight, το Butterflight, το Raceflight και το KISS [14]

- Το KISS είναι από τα πιο απλά software διαθέσιμα με τακτά ενημερωμένο λογισμικό και χωρίς πολλές επιλογές να μπερδεύουν τον χρήστη. Η έλλειψη επιπρόσθετων ρυθμίσεων βέβαια είναι και ένα από τα μεγάλα μειονεκτήματα του, καθώς ένας έμπειρος πιλότος δεν μπορεί να έχει πρόσβαση σε πιο εξειδικευμένες ρυθμίσεις.
- Το Raceflight (FlightOne) έχει παρόμοια λειτουργικά χαρακτηριστικά με το Betaflight και το Cleanflight. Το λογισμικό αυτό επικεντρώνεται πιο πολύ στα αγωνιστικά και ακροβατικά χαρακτηριστικά ενός drone.
- Το Butterflight είναι μια εκδοχή του Betaflight που στοχεύει στα mini quads.
- Τέλος το Betaflight αν όχι η καλύτερη, είναι μια από τις καλύτερες επιλογές για εφαρμογές αυτού του σκοπού. Η κοινότητα γύρω του είναι αρκετά μεγάλη με αποτέλεσμα να υπάρχουν διαθέσιμες λύσεις για κάθε πρόβλημα που πιθανόν θα προκύψει. Υποστηρίζει ένα μεγάλο πλήθος από Flight Controllers και στοχεύει να εξυπηρετήσει τις ανάγκες του κάθε χειριστή ανεξαρτήτως στο τι στοχεύει. Με τις συνεχόμενες αναβαθμίσεις το Betaflight είναι ένα πολύ δυνατό εργαλείο για κάθε drone pilot.

3.3 NodeMCU (υλικό)

Ο NodeMCU είναι μια προγραμματιζόμενη πλακέτα που στηρίζεται στον ESP8266-12. Ο ESP8266-12 είναι ένα χαμηλού κόστους microchip με ενσωματωμένο TCP/IP stack (ένα σετ από πρωτόκολλα επικοινωνίας) και ικανότητες μικροελεγκτή. Ο NodeMCU είναι ένα εξαιρετικό εργαλείο για αρκετούς λόγους

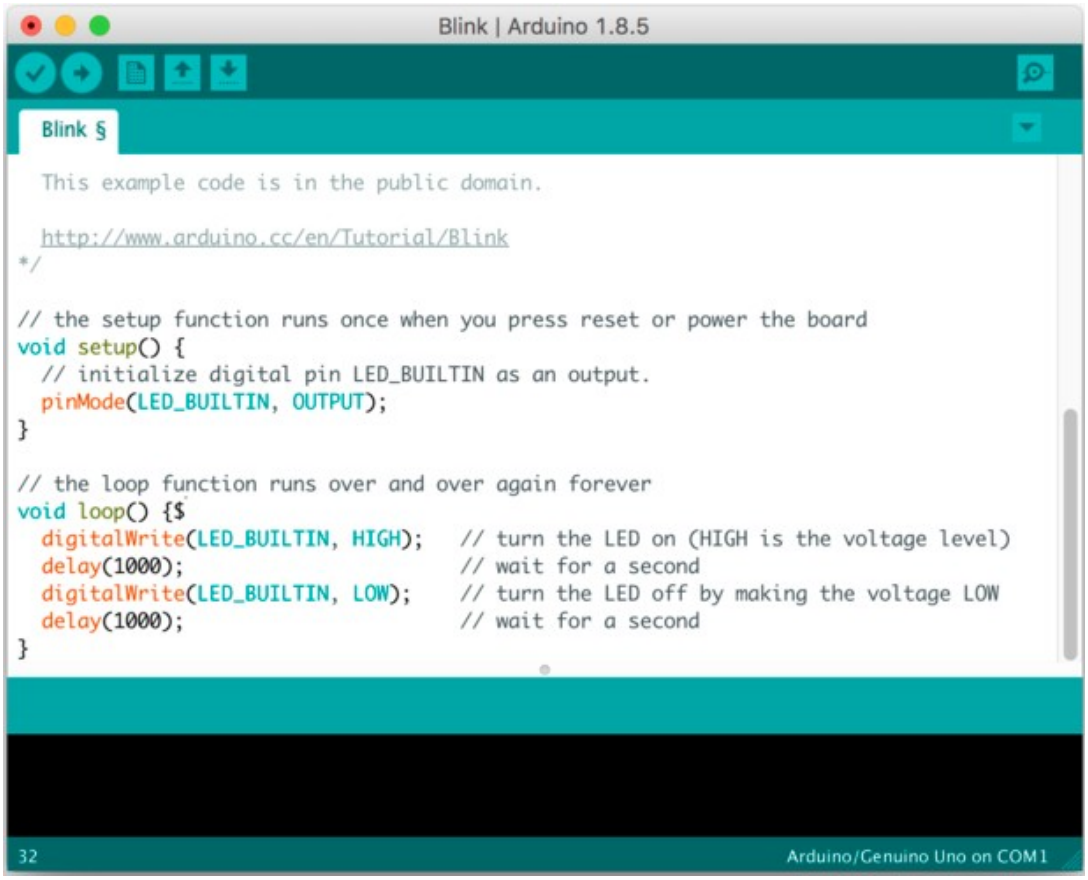
Αρχικά είναι μια μικρή και συμβατή πλακέτα που μπορεί να εξυπηρετήσει μικρά αλλά και μεγάλα projects. Το κόστος είναι ένας σημαντικός παράγοντας, καθώς η συγκεκριμένη πλακέτα μπορεί να βρεθεί σε πολύ χαμηλή τιμή σε σχέση με τις δυνατότητες της. Είναι μια αρκετά δυνατή πλακέτα, με 13 GPIO pins από τα οποία τα 12 είναι ψηφιακά και το ένα αναλογικό. Επιπλέον από το d1 μέχρι το d8 pin μπορεί να εκπέμψει PWM σήμα που είναι και αναγκαίο για το συγκεκριμένο project. Ένα ακόμα χαρακτηριστικό είναι η χαμηλή κατανάλωση ενέργειας, πολύ σημαντικό πλεονέκτημα κρίνοντας ότι θα αντλεί ενέργεια από το drone που από μόνο του έχει αρκετά μεγάλη κατανάλωση. Τέλος, είναι πολύ εύκολη στον προγραμματισμό της. Το open source community έχει δημιουργήσει βιβλιοθήκες ικανές να προγραμματίσουν την συγκεκριμένη πλακέτα από το Arduino IDE, ένα περιβάλλον γνωστό για τον προγραμματισμό του μικροελεγκτή Arduino. [15]

Για τη συγκεκριμένη πτυχιακή εργασία θα χρησιμοποιηθεί ο NodeMCU Devkit V1.0 (NoceMCU V2), ο δεύτερος στη γενιά του μετά τον NodeMCU Devkit V0.9, έχοντας πάνω τον αναβαθμισμένο esp8266-12e (WiFi module).

Εικόνα 19: NodeMCU V2.
Μικροελεγκτής NodeMCU

3.4 NodeMCU (λογισμικό)

Όπως προαναφέρθηκε, για τον προγραμματισμό της συγκεκριμένης πλακέτας χρησιμοποιήθηκε το περιβάλλον προγραμματισμού του Arduino. Το Ολοκληρωμένο Περιβάλλον Ανάπτυξης Arduino IDE είναι ένα multi-platform πρόγραμμα, δηλαδή μπορεί να τρέξει σε διαφορετικά λογισμικά. Το συγκεκριμένο πρόγραμμα χρησιμοποιείται για να προγραμματιστούν πλακέτες σαν τον Arduino, όμως ο χρήστης με τις κατάλληλες βιβλιοθήκες, μπορεί να γράψει κώδικα για κάθε συμβατή πλακέτα. Είναι άξιο να σημειωθεί, ότι αυτό το περιβάλλον ανάπτυξης χρησιμοποιεί την γλώσσα C και C++. [16]

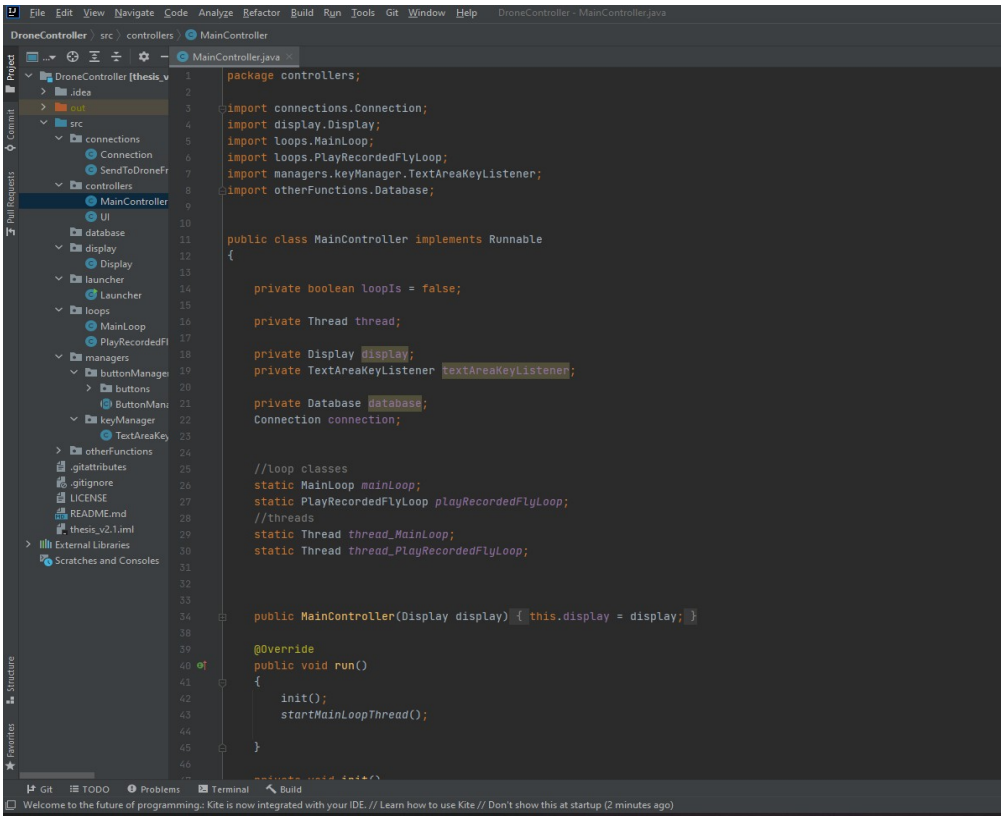


Εικόνα 20: Arduino IDE. Περιβάλλον προγραμματισμού για Arduino και άλλες πλακέτες

3.5 PC Drone Controller (λογισμικό)

Για τη δημιουργία του προγράμματος πλοήγησης του drone, μέσω ηλεκτρονικού υπολογιστή, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java και το περιβάλλον προγραμματισμού (IDE) IntelliJ. Η Java είναι μια προγραμματιστική γλώσσα, που βασίζεται στη δημιουργία κλάσεων και αντικειμένων. Είναι μια γενικού σκοπού γλώσσα και τα προγράμματα της μπορούν να τρέξουν σε οποιαδήποτε συσκευή υποστηρίζει τη γλώσσα μέσω του JVM (Java Virtual Machine)[17].

Για την δημιουργία του λογισμικού χρησιμοποιήθηκε το IDE, της JetBrain, IntelliJ. Η JetBrain είναι μία εταιρεία δημιουργίας λογισμικού, η οποία στοχεύει να ικανοποιήσει τις ανάγκες των προγραμματιστών λογισμικού και των project managers[18]. Ένα από τα καλύτερα IDE για την δημιουργία λογισμικών είναι διαθέσιμο στους φοιτητές δωρεάν.



Εικόνα 21: IntelliJ UI. Περιβάλλον προγραμματισμού από την JetBrain

Κεφάλαιο 4: Υλοποίηση πτυχιακής εργασίας

Σε αυτό το κεφάλαιο θα εξεταστεί αναλυτικότερα τα εργαλεία που χρησιμοποιήθηκαν, τον ρόλο που είχε το κάθε ένα, καθώς και τα συνολικά βήματα που ακολουθήθηκαν για την ολοκλήρωση της εργασίας.

Στην αρχή, αναλύεται η διαδικασία συναρμολόγησης του quad, ο τρόπος δηλαδή με τον οποίο κατασκευάζεται βήμα προς βήμα. Στην συνέχεια, θα ακολουθήσει ο προγραμματισμός του Flight Controller μέσω του Cleanflight. Ολοκληρώνοντας την συναρμολόγηση και τον προγραμματισμό του drone, θα γίνει μία αναφορά στα σημαντικά κομμάτια του κώδικα του NodeMCU αλλά και του PC Drone Controller (πρόγραμμα καθοδήγησης quad μέσω ηλεκτρονικού υπολογιστή).

4.1 Συναρμολόγηση quad

Για την συναρμολόγηση του quad, πέρα από το βασικό υλικό που αναφέρθηκε στο κεφάλαιο 3.1 θα πρέπει να υπάρχουν και τα επακόλουθα εργαλεία:

1. Bullet Connectors. Για την εύκολη συνδεσμολογία των ESC με τα motors.

Εικόνα 22: Bullet Connector. Ακροδέκτες μπανάνα

2. Male XT60 Connector. Θα συγκολληθεί πάνω στην πλακέτα διανομής ρεύματος (PDB).

Εικόνα 23: Αρσενικό βύσμα XT60

3. Soldering Kit. Εργαλείο απαραίτητο για τη συγκόλληση. Είναι αναγκαία η γνώση συγκόλλησης με καλάι και η τήρηση των μέτρων ασφαλείας .

Εικόνα 24: Solder Iron. Σίδερο κόλλησης

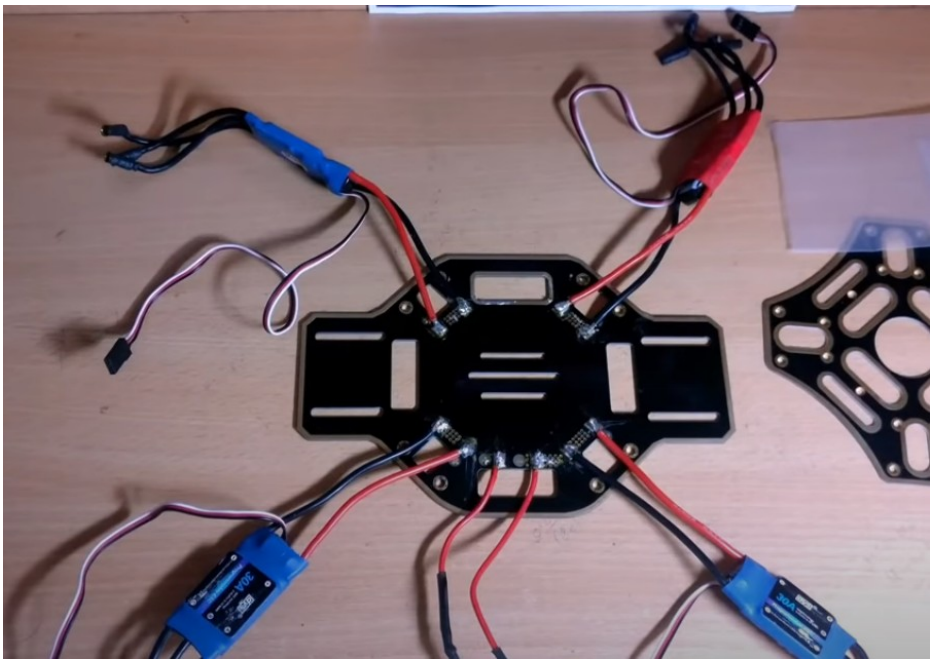
4. Tire ups. Για την ασφαλής πρόσδεση των διαφόρων εξαρτημάτων πάνω στο quad

Εικόνα 25: Tire ups. Δεματικά

Αρχικά, πρέπει να μπουν τα bullet connectors στις άκρες των ESC και των motors έτσι ώστε να γίνεται εύκολα η αλλαγή των συνδέσεων.

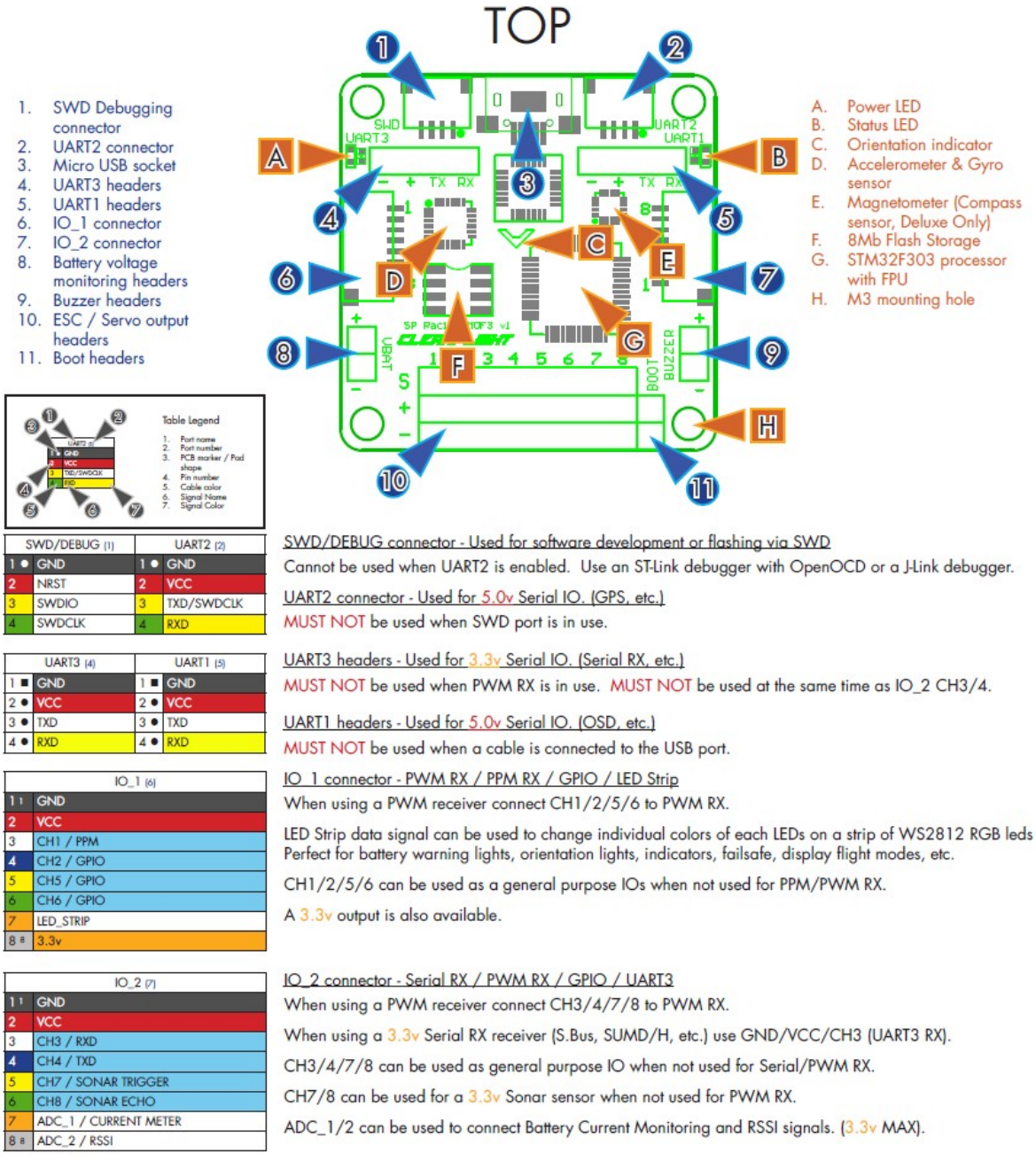
Εικόνα 26: Τα μοτέρ και οι ελεγκτές ταχύτητας με τους ακροδέκτες μπανάνα για ευκολότερη σύνδεση μεταξύ τους.

Ύστερα, γίνεται η κόλληση των ESC και του αρσενικού XT60 πάνω στο power distribution board (PDB) όπως φαίνεται στην εικόνα 27. Το χρώμα υποδηλώνει που πρέπει να μπει το κάθε καλώδιο, μαύρο για (-), κόκκινο για (+).



Εικόνα 27: Πίνακας διανομής ρεύματος με συνδεδεμένους τους ελεγκτές ταχύτητας.

Αφού έχουν γίνει οι απαραίτητες κολλήσεις, ήρθε η ώρα να συνδεθεί ο Flight Controller με τα motors και τον receiver. Με την καθοδήγηση του εγχειριδίου ο ESC και ο receiver συνδέονται πάνω στον FC. Στην συγκεκριμένη εργασία χρησιμοποιείται ο SP Racing F3 και θα ακολουθείται το δικό του manual.



Εικόνα 28: Εγχειρίδιο ελεγκτή πτήσης FC SP Racing F3.

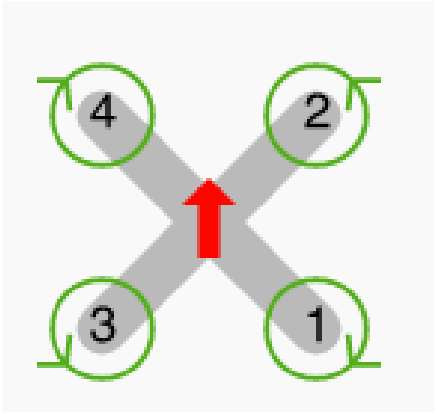
Οδηγός στα ελληνικά:

1. SWD/DEBUG connector - θύρα για τον προγραμματισμό ή την αναβάθμιση του λογισμικού.
2. Universal Asynchronous Receiver - Transmitter 2(UART) - αποστολή και συλλογή πληροφοριών μεταξύ πομπού και δέκτη.
3. Θύρα Micro USB - τροφοδοσία και σύνδεση της πλακέτας με τον υπολογιστή.
4. UART 3.
5. UART 1.
6. Input / Output (IO) connectors - συνδέσεις εισόδου / εξόδου.
7. IO connectors.
8. Pins για την παρακολούθηση του ποσοστού της μπαταρίας.
9. Pins για την σύνδεση με buzzer με σκοπό την χρήση ήχου για προειδοποίηση.
10. Pins για την σύνδεση των ESC.
11. Pins έτσι ώστε να θέσουμε τον FC σε boot λειτουργία και να εγκαταστήσουμε / αναβαθμίσουμε το λογισμικό μέσω του USB.

A. Led τροφοδοσίας - ένδειξη για το αν ο FC τροφοδοτείται επαρκώς
B. Led κατάστασης - δείχνει σε τι κατάσταση βρίσκεται ο FC ανάλογα με τον ρυθμό που αναβοσβήνει.
C. Βέλος προσανατολισμού - δείχνει στον χρήστη που είναι η προκαθορισμένη μπροστά θέση (θέση 0) του FC.
D. Επιταχυνσιόμετρο και γυροσκόπιο.
E. Μαγνητόμετρο.
F. Αποθηκευτικός χώρος.
G. Επεξεργαστής.
H. Τρύπες για την στήριξη σε βάση M3.

Με βάση το παραπάνω εγχειρίδιο θα πρέπει να τοποθετηθούν οι ESC με την σειρά στα pins που βρίσκονται στη θέση (10 από το manual) δίνοντας βάση στην παρακάτω εικόνα.

ΠΡΟΣΟΧΗ: κάθε ESC έχει και ένα βύσμα Dupont 3p, όπως της εικόνας 30 (μαύρο/καφέ: Ground, κόκκινο: 5V, κίτρινο/άσπρο: σήμα). Ο FC δεν πρέπει να παίρνει ρεύμα από τέσσερις διαφορετικές πηγές. Γι' αυτό το λόγο από τους τρεις ESC θα πρέπει να αποσυνδεθεί το κόκκινο καλώδιο.



Εικόνα 30: Πτήση του quad σε σχήμα X

Εικόνα 29: Σύνδεση Dupont 3p

Τέλος, συνδέεται ο Receiver στις θύρες εξόδου (6 και 7 από το manual). Οι έξοδοι του FC μπαίνουν ως εξής:

IO_1.pin1 **CH1** στο GND pin του receiver,

IO_1.pin2 **CH1** στο VCC pin,

IO_1.pin3 **CH1** signal pin,

IO_1.pin4 **CH2** signal pin,

IO_1.pin5 **CH5** signal pin,

IO_2.pin3 **CH3** signal pin,

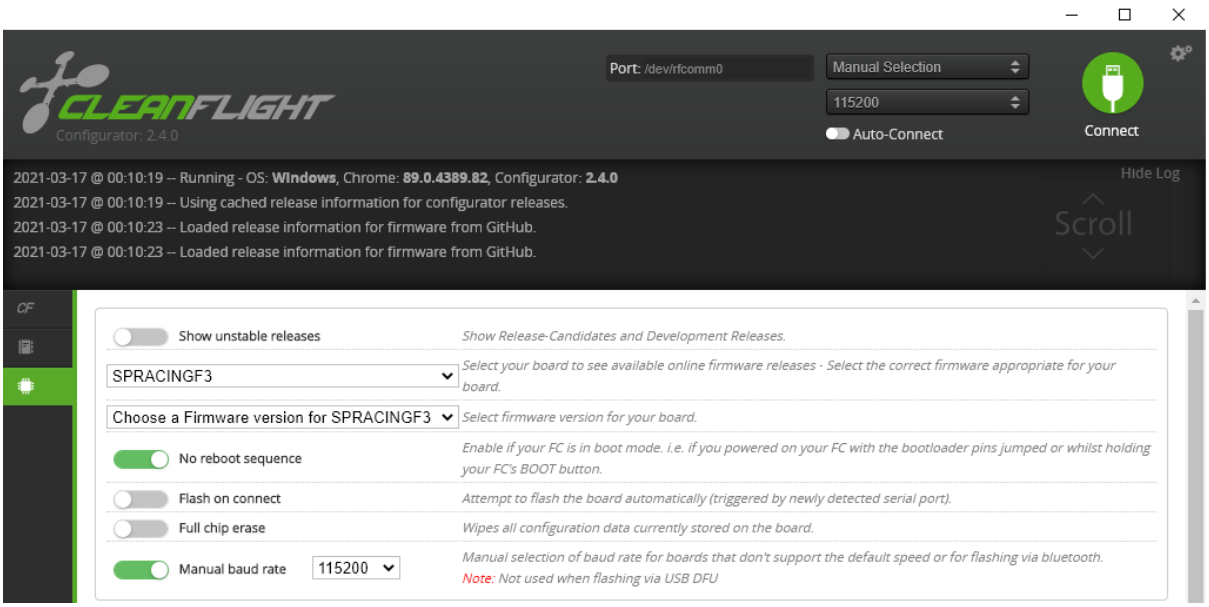
IO_2.pin4 **CH4** signal pin.

Εικόνα 31: Δέκτης σήματος FS-iA6B

4.2 Προγραμματισμός FC και ολοκλήρωση συναρμολόγησης quad

Ξεκινώντας με το Cleanflight πρέπει να οριστεί ποιος είναι ο Flight Controller που χρησιμοποιείτε καθώς και ποια είναι η επιθυμητή του firmware έκδοση.

Μετέπειτα, φορτώνεται το firmware και γίνεται flash. Αφού ολοκληρωθεί η διαδικασία πρέπει να γίνει έλεγχος της θύρας επικοινωνίας (στην οθόνη πάνω δεξιά) στην οποία έχει συνδεθεί ο FC και πατώντας το Connect θα εμφανιστεί η αρχή οθόνη των ρυθμίσεων.



Εικόνα 32: Εγκατάσταση λογισμικού για τον Flight Controller μέσω CleanFlight.

Στην αρχική οθόνη μπορεί να δει κανείς το drone και κάποιες βασικές πληροφορίες. Από την οθόνη αυτή θα γίνει η βαθμονόμηση του επιταχυνσιόμετρου, πατώντας το κουμπί Calibrate accelerometer. Είναι σημαντικό, πριν γίνει η παραπάνω διαδικασία, το drone να τοποθετηθεί σε μια επίπεδη επιφάνεια.

Εικόνα 33: Αρχικό παράθυρο του Cleanflight μετά την είσοδο.

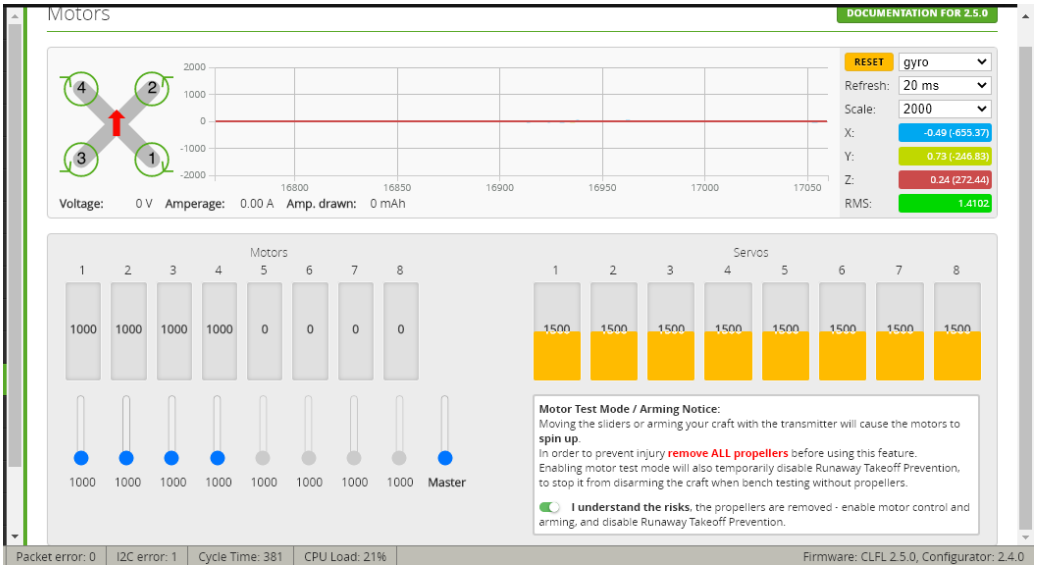
Στην συνέχεια, στο παράθυρο Configuration θα πρέπει να μπει η επιλογή **PWM** για το **ESC/Motor protocol**, **Quad X** στο **Mixer**, καθώς και η επιλογή **PWM RX input** για τον **Receiver.**

Έπειτα, στο παράθυρο Modes προστίθεται το **ARM** και **ANGLE** mode για τον εύκολο τηλεχειρισμό του quad (βλέπε Εικόνα 34). Και τα δύο modes μπορούν να τα τεθούν στο κανάλι AUX 1. Το ARM ενεργοποιεί τους κινητήρες και βάζει το drone σε λειτουργία πτήσης. Με το ANGLE mode το drone χρησιμοποιεί το επιταχυνσιόμετρο και το γυροσκόπιο, έτσι ώστε να ισορροπεί. Είναι μια πολύ καλή επιλογή για αρχάριους που ξεκινάνε να μαθαίνουν την πλοήγηση ενός UAV

Εικόνα 34: Arm και Angle mode στην καρτέλα Modes του Cleanflight.

Τέλος στην καρτέλα Motors μπορεί να γίνει η βαθμονόμηση των κινητήρων. Κατά τη συναρμολόγηση του quad, στο προηγούμενο υποκεφάλαιο (5.1), δεν δέθηκαν οι ESC με τους κινητήρες. Για αυτόν τον λόγο πρέπει να βιδωθούν οι κινητήρες πάνω στο frame και να συνδεθούν με τους Ελεγκτές Ταχύτητας (ESC) μέσω των bullet connectors. Δεν είναι απαραίτητο να τοποθετηθεί τα βύσμα σε συγκεκριμένη θέση καθώς η σωστή σύνδεση θα ακολουθήσει αργότερα.

Πριν γίνει arm πρέπει να δοθεί ιδιαίτερη σημασία στις οδηγίες και να αφαιρεθούν οι έλικες. Στην συνέχεια, με αποσυνδεδεμένη την μπαταρία, γίνεται arm του drone πατώντας το κουμπί κάτω δεξιά (βλέπε Εικόνα 35) και πρέπει να τεθεί η μπάρα Master, που ελέγχει την τιμή όλων των κινητήρων, στην τιμή 2000. Μετά από αυτό επανατοποθετείται η μπαταρία. Αφού ακολουθήσει χαρακτηριστικός ήχος, χαμηλώνεται η μπάρα στο 1000. Ένας δεύτερος χαρακτηριστικός ήχος θα σημάνει πως η βαθμονόμηση των κινητήρων έχει επιτευχθεί.



Εικόνα 35: Καρτέλα Motors του Cleanflight με ενεργοποιημένο το κουμπί arm

Στο σημείο αυτό, γίνεται η σωστή τοποθέτηση των bullet connectors, έτσι ώστε οι ρότορες να γυρνάνε με τον επιθυμητό τρόπο. Δίνοντας σήμα σε έναν - έναν τους κινητήρες μπορεί να παρατηρηθεί η στροφή τους. Στην περίπτωση που δεν γυρίζει σύμφωνα με την Εικόνα 29 πρέπει να αλλάξει η σύνδεση των δύο από τους τρείς bullet connectors.

4.3 Προγραμματισμός NodeMCU (Arduino IDE, C++)

Για τον προγραμματισμό του NodeMCU θα επιστρατευτεί το πρόγραμμα ανάπτυξης κώδικα Arduino IDE.

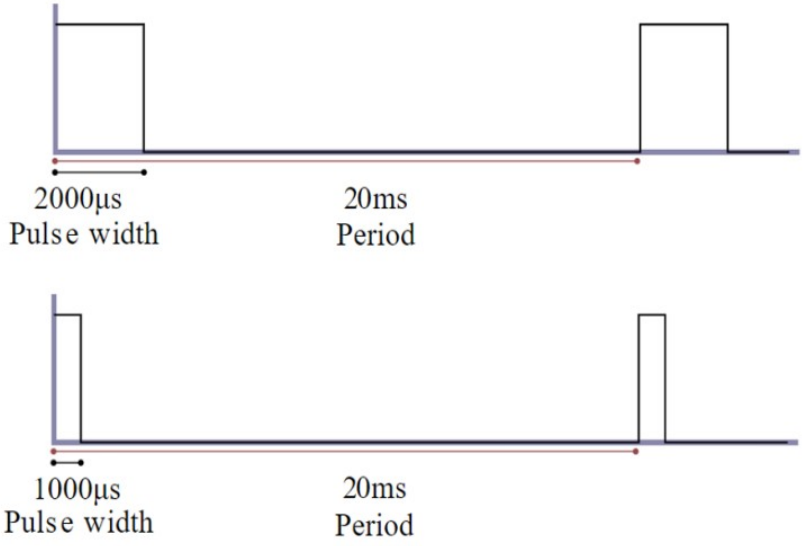
Ξεκινώντας πρέπει να οριστεί στο IDE η πλακέτα που θα προγραμματιστεί. Αυτό επιτυγχάνεται μετά από τρία βήματα.

1. File> Preferences > Additional Boards Manager URLs:
   (paste)
   https://dl.espressif.com/dl/package_esp32_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json > restart IDE
2. Tools > Boards > Boards Manager… > (download latest version) esp8266
3. Tools > Boards > Generic ESP8266 Module

Ο προγραμματισμός του μικροελεγκτή ξεκινάει με την αρχικοποίηση των pins της πλακέτας ανάλογα με τη λειτουργία που θέλουμε να υλοποιούν. Στην συνέχεια γίνεται η μετατροπή του esp σε access point (πύλη για τη σύνδεση συσκευών) και τέλος γράφουμε τον κώδικα συλλογής πληροφοριών από τον PC Drone Controller (λογισμικό που χειρίζεται το drone από τον ηλεκτρονικό υπολογιστή). Ο κώδικας αναλυτικά μπορεί να βρεθεί στο παράρτημα Δ, σε μορφή .txt στα συνημμένα έγγραφα και στο github repository: https://github.com/AimiKass/DroneControllerESP

Στη συνάρτηση void setup() αρχικοποιούνται οι servo μεταβλητές με τα αντίστοιχα pins. Έπειτα ενεργοποιείται το μικροτσίπ ESP8266 σαν access point. Με αυτόν τον τρόπο δημιουργεί ένα αυτόνομο ασύρματο δίκτυο που θα μπορεί ο υπολογιστής να συνδεθεί και να ανταλλάξουν πληροφορίες (βλέπε Παράρτημα Α).

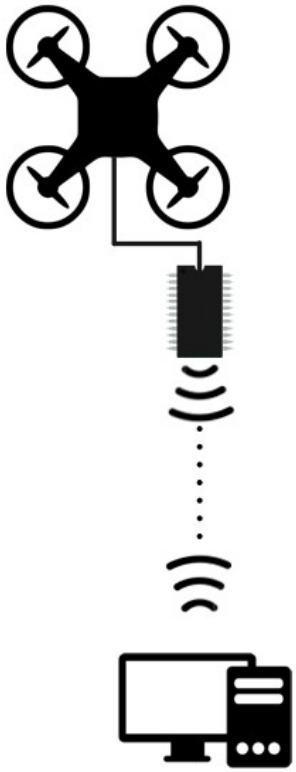
Στη βασική συνάρτηση, void loop(), εξετάζεται η σύνδεση του PC Drone Controller με τον μικροελεγκτή και ξεκινάει η λήψη και επεξεργασία δεδομένων. Στην περίπτωση που ο χρήστης πατήσει το κουμπί "P", τα δεδομένα που δέχεται το πρόγραμμα από τον PC Drone Controller είναι τέσσερις ακέραιοι αριθμοί της μορφής, 0 0 123 12. Κάνοντας την πράξη,123*256 + 12, αυτό το νούμερο μεταφράζεται ως 31500 όπου στην συνέχεια αποθηκεύεται στην μεταβλητή servoData (βλέπε Παράρτημα Α). Έπειτα η μεταβλητή αυτή σπάει σε δύο ακέραιους αριθμούς, το πρώτο ψηφίο, το 3 και τα υπόλοιπα 4, το 1500. Ανάλογα με τον πρώτο ακέραιο, ο κώδικας στέλνει τον δεύτερο σε συγκεκριμένη λειτουργία. Για παράδειγμα, το 3 υποδεικνύει πως η throttle λειτουργία (το γκάζι) θα πάρει την τιμή 1500. Με αυτόν τον τρόπο προσπαθεί να παρομοιάσει το PWM σήμα που στέλνει ο receiver στο drone. Πιο συγκεκριμένα, το κάθε κανάλι του receiver στέλνει ένα σήμα PWM ανάλογα με τη θέση του μοχλού στον πομπό. Το σήμα αυτό είναι στα 50Hz με παλμούς HIGH και duty cycle από 1000μsec (1msec) έως 2000μsec (2msec).



Εικόνα 36: Γραφική απεικόνιση σήματος PWM

Για παράδειγμα, όταν ο μοχλός για το κανάλι με τη λειτουργία Throttle είναι στην χαμηλότερη θέση, ο receiver εκπέμπει ένα PWM σήμα με duty cycle 1000μsec και οι κινητήρες περιστρέφονται με την χαμηλότερη ταχύτητα. Όταν ο χρήστης ανεβάσει τον μοχλό στην υψηλότερη θέση, το σήμα που θα σταλεί θα έχει duty cycle 2000μsec και οι κινητήρες θα περιστρέφονται με την μέγιστη ταχύτητα τους.

Ο μικροελεγκτής NodeMCU έχει τοποθετηθεί και προγραμματιστεί με τέτοιο τρόπο ώστε να παρομοιάσει την λειτουργία ενός receiver. Έτσι έχει επιτευχθεί η πλήρος τηλεκατεύθηνσει του drone μέσω ηλεκτονικού υπολογιστή.



Εικόνα 37: Γραφική απεικόνιση σύνδεσης Controller-Receiver, PC Remote Controller-NodeMCU, NodeMCU-Receiver, NodeMCU-Drone

4.4 Προγραμματισμός PC Drone Controller (IntelliJ IDE, Java)

Για την ανάπτυξη του PC Drone Controller χρησιμοποιήθηκε το λογισμικό της JetBrains IntelliJ και η γλώσσα προγραμματισμού Java, Version 15.0.1.

Το πρόγραμμα, ανοίγοντας ένα socket connection επικοινωνεί με τον ESP8266 και στέλνει το κατάλληλο σήμα ανάλογα με το τι θα πληκτρολογήσει ο χρήστης. Ο κώδικας αναλυτικά μπορεί να βρεθεί σε μορφή .txt στα συνημμένα έγγραφα και στο github repository: https://github.com/AimiKass/DroneController

Με την κλάση Connection, θέτοντας την IP του ESP8266 και το port που θα χρησιμοποιηθεί για την επικοινωνία, πραγματοποιείται η σύνδεση μεταξύ εφαρμογής και μικροελεγκτή (βλέπε Παράρτημα Ε). Στη κλάση MainLoop έχει υλοποιηθεί η συνάρτηση startLoop() η οποία επιτρέπει την σταθερή μετάδοση των δεδομένων (βλέπε Παράρτημα Ι). Με την βοήθεια της συνάρτησης αυτής το πρόγραμμα στέλνει με σταθερό ρυθμό τα δεδομένα έτσι ώστε να είναι εφικτή η χρήση του προγράμματος από κάθε μηχανή, ανεξαρτήτως των πόρων που διαθέτει.

Με την έναρξη του προγράμματος γίνεται η σύνδεση με τον ESP και πλέον ο χρήστης μπορεί να δώσει οδηγίες μέσα από το textArea.

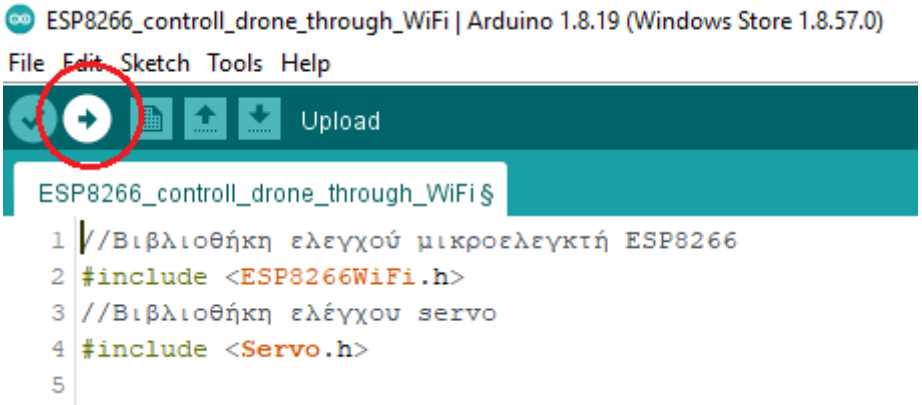
A και **D** για Roll,
W και **S** για Pitch,
P και **L** για Throttle,
Q και **E** για Yaw,
Ctrl για Arm,
Space για Disarm,
Shift (παρατεταμένα) για να στείλει σταθερό σήμα σε ένα από τα παραπάνω κανάλια.

Πέρα από την βασική πλοήγηση, το πρόγραμμα προσφέρει μια επιπλέον λειτουργία. Στο παράθυρο πλοήγησης υπάρχουν 2 κουμπιά, Start Recording και Play Recorded Flight (βλέπε Εικόνα 42). Το πρόγραμμα είναι ικανό να καταγράφει και να αναπαράγει τη διαδρομή που θα καθορίσει ο χρήστης. Η κλάση PlayRecordedFly χρησιμοποιεί την ίδια συνάρτηση με την κλάση MainLoop για να αναπαράγει τα κουμπιά που καταγράφηκαν.

4.5 Τελικό στάδιο

Αφού έχουν ολοκληρωθεί τα προηγούμενα στάδια υπάρχουν κάποια απλά βήματα μέχρι να γίνει εφικτή η τηλεκατεύθυνση του quad από τον ηλεκτρονικό υπολογιστή.

Στο κομμάτι του λογισμικού: Για να γίνει εφικτή η σύνδεση μεταξύ του NodeMCU και του PC Drone Controller θα πρέπει να θέσουμε την IP του μικροελεγκτή στο πρόγραμμα πλοήγησης. Αφού γίνει upload ο κώδικας στον μικροελεγκτή (βλέπε Εικόνα 38) μπορεί να παρατηρηθεί στο Serial Monitor του Arduino IDE η IP που έχει πάρει ο ESP στο τοπικό δίκτυο

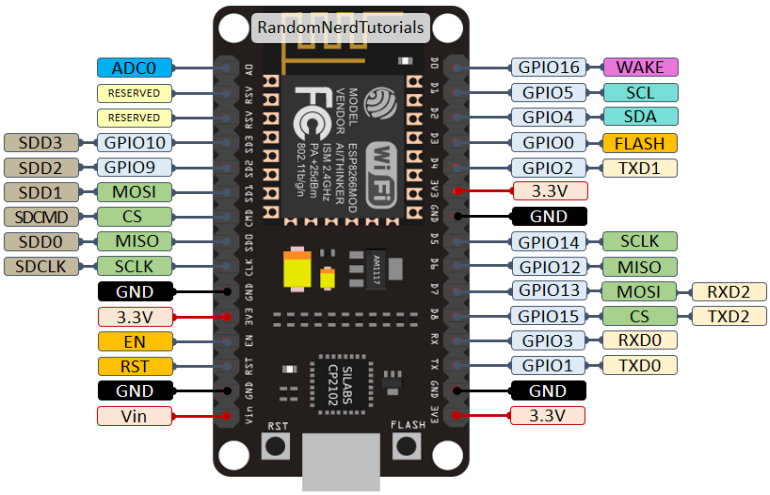


Εικόνα 38: Περιβάλλον προγραμματισμού Arduino IDE, κουμπί upload.

Αφού γίνει η σύνδεση στο τοπικό δίκτυο, μετά από λίγα δευτερόλεπτα εμφανίζεται η IP (192.168.4.1) του μικροελεγκτή (Βλέπε Εικόνα 39). Με αυτό το μοναδικό νούμερο ο PC Drone Controller θα μπορέσει να αναγνωρίσει τον ESP και να κάνει την σύνδεση.

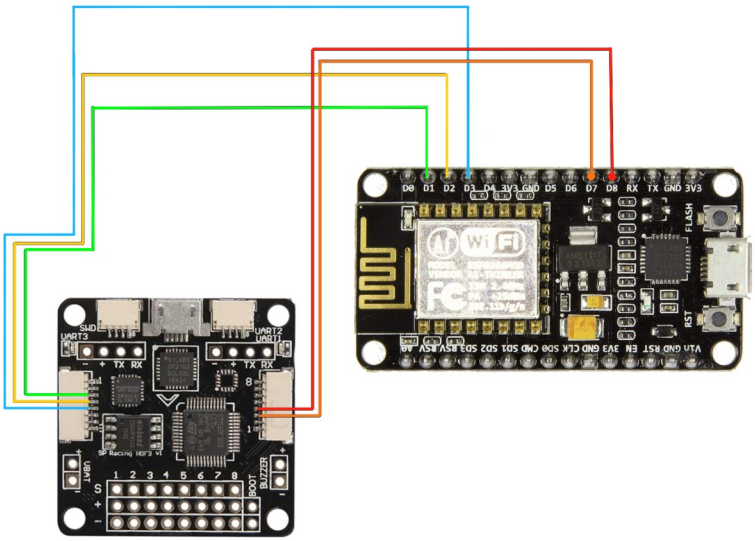
Εικόνα 39: Αποτελέσματα εξόδου σειριακής οθόνης του Arduino IDE.

Στο κομμάτι του υλικού: Θα πρέπει να συνδεθεί ο μικροελεγκτή με τον Flight Controller και τον receiver. Για την διευκόλυνση της σύνδεσης υπάρχουν οι παρακάτω εικόνες .



Εικόνα 40: Αναπαράσταση εισόδων-εξόδων του NodeMCU.

Οι συνδέσεις που πρέπει να γίνουν είναι οι εξής:



Εικόνα 41: Συνδεσμολογία μεταξύ receiver, FC και NodeMCU

NodeMCU με FC:

- GPIO5(D1) με IO_1.pin3 (βλέπε Εικόνα 28)
- GPIO4(D2) με IO_1.pin4 (βλέπε Εικόνα 28)
- GPIO13(D7) με IO_2.pin3 (βλέπε Εικόνα 28)
- GPIO15(D8) με IO_2.pin4 (βλέπε Εικόνα 28)
- GPIO14(D5) με IO_1.pin5 (βλέπε Εικόνα 28)

Τροφοδοσία NodeMCU:

- Vin με μία θετική έξοδος ενός από τους τέσσερις ESC
- GND με ένα από τα ground pins(10) του FC (βλέπε Εικόνα 28).

Πλέον είναι τα πάντα έτοιμα για την πρώτη δοκιμή. Πρώτα πρέπει να συνδεθεί ο μικροελεγκτής στο quad και μετέπειτα να γίνει η τροφοδοσία του quad. Μέσα σε λίγα δευτερόλεπτα ο ESP8266 θα έχει πάρει την ip του. Αμέσως μετά μπορεί να τρέξει ο PC Drone Controller (το java πρόγραμμα τηλεκατεύθυνσης του drone) και θα εμφανιστεί το παράθυρο πλοήγησης όπως στην εικόνα 37. Εκεί πρέπει να συμπληρωθεί το port (δόθηκε κατα τον προγραμματισμό του μικροελεγκτή) και η IP. Τέλος πατώντας το κουμπί connect μπορεί πλέον ο χειριστής να τηλεκατευθύνει το quad από τον ηλεκτρονικό του υπολογιστή.

Η πλοήγηση γίνεται μέσω του Text Area και των παρακάτω κουμπιών.

A,D: Κύλιση δεξιά-αριστερά γύρω από τον οριζόντιο άξονα του quad (Roll)

W,S: Κίνηση μπρός-πίσω (Pitch)

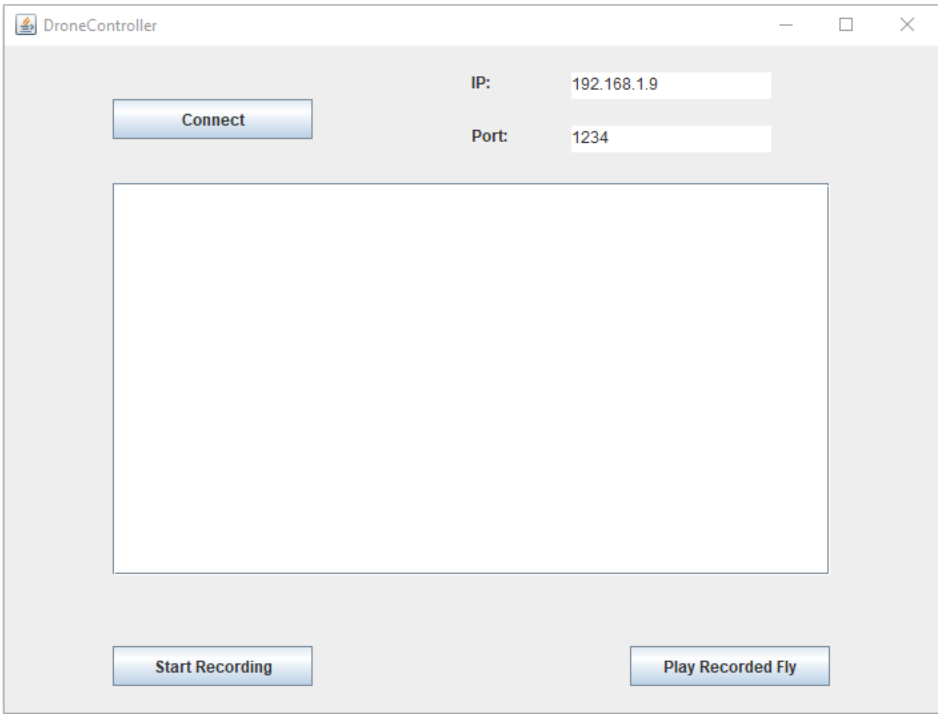
Q,E: Περιστροφή δεξιά-αριστερά γύρω από τον κατακόρυφο άξονα του drone (Yaw)

P,L: Ρύθμιση του υψομέτρου ή γκάζι (Throttle)

Ctrl: ARM

Space: DISARM και θέτει το Throttle στο ελάχιστο σήμα

Shift (παρατεταμένα)**:** για να στείλει σταθερό σήμα σε ένα από τα παραπάνω κανάλια.



Εικόνα 42: Παράθυρο πλοήγησης του drone. PC Drone Controller.

Κεφάλαιο 5: Συμπεράσματα

Η συγκεκριμένη πτυχιακή εργασία ήταν μια πολύ καλή ευκαιρία στην εμβάθυνση των γνώσεων σχετικά με τα drones, τόσο στο υλικό όσο και στο λογισμικό κομμάτι. Επιπλέον, βοήθησε στην κατανόηση του τρόπου που λειτουργεί το κάθε κομμάτι μεμονωμένα αλλά και σε συνεργασία με τα υπόλοιπα. Μελετώντας και κατανοώντας, λοιπόν, τα δύο στοιχεία, το υλικό και το λογισμικό, έγινε δυνατή η αναβάθμιση του quad σύμφωνα με τις ανάγκες της εργασίας

5.1 Κατανόηση και επίλυση βασικών προβλημάτων

Για την επικοινωνία δύο συσκευών μέσω WiFi είναι αναγκαία η χρήση των sockets. Το πρόγραμμα αποστολής δεδομένων (πομπός) μπορεί να δημιουργηθεί με μια γλώσσα προγραμματισμού, αλλά ποιος θα είναι ο δέκτης; Ο esp8266 είναι αρκετά γνωστός για IoT εφαρμογές και φάνηκε μια πολύ καλή λύση. Ειδικά πάνω στην πλακέτα NodeMCU με 12 ψηφιακά κανάλια, όπου τα 11 από αυτά διαθέτουν και την λειτουργία των Interrupts. Αφού ξεκίνησε ο προγραμματισμός του PC Drone Controller και έγινε η σύζευξή με τον NodeMCU μέσω των sockets έγινε το πρώτο βήμα για την προσομοίωση πομπού - δέκτη.

Το επόμενο πρόβλημα ήταν η αποστολή σήματος με σταθερή συχνότητα. Την στιγμή που ο χρήστης δίνει εντολή κίνησης, το πρόγραμμα πρέπει να στείλει δεδομένα για τον συγκεκριμένο χρόνο που κάποιο από τα πλήκτρα είναι πατημένο. Για παράδειγμα αν ο χρήστης πατήσει το πλήκτρο “q” για 0,5 δευτερόλεπτα, το πρόγραμμα πρέπει να στείλει την εντολή ότι το drone θα εκτελεί την κίνηση περιστροφή αριστερά για τον ίδιο χρόνο. Αν το πρόγραμμα στέλνει τα δεδομένα στον μικροελεγκτή με έναν απλό βρόχο επανάληψης, for loop ή do-while, η συχνότητα με την οποία επαναλαμβάνεται ο βρόχος αυτός σχετίζεται άμεσα με τους πόρους του εκάστοτε μηχανήματος. Για παράδειγμα, αν δύο υπολογιστές που το υλικό τους δεν έχουν την ίδια τεχνολογική εξέλιξη, τρέξουν την ίδια for loop, ο υπολογιστής με το πιο σύγχρονο υλικό (μεγαλύτερη επεξεργαστική ισχύ) θα εκτελέσει τον βρόχο πιο γρήγορα. Η λύση του προβλήματος βρέθηκε μέσα από τον προγραμματισμό παιχνιδιών και στο πώς διατηρούν σταθερά τα Frames, γνωστό και ως Frame Per Second (FPS). Μία αντίστοιχη λειτουργία αναπτύχθηκε στον PC Drone Controller για να στέλνει με συγκεκριμένη συχνότητα το σήμα στον NodeMCU από οποιαδήποτε συσκευή και να τρέξει ανεξαρτήτως πόρων.

5.2 Μελλοντική εξέλιξη εργασίας

Το λογισμικό πλοήγησης του quad (PC Drone Controller) στη παρούσα κατάσταση δεν διαθέτει ούτε αρκετές λειτουργίες ούτε αρκετά οπτικά χαρακτηριστικά. Μια σημαντική εξέλιξη θα ήταν ο χρήστης να μπορεί να έχει πρόσβαση σε περισσότερα δεδομένα κατά τη διάρκεια της χρήσης του προγράμματος. Τέτοια χαρακτηριστικά μπορούν να είναι, η τηλεμετρία (Telemetry) ή οι συντεταγμένες xyz που παίρνει το drone κατά την διάρκεια της πτήσης του στον χώρο .

Επιπλέον, θα ήταν αρκετά χρήσιμη η προσθήκη νέων λειτουργιών. Η λειτουργία δημιουργίας και αναπαραγωγής μιας διαδρομής θα μπορούσε να εξελιχθεί και να υπάρχει μία λίστα από αποθηκευμένες/αγαπημένες διαδρομές.

Μία σημαντική αναβάθμιση της εργασίας θα ήταν η προσθήκη επιπλέον περιφερειακών συστημάτων. Ένα τέτοιο σύστημα θα μπορούσε να είναι μια κάμερα.

Με την προσθήκη μιας κάμερας στο drone ο χρήστης θα μπορεί να έχει την ικανότητα να βλέπει που οδηγεί το UAV από τον ηλεκτρονικό του υπολογιστή. Μία τόσο σημαντική αλλαγή θα αποτελούσε μεγάλο προνόμιο, καθώς δεν θα ήταν αναγκαία η φυσική του παρουσία στον χώρο κατά την πλοήγηση.

Βιβλιογραφία/Πηγές

[1] Μαγκίρης, Β. (1998). *UAV και MAV: Τα ιπτάμενα ρομπότ. Περισκόπιο της Επιστήμης*, τεύχος 219 , σσ. 66-75.

[2] Κοσμέτος, Ε. (2018). *Παραμετροποίηση συστήματος μη επανδρωμένου ηλεκτροκινητήριου ανεμοπτέρου σε εφαρμογές τρισδιάστατης αποτύπωσης*, Διπλωματική εργασία, Πολυτεχνείο Κρήτης.

[3] Πραπόπουλος, Μ. (2020). Drone Σταθερής πτέρυγας ή Πολυκόπτερο; Τι επιλέγουμε; Ανακτήθηκε από: Prapopoulos.com

[4] McKenna, A. *The Future of Drone Use: Opportunities and Threats from Ethical and Legal Perspectives*. Σελίδα 355. Ανακτήθηκε από: Wikipedia.

[5] (2014) *Iranian student create drone that saves lives in-stead of dropping bombs.* Άρθρο που ανακτήθηκε από: THE OBSERVER.

[6] (2014) *Ambulance drone TU Delf drastically increases the chance of survival in the event of cardiac arrest.* Άρθρο που ανακτήθηκε από: TUDelf.

[7] J.V. SARATH, P. BINDU (PALAKKAL), K.S. BIJU, 4. L. RANI. (2021). R*eview of antennas used in fpv/wlan applications.* Department of Electronics & Communication Engineering, Kerala Technological University, Govt. Engineering College Wayanad, Kerala, India.

[8] Liang, O. (2015). *RC TX RX Protocols Explained: PWM, PPM, SBUS, DSM2, DSMX, SUMD*. Πηγή: https://oscarliang.com/pwm-ppm-sbus-dsm2-dsmx-sumd-difference/

[9] perrytsao. *Fly a Mini-drone With Your Computer!* Ανακτήθηκε από: instructables circuit.

[10] *Drone Motor Fundamentals - How Brushless Motor Works*. Άρθρο από: Drone Nodes.

[11] *How to choose ESC for quadcopter | Electronic Speed Controller.* Άρθρο από: Drone Nodes.

[12] *How to Choose a Flight Controller for FPV Quadcopter. Άρθρο* από: Drone Nodes.

[13] *How to Choose Best LiPo Battery for your Drone | Quadcopter 4S | 3S | 2S | 1S.* Άρθρο από: Drone Nodes.

[14] (2021) *FPV Drone Flight Controller Firmware Overview.* Ανακτήθηκε από: OscarLiang.com.

[15] Carlos Bento, A. (2018). *IoT: NodeMCU 12e X Arduino Uno, Results of an experimental and comparative survey.* International Journal of Advance Research in Computer Science and Management Studies. Sao Paulo - Brazil.

[16] Louis, L. (2016). *WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH.* Department of Electronics and Communication Engineering, Gujarat Technological University. Ahmedabad, India.

[17] Arnold, K. Gosling, J. Holmes, D. (2005). *THE Java™ Programming Language, Fourth Edition.*

[18] Pech, V. Shatalin, A. Voelter, M. (2013). *JetBrains MPS as a tool for extending Java.*

Top-Down ανάλυση της εφαρμογής

1. Φόρμες τις εφαρμογής:

- Η εφαρμογή έχει μονο μια φόρμα και βρίσκεται στην κλάση Display:

  Έχει 3 text areas, ένα για την ip, ένα για το port και ένα για να διαβάζει το input του χρήστη. Επιπλέον υπάρχουν 3 κουμπιά, το πρώτο είναι για να συνδεθεί η εφαρμογή με τον μικροελεγκτή και τα υπόλοιπα 2 σχετίζονται με την λειτουργία της καταγραφής και αναπαραγωγής πτήσης (Βλέπε Εικόνα 42).

2. Σημαντικές κλάσεις:

- Connection: Κλάση υπεύθυνη για την διασύνδεση της εφαρμογής με τον μικροελεγκτή.
- SendToDroneFrom: Κλάση υπεύθυνη για την αποστολή των δεδομένων στον μικροελεγκτή
- MainController: Είναι ο controller της φόρμας. Διαχειρίζεται τα κουμπιά και αρχικοποιεί τον listener για το textarea εντολών
- MainLoop: Κλάση η οποία στέλνει τα δεδομένα με σταθερή συχνότητα
- PlayRecordedFlyLoop: Κλάση η οποία εκτελεί την καταγεγραμμένη πτήση
- Database: Κλάση η οποία διαχειρίζεται την βάση
- ProcessKeyInput: Κλάση υπεύθυνη για την μετατροπή των κουμπιών που θα δώσει ο χρήστης στην κατάλληλη μορφή έτσι ώστε ο μικροελεγκτής να τα επεξεργαστεί σωτά.
- ConnectionBtnActionListener: Κλάση υπεύθυνη για το κουμπί connection.
- RecordingBtnActionListener: Κλάση υπεύθυνη για το κουμπί της καταγραφής και αναπαραγωγής πτήσης

3. Σημαντικές συναρτήσεις:

Στη κλάση Connection

- public Connection()
  Ανοίγει νέο socket
- public Socket getSocket()
  Επιστρέφει το socket

Στη κλάση SendToDroneFrom

- public void directions(int data)
  Στέλνει τα δεδομένα στον μικροελεγκτή
  Στη κλάση MainController
- public static void startRecordedFly()
  Ξεκινάει την αναπαραγωγή της καταγεγραμμένης πτήσης
- public static void startMainLoopThread()
  Ξεκινάει την διαδικασία αποστολής δεδομένων στον μικροελεγκτή από το textArea που πληκτρολογεί ο χρήστης.

Στην κλάση MainLoop

- private void loop()
  Συνάρτηση η οποία στέλνει τα κωδικοποιημένα δεδομένα στον μικροελεγκτή και αν είναι πατημένο το κουμπί για καταγραφή πτήσης αποθηκεύει στην βάση την πτήση.
- private void startLoop()
  Συνάρτηση υπεύθυνη για την κλήση της loop() ανα συγκεκριμένο χρόνο

Στην κλάση PlayRecordedFlyLoop

- private void loop()
  Συνάρτηση η οποία στέλνει τα κωδικοποιημένα δεδομένα στον μικροελεγκτή.
- private void startLoop()
  Συνάρτηση υπεύθυνη για την κλήση της loop() ανα συγκεκριμένο χρόνο

Στην κλάση Database

- public Database()
  Αν δεν υπάρχει η βάση αρχικοποιεί τους φακέλους που αποθηκεύονται τα διάφορα δεδομένα
- public void setRecordedFly(String msg)
  Αποθηκεύει την καταγεγραμμένη πτήση
- public List<ArrayList<Integer>> getRecordedFly()
  Επιστρέφει την αποθηκευμένη πτήση από την βάση
- public void eraseRecordedFly()
  Διαγράφει την καταγεγραμμένη πτήση
- public void setIP(String ip)
  Αποθηκεύει την IP
- public String getIP()
  Επιστρέφει την αποθηκευμένη IP
- public void setPort(String port)

Αποθηκεύει το Port

- public String getPort()
  Επιστρέφει το αποθηκευμένο Port
- public void setRecordingButtonIs(boolean recordingButtonIs)
  Αποθηκεύει το αποτέλεσμα από το κουμπί καταγραφής πτήσης. True στην περίπτωση που είναι πατημένο και false στην περίπτωση που δεν είναι.
- public boolean getRecordingButtonIs()
  Επιστρέφει το αποτέλεσμα από το κουμπί καταγραφής πτήσης

Στην κλάση ProcessKeyInput

- public Set<Integer> values(Set<Integer> keys)
  Συνάρτηση η δέχεται το input του χρήστη και επιστρέφει ένα set από δεδομένα προς αποστολή στον μικροελεγκτή

4. Σημαντικές σταθερές και μεταβλητές:

Στη κλάση Connection

- private Socket socket;
  Μεταβλητή τύπου Socket η οποία κρατάει το socket που έχει ανοίξει ο χρήστης για συγκεκριμένη ip και port

Στη κλάση SendToDroneFrom

- private DataOutputStream dataOutputStream;
  Μεταβλητή τύπου DataOutputStream η οποία επιτρέπει την αποστολή δεδομένων σε συγκεκριμένο socket
- private Socket socket;
  Μεταβλητή τύπου Socket η οποία κρατάει το socket που έχει ανοίξει ο χρήστης για συγκεκριμένη ip και port

Στη κλάση MainController

- private boolean loopIs;
  Μεταβλητή τύπου boolean η οποία καθορίζει την έναρξη και λήξη της συνάρτησης αποστολής δεδομένων στο drone
- private Thread thread;
  Μεταβλητή τύπου thread η οποία ελέγχει το νήμα για την κλάση MainController
- static Thread thread_MainLoop;

Μεταβλητή τύπου thread η οποία ελέγχει το νήμα για την κλάση MainLoop

- static Thread thread_PlayRecordedFlyLoop;
  Μεταβλητή τύπου thread η οποία ελέγχει το νήμα για την κλάση PlayRecordedFlyLoop

Στην κλάση MainLoop

- private boolean exit = false;
  Μεταβλητή τύπου boolean η οποία καθορίζει την έναρξη και λήξη της κλάσης MainLoop

Στην κλάση Database

- final String path_OfRecordedFly = "src/database/recordedFly.txt";
  Μεταβλητή η οποία κρατάει το path για το .txt αρχείο που αποθηκεύεται η καταγεγραμμένη πτήση
- final String path_OfBooleanForRecordingButton = "src/database/recordingButtonIs.txt";
  Μεταβλητή η οποία κρατάει το path για το .txt αρχείο που κρατάει το αν είναι πατημένο το κουμπί για την καταγραφή της πτήσης
- final String path_OfIPStored = "src/database/ip.txt";
  Μεταβλητή η οποία κρατάει το path για το .txt αρχείο που αποθηκεύεται η IP
- final String path_OfPortStored = "src/database/port.txt";
  Μεταβλητή η οποία κρατάει το path για το .txt αρχείο που αποθηκεύεται το port
- File file_recordedFly = new File(path_OfRecordedFly);
  Μεταβλητή τύπου File η χειρίζεται το αρχείο που έχει αποθηκευτεί η καταγεγραμμένη πτήση
- File file_recordingButtonIs = new File(path_OfBooleanForRecordingButton);
  Μεταβλητή τύπου File η χειρίζεται το αρχείο που έχει κρατήσει το αν είναι πατημένο το κουμπί για την καταγραφή της πτήσης
- File file_ip = new File(path_OfIPStored);
  Μεταβλητή τύπου File η χειρίζεται το αρχείο που έχει αποθηκευτεί η IP
- File file_port = new File(path_OfPortStored);
  Μεταβλητή τύπου File η χειρίζεται το αρχείο που έχει αποθηκευτεί το port

Στην κλάση ProcessKeyInput

- private static final int MOTOR_STEP = 10;
  Μεταβλητή η οποία καθορίζει το βήμα με το οποίο θα στέλνονται τα δεδομένα στο drone
- private static final int PITCH_FORWARD_KEY = KeyEvent.VK_W;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “W” του πληκτρολογίου
- private static final int PITCH_BACKWARD_KEY = KeyEvent.VK_S;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “S” του πληκτρολογίου
- private static final int ROLL_LEFT_KEY = KeyEvent.VK_A;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “A” του πληκτρολογίου
- private static final int ROLL_RIGHT_KEY = KeyEvent.VK_D;

Μεταβλητή η οποία κρατάει το event για το κουμπί “D” του πληκτρολογίου

- private static final int YAW_LEFT_KEY = KeyEvent.VK_Q;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “Q” του πληκτρολογίου
- private static final int YAW_RIGHT_KEY = KeyEvent.VK_E;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “E” του πληκτρολογίου
- private static final int THROTTLE_UP_KEY = KeyEvent.VK_P;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “P” του πληκτρολογίου
- private static final int THROTTLE_DOWN_KEY = KeyEvent.VK_L;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “L” του πληκτρολογίου
- private static final int ARM_KEY = KeyEvent.VK_SPACE;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “Space” του πληκτρολογίου
- private static final int DISARM_KEY = KeyEvent.VK_CONTROL;
  Μεταβλητή η οποία κρατάει το event για το κουμπί “Control” του πληκτρολογίου
- private int roll = 11500;
  Μεταβλητή που αρχικοποιεί την τιμή της λειτουργίας roll που θα στέλνεται στο drone
- private int pitch = 21500;
- Μεταβλητή που αρχικοποιεί την τιμή της λειτουργίας pitch που θα στέλνεται στο drone
- private int yaw = 41500;
  Μεταβλητή που αρχικοποιεί την τιμή της λειτουργίας yaw που θα στέλνεται στο drone
- private int throttle = 31000;
  Μεταβλητή που αρχικοποιεί την τιμή της λειτουργίας throttle που θα στέλνεται στο drone
- private int arm = 51000;
  Μεταβλητή που αρχικοποιεί την τιμή της λειτουργίας arm που θα στέλνεται στο drone

Παραρτήματα

Παράρτημα Α (Ο κώδικας του μικροελεγκτή):

```
//Βιβλιοθήκη ελέγχου μικροελεγκτή ESP8266
#include <ESP8266WiFi.h>
//Βιβλιοθήκη ελέγχου servo
#include <Servo.h>

//Αριθμός θύρας σύνδεσης
const int port = 1234;

//Δημιουργία server που "ακούει" τις συνδέσεις από συγκεκριμένη θύρα
// WiFiServer wifiServer(port);
WiFiServer server(port);


//Μεταβλητή που καθορίζει το διάστημα(σε δευτερόλεπτα) που ο
receiver επιτρέπει τον
//DroneController να πάρει ξανά τον χειρισμό
const long interval = 2000;

//Μεταβλητή που κρατάει τον χρόνο πριν την τελευταία αλλαγή των
//interrupt flags
unsigned long previousMillis = 0;

//Μεταβλητή που καθορίζει την τιμή της διαφοράς που θα
//ενεργοποιήσουν τα σήματα interrupt
int static interuptSensitivity = 4;

//Μεταβλητή που διορθώνει το "λάθος" στο σήμα των καναλιών
const int channelError = 0;

//Ορισμός ονομάτων pin εξόδου
//Connect with FlightController

#define CH1_Output 5   //ROLL_FROM_PC
#define CH2_Output 4   //PITCH_FROM_PC
#define CH3_Output 13  //THROTTLE_FROM_PC
#define CH4_Output 15  //YAW_FROM_PC
#define CH5_Output 14   //ARM_FROM_PC
```

```
//Ορισμός των servo που θα στέλνουν σήμα στον Flight Controller
Servo servo_1;
Servo servo_2;
Servo servo_3;
Servo servo_4;
Servo servo_5;

//Μεταβλητές που κρατάνε το παλιό σήμα που δέχεται
//ο μικροελεγκτής από τα interrupts
int CH1_Receiver_Old = 0;
int CH2_Receiver_Old = 0;
int CH3_Receiver_Old = 0;
int CH4_Receiver_Old = 0;

//Boolean μεταβλητές που ενημερώνουν αν το εκάστοτε κανάλι
//δέχεται σήμα από τον receiver ή τον DroneController
bool CH1_Interrupts_Is = false;
bool CH2_Interrupts_Is = false;
bool CH3_Interrupts_Is = false;
bool CH4_Interrupts_Is = false;

void setup() {
 //Αρχικοποίηση των servo με τα αντίστοιχα pins
 servo_1.attach(CH1_Output);
 servo_2.attach(CH2_Output);
 servo_3.attach(CH3_Output);
 servo_4.attach(CH4_Output);
 servo_5.attach(CH5_Output);


 Serial.begin(115200);
 Serial.println();
 Serial.println();
 Serial.println();
 Serial.println();
 Serial.println();

 Serial.print("Setting AP (Access Point)…");
 //Θέτω Acces Point με το παρακάτω όνομα
 WiFi.softAP("ESP32-Access-Point");

 //Παίρνω την ip του AP
 IPAddress IP = WiFi.softAPIP();
 Serial.print("AP IP address: ");
 Serial.println(IP);
```

```
 server.begin();
}
void loop() {
 //Μεταβλητή που ακούει για εισερχόμενους clients
 // WiFiClient client = wifiServer.available();
 WiFiClient client = server.available();


 if (client) {
  //Επιστρέφει true εάν ο πελάτης είναι συνδεδεμένος
  if (client.connected()) { Serial.println("Client Connected"); }

  while (client.connected()) {
   int len = client.available();  //Αριθμός byte που είναι διαθέσιμα
για ανάγνωση
   int servoData = 0;             //Αρχικοποίηση μεταβλητής που θα
αποθηκεύεται το σήμα από τον client

   //Μεταβλητές που ελέγχουν αν ο χειριστής έχει σταθερό σήμα στο
throttle
   int servo3_old = 0;
   int servo3_new = 0;
   int roll = 0;
   int pitch = 0;
   int throttle = 0;
   int yaw = 0;
   int arm = 0;

   while (len > 0) {
    int dataFromClient = client.read();  //Διαβάζει το επόμενο byte
που έχει στείλει ο client για "ανάγνωση"

    if (dataFromClient != -1) {
     if (servoData == 0)
      servoData = dataFromClient * 256;
     else {
      servoData = servoData + dataFromClient;

      //Παίρνει το πρώτο ακέραιο αριθμό που καθορίζει ποια
      //θα είναι η λειτουργία που θα εκτελεστεί
      switch (servoData / 10000) {
       case 1:  //Λειτουργία ROLL
        //"Γράφει" το σήμα που έχει στείλει ο χρήστης, στο pin που
        //ακούει ο Flight Controller για την λειτουργία ROLL
        roll = (servoData % 10000);
        servo_1.writeMicroseconds((servoData % 10000) -
channelError);
        break;
```

```
       case 2:  //Λειτουργία PITCH
        pitch = (servoData % 10000);
        servo_2.writeMicroseconds((servoData % 10000) -
channelError);
        break;
       case 3:  //Λειτουργία THROTTLE
        throttle = (servoData % 10000);
        servo3_new = ((servoData % 10000) - 5);
        if (!(servo3_new == servo3_old)) {
         servo_3.writeMicroseconds(servo3_new);
         servo3_old = servo3_new;
        }
        break;
       case 4:  //Λειτουργία YAW
        yaw = (servoData % 10000);
        servo_4.writeMicroseconds((servoData % 10000) -
channelError);
        break;
       case 5:  //Λειτουργία ARM
        servo_5.writeMicroseconds(servoData % 10000);
        arm = (servoData % 10000);
        break;
      }
      servoData = 0;
     }
    }
   }
  }
  Serial.println("Client Disconnected");
 }
}
```

Παράρτημα Β (Κλάση Launcher του PC Drone Controller):

```
//Κλάση Launcher στην οποία υπάρχει υπάρχει η main συνάρτηση του
//προγράμματος
public class Launcher{
 //Η main συνάρτηση που αρχικοποιεί μια μεταβλητή UI με πεδία, Τίτλος,
 //πλάτος και μήκος
 public static void main(String[] args)
 { UI UI = new UI("DroneController", 700, 500);
   UI.start();
 }}
```

Παράρτημα Γ (Κλάση UI του PC Drone Controller):

```
//Κλάση UI που ενεργοποιεί τα γραφικά στοιχεία αλλά και τις λειτουργίες στο
//πίσω μέρος της εφαρμογής
public class UI{
 public int width,height;
 public String title;
 private Display display;
 private MainController mainController;
 //Constructor υπεύθυνος για την αρχικοποίηση αναγκαίων μεταβλητών
 public UI(String title, int width, int height) {
   this.width = width;
   this.height = height;
   this.title = title;  }

 //Συνάρτηση init που αρχικοποιεί την μεταβλητές display και
 //mainController
 private void init() {
   display = new Display(title, width, height);
   mainController = new MainController(display);
 }

 //Συνάρτηση start που δίνει την εντολή αρχικοποίησης των απαραίτητων
 //μεταβλητών και ξεκινάει τον Controller της εφαρμογής
 public void start()
 {
   init();
   mainController.startMainLoop();
  }
}
```

Παράρτημα Δ (Κλάση Display του PC Drone Controller):

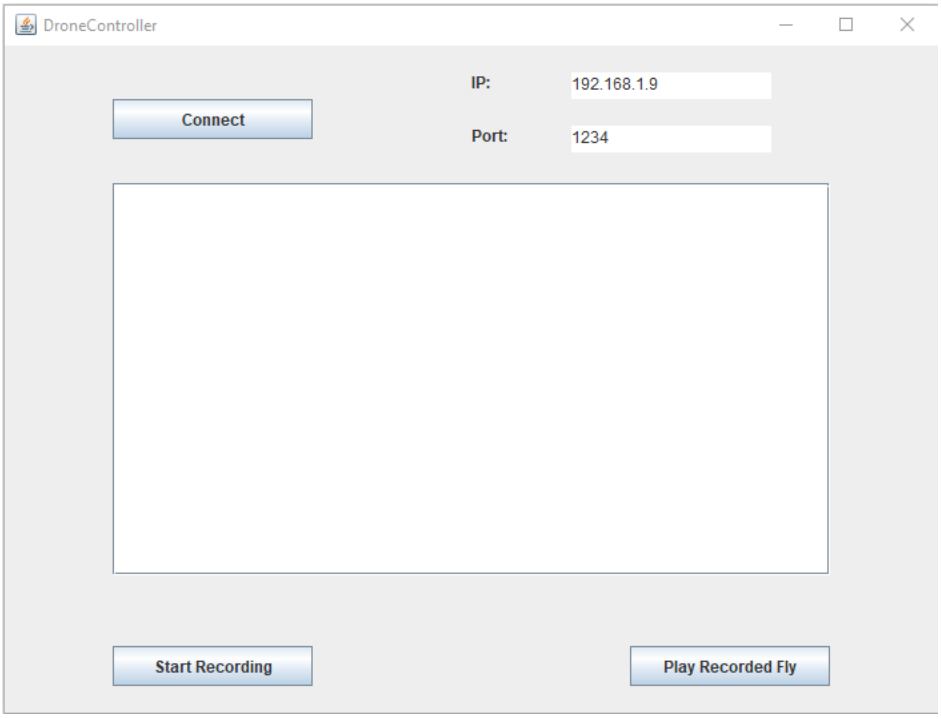
```
//Κλάση Display που χειρίζεται τα γραφικά στοιχεία της εφαρμογής
public class Display extends JFrame{
 private Canvas canvas;
 private JScrollPane scrollPane;
 private JTextArea textArea, iPTextArea,portTextArea;
 private JButton recordingBtn,playRecordedFlyBtn, connectBtn;
 private JLabel portLabel,IPLabel;
 private  int width,height;
 private TextAreaKeyListener textAreaKeyListener;
 private ButtonManager
recordingBtnManager,playRecordedFlyBtnManager,connectionBtnManager;
 //Ο Constructor της Display που αρχικοποιεί τον τίτλο, το μήκος και το
 //πλάτος και και αρχικοποιεί όλα τα υπόλοιπα γραφικά στοιχεία
 public Display(String title, int width, int height) {
   super(title);
   this.width = width;
   this.height = height;
   createDisplay();}
 //Συνάρτηση η οποία θέτει το μέγεθος, την τοποθεσία και άλλα στοιχεία
 //των γραφικών της εφαρμογής
 private void createDisplay(){
   //==============JFrame================
   setSize(width,height);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setResizable(true);
   setLocationRelativeTo(null); //window appears in the center of the
screen
   setVisible(true);
   //==============JTextArea================
   int mainTextAreaWidth = (int) (width/1.3);
   int mainTextAreaHeight = (int) (height/1.7);
   textArea = new JTextArea();
```

```
   textArea.setBounds(width/2-mainTextAreaWidth/2,height/2-
mainTextAreaHeight/2,mainTextAreaWidth,mainTextAreaHeight); //set textArea
in center
   textArea.setLineWrap(true);
   textArea.setVisible(true);
   textAreaKeyListener = new TextAreaKeyListener(textArea);
   textArea.addKeyListener(textAreaKeyListener);
   add(textArea);
   //=============IPTextArea==========================
   iPTextArea = new JTextArea("192.168.1.9");
   int secondaryTextAreaWidth = 150;
   int secondaryTextAreaHeight = 20;
   iPTextArea.setBounds((width/2)+
(secondaryTextAreaWidth
/
2),secondaryTextAreaHeight,secondaryTextAreaWidth,secondaryTextAreaHeight);
   //=========portTextArea==========================
   portTextArea = new JTextArea("1234");
   portTextArea.setBounds((width/2)+
(secondaryTextAreaWidth
/
2
),secondaryTextAreaHeight*3,secondaryTextAreaWidth,secondaryTextAreaHeight)
;
   //==============JScrollPane=================
   scrollPane = new JScrollPane(textArea);
   scrollPane.setBounds(width/2-mainTextAreaWidth/2,height/2-
mainTextAreaHeight/2,mainTextAreaWidth,mainTextAreaHeight);
   //================PortLabel======================
   int portLabelWidth = 75;
   int portLabelHeight = 15;
   portLabel = new JLabel("Port:");
   portLabel.setBounds((width/2)+(secondaryTextAreaWidth/2)-
portLabelWidth,secondaryTextAreaHeight*3,portLabelWidth,portLabelHeight);
   //================IPLabel======================
   int IPLabelWidth = 75;
   int IPLabelHeight = 15;
   IPLabel = new JLabel("IP:");
```

```
   IPLabel.setBounds((width/2)+(secondaryTextAreaWidth/2)-
IPLabelWidth,secondaryTextAreaHeight,IPLabelWidth,IPLabelHeight);
   //==============JRecordingBtn================
   int buttonWidth = 150;
   int buttonHeight = 30;
   recordingBtn = new JButton("Start Recording");
   recordingBtn.setBounds(width/2-mainTextAreaWidth/2, height-buttonHeight-
20, buttonWidth, buttonHeight);
   recordingBtnManager = new RecordingBtnActionListener(recordingBtn);
   recordingBtn.addActionListener(recordingBtnManager);
   //==============JPlayRecordedFlyBtn================
   playRecordedFlyBtn = new JButton("Play Recorded Fly");
   playRecordedFlyBtn.setBounds((width/2-
mainTextAreaWidth/2)+mainTextAreaWidth-buttonWidth, height-buttonHeight-20,
buttonWidth, buttonHeight);
   playRecordedFlyBtnManager = new
StartRecordedFlyBtnActionListener(playRecordedFlyBtn);
   playRecordedFlyBtn.addActionListener(playRecordedFlyBtnManager);
   //==============JPlayRecordedFlyBtn================
   connectBtn = new JButton("Connect");
   connectBtn.setBounds(width/2-mainTextAreaWidth/2, buttonHeight+10,
buttonWidth, buttonHeight);
   connectionBtnManager = new
ConnectionBtnActionListener(connectBtn,iPTextArea,portTextArea);
   connectBtn.addActionListener(connectionBtnManager);
   //==============Canvas================
   canvas = new Canvas();
   canvas.setPreferredSize(new Dimension(width,height));
   canvas.setMaximumSize(new Dimension(width,height));
   canvas.setMinimumSize(new Dimension(width,height));
   canvas.setFocusable(false);
   add(iPTextArea);
   add(IPLabel);
   add(portTextArea);
   add(portLabel);
   add(connectBtn);
   add(playRecordedFlyBtn);
```

```
    add(recordingBtn);
    add(scrollPane);
    add(canvas);
    pack();
  }
 //Επιστρέφει τον listener του Text Area με σκοπό την περαιτέρω επεξεργασία
 //πλήκτρων που εισάγει ο χρήστης
  public TextAreaKeyListener getTextAreaKeyListener()
  {
    return textAreaKeyListener;
  }

  public ButtonManager getRecordingBtnManager()
  {
    return recordingBtnManager;
  }
}
```



Εικόνα 43: Αποτελέσμα κλάσης Display του PC Drone Controller

Παράρτημα Ε (Κλάση Connection του PC Drone Controller):

```
//Η κλάση Connection διαχειρίζεται
//το socket για την μεταφορά δεδομένων
public class Connection{
 private Socket socket ;
 //Συνάρτηση που εκτελεί την σύνδεση του
 //PC Drone Controller με τον μικροελεγκτή
 public Connection() {
   //Μεταβλητή που επιστρέφει τα απαραίτητα στοιχεία για
   //την επιτυχής σύνδεση του socket με τον μικροελεγκτή
   Database database = new Database();
   try {
     socket = new
Socket(database.getIP(),Integer.parseInt(database.getPort()));
   } catch (IOException e)
   {
     System.out.println("socket can not open for some reason");
   }
```

```
 }
 //Συνάρτηση που επιστρέφει την μεταβλητή Socket
 public Socket getSocket()
 {
   return socket;
 }
}
```

Παράρτημα Ζ (Κλάση SendToDrone του PC Drone Controller):

```
//Κλάση υπεύθυνη για την αποστολή δεδομένων στον μικροελεγκτή
public class SendToDroneFrom {
  private DataOutputStream dataOutputStream;
  private Socket socket;

  //constructor που αρχικοποιεί την μεταβλητή socket
  public SendToDroneFrom(Socket socket)
  {this.socket = socket; }
  //Συνάρτηση αποστολής δεδομένων στον μικροελεγκτή
  public void directions(int data) {
 try {
        dataOutputStream = new DataOutputStream(socket.getOutputStream());
        dataOutputStream.writeInt(data);
     } catch (IOException e) {
        e.printStackTrace();
     }
}
}
```

Παράρτημα Η (Κλάση ProcessKeyInput του PC Drone Controller):

```
//Κλάση που μεταφράζει την είσοδο του χρήστη από πλήκτρο σε δεδομένα που
//επεξεργάζεται ο μικροελεγκτής
public class ProcessKeyInput{
 //Το βήμα με το οποίο θα μεταβάλλεται η κάθε τιμή
 private static final int MOTOR_STEP = 10;
 private static final int PITCH_FORWARD_KEY = KeyEvent.VK_W;
 private static final int PITCH_BACKWARD_KEY = KeyEvent.VK_S;
 private static final int ROLL_LEFT_KEY = KeyEvent.VK_A;
 private static final int ROLL_RIGHT_KEY = KeyEvent.VK_D;
 private static final int YAW_LEFT_KEY = KeyEvent.VK_Q;
 private static final int YAW_RIGHT_KEY = KeyEvent.VK_E;
 private static final int THROTTLE_UP_KEY = KeyEvent.VK_P;
 private static final int THROTTLE_DOWN_KEY = KeyEvent.VK_L;
 private static final int ARM_KEY = KeyEvent.VK_SPACE;
 private static final int DISARM_KEY = KeyEvent.VK_CONTROL;

 //Αρχικές τιμές για κάθε μια λειτουργία
 private int roll = 11500;
 private int pitch = 21500;
 private int yaw = 41500;
 private int throttle = 31000;
 private int arm = 51000;
 //Συνάρτηση που δέχεται τα πλήκτρα σαν είσοδο, τα επεξεργάζεται ανάλογα
//και στέλνει τα ανάλογα δεδομένα σαν έξοδο
 public Set<Integer> values(Set<Integer> keys){
   Set<Integer> returnValues = new HashSet<>();
   if (!keys.contains(KeyEvent.VK_SHIFT)) //If shift is pressed values stay
constant
   {
```

```
    if (keys.contains(PITCH_FORWARD_KEY)) //PITCH FORWARD
    {  if (pitch < 22000)
       pitch += MOTOR_STEP; }
    if (keys.contains(PITCH_BACKWARD_KEY)) //PITCH BACKWARD
    {  if (pitch > 21000)
       pitch -= MOTOR_STEP;}
    if (keys.contains(ROLL_LEFT_KEY)) //ROLL LEFT
    {   if (roll > 11000)
       roll -= MOTOR_STEP; }
    if (keys.contains(ROLL_RIGHT_KEY)) //ROLL RIGHT
    {  if (roll < 12000)
       roll += MOTOR_STEP; }
    if (keys.contains(YAW_LEFT_KEY)) //YAW LEFT
    {  if (yaw > 41000)
       yaw -= MOTOR_STEP;}
    if (keys.contains(YAW_RIGHT_KEY)) //YAW RIGHT
    { if (yaw < 42000)
      yaw += MOTOR_STEP; }
    if (keys.contains(THROTTLE_UP_KEY)) //THROTTLE UP
    { if (throttle < 32000)
      throttle += MOTOR_STEP; }
    if (keys.contains(THROTTLE_DOWN_KEY)) //THROTTLE DOWN
    { if (throttle > 31000)
      throttle -= MOTOR_STEP;}
    if (keys.contains(ARM_KEY)) //Space disarms the drone and sets the
minimum throttle
    { arm = 51000;
      throttle = 31000;}
    if (keys.contains(DISARM_KEY)) // Control arms the drone
    {     arm = 52000; }
  }
```

```
   boolean stopPitch = !keys.contains(PITCH_FORWARD_KEY) && !
keys.contains(PITCH_BACKWARD_KEY);
   boolean stopRoll = !keys.contains(ROLL_RIGHT_KEY) && !
keys.contains(ROLL_LEFT_KEY);
   boolean stopYaw = !keys.contains(YAW_RIGHT_KEY) && !
keys.contains(YAW_LEFT_KEY);
   if (stopRoll)
     roll = 11500;
   if (stopPitch)
     pitch = 21500;
   if (stopYaw)
     yaw = 41500;
   returnValues.add(roll);
   returnValues.add(pitch);
   returnValues.add(yaw);
   returnValues.add(throttle);
   returnValues.add(arm);
   return returnValues;
 }
}
```

Παράρτημα Θ (Κλάση TextAreaKeyListener του PC Drone Controller):

```
//Κλάση υπεύθυνη για την λειτουργικότητα του TextArea
public class TextAreaKeyListener implements KeyListener{
 protected  JTextArea textArea;
 //Μεταβλητή που γνωρίζει αν ένα πλήκτρο έχει πατηθεί
 boolean pressedOnesIs = false;
 //HashSet η είσοδος που πληκτρολογεί ο χρήστης
 private Set<Integer> pressed = new HashSet<>();
 //Μεταβλητή μέτρησης χρόνου
 private long startTime = 0;
 //Μεταβλητή καταγραφής του αριθμού των γραμμάτων που
 //έχει εισάγει ο χρήστης
 private int lettersInTextArea = 0;
 //Σταθερά για το κουμπί backspace
 private final int BACKSPACE_KEY = 8;
 private Database database;
//Constructor υπεύθυνος για την αρχικοποίηση αναγκαίων μεταβλητών
 public TextAreaKeyListener(JTextArea textArea)
 { this.textArea = textArea;
   database = new Database(); }
//Οταν πατιέται ένα πλήκτρο αποθηκεύεται στο hashset pressed
//και αρχίζει η μέτρηση του χρόνου
 @Override
 public void keyPressed(KeyEvent e)
 {
   if (e.getKeyCode() == BACKSPACE_KEY)
     lettersInTextArea--;
   else {
     pressed.add(e.getKeyCode());
     if (!pressedOnesIs)
```

```
    { startTime = System.nanoTime();
      pressedOnesIs = true;
    }   } }
 //Όταν το πλήκτρο σταματάει να πατιέται αφαιρείται από το hashset
 //pressed, καταγράφεται ο τελικός χρόνος και παρουσιάζεται στον χρήστη
 //το πλήκτρο΄και την διάρκεια που πατήθηκε
 @Override
 public void keyReleased(KeyEvent e) {
  pressed.remove(e.getKeyCode());
  if (e.getKeyCode() != BACKSPACE_KEY) {
    pressedOnesIs = false;

    textArea.replaceRange("", lettersInTextArea,
textArea.getText().length());
    long totalTime = (System.nanoTime() - startTime);
    textArea.append("Time the key \"" + e.getKeyChar() + "\" pressed :" +
((System.nanoTime() - startTime) / 1000000) + " ms");
    textArea.append("\n");

    lettersInTextArea += (21 + String.valueOf(totalTime).length() + 3);}
  if (lettersInTextArea < 1)
    lettersInTextArea = 1;
 }
 @Override
 public void keyTyped(KeyEvent e) {}
 //Συνάρτηση που επιστρέφει το Hashset με τα πλήκτρα που είναι πατημένα
 public Set<Integer> getPressed()
 { return pressed; }
}
```

Παράρτημα Ι (Κλάση MainLoop του PC Drone Controller):

```
//Η κλάση MainLoop είναι υπεύθυνη για τον χειρισμό αποστολής δεδομένων
//στον μικροελεγκτή με σταθερή συχνότητα κάθε φορά που ο χρήστης
//πληκτρολογεί στο TextArea
public class MainLoop implements Runnable{
 private static final double ONE_SECOND = 1000000000;
 //Μεταβλητή που ελέγχει το πότε θα σταματήσει η κεντρική συνάρτηση
//αποστολής δεδομένων
 private boolean exit = false;
 private ProcessKeyInput processKeyInput;
 private SendToDroneFrom sendToDrone;
 private Connection connection;
 private TextAreaKeyListener textAreaKeyListener;
 Database database;
 //Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
 public MainLoop(Connection connection, TextAreaKeyListener
textAreaKeyListener) {
   this.connection = connection;
   this.textAreaKeyListener = textAreaKeyListener; }
 // Συνάρτηση αρχικοποίησης αναγκαίων μεταβλητών
 private void init() {
   sendToDrone = new SendToDroneFrom(connection.getSocket());
   processKeyInput = new ProcessKeyInput();
   database = new Database(); }
//Συνάρτηση που χειρίζεται την λειτουργία αποστολής δεδομένων
 @Override
 public void run(){
   init();
   while (!exit) {
     //code here
     startLoop();
     exit = true;
   }
 }
```

```
//Συνάρτηση που ελέγχει την συχνότητα αποστολής δεδομένων
private void startLoop(){
   int tps = 20;  //ticks per second
   double timePerTick = ONE_SECOND / tps; // 1000000000 / 20
   double delta = 0;
   long now;
   long lastTime = System.nanoTime();
   long timer = 0;
   while (!exit){
     now = System.nanoTime();
     delta += (now - lastTime) / timePerTick ;
     timer += now - lastTime;
     lastTime = now;
     if (delta >= 1) {
       loop();
       delta--; }
     if (timer >= ONE_SECOND)
       timer = 0;
   } }
//Διαδικασία η οποία στέλνει τα απαραίτητα δεδομένα στον μικροελεγκτή
//και αποθηκεύει τις πληροφορίες σε περίπτωση που ο χρήστης έχει πατήσει
//την λειτουργία αποθήκευσης πτήσης
 private void loop() {
   for (Integer
directions:processKeyInput.values(textAreaKeyListener.getPressed()))
   {
     sendToDrone.directions(directions);

     if (database.getRecordingButtonIs())
       database.setRecordedFly(directions.toString());
   }
   if (database.getRecordingButtonIs())
     database.setRecordedFly(".");
 }
  public void startMainLoop(){exit = false;}
```

```
 public void stopMainLoop(){exit = true;}
}
```

Παράρτημα Κ (Κλάση PlayRecordedFlyLoop του PC Drone Controller):

```
//Κλάση υπεύθυνη για την αναπαραγωγή της καταγραφόμενης  πτήσης
public class PlayRecordedFlyLoop implements Runnable{
//Σταθερά για την δήλωση του ενός δευτερολέπτου
private static final double ONE_SECOND = 1000000000;
//Λογική μεταβλητή που ελέγχει το πότε σταματάει η διαδικασία
private boolean exit = false;
private  SendToDroneFrom sendToDrone;
private Connection connection;
private Database database;
//Ακέραια μεταβλητή που ελέγχει το πότε θα τελειώσει ο βρόχος
//επανάληψης για την καταγεγραμμένη πτήση
Int loopSize;
//Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
public PlayRecordedFlyLoop(Connection connection)
{  this.connection = connection; }
//Συνάρτηση αρχικοποίησης αναγκαίων μεταβλητών
private void init() {
  sendToDrone = new SendToDroneFrom(connection.getSocket());
  database = new Database();
  exit = false;
  loopSize=0; }

//Συνάρτηση που χειρίζεται την λειτουργία αποστολής δεδομένων
@Override
public void run() {
  init();
  System.out.println("PlayRecordedFlyLoop started");
  while (!exit) {
    startLoop();
    exit = true; }
```

```
    System.out.println("PlayRecordedFlyLoop stopped...."); }



//Διαδικασία η οποία στέλνει τα απαραίτητα δεδομένα στον μικροελεγκτή
//και αποθηκεύει τις πληροφορίες σε περίπτωση που ο χρήστης έχει πατήσει
//την λειτουργία αποθήκευσης πτήσης
private void loop() {
   for (Integer directions : database.getRecordedFly().get(loopSize))
     sendToDrone.directions(directions);
   if (loopSize <database.getRecordedFly().size()-1)
     loopSize++;
   else
     exit = true; }
 //Συνάρτηση που ελέγχει την συχνότητα αποστολής δεδομένων
 private void startLoop() {
   int tps = 20;  //ticks per second
   double timePerTick = ONE_SECOND / tps;
   double delta = 0;
   long now;
   long lastTime = System.nanoTime();
   long timer = 0;
   while (!exit){
     now = System.nanoTime();
     delta += (now - lastTime) / timePerTick ;
     timer += now - lastTime;
     lastTime = now;
     if (delta >= 1) {
       loop();
       delta--; }
     if (timer >= ONE_SECOND)
       timer = 0;
   }
 }
 public void startPlayRecordedFlyLoop(){ exit = false;}
 public void stopPlayRecordedFlyLoop(){ exit = true;}
```

```
}
```

Παράρτημα Λ (Κλάση MainController του PC Drone Controller):

```
//Κλάση υπεύθυνη για τον χειρισμό όλων των λειτουργιών του προγράμματος
public class MainController implements Runnable{
 //Λογική μεταβλητή υπεύθυνη για την έναρξη
 //του βασικού βρόχου αποστολής δεδομένων
 private boolean loopIs = false;
 private Thread thread;
 private Display display;
 private static TextAreaKeyListener textAreaKeyListener;
 private Database database;
 private static Connection connection;
 static MainLoop mainLoop;
 static PlayRecordedFlyLoop playRecordedFlyLoop;
 static Thread thread_MainLoop;
 static Thread thread_PlayRecordedFlyLoop;
 //Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
 public MainController(Display display)
 { this.display = display;}
//Συνάρτηση από την κλάση Runnable που εκτελείται όταν δημιουργείτε
//καινούργιο thread
 @Override
 public void run()
 { init(); }
//Συνάρτηση αρχικοποίησης αναγκαίων μεταβλητών
 private void init() {
   textAreaKeyListener = display.getTextAreaKeyListener();
   database = new Database();
   database.eraseRecordedFly();
   database.setRecordingButtonIs(false); }

//Συνάρτηση υπεύθυνη για την σωστή έναρξη της λειτουργίας
//αναπαραγωγής της καταγεγραμμένης πτήσης
```

```
 public static void startRecordedFly() {
   mainLoop.stopMainLoop();
   try {
     playRecordedFlyLoop = new PlayRecordedFlyLoop(connection);
     thread_PlayRecordedFlyLoop = new Thread(playRecordedFlyLoop);
     thread_PlayRecordedFlyLoop.start();
     thread_PlayRecordedFlyLoop.join();
   } catch (InterruptedException e) {
     e.printStackTrace(); }
   startMainLoopThread(); }
 //Συνάρτηση ενεργοποίησης του βασικού βρόχου αποστολής δεδομένων
 public static void startMainLoopThread(){
   mainLoop = new MainLoop(connection,textAreaKeyListener);
   thread_MainLoop = new Thread(mainLoop);
   thread_MainLoop.start();
   mainLoop.startMainLoop(); }
 //Δήλωση μεταβλητής connection που είναι αναγκαία και από
 //άλλες κλάσεις
 public static void setConnection(Connection con)
 {connection = con;}
 //Συνάρτηση που τρέχει την κλάση σε νέο thread
 public void startMainController(){
   if (loopIs)
     return;
   loopIs = true;
   thread = new Thread(this);
   thread.start();}
 //Συνάρτηση που διακόπτει την κλάση
 public void stopMainController(){
   if (!loopIs)
     return;
   loopIs = false;
   try { thread.join();}
   catch (InterruptedException e) {
     e.printStackTrace(); } }}
```

Παράρτημα Μ (Κλάση ButtonManager του PC Drone Controller):

```
//Abstract κλάση που καθορίζει τις λειτουργίες για τα κουμπιά της εφαρμογής
public abstract class ButtonManager implements ActionListener{
 //Μεταβλητή για το εκάστοτε κουμπί
 protected JButton button;
 //Αρχικοποίηση της μεταβλητής button από τον constructor
 public ButtonManager(JButton button) {  this.button = button;}
 //Μέθοδος που εκτελείται κατά το πάτημα του κουμπιού
 @Override
 public void actionPerformed(ActionEvent e)
 {}
 }
```

Παράρτημα Ν (Κλάση ConnectionBtnActionListener του PC Drone Controller):

```
//Κλάση που διαχειρίζεται το Connection button
public class ConnectionBtnActionListener extends ButtonManager{
 JTextArea ipTextArea,portTextArea;
 //Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
 public ConnectionBtnActionListener(JButton button,JTextArea ipTextArea,
JTextArea portTextArea) {
   super(button);
   this.ipTextArea = ipTextArea;
   this.portTextArea = portTextArea; }

 //Συνάρτηση που παίρνει την ip και το port πεδίο και δίνει την εντολή να
 //ξεκινήσει η σύνδεση με τον μικροελεγκτή
 @Override
 public void actionPerformed(ActionEvent e) {
   Database database = new Database();
   database.setIP(ipTextArea.getText());
   database.setPort(portTextArea.getText());
   Connection connection = new Connection();
   MainController.setConnection(connection);
   MainController.startMainLoopThread();
 }
}
```

Παράρτημα Ξ (Κλάση RecordingBtnActionListener του PC Drone Controller):

```
//Κλάση που διαχειρίζεται το Recording button
public class RecordingBtnActionListener extends ButtonManager{
 Database database;
 boolean pressedOneIs = false;
 //Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
 public RecordingBtnActionListener(JButton button) {
   super(button);
   database = new Database(); }
 //Συνάρτηση που δίνει την εντολή να ξεκινήσει ή να σταματήσει η
 //καταγραφή της πτήσης
 @Override
 public void actionPerformed(ActionEvent e) {
   if (e.getSource() == button){
     if (pressedOneIs){
       pressedOneIs = false;
       // code for second press
       database.setRecordingButtonIs(false);
       button.setText("Start Recording");
     }else{
       pressedOneIs = true;
       //code for first press
       database.eraseRecordedFly();
       database.setRecordingButtonIs(true);
       button.setText("Stop Recording");
     }
   }
 }
}
```

Παράρτημα Ο (Κλάση StartRecordedBtnActionListener του PC Drone Controller):

```
//Κλάση που διαχειρίζεται το Start Recorded Fly button
public class StartRecordedFlyBtnActionListener extends ButtonManager{
 //Constructor για την αρχικοποίηση αναγκαίων μεταβλητών
 public StartRecordedFlyBtnActionListener(JButton button)
 {  super(button);}
 //Συνάρτηση που δίνει την εντολή να ξεκινήσει η καταγεγραμμένη πτήση
 @Override
 public void actionPerformed(ActionEvent e) {
   MainController.startRecordedFly();
 }
}
```

Παράρτημα Π (Κλάση Database του PC Drone Controller):

```
//Κλάση υπεύθυνη για τον χειρισμό της βάση δεδομένων
public class Database{
 //Μεταβλητές που καθορίζουν το που θα αποθηκευτούν τα διάφορα δεδομένα
 final String path_OfRecordedFly = "src/database/recordedFly.txt";
 final String path_OfBooleanForRecordingButton =
"src/database/recordingButtonIs.txt";
 final String path_OfIPStored = "src/database/ip.txt";
 final String path_OfPortStored = "src/database/port.txt";
 //Μεταβλητές που δείχνουν στο σημείο που βρίσκονται τα δεδομένα
 File file_recordedFly = new File(path_OfRecordedFly);
 File file_recordingButtonIs = new File(path_OfBooleanForRecordingButton);
 File file_ip = new File(path_OfIPStored);
 File file_port = new File(path_OfPortStored);
 //Constructor υπεύθυνος για την αρχικοποίηση της βάσης
 public Database() {
  if (!file_recordedFly.exists()) {
    try {
      file_recordedFly.createNewFile();
    } catch (IOException e) {
      new File("src/database").mkdirs();
    }
  }
  if (!file_recordingButtonIs.exists()) {
    try {
      file_recordingButtonIs.createNewFile();
    } catch (IOException e) {
      new File("src/database").mkdirs(); }
  }
```

```
    if (!file_ip.exists()) {
      try {
        file_ip.createNewFile();
      } catch (IOException e) {
        new File("src/database").mkdirs(); }
    }
    if (!file_port.exists()) {
      try {
        file_port.createNewFile();
      } catch (IOException e) {
        new File("src/database").mkdirs();
      }}}
//Μέθοδος αποθήκευσης την καταγραφόμενη πτήση
public void setRecordedFly(String msg) {
    try {
      FileWriter fw = new FileWriter(path_OfRecordedFly, true);
      BufferedWriter writer = new BufferedWriter(fw);
      writer.append(msg);
      writer.newLine();
      writer.close();
    } catch (IOException e) {
      e.printStackTrace();
    } }
  //Μέθοδος ανάκτησης της καταγεγραμμένης πτήσης
  public List<ArrayList<Integer>> getRecordedFly() {
    List<ArrayList<Integer>> list = new ArrayList<ArrayList<Integer>>();
    Scanner scanner = null;
    try {
      scanner = new Scanner(file_recordedFly);
      int i = 0;
      list.add(new ArrayList<>());
```

```
    while (scanner.hasNextLine()) {
      String line = scanner.nextLine();
      if (line.equals(".")) {
        list.add(new ArrayList<>());
        i++;
      } else {
        list.get(i).add(Integer.parseInt(line)); }
    }
  } catch (FileNotFoundException e) {
    System.out.println(scanner.nextLine());
    e.printStackTrace();}
  return list;
}
//Μέθοδος διαγραφής της καταγεγραμμένης πτήσης
public void eraseRecordedFly() {
  try {
    new FileWriter(path_OfRecordedFly, false).close();
  } catch (IOException ioException) {
    ioException.printStackTrace();
  }
}
//Μέθοδος αποθήκευσης της IP
public void setIP(String ip) {
  try {
    FileWriter fw = new FileWriter(path_OfIPStored);
    BufferedWriter writer = new BufferedWriter(fw);
    writer.append(ip);
    writer.newLine();
    writer.close();
  } catch (IOException e) {
```

```
    e.printStackTrace();
  }
 }
//Μέθοδος ανάκτησης της IP
public String getIP() {
  try {
    Scanner scanner = new Scanner(file_ip);
    return scanner.nextLine();
  } catch (FileNotFoundException e)
  { return null; }
 }
//Μέθοδος αποθήκευσης της πόρτας διασύνδεσης
public void setPort(String port){
  try {
    FileWriter fw = new FileWriter(path_OfPortStored);
    BufferedWriter writer = new BufferedWriter(fw);
    writer.append(port);
    writer.newLine();
    writer.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
 }
//Μέθοδος ανάκτησης της πόρτας διασύνδεσης
public String getPort() {
  try {
    Scanner scanner = new Scanner(file_port);
    return scanner.nextLine();
  } catch (FileNotFoundException e)
  { return null; }
```

```
 }
//Μέθοδος αποθήκευσης της ένδειξης του κουμπιού καταγραφής πτήσης
 public void setRecordingButtonIs(boolean recordingButtonIs) {
   String buttonIs = recordingButtonIs ? "True" : "False";
   try {
     FileWriter fw = new FileWriter(path_OfBooleanForRecordingButton);
     BufferedWriter writer = new BufferedWriter(fw);
     writer.append(buttonIs);
     writer.newLine();
     writer.close();
   } catch (IOException e) {
     e.printStackTrace();
   }
 }
//Μέθοδος ανάκτησης της ένδειξης του κουμπιού καταγραφής πτήσης
 public boolean getRecordingButtonIs() {
   boolean recordingButtonIs;
   try {
     Scanner scanner = new Scanner(file_recordingButtonIs);
     if (scanner.nextLine().equals("True"))
       recordingButtonIs = true;
     else
       recordingButtonIs = false;
   } catch (FileNotFoundException e) {
     e.printStackTrace();
     recordingButtonIs = false;
   }
   return recordingButtonIs;
 }
}
```