ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη drone αποφυγής εμποδίων με χρήση κάμερας, αισθητήρων και τεχνητής νοημοσύνης. Ζωντανή παρακολούθηση θέσης, ταχύτητας και κατάστασης μπαταρίας μέσω διαδικτυακού πίνακα ελέγχου»

Του φοιτητή
Αργύρη Αητού
Αρ. Μητρώου: 2020002

Επιβλέπων
Γεώργιος Κοκκώνης
Αναπληρωτής Καθηγητής

30 Μαΐου 2026

Τίτλος Δ.Ε.: Ανάπτυξη drone αποφυγής εμποδίων με χρήση κάμερας, αισθητήρων και τεχνητής νοημοσύνης. Ζωντανή παρακολούθηση θέσης, ταχύτητας και κατάστασης μπαταρίας μέσω διαδικτυακού πίνακα ελέγχου.

Κωδικός Δ.Ε.: 25278

Ονοματεπώνυμο φοιτητή: Αργύριος Αητός

Ονοματεπώνυμο εισηγητή: Γεώργιος Κοκκώνης

Ημερομηνία ανάληψης Δ.Ε.: 01-07-2025

Ημερομηνία περάτωσης Δ.Ε.: 30-05-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Σε αυτούς που πίστεψαν σε μένα»

Πρόλογος

Η επιλογή της συγκεκριμένης διπλωματικής εργασίας προέκυψε από το ενδιαφέρον μου για τα drone και τις δυνατότητες που προσφέρουν οι σύγχρονες τεχνολογίες που ενσωματώνονται σε ένα μη επανδρωμένο αεροσκάφος. Η ιδέα της δημιουργίας ενός drone που να μπορεί να αντιλαμβάνεται το περιβάλλον του και να αποφεύγει εμπόδια αποτέλεσε για εμένα μια ενδιαφέρουσα πρόκληση καθώς συνδυάζει διαφορετικούς τομείς της τεχνολογίας όπως αισθητήρες, προγραμματισμό και επεξεργασία δεδομένων.

Μέσα από την ανάπτυξη της εργασίας είχα την ευκαιρία να γνωρίσω καλύτερα τον τρόπο λειτουργίας ενός drone, τη συνεργασία ηλεκτρονικών υποσυστημάτων και τη χρήση τεχνολογιών όπως το LiDAR, η επεξεργασία εικόνας και ενσωματωμένα συστήματα. Η διαδικασία σχεδιασμού, συναρμολόγησης και υλοποίησης του UAV με βοήθησε να αποκτήσω σημαντικές πρακτικές και τεχνικές γνώσεις γύρω από την ανάπτυξη ολοκληρωμένων συστημάτων πτήσης.

Περίληψη

Η συγκεκριμένη διπλωματική εργασία επικεντρώνεται στην ανάπτυξη ενός UAV με δυνατότητα αποφυγής εμποδίων και παρακολούθησης της κατάστασης του σε πραγματικό χρόνο. Μέσα από τεχνολογίες που χρησιμοποιούνται στα σύγχρονα drone σχεδιάστηκε και υλοποιήθηκε ένα ολοκληρωμένο σύστημα το οποίο συνδυάζει αισθητήρες, υπολογιστική όραση και ενσωματωμένα συστήματα.

Στο πλαίσιο της υλοποίησης πραγματοποιήθηκε η κατασκευή και συναρμολόγηση του ίδιου του drone καθώς και η διασύνδεση των απαραίτητων ηλεκτρονικών υποσυστημάτων. Για την αποφυγή εμποδίων χρησιμοποιήθηκε αισθητήρας μέτρησης αποστάσεων και αναπτύχθηκε κατάλληλος αλγόριθμος επεξεργασίας δεδομένων ώστε το UAV να μπορεί να αντιλαμβάνεται εμπόδια και να τα αποφεύγει. Παράλληλα, ενσωματώθηκε κάμερα και αλγόριθμος αναγνώρισης ανθρώπινης παρουσίας, επιτρέποντας στο drone να την εντοπίζει και να προσαρμόζεται ανάλογα κατά την πτήση. Επιπλέον, αναπτύχθηκε σύστημα τηλεμετρίας και ζωντανής μετάδοσης εικόνας δίνοντας την δυνατότητα για παρακολούθηση της κατάστασης του UAV και της εικόνας από την κάμερα κατά τη διάρκεια της πτήσης.

Το σύστημα κατάφερε να εκτελεί σε ελεγχόμενο περιβάλλον βασικές λειτουργίες αποφυγής εμποδίων. Τα αποτελέσματα δείχνουν ότι ο συνδυασμός των επιμέρους τεχνολογιών μπορεί να υποστηρίξει τη λειτουργία ενός UAV με πιο ασφαλή και ευέλικτη συμπεριφορά κατά την πτήση, προσφέροντας μια βάση για μελλοντικές βελτιώσεις και πιο εξελιγμένες εφαρμογές αυτόνομης πλοήγησης.

« Development of an Obstacle Avoidance Drone Using a Camera, Sensors and Artificial Intelligence. Real-Time Position, Speed, and Battery Status Monitoring via a Web Dashboard»

«Argyrios Aitos»

Abstract

This diploma thesis focuses on the development of a UAV featuring obstacle avoidance capabilities and real-time status monitoring. Leveraging technologies utilized in modern drones, an integrated system was designed and implemented, combining sensors, computer vision, and embedded systems.

Within the scope of the implementation, the physical construction and assembly of the drone itself were carried out along with the interfacing of the necessary electronic subsystems. For obstacle avoidance a distance measurement sensor was utilized, and a suitable data processing algorithm was developed to enable the UAV to perceive and avoid obstacles. Concurrently, a camera and a human presence recognition algorithm were integrated, allowing the drone to detect individuals and adapt accordingly during flight. Additionally, a telemetry and live video streaming system was developed providing the capability to monitor both the UAV's status and the camera feed throughout the flight.

The system successfully executed basic obstacle avoidance functions in a controlled environment. The results demonstrate that the combination of these individual technologies can support the operation of a UAV with safer and more agile in-flight behavior providing a solid foundation for future improvements and more advanced autonomous navigation applications.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για τη υποστήριξη, την κατανόηση και την ενθάρρυνση που μου προσέφεραν σε όλη τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω θερμά τον κ. Κοκκώνη για την καθοδήγηση και τη βοήθεια που μου παρείχε σε κάθε στάδιο της προσπάθειας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

LiDAR	Light Detection and Ranging
UAV	Unmanned Aerial Vehicle
CSI	Camera Serial Interface
ESC	Electronic Speed Controller
TX	Transmit
RX	Receive
GND	Ground
GPS	Global Positioning System
IMU	Inertial Measurement Unit
SLAM	Simultaneous Localization and Mapping
IP	Internet Protocol
JSON	JavaScript Object Notation
UDPIN	User Datagram Protocol Input
MAVLink	Micro Air Vehicle Link
PWM	Pulse Width Modulation
ROS	Robot Operating System
RTL	Return to Launch
APM	ArduPilot Mega
I2C	Inter Integrated Circuit
LiPo	Lithium Polymer
RRTS	Rapidly exploring Random Tree Star
RPM	Revolutions Per Minute
CH	Channel
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VCC	Voltage Common Collector
AFHDS	Automatic Frequency Hopping Digital System
CPU	Central Processing Unit
DC	Direct Current
GPIO	General Purpose Input Output
RC	Radio Control

RF	Radio Frequency
CW	Clockwise
CCW	Counterclockwise
Gbps	Gigabits per second
MP	Megapixel
PID	Proportional, Integral, Derivative
V	Volt

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Τα Μη επανδρωμένα εναέρια οχήματα (UAVs) αποτελούν μια από τις ταχύτερα αναπτυσσόμενες τεχνολογίες των τελευταίων δεκαετιών στους τομείς της ρομποτικής, των αυτοματισμών και των ενσωματωμένων συστημάτων. Η συνεχής πρόοδος στην υπολογιστική ισχύ, στους αισθητήρες και στις τεχνολογίες ασύρματης επικοινωνίας έχει συμβάλει σημαντικά στη διάδοση των UAVs, τα οποία πλέον χρησιμοποιούνται ευρέως τόσο στην έρευνα όσο και σε πλήθος βιομηχανικών εφαρμογών. Τα εναέρια αυτά συστήματα λειτουργούν χωρίς επιβαίνοντα χειριστή και μπορούν να ελέγχονται είτε με τηλεχειρισμό είτε μέσω αυτόνομων διαδικασιών πλοήγησης.

Η αρχική χρησιμότητα ήταν κυρίως σε στρατιωτικές εφαρμογές όπως βασικές μορφές επιτήρησης και αναγνώρισης περιοχών. Με την πάροδο του χρόνου και την εξέλιξη της τεχνολογίας η χρήση επεκτάθηκε και σε πολιτικές εφαρμογές. Σήμερα τα UAVs ή ευρύτερα γνωστά ως drones, χρησιμοποιούνται σε πολλούς τομείς όπως η εναέρια φωτογράφηση και βιντεοσκόπηση, η γεωργία ακριβείας, η επιτήρηση υποδομών, η έρευνα και διάσωση, καθώς και σε πλήθος άλλων εφαρμογών. Οι εφαρμογές αυτές αυξάνονται διαρκώς καθώς η τεχνολογία εξελίσσεται και νέες ανάγκες προκύπτουν σε βιομηχανικά και ερευνητικά περιβάλλοντα.

Η δομή ενός σύγχρονου UAV αποτελεί ένα σύνθετο τεχνολογικό σύστημα, όπου διάφορα ηλεκτρονικά μέρη συνεργάζονται. Στον πυρήνα του συστήματος βρίσκεται ο ελεγκτής πτήσης ο οποίος λειτουργεί ως ο εγκέφαλος του σκάφους λαμβάνοντας συνεχώς δεδομένα από τους αισθητήρες για να διατηρεί την ισορροπία και τη σταθερότητα της πτήσης. Η κίνηση του drone βασίζεται στον έλεγχο της ταχύτητας περιστροφής των κινητήρων του μέσω των οποίων το drone μπορεί να μεταβάλλει τη θέση και τον προσανατολισμό του στον χώρο. Οι βασικές κινήσεις πραγματοποιούνται γύρω από τρεις άξονες περιστροφής. Η κίνηση Roll αφορά την κίνηση προς τα αριστερά ή δεξιά. Η κίνηση Pitch σχετίζεται με την κίνηση προς τα εμπρός ή προς τα πίσω. Η κίνηση Yaw αντιστοιχεί στην περιστροφή γύρω από τον κατακόρυφο άξονα του UAV. Κάθε μία από αυτές τις κινήσεις επηρεάζει διαφορετικό άξονα του σκάφους επιτρέποντάς του να αλλάζει πορεία, να σταθεροποιείται και να εκτελεί τους απαραίτητους ελιγμούς κατά την πτήση. Για να πραγματοποιηθούν όμως αυτοί οι ελιγμοί το drone βρίσκεται σε συνεχή ασύρματη επικοινωνία με το χειριστή λαμβάνοντας σε πραγματικό χρόνο τις εντολές κατεύθυνσης μέσω του τηλεχειριστηρίου του.

Παρά τη σημαντική πρόοδο στον τομέα των UAVs η ασφαλής πλοήγηση σε περιβάλλοντα με εμπόδια εξακολουθεί να αποτελεί μια από τις μεγαλύτερες προκλήσεις για τα σύγχρονα συστήματα. Η αποφυγή εμποδίων αποτελεί κρίσιμο παράγοντα για την αξιόπιστη και ευρεία χρήση τους σε πραγματικές εφαρμογές όπως πτήση σε περιοχές με έντονη παρουσία ανθρώπων και κατασκευών. Η σύγκρουση ενός UAV με εμπόδιο μπορεί να οδηγήσει σε υλικές ζημιές, απώλεια εξοπλισμού ή ακόμα να βάλει σε κίνδυνο την ανθρώπινη ασφάλεια, γεγονός που καθιστά την ανάπτυξη αξιόπιστων μηχανισμών αποφυγής ιδιαίτερα σημαντική.

Οι κλασικές μέθοδοι αποφυγής εμποδίου σε UAV βασίζονται κυρίως στη χρήση αισθητήρων απόστασης όπως υπερηχητικών αισθητήρων και αισθητήρων απόστασης τεχνολογίας LiDAR (Light Detection and Ranging) οι οποίοι παρέχουν πληροφορίες σχετικά με την απόσταση του από αντικείμενα στο περιβάλλον. Αν και επιτρέπουν την έγκαιρη ανίχνευση εμποδίων δεν προσφέρουν πληροφορία για το είδος τους με αποτέλεσμα το UAV να αντιδρά με τον ίδιο τρόπο σε κάθε είδος εμπόδιου.

Τα τελευταία χρόνια η ανάπτυξη της τεχνητής νοημοσύνης και ειδικότερα της βαθιάς μάθησης (deep learning) έχει προσφέρει νέες δυνατότητες επιτρέποντας στο UAV να αναγνωρίζει συγκεκριμένες κατηγορίες μέσα στο οπτικό πεδίο της κάμερας όπως ανθρώπους ή άλλα αντικείμενα. Με αυτόν τον τρόπο το σύστημα δεν περιορίζεται μόνο στην απλή αποφυγή εμποδίων αλλά μπορεί να προσαρμόζει τη συμπεριφορά του ανάλογα με το είδος του εμποδίου ενισχύοντας σημαντικά την ασφάλεια της πτήσης.

Παρά την πρόοδο που έχει σημειωθεί τα τελευταία χρόνια, η πλήρης αυτονομία των drone εξακολουθεί να αποτελεί μια απαιτητική πρόκληση. Η αυτόνομη πλοήγηση απαιτεί αλγορίθμους που μπορούν να λαμβάνουν αξιόπιστες αποφάσεις σε μεταβαλλόμενα και απρόβλεπτα περιβάλλοντα. Αν και υπάρχουν συστήματα που υποστηρίζουν αυτόνομες αποστολές, όπως πτήση από σημείο σε σημείο με μηχανισμούς αποφυγής εμποδίων στην πράξη τα περισσότερα συστήματα λειτουργούν σε ημιαυτόνομο επίπεδο διατηρώντας τον άνθρωπο ως τον τελικό εγγυητή της ασφάλειας της πτήσης.

Στο πλαίσιο αυτό, η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη ενός πρακτικού συστήματος αποφυγής εμποδίων το οποίο αξιοποιεί δεδομένα από αισθητήρες απόστασης σε συνδυασμό με τεχνητή νοημοσύνη. Η υλοποίηση και οι δοκιμές πραγματοποιήθηκαν σε πραγματικές συνθήκες πτήσης, με στόχο το σύστημα να αναγνωρίζει με αξιοπιστία τα εμπόδια και να προσαρμόζει ανάλογα τη συμπεριφορά του.

1.2 Στόχος Διπλωματικής

Ο βασικός στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και η αξιολόγηση ενός ημιαυτόνομου συστήματος αποφυγής εμποδίων για μη επανδρωμένο εναέριο όχημα. Το σύστημα αυτό επιδιώκει να συνδυάσει δεδομένα από αισθητήρες απόστασης και κάμερα με τεχνικές τεχνητής νοημοσύνης, ώστε το drone να μπορεί να αντιλαμβάνεται το περιβάλλον του και να αντιδρά με ασφαλή και προβλέψιμο τρόπο. Παράλληλα, ο στόχος περιλαμβάνει τη διατήρηση του ανθρώπινου χειριστή στον κεντρικό έλεγχο της πτήσης, ώστε η αυτονομία να λειτουργεί υποστηρικτικά.

Συγκεκριμένα για την επίτευξη του στόχου ενός συστήματος αποφυγής εμποδίων χρησιμοποιούνται δεδομένα από έναν αισθητήρα LiDAR και οπτική πληροφορία από κάμερα, επιτρέποντας στο drone όχι μόνο να ανιχνεύει την ύπαρξη εμποδίων, αλλά και να διαφοροποιεί τη συμπεριφορά του ανάλογα με το είδος τους.

Παράλληλα η διπλωματική εργασία στοχεύει στην υλοποίηση μιας ολοκληρωμένης αρχιτεκτονικής συστήματος η οποία συνδυάζει έναν ελεγκτή πτήσης για τη διαχείριση της συμπεριφοράς κατά την πτήση και μια υπολογιστική μονάδα για την εκτέλεση αλγορίθμων τεχνητής νοημοσύνης και λήψης αποφάσεων. Ακόμα η επικοινωνία μεταξύ των συστημάτων σχεδιάζεται με τρόπο ώστε να εξασφαλίζεται αξιόπιστη και έγκαιρη ανταλλαγή δεδομένων κατά την διάρκεια της πτήσης. Επιπλέον στόχος της εργασίας αποτελεί η ανάπτυξη ενός διαδικτυακού πίνακα ελέγχου (dashboard) μέσω του οποίου παρέχεται ζωντανή παρακολούθηση κρίσιμων παραμέτρων του drone.

Τελικός στόχος της εργασίας είναι να αποδείξει ότι ο συνδυασμός αισθητήρων απόστασης και τεχνητής νοημοσύνης μπορεί να αποτελέσει μια αποτελεσματική και πρακτική λύση για την αποφυγή εμποδίων σε UAV. Η προσέγγιση αυτή δεν στοχεύει μόνο στη βελτίωση της ασφάλειας αλλά και στη δημιουργία ενός συστήματος που μπορεί να επεκταθεί και να προσαρμοστεί σε μελλοντικές εφαρμογές.

1.3 Δομή Διπλωματικής

Η παρούσα διπλωματική έχει οργανωθεί έτσι ώστε να παρουσιάζει με ξεκάθαρο και ομαλό τρόπο όλα τα στάδια της ανάπτυξης του συστήματος αποφυγής εμποδίων. Το πρώτο κεφάλαιο λειτουργεί ως εισαγωγή στο αντικείμενο και εξηγεί το πλαίσιο της εργασίας , ενώ τα επόμενα κεφάλαια αναπτύσσουν σταδιακά το θεωρητικό υπόβαθρο , το σχεδιασμό την υλοποίηση και τα αποτελέσματα.

1. Στο κεφάλαιο 2 περιλαμβάνει τη βιβλιογραφική ανασκόπηση. Παρουσιάζονται τα UAV, οι βασικές κατηγορίες μεθόδων αποφυγής εμποδίων και σχετικές προσεγγίσεις από την ερευνά. Στο τέλος γίνεται μια σύντομη σύνοψη των σημαντικότερων συμπερασμάτων.
2. Στο κεφάλαιο 3 ασχολείται με το σχεδιασμό και την αρχιτεκτονική συστήματος. Περιγράφονται τα υλικά που χρησιμοποιήθηκαν, οι συνδέσεις μεταξύ των υποσυστημάτων και ο τρόπος επικοινωνίας τους καθώς και το συνολικό κόστος κατασκευής.
3. Στο κεφάλαιο 4 παρουσιάζει τις ρυθμίσεις πτήσης και τη λογισμική διαμόρφωση. Επιπλέον αναφέρονται οι διαδικασίες βαθμονόμησης (Calibration), οι ρυθμίσεις του ελεγκτή πτήσης (flight controller) καθώς και οι διαθέσιμες λειτουργιές του Ardupilot.
4. Στο κεφάλαιο 5 περιγράφει τον αλγόριθμο αποφυγής εμποδίων. Αναλύεται ο ρόλος του LiDAR, η λειτουργιά του YOLO και ο τρόπος με τον οποίο συνδυάζονται για να ληφθούν οι αποφάσεις πτήσης.
5. Στο κεφάλαιο 6 παρουσιάζει τον σχεδιασμό του Dashboard για τη ζωντανή παρακολούθηση της κατάστασης του UAV, καθώς και την υλοποίηση του συστήματος για τη μετάδοση εικόνας από την κάμερα.
6. Στο κεφάλαιο 7 παρουσιάζονται τα πειραματικά αποτελέσματα. Επίσης περιλαμβάνει τις δοκιμές με χρήση LiDAR, τις δοκιμές με LiDAR και κάμερα, καθώς και μια συγκριτική αξιολόγηση της συνολικής απόδοσης του συστήματος.
7. Στο κεφάλαιο 8 συνοψίζει τα βασικά συμπεράσματα της εργασίας και προτείνει πιθανές μελλοντικές επεκτάσεις. Μετά τα κύρια κεφάλαια ακολουθεί και η βιβλιογραφία.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Μη Επανδρωμένα Εναέρια Οχήματα και Αποφυγή Εμποδίων

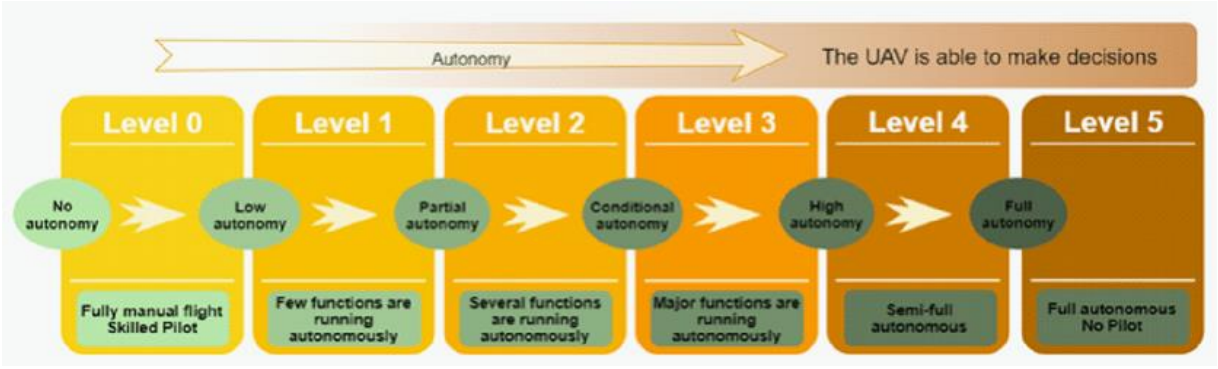
Τα Μη επανδρωμένα εναέρια οχήματα αποτελούν ολοκληρωμένα συστήματα που περιλαμβάνουν το εναέριο όχημα, τον ελεγκτή πτήσης, τους αισθητήρες, τις υπολογιστικές μονάδες και τα συστήματα επικοινωνίας, φτιάχνοντας αυτό που συχνά αναφέρεται ως Μη επανδρωμένο σύστημα αεροσκαφών (UAS) [1]. Η αρχιτεκτονική αυτή επιτρέπει την εκτέλεση εναέριων αποστολών με ποικίλους βαθμούς ανθρώπινης παρέμβασης, οι οποίοι κυμαίνονται από τον άμεσο τηλεχειρισμό έως την πλήρη αυτονομία, ανάλογα με την εφαρμογή και το επιχειρησιακό περιβάλλον.

Ωστόσο η λειτουργία των UAVs σε σύνθετα και δυναμικά περιβάλλοντα όπως τα αστικά έχει ως αποτέλεσμα την παρουσία πλήθους εμποδίων τα οποία αυξάνουν σημαντικά την πολυπλοκότητα της πλοήγησης. Τα εμπόδια συχνά διακρίνονται σε στατικά όπως κτίρια, δέντρα και υποδομές και σε δυναμικά όπως άνθρωποι, οχήματα [2]. Η έγκαιρη ανίχνευση και αξιόπιστη αποφυγή τους αποτελεί βασική προϋπόθεση για την ασφαλή λειτουργία των UAVs ιδιαίτερα σε αποστολές πέραν της οπτικής επαφής.

Η ικανότητα του συστήματος να αντιλαμβάνεται και να αποφεύγει κινδύνους χωρίς ανθρώπινη παρέμβαση, αποτελεί βασικό κριτήριο για τον καθορισμό του βαθμού αυτονομίας του. Σύμφωνα με το επίπεδο αυτονομίας (LoA) η αυτονομία διαβαθμίζεται σε έξι επίπεδα από πλήρως χειροκίνητη λειτουργία έως πλήρη αυτοματοποίηση [3].

1. LoA 0-2 (Χειροκίνητη Πτήση): Ο χειριστής φέρει την πλήρη ευθύνη ανίχνευσης και αποφυγής εμποδίων με το σύστημα να παρέχει βασική σταθεροποίηση ή ειδοποιήσεις απόστασης.
2. LoA 3 (Υπό Συνθήκη Αυτοματισμός): Σε αυτό το επίπεδο το UAV μπορεί να εκτελεί αυτόνομα ελιγμούς αποφυγής εμποδίων αξιοποιώντας αισθητήρες και αλγορίθμους πλοήγησης. Ωστόσο απαιτείται συνεχής επίβλεψη από τον χειριστή, ο οποίος πρέπει να είναι έτοιμος να παρεμβεί σε περίπτωση αποτυχίας του συστήματος.
3. LoA 4-5 (Υψηλή και πλήρης Αυτονομία): Το σύστημα διαθέτει προηγμένη αντίληψη αναγνωρίζει το είδος του εμποδίου και επανασχεδιάζει την τροχιά του σε πραγματικό χρόνο. Σε αυτά τα επίπεδα δεν απαιτείται ούτε επίβλεψη ούτε παρέμβαση από χειριστή.

Όπως απεικονίζεται και στο σχήμα 2.1, η αυτονομία ενός UAV εξελίσσεται από πλήρως χειροκίνητη λειτουργία (Level 0) έως πλήρη αυτοματοποίηση (Level 5), ενώ η ικανότητα του συστήματος να λαμβάνει αποφάσεις να αυξάνεται προοδευτικά.



Σχήμα 2.1: Τα έξι επίπεδα αυτονομίας των UAVs [4]

Συνοψίζοντας, η ασφαλής λειτουργία των μη επανδρωμένων εναέριων οχημάτων σε σύνθετα και δυναμικά περιβάλλοντα εξαρτάται άμεσα από την ικανότητά τους να αντιλαμβάνονται έγκαιρα το περιβάλλον και να αντιδρούν αποτελεσματικά σε πιθανούς κινδύνους. Καθώς ο βαθμός αυτονομίας αυξάνεται, οι απαιτήσεις για αξιόπιστη ανίχνευση και αποφυγή εμποδίων καθίστανται κρίσιμες, ιδίως σε εφαρμογές όπου η ανθρώπινη παρέμβαση είναι περιορισμένη. Για τον λόγο αυτό, η επιλογή κατάλληλων αισθητήρων και μεθόδων επεξεργασίας δεδομένων αποτελεί βασικό παράγοντα για την επίτευξη ασφαλούς και αποδοτικής πλοήγησης.

2.2 Αισθητήρες για Αποφυγή Εμποδίων

Η αποφυγή εμποδίων αποτελεί ένα από τα πιο κρίσιμα στοιχεία της αυτόνομης πλοήγησης ενός UAV καθώς καθορίζει την ικανότητα του να κινείται με ασφάλεια σε πραγματικές συνθήκες. Για να μπορέσει ένα τέτοιο σύστημα να ανταποκριθεί σε περιβάλλοντα που μπορεί να είναι άγνωστα ή δυναμικά, χρειάζεται να αντιλαμβάνεται τον χώρο γύρω του και να προσαρμόζει την πορεία του σε πραγματικό χρόνο. Αυτό απαιτεί μηχανισμούς που μπορούν να εντοπίζουν έγκαιρα εμπόδια και να υποστηρίζουν άμεσες διορθωτικές κινήσεις χωρίς συνεχή ανθρώπινη παρέμβαση.

Οι αισθητήρες αποτελούν την βάση κάθε συστήματος αποφυγής εμποδίων, καθώς παρέχουν τις απαραίτητες πληροφορίες στους αλγορίθμους πλοήγησης για την κατανόηση του περιβάλλοντος. Η επιλογή ενός κατάλληλου αισθητήρα καθορίζει τον τρόπο με τον οποίο γίνεται αντιληπτός ο χώρος είτε μέσω μετρήσεων απόστασης είτε μέσω οπτικών δεδομένων.

Στη συνέχεια, παρουσιάζονται οι δύο βασικές κατηγορίες αισθητήρων που χρησιμοποιούνται για την αποφυγή εμποδίων οι αισθητήρες μέτρησης απόστασης και οι αισθητήρες όρασης.

2.2.1 Αισθητήρες Μέτρησης Απόστασης

Οι αισθητήρες μέτρησης απόστασης αποτελούν μία από τις πιο διαδεδομένες προσεγγίσεις στην αυτόνομη πλοήγηση των UAVs. Η λειτουργία τους βασίζεται στη συνεχή μέτρηση της απόστασης ανάμεσα στο σκάφος και τα αντικείμενα του περιβάλλοντος, επιτρέποντας τον έγκαιρο εντοπισμό εμποδίων και την εκτέλεση των απαραίτητων ελιγμών αποφυγής. Οι αισθητήρες αυτοί θεωρούνται αξιόπιστοι σε πραγματικό χρόνο και απαιτούν σχετικά χαμηλή υπολογιστική ισχύ κάτι που τους καθιστά κατάλληλους για συστήματα με περιορισμένους πόρους. Στη συνέχεια, παρουσιάζονται αναλυτικά οι κυριότερες κατηγορίες αυτών των αισθητήρων:

1. Οι υπερηχητικοί αισθητήρες λειτουργούν στέλνοντας παλμούς ήχου υψηλής συχνότητας και μετρώντας τον χρόνο που χρειάζεται το ανακλώμενο σήμα για να επιστρέψει, ώστε να υπολογιστεί η απόσταση ενός εμποδίου. Τα κύρια πλεονεκτήματά τους είναι το χαμηλό κόστος, η απλότητα στη χρήση και το γεγονός ότι δεν επηρεάζονται από το φωτισμό [5]. Ωστόσο οι υπερηχητικοί αισθητήρες παρουσιάζουν σημαντικούς περιορισμούς, καθώς η ακρίβεια τους μειώνεται σε μεγάλες αποστάσεις ενώ η φαρδιά δέσμη τους δυσκολεύεται να εντοπίσει λεπτά αντικείμενα [6]. Επιπλέον, παράγοντες όπως ο άνεμος, η θερμοκρασία και η υγρασία μπορούν να επηρεάσουν τις μετρήσεις τους. Για τον λόγο αυτό η χρήση τους περιορίζεται συνήθως σε μικρές αποστάσεις και σε εφαρμογές που δεν απαιτούν υψηλή ακρίβεια.
2. Οι υπέρυθροι αισθητήρες λειτουργούν εκπέμποντας υπέρυθρη ακτινοβολία και ανιχνεύοντας την αντανάκλαση της ώστε να υπολογίσουν την απόσταση με βάση την ένταση του σήματος που επιστρέφει. Σε κοντινές αποστάσεις προσφέρουν γρήγορη και αρκετά ακριβή απόκριση καθιστώντας τους χρήσιμους για εφαρμογές αποφυγής εμποδίων. Το κύριο πλεονέκτημα των υπέρυθρων αισθητήρων αποτελεί ο μικρός χρόνος απόκρισης και η απλή υλοποίηση τους [3].

Από την άλλη η απόδοση τους εξαρτάται από τις ιδιότητες των επιφανειών και τις συνθήκες φωτισμού. Συγκεκριμένα, σκούρα ή απορροφητικά υλικά μειώνουν την ανακλαστικότητα, ενώ η έντονη ηλιακή ακτινοβολία μπορεί να επηρεάσει τη μέτρηση περιορίζοντας την αξιοπιστία τους σε εξωτερικούς χώρους [6].

3. Τα συστήματα LiDAR (Light Detection and Ranging) αποτελούν την πιο εξελιγμένη λύση για μέτρηση απόστασης και αποφυγής εμποδίων. Λειτουργούν εκπέμποντας παλμούς λέιζερ και μετρώντας τον χρόνο που χρειάζεται το φως για να επιστρέψει (Time of Flight). Ανάλογα με τη λειτουργεία τους διακρίνονται σε δύο κατηγορίες τα σαρωτικά LiDAR που δημιουργούν τρισδιάστατες (3D) απεικόνιση του περιβάλλοντος, και τα μονοδιάστατα (1D) τα οποία μετρούν απόσταση σε συγκεκριμένη κατεύθυνση [7]. Η μεγάλη ακρίβεια και η υψηλή εμβέλεια τα καθιστούν ιδιαίτερα αποτελεσματικά σε σύνθετα περιβάλλοντα, καθώς μπορούν να εντοπίσουν ακόμα και λεπτά ή δύσκολα εμπόδια. Παράλληλα η απόδοση δεν επηρεάζεται από τον φωτισμό γεγονός που το κάνει αξιόπιστο τόσο την ημέρα όσο και την νύχτα. Παρόλα αυτά, τα σαρωτικά LiDAR έχουν κάποιους περιορισμούς, καθώς το υψηλό κόστος, η αυξημένη κατανάλωση ενέργειας και το μεγαλύτερο βάρος αποτελούν σημαντικά μειονεκτήματα ειδικά για μικρά drone [8]. Αντίθετα τα μονοδιάστατα LiDAR προσφέρουν μια πιο οικονομική λύση διατηρώντας την ακρίβεια ενός LiDAR.

Στον πίνακα 2.1 παρουσιάζονται συνοπτικά τα χαρακτηριστικά των βασικών αισθητήρων απόστασης που χρησιμοποιούνται σε συστήματα αποφυγής εμποδίων, ως προς την επίδραση του φωτισμού, την ακρίβεια και το κόστος.

Πίνακας 2.1: Χαρακτηριστικά αισθητήρων απόστασης

Αισθητήρες	Επίδραση Φωτός	Ακρίβεια	Κόστος
Υπερηχητικοί	Καμία	Μέτρια	Χαμηλό
Υπέρυθροι (IR)	Υψηλή	Καλή	Χαμηλό
LiDAR 1D	Καμία	Πολύ Υψηλή	Χαμηλό
LiDAR 3D	Καμία	Πολύ Υψηλή	Υψηλό

Συνοψίζοντας οι αισθητήρες απόστασης αποτελούν βασικό στοιχείο στις μεθόδους αποφυγής εμποδίων για τα μη επανδρωμένα εναέρια οχήματα, καθώς προσφέρουν άμεση και αξιόπιστη πληροφορία για το πόσο κοντά βρίσκονται τα αντικείμενα στο περιβάλλον πτήσης. Κάθε τύπος αισθητήρα έχει τα δικά του χαρακτηριστικά σε ακρίβεια εμβέλειά και αντοχή σε περιβαλλοντικές συνθήκες. Συνεπώς, η επιλογή του κατάλληλου αισθητήρα εξαρτάται από τις απαιτήσεις τις εκάστοτε εφαρμογής.

2.2.2 Αισθητήρες Όρασης

Οι αισθητήρες όρασης χρησιμοποιούν εικόνες από την κάμερα για να καταλάβουν τι υπάρχει στο περιβάλλον. Σε αντίθεση με τις μεθόδους που βασίζονται σε άμεσες μετρήσεις απόστασης, η μηχανική όραση δεν υπολογίζει απευθείας την απόσταση από τα εμπόδια. Αντίθετα, εξάγει έμμεση πληροφορία μέσα από την ανάλυση της οπτικής σκηνής. Με αυτόν τον τρόπο, το UAV αποκτά την ικανότητα να αντιλαμβάνεται τον χώρο με τρόπο παρόμοιο με την ανθρώπινη όραση. Ανάλογα με την αρχιτεκτονική τους και τον τρόπο λειτουργίας τους, οι αισθητήρες αυτοί διακρίνονται στις εξής κατηγορίες:

1. Η στερεοσκοπική όραση (Stereo Vision) βασίζεται στην χρήση δύο καμερών τοποθετημένων σε μια συγκεκριμένη απόσταση μεταξύ τους ώστε να υπολογίζει το βάθος του χώρου συγκρίνοντας τις διαφορές ανάμεσα στις δύο εικόνες. Με αυτόν το τρόπο το σύστημα μπορεί να εκτιμά άμεσα την απόσταση των αντικειμένων κάτι που καθιστά ιδιαίτερα χρήσιμη σε εφαρμογές αποφυγής εμποδίων [3]. Ένα στερεοσκοπικό σύστημα εντοπίζει κοινά σημεία στις δύο εικόνες και δημιουργεί έναν χάρτη βάθους (Depth map) ο οποίος περιγράφει τη γεωμετρία του περιβάλλοντος [9]. Ο χάρτης αυτός παρέχει λεπτομερή πληροφορία για τη θέση και το σχήμα των αντικειμένων, επιτρέποντας την ακριβή αναγνώριση εμποδίων και την εκτίμηση ελεύθερων περιοχών κίνησης, τόσο για στατικά όσο και για κινούμενα στοιχεία. Ωστόσο, η απόδοση της επηρεάζεται σημαντικά από τις συνθήκες χαμηλού φωτισμού καθώς η έλλειψη φωτός εμποδίζει την εύρεση κοινών σημείων αναμεσά στις εικόνες του stereo συστήματος. Επιπλέον, η επεξεργασία της αντιστοίχισης των pixels απαιτεί ισχυρή υπολογιστική ισχύ κάτι που μπορεί να περιορίσει τη χρήση της σε συστήματα με περιορισμένους πόρους. Τέλος σε επιφάνειες χωρίς ιδιαίτερη υφή ο αλγόριθμος αδυνατεί να βρει σημεία αντιστοίχισης με αποτέλεσμα να χάνει το εμπόδιο [10].
2. Οι μονές κάμερες (Monocular Cameras) βασίζονται στη χρήση μιας και μόνο κάμερας για την αντίληψη του περιβάλλοντος. Παρόλο που η διάταξη αυτή δεν παρέχει άμεση πληροφορία βάθους , προτιμάται συχνά λόγω του εξαιρετικά χαμηλού κόστους, του περιορισμένου βάρους και της υπολογιστικής απλότητας [11]. Τα χαρακτηριστικά αυτά την καθιστούν ιδιαίτερα κατάλληλη για μικρά drones. Η εκτίμηση της απόστασης επιτυγχάνεται μέσω προηγμένων τεχνικών επεξεργασίας εικόνας, όπως η ανάλυση της οπτικής ροής και η αντιστοίχιση χαρακτηριστικών, οι οποίες επιτρέπουν τον έμμεσο εντοπισμό αντικειμένων που πλησιάζουν [12]. Παρά τα πλεονεκτήματα της παρουσιάζει και σημαντικούς περιορισμούς. Η απουσία άμεσης μέτρησης βάθους μπορεί να μείωση την ακρίβεια σε απαιτητικές εφαρμογές, ενώ η απόδοση της επηρεάζεται έντονα από τις συνθήκες φωτισμού [12].
3. Οι κάμερες RGB-D (Red Green Blue Depth) συνδυάζουν μια κλασική RGB κάμερα και έναν ενεργό αισθητήρα βάθους. Με αυτόν τον τρόπο μπορούν να παρέχουν ταυτόχρονα την έγχρωμη εικόνα και έναν πλήρη χάρτη απόστασης για κάθε pixel προσφέροντας άμεση και αξιόπιστη πληροφορία βάθους χωρίς την ανάγκη υπολογιστικά απαιτητικών διαδικασιών [13]. Παρότι έχουν σημαντικά οφέλη οι RGB-D κάμερες παρουσιάζουν και σημαντικούς περιορισμούς. Η εμβέλεια τους είναι συνήθως περιορισμένη σε αποστάσεις της τάξης των 5 έως 10 μέτρων. Ακόμα ο αισθητήρας βάθους που βασίζεται σε υπέρυθρη ακτινοβολία επηρεάζεται έντονα από το ηλιακό φως το οποίο μπορεί να προκαλέσει παρεμβολές και μείωση την ακρίβεια [14].

Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά οι βασικοί τύποι καμερών που χρησιμοποιούνται σε μεθόδους αποφυγής εμποδίων βασισμένες στη μηχανική όραση. Η σύγκριση αφορά την παροχή πληροφορίας βάθους, την επίδραση των συνθηκών φωτισμού, το κόστος και τις υπολογιστικές απαιτήσεις κάθε προσέγγισης.

Πίνακας 2.2: Χαρακτηριστικά καμερών μηχανικής όρασης

Τύπος κάμερας	Πληροφορία Βάθους	Επίδραση Φωτός	Κόστος	Υπολογιστική απαίτηση
Stereo Vision	Άμεση	Υψηλή	Χαμηλό	Υψηλή
Monocular	Δεν Παρέχεται	Πολύ Υψηλή	Χαμηλό	Χαμηλή
RGB-D	Άμεση	Υψηλή	Υψηλό	Μέτρια

Κλείνοντας, μηχανική όραση αποτελεί βασικό εργαλείο για την αποφυγή εμποδίων καθώς επιτρέπουν στο σύστημα να κατανοεί το περιβάλλον πτήσης μέσα από την ανάλυση της εικόνας. Από τη στερεοσκοπική όραση που προσφέρει λεπτομερή πληροφορία βάθους μέχρι τις κάμερες RGB-D και τις πιο ελαφριές μονοφθαλμικές λύσεις οι διαθέσιμες τεχνολογίες καλύπτουν ένα ευρύ φάσμα δυνατοτήτων. Κάθε μια από αυτές φέρει τα δικά της χαρακτηριστικά και απαιτήσεις σε ακρίβεια, κόστος και υπολογιστική ισχύ. Επειδή όλες οι οπτικές μέθοδοι επηρεάζονται από τον φωτισμό και την υφή των επιφανειών η επιλογή της κατάλληλης τεχνολογίας εξαρτάται τελικά από το βάρος του UAV και τις συνθήκες του χώρου όπου πρόκειται να επιχειρήσει.

2.3 Μεθοδολογίες και Αλγόριθμοι Αποφυγής Εμποδίων

Αφού το σύστημα συλλέξει τα δεδομένα από τους αισθητήρες (όπως περιεγράφηκε στην ενότητα 2.2) το επόμενο στάδιο είναι η επεξεργασία αυτών των πληροφοριών ώστε να ληφθούν οι κατάλληλες αποφάσεις πλοήγησης. Τα δεδομένα που προέρχονται από τους αισθητήρες από μόνα τους δεν έχουν πρακτική αξία αν δεν μετατραπούν σε ενεργείς που θα καθοδηγήσουν το UAV με ασφάλεια. Μέσα από κατάλληλους αλγορίθμους το σύστημα αξιοποιεί τα δεδομένα για να παράγει τις εντολές κίνησης που μπορούν να εφαρμοστούν σε πραγματικό χρόνο. Με την εξέλιξη της τεχνολογίας έχουν αναπτυχθεί διαφορετικές προσεγγίσεις για την αποφυγή εμποδίων κάθε μια μετρά δικά της χαρακτηριστικά και πλεονεκτήματα. Ορισμένες μέθοδοι βασίζονται σε γεωμετρικές αρχές επιτρέποντα να λαμβάνει άμεσες αποφάσεις με βάση τη σχετική θέση και κατεύθυνση των εμποδίων. Άλλες τεχνικές χρησιμοποιούν χάρτες του περιβάλλοντος (Depth map data) προσφέροντας μια πιο ολοκληρωμένη εικόνα του χώρου και επιτρέποντας πιο προσεκτικό σχεδιασμό της πορείας. Οι πιο σύγχρονες προσεγγίσεις που βασίζονται στην μηχανική μάθηση που δίνουν τη δυνατότητα στο σύστημα να μαθαίνει από δεδομένα να αναγνωρίζει μοτίβα και να προσαρμόζεται σε δυναμικές συνθήκες που δεν μπορούν να προβλεφθούν ευκολά.

2.3.1 Μέθοδοι Χαρτογράφησης

Η χαρτογράφηση αποτελεί βασικό κομμάτι της αυτόνομης πλοήγησης γιατί επιτρέπει στο drone να δημιουργεί μια ψηφιακή εικόνα χώρου γύρω του και να μην βασίζεται μόνο σε στιγμιαίες μετρήσεις. Μια από τις πιο απλές μορφές χαρτογράφησης είναι οι χάρτες πλέγματος (Grid Maps), όπου ο χώρος χωρίζεται σε μικρά κελιά που χαρακτηρίζονται ως ελευθέρα ή κατειλημμένα από εμπόδια [15].

Για πιο συνθέτους τρισδιάστατους χώρους χρησιμοποιείται το OctoMap το οποίο οργανώνει τον χώρο σε 3D μέσω μιας δεντρικής δομής. Η προσέγγιση αυτή χρησιμοποιείται συχνά σε αισθητήρες όπως οι κάμερες RGB D που προσφέρουν ταυτόχρονα πληροφορία χρώματος και βάθους επιτρέποντας στο σκάφος να αντιλαμβάνεται τον όγκο και τη θέση των εμποδίων με μεγαλύτερη ακρίβεια.

Σε περιπτώσεις όπου το GPS δεν είναι διαθέσιμο, όπως σε εσωτερικούς χώρους χρησιμοποιούνται τεχνικές SLAM (Simultaneous Localizations and Mapping) οι οποίες επιτρέπουν στο όχημα να χαρτογραφεί τον χώρο και ταυτόχρονα να εντοπίζει τη θέση του μέσα σε αυτόν.

Η επιλογή της κατάλληλης τεχνικής εξαρτάται σε μεγάλο βαθμό από τους διαθέσιμους υπολογιστικούς πόρους καθώς για να δημιουργηθούν οι λεπτομερείς τρισδιάστατοι χάρτες απαιτούν επεξεργαστική ισχύ σε σχέση με τις απλούστερες λύσεις πλέγματος. Τελικά, κάθε μέθοδος έχει τη δική της αξία, ανάλογα με το ποσό δυναμικό πολύπλοκο ή περιορισμένο είναι το περιβάλλον πτήσης.

2.3.2 Μέθοδοι με Μηχανική Μάθηση

Οι μέθοδοι μηχανικής μάθησης αποτελούν μια από τις πιο σύγχρονες και εξελιγμένες προσεγγίσεις στην αποφυγή εμποδίων και επιτρέπουν στο σκάφος να προσαρμόζεται σε δυναμικά περιβάλλοντα και να λαμβάνει αποφάσεις με βάση πραγματικά δεδομένα [16]. Σε σχέση με τις κλασικές γεωμετρικές ή χαρτογραφικές τεχνικές, η μηχανική μάθηση επιτρέπει στο σύστημα να προσαρμόζεται δυναμικά, μαθαίνοντας από τα δεδομένα του περιβάλλοντος.

Η βασική μάθηση αποτελεί βασική κατηγορία αξιοποιώντας νευρωνικά δίκτυα για την ανάλυση της εικόνας. Αλγόριθμοι όπως το YOLO (You Only Look Once) [17] και το Faster R CNN μπορούν να εντοπίζουν αντικείμενα σε πραγματικό χρόνο προσδιορίζοντας όχι μόνο την ύπαρξη ενός εμποδίου αλλά και τη θέση και κατηγορία του. Έτσι το σύστημα καταλαβαίνει καλυτέρα το περιβάλλον κάτι που βελτιώνει την ακρίβεια της πλοήγησης.

Μια ακόμη προσέγγιση είναι η ενισχυτική μάθηση όπου το σκάφος μαθαίνει μέσω από συνεχείς δοκιμές. Αλγόριθμοι όπως το Deep Q Learning εκπαιδεύουν το σύστημα να επιλέγει ενέργειες που οδηγούν σε ασφαλέστερη και πιο αποδοτική πτήση [18].

Αν και οι μέθοδοι μηχανικής μάθησης προσφέρουν υψηλή προσαρμοστικότητα σε άγνωστα περιβάλλοντα απαιτούν συνήθως σημαντική υπολογιστική ισχύ και μεγάλο όγκο δεδομένων εκπαίδευσης για να λειτουργήσουν αξιόπιστα.

2.3.3 Γεωμετρικές Μέθοδοι

Οι γεωμετρικοί μέθοδοι αποτελούν μια από τις πιο βασικές κατηγορίες αλγορίθμων αποφυγή εμποδίων. Βασίζονται στη χρήση της γεωμετρίας του χώρου και μαθηματικών μοντέλων για τον υπολογισμό της πιο ασφαλής πορείας με περιορισμένες απαιτήσεις σε υπολογιστική ισχύ. Η μέθοδος αυτή περιλαμβάνει τεχνικές όπως οι ζώνες ασφαλείς (Safety Zones) όπου γύρω από κάθε εμπόδιο ορίζεται μια περιοχή που το μη επανδρωμένο αεροσκάφος δεν πρέπει να εισέρχεται [19]. Έπειτα υπάρχει και η μέθοδος εμπόδια ταχύτητας (Velocity Obstacles) η οποία προβλέπει πιθανές συγκρούσεις εξετάζοντας την σχετική ταχύτητα και θέση των αντικειμένων. Μια ακόμα προσέγγιση είναι ο κώνος σύγκρουσης (Collision Cone) όπου προσφέρει μεγαλύτερη προσαρμοστικότητα υπολογίζοντας ένα ευρύ φάσμα γωνιών κατεύθυνσης που εξασφαλίζουν την αποφυγή εμποδίων.

Οι γεωμετρικές μέθοδοι έχουν και ορισμένα μειονεκτήματα. Αρχικά, δεν ανταποκρίνονται πάντα καλά όταν υπάρχουν πολλά κινούμενα εμπόδια ή όταν το περιβάλλον αλλάζει γρηγορά επειδή βασίζονται σε απλά μοντέλα που δεν περιγράφουν πλήρως τη δυναμική του χώρου. Επιπλέον, υπάρχει το ενδεχόμενο το UAV να παγιδευτεί σε μια θέση όπου οι δυνάμεις από τον στόχο και τα εμπόδια ισορροπούν με αποτέλεσμα να μην μπορεί να συνεχίσει την πορεία του. Παρόλα αυτά οι μέθοδοι αυτές λόγω της απλότητας τους και της ταχύτητας τους και της αξιοπιστίας τους καθίστανται χρήσιμες σε εφαρμογές πραγματικού χρόνου [20].

2.4 Σύγκριση Παρομοίων Υλοποιήσεων

Στην παρούσα ενότητα παρουσιάζεται μια συνοπτική σύγκριση ανάμεσα στην προτεινόμενη υλοποίηση και σε άλλες σχετικές προσεγγίσεις που εντοπίζονται στη βιβλιογραφία. Η ανάγκη για αποφυγή εμποδίων στα μη επανδρωμένα αεροσκάφη έχει οδηγήσει στην ανάπτυξη πολλών διαφορετικών συστημάτων, τα οποία αξιοποιούν διαφορετικές τεχνικές και αρχιτεκτονικές.

Στόχος της ανάλυσης είναι να παρουσιαστούν οι βασικές τεχνολογικές επιλογές που χρησιμοποιούνται πιο συχνά, καθώς και η θέση της προτεινόμενης υλοποίησης σε σχέση με τις υπάρχουσες λύσεις.

Στην εργασία των Moffatt et al. [21] παρουσιάζεται ένα αυτόνομο σύστημα ανίχνευσης και αποφυγής εμποδίων για μικρά drone. Η λύση βασίζεται σε αισθητήρα LiDAR velodyne VLP16, σε γεωμετρικούς αλγορίθμους πλοήγησης και στον ελεγκτή πτήσης Pixhawk 2. Τον κεντρικό ρόλο στην επεξεργασία έχει ένας ισχυρός υπολογιστής Intel NUC ο οποίος επικοινωνεί με τον ελεγκτή πτήσης μέσω της βιβλιοθήκης DroneKit και εκτελεί τους αλγόριθμους αποφυγής. Με αυτό τον συνδυασμό το σύστημα δημιουργεί σε πραγματικό χρόνο ένα τρισδιάστατο μοντέλο του περιβάλλοντος. Έτσι μπορεί να υπολογίζει συνεχώς νέες κατευθύνσεις κίνησης και να αποφεύγει στατικά και κινούμενα εμπόδια χωρίς ανθρώπινη παρέμβαση. Η λειτουργικότητα των αλγορίθμων επιβεβαιώθηκε τόσο σε προσομοιώσεις όσο και σε επίγειες δοκιμές όπου παράχθηκαν σωστές εντολές αποφυγής. Ωστόσο κατά τη μετάβαση σε εναέριες δοκιμές προέκυψαν σημαντικές προκλήσεις καθώς η αυξημένη υπολογιστική πολυπλοκότητα οδήγησε σε καθυστερήσεις στην απόκριση. Τελικά, ένα σφάλμα στην καταγραφή δεδομένων προκάλεσε ατύχημα, δείχνοντας πόσο απαιτητική είναι η διαχείριση δεδομένων σε πραγματικό χρόνο.

Επίσης οι Kumar et al. [22] παρουσίασαν ένα σύστημα αποφυγής εμποδίων σε πραγματικό χρόνο για μη επανδρωμένο αεροσκάφος που πετάει σε περιοχές χωρίς κάλυψη GPS. Η αρχιτεκτονική τους συνδυάζει στερεοσκοπική κάμερα (OAK D Lite) με τον αλγόριθμο YOLO ο οποίος έχει εκπαιδευτεί στο σύνολο δεδομένων COCO ώστε το drone να μπορεί να αναγνωρίζει αντικείμενα στο περιβάλλον του. Το σύστημα βασίζεται σε έναν μικροϋπολογιστή Raspberry Pi 4B για τη επεξεργασία εικόνας και σε έναν ελεγκτή πτήσης Pixhawk Cube Orange για τον έλεγχο του UAV. Η λειτουργία του συστήματος στηρίζεται στην πληροφορία βάθους που παρέχει η στερεοσκοπική κάμερα. Όταν εντοπιστεί εμπόδιο σε απόσταση μικρότερη από δύο μέτρα το σκάφος εκτελεί αυτόματους ελιγμούς αποφυγής ώστε να κινηθεί προς ασφαλή κατεύθυνση. Στα πειράματα που πραγματοποιήθηκαν το σύστημα πέτυχε ποσοστό ανίχνευσης περίπου 90% με ταχύτητα πτήσης 2 m/s (metre per second).

Επιπλέον, μια διαφορετική υλοποίηση παρουσιάζεται από τους Zhang et al. [23], οι οποίοι ανέπτυξαν ένα σύστημα αποφυγής εμποδίων χρησιμοποιώντας ένα εμπορικό τετρακόπτερο. Η προσέγγιση τους βασίζεται αποκλειστικά στην μπροστινή κάμερα του τετρακοπτέρου αξιοποιώντας ένα νευρωνικό δίκτυο (MegaDepth) που υπολογίζει το βάθος απευθείας από την εικόνα. Επειδή το μοντέλο απαιτεί μεγάλη υπολογιστική ισχύ η επεξεργασία δεν πραγματοποιήθηκε πάνω στο drone. Οι εικόνες μεταφέρονταν μέσω Wi-Fi σε έναν φορητό υπολογιστή στο έδαφος όπου γινόταν η ανάλυση. Στη συνέχεια μέσω ROS (Robot Operating System) ο υπολογιστής έστελνε πίσω στο τετρακόπτερο τις απαραίτητες εντολές για να αποφύγει το εμπόδιο και να συνεχίσει την πορεία του.

Ακόμα Ait-Jellal και Zell [24] παρουσίασαν ένα αυτόνομο σύστημα αποφυγής εμποδίων για τετρακόπτερα το οποίο βασίζεται σε στερεοσκοπική κάμερα και τρισδιάστατή χαρτογράφηση. Το drone χρησιμοποίει τη κάμερα για την παραγωγή στερεοσκοπικών σημείων τα οποία επεξεργάζονται από έναν αλγόριθμο Visual Stereo SLAM. Με αυτόν τον τρόπο επιτυγχάνεται σταθερή εκτίμηση θέσης και δημιουργία τρισδιάστατου χάρτη του περιβάλλοντος σε πραγματικό χρόνο. Οι πληροφορίες βάθους

οργανώνονται σε ένα τρισδιάστατο χάρτη (OctoMap) πάνω στο οποίο εφαρμόζονται ο αλγόριθμος RRTS (Rapidly exploring Random Tree Star) για τον σχεδιασμό ασφαλών διάδρομων αποφυγής. Η επεξεργασία γίνεται πάνω στο σκάφος μέσω ενός υπολογιστή Intel NUC, ενώ ο Pixhawk αναλαμβάνει τον έλεγχο πτήσης. Το σύστημα δοκιμάστηκε σε πραγματικές πτήσεις σε εξωτερικούς χώρους και κατάφερε να προηγηθεί αυτόνομα χωρίς ανθρώπινη παρέμβαση.

Συμπερασματικά, η επισκόπηση της βιβλιογραφίας δείχνει ότι υπάρχουν αρκετές διαφορετικές προσεγγίσεις ως προς το υλικό και τον τρόπο επεξεργασίας. Οι υλοποιήσεις των Moffat et al. [21] και Ait-Jellal & Zell [24] πέτυχαν υψηλή ακρίβεια, όμως βασίζονται σε βαριές υπολογιστικές μονάδες όπως Intel NUC και ακριβούς αισθητήρες όπως ο LiDAR 3D και στερεοσκοπικές κάμερες. Αυτό οδηγεί σε σημαντική αύξηση του κόστους, του βάρους αλλά και της κατανάλωσης ενέργειας για μικρού μεγέθους και χαμηλού κόστους εναέρια συστήματα.

Από την άλλη πλευρά, η προσέγγιση των Zhang et al. [23] μειώνει το βάρος αφού χρησιμοποιεί απλή κάμερα. Ωστόσο, παρουσιάζει λειτουργικούς περιορισμούς καθώς η επεξεργασία γίνεται σε εξωτερικό υπολογιστή εδάφους γεγονός που καθιστά το σύστημα εξαρτημένο από την ασύρματη σύνδεση. Πιο κοντά στη φιλοσοφία της παρούσας διατριβής βρίσκεται η εργασία των Kumar et al. [22], η οποία εκτελεί την επεξεργασία πάνω στο ίδιο το σκάφος.

Η προτεινόμενη υλοποίηση επιχειρεί να συνδυάσει τα θετικά στοιχεία των παραπάνω προσεγγίσεων προσφέροντας μια λύση που παραμένει αυτόνομη, ελαφριά και χαμηλού κόστους. Αποφεύγοντας τόσο τον βαρύ εξοπλισμό των Moffat et al. και Ait-Jellal & Zell όσο και την εξάρτηση από σταθμούς εδάφους, το σύστημά μας αξιοποιεί την αρχιτεκτονική του Raspberry Pi για να εκτελεί όλη την επεξεργασία πάνω στο drone σε συνεργασία με τον ελεγκτή πτήση Pixhawk. Με αυτό τον τρόπο επιτυγχάνεται ανίχνευση και αποφυγή εμποδίων σε πραγματικό χρόνο προσφέροντας μια πρακτική λύση για εφαρμογές περιορισμένων πόρων.

Κεφάλαιο 3ο: Σχεδιασμός και Αρχιτεκτονική του UAV

3.1 Εισαγωγή

Το κεφάλαιο αυτό επικεντρώνεται στον συνολικό σχεδιασμό και την αρχιτεκτονική του μη επανδρωμένου εναέριου οχήματος που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας. Στόχος του κεφαλαίου είναι να παρουσιαστεί η συνολική δομή του συστήματος και ο τρόπος με τον οποίο τα επιμέρους υποσυστήματα επιλέχθηκαν και συνδυάστηκαν ώστε να λειτουργούν ως ένα ενιαίο και αξιόπιστο σύνολο.

Αρχικά περιγράφονται τα υλικά που χρησιμοποιήθηκαν για την κατασκευή του UAV από τα μηχανικά μέρη όπως το πλαίσιο και τα μέρη που κινούν το σύστημα (μοτέρ, έλικες) έως τα ηλεκτρονικά και υπολογιστικά υποσυστήματα. Για κάθε στοιχείο παρουσιάζεται ο ρόλος του και η συμβολή του στη συνολική λειτουργία λαμβάνοντας υπόψη την καταλληλόλητά του για την προτεινόμενη υλοποίηση.

Στη συνέχεια παρουσιάζονται αναλυτικά οι καλωδιώσεις και οι συνδέσεις των επιμέρους υποσυστημάτων του UAV. Ιδιαίτερη έμφαση δίνεται στη διανομή ισχύος, στη σύνδεση των μοτέρ και των ηλεκτρονικών ελεγκτών ταχύτητας (ESC), καθώς και στη διασύνδεση της υπολογιστικής μονάδας Raspberry Pi με τον ελεγκτή πτήσης Pixhawk, την κάμερα και τον αισθητήρα LiDAR. Η σωστή υλοποίηση των συνδέσεων αυτών αποτελεί βασική προϋπόθεση για την αξιόπιστη ανταλλαγή δεδομένων και την ασφαλή λειτουργία του συστήματος.

Τέλος, παρουσιάζεται το συνολικό κόστος των υλικών που χρησιμοποιήθηκαν για την κατασκευή και την ολοκλήρωση του μη επανδρωμένου εναέριου οχήματος.

3.2 Επιλογή Υλικών

Η επιλογή των υλικών για την κατασκευή του συστήματος αποτελεί βασικό στάδιο της υλοποίησης καθώς επηρεάζει άμεσα τη λειτουργία, την αξιοπιστία και τη συμπεριφορά του κατά την πτήση. Στο πλαίσιο της εργασίας τα επιμέρους εξαρτήματα επιλέχθηκαν με στόχο τη δημιουργία ενός σταθερού και λειτουργικού μη επανδρωμένου εναέριου οχήματος ικανού να υποστηρίξει αισθητήρες αποφυγής εμποδίων και επεξεργασία δεδομένων σε πραγματικό χρόνο.

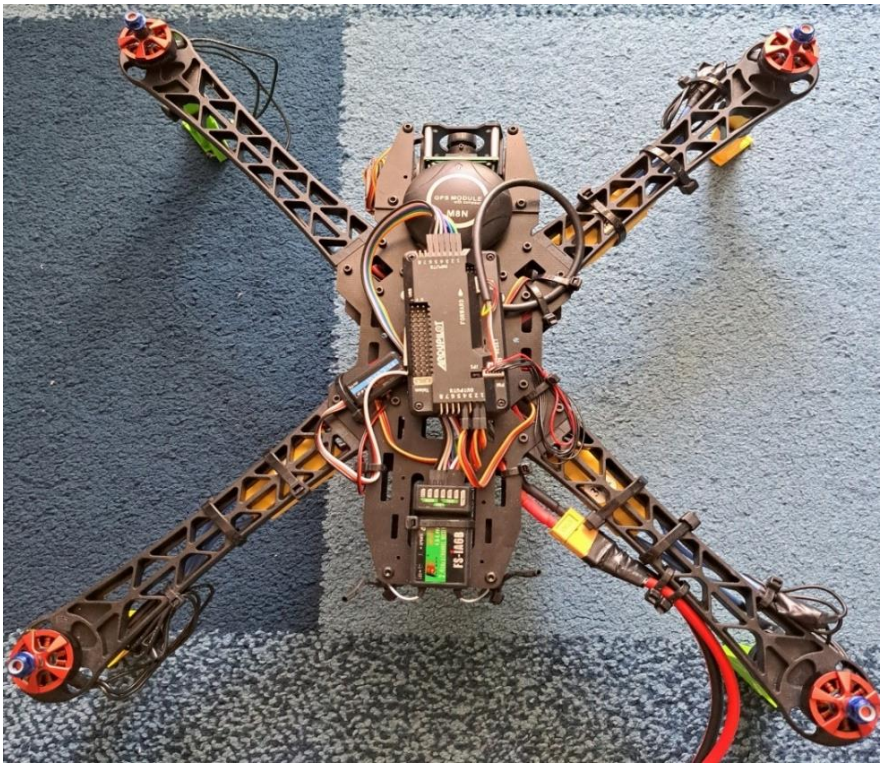
Επίσης, κατά την επιλογή των υποσυστημάτων εξετάστηκαν η μεταξύ τους συμβατότητα, το συνολικό βάρος, η κατανάλωση ενέργειας και η ευκολία ενσωμάτωσης και ρύθμισης. Παράλληλα, λήφθηκαν υπόψη πρακτικοί παράγοντες, όπως η διαθεσιμότητα των εξαρτημάτων και το κόστος.

3.2.1 Πλαίσιο

Το πλαίσιο (Frame) ενός αεροσκάφους αποτελεί τη βασική δομική υποδομή καθώς πάνω σε αυτό στηρίζονται και συνδέονται όλα τα επιμέρους μηχανικά, ηλεκτρικά και υπολογιστικά υποσυστήματα. Ο ρόλος του πλαισίου δεν περιορίζεται μόνο ως μηχανική βάση αλλά επηρεάζει άμεσα τη σταθερότητα πτήσης, τη συμπεριφορά στους κραδασμούς και την ευκολία ενσωμάτωσης πρόσθετων υποσυστημάτων. Για τον λόγο αυτό η επιλογή του πλαισίου αποτελεί κρίσιμη παράμετρο στον σχεδιασμό του συστήματος.

Για τις ανάγκες της παρούσας εργασίας χρησιμοποιήθηκε πλαίσιο τύπου F450, το οποίο ανήκει στην κατηγορία των τετρακοπτέρων και χαρακτηρίζεται από διάταξη κινητήρων σε σχήμα «X» όπως απεικονίζεται και στο σχήμα 3.1. Το συγκεκριμένο πλαίσιο χρησιμοποιείται συχνά σε ερευνητικά και

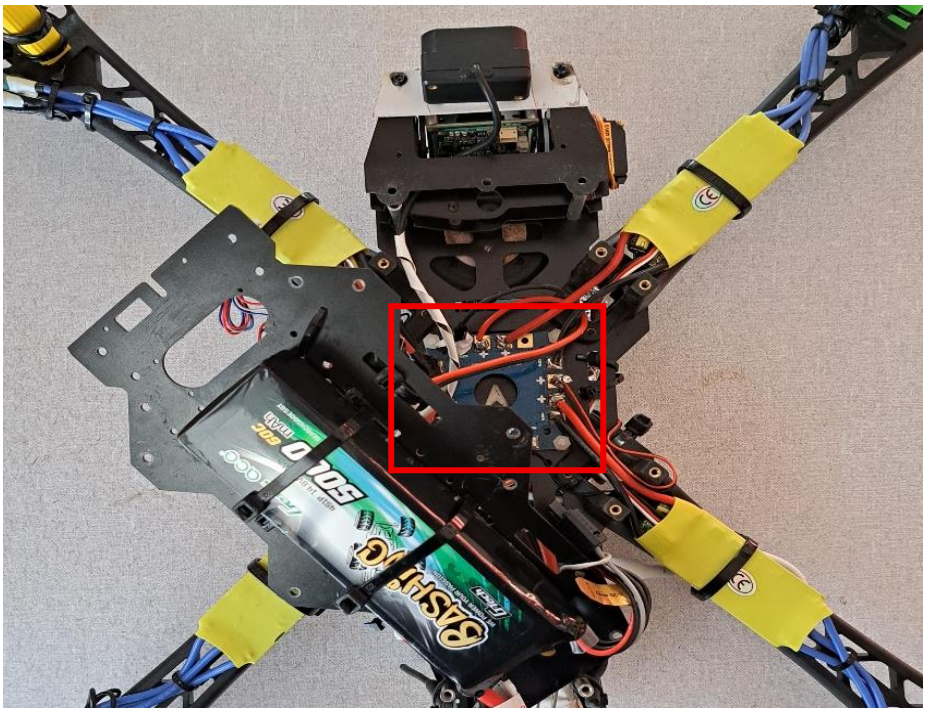
εκπαιδευτικά έργα επειδή είναι απλό αξιόπιστο και προσφέρει αρκετό χώρο για την τοποθέτηση επιπλέον εξοπλισμού.



Σχήμα 3.1: Το πλαίσιο F450 με τη διάταξη των κινητήρων

Το υλικό της κατασκευής είναι φτιαγμένο από μέταλλο προσφέροντας καλή μηχανική αντοχή και σταθερότητα. Η χρήση του μεταλλικού πλαισίου βοηθά στη διατήρηση της σταθερότητας στο όχημα κατά την πτήση. Στο κεντρικό μέρος της κατασκευής υπάρχει αρκετός χώρος για την τοποθέτηση των ηλεκτρικών μονάδων, γεγονός που διευκολύνει την οργανωμένη εγκατάσταση του συστήματος.

Επιπλέον στην κάτω πλευρά του κεντρικού τμήματος έχει τοποθετηθεί ξεχωριστή πλακέτα διανομής ισχύος (PDB) όπως απεικονίζεται και στο σχήμα 3.2, η οποία λειτουργεί ως κόμβος παροχής ενέργειας από την μπαταρία προς τους ρυθμιστές ταχύτητας και τα υπόλοιπα ηλεκτρικά συστήματα. Η ύπαρξη ανεξάρτητης πλακέτας απλοποιεί σημαντικά την ηλεκτρική αρχιτεκτονική του συστήματος, μειώνει την πολυπλοκότητα των καλωδιώσεων και διευκολύνει τον έλεγχο, τη συντήρηση και τις μελλοντικές επεκτάσεις.



Σχήμα 3.2: Πλακέτα διανομής Ισχύος (PDB)

Οι τέσσερις βραχίονες που κρατούν τους κινητήρες είναι φτιαγμένοι από ενισχυμένο πολυμερές (PA66 με 30% υαλονήματα), υλικό που αποτελεί τυπική επιλογή στη συγκεκριμένη κατηγορία πλαισίων. Το υλικό αυτό συνδυάζει ικανοποιητική ακαμψία και ελαστικότητα. Η ελαστικότητα του υλικού βοηθά στην απορρόφηση των φορτίων κατά την προσγείωση ή σε μικρές προσκρούσεις και μειώνει τους κραδασμούς που δημιουργούνται από τη λειτουργία των κινητήρων. Έτσι περιορίζεται η μετάδοση των δονήσεων προς το κεντρικό σώμα του UAV, όπου βρίσκονται ευαίσθητα ηλεκτρονικά και αισθητήρια συστήματα.

Συνολικά, το πλαίσιο F450 αποδείχθηκε μια πρακτική και αξιόπιστη επιλογή για τις ανάγκες της εργασίας καθώς προσφέρει καλή αντοχή, σταθερή πτήση, εύκολη ενσωμάτωση των εξαρτημάτων και αρκετή ευελιξία για δοκιμές σε πραγματικές συνθήκες. Παράλληλα, η δομή του επιτρέπει την εύκολη ενσωμάτωση των απαραίτητων ηλεκτρονικών εξαρτημάτων, χωρίς να απαιτούνται πολύπλοκες τροποποιήσεις ή ειδικές βάσεις.

3.2.2 Ηλεκτροκινητήρες

Οι ηλεκτροκινητήρες αποτελούν το βασικότερο στοιχείο του UAV καθώς μετατρέπουν την ηλεκτρική ενέργεια σε ώση, επιτρέποντας την απογείωση και τους ελιγμούς του σκάφους. Η επιλογή τους επηρεάζει άμεσα τη σταθερότητα πτήσης, την ταχύτητα απόκρισης αλλά και την κατανάλωση ενέργειας.

Στην παρούσα εργασία χρησιμοποιήθηκαν τέσσερις ηλεκτροκινητήρες τύπου Brushless DC (BLDC) εξωτερικού ρότορα μοντέλου Returner R3 2207-2550KV, όπως απεικονίζεται στο παρακάτω σχήμα 3.3.

Σχήμα 3.3: Ηλεκτροκινητήρας Brushless DC Returner R3 2207-2550KV [25]

Οι BLDC κινητήρες χρησιμοποιήθηκαν επειδή προσφέρουν υψηλή απόδοση και μεγαλύτερη αξιοπιστία σε σχέση με κινητήρες με ψήκτρες. Η απουσία μηχανικών επαφών όπως οι ψήκτρες μειώνει τις απώλειες λόγω τριβών και περιορίζει τη φθορά, προσφέροντας μεγαλύτερη διάρκεια ζωής και σταθερότερη λειτουργία, χαρακτηριστικά ιδιαίτερα σημαντικά για πειραματικές και ερευνητικές εφαρμογές.

Ο κωδικός 2207 αναφέρεται στις διαστάσεις του στάτορα με διάμετρο 22mm και ύψος 7mm. Το συγκεκριμένο μέγεθος αποτελεί μια συνηθισμένη επιλογή για μη επανδρωμένο εναέριο όχημα μεσαίου μεγέθους διότι προσφέρει καλή ισορροπία ανάμεσα σε βάρος, ροπή και μέγιστη ισχύ.

Ακόμη, η σταθερά ταχύτητας KV αποτέλεσε βασικό κριτήριο για την επιλογή των κινητήρων καθώς δείχνει πόσες στροφές ανά λεπτό παράγει ο κινητήρας για κάθε Volt τάσης χωρίς φορτίο (RPM per Volt). Στην παρούσα υλοποίηση χρησιμοποιήθηκαν κινητήρες 2550KV δηλαδή κινητήρες υψηλών στροφών. Η επιλογή αυτή βοηθά το σκάφος να ανταποκρίνεται πιο γρήγορα στις εντολές του ελεγκτή πτήσης και να κινείται πιο ευέλικτα, κάτι που βελτιώνει τη σταθερότητα του συστήματος αποφυγής εμποδίων.

Ωστόσο, οι κινητήρες υψηλού KV αποδίδουν μικρότερη ροπή σε σχέση με κινητήρες χαμηλότερου KV. Για τον λόγο αυτό η χρήση τους χρειάζεται να συνδυάζεται με προπέλες μικρής διαμέτρου ώστε να αποφεύγεται η υπερφόρτωση και υπερθέρμανση των κινητήρων κατά τη λειτουργία.

Σε διάταξη τετρακοπτέρου τύπου Quad-X, οι κινητήρες λειτουργούν σε ζεύγη αντίθετης περιστροφής δηλαδή δύο δεξιόστροφοι (CW) και δύο αριστερόστροφοι (CCW). Η διάταξη αυτή συμβάλει στη μείωση της ροπής γύρω από τον κατακόρυφο άξονα και στη διατήρηση της σταθερότητας κατά την αιώρηση.

Συνολικά, οι κινητήρες 2207–2550KV αποτέλεσαν μια λειτουργική και αξιόπιστη επιλογή για την παρούσα υλοποίηση, καλύπτοντας τις απαιτήσεις του συστήματος ως προς την απόδοση, την απόκριση και τη συνεργασία με τα υπόλοιπα υποσυστήματα του UAV.

Στον πίνακα 3.1 παρουσιάζονται συνοπτικά τα κυριά τεχνικά χαρακτηριστικά των ηλεκτροκινητήρων που χρησιμοποιήθηκαν.

Πίνακας 3.1: Ηλεκτρικά χαρακτηριστικά ηλεκτροκινητήρων

KV (Στροφές/Volt)	2550
Διάμετρος	27.5mm
Μήκος	32mm
Προτεινομένη Μπαταριά	4-5S
Μέγιστη Ώση	1500g
Έλικα	5¨ (π.χ. 5040/5045)
Βάρος	30.8g

3.2.3 Ηλεκτρονικοί Ελεγκτές Ταχύτητας

Οι ηλεκτρονικοί ελεγκτές ταχύτητας (ESC) ελέγχουν τη λειτουργία των ηλεκτροκινητήρων, ρυθμίζοντας την ταχύτητα περιστροφής τους σύμφωνα με τις εντολές του ελεγκτή πτήσης. Αποτελούν βασικό στοιχείο του συστήματος κίνησης, καθώς χωρίς αυτό δεν είναι ομαλή και ελεγχόμενη λειτουργία των κινητήρων και συνεπώς σταθερή η πτήση του drone.

Στην παρούσα εργασία χρησιμοποιήθηκαν τέσσερις ESC τύπου Brushless ονομαστικής έντασης 40A (Ampere) όπως απεικονίζεται στο σχήμα3.4, ένας για την οδήγηση κάθε ηλεκτροκινητήρα . Η επιλογή τους βασίστηκε στις απαιτήσεις των κινητήρων 2207-2550KV, οι οποίοι σε συνθήκες μέγιστου φορτίου εμφανίζουν καταναλώσεις που προσεγγίζουν 35-37A. Συνεπώς, η επιλογή των 40A κρίθηκε απαραίτητη προκειμένου να καλύπτεται με επάρκεια η μέγιστη ζήτηση ρεύματος, διασφαλίζοντας ότι τα ηλεκτρονικά ισχύος παραμένουν εντός των ορίων λειτουργίας τους ακόμη και κατά τις κρίσιμες φάσεις της απογείωσης ή των απότομων ελιγμών. Με τον τρόπο αυτό αποφεύγεται η υπερθέρμανση και ενισχύεται η συνολική αξιοπιστία του συστήματος πρόωσης του UAV.

Σχήμα 3.4: Ελεγκτής Ταχύτητας ESC 40A

Οι συγκεκριμένοι ESC επιλέχθηκαν επειδή καλύπτουν πλήρως τις απαιτήσεις των κινητήρων, προσφέρουν επιπλέον περιθώριο προστασίας και συμβάλλουν στην αξιόπιστη λειτουργία του συστήματος. Τα πλήρη τεχνικά τους χαρακτηριστικά παρουσιάζονται συγκεντρωτικά στον πίνακα 3.2 που ακολουθεί.

Πίνακας 3.2: Τεχνικά Χαρακτηριστικά ESC

Μοντέλο	Xida XXD
Ampere Εξόδου	40
Μέγιστα Ampere Εξόδου	55
Μπαταριά	2-4S
Μέγεθος	68*25*8mm
Βάρος	35g

3.2.4 Έλικες

Οι έλικες είναι βασικό μέρος του συστήματος κίνησης του UAV, καθώς μετατρέπουν την ενέργεια των κινητήρων σε ώση. Η επιλογή τους επηρεάζει την παραγόμενη ελκτική δύναμη, την κατανάλωση ενέργειας, τη σταθερότητα και τη συνολική απόδοση του συστήματος.

Στη παρούσα υλοποίηση χρησιμοποιήθηκαν έλικες 5045 τριπτέρυγες (3 blade) όπως απεικονίζεται και στο σχήμα 3.5. Η ονομασία 5045 περιγράφει τα γεωμετρικά χαρακτηριστικά της έλικας, όπου ο αριθμός 5.0 αντιστοιχεί στη διάμετρο της προπέλας σε ίντσες και ο αριθμός 4.5 στο βήμα (pitch), δηλαδή τη θεωρητική απόσταση που καλύπτει η έλικα σε μια πλήρη περιστροφή.

Σχήμα 3.5: Τριπτέρυγη έλικα τύπου 5045

Η επιλογή τριπτέρυγων ελίκων έγινε με σκοπό τη βελτίωση της σταθερότητας και της ομαλής συμπεριφοράς του τετρακοπτέρου κατά την πτήση. Οι έλικες με τρία πτερύγια δημιουργούν πιο ομοιόμορφη ροή αέρα από τις δίπτερες μειώνοντας έτσι τις ταλαντώσεις και τους κραδασμούς ειδικά στην αιώρηση.

Επιπλέον οι έλικες 5045 προσφέρουν ικανοποιητική ώση χωρίς να επιβαρύνουν υπερβολικά τους κινητήρες, μειώνοντας την κατανάλωση ρεύματος και τον κίνδυνο υπερθέρμανσης. Σε συνδυασμό με τους ηλεκτροκινητήρες 2550KV που χρησιμοποιούμε επιτυγχάνεται μια καλή ισορροπία ανάμεσα στην απόκριση, τον έλεγχο και την ενεργειακή απόδοση.

Οι έλικες αυτοί επιλέχθηκαν επειδή προσφέρουν σταθερή πτήση, συνεργάζονται καλά με τους κινητήρες και λειτουργούν με ασφάλεια, καλύπτοντας τις ανάγκες της παρούσας κατασκευής.

3.2.5 Μπαταρία

Η μπαταρία αποτελεί ένα από τα σημαντικότερα υποσυστήματα του UAV, καθώς τροφοδοτεί με ενέργεια όλα τα ηλεκτρονικά και ηλεκτρικά μέρη του όπως τους ESC, τον ελεγκτή πτήσης και το υπολογιστικό σύστημα. Η σωστή επιλογή της επηρεάζει άμεσα τη διάρκεια πτήσης, τη διαθέσιμη ισχύ και τη γενικότερη αξιοπιστία του συστήματος.

Στο σύστημα χρησιμοποιήθηκε μπαταριά τύπου Li-Po (Lithium Polymer) μοντέλου Gens Ace G-Tech 5000mAh 14.8V 4S 60C με ακροδέκτη EC5 όπως απεικονίζεται και στο σχήμα 3.6. Οι Li-PO είναι ιδιαίτερα δημοφιλείς στα UAV, επειδή συνδυάζουν υψηλή ισχύ με χαμηλό βάρος κάτι που είναι κρίσιμο για την πτήση.

Η χωρητικότητα των 5000mAh καθορίζει την ενέργεια που μπορεί να αποθηκεύσει η μπαταρία και επηρεάζει τον χρόνο πτήσης. Η συγκεκριμένη χωρητικότητα επιλέχθηκε λαμβάνοντας υπόψη τη μέση κατανάλωση των κινητήρων, η οποία σε συνθήκες κανονικής πτήσης εκτιμάται περίπου στα 10A ανά κινητήρα, δηλαδή 40A συνολικά. Με βάση αυτή την κατανάλωση και χωρητικότητα 5000mAh (5Ah), ο θεωρητικός χρόνος πτήσης προκύπτει από τη σχέση (5Ah/40) × 60 = 7,5 λεπτά χρόνος που θεωρείται ικανοποιητικός για τη συγκεκριμένη εφαρμογή. Η επιλογή αυτή επιτρέπει την επίτευξη ικανοποιητικής αυτονομίας χωρίς σημαντική αύξηση του συνολικού βάρους, η οποία θα επηρέαζε αρνητικά τη σταθερότητα και την αποδοτικότητα του UAV.

Η ονομαστική τάση της μπαταρίας είναι 14.8V, καθώς πρόκειται για μια διαμόρφωση 4S δηλαδή τέσσερα στοιχεία (cells) Li-Po συνδεδεμένα σε σειρά. Κάθε στοιχείο έχει ονομαστική τάση 3.7V, παρέχοντας έτσι την απαιτούμενη τάση για να λειτουργεί σωστά το σύστημα.

Σημαντικό χαρακτηριστικό της μπαταρίας αποτελεί ο ρυθμός εκφόρτισης (C Rating) των 60C. Ο δείκτης C εκφράζει το μέγιστο ρεύμα που μπορεί να αποδώσει η μπαταρία με ασφάλεια. Στην προκειμένη περίπτωση η μπαταρία μπορεί να παρέχει έως και 300A τιμή που μπορεί να καλύψει τις στιγμιαίες αυξημένες απαιτήσεις ρεύματος των κινητήρων κατά την απογείωση χωρίς πτώση τάσης ή υπερθέρμανση.

Επιπλέον, για τη φόρτιση της μπαταρίας χρησιμοποιήθηκε ο φορτιστής SkyRC E430, κατάλληλος για Li-Po 4S.

Η συγκεκριμένη μπαταρία επιλέχθηκε επειδή καλύπτει πλήρως τις ενεργειακές απαιτήσεις του UAV, προσφέροντας ισορροπία μεταξύ ισχύος, αυτονομίας και αξιοπιστίας. Τα αναλυτικά τεχνικά της χαρακτηριστικά παρουσιάζονται συγκεντρωτικά στον πίνακα 3.3 που ακολουθεί.

Σχήμα 3.6: Μπαταρία Li-Po Gens Ace G-Tech 14.8V 5000mah

Πίνακας 3.3: Τεχνικά χαρακτηριστικά μπαταρίας Li-Po

Χωρητικότητα	5000mAh
Τάση	14.8V (4S)
Αντίσταση	60C
Σύνδεση	EC5
Βάρος	440g
Διαστάσεις	136*42*34mm

3.2.6 Μονάδα Διαχείρισης Ισχύος

Για την ασφαλή τροφοδοσία των ηλεκτρονικών συστημάτων και την παρακολούθηση της ενεργειακής κατάστασης του drone επιλέχθηκε η μονάδα APM Power Module 28V 90A η οποία απεικονίζεται στο σχήμα 3.7, ειδικά σχεδιασμένη για ελεγκτές πτήσης Pixhawk. Το συγκεκριμένο εξάρτημα παρεμβάλλεται μεταξύ της μπαταρίας Li-Po και του PDB, εξυπηρετώντας δύο βασικές λειτουργίες.

1. Μέσω του ενσωματωμένου ρυθμιστή τάσης υποβιβάζει την τάση εισόδου της μπαταρίας σε σταθερή έξοδο 5.3V με μέγιστο ρεύμα 3A. Η σταθεροποιημένη αυτή τάση μεταφέρεται μέσω του καλωδίου 6 pin και τροφοδοτεί τον ελεγκτή πτήσης ο οποίος στη συνέχεια αναλαμβάνει την διανομή ισχύος προς τα υπόλοιπα περιφερειακά συστήματα. Αξίζει να σημειωθεί ότι το Power Module δεν τροφοδοτεί τα μοτέρ, τα οποία τροφοδοτούνται απευθείας από την μπαταρία μέσω των ESC.
2. Διαθέτει αισθητήρες που μετρούν συνεχώς την τάση της μπαταρίας και την κατανάλωση ρεύματος. Τα δεδομένα μεταφέρονται στον ελεγκτή πτήσης μέσω του καλωδίου 6 pin, επιτρέποντας στο λογισμικό να υπολογίζει την υπολειπόμενη ενέργεια και να ενεργοποιεί διαδικασίες ασφάλειας σε περίπτωση χαμηλής στάθμης μπαταρίας.

Οι δύο αυτές λειτουργίες καθιστούν το Power Module κρίσιμο στοιχείο για την αξιόπιστη τροφοδοσία και την ενεργειακή επίβλεψη του συστήματος.

Σχήμα 3.7: Μονάδα Διαχείρισης Ισχύος

3.2.7 Ελεγκτής Πτήσης

Ο ελεγκτής πτήσης (flight controller) αποτελεί το κεντρικό υποσύστημα ελέγχου ενός drone, καθώς είναι υπεύθυνο για τη σταθεροποίηση του σκάφους, την επεξεργασία των δεδομένων που λαμβάνονται από τους αισθητήρες και την εκτέλεση των εντολών πτήσης. Ο ρόλος του είναι κρίσιμος, καθώς συγκεντρώνει και συντονίζει τη λειτουργία όλων των επιμέρους υποσυστημάτων διασφαλίζοντας την ασφαλή και ελεγχόμενη πτήση.

Στα αρχικά στάδια της υλοποίησης χρησιμοποιήθηκε ο ελεγκτής πτήσης APM 2.6 (ArduPilot Mega), ο οποίος ήταν διαθέσιμος για τις πρώτες δοκιμές του συστήματος. Κατά τη διαδικασία αρχικής αξιολόγησης και δοκιμής του διαπιστώθηκε η τεχνολογική του παλαιότητα, καθώς και η περιορισμένη επεξεργαστική ισχύς που προκύπτει από την αρχιτεκτονική 8-bit, σε συνδυασμό με τη μικρή διαθέσιμη μνήμη των 8 KB (kilobyte). Τα χαρακτηριστικά αυτά αποδείχθηκαν ανεπαρκή για τις απαιτήσεις της εφαρμογής. Ως αποτέλεσμα ο APM 2.6 δεν μπορούσε να υποστηρίξει την ομαλή εκτέλεση της πτήσης και τη διατήρηση επικοινωνίας υψηλού ρυθμού με εξωτερικά υποσυστήματα, όπως το Raspberry Pi. Για τον λόγο αυτό κρίθηκε αναγκαία η μετάβαση σε νεότερη πλατφόρμα ελεγκτή πτήσης.

Για την κάλυψη αυτών των απαιτήσεων επιλέχθηκε ο ελεγκτής πτήσης Pixhawk 2.4.8. Ο συγκεκριμένος ελεγκτής βασίζεται σε αρχιτεκτονική 32-bit και διαθέτει αυξημένη υπολογιστική ισχύ και μνήμη 256 KB, ενώ υποστηρίζεται πλήρως από λογισμικά ανοιχτού κώδικα όπως το ArduPilot και το PX4, αποτελώντας μια αξιόπιστη και ευρέως χρησιμοποιημένη λύση για συστήματα UAV.

Ο Pixhawk 2.4.8 ενσωματώνει αδρανειακή μονάδα μέτρησης (IMU) η οποία περιλαμβάνει γυροσκόπιο και επιταχυνσιόμετρο για τον προσδιορισμό της θέσης και κλίσης του σκάφους σε πραγματικό χρόνο. Επιπλέον, διαθέτει βαρόμετρο για την εκτίμηση του υψομέτρου, καθώς και μαγνητόμετρο για τον προσανατολισμό ως προς τον βορρά. Ο συνδυασμός αυτών των αισθητήρων συμβάλλει στη σταθεροποίηση και στην πλοήγηση του UAV.

Η λειτουργία του Pixhawk βασίζεται σε έλεγχο με ανάδραση. Ο ελεγκτής επεξεργάζεται τα δεδομένα που λαμβάνει από τους αισθητήρες και υπολογίζει διορθωτικές εντολές προς τους κινητήρες εξασφαλίζοντας σταθερή αιώρηση και ομαλή πτήση ακόμη και όταν επηρεάζεται από εξωτερικούς παράγοντες όπως ο άνεμος.

Παράλληλα, ο Pixhawk λειτουργεί ως το κεντρικό σημείο εισόδου των εντολών χειρισμού. Ο δέκτης τηλεχειρισμού συνδέεται απευθείας στον ελεγκτή και μεταφέρει τις εντολές του χρήστη. Στη συνέχεια ο Pixhawk υπολογίζει τα κατάλληλα σήματα προς τους ESC επιτρέποντας τον ελεγχόμενο χειρισμό των κινητήρων.

Για την εκτέλεση πιο σύνθετων λειτουργειών ο ελεγκτής πτήσης συνεργάζεται με εξωτερικά υποσυστήματα μέσω των θυρών επέκτασης που διαθέτει η διάταξη των οποίων απεικονίζεται στο σχήμα 3.8. Συγκεκριμένα παρέχονται θύρες GPS (Global Positioning System) για τη λήψη δεδομένων θέσης. Ακόμα, υπάρχουν θύρες τηλεμετρίας (telemetry) οι οποίες χρησιμοποιούνται για τη σειριακή επικοινωνία του ελεγκτή πτήσης με εξωτερικά συστήματα. Επιπλέον ο Pixhawk διαθέτει και θύρες AUX (Auxiliary), οι οποίες επιτρέπουν τη σύνδεση πρόσθετων περιφερειακών όπως σερβοκινητήρες.

Η επικοινωνία μέσω των θυρών τηλεμετρίας βασίζεται στο πρωτόκολλο MAVLink (Micro Air Vehicle Link) το οποίο επιτρέπει την ανταλλαγή εντολών και δεδομένων σε πραγματικό χρόνο. Μέσω του MAVLink καθίσταται δυνατή η αποστολή πληροφοριών κατάστασης όπως θέση, ύψος, ταχύτητα και τάση μπαταρίας. Επίσης μέσω του MAVLink δίνεται η δυνατότητα λήψης εντολών υψηλού επιπέδου από εξωτερικά υπολογιστικά συστήματα όπως το Raspberry Pi.

Συνολικά ο Pixhawk αποτέλεσε μια αξιόπιστη και αποδοτική επιλογή για την παρούσα υλοποίηση. Καλύπτοντας πλήρως τις απαιτήσεις του συστήματος ως προς το έλεγχο πτήσης, την επικοινωνία με εξωτερικά υποσυστήματα και υποστήριξη προηγμένων λειτουργειών αυτονομίας.

Σχήμα 3.8: Θύρες ελεγκτή πτήσης Pixhawk [26]

3.2.8 Μονάδα GPS M8N με Πυξίδα

Για τις ανάγκες πλοήγησης και εντοπισμού θέσης του UAV επιλέχθηκε η χρήση της εξωτερικής μονάδας M8N GPS Module με πυξίδα, η οποία συνεργάζεται άμεσα με τον ελεγκτή πτήσης Pixhawk 2.4.8. Αν και ο Pixhawk διαθέτει ενσωματωμένη πυξίδα στην παρούσα υλοποίηση προτιμήθηκε η χρήση εξωτερικής πυξίδας, ώστε να επιτευχθεί μεγαλύτερη ακρίβεια.

Η μονάδα GPS χρησιμοποιείται για τον προσδιορισμό της γεωγραφικής θέσης, της ταχύτητας και της πορείας του UAV. Τα δεδομένα αυτά είναι κρίσιμα για λειτουργίες όπως η διατήρηση θέσης και η αυτόματη επιστροφή στο σημείο. Με τη χρήση GPS, ο ελεγκτής πτήσης απόκτα πλήρη εικόνα της κίνησης του UAV στον χώρο γεγονός που συμβάλει στη βελτίωση της σταθερότητας και της συνολικής ασφάλειας της πτήσης.

Η εξωτερική πυξίδα μπορεί να τοποθετηθεί σε κατάλληλο σημείο του πλαισίου μακριά από κινητήρες, τα ESC και καλώδια τροφοδοσίας, μειώνοντας σημαντικά τις ηλεκτρομαγνητικές παρεμβολές που επηρεάζουν τις μετρήσεις. Με αυτόν τον τρόπο επιτυγχάνεται πιο αξιόπιστος υπολογισμός του προσανατολισμού του UAV.

Η σύνδεση της μονάδας με τον Pixhawk πραγματοποιείται μέσω δύο ξεχωριστών συνδέσεων όπως μπορούμε να διακρίνουμε και στο σχήμα 3.9. Το GPS συνδέεται στη θύρα GPS του ελεγκτή πτήσης μέσω σειριακής επικοινωνίας (UART) και μεταφέρει δεδομένα θέσης και ταχύτητας. Παράλληλα, το ενσωματωμένο μαγνητόμετρο συνδέεται μέσω της θύρας I2C (Inter integrated circuit), επιτρέποντας την ακριβή μέτρηση του προσανατολισμού εξασφαλίζοντας αξιόπιστη λειτουργία και εύκολη ενσωμάτωση στο σύστημα πτήσης.

Συνολικά η επιλογή του M8N GPS Module με πυξίδα προσφέρει βελτιωμένη ακρίβεια πλοήγησης αξιόπιστο προσανατολισμό και ομαλή συνεργασία με τον ελεγκτή πτήσης καλύπτοντας τις απαιτήσεις της παρούσας διπλωματικής εργασίας χωρίς να αυξάνει σημαντικά το βάρος ή την πολυπλοκότητα του συστήματος.

Σχήμα 3.9: Κεραία GPS και ηλεκτρονική πυξίδα [27]

3.2.9 Πομποδέκτης Χειρισμού

Το σύστημα πομπού δέκτη (RF τηλεχειρισμός) επιτρέπει την ασύρματη μετάδοση εντολών από τον χειριστή προς τον ελεγκτή πτήσης, επιτρέποντας την άμεση αλληλεπίδραση του χρήστη με το UAV κατά τη διάρκεια της πτήσης. Η ασύρματη επικοινωνία εξασφαλίζει ότι τα σήματα ελέγχου μεταδίδονται με αξιοπιστία και χωρίς καθυστέρησης, κάτι που είναι κρίσιμο για την ασφαλή και ακριβή καθοδήγηση του αεροσκάφους.

Στην παρούσα υλοποίηση χρησιμοποιήθηκε ο πομπός FlySky FS-i6X σε συνδυασμό με τον δέκτη iA6B όπως απεικονίζεται στο σχήμα 3.10. Οι συγκεκριμένες μονάδες λειτουργούν στην ζώνη συχνοτήτων 2.4 Ghz. Η συχνότητα αυτή είναι ιδιαίτερα διαδεδομένη σε εφαρμογές τηλεκατεύθυνσης drone, καθώς προσφέρει ικανοποιητική εμβέλεια και αντοχή σε παρεμβολές. Το σύστημα υποστηρίζει εμβέλεια χειρισμού που μπορεί να φτάσει έως περίπου 1 km σε συνθήκες απευθείας οπτικής επαφής, τιμή που θεωρείται επαρκής για τις ανάγκες της παρούσας κατασκευής.

Ένα σημαντικό τεχνικό χαρακτηριστικό του συστήματος είναι η χρήση του πρωτοκόλλου AFHDS 2A (Automatic Frequency Hopping Digital System). Η τεχνολογία αυτή επιτρέπει στο πομπό να αλλάζει συνεχώς τη συχνότητες εκπομπής ανάμεσα σε πολλά κανάλια μέσα στη ζώνη των 2.4Ghz. Με αυτόν τον τρόπο μειώνονται οι παρεμβολές από άλλες συσκευές που λειτουργούν στην ίδια ζώνη συχνοτήτων.

Ο πομπός FlySky FS-i6X διαθέτει έξι κανάλια ελέγχου (6 CH) τα οποία αντιστοιχούν στις βασικές εντολές πτήσης (Roll, Pitch, Yaw και Throttle) καθώς και σε βοηθητικές λειτουργίες που ενεργοποιούνται μέσω διακοπτών. Οι κινήσεις των μοχλών και των διακοπτών μετατρέπονται σε ψηφιακά σήματα, τα οποία μεταδίδονται ασύρματα προς τον δέκτη.

Ο δέκτης iA6B είναι τοποθετημένος πάνω στο UAV και αναλαμβάνει τη λήψη και αποκωδικοποίηση των σημάτων που στέλνει ο πομπός. Τα δεδομένα που έρχονται από το πομπό προωθούνται στον ελεγκτή πτήσης μέσω του ψηφιακού καναλιού iBus. Το κανάλι αυτό προσφέρει ταχύτερη και αξιόπιστη μετάδοση σε σχέση με τις αναλογικές εξόδους PWM (Pulse Width Modulation).

Ακόμη, το σύστημα υποστηρίζει λειτουργία τηλεμετρίας, επιτρέποντας την επιστροφή βασικών πληροφοριών από το αεροσκάφος προς τον πομπό, όπως η τάση της μπαταρίας ή η κατάσταση σύνδεσης. Η δυνατότητα αυτή επιτρέπει στον χειριστή να έχει πλήρη εικόνα της κατάστασης του συστήματος.

Συνολικά ο συνδυασμό του πομπού FlySky FS-i6X με τον δέκτη iA6B επιλέχθηκε επειδή προσφέρει αξιόπιστη ραδιοεπικοινωνία, επαρκή αριθμό καναλιών, χαμηλή καθυστέρηση και καλή συμβατότητα με τον ελεγκτή πτήσης.

Σχήμα 3.10: Πομποδέκτης Flysky FS-i6x & FS-iA6B [28]

3.2.10 Τηλεμετρία Τάσης

Ο FlySky FS-CVT01 αποτελεί μονάδα τηλεμετρίας τάσης η οποία απεικονίζεται στο σχήμα 3.11, και χρησιμοποιείται στο UAV για την παρακολούθηση της κατάστασης της μπαταρίας σε πραγματικό χρόνο. Ο βασικός του ρόλος είναι να μετρά την τάση τροφοδοσίας του συστήματος και να μεταδίδει την πληροφορία αυτή ασύρματα προς τον πομπό του χειριστή, μέσω του δέκτη FlySky iA6B που είναι εγκατεστημένος στο drone. Με τον τρόπο αυτό ο χειριστής έχει άμεση εικόνα της ενεργειακής κατάστασης του UAV κατά τη διάρκεια της πτήσης.

Η λειτουργία του FS-CVT01 βασίζεται στη σύνδεσή του στη γραμμή τροφοδοσίας της μπαταρίας. Η μονάδα μετρά συνεχώς την τάση και τη μετατρέπει σε ψηφιακή πληροφορία, η οποία αποστέλλεται στον δέκτη μέσω του πρωτοκόλλου τηλεμετρίας της FlySky. Η σύνδεση με τον δέκτη γίνεται μέσω του καλωδίου των τριών αγωγών, το οποίο παρέχει τροφοδοσία στη μονάδα και μεταφέρει πίσω την ένδειξη της τάσης ώστε να εμφανίζεται στον πομπό.

Η συγκεκριμένη μονάδα επιλέχθηκε επειδή είναι πλήρως συμβατή με το σύστημα πομποδέκτη FlySky που χρησιμοποιήθηκε, διαθέτει απλή συνδεσμολογία και δεν απαιτεί πολύπλοκες ρυθμίσεις. Επιπλέον, λειτουργεί ανεξάρτητα από τον ελεγκτή πτήσης, προσφέροντας ένα επιπλέον επίπεδο ασφάλειας για την μπαταρία.

Σχήμα 3.11: Μονάδα Τηλεμετρίας Τάσης FlySky FS-CVT01 [29]

3.2.11 Raspberry Pi

Το Raspberry Pi χρησιμοποιήθηκε στο παρόν σύστημα ως υπολογιστική μονάδα υψηλού επιπέδου με σκοπό την υποστήριξη λειτουργιών που δεν μπορούν να υλοποιηθούν από έναν ελεγκτή πτήσης. Σε ένα UAV ο ελεγκτή πτήσης αναλαμβάνει τον άμεσο έλεγχο της πτήσης και τη σταθεροποίηση του σκάφους. Ωστόσο δεν είναι σχεδιασμένο για απαιτητικές υπολογιστικές διεργασίες όπως επεξεργασία εικόνας συνδυασμό δεδομένων αισθητήρων ή αλγορίθμων υψηλού επιπέδου. Για τον λόγο αυτό κρίθηκε απαραίτητο η ενσωμάτωση ενός βοηθητικού υπολογιστή.

Το Raspberry Pi ανήκει στην οικογένεια των single board υπολογιστών. Χαρακτηρίζεται από υψηλή υπολογιστική ισχύ, χαμηλή κατανάλωση ενέργειας και μικρές διαστάσεις χαρακτηριστικά που καθιστούν ιδιαίτερα κατάλληλο για ενσωματωμένα συστήματα και εφαρμογές UAV. Επιπλέον, υποστηρίζει πληθώρα περιφερειακών και αισθητήρων επιτρέποντας την εύκολη επέκταση των δυνατοτήτων του συστήματος.

Για την παρούσα υλοποίηση επιλέχθηκε το Raspberry Pi 5 το οποίο αποτελεί την πιο σύγχρονη και ισχυρή έκδοση της σειράς. Το συγκεκριμένο μοντέλο διατίθεται σε εκδόσεις των 2, 4, 8 και 16GB (Gigabyte) μνήμης RAM (Random access memory). Στο σύστημα χρησιμοποιήθηκε η έκδοση των 8GB, καθώς η αυξημένη διαθέσιμη μνήμη είναι απαραίτητη για την ταυτόχρονη χρήση κάμερας, την επεξεργασία εικόνας, την συλλογή δεδομένων από αισθητήρες και την σταθερή εκτέλεση πολλαπλών διεργασιών χωρίς καθυστερήσεις.

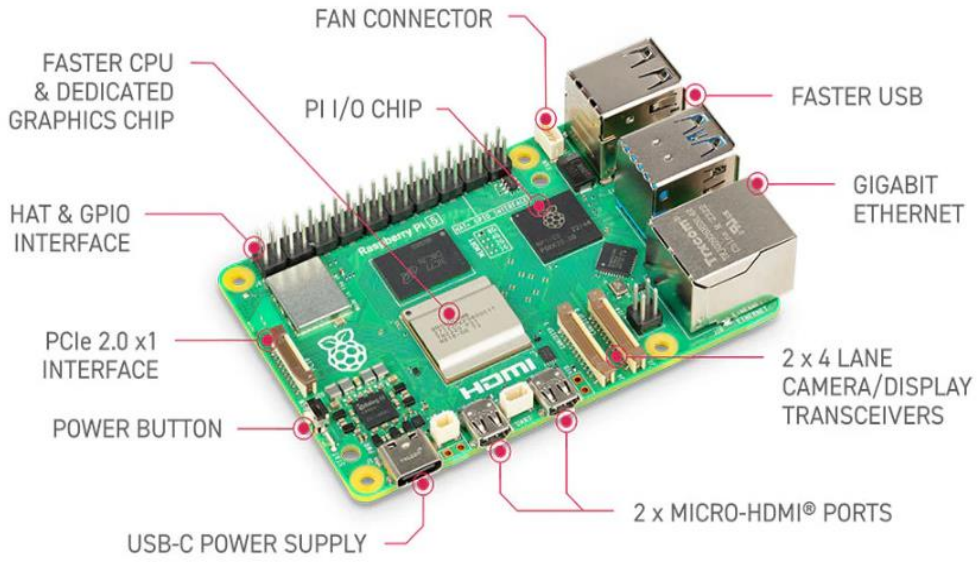
Ο κεντρικός επεξεργαστής του Raspberry Pi 5 βασίζεται σε αρχιτεκτονική ARM 64-bit και διαθέτει τέσσερις πυρήνες στα 2.4GHz. Παράλληλά η ενσωματωμένη μονάδα γραφικών λειτουργεί στα 800 MHz (Megahertz) και υποστηρίζει OpenGL (Open Graphics Library) καθώς και αναπαραγωγή και επεξεργασία βίντεο ανάλυσης έως 4K, διευκολύνοντας έτσι εφαρμογές που βασίζονται σε οπτικά δεδομένα.

Ένα από τα βασικά πλεονεκτήματα του Raspberry Pi 5 είναι ότι υποστηρίζει σύγχρονα λειτουργικά συστήματα βασισμένα στο Linux. Η ύπαρξη ενός πλήρους λειτουργικού συστήματος επιτρέπει την ανάπτυξη και εκτέλεση απαιτητικών εφαρμογών, τη χρήση βιβλιοθηκών υπολογιστικής όρασης και την ομαλή διαχείριση πολλαπλών διεργασιών. Παράλληλα προσφέρει δυνατότητες όπως απομακρυσμένη πρόσβαση και διαχείριση αρχείων.

Το Raspberry Pi 5 διαθέτει πληθώρα διεπαφών και θυρών, όπως απεικονίζεται αναλυτικά στο σχήμα 3.12, που το καθιστούν ιδιαίτερα ευέλικτο.

Συγκεκριμένα περιλαμβάνει:

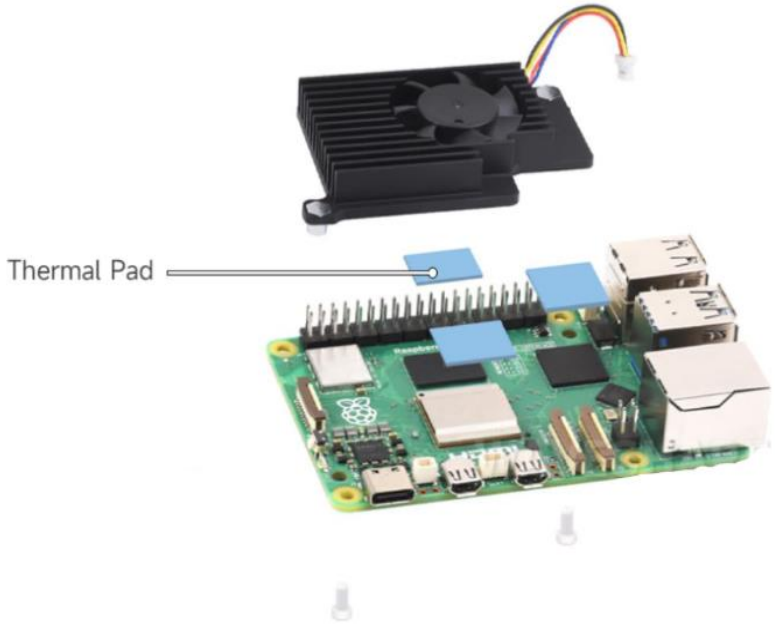
1. Υποδοχή κάρτας micro SD η οποία χρησιμοποιείται ως βασικό μέσο αποθήκευσης. Υποστηρίζει κάρτες χωρητικότητας 32GB, 64GB και 128GB ανάλογα με τις απαιτήσεις του συστήματος.
2. Δυνατότητα σύνδεσης συστήματος ψύξης μέσω ειδικής υποδοχής για ψήκτρα και heatsink, με σκοπό τη διαχείριση της θερμότητας που παράγεται κατά τη λειτουργία της συσκευής.
3. Επιλογή σύνδεσης στο δίκτυο είτε μέσω θύρας Ethernet είτε μέσω WiFi. Υποστηρίζει WiFi 5ης γενιάς (πρότυπο 802.11ac) διπλής ζώνης συχνοτήτων 2.4 και 5GHz.
4. Ενσωματωμένη μονάδα Bluetooth έκδοσης 5.0 για ασύρματη επικοινωνία με συμβατές συσκευές.
5. Θύρες CSI (Camera Serial Interface), οι οποίες επιτρέπουν τη σύνδεση καμερών απευθείας στη συσκευή προσφέροντας υψηλές ταχύτητές μεταφοράς δεδομένων.
6. Θύρες USB, συγκεκριμένα δύο USB 2.0 και δύο USB 3.0, εκ των οποίων οι εκδόσεις 3.0 υποστηρίζουν μεταφορά δεδομένων έως και 5Gbps (Gigabits per second).
7. Υποδοχή GPIO (General purpose Input/Output) 40 ακίδων για σύνδεση αισθητήρων και βοηθητικών κυκλωμάτων.



Σχήμα 3.12: Τα κυριά μέρη και οι θύρες σύνδεσης του Raspbery Pi 5 [30]

Το Raspberry Pi δεν διαθέτει ενσωματωμένο αποθηκευτικό χώρο. Για αυτό τον λόγο το λειτουργικό σύστημα, οι εφαρμογές και τα δεδομένα του συστήματος αποθηκεύονται σε εξωτερικό μέσο μέσω της υποδοχής κάρτας microSD. Στο παρόν σύστημα χρησιμοποιήθηκε κάρτα microSD χωρητικότητας 128GB, η οποία προσφέρει αρκετό χώρο για την ομαλή λειτουργία του συστήματος. Η επιλογή κάρτας τόσο μεγάλης χωρητικότητας όσο και υψηλής ταχύτητας ανάγνωσης και εγγραφής κρίθηκε απαραίτητη, καθώς στο σύστημα εκτελούνται απαιτητικές διεργασίες. Σε αυτές περιλαμβάνονται το λειτουργικά συστήματα, επεξεργασία εικόνα από κάμερα, η διαχείριση δεδομένων αισθητήρων και η αποθήκευση αρχείων. Οι αυξημένες ταχύτητες της κάρτας εξασφαλίζουν αξιόπιστη λειτουργία, ομαλή πρόσβαση στα δεδομένα και αποφυγή καθυστερήσεων.

Λόγω της αυξημένες επεξεργαστικής ισχύος του Raspberry Pi 5, η θερμική διαχείριση αποτελεί κρίσιμο παράγοντα για τη σωστή λειτουργία του. Για τον λόγο αυτό, χρησιμοποιήθηκε το σύστημα ενεργής ψύξης, το οποίο συνδυάζει ψύκτρα και ανεμιστήρα. Η εγκατάστασή του απαιτεί την τοποθέτηση ειδικών θερμικών επιθεμάτων (thermal pads) στα κρίσιμα εξαρτήματα της πλακέτας για τη βέλτιστη απαγωγή της θερμότητας, όπως φαίνεται αναλυτικά στο σχήμα 3.13.

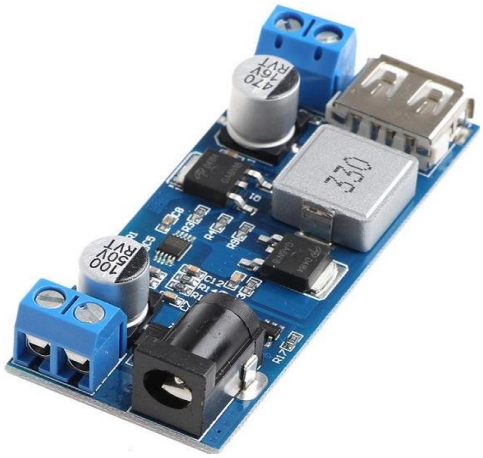


Σχήμα 3.13: Τοποθέτηση του συστήματος ψύξης στο Raspberry Pi 5 [31]

3.2.12 Μετατροπέας DC-DC

Ο μετατροπέας τάσης DC-DC step-down όπως φαίνεται και στο σχήμα 3.14 χρησιμοποιήθηκε στο σύστημα για τη σταθερή και ασφαλή τροφοδοσία του Raspberry Pi από την κύρια μπαταρία του UAV. Η μπαταρία Li-Po παρέχει τάση 14.8V, η οποία είναι ακατάλληλη για απευθείας σύνδεση με το Raspberry Pi, καθώς αυτό απαιτεί τάση περίπου 5V για σωστή λειτουργία. Ο συγκεκριμένος μετατροπέας δέχεται εύρος τάσης εισόδου 9–36V και αποδίδει σταθερή τάση εξόδου 5.2V, εξασφαλίζοντας αξιόπιστη τροφοδοσία ακόμα και όταν η τάσης της μπαταρίας μεταβάλλεται.

Η επιλογή του συγκεκριμένου DC-DC μετατροπέα έγινε λόγω της ικανότητάς του να παρέχει επαρκές ρεύμα έως 6A, καλύπτοντας τις αυξημένες ενεργειακές απαιτήσεις του Raspberry Pi και των συνδεδεμένων περιφερειακών του.



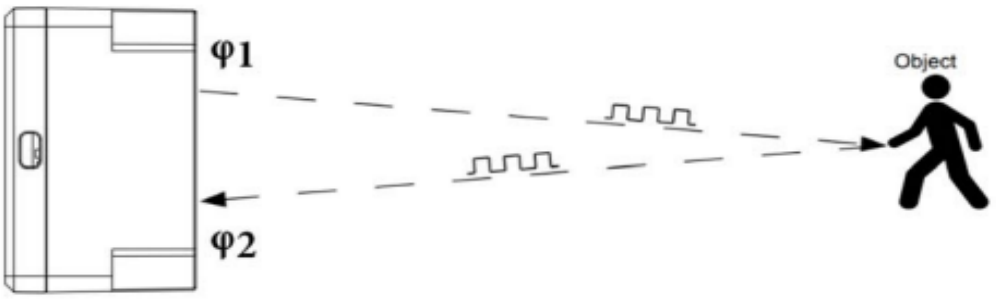
Σχήμα 3.14: Μετατροπέας τάσης DC-DC Step down

3.2.13 Αισθητήρας LiDAR

Ο αισθητήρας LiDAR (Light Detection and Ranging) αποτελεί βασικό υποσύστημα αντίληψης του περιβάλλοντος, καθώς επιτρέπει την ακριβή μέτρηση αποστάσεων από εμπόδια. Στο παρόν σύστημα χρησιμοποιήθηκε ο Benewake TFmini Plus, όπως απεικονίζεται στο σχήμα 3.15, ένας αισθητήρας μικρού μεγέθους και βάρους κατάλληλος για εφαρμογές drone και ρομποτικής.

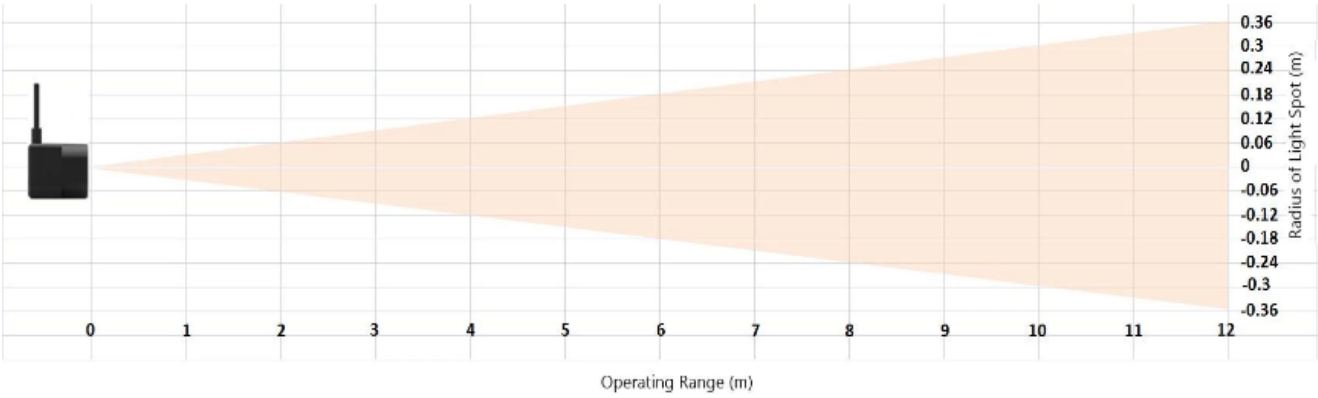
Σχήμα 3.15: Αισθητήρας LiDAR Benewake TFmini Plus [32]

Ο TFmini Plus βασίζεται στην τεχνολογία ToF (Time of Flight). Η αρχή λειτουργίας του, όπως αναπαρίσταται στο σχήμα 3.16, στηρίζεται στην εκπομπή υπέρυθρων παλμών και στη μέτρηση του χρόνου επιστροφής τους μετά την ανάκλαση σε κάποιο εμπόδιο. Γνωρίζοντας την ταχύτητα διάδοσης του φωτός και τον χρόνο που χρειάζεται ο παλμός για να επιστρέψει, ο αισθητήρας υπολογίζει με ακρίβεια την απόσταση από το αντικείμενο.



Σχήμα 3.16: Απεικόνισή της αρχής λειτουργίας ToF [32]

Ο συγκεκριμένος ο αισθητήρας χρησιμοποιεί υπέρυθρη ακτίνα λέιζερ και διαθέτει γωνία δέσμης (FoV) 3.6 μοίρες, κάτι που του επιτρέπει να πραγματοποίει στοχευμένες μετρήσεις. Οι αισθητήρες που βασίζονται σε τεχνολογία λέιζερ προσφέρουν υψηλή ακρίβεια στις μετρήσεις. Παράλληλά επηρεάζονται ελάχιστα από μεταβολές φωτισμού ή άλλους εξωτερικούς παράγοντες κάτι που τους καθιστά κατάλληλους για χρήση σε εξωτερικά περιβάλλοντα. Το εύρος μέτρησης του TFmini Plus κυμαίνεται από 0.1 έως 12 m, προσφέροντας σταθερή και αξιόπιστη απόδοση στο διαθέσιμο εύρος λειτουργίας του. Η σχέση ανάμεσα στην απόσταση και το εύρος της δέσμης φαίνεται στο σχήμα 3.17, όπου διακρίνεται ότι η περιοχή στόχευσης μεγαλώνει όσο αυξάνεται η απόσταση από το αντικείμενο. Επιπλέον ο αισθητήρας υποστηρίζει ρυθμιζόμενο ρυθμό ανανέωσης από 1 έως 1000 Hz προσφέροντας υψηλής συχνότητας μετρήσεις και άμεση αντίδραση του συστήματος σε μεταβολές του περιβάλλοντος.



Σχήμα 3.17: Περιοχή ανίχνευσής του TFmini Plus σε συνάρτηση με την απόσταση [32]

Ο TFmini Plus αποτελείται από βασικά υποσυστήματα τα οποία συνεργάζονται για να πραγματοποιηθεί η μέτρηση της απόστασης:

1. Τον πομπό λέιζερ για την εκπομπή των παλμών υπέρυθρης ακτινοβολίας προς το περιβάλλον.
2. Τον δέκτη φωτοδιόδου για την ανίχνευση της ανακλώμενης ακτινοβολίας.
3. Τον μικροελεγκτή για τον υπολογισμό του χρόνου πτήσης (ToF) του παλμού και την επεξεργασία των μετρήσεων.

Μόλις γίνει η λήψη του ανακλώμενου παλμού, το εσωτερικό κύκλωμα του αισθητήρα επεξεργάζεται το σήμα και το μετατρέπει σε ψηφιακή πληροφορία απόστασης. Τα δεδομένα αυτά αποστέλλονται προς το υπολογιστικό σύστημα, όπως για παράδειγμα σε έναν μικροελεγκτή ή στο Raspberry Pi, μέσω της σειριακής επικοινωνίας UART (Universal Asynchronous Receiver/Transmitter).

Σε αντίθεση με περιστρεφόμενα LiDAR, ο συγκεκριμένος αισθητήρας δεν διαθέτει κινούμενα μηχανικά μέρη, γεγονός που απλοποιεί τη λειτουργία στοιχείο που τον καθιστά απλό και ανθεκτικό στην χρήση.

Η τροφοδοσία του TFmini Plus γίνεται με σταθερή τάση 5V , ενώ το ρεύμα λειτουργίας του είναι περίπου 110mA (milliampere) τιμή κατάλληλη για ενσωματωμένα συστήματα με περιορισμένη ενεργειακή διαθεσιμότητα. Ο αισθητήρας λειτουργεί αξιόπιστα σε θερμοκρασίες από -20ºC έως +60ºC κάτι που τον καθιστά κατάλληλο για χρήση σε εξωτερικά περιβάλλοντα. Ακόμα, διαθέτει βαθμό προστασίας IP65, που τον καθιστά ανθεκτικό στη σκόνη και στο νερό.

Η επιλογή του συγκεκριμένου αισθητήρα έγινε λόγω του μικρού βάρους, της χαμηλής κατανάλωσης της απλής διασύνδεσης μέσω UART και της ακρίβειας που προσφέρει για εφαρμογές UAV.

Τα βασικά τεχνικά χαρακτηριστικά του Benewake TFmini Plus συνοψίζονται στον πίνακα 3.4 που ακολουθεί.

Πίνακας 3.4: Τεχνικά χαρακτηριστικά TFmini Plus

Εύρος μέτρησης	0.1 m-12 m
Συχνότητα ενημερώσεις	1-1000Hz
Ταση τροφοδοσίας	5V DC
Πρωτόκολλο επικοινωνίας	UART
Γωνιά Δέσμης	3.6ο
Κατανάλωση ρεύματος	110 mA
Διαστάσεις	35mm*18.5mm*21mm
Βάρος	11g

3.2.14 Κάμερα

Η κάμερα αποτελεί ένα σημαντικό αισθητήριο μέσο του UAV καθώς του δίνει τη δυνατότητα να συλλέγει οπτικά δεδομένα από το περιβάλλον. Μέσω αυτής το σύστημα μπορεί να καταγράφει εικόνες και βίντεο και να μετατρέπει την οπτική πληροφορία σε ψηφιακά δεδομένα. Τα δεδομένα αυτά μπορούν να αποθηκευτούν ή να υποβληθούν σε περαιτέρω επεξεργασία από το υπολογιστικό σύστημα συμβάλλοντας στη συνολική αντίληψη του περιβάλλοντος κατά την πτήση.

Στο παρόν σύστημα χρησιμοποιήθηκε η Raspberry Pi camera Module V2, όπως απεικονίζεται στο σχήμα 3.18, η οποία ενσωματώνεται απευθείας στο Raspberry Pi και έχει σχεδιαστεί για εφαρμογές ενσωματωμένων συστημάτων. Λόγω των μικρών διαστάσεων και του χαμηλού βάρους της αποτελεί κατάλληλη επιλογή για εφαρμογές όπου το βάρος παίζει καθοριστικό ρόλο όπως ένα drone. Επιπλέον, επιλέχθηκε διότι η σύνδεση μέσω της διεπαφής CSI προσφέρει ταχύτερη και πιο αποδοτική μεταφορά δεδομένων σε σχέση με κάμερες USB. Στο συγκεκριμένο drone η κάμερα αξιοποιείται για την παρατήρηση του περιβάλλοντος και τη συλλογή οπτικών δεδομένων κατά τη διάρκεια της πτήσης. Μέσω αυτής το σύστημα μπορεί να εντοπίζει και να αναγνωρίζει αντικείμενα εντός οπτικού πεδίου όπως ανθρώπους παρέχοντας πληροφορίες που δεν μπορούν να προκύψουν από άλλους αισθητήρες.

Σχήμα 3.18: Raspberry Pi Camera Module V2 [33]

Τα βασικά τεχνικά χαρακτηριστικά της κάμερας που συνέβαλαν στην επιλογή της συνοψίζονται παρακάτω:

1. Ανάλυση αισθητήρα 8 Megapixel με δυνατότητα καταγραφής βίντεο 1080p.
2. Οπτικό πεδίο 73.8 μοίρες για την καταγραφή του περιβάλλοντα χώρου μπροστά από το μη επανδρωμένο αεροσκάφος.
3. Σύνδεση μέσω διεπαφής CSI (Camera Serial Interface) απευθείας στο Raspberry Pi εξασφαλίζοντας γρήγορη και αποδοτική μεταφορά δεδομένων εικόνας.

Λόγω της αυξημένης ευαισθησίας της κάμερας και της έκθεσης της σε εξωτερικές συνθήκες κρίθηκε απαραίτητη η χρήση προστατευτικής θήκης, η οποία παρουσιάζεται στο σχήμα 3.19. Η θήκη προστατεύει την κάμερα από μηχανικές καταπονήσεις σκόνη και άλλους περιβαλλοντικούς παράγοντες. Έτσι εξασφαλίζεται η σωστή λειτουργία της κάμερας και η αξιοπιστία του συστήματος σε βάθος χρόνου.

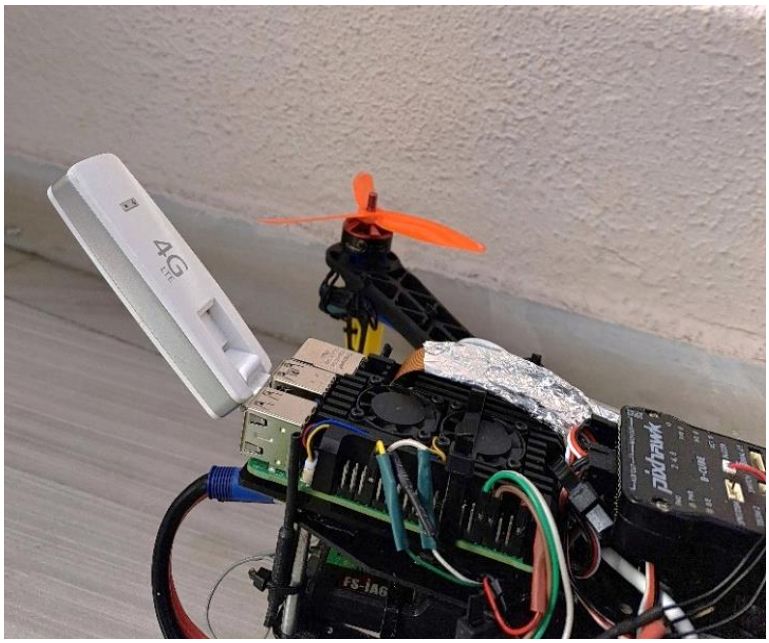
Σχήμα 3.19: Προστατευτική θήκη για την κάμερα Raspberry Pi V2 [34]

Η επιλογή της συγκεκριμένης κάμερας καλύπτει πλήρως τις ανάγκες του UAV ως προς την καταγραφή και ανάλυση οπτικών δεδομένων ενισχύοντας τις δυνατότητες αντίληψης και λειτουργικότητας του συστήματος.

3.2.15 Εξωτερικός Δρομολογητής

Για τις ανάγκες ασύρματης επικοινωνίας του συστήματος ενσωματώθηκε ένας εξωτερικός δρομολογητής USB, ο οποίος συνδέεται απευθείας στο Raspberry Pi όπως απεικονίζεται στο σχήμα 3.20. Η επιλογή αυτή έγινε επειδή η ενσωματωμένη μονάδα Wi-Fi του Raspberry Pi παρουσιάζει

περιορισμένη εμβέλεια και δεν προσφέρει σταθερή σύνδεση σε απαιτητικά περιβάλλοντα. Σε εφαρμογές όπου χρειάζεται αξιόπιστη μετάδοση δεδομένων σε μεγαλύτερες αποστάσεις η απόδοσή της δεν επαρκεί.



Σχήμα 3.20: 4G USB Router συνδεδεμένο στο Raspberry Pi

Ο βασικός ρόλος του router είναι η δημιουργία ενός τοπικού ασυρμάτου δικτύου μέσω του οποίου είναι δυνατή η σύνδεση στο Raspberry Pi από εξωτερικές συσκευές όπως φορητός υπολογιστής. Η επικοινωνία πραγματοποιείται μέσω τοπικής διεύθυνσης IP επιτρέποντας την απομακρυσμένη πρόσβαση στο σύστημα και τη μετάδοση δεδομένων όπως εικόνας και βίντεο από την κάμερα καθώς και άλλων κρίσιμων πληροφοριών. Η χρήση του router συμβάλλει σε πιο σταθερή σύνδεση και μειώνει την πιθανότητα αποσυνδέσεων κατά τη λειτουργία του συστήματος.

3.3 Καλωδιώσεις και Συνδέσεις Εξαρτημάτων

Η καλωδίωση και η ηλεκτρική διασύνδεση του UAV πραγματοποιήθηκαν σε έτοιμο πλαίσιο στο οποίο τοποθετήθηκαν σταδιακά τα επιμέρους ηλεκτρονικά υποσυστήματα. Για την υλοποίηση των ηλεκτρικών συνδέσεων απαιτήθηκαν βασικές γνώσεις χρήσης ηλεκτρικού σταθμού κόλλησης, καθώς και απλά εργαλεία, όπως κατσαβίδια, μύτες Allen και δεματικά για τη στερέωση καλωδίων και εξαρτημάτων.

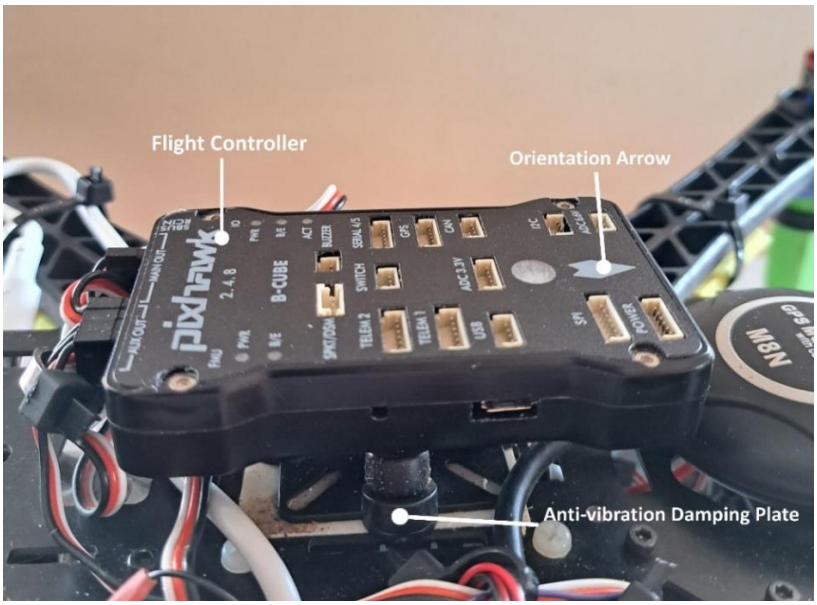
Στην παρούσα ενότητα παρουσιάζονται αναλυτικά οι συνδέσεις που πραγματοποιήθηκαν και αφορούν κυρίως την τροφοδοσία των υποσυστημάτων, τη σύνδεση των κινητήρων με τα ESC, την εγκατάσταση και διασύνδεση του ελεγκτή πτήσης και των αισθητήρων, καθώς και την ενσωμάτωση του Raspberry Pi στο σύστημα.

Στόχος της ενότητας είναι να αποτυπωθεί η συνολική εικόνα της κατασκευαστικής διαδικασίας και να αναδειχθεί ο τρόπος με τον οποίο τα επιμέρους υποσυστήματα συνεργάζονται μεταξύ τους για τη λειτουργία του UAV.

3.3.1 Τοποθέτηση Ελεγκτή Πτήσης και Υποσυστημάτων

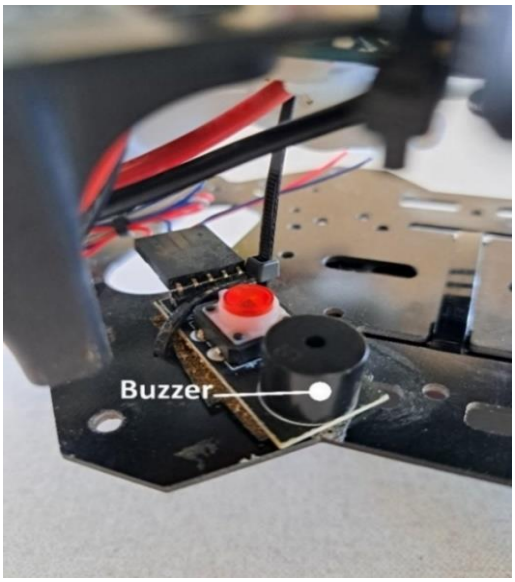
Στα αρχικά στάδια της υλοποίησης πραγματοποιήθηκε η τοποθέτηση του ελεγκτή πτήσης, ο οποίος εγκαταστάθηκε στο κέντρο του πάνω μέρους του πλαισίου. Η συγκεκριμένη θέση επιλέχθηκε για τη

βελτίωση μέτρησης των κινήσεων και τη σωστή λειτουργία των αισθητήρων. Ο flight controller τοποθετήθηκε με το βέλος προσανατολισμού προς το εμπρόσθιο μέρος του πλαισίου διασφαλίζοντας την σωστή αντιστοίχιση των αξόνων μέτρησης. Για την καλύτερη απόδοση των γυροσκοπίων και τον περιορισμό των κραδασμών, ο ελεγκτής πτήσης τοποθετήθηκε πάνω σε αντικραδασμική βάση (anti vibration damping plate), όπως φαίνεται και στο σχήμα 3.21.



Σχήμα 3.21: Τοποθέτηση και προσανατολισμός Pixhawk σε αντικραδασμική βάση

Στη συνέχεια, επειδή ο ελεγκτής πτήσης υποστηρίζει τη χρήση buzzer για ηχητικές ειδοποιήσεις σε περίπτωση σφάλματος, τοποθετήθηκε ένα buzzer στη σχετική θύρα του Pixhawk. Το buzzer τοποθετήθηκε ανάμεσα στα 2 επίπεδα του frame και στερεώθηκε με δεματικά ώστε να παραμείνει σταθερό και να μην μετακινηθεί κατά την πτήση, όπως φαίνεται στο σχήμα 3.22.

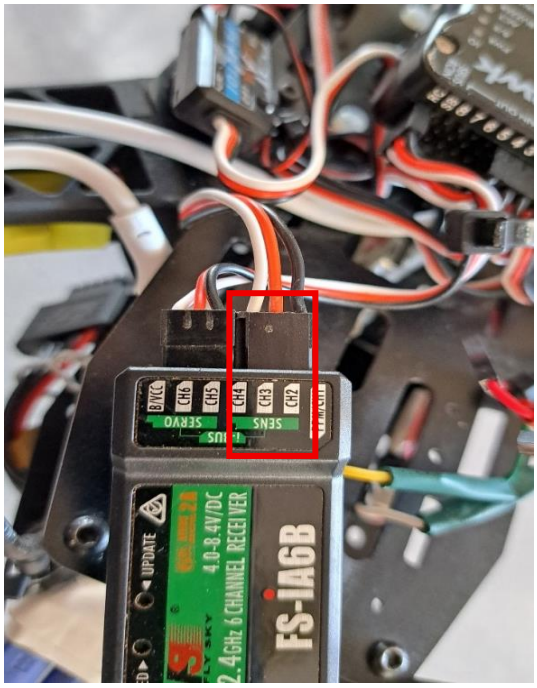


Σχήμα 3.22: Τοποθέτηση Buzzer στο πλαίσιο

Έπειτα, προστέθηκε ο μετρητής τάσης FlySky FS-CVT01 για την παρακολούθηση της τάσης της μπαταρίας. Η μία πλευρά της μονάδας συνδέθηκε στο PDB, ώστε να λαμβάνει απευθείας την τάση της μπαταρίας, όπως διακρίνεται και από το σχήμα 3.23, ενώ η άλλη πλευρά συνδέθηκε στη θύρα SENS

του δέκτη τηλεχειρισμού, όπως απεικονίζεται στο σχήμα 3.24 επιτρέποντας τη μετάδοση των δεδομένων τάσης προς το τηλεχειριστήριο.

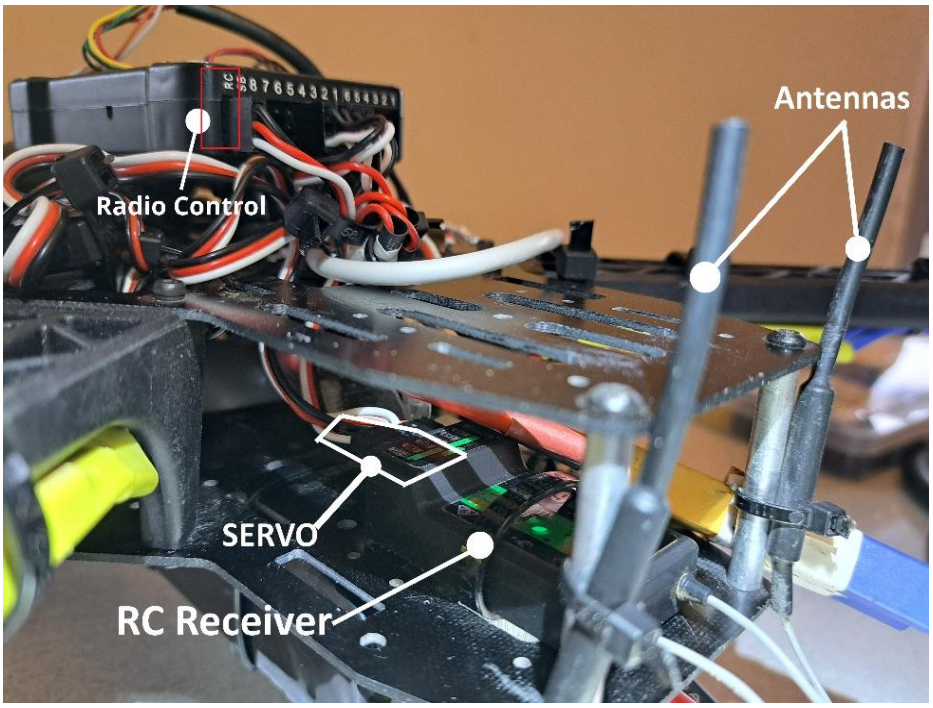
Σχήμα 3.23: Τοποθέτηση του αισθητήρα τάσης στο PDB



Σχήμα 3.24: Σύνδεση του αισθητήρα στη θύρα τηλεμετρίας του δέκτη

Έπειτα, ακολούθησε η σύνδεση του δέκτη (FS-iA6B) με τον ελεγκτή πτήσης. Η σύνδεση έγινε μέσω της θύρας RC (Radio Control) του ελεγκτή με το αντίστοιχο καλώδιο να καταλήγει στη θύρα servo του δέκτη. Ο δέκτης τοποθετήθηκε ανάμεσα στα δύο επίπεδα του πλαισίου και στερεώθηκε με δεματικά.

Οι κεραίες του στερεώθηκαν κατά μήκος των κατακόρυφων στηριγμάτων του πλαισίου ώστε να εξασφαλίζεται καλή λήψη σήματος και να αποφεύγονται παρεμβολές. Η συνολική διάταξη παρουσιάζεται σχήμα 3.25.

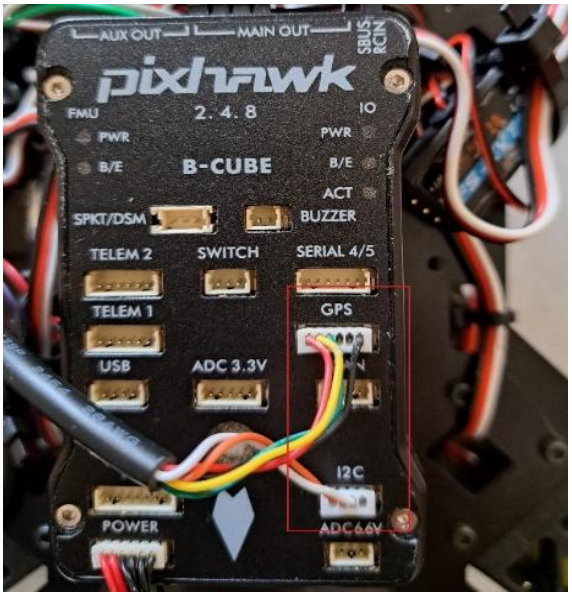


Σχήμα 3.25: Εγκατάσταση δέκτη και κεραιών τηλεχειρισμού στο drone

Μετά την τοποθέτηση του δέκτη τοποθετήθηκε και η κεραία GPS με την ενσωματωμένη πυξίδα. Η κεραία GPS εγκαταστάθηκε στο εμπρόσθιο τμήμα του επάνω μέρους του πλαισίου σε θέση απομακρυσμένη από τα ESC και τα καλώδια τροφοδοσίας με σκοπό τη μείωση ηλεκτρομαγνητικών παρεμβολών. Η τοποθέτηση έγινε απευθείας πάνω στο πλαίσιο με τη χρήση ταινίας διπλής όψης. Όπως και στον ελεγκτή πτήσης το βέλος προσανατολισμού της κεραίας GPS τοποθετήθηκε προς το εμπρόσθιο μέρος του UAV ώστε να επιτυγχάνεται σωστός υπολογισμός της κατεύθυνσης. Η τελική μορφή της τοποθέτησης φαίνεται στο σχήμα 3.26.

Σχήμα 3.26: Τοποθέτηση της μονάδας GPS/Compass

Τα καλώδια του GPS και της πυξίδας συνδέθηκαν στις αντίστοιχες θύρες του Pixhawk, όπως φαίνεται στο σχήμα 3.27.



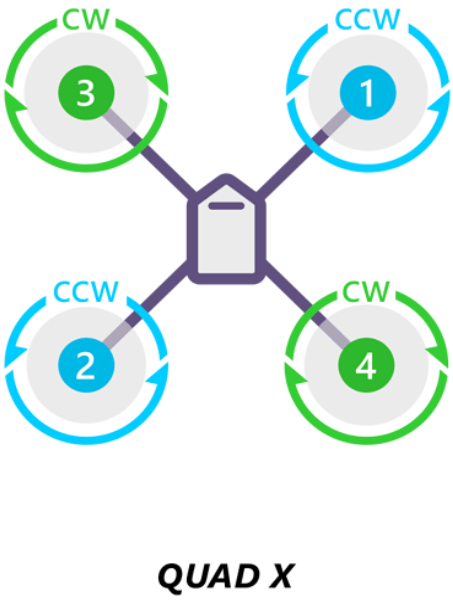
Σχήμα 3.27: Σύνδεση καλωδίων GPS και πυξίδας στον ελεγκτή πτήσης

3.3.2 Σύνδεση Κινητήρων και ESC

Μετά την ολοκλήρωση της εγκατάστασης του ελεγκτή πτήσης ακολούθησε η τοποθέτηση και σύνδεση των κινητήρων με τα ESC. Οι τέσσερις κινητήρες στερεωθήκαν στις βάσεις που βρίσκονται στα άκρα των βραχιόνων του πλαισίου όπως βλέπουμε και στο σχήμα 3.28.

Σχήμα 3.28: Τοποθέτηση κινητήρων στην βάση

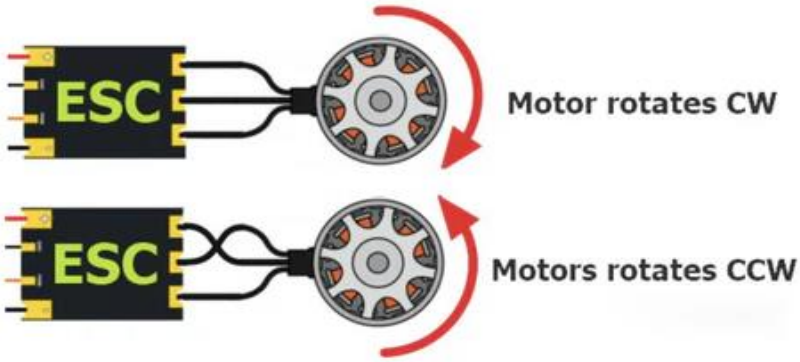
Ιδιαίτερη προσοχή δόθηκε στη σωστή διάταξη των κινητήρων σύμφωνα με τη διαμόρφωση Quad X όπως παρατηρούμε και στο σχήμα 3.29. Στη συγκεκριμένη διάταξη δύο κινητήρες περιστρέφονται δεξιόστροφα (CW) και δύο αριστερόστροφα (CCW). Η επιλογή είναι απαραίτητη για τη ισορροπία κατά την πτήση.



Σχήμα 3.29: Διάταξη κινητήρων Quad-X [35]

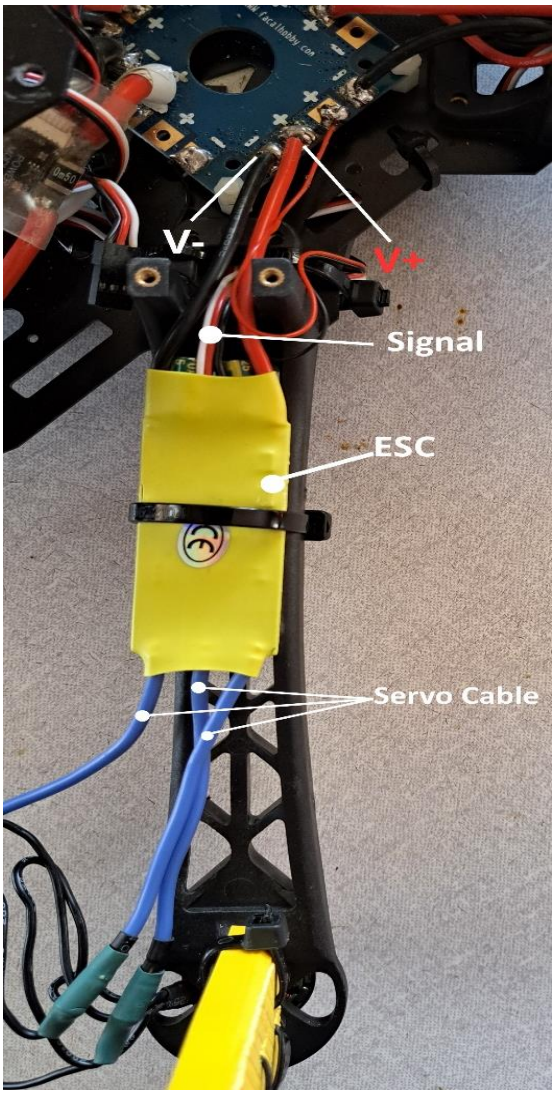
Στη συνέχεια κάθε κινητήρας συνδέθηκε με το αντίστοιχο ESC χρησιμοποιώντας τα τρία καλώδια φάσης όπως απεικονίζεται και στο σχήμα 3.30. Αφού ολοκληρώθηκε η σύνδεση ελέγχθηκε η φορά περιστροφής του κινητήρα και όπου χρειάστηκε, έγινε αλλαγή θέσης σε δύο από τα τρία καλώδια ώστε να διορθωθεί η περιστροφή όπως φαίνεται και στο σχήμα 3.31.

Σχήμα 3.30: Κόλληση των καλωδίων του μοτέρ με το ESC



Σχήμα 3.31: Αλλαγή φοράς περιστροφής κινητήρα μέσω εναλλαγής δύο καλωδίων

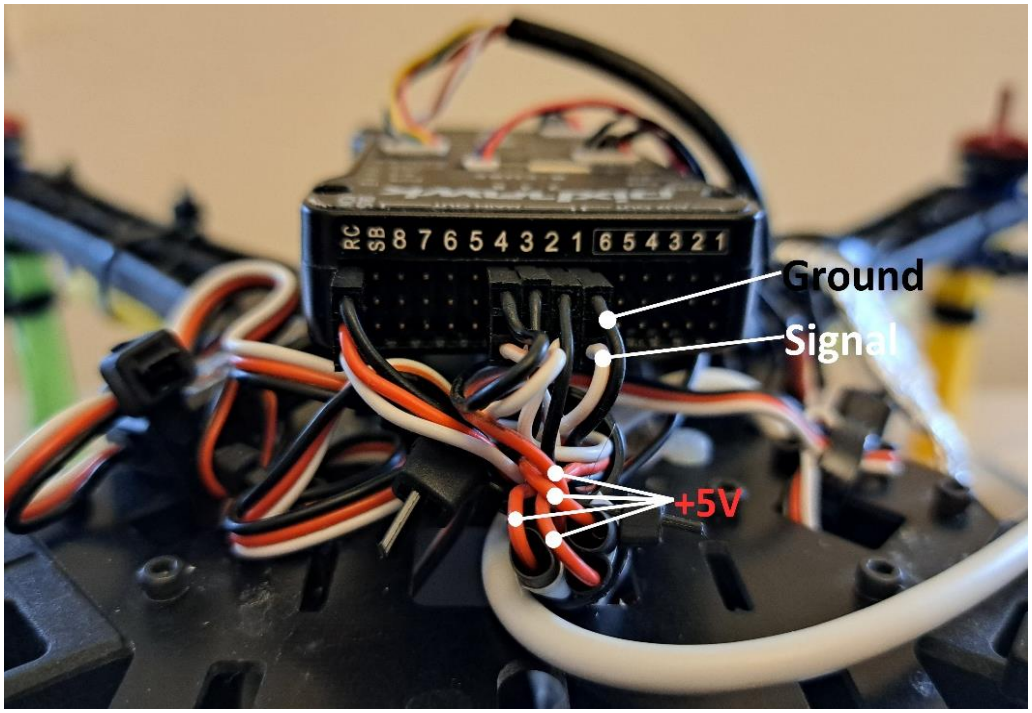
Έπειτα από την άλλη πλευρά συνδέθηκαν τα καλώδια τροφοδοσίας των ESC (θετικός και αρνητικός πόλος) στη PDB, από όπου λαμβάνουν την τάση της μπαταρίας.



Σχήμα 3.32: Συνδέσεις τροφοδοσίας ESC

Στη συνέχεια, ακολούθησε η σύνδεση του καλώδιού σήματος κάθε ESC στις αντίστοιχες εξόδους του Pixhawk. Συγκεκριμένα η σύνδεση των ESC στον ελεγκτή πτήσης πραγματοποιήθηκε σύμφωνα με την αρίθμηση των εξόδων MAIN OUT. Συγκεκριμένα το ESC του κινητήρα 1 συνδέθηκε στην έξοδο 1, το ESC του κινητήρα 2 στην έξοδο 2 και αντίστοιχα για τους κινητήρες 3 και 4 , όπως αποτυπώνεται και στο σχήμα 3.33. Με τον τρόπο αυτό διασφαλίζεται ότι κάθε κινητήρας θα έχει τη σωστή έξοδο σύμφωνα με τη διαμόρφωση Quad-X που έχει οριστεί στο λογισμικό του ελεγκτή πτήσης. Από τα τρία καλώδια κάθε ESC (σήμα, γείωση και +5V) το καλώδιο τροφοδοσίας (+5V) αφαιρέθηκε, όπως μπορούμε να δούμε και από το σχήμα 3.34, αφού ο Pixhawk τροφοδοτείται αποκλειστικά από το Power Module. Έτσι αποφεύγεται η παράλληλη τροφοδοσία και το σύστημα λειτουργεί σταθερά.

Σχήμα 3.33: Αντιστοίχιση ESC στις εξόδους MAIN OUT του Pixhawk [35]



Σχήμα 3.34: Σύνδεση των ESC χωρίς τα καλώδια +5V

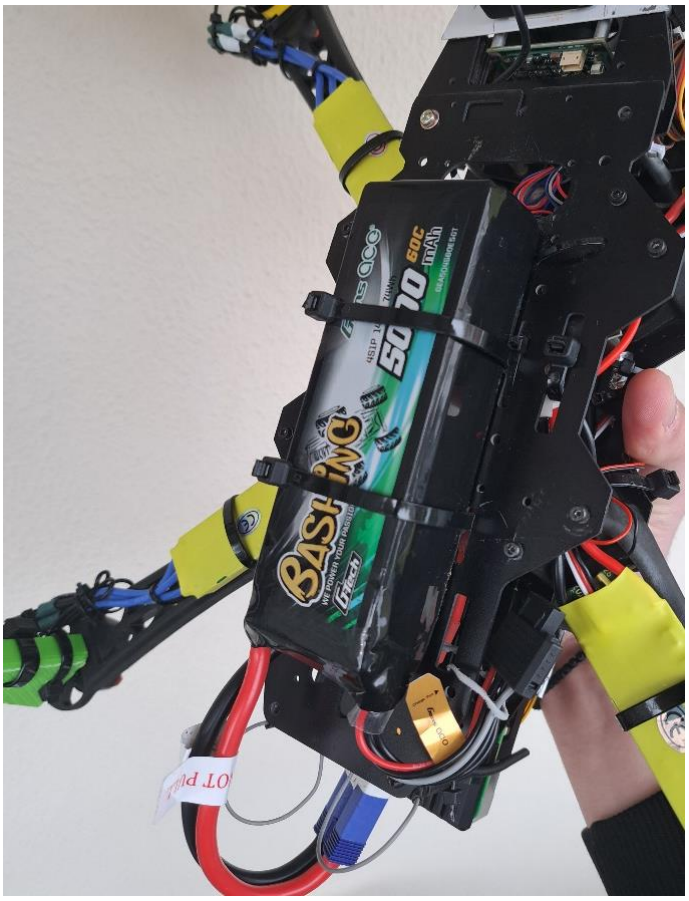
Μετά την ολοκλήρωση της συναρμολόγησης των κινητήρων και των ESC τοποθετήθηκαν οι έλικες. Χρησιμοποιήθηκαν δύο έλικες δεξιόστροφης περιστροφής (CW) και δύο αριστερόστροφης (CCW). Κάθε έλικας μπήκε στον αντίστοιχο κινητήρα με τη σωστή φορά και από πάνω βιδώθηκε το παξιμάδι εξασφαλίζοντας έτσι σταθερούς τους έλικες κατά την διάρκεια της πτήσης, όπως φαίνεται και στο σχήμα 3.35.

Σχήμα 3.35: Τοποθέτηση ελίκων και παξιμαδιών στους κινητήρες

3.3.3 Τροφοδοσία και Διαχείριση Ισχύος

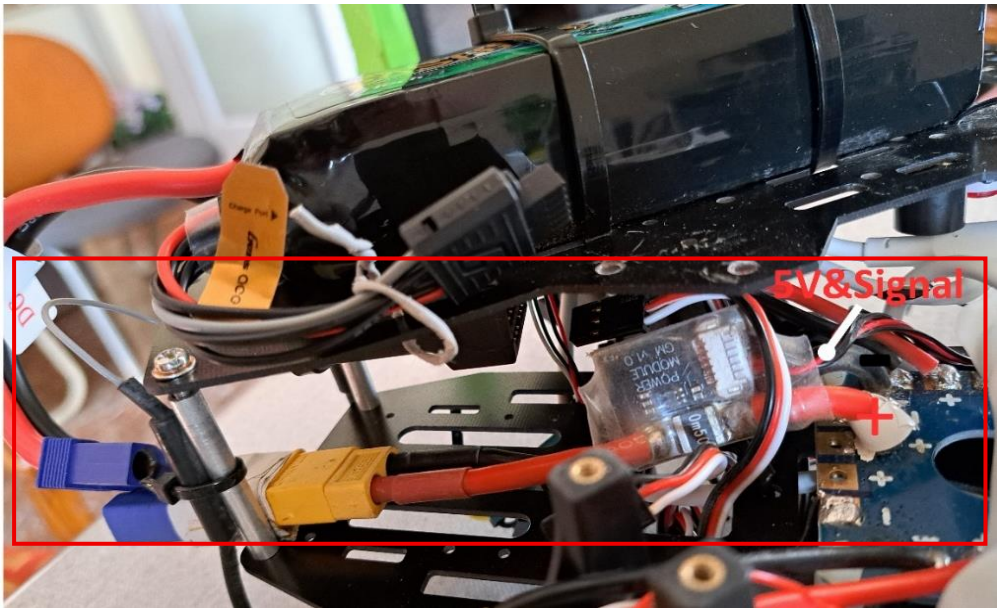
Έπειτα ακολούθησε η εγκατάσταση του συστήματος τροφοδοσίας το οποίο αποτελεί τη βάση για τη λειτουργία όλων των ηλεκτρονικών υποσυστημάτων.

Πρώτο βήμα αποτέλεσε η τοποθέτηση της μπαταρίας Li-Po στο κάτω μέρος του πλαισίου όπως διακρίνεται και στο σχήμα 3.36. Εκεί στερεώθηκε με δεματικά ώστε να εξασφαλίζεται σταθερότητα κατά τη λειτουργία και χαμηλό κέντρο βάρους.



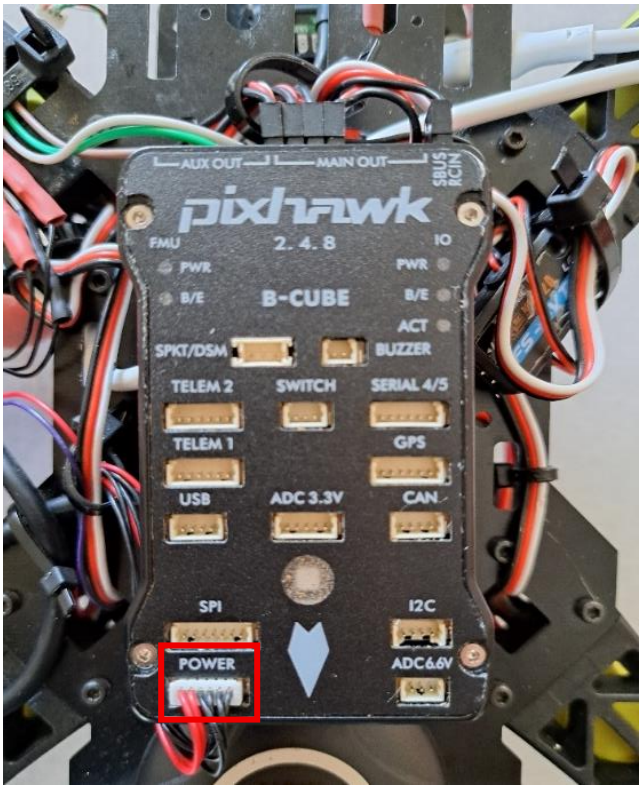
Σχήμα 3.36: Τοποθέτηση μπαταρίας Li-Po

Στη συνέχεια τοποθετήθηκε το Power Module το οποίο παρεμβλήθηκε ανάμεσα στην μπαταρία Li-Po και την πλακέτα διανομής ισχύος. Συγκεκριμένα, η είσοδος του Power Module συνδέθηκε στο βύσμα της μπαταρίας, ενώ η έξοδός του συνδέθηκε στην PDB, όπως απεικονίζεται και στο σχήμα 3.37, επιτρέποντας τη διέλευση της τάσης της μπαταρίας προς τα υπόλοιπα εξαρτήματα.



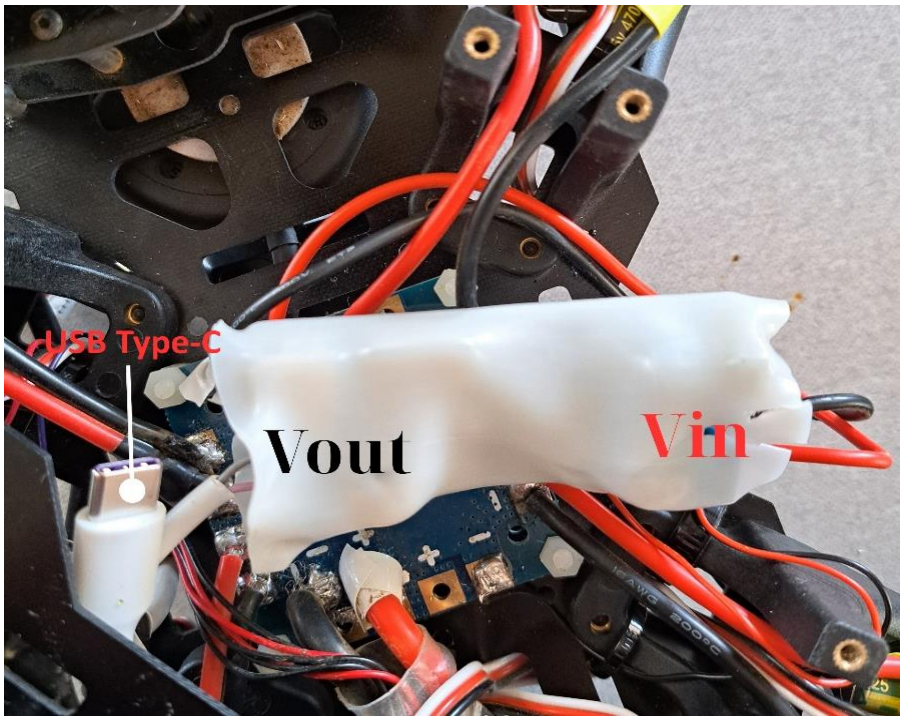
Σχήμα 3.37: Συνδεσμολογία Power Module

Παράλληλα, το καλώδιο 6-pin του Power Module συνδέθηκε στη θύρα POWER του Pixhawk. Μέσω της σύνδεσης αυτής το Power Module τροφοδοτεί τον ελεγκτή με 5V και μεταφέρει πληροφορίες για την τάση και το ρεύμα της μπαταρίας στο Pixhawk. Η συγκεκριμένη σύνδεση απεικονίζεται στο σχήμα 3.38.

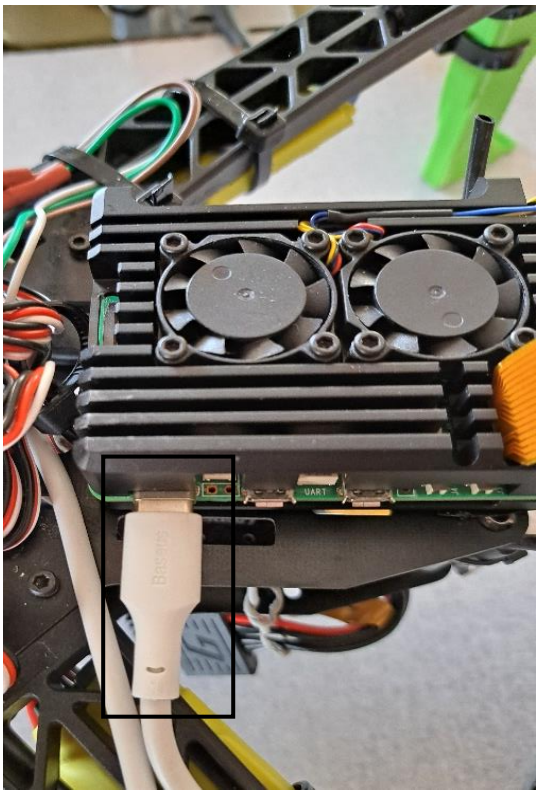


Σχήμα 3.38: Σύνδεση του Power Module στο Pixhawk

Στη συνέχεια τοποθετήθηκε ο μετατροπέας τάσης DC-DC για την τροφοδοσία του Raspberry Pi. Οι ακροδέκτες εισόδου (Vin) του μετατροπέα συνδέθηκαν απευθείας στους θετικούς και αρνητικούς πόλους της PDB ώστε να λαμβάνει την τάση της μπαταρίας, με τα καλώδια να κολλιούνται με καλάι για σταθερή και αξιόπιστη ηλεκτρική επαφή. Στην έξοδο (Vout) χρησιμοποιήθηκε καλώδιο USB A σε USB Type-C από το οποίο αφαιρέθηκε το άκρο USB A και απογυμνώθηκαν οι αγωγοί τροφοδοσίας προκειμένου να τοποθετηθούν στις κλέμες εξόδου του μετατροπέα, με τη διάταξη να παρουσιάζεται στο σχήμα 3.39. Μέσω αυτής της σύνδεσης παρέχεται στο Raspberry Pi τάση 5V και η απαιτούμενη ένταση ρεύματος για την ασφαλή λειτουργία του, μέσω του καλωδίου USB Type-C που καταλήγει στη θύρα τροφοδοσίας του, όπως φαίνεται στο σχήμα 3.40. Για επιπλέον προστασία και ηλεκτρική μόνωση, ο μετατροπέας καλύφθηκε με θερμοσυστελλόμενο υλικό.

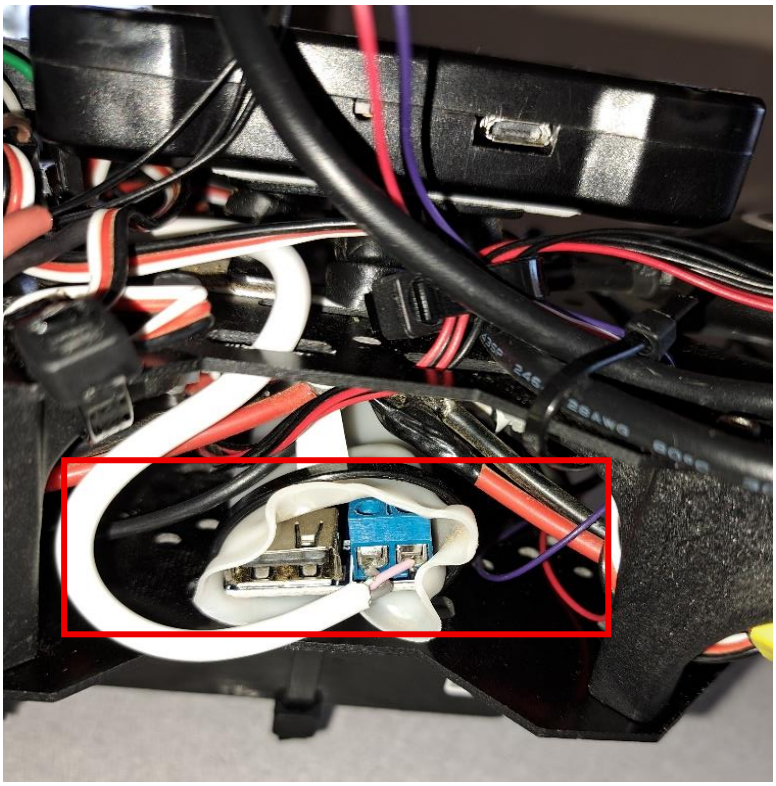


Σχήμα 3.39: Διάταξη τροφοδοσίας με το Vin από την μπαταριά και το Vout προς το Raspberry Pi



Σχήμα 3.40: Σύνδεση USB Type C στο Raspberry Pi.

Τέλος, ο μετατροπέας τοποθετήθηκε ανάμεσα στα δύο επίπεδα του πλαισίου σε θέση κοντά στην PDB ώστε να διατηρούνται μικρά τα μήκη των καλωδίων και να περιορίζονται οι απώλειες. Στερεώθηκε με δεματικά για να παραμένει σταθερός κατά τη λειτουργία όπως απεικονίζεται και στο σχήμα 3.41.

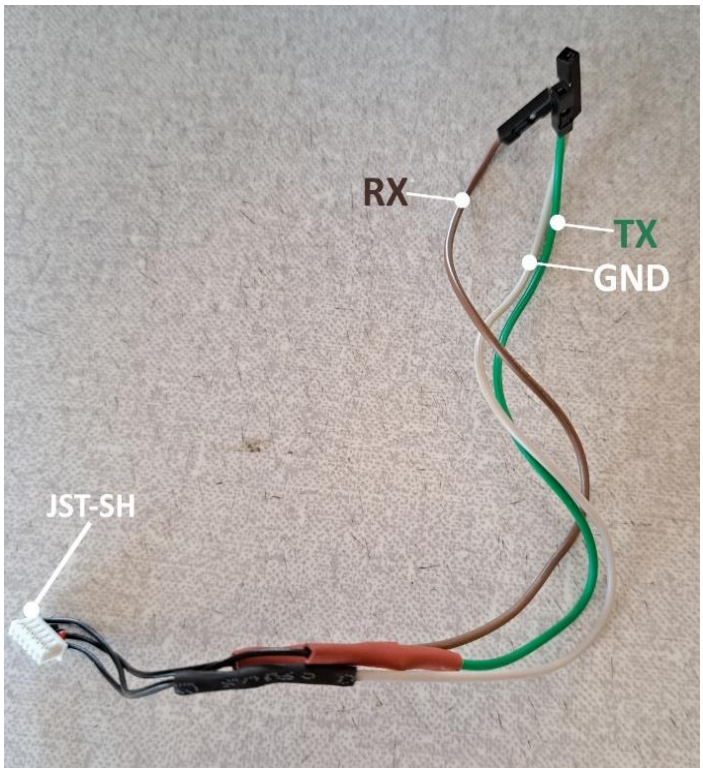


Σχήμα 3.41: Τοποθέτηση μετατροπέα τάσης στο πλαίσιο

3.3.4 Τοποθέτηση Raspberry Pi και επικοινωνία με ελεγκτή πτήσης

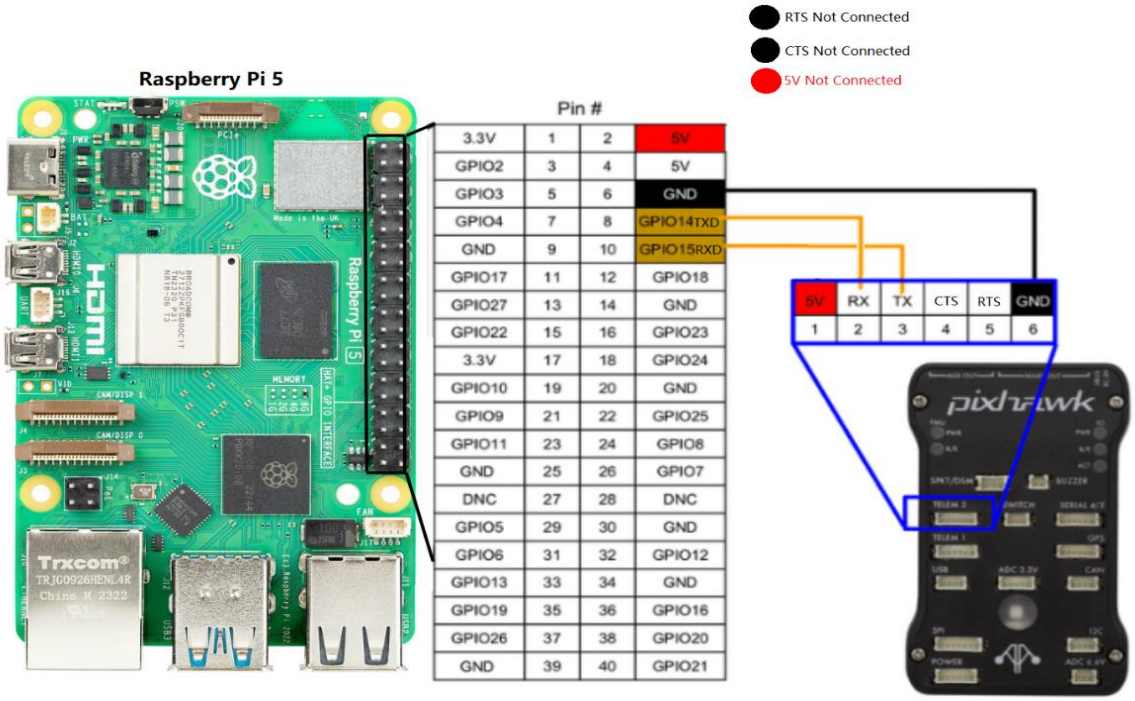
Αφού εξασφαλίστηκε η τροφοδοσία του Raspberry Pi μέσω του μετατροπέα DC-DC, ακολούθησε η διαδικασία σύνδεσης με τον ελεγκτή πτήσης. Αρχικά, το Raspberry Pi τοποθετήθηκε σε κοντινή απόσταση από το flight controller ώστε να ελαχιστοποιηθεί το μήκος των καλωδιώσεων και να περιοριστούν πιθανές απώλειες λόγω ηλεκτρομαγνητικού θορύβου.

Έπειτα, η επικοινωνία μεταξύ Raspberry Pi και του ελεγκτή πτήσης υλοποιήθηκε μέσω της θύρας τηλεμετρίας (TELEM), χρησιμοποιώντας καλώδιο JST-GH 6 pin συμβατό με τη συγκεκριμένη θύρα του Pixhawk. Από το καλώδιο αυτό αφαιρέθηκαν οι αγωγοί που δεν ήταν απαραίτητοι, κρατώντας μόνο τα καλώδια TX (Transmit), RX (Receive) και GND (Ground), όπως μπορούμε να δούμε και από το σχήμα 3.42. Τα καλώδια απογυμνώθηκαν από τη μια πλευρά και συνδέθηκαν με θηλυκά jumper μέσω κόλλησης.



Σχήμα 3.42: Τοποθέτηση μετατροπέα τάσης στο πλαίσιο

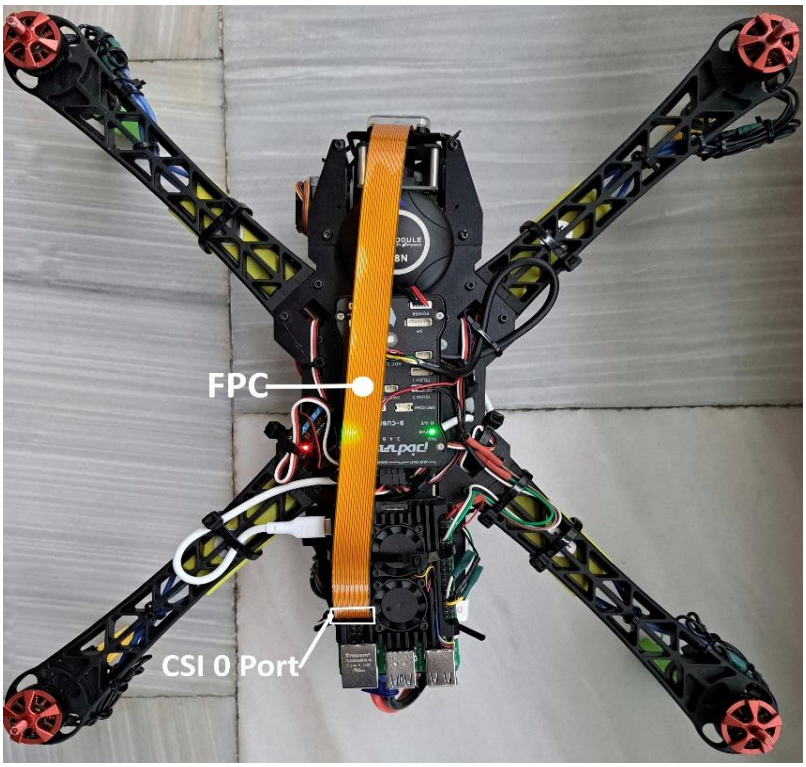
Η σύνδεση πραγματοποιήθηκε με την έξοδο TX του ελεγκτή πτήσης να οδηγείται στην είσοδο RX του Raspberry Pi και αντίστροφα η έξοδος TX του Raspberry Pi στην είσοδο RX του ελεγκτή πτήσης. Όπως αποτυπώνεται και στο σχήμα 3.43 η σύνδεση πραγματοποιήθηκε μέσω των ακίδων GPIO όπου το RX του Pixhawk συνδέθηκε στο pin 8 του Raspberry Pi και το TX του Pixhawk στο pin 10 του Raspberry Pi, ενώ χρησιμοποιήθηκε το pin 6 για την κοινή γείωση των δύο συστημάτων.



Σχήμα 3.43: Συνδεσμολογία αναμεσά σε Raspberry Pi και Pixhawk

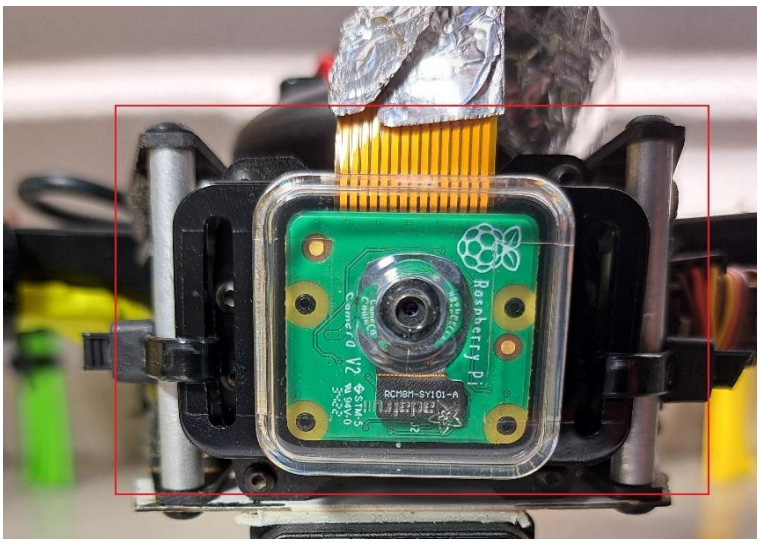
3.3.5 Σύνδεση των Περιφερικών του Raspberry Pi

Στη συνέχεια και μετά την τοποθέτηση του Raspberry Pi πραγματοποιήθηκε η σύνδεση της κάμερας και του αισθητήρα LiDAR (TFmini Plus). Αρχικά η κάμερα συνδέθηκε στην θύρα CSI 0 του Raspberry Pi μέσω καλωδίου τύπου FPC (Flexible Printed Circuit) όπως φαίνεται στο σχήμα 3.44.



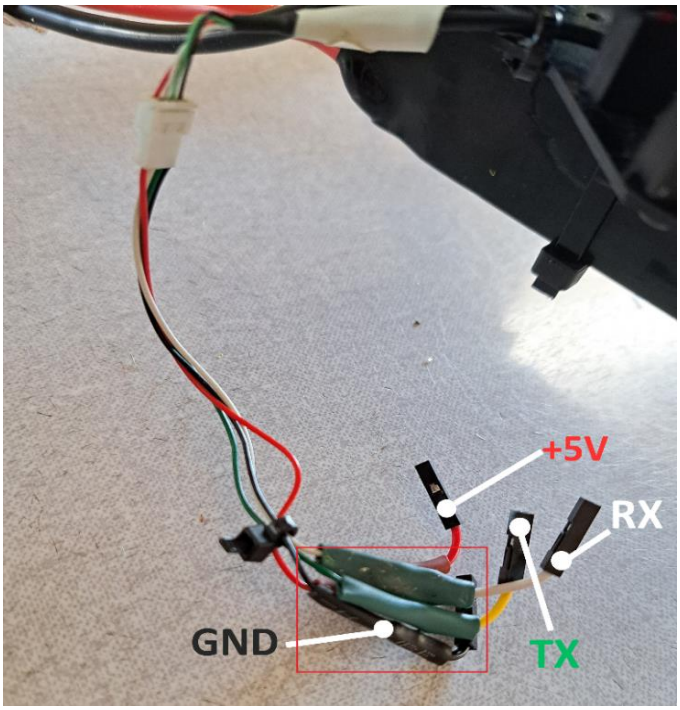
Σχήμα 3.44: Σύνδεση κάμερας και Raspberry Pi μέσω FPC

Έπειτα τοποθετήθηκε η κάμερα στο μπροστινό μέρος του πλαισίου ώστε να έχει καθαρό και ευρύ οπτικό πεδίο προς τα εμπρός και στερεώθηκε στις κατακόρυφες κολόνες που συνδέουν τα 2 επίπεδα του πλαισίου με τη χρήση δεματιών, όπως παρουσιάζεται στο σχήμα 3.45. Η συγκεκριμένη θέση επιλέχθηκε με σκοπό η κάμερα να μπορεί να ανιχνεύει ανθρώπινη παρουσία κατά τη διάρκεια της πτήσης.



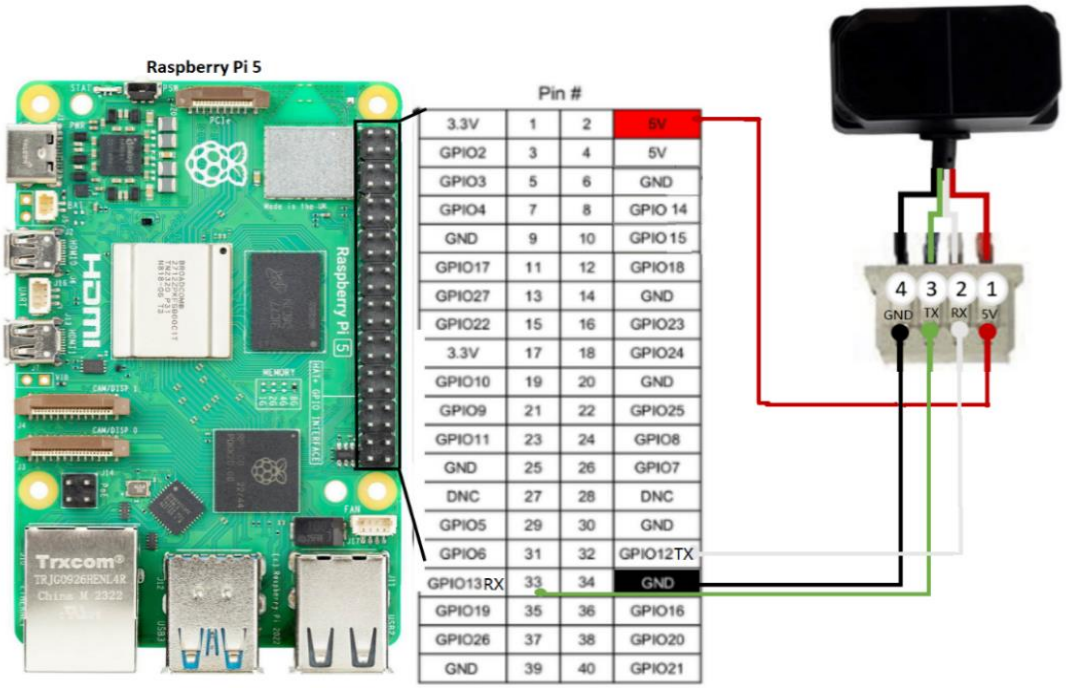
Σχήμα 3.45: Τοποθέτηση της κάμερας στο μπροστινό μέρος

Έπειτα ακολούθησε η εγκατάσταση του αισθητήρα LiDAR. Επειδή το καλώδιο του αισθητήρα κατέληγε σε jumper αρσενικού τύπου, ενώ για το Raspberry Pi απαιτούνται θηλυκοί ακροδέκτες αντικαταστάθηκαν οι ακροδέκτες με jumper θηλυκά. Τα καλώδια απογυμνώθηκαν και συνδέθηκαν μέσω κόλλησης με jumper θηλυκά ώστε να μπορεί να συνδεθεί ο LiDAR με τα GPIO pins του Raspberry Pi, όπως βλέπουμε στο σχήμα 3.46.



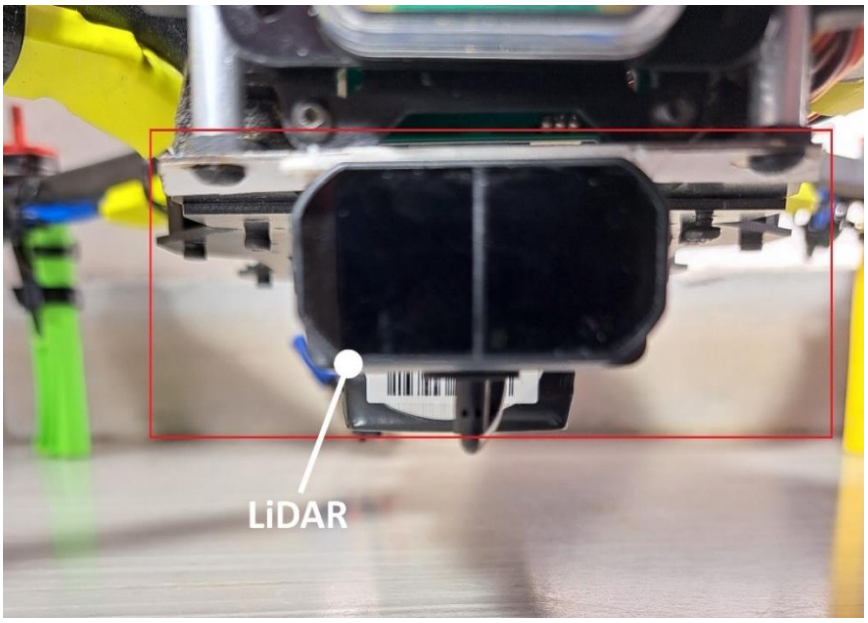
Σχήμα 3.46: Μετατροπή καλωδίου με θηλυκούς ακροδέκτες jumper

Η σύνδεση του αισθητήρα LiDAR με το Raspberry Pi έγινε σύμφωνα με τη διάταξη που παρουσιάζεται στο σχήμα 3.47. Συγκεκριμένα ο ακροδέκτης TX του LiDAR συνδέθηκε στο RX του Raspberry Pi στο pin 33 και αντίστοιχα ο RX του αισθητήρα στο TX του Raspberry pi στο pin 32. Για την τροφοδοσία του αισθητήρα χρησιμοποιήθηκαν τα 5V από το Raspberry Pi από το pin 2, ενώ η γείωση συνδέθηκε στο pin 34.



Σχήμα 3.47: Συνδεσμολογία TFmini Plus με το Raspberry Pi

Ο αισθητήρας τοποθετήθηκε στο εμπρόσθιο κάτω μέρος του πλαισίου, κάτω από την κάμερα. Η θέση επιλέχθηκε ώστε ο LiDAR να μπορεί να ανιχνεύσει τα εμπόδια που βρίσκονται μπροστά από το drone και να εκτιμά την απόστασή τους, βοηθώντας στο εντοπισμό ελεύθερου χώρου. Για την εγκατάστασή του κατασκευάστηκε μεταλλική βάση, η οποία προσαρμόστηκε απευθείας στο πλαίσιο του drone. Ο αισθητήρας στερεώθηκε πάνω στη βάση με ταινία διπλής όψης όπως φαίνεται και στο σχήμα 3.48.



Σχήμα 3.48: Τοποθέτηση του αισθητήρα LiDAR στη μεταλλική βάση

3.4 Συνολικό Κόστος Υλοποίησης

Η επιλογή υλικών, όπως παρουσιάστηκε στην ενότητα 3.2, έγινε με βάση τις τεχνικές απαιτήσεις της κατασκευής και με στόχο τη διατήρηση του κόστους σε λογικά επίπεδα. Ο πίνακας 3.5 δείχνει το προσεγγιστικό κόστος των εξαρτημάτων, με τιμές λιανικής πώλησης.

Πίνακας 3.5: Κόστος υλικών, Δεκέμβριος 2025

Εξάρτημα	Μοντέλο	Προσεγγιστικό κόστος (€)
Πλαίσιο	F450	13
Ηλεκτροκινητήρες	Brotherhobby Returner 2207-2550kv	32
Έλικες	5045 (5x4.5)	5
ESC	Xida XXD 40A	28
Μπαταριά	Gens Ace G-tech 5000mah 14.8v 4s 60c	56
Φορτιστής Μπαταρίας	SkyRC E430	34
Μονάδα διαχείρισης ισχύος	-	15
Ελεγκτής πτήσης	Pixhawk 2.4.8	65
Μονάδα GPS με πυξίδα	NEO-M8N	35
Πομποδέκτης χειρισμού	FlySky FS-i6X και iA6B	90
Τηλεμετρία τάσης	FlySky FS-CVT01	11
Raspberry Pi 5	8GB RAM	98
Σύστημα ψύξης	Waveshare Aluminum Case	9
Raspberry Pi MicroSD Card	128GB	23
Καλώδιο Τροφοδοσίας	Baseus USB-A σε USB-C	4
Μετατροπέας DC-DC	-	7
LiDAR	Benewake TFmini Plus	42
Raspberry Pi Camera	V2 - 8MP, 1080p	18
Καλώδιο ταινία CSI	-	2
Συνολικό Κόστος		587

Κεφάλαιο 4ο: Ρυθμίσεις Πτήσης και Λογισμική Διαμόρφωση

4.1 Εισαγωγή

Αφού ολοκληρώθηκε η μηχανική και ηλεκτρική κατασκευή του UAV το επόμενο στάδιο αφορά τη λογισμική διαμόρφωση και εφαρμογή των απαραίτητων ρυθμίσεων που επιτρέπουν στο σύστημα να λειτουργεί σωστά. Η φάση αυτή είναι κρίσιμη καθώς η αξιόπιστη πτήση δεν εξαρτάται μόνο από τη συναρμολόγηση. Απαιτείται επίσης η κατάλληλη παραμετροποίηση του λογισμικού που καθοδηγεί τον ελεγκτή πτήσης, ώστε το σύστημα να λειτουργεί με σταθερότητα και ασφάλεια. Ο ελεγκτής πτήσης απαιτεί συγκεκριμένες ρυθμίσεις και βαθμονομήσεις ώστε το UAV να μπορεί να παρουσιάζει αξιόπιστη συμπεριφορά στον αέρα. Παράλληλα και το υπολογιστικό σύστημα χρειάζεται την κατάλληλη προετοιμασία ώστε να μπορεί να υποστηρίζει αποτελεσματικά τις λειτουργίες που του ανατίθενται και να συνεργάζεται ομαλά με τα υπόλοιπα υποσυστήματα.

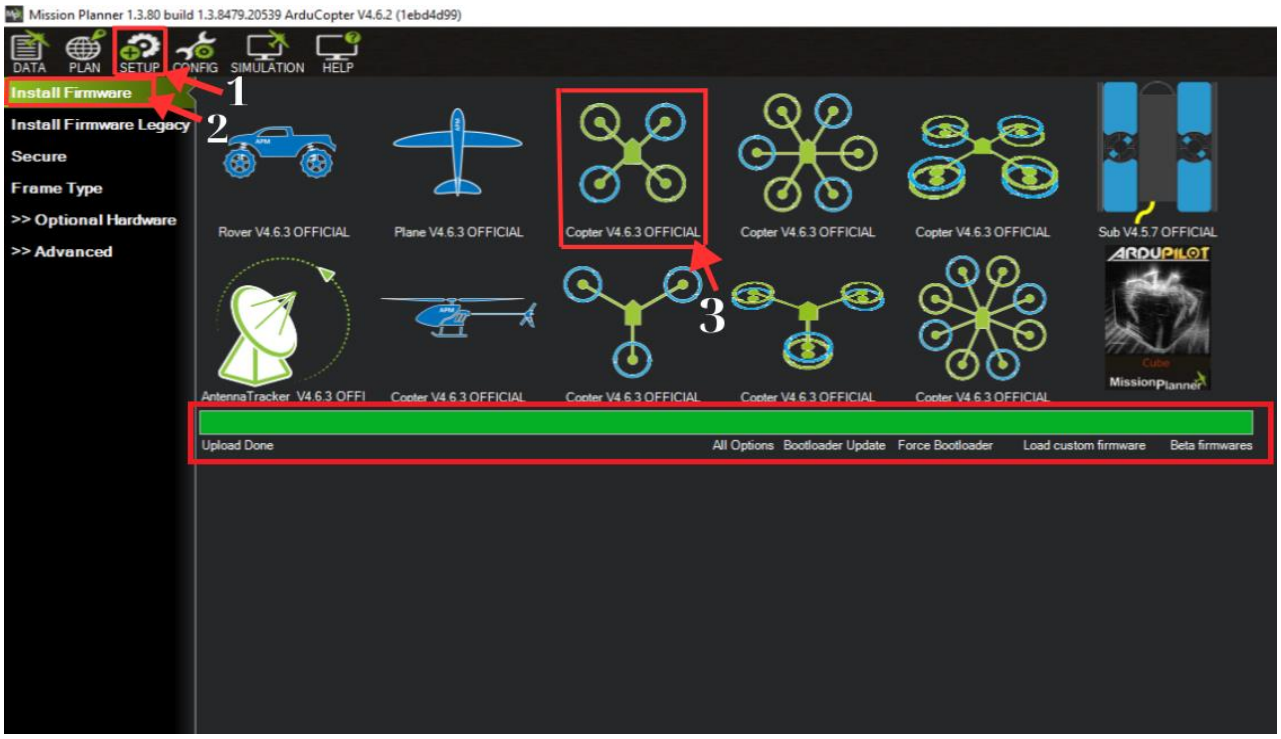
Στις ενότητες που ακολουθούν παρουσιάζεται το λογισμικό ελέγχου πτήσης. Παρουσιάζονται επίσης οι βασικές ρυθμίσεις που χρειάζεται το UAV για να λειτουργεί με ασφάλεια. Στη συνέχεια εξετάζονται τα διαθέσιμα πτητικά modes και ο τρόπος με τον οποίο ρυθμίζονται στο σύστημα. Τέλος, περιγράφεται η προετοιμασία του υπολογιστικού συστήματος του Raspberry Pi, ώστε να επικοινωνεί με το ελεγκτή πτήσης.

4.2 Λογισμικό Ελέγχου Πτήσης

Στην παρούσα ενότητα παρουσιάζεται το λογισμικό ελέγχου πτήσης που χρησιμοποιήθηκε για τη διαμόρφωση του συστήματος. Για την επικοινωνία με τον ελεγκτή πτήσης επιλέχθηκε η εφαρμογή Mission Planner, η οποία λειτουργεί ως σταθμός εδάφους (Ground Control Station). Η συγκεκριμένη εφαρμογή συνεργάζεται με το λογισμικό ArduPilot το οποίο εγκαθίσταται στον ελεγκτή και χρησιμοποιείται για την λειτουργία του drone. Παρότι υπάρχουν και άλλες αντίστοιχες λύσεις όπως η εφαρμογή QGroundControl σε συνδυασμό με το PX4, στην παρούσα υλοποίηση επιλέχθηκε ο συνδυασμός ArduPilot με το Mission Planner λόγω του εύχρηστου περιβάλλοντος και της δυνατότητας παραμετροποίησης του ελεγκτή.

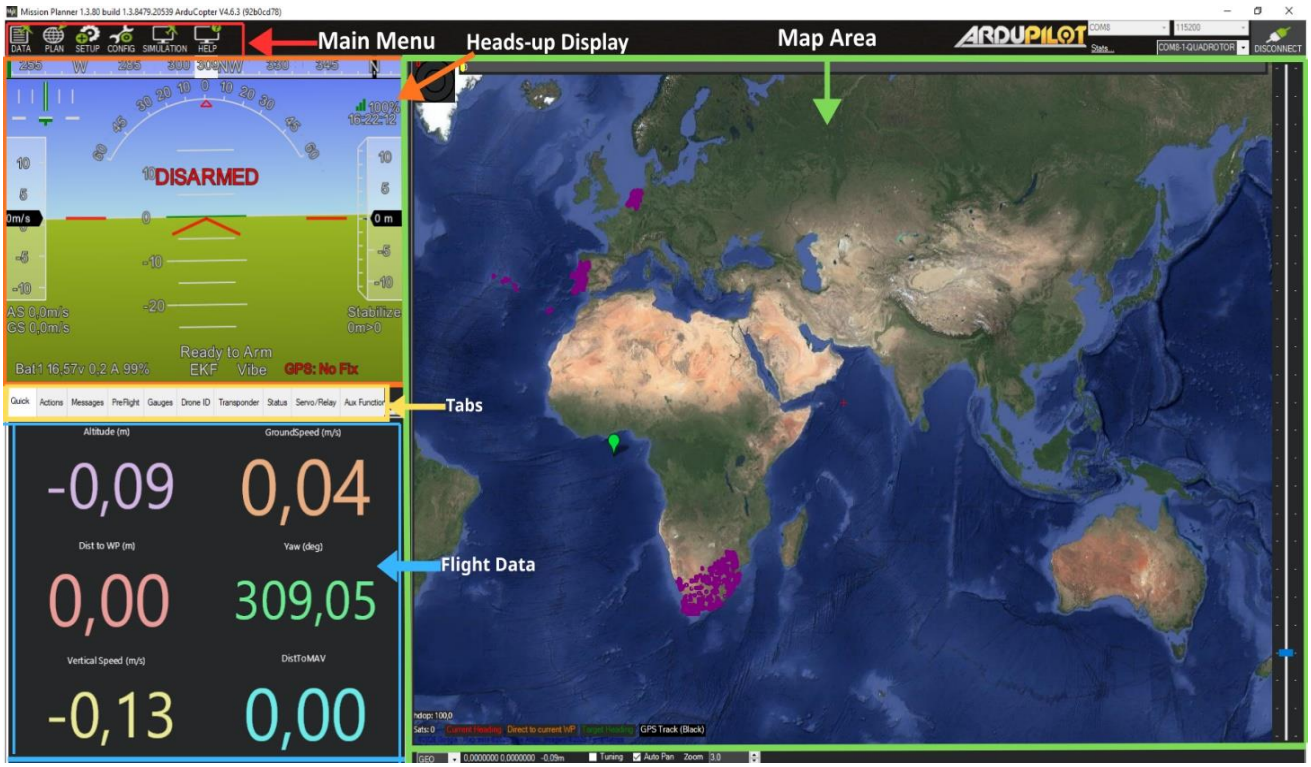
Για την επικοινωνία του ελεγκτή πτήσης με την εφαρμογή Mission Planner πραγματοποιήθηκε σύνδεση μέσω καλωδίου Micro USB το οποίο συνδέθηκε στον ελεγκτή και στον υπολογιστή. Μέσω της ενσύρματης αυτής σύνδεσης γίνεται δυνατή η ρύθμιση και παραμετροποίηση του ελεγκτή πτήσης.

Έπειτα ακολούθησε η εγκατάσταση του ArduPilot και συγκεκριμένα της έκδοσης ArduCopter η οποία αντιστοιχεί σε διαμόρφωση τετρακόπτερου (Quad). Η εγκατάσταση πραγματοποιήθηκε μέσω της καρτέλας Setup και της επιλογής Install Firmware του Mission Planner. Από το διαθέσιμο μενού επιλέχθηκε το εικονίδιο του τετρακόπτερου και στη συνέχεια έγινε η λήψη και η μεταφόρτωση της αντίστοιχης έκδοσης λογισμικού στον ελεγκτή, βάσει της διαδικασίας που απεικονίζεται στο σχήμα 4.1. Το ArduCopter αποτελεί το λογισμικό του ελεγκτή πτήσης που εκτελείται εσωτερικά και υλοποιεί τον αυτόματο πιλότο του UAV. Μετά την εγκατάσταση του firmware ο ελεγκτής αναλαμβάνει την επεξεργασία των δεδομένων από τους αισθητήρες και την εκτέλεση αλγορίθμων σταθεροποίησης.



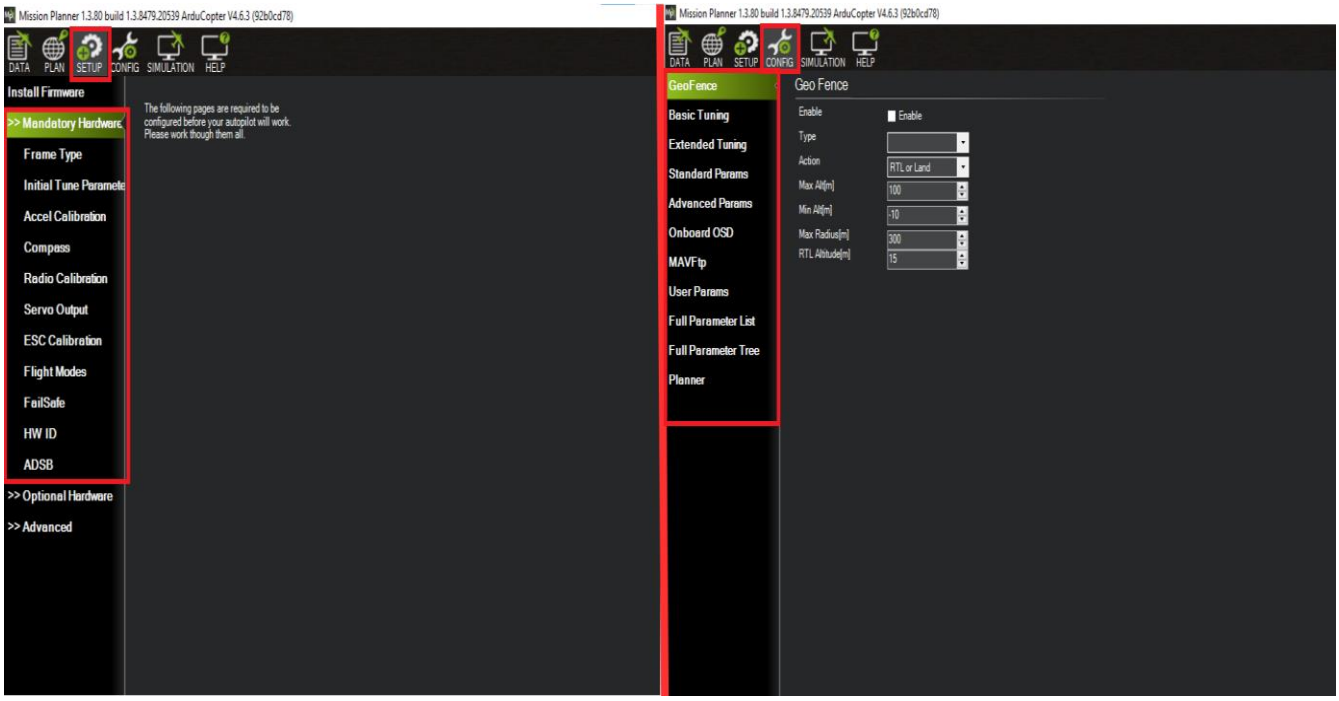
Σχήμα 4.1: Εγκατάσταση ArduCopter (Quad) στον ελεγκτή πτήσης

Η εφαρμογή Mission Planner λειτουργεί ως εργαλείο παρακολούθησης πριν από την πτήση. Μέσω αυτής είναι δυνατή η προβολή δεδομένων σε πραγματικό χρόνο όπως ο προσανατολισμός του UAV, οι ενδείξεις αισθητήρων και η κατάσταση του GPS. Στο σχήμα 4.2 παρουσιάζεται το βασικό περιβάλλον της εφαρμογής, όπου απεικονίζονται τα κύρια στοιχεία παρακολούθησης, όπως το main menu, το Head-up display (HUD), η περιοχή του χάρτη και ο πίνακας τηλεμετρίας.



Σχήμα 4.2: Το περιβάλλον της εφαρμογής Mission Planner

Επιπλέον, η εφαρμογή παρέχεται τη δυνατότητα εκτέλεσης κάποιων διαδικασιών βαθμονόμησης, όπως της πυξίδας, του επιταχυνσιομέτρου και της τηλεκατεύθυνσης, καθώς και άλλων απαραίτητων ρυθμίσεων λειτουργίας και ασφάλειας του συστήματος όπως φαίνεται στο σχήμα 4.3. Οι διαδικασίες αυτές αναλύονται περισσότερο στις επόμενες ενότητες.



Σχήμα 4.3: Μενού ρυθμίσεων του Mission Planner

4.3 Βασικές Ρυθμίσεις Πτήσης

Πριν από την πρώτη πτήση απαιτείται η εκτέλεση βασικών ρυθμίσεων που εξασφαλίζουν τη σωστή λειτουργία και την ασφάλεια του συστήματος. Οι ρυθμίσεις αυτές περιλαμβάνουν τη σύζευξη του τηλεχειριστηρίου με τον δεκτή, τη βαθμονόμηση του τηλεχειρισμού, της πυξίδας και των ESC, καθώς και τον έλεγχο κρίσιμων παραμέτρων λειτουργίας. Ιδιαίτερη σημασία δίνεται στη ρύθμιση των παραμέτρων PID οι οποίες καθορίζουν τη σταθερότητά και την απόκριση του UAV. Όπως θα παρουσιαστεί παρακάτω καθεμία από αυτές τις διαδικασίες αποτελεί απαραίτητο βήμα για την ολοκληρωμένη προετοιμασία του συστήματος.

4.3.1 Σύζευξη Τηλεχειριστηρίου με το Δεκτή

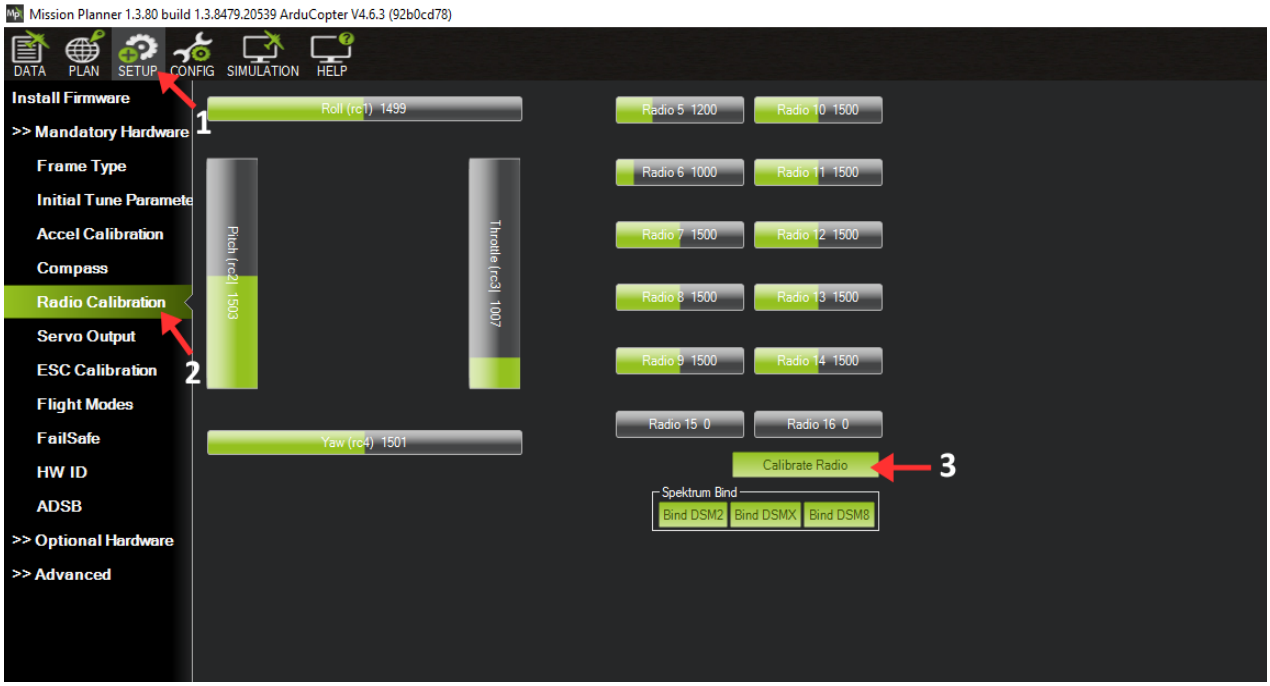
Για να μπορέσει το UAV να λαμβάνει εντολές από το χειριστή απαιτείται η σύζευξη του πομπού με τον δέκτη. Η διαδικασία αυτή είναι απαραίτητη ώστε ο πομπός και ο δέκτης να αναγνωρίζουν ο ένας τον άλλον και να μπορούν να επικοινωνούν σωστά.

Όπως απεικονίζεται και στο παρακάτω σχήμα 4.4, η διαδικασία σύζευξης του τηλεχειριστηρίου με το δέκτη ξεκινά με την τοποθέτηση bind plug στη θύρα B/VCC του δέκτη. Με τον τρόπο αυτό ενεργοποιείται η λειτουργία bind, η οποία επιτρέπει στον δέκτη να αναζητήσει και να αναγνωρίσει το αντίστοιχο τηλεχειριστήριο. Στη συνέχεια ο δέκτης τροφοδοτείται μέσω τις θύρας servo και το τηλεχειριστήριο τίθεται σε λειτουργία σύζευξης πατώντας το πλήκτρο bind και ενεργοποιώντας το ταυτόχρονα ώστε να ολοκληρωθεί η διαδικασία αντιστοίχισης μεταξύ πομπού και δέκτη. Όταν η διαδικασία ολοκληρωθεί επιτυχώς ο δέκτης και ο πομπός συνδέονται μεταξύ τους και επιτυγχάνεται σταθερή επικοινωνία, οπότε αφαιρείτε το bind plug.

Σχήμα 4.4: Διαδικασία σύζευξης πομπού και δεκτή

4.3.2 Βαθμονόμηση Τηλεχειριστηρίου

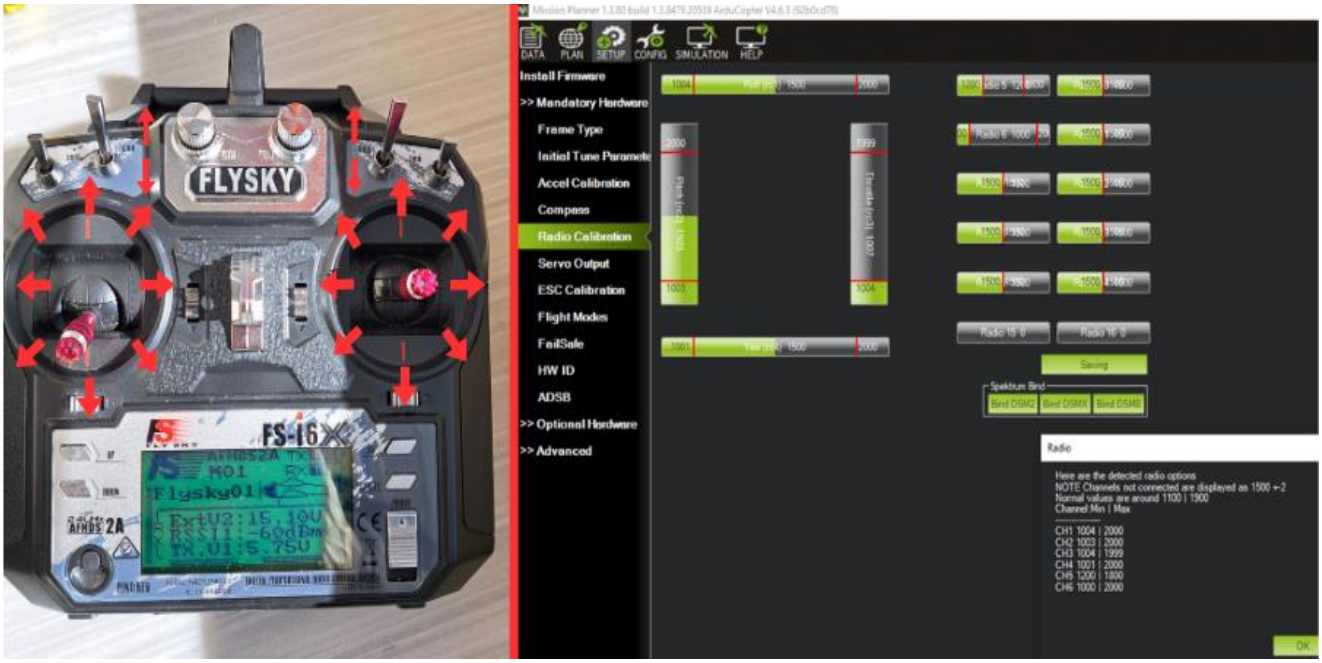
Η βαθμονόμηση του τηλεχειριστηρίου αποτελεί ένα από τα σημαντικότερα βήματα πριν από την πρώτη πτήση καθώς επιτρέπει στο flight controller να αναγνωρίζει σωστά τις εντολές του χειριστή. Η διαδικασία πραγματοποιείται μέσω της εφαρμογής Mission Planner επιλέγοντας από το μενού ¨SETUP¨ την καρτέλα “Radio Calibration”, με τη σχετική διεπαφή να παρουσιάζεται στο σχήμα 4.5.



Σχήμα 4.5: Βήματα για τη βαθμονόμηση τηλεχειριστηρίου

Κατά τη διάρκεια της βαθμονόμησης οι μοχλοί του τηλεχειριστηρίου μετακινούνται από το χρήστη σε όλες τις κατευθύνσεις πάνω, κάτω, αριστερά, δεξιά και διαγώνια ώστε ο ελεγκτής πτήσης να καταγράψει τις ακραίες τιμές και την ουδέτερη θέση κάθε καναλιού. Με αυτόν τον τρόπο

προσδιορίζονται τα ελάχιστα και μέγιστα όρια λειτουργίας, καθώς και το κεντρικό σημείο των μοχλών. Οι κινήσεις αυτές μετατρέπονται σε ηλεκτρικά σήματα PWM, τα οποία αποστέλλονται στον δέκτη και στη συνέχεια στον ελεγκτή πτήσης. Τα σήματα αυτά μεταφράζονται σε αριθμητικές τιμές που συνήθως κυμαίνονται από το 1000 έως το 2000. Μέσω της βαθμονόμησης ο ελεγκτής καταγράφει τα πραγματικά όρια των σημάτων ώστε να μπορεί να ερμηνεύει με ακρίβεια τις εντολές του χειριστή, όπως μπορούμε να δούμε και από το σχήμα 4.6. Έτσι εξασφαλίζεται ότι οι κινήσεις των μοχλών αντιστοιχούν σωστά στη συμπεριφορά του αεροσκάφους κατά την πτήση, προσφέροντας ομαλή και προβλέψιμη απόκριση.

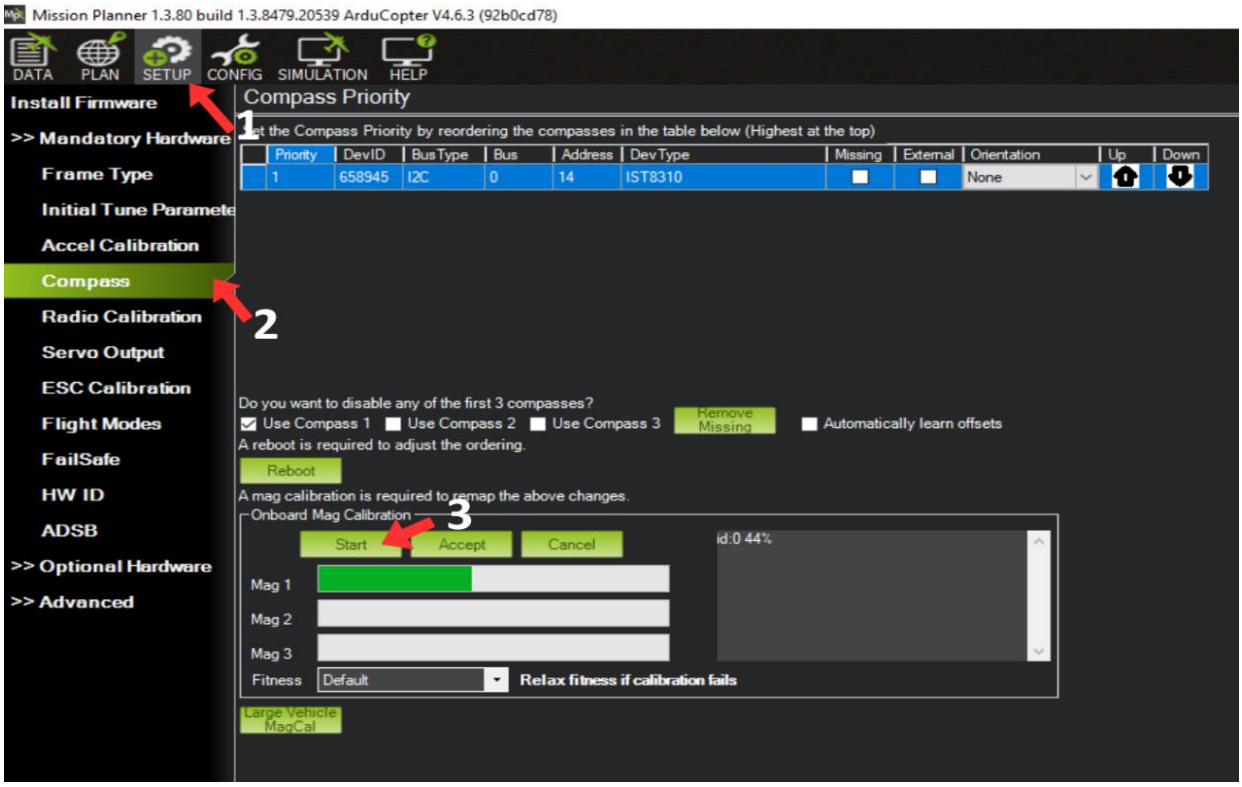


Σχήμα 4.6: Βαθμονόμηση του τηλεχειριστηρίου

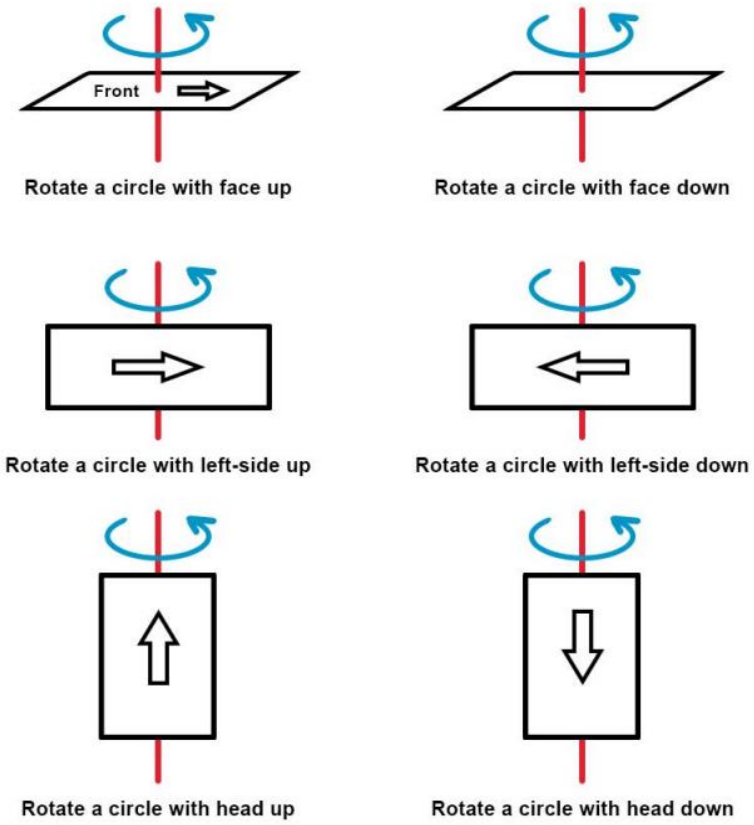
4.3.3 Βαθμονόμηση της Πυξίδας

Η βαθμονόμηση της πυξίδας είναι μια από τις πιο κρίσιμες ρυθμίσεις του συστήματος πτήσης καθώς επηρεάζει άμεσα τον προσανατολισμό και την πλοήγηση του drone. Μέσω αυτής της διαδικασίας, ο ελεγκτής πτήσης λαμβάνει σωστές πληροφορίες για την κατεύθυνση βασιζόμενος σε έναν αισθητήρα που μετρά το μαγνητικό πεδίο της Γης και επιτρέπει τον υπολογισμό του προσανατολισμού σε σχέση με τον μαγνητικό βορρά.

Η ρύθμιση της πυξίδας γίνεται από την εφαρμογή Mission Planner, επιλέγοντας από το μενού “SETUP” την επιλογή “Compass Calibration” και πατώντας στη συνέχεια το κουμπί “Start”, όπως απεικονίζεται στο σχήμα 4.7. Κατά τη διάρκειά της, ο χειριστής κρατάει το drone και το περιστρέφει σε διάφορους προσανατολισμούς για να συλλεχθούν δεδομένα από όλες τις πλευρές. Οι κινήσεις αυτές διακρίνονται στο σχήμα 4.8. Με αυτόν τον τρόπο το μαγνητόμετρο καταγράφει το μαγνητικό πεδίο σε όλο το εύρος κίνησης του αεροσκάφους.

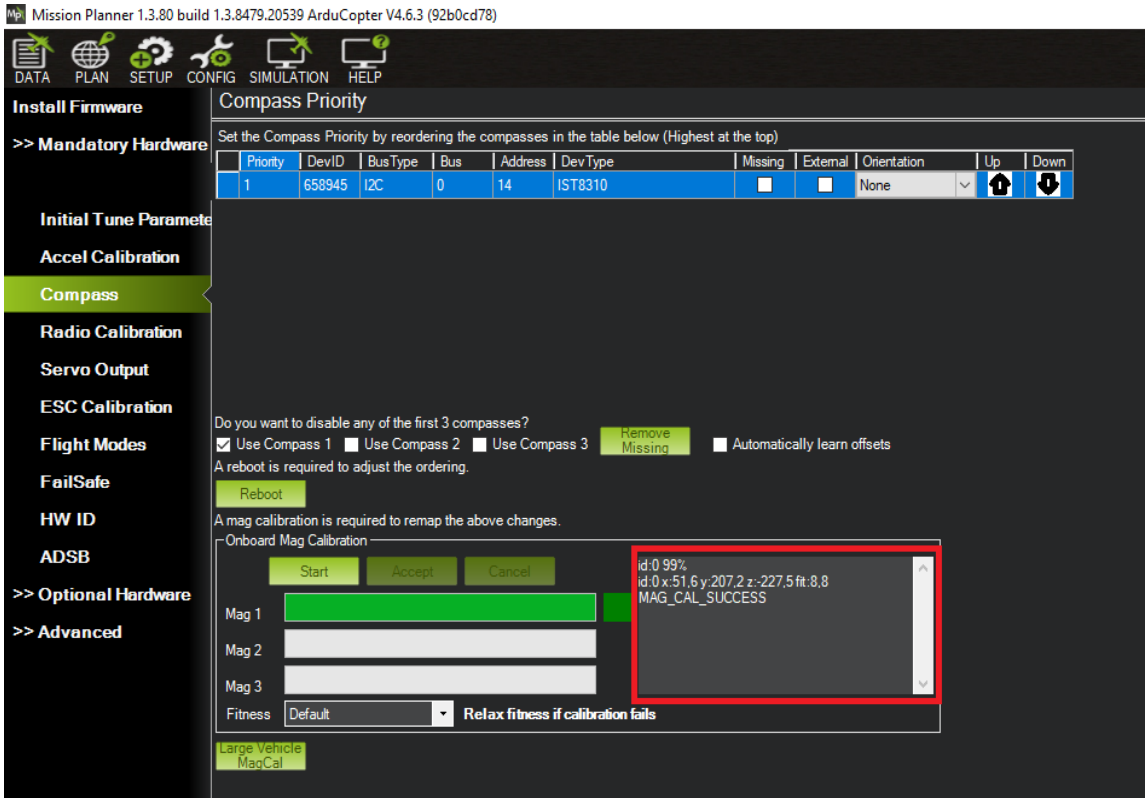


Σχήμα 4.7: Διαδικασία έναρξης βαθμονόμησης



Σχήμα 4.8: Σχηματική απεικόνιση των περιστρόφων για τη βαθμονόμηση της πυξίδας [36]

Η όλη διαδικασία στοχεύει στο να εντοπιστούν τυχόν αποκλίσεις στις μετρήσεις που μπορεί να προκαλούνται από μεταλλικά μέρη, ηλεκτρονικά κυκλώματα ή άλλες μαγνητικές παρεμβολές. Αφού ολοκληρωθεί η συλλογή των δεδομένων, το σύστημα υπολογίζει αυτόματα τις απαραίτητες διορθώσεις (offsets), ώστε οι μετρήσεις του μαγνητόμετρου να γίνουν όσο το δυνατόν πιο ακριβείς όπως μπορούμε να δούμε και από το σχήμα 4.9.



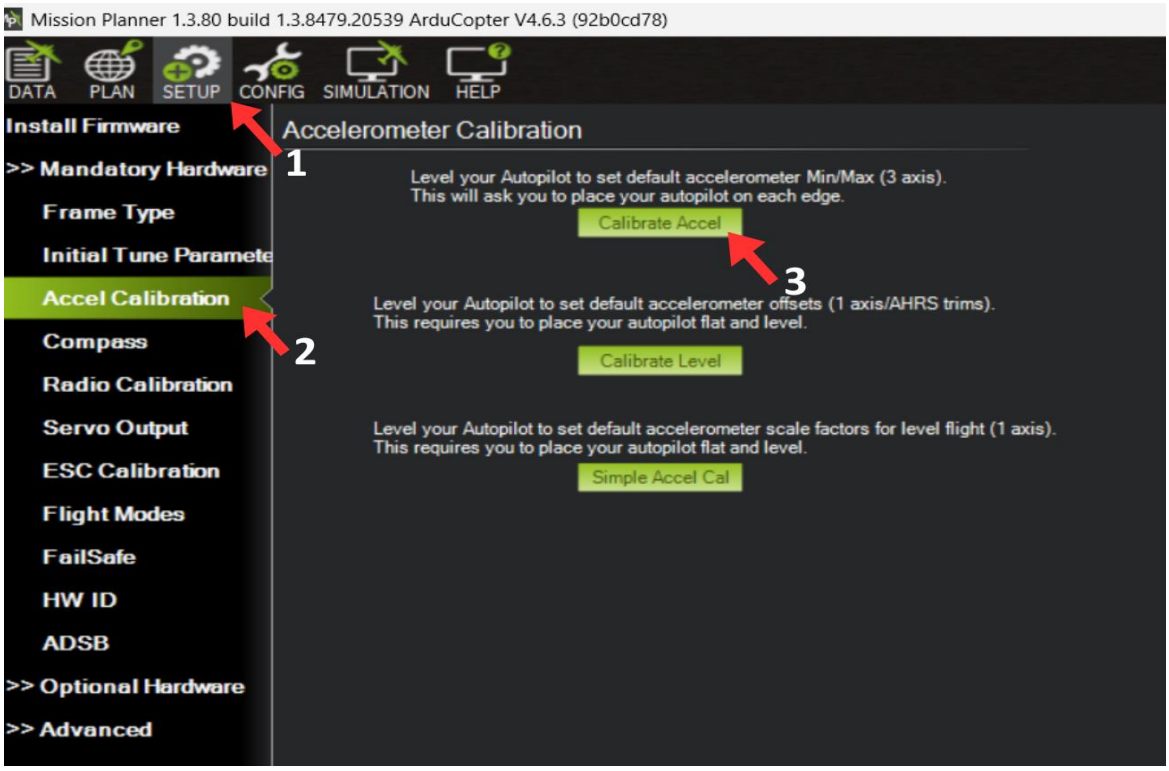
Σχήμα 4.9: Τελικό αποτέλεσμα των τιμών offsets μετά τη βαθμονόμηση

Αξίζει να σημειωθεί ότι η βαθμονόμηση της πυξίδας γίνεται κατά την αρχική ρύθμιση του συστήματος και είναι απαραίτητη πριν από την πρώτη πτήση. Σε ορισμένες περιπτώσεις χρειάζεται να επαναληφθεί για παράδειγμα όταν αλλάξει η θέση κάποιου εξαρτήματος ή όταν το drone χρησιμοποιηθεί σε περιοχή με διαφορετικά μαγνητικά χαρακτηριστικά. Η σωστή βαθμονόμηση εξασφαλίζει ακριβή προσανατολισμό και σταθερή πλοήγηση κατά την πτήση.

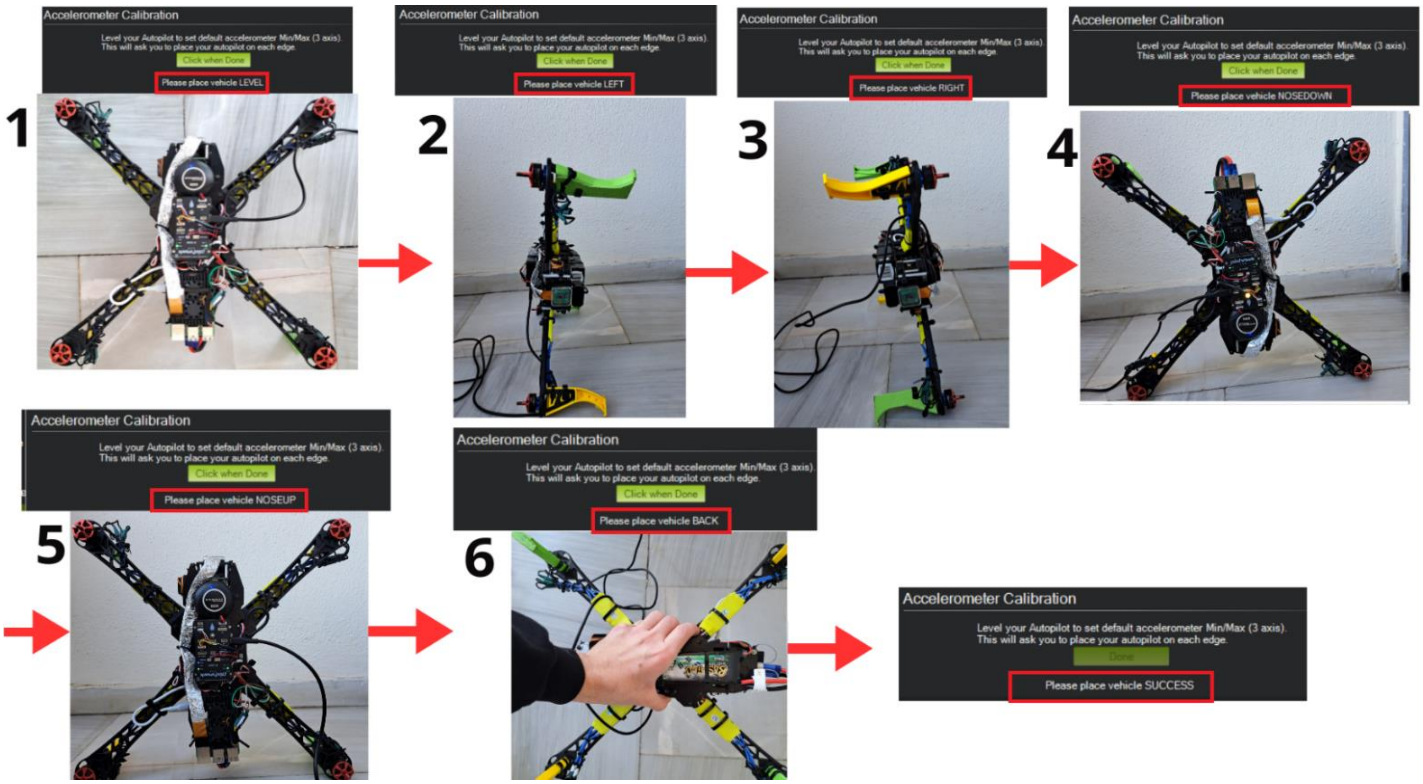
4.3.4 Βαθμονόμηση Επιταχυνσιόμετρου

Η βαθμονόμηση του επιταχυνσιόμετρου αποτελεί μια από τις σημαντικότερες ρυθμίσεις του συστήματος πτήσης. Ο συγκεκριμένος αισθητήρας μετρά τις επιταχύνσεις και βοηθά τον ελεγκτή πτήσης να αντιλαμβάνεται την κλίση και τη στάση του σκάφους κατά την πτήση.

Η διαδικασία βαθμονόμησης πραγματοποιείται μέσω της εφαρμογής Mission Planner από το μενού “SETUP” και την επιλογή “Calibrate Accel” όπως διακρίνεται από το σχήμα 4.10. Κατά την διάρκεια της διαδικασίας, το UAV τοποθετείται σε διάφορες θέσεις ώστε το σύστημα να καταγράψει τις μετρήσεις του αισθητήρα σε κάθε άξονα. Οι θέσεις αυτές περιλαμβάνουν την τοποθέτηση του σε επίπεδη θέση, στη δεξιά και αριστερή πλευρά, καθώς και με τη μύτη προς τα πάνω, προς τα κάτω και σε ανάποδη θέση όπως απεικονίζεται και στο σχήμα 4.11.



Σχήμα 4.10: Βήματα για την βαθμονόμηση



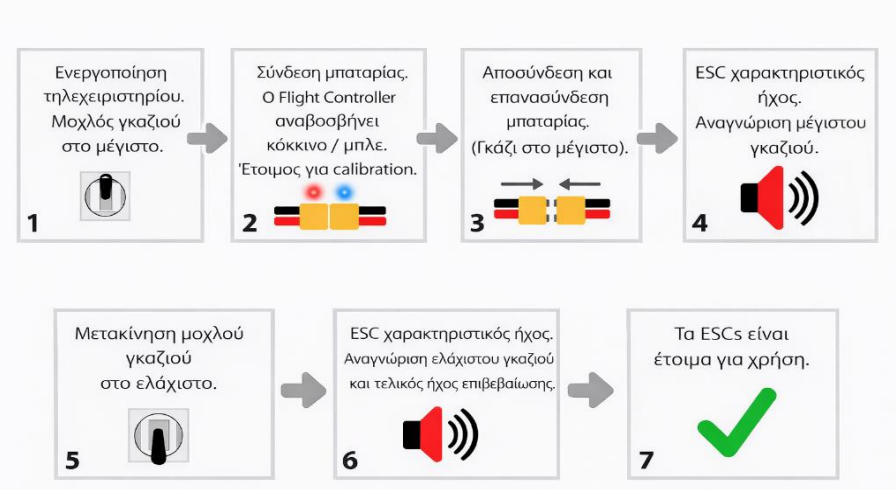
Σχήμα 4.11: Διαδικασία βαθμονόμησης του UAV

Σκοπός της διαδικασίας είναι να προσδιοριστούν με ακρίβεια οι τιμές αναφοράς του αισθητήρα ώστε να μειωθούν πιθανά σφάλματα στις μετρήσεις. Μετά την ολοκλήρωση της βαθμονόμησης, ο ελεγκτής πτήσης μπορεί να υπολογίζει σωστά την κλίση και τη στάση του αεροσκάφους συμβάλλοντας στη σταθερότητα και στην ασφαλή λειτουργία του κατά την πτήση.

4.3.5 Βαθμονόμηση ESC

Η βαθμονόμηση των ESC αποτελεί μια σημαντική διαδικασία κατά την αρχική εγκατάσταση των ηλεκτρονικών ελεγκτών ταχύτητας στο drone. Τα ESC ελέγχουν την ταχύτητα περιστροφής των κινητήρων λαμβάνοντας σήματα από τον ελεγκτή πτήσης. Για να λειτουργούν σωστά πρέπει να αναγνωρίζουν με ακρίβεια τα όρια του σήματος γκαζιού δηλαδή το ελάχιστο και μέγιστο σήμα που έρχεται από την τηλεκατεύθυνση. Η βαθμονόμηση πραγματοποιείται συνήθως την πρώτη φορά που τα ESC συνδέονται με τον ελεγκτή πτήσης ώστε όλοι οι κινητήρες να ανταποκρίνονται σωστά στις εντολές του χειριστή.

Η διαδικασία πραγματοποιείται με τη χρήση του τηλεχειριστήριου όπως παρουσιάζεται και στο σχήμα 4.12. Αρχικά ενεργοποιείται το τηλεχειριστήριο και ο μοχλός του γκαζιού τοποθετείται στη μέγιστη θέση. Στη συνέχεια συνδέεται η μπαταρία στο αεροσκάφος ώστε να ενεργοποιηθεί ο ελεγκτής πτήσης. Εκείνη τη στιγμή ο ελεγκτής πτήσης υποδεικνύει ότι το σύστημα είναι έτοιμο για τη διαδικασία βαθμονόμησης, με τα φωτάκια του να αναβοσβήνουν κόκκινο και μπλε. Έπειτα η μπαταρία αποσυνδέεται και επανασυνδέεται διατηρώντας τον μοχλό του γκαζιού (throttle) στη μέγιστη θέση. Μετά την ενεργοποίηση τα ESC εκπέμπουν έναν χαρακτηριστικό ήχο που υποδηλώνει ότι αναγνώρισαν το ανώτατο όριο σήματος γκαζιού (throttle). Στη συνέχεια, ο μοχλός του γκαζιού μετακινείται στην ελάχιστη θέση. Τα ESC αναγνωρίζουν το κατώτατο όριο και εκπέμπουν έναν τελικό ήχο επιβεβαίωσης, αποθηκεύοντας πλέον το συνολικό εύρος του σήματος.



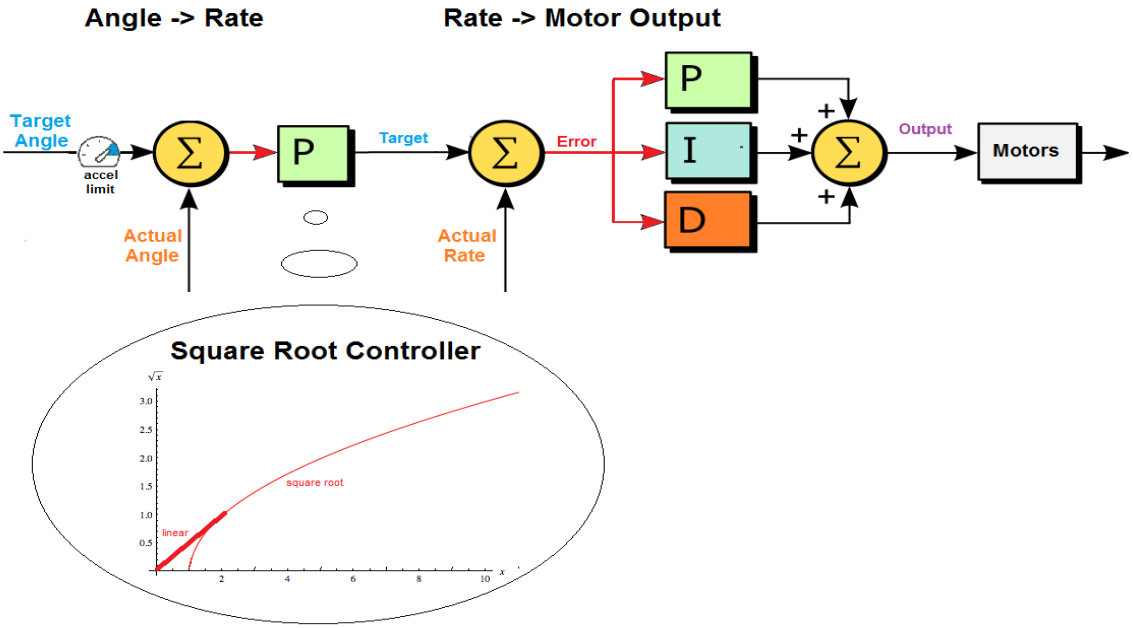
Σχήμα 4.12: Βήματα βαθμονόμησης ESC

Με την διαδικασία αυτή διασφαλίζεται ότι όλα τα ESC αντιλαμβάνονται με τον ίδιο τρόπο τις εντολές γκαζιού που προέχονται από τον ελεγκτή πτήσης. Έτσι, οι κινητήρες ξεκινούν και αυξάνουν τις στροφές τους ομοιόμορφα χωρίς αποκλίσεις μεταξύ τους.

4.3.6 Ρύθμιση Παραμέτρων PID

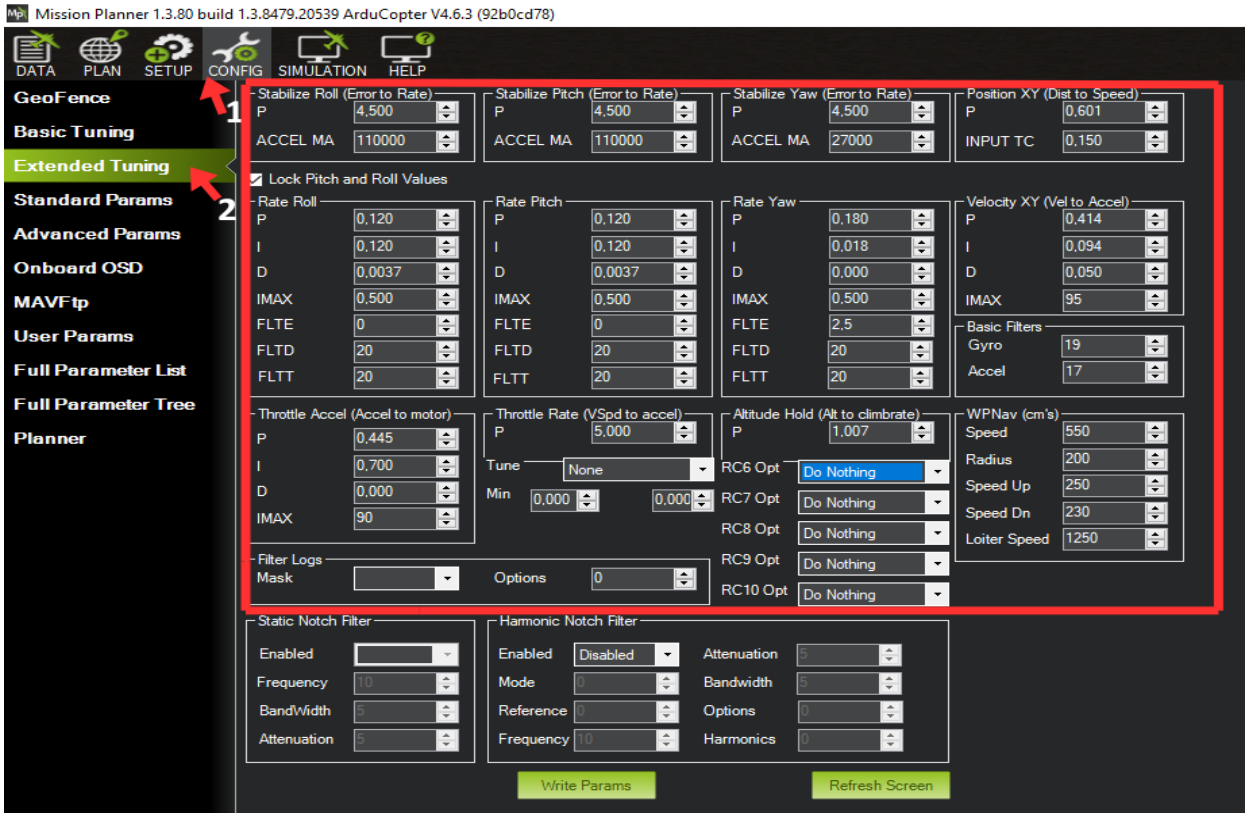
Η ρύθμιση των παραμέτρων PID (Proportional, Integral, and Derivative) αποτελεί ένα από τα σημαντικότερα στάδια παραμετροποίησης του ελεγκτή πτήσης καθώς επηρεάζει τη σταθερότητα και τη συμπεριφορά του drone κατά τη πτήση. Ο όρος P διορθώνει άμεσα το σφάλμα αυξάνοντας ή μειώνοντας την απόκριση ανάλογα με το μέγεθος της απόκλισης. Ο όρος I λαμβάνει υπόψη το σφάλμα που παραμένει με τη πάροδο του χρόνου και βοηθά στην εξάλειψη μικρών αποκλίσεων από την επιθυμητή θέση. Ο όρος D εξετάζει τον ρυθμό μεταβολής του σφάλματος και συμβάλλει στη μείωση των ταλαντώσεων, βελτιώνοντας τη συνολική σταθερότητα της πτήσης. Ο συνδυασμός των τριών όρων

οδηγεί σε πιο ομαλή και ελεγχόμενη συμπεριφορά του αεροσκάφους, όπως φαίνεται και από το σχήμα 4.13.



Σχήμα 4.13: Σχηματική απεικόνιση της λειτουργίας του PID [37]

Πιο συγκεκριμένα η ρύθμιση των παραμέτρων πραγματοποιείται μέσω της εφαρμογής Mission Planner στην καρτέλα “CONFIG” και επιλέγοντας το μενού “Extended Tuning”, όπου είναι διαθέσιμες οι βασικές παράμετροι ελέγχου του συστήματος. Κατά τις αρχικές δοκιμαστικές πτήσεις παρατηρήθηκε ότι το drone παρουσίαζε μικρές ταλαντώσεις και δεν παρέμενε σταθερό στον αέρα. Για τον λόγο αυτό πραγματοποιήθηκε προσαρμογή των βασικών παραμέτρων PID μέσα από δοκιμαστικές πτήσεις. Οι τιμές τροποποιούνταν σταδιακά μέχρι να επιτευχθεί πιο ομαλή απόκριση και καλύτερη σταθερότητα. Μετά από αρκετές δοκιμές επιλέχθηκαν οι κατάλληλες ρυθμίσεις, όπως φαίνεται και από το σχήμα 4.14, με αποτέλεσμα το αεροσκάφος να παρουσιάζει πιο σταθερή συμπεριφορά κατά την πτήση.



Σχήμα 4.14: Ρύθμιση παραμέτρων PID στο Mission Planner

4.4 Πτητικά modes

Τα πτητικά modes αποτελούν βασικό στοιχείο του λογισμικού ελέγχου πτήσης, καθώς καθορίζουν τον τρόπο με τον οποίο το drone ανταποκρίνεται στις εντολές του χειριστή. Κάθε mode αντιστοιχεί σε διαφορετικό τρόπο λειτουργίας, προσφέροντας επιλογές που κυμαίνονται από πλήρως χειροκίνητο έλεγχο έως υποβοηθούμενη σταθεροποίηση μέσω των αισθητήρων ή ακόμα και αυτόματη πλοήγηση. Μέσω των διαθέσιμων modes, ο χειριστής μπορεί να επιλέγει τον τρόπο με τον οποίο θα ελέγχεται η πτήση.

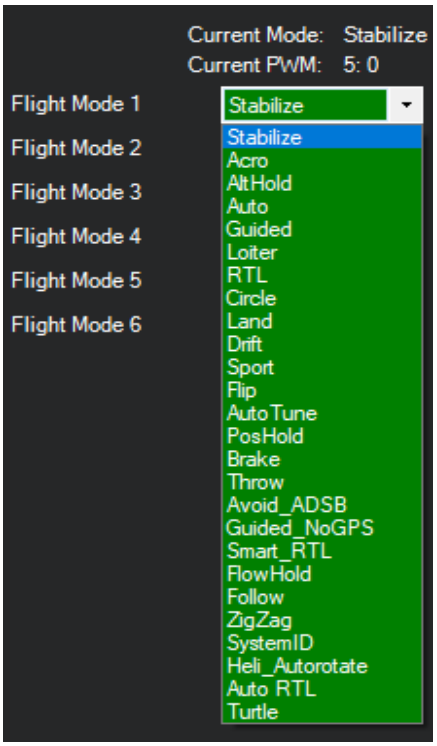
Στις υποενότητες που ακολουθούν, αρχικά παρουσιάζονται τα βασικά πτητικά modes που υποστηρίζει το ArduPilot. Έπειτα, αναλύεται η επιλογή των συγκεκριμένων modes που χρησιμοποιήθηκαν στην υλοποίηση. Τέλος, αναφέρεται και η διαδικασία παραμετροποίησης της τηλεκατεύθυνσης για την αντιστοίχιση και την εναλλαγή τους

4.4.1 Περιγραφή Διαθέσιμων Modes

Το λογισμικό ArduPilot μέσω του Mission Planner παρέχει μεγάλο αριθμό πτητικών modes που μπορούν να χρησιμοποιηθούν κατά τη λειτουργία ενός UAV, όπως διακρίνεται και στο σχήμα 4.15. Τα modes αυτά έχουν σχεδιαστεί ώστε να καλύπτουν διαφορετικές ανάγκες πτήσης και να προσφέρουν πολλαπλές δυνατότητες ελέγχου. Παρόλο που το σύστημα διαθέτει αρκετές επιλογές στην πράξη χρησιμοποιούνται συχνότερα ορισμένα βασικά modes τα οποία καλύπτουν τις πιο συνηθισμένες λειτουργίες ενός πολυκόπτερου drone.

Με βάση τη λειτουργία τους τα πτητικά modes του ArduPilot μπορούν να ομαδοποιηθούν σε τέσσερις βασικές κατηγορίες. Στις κατηγορίες αυτές περιλαμβάνονται διάφορα modes από τα οποία παρουσιάζονται στη συνέχεια ορισμένα από τα πιο συνηθισμένα :

1. Τα modes χειροκίνητου ελέγχου και σταθεροποίησης περιλαμβάνουν λειτουργίες στις οποίες ο χειριστής έχει τον κύριο έλεγχο της πτήσης, ενώ ο ελεγκτής πτήσης παρεμβαίνει μόνο για να προσφέρει βασική σταθεροποίηση. Σε αυτή την κατηγορία ανήκουν το Stabilize, όπου το σύστημα βοηθά στη διατήρηση της ισορροπίας του drone μέσω του IMU. Επιπλέον περιλαμβάνεται το AltHold το οποίο αξιοποιεί συνδυαστικά και το βαρόμετρο για να διατηρήσει το ύψος του σταθερό. Τα modes αυτά ενεργοποιούνται από τον χειριστή καθώς βασίζονται κυρίως στη δική του παρέμβαση.
2. Τα modes ημιαυτόνομης πτήσης και διατήρησης θέσης αποτελούν την κατηγορία στην οποία το σύστημα αναλαμβάνει μέρος της πτήσης, ενώ ο χειριστής εξακολουθεί να κατευθύνει το UAV. Για τη σταθεροποίηση, το σύστημα βασίζεται στην IMU και αξιοποιεί επιπλέον δεδομένα από το GPS, την πυξίδα και το βαρόμετρο. Σε αυτή την κατηγορία περιλαμβάνεται το Loiter στο οποίο ο χειριστής ελέγχει την κίνηση του drone μέσω του τηλεχειριστηρίου, ενώ το σύστημα διατηρεί αυτόματα τη θέση και το ύψος του στον χώρο όταν οι μοχλοί αφεθούν ελεύθεροι. Παρόμοια λειτουργία παρουσιάζει και το PosHold το οποίο προσφέρει στον χειριστή πιο άμεσο και γρήγορο έλεγχο της κλίσης του drone. Τα modes αυτά ενεργοποιούνται από τον χειριστή μέσω της τηλεκατεύθυνσης και χρησιμοποιούνται κυρίως για σταθερή πτήση και καλύτερο έλεγχο του αεροσκάφους.
3. Τα modes αυτόνομης πλοήγησης αποτελούν την κατηγορία στην οποία το αεροσκάφος μπορεί να εκτελεί πτήση χωρίς παρέμβαση από τον χειριστή. Σε αυτά τα modes το σύστημα αξιοποιεί δεδομένα από αισθητήρες όπως το GPS, την πυξίδα το βαρόμετρο και το IMU ώστε να πραγματοποιεί αυτόνομα τις λειτουργίες πλοήγησης. Σε αυτή την κατηγορία περιλαμβάνεται το Auto, το οποίο εκτελεί αυτόματα μια προγραμματισμένη αποστολή πτήσης με βάση προκαθορισμένα σημεία (waypoints). Επιπλέον, υπάρχει το Guided, το οποίο επιτρέπει τον έλεγχο του UAV μέσω εντολών που αποστέλλονται από εξωτερικά υπολογιστικά συστήματα, όπως για παράδειγμα ένα Raspberry Pi. Τα modes αυτά μπορούν να ενεργοποιηθούν είτε από τον χειριστή είτε από εξωτερικό υπολογιστικό σύστημα, ανάλογα με την εφαρμογή.
4. Τα modes ασφαλείας που έχουν σχεδιαστεί να προστατεύουν το UAV και να εξασφαλίζουν ότι η πτήση ολοκληρώνεται με ασφάλεια σε απρόβλεπτες συνθήκες. Σε αυτή την κατηγορία περιλαμβάνονται λειτουργίες όπως το RTL (Return to Launch), που επιτρέπει στο αεροσκάφος να επιστρέφει αυτόματα στο σημείο απογείωσης. Έπειτα έχουμε το Land το οποίο εκτελεί αυτόματη προσγείωση, καθώς και το Brake που σταματά την οποιαδήποτε κίνηση του drone στον αέρα. Τα modes αυτά μπορούν να ενεργοποιηθούν είτε από τον χειριστή είτε αυτόματα από το σύστημα σε περίπτωση απώλειας σήματος ή χαμηλής στάθμης μπαταρίας.



Σχήμα 4.15: Λίστα πτητικών modes του ArduPilot στο Mission Planner

4.4.2 Επιλογή Modes για το Σύστημα

Με βάση τα διαθέσιμα πτητικά modes που παρουσιάστηκαν στην προηγούμενη ενότητα για την υλοποίηση του συστήματος επιλέχθηκε η χρήση συγκεκριμένων modes. Η επιλογή αυτή πραγματοποιήθηκε ώστε να επιτρέπεται τόσο η σταθερή πτήση του drone όσο και η δυνατότητα ελέγχου του από εξωτερικό υπολογιστικό σύστημα, καθώς και η ασφαλής διαχείρισή του σε περιπτώσεις εκτατής ανάγκης.

Το βασικό mode λειτουργίας του συστήματος είναι το Loiter το οποίο χρησιμοποιείται για την πλοήγηση του UAV, προσφέροντας σταθερότητα και ευκολία ελέγχου στο χειριστή.

Επιπλέον χρησιμοποιείται και το Guided mode όταν απαιτείται αποστολή εντολών πλοήγησης από εξωτερικό υπολογιστικό σύστημα. Στην παρούσα υλοποίηση το mode αυτό ενεργοποιείται αυτόματα μέσω επικοινωνίας MAVLink από το Raspberry Pi, επιτρέποντας την αποστολή εντολών προς τον ελεγκτή πτήσης και τον έλεγχο της πορείας του UAV.

Ακόμα, έχουν ενσωματωθεί τα modes Stabilize και RTL για χρήση σε περιπτώσεις χειροκίνητης παρέμβασης ή έκτακτης ανάγκης.

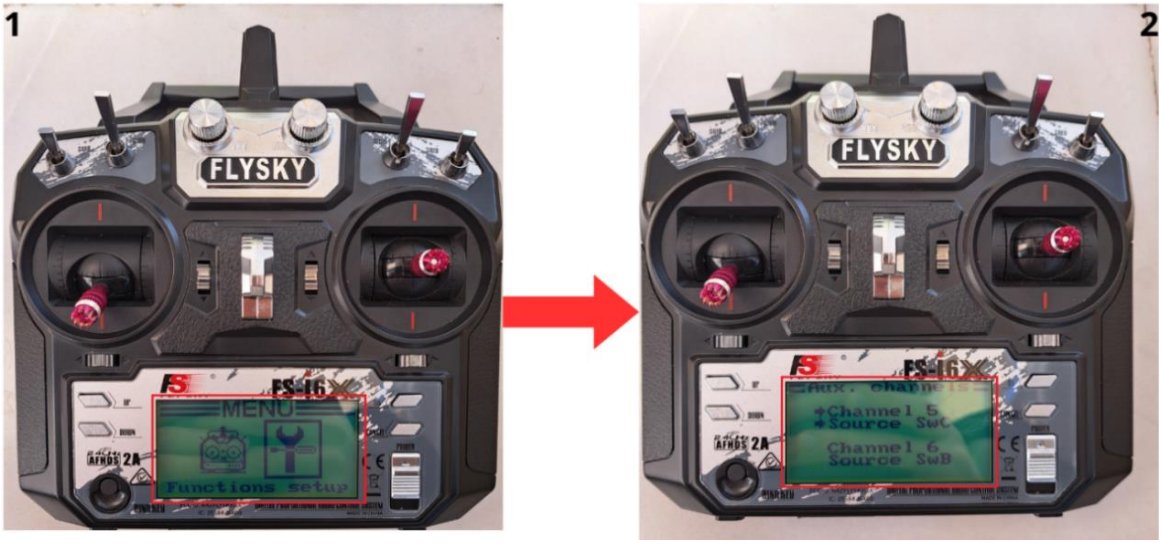
4.4.3 Διαμόρφωση και αντίστοιχη πτητικών modes στο τηλεχειριστήριο

Για την εναλλαγή των πτητικών modes έγιναν οι απαραίτητες ρυθμίσεις στο τηλεχειριστήριο Flysky FS-i6x ώστε να προσαρμοστεί στις ανάγκες του συστήματος. Πιο συγκεκριμένα, η τηλεκατεύθυνση (FS-iA6B) διαθέτει έξι κανάλια, με τα τέσσερα πρώτα (Channels 1-4) να χρησιμοποιούνται για τον βασικό έλεγχο της πτήσης. Έτσι, τα δύο κανάλια που απομένουν channel 5 και channel 6 χρησιμοποιούνται για την επιλογή των πτητικών modes, αντιστοιχίζοντας τα σε δύο από τους διαθέσιμους διακόπτες. Για τον σκοπό αυτό, επιλέχθηκαν οι διακόπτες SwB (Switch B) δύο θέσεων και

SwC (Switch C) τριών θέσεων, η διάταξη των οποίων φαίνεται στο σχήμα 4.16, προκειμένου να χρησιμοποιηθούν συνδυαστικά για την εναλλαγή των modes.

Σχήμα 4.16: Διακόπτες SwB και SwC

Για να μπορέσουν οι διακόπτες να χρησιμοποιηθούν για την αλλαγή των πτητικών modes πραγματοποιήθηκε η απαραίτητη παραμετροποίηση μέσα από το μενού της τηλεκατεύθυνσης. Αρχικά, από το βασικό μενού του τηλεχειριστηρίου επιλέχθηκε το Functions setup και αμέσως μετά το Aux. channels. Στο Aux Channels έγινε η επιλογή και η αντιστοίχιση των διακοπτών στα αντίστοιχα κανάλια. Για το Channel 5 ορίστηκε ο διακόπτης SwC, ενώ για το Channel 6 επιλέχθηκε ο διακόπτης SwB, όπως φαίνεται και στο σχήμα 4.17.



Σχήμα 4.17: Αντιστοίχιση των διακοπτών SwC και SwB στα κανάλια 5 και 6

Στη συνέχεια μέσα από το μενού Function Setup επιλέχθηκε η ρύθμιση End points όπου τροποποιήθηκαν τα όρια του Channel 5. Η τιμή μειώθηκε από το προεπιλεγμένο 100% στο 60% και για το ανώτερο και για το κατώτερο όριο, όπως παρουσιάζεται και στο σχήμα 4.18. Η ρύθμιση αυτή

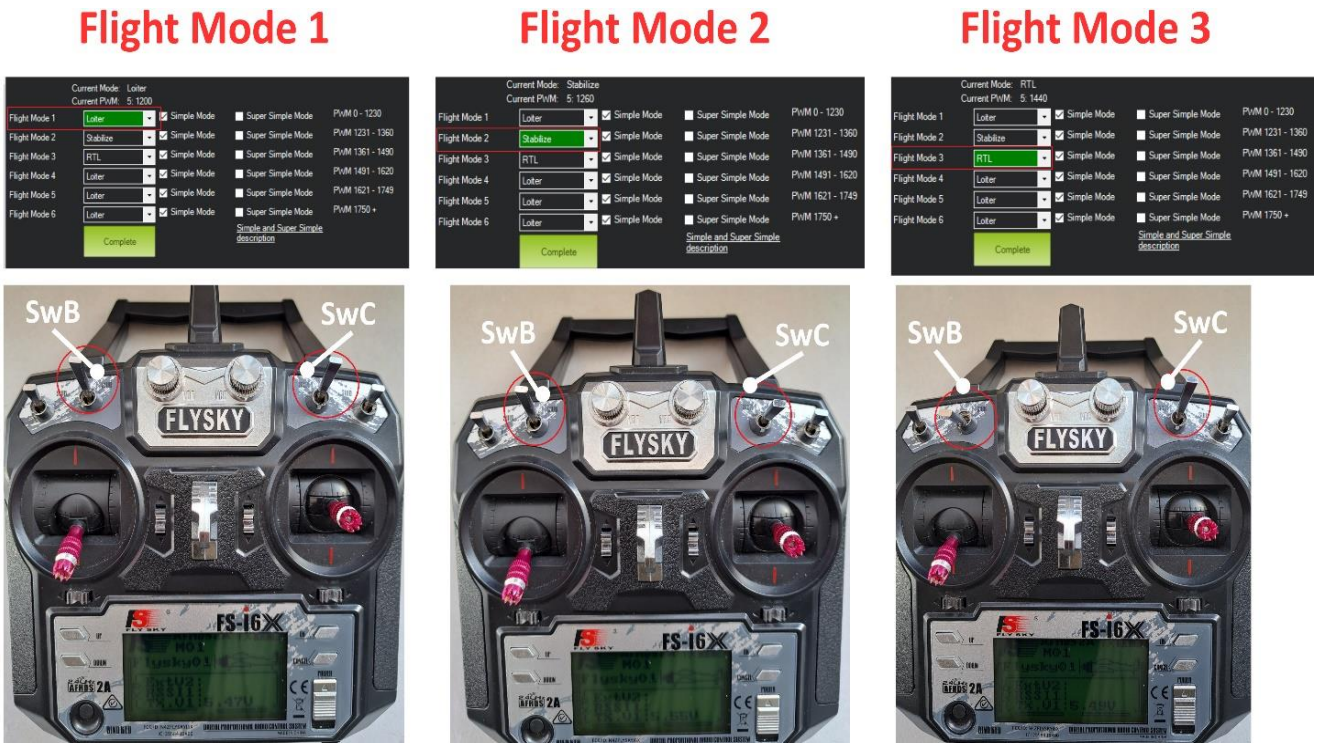
πραγματοποιήθηκε ώστε να περιοριστεί το εύρος του σήματος του καναλιού, έτσι ώστε οι τιμές του να παραμένουν εντός των επιτρεπτών ορίων κατά την εφαρμογή της μίξης που ακολουθεί.

Σχήμα 4.18: Ρύθμιση των End Points του καναλιού 5 στο 60%

Έπειτα, και πάλι μέσα από το Function Setup επιλέχθηκε το Mixes όπου ενεργοποιήθηκε το Mix 1. Στη συγκεκριμένη ρύθμιση ορίστηκε ως Master το channel 6 και Slave το channel 5, με ποσοστό μίξης 80% τόσο για τη θετική όσο και για την αρνητική κατεύθυνση. Το ποσοστό 80% επιλέχθηκε ώστε να επιτευχθεί η απαραίτητη μετατόπιση του σήματος PWM ανάμεσα στα δύο κανάλια. Με αυτόν τον τρόπο ο διακόπτης δύο θέσεων channel 6 επηρεάζει το σήμα του channel 5 στο οποίο είναι συνδεδεμένος ο διακόπτης τριών θέσεων. Έτσι εξασφαλίζεται ότι οι τελικές τιμές που λαμβάνει ο ελεγκτής πτήσης παραμένουν μέσα στα προκαθορισμένα όρια που απαιτεί το ArduPilot για την αναγνώριση και την εναλλαγή των modes. Αξίζει να σημειωθεί ότι η συγκεκριμένη παραμετροποίηση επιτρέπει τη δημιουργία έως και έξι διαφορετικών πτητικών modes μέσω του συνδυασμού των δύο διακοπτών. Παρ' όλα αυτά, για τις ανάγκες της παρούσας υλοποίησης αξιοποιήθηκαν τρία από αυτά.

Σχήμα 4.19: Ρύθμιση της μίξης (Mix1)

Μετά την ολοκλήρωση των παραπάνω ρυθμίσεων τα τρία αυτά modes αντιστοιχιστήκαν μέσω της εφαρμογής Mission Planner. Συγκεκριμένα όταν ο διακόπτης SwC βρίσκεται στην επάνω θέση και ο SwB επίσης στη πάνω θέση ενεργοποιείται το Flight Mode 1 δηλαδή το Loiter . Όταν ο SwC μετακινηθεί στη μεσαία θέση και ο SwB και αυτός στην επάνω θέση τότε ενεργοποιείται το Flight Mode 2 δηλαδή το Stabilize. Έπειτα, όταν ο SwC παραμένει στην επάνω θέση και ο SwB μετακινηθεί στην κάτω θέση ενεργοποιείται το Flight Mode 3 δηλαδή το RTL. Η αντιστοίχιση αυτή φαίνεται και στο σχήμα 4.20.



Σχήμα 4.20: Τρόπος ενεργοποίησης των modes

Κλείνοντας αξίζει να σημειωθεί ότι το Guided mode δεν αντιστοιχίστηκε σε κάποιο flight mode καθώς ενεργοποιείται από το Raspberry Pi μέσω MAVlink. Ο τρόπος ενεργοποίησης του mode αυτού θα παρουσιαστεί στα επόμενα κεφάλαια.

4.5 Προετοιμασία Υπολογιστικού Συστήματος

Για την υλοποίηση του συστήματος αποφυγής εμποδίου απαιτήθηκε η κατάλληλη προετοιμασία του υπολογιστικού συστήματος. Η προετοιμασία αυτή είναι κρίσιμη καθώς επιτρέπει στο Raspberry Pi να αναγνωρίζει και να διαβάζει τα δεδομένα του αισθητήρα και της κάμερας, καθώς και να επικοινωνεί αξιόπιστα με τον ελεγκτή πτήσης.

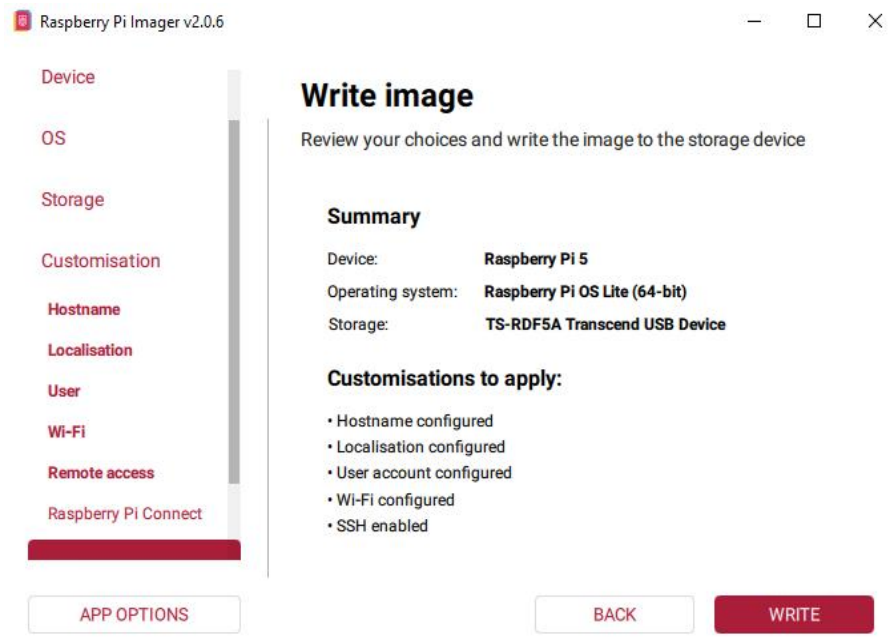
Στις υποενότητες που ακολουθούν, αναλύονται τα βασικά στάδια αυτής της προετοιμασίας. Αρχικά περιγράφεται η εγκατάσταση του λειτουργικού συστήματος Raspberry Pi OS, το οποίο παρέχει το βασικό περιβάλλον για τη λειτουργία του Raspberry Pi. Στη συνέχεια παρουσιάζεται η εγκατάσταση των απαραίτητων βιβλιοθηκών που χρησιμοποιούνται για την ανάπτυξη του αλγορίθμου και την λήψη δεδομένων από τον αισθητήρα και την κάμερα. Έπειτα, παρουσιάζεται η ρύθμιση των σειριακών θυρών για την επικοινωνία του Raspberry Pi με τον ελεγκτή πτήσης και τον αισθητήρα. Τέλος, αναλύεται η ρύθμιση της επικοινωνίας MAVLink, η οποία επιτρέπει την αποστολή και λήψη εντολών και δεδομένων μεταξύ του υπολογιστικού συστήματος και του ελεγκτή πτήσης.

4.5.1 Εγκατάσταση Λειτουργικού Raspberry Pi OS

Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, ως υπολογιστικού συστήματος χρησιμοποιήθηκε το Raspberry Pi 5 με μνήμη RAM 8GB σε συνδυασμό με κάρτα microSD χωρητικότητας 128GB. Η εγκατάσταση του λειτουργικού συστήματος πραγματοποιήθηκε μέσω της εφαρμογής Raspberry Pi Imager, όπως φαίνεται και στο σχήμα 4.21, με την οποία έγινε η εγγραφή του Raspberry Pi OS Lite στην κάρτα μνήμης. Το συγκεκριμένο λειτουργικό βασίζεται στο Linux και είναι σχεδιασμένο ειδικά για το Raspberry Pi. Η έκδοση Lite δεν περιλαμβάνει γραφικό περιβάλλον κάτι που δεν απαιτείται για τις ανάγκες της παρούσας υλοποίησης. Η απουσία γραφικού περιβάλλοντος μειώνει σημαντικά την κατανάλωση πόρων του συστήματος κάτι ιδιαίτερα σημαντικό για το Raspberry Pi. Με αυτόν τον τρόπο διατίθεται περισσότερη υπολογιστική ισχύς για απαιτητικές διεργασίες, όπως η εκτέλεση μοντέλων ανίχνευσης αντικειμένων.

Η διαδικασία εγκατάσταση του λειτουργικού συστήματος ολοκληρώθηκε μέσα από μια σειρά βασικών βημάτων:

1. Επιλογή του μοντέλου Raspberry Pi μέσα από την εφαρμογή Raspberry Pi Imager.
2. Επιλογή του λειτουργικού συστήματος, οπού διαθέτει διάφορες εκδόσεις βασισμένες στο Linux, με ή χωρίς γραφικό περιβάλλον.
3. Επιλογή της συσκευής αποθήκευσης, δηλαδή της κάρτας microSD στην οποία θα εγγραφεί το λειτουργικό σύστημα.
4. Ρύθμιση βασικών παραμέτρων, όπως το όνομα της συσκευής (hostname), τα στοιχεία χρήστη και κωδικού πρόσβασης, καθώς και η σύνδεση σε ασύρματο δίκτυο WiFi. Σε αυτό το στάδιο ενεργοποιείται και η ρύθμιση SSH (Secure Shell) ώστε να είναι εφικτή η απομακρυσμένη διαχείριση του Raspberry Pi.
5. Τέλος, η κάρτα SD εισάγεται στο Raspberry Pi και πραγματοποιείται η εγκατάσταση. Μετά την ολοκλήρωση της διαδικασίας γίνεται το boot του συστήματος.



Σχήμα 4.21: Desktop application Raspberry Pi Imager

Μετά την εγκατάσταση του λειτουργικού συστήματος, το Raspberry Pi κρίθηκε έτοιμο για τις επόμενες ρυθμίσεις και την εγκατάσταση των βιβλιοθηκών που απαιτούνται για τη λειτουργία του.

4.5.2 Βασικές Ρυθμίσεις

Το πρώτο βήμα της παραμετροποίησης είναι η σύνδεση στο Raspberry Pi, ώστε να πραγματοποιηθούν οι βασικές ρυθμίσεις του συστήματος. Με την ολοκλήρωσή τους, το περιβάλλον είναι πλέον έτοιμο για την εγκατάσταση των βιβλιοθηκών που θα χρησιμοποιηθούν στην υλοποίηση της εφαρμογής Η σύνδεση στο Raspberry Pi πραγματοποιήθηκε μέσω του πρωτοκόλλου SSH το οποίο επιτρέπει την απομακρυσμένη διαχείριση του συστήματος μέσω τοπικού δικτύου. Για να είναι δυνατή η σύνδεση πρέπει να ικανοποιούνται ορισμένες βασικές προϋποθέσεις :

1. Η γνώση του ονόματος χρήστη και του κωδικού πρόσβασης που ορίστηκαν κατά την εγκατάσταση του λειτουργικού συστήματος.
2. Η αναγνώριση της συσκευής στο τοπικό δίκτυο, η οποία στην παρούσα υλοποίηση επιτεύχθηκε μέσω του τοπικού ονόματος της συσκευής.
3. Το Raspberry Pi και η συσκευή από την οποία γίνεται η σύνδεση πρέπει να βρίσκονται στο ίδιο τοπικό δίκτυο.

Αφού ικανοποιηθούν οι παραπάνω προϋποθέσεις, η σύνδεση πραγματοποιείται μέσω της ακόλουθης εντολής στο τερματικό:

```
ssh aitos@raspberrypi.local
```

Με την εκτέλεση της εντολής ζητείται ο κωδικός πρόσβασης του χρήστη και μόλις αυτός εισαχθεί πραγματοποιείται η απομακρυσμένη πρόσβαση στο σύστημα του Raspberry Pi.

Μετά την επιτυχή σύνδεση στο σύστημα ακολουθεί η ενημέρωση του μέσω του εργαλείου διαχείρισης πακέτων APT (Advanced Package Tool):

```
aitos@raspberrypi:~ $ sudo apt update && sudo apt upgrade -y
```

Η διαδικασία αυτή είναι απαραίτητη ώστε το λειτουργικό σύστημα να διαθέτει τις πιο πρόσφατες ενημερώσεις και τις νεότερες βιβλιοθήκες.

Στη συνέχεια, εγκαταστάθηκε η βιβλιοθήκη rpicam, η οποία είναι απαραίτητη για να μπορεί το Raspberry Pi να επικοινωνεί με την κάμερα που έχει τοποθετηθεί σε αυτό. Το βήμα αυτό είναι σημαντικό καθώς η κάμερα θα χρησιμοποιηθεί αργότερα για την ανίχνευση αντικειμένων στον αλγόριθμο αποφυγής εμποδίων. Η εγκατάσταση πραγματοποιήθηκε με την εντολή:

```
aitos@raspberrypi:~ $ sudo apt install rpicam-apps
```

Η επαλήθευση ότι η κάμερα λειτουργεί σωστά πραγματοποιήθηκε με την εκτέλεση της εντολής:

```
aitos@raspberrypi:~ $ rpicam-hello --list
Available cameras
-----------------
0 : imx219 [3280x2464 10-bit RGGB] (/base/axi/pcie@1000120000/rp1/i2c@88000/imx219@10)
    Modes: 'SRGGB10_CSI2P' : 640x480 [103.33 fps - (1000, 752)/1280x960 crop]
                             1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
                             1920x1080 [47.57 fps - (680, 692)/1920x1080 crop]
                             3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
           'SRGGB8' : 640x480 [103.33 fps - (1000, 752)/1280x960 crop]
                      1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
                      1920x1080 [47.57 fps - (680, 692)/1920x1080 crop]
                      3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
```

Έπειτα, δημιουργήθηκε ο βασικός φάκελος του έργου με την ονομασία drone_ai μέσα στον οποίο θα αποθηκευτούν τα scripts για την υλοποίηση του έργου. Η δημιουργία και η πρόσβαση στον φάκελο πραγματοποιήθηκαν με τις εντολές:

```
aitos@raspberrypi:~ $ mkdir ~/drone_ai && cd ~/drone_ai
aitos@raspberrypi:~/drone_ai $
```

Αφού δημιουργήθηκε ο φάκελος του έργου, ρυθμίστηκε ένα εικονικό περιβάλλον Python μέσω της εντολής:

```
aitos@raspberrypi:~/drone_ai $ python3 -m venv venv
```

Η ενεργοποίηση του πραγματοποιήθηκε με την εντολή:

```
aitos@raspberrypi:~/drone_ai $ source venv/bin/activate
(venv) aitos@raspberrypi:~/drone_ai $
```

Το εικονικό περιβάλλον χρησιμοποιείται ώστε οι υπόλοιπες βιβλιοθήκες που απαιτεί η υλοποίηση να εγκαθίστανται σε έναν απομονωμένο χώρο, χωρίς να επηρεάζουν τις βιβλιοθήκες του λειτουργικού συστήματος. Με αυτόν τον τρόπο εξασφαλίζεται καλύτερη οργάνωση του έργου και μεγαλύτερη σταθερότητα κατά την ανάπτυξη.

Αρχικά, στο εικονικό περιβάλλον εγκαταστάθηκαν οι βιβλιοθήκες που απαιτούνται για την επικοινωνία με τον ελεγκτή πτήσης, μέσω της εντολής:

```
(venv) aitos@raspberrypi:~/drone_ai $ pip install pymavlink MAVProxy
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pymavlink
  Using cached pymavlink-2.4.49-cp313-cp313-manylinux2014_aarch64.manylinux_2_17_aarch64.manylinux_2_28_aarch64.whl.metadata (6.0 kB)
Collecting MAVProxy
  Downloading https://www.piwheels.org/simple/mavproxy/mavproxy-1.8.74-py3-none-any.whl (28.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 28.8/28.8 MB 2.6 MB/s eta 0:00:00
Collecting lxml (from pymavlink)
  Using cached lxml-6.0.2-cp313-cp313-manylinux_2_26_aarch64.manylinux_2_28_aarch64.whl.metadata (3.6 kB)
Collecting fastcrc (from pymavlink)
  Downloading fastcrc-0.3.5-cp313-cp313-manylinux_2_17_aarch64.manylinux2014_aarch64.whl.metadata (2.5 kB)
Collecting pyserial>=3.0 (from MAVProxy)
  Downloading https://www.piwheels.org/simple/pyserial/pyserial-3.5-py2.py3-none-any.whl (90 kB)
Collecting numpy (from MAVProxy)
  Downloading numpy-2.4.3-cp313-cp313-manylinux_2_27_aarch64.manylinux_2_28_aarch64.whl.metadata (6.6 kB)
Collecting pynmeagps (from MAVProxy)
  Downloading https://www.piwheels.org/simple/pynmeagps/pynmeagps-1.1.2-py3-none-any.whl (62 kB)
Using cached pymavlink-2.4.49-cp313-cp313-manylinux2014_aarch64.manylinux_2_17_aarch64.manylinux_2_28_aarch64.whl (6.4 MB)
Downloading fastcrc-0.3.5-cp313-cp313-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (282 kB)
Using cached lxml-6.0.2-cp313-cp313-manylinux_2_26_aarch64.manylinux_2_28_aarch64.whl (5.0 MB)
Downloading numpy-2.4.3-cp313-cp313-manylinux_2_27_aarch64.manylinux_2_28_aarch64.whl (15.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.7/15.7 MB 3.2 MB/s eta 0:00:00
Installing collected packages: pyserial, pynmeagps, numpy, lxml, fastcrc, pymavlink, MAVProxy
Successfully installed MAVProxy-1.8.74 fastcrc-0.3.5 lxml-6.0.2 numpy-2.4.3 pymavlink-2.4.49 pynmeagps-1.1.2 pyserial-3.5
```

Οι υπόλοιπες βιβλιοθήκες που απαιτούνται για τη λειτουργία του συστήματος θα εγκατασταθούν στο ίδιο περιβάλλον και θα παρουσιαστούν αναλυτικά στα επόμενα κεφάλαια.

4.5.3 Ρύθμιση Σειριακής Επικοινωνίας

Για την επικοινωνία του Raspberry Pi με τις εξωτερικές συσκευές που έχουν τοποθετηθεί, δηλαδή τον ελεγκτή πτήσης και τον αισθητήρα LiDAR, απαιτείται η ενεργοποίηση της σειριακής επικοινωνίας (UART) μέσω των ρυθμίσεων του λειτουργικού, εκτελώντας την εντολή:

```
aitos@raspberrypi:~ $ sudo raspi-config
```

Μέσα από το περιβάλλον του εργαλείου επιλέχθηκε η ρύθμιση Serial Port, όπως φαίνεται και στο σχήμα 4.22 και στη συνέχεια πραγματοποιήθηκε η ενεργοποίηση της .

```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

  I1 SSH         Enable/disable remote command line access using SSH
  I2 RPi Connect Enable/disable Raspberry Pi Connect
  I3 VNC         Enable/disable graphical remote desktop access
  I4 SPI         Enable/disable automatic loading of SPI kernel module
  I5 I2C         Enable/disable automatic loading of I2C kernel module
  I6 Serial Port Enable/disable shell messages on the serial connection
  I7 1-Wire      Enable/disable one-wire interface

                  <Select>                    <Back>
```

Σχήμα 4.22: Ενεργοποίηση Serial Port

Για τη λειτουργία του αισθητήρα LiDAR, ο οποίος συνδέθηκε στους ακροδέκτες 33 και 34 του Raspberry Pi, απαιτήθηκε η ενεργοποίηση της αντίστοιχης σειριακής θύρας UART4. Αυτό είναι απαραίτητο επειδή, στο λειτουργικό σύστημα του Raspberry Pi, πέρα από τη βασική σειριακή θύρα, οι υπόλοιπες παραμένουν απενεργοποιημένες και πρέπει να ενεργοποιηθούν από τον χρήστη.

Η ενεργοποίηση της συγκεκριμένης θύρας πραγματοποιήθηκε μέσω τροποποίησης του αρχείου ρυθμίσεων του συστήματος:

```
aitos@raspberrypi:~ $ sudo nano /boot/firmware/config.txt
```

Εντός του αρχείου προστέθηκε η παράμετρος:

```
dtoverlay=uart4
```

ώστε να ενεργοποιηθεί η αντίστοιχη σειριακή διεπαφή για το LiDAR. Από την άλλη, για τον ελεγκτή πτήσης ο οποίος είναι συνδεδεμένος στη βασική σειριακή θύρα του Raspberry Pi δεν χρειάστηκε επιπλέον ενεργοποίηση κάποιας νέας διεπαφής. Η συγκεκριμένη θύρα είναι ήδη ενεργοποιημένη από το σύστημα.

Στο ίδιο αρχείο πραγματοποιήθηκε επίσης η απενεργοποίηση του Bluetooth με την προσθήκη της παραμέτρου:

```
  GNU nano 8.4                                  /boot/firmware/config.txt
arm_64bit=1

# Disable compensation for displays with overscan
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[cm5]
dtoverlay=dwc2,dr_mode=host

[all]
dtparam=uart0=on
enable_uart=1
dtoverlay=uart0
dtoverlay=disable-bt
core_freq_min=250
dtoverlay=uart4
```

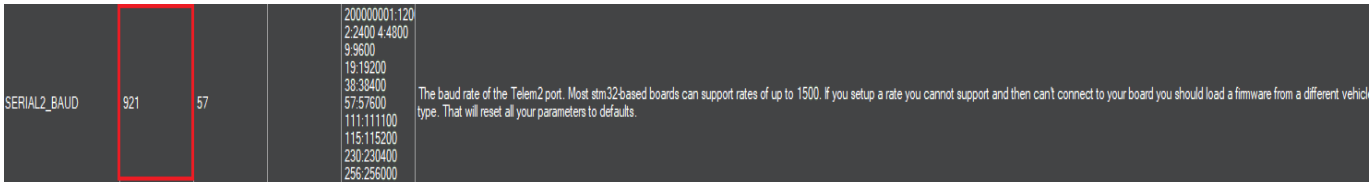
Η απενεργοποίηση του Bluetooth είναι σημαντική, καθώς χρησιμοποιεί επίσης σειριακή επικοινωνία και ενδέχεται να προκαλέσει παρεμβολές. Με αυτόν τον τρόπο διασφαλίζεται ότι οι UART που χρησιμοποιούνται από το σύστημα λειτουργούν σταθερά και αξιόπιστα.

4.5.4 Ρύθμιση Επικοινωνίας Pixhwak με Raspberry Pi

Αφού ολοκληρώθηκε η σύνδεση του ελεγκτή πτήσης με το Raspberry Pi στις αντίστοιχες θύρες UART και εγκαταστάθηκαν οι απαραίτητες βιβλιοθήκες MAVLink και MAVProxy, ακολούθησε η ρύθμιση της μεταξύ τους επικοινωνίας, ώστε οι δύο συσκευές να μπορούν να ανταλλάσσουν δεδομένα.

Αρχικά, για την προετοιμασία της επικοινωνίας του ελεγκτή πτήσης με το Raspberry Pi, απαιτήθηκε η κατάλληλη ρύθμιση μέσω της εφαρμογής Mission Planner. Συγκεκριμένα ο ελεγκτής ρυθμίστηκε ώστε να ανταλλάσσει δεδομένα με ρυθμό μετάδοσης 921600 bps, όπως φαίνεται και στο σχήμα 4.23. Η ταχύτητα αυτή επιλέχθηκε ως η βέλτιστη λύση καθώς χαμηλότερες ταχύτητες όπως τα 115200bps είναι ανεπαρκείς για τον όγκο των δεδομένων και θα οδηγήσουν σε καθυστέρηση κατά τη μεταφορά εντολών. Από την άλλη μεγαλύτερες τιμές από το 921600 bps αποφεύχθηκαν καθώς είναι πιθανό να εμφανιστεί

ηλεκτρομαγνητικός θόρυβος γεγονός που μπορεί να προκαλέσει αλλοίωση ή απώλεια πακέτων δεδομένων.



Σχήμα 4.23: Ρύθμιση baudrate μέσω της εφαρμογής Mission Planner

Η σειριακή επικοινωνία ανάμεσα στον ελεγκτή και στο Raspberry Pi μπορεί να χρησιμοποιηθεί μόνο από μια διεργασία κάθε φορά. Αυτό σημαίνει ότι αν ένα script τη χρησιμοποιεί δεν μπορεί να την ανοίξει ταυτόχρονα κάποια άλλη διεργασία. Επειδή όμως στην υλοποίησή μας χρειάζεται να τρέχουν δύο scripts ταυτόχρονα κρίθηκε απαραίτητο να υπάρχει ένας τρόπος ώστε η σειριακή θύρα να μοιράζει σωστά τα δεδομένα σε κάθε διαδικασία που τα χρειάζεται. Για τον σκοπό αυτό, αξιοποιήθηκε το MAVProxy, το οποίο λειτουργεί ως ενδιάμεσος (proxy), λαμβάνοντας τα δεδομένα από τη φυσική σειριακή θύρα και προωθώντας τα σε πολλαπλές δικτυακές θύρες.

Προκειμένου το MAVProxy να είναι συνεχώς διαθέσιμο και να μην απαιτείται χειροκίνητη εκκίνηση, ρυθμίστηκε ως υπηρεσία συστήματος (service), ώστε να εκτελείται αυτόματα στο παρασκήνιο με την εκκίνηση του λειτουργικού συστήματος. Η διαδικασία πραγματοποιήθηκε με τη δημιουργία του αρχείου:

```
aitos@raspberrypi:~ $ sudo nano /etc/systemd/system/mavproxy.service
```

Το περιεχόμενο του αρχείου διαμορφώθηκε ως εξής:

```
 GNU nano 8.4                                  /etc/systemd/system/mavproxy.service
[Unit]
Description=Mavproxy Router Service
After=network.target

[Service]
Type=simple
User=aitos
WorkingDirectory=/home/aitos
ExecStart=/usr/local/bin/mavproxy.py --daemon --master=/dev/serial0 --baudrate 921600 --out=udp:127.0.0.1:14550 --out=udp:127.0.0.1:14551 --aircraft MyDrone
Restart=always
RestartSec=1

[Install]
WantedBy=multi-user.target
```

Στο παραπάνω αρχείο περιλαμβάνονται όλες οι ρυθμίσεις λειτουργίας. Η δομή του χωρίζεται σε τρεις βασικές ενότητες:

1. Ενότητα [Unit]

   Η παράμετρος After=network.target εξασφαλίζει ότι το MAVProxy θα ξεκινήσει μόνο αφού το Raspberry Pi έχει ενεργοποιήσει το εσωτερικό δίκτυο. Αυτό είναι απολύτως αναγκαίο επειδή η μεταφορά των δεδομένων γίνεται μέσω δικτυακών θυρών UDP.

2. Ενότητα [Service]

   Η παράμετρος Type=simple δηλώνει στο σύστημα ότι η διεργασία εκτελείται απευθείας, χωρίς να χρειάζεται να δημιουργήσει ή να τρέξει κάποια άλλη διεργασία.

   Συνεχίζουμε με τις παραμέτρους User και WorkingDirectory οπού ορίζουν ότι η υπηρεσία εκτελείται με δικαιώματα του απλού χρήστη και όχι ως root.

   Έπειτα συνεχίζεται η γραμμή με το ExecStart οπού δίνει την εντολή να τρέξει το MAVProxy, ορίζοντας παράλληλα του κανόνες λειτουργίας του. Η επιλογή --daemon δίνει τη δυνατότητα στο πρόγραμμα αν τρέχει παρασκήνιο. Με αυτόν τον τρόπο το MAVProxy δεν περιμένει χειροκίνητες εντολές από εμάς μέσω του πληκτρολογίου αλλά λειτουργεί ως μεσολαβητής. Λαμβάνει τα δεδομένα από το Pixhawk και τα προωθεί είτε στην τηλεμετρία είτε στο script αποφυγής εμποδίων, ενώ ταυτόχρονα μπορεί να στέλνει πίσω στον ελεγκτή εντολές. Η παράμετρος --master διαβάζει τα δεδομένα από τη σειριακή θύρα. Η παράμετρος –baudrate ρυθμίζει την ταχύτητα που επικοινωνούν ο Raspberry Pi και Pixhawk. Η τιμή αυτή πρέπει να είναι ίδια με αυτή που ορίστηκε στο Mission Planner. Με αυτόν τον τρόπο το Raspberry Pi και ο ελεγκτής πτήσης επικοινωνούν με την ίδια ταχύτητα και δεν χάνονται δεδομένα. Με την εντολή --out δημιουργούνται δύο έξοδοι στην τοπική διεύθυνση 127.0.0.1. Η πρώτη θύρα 14550 και η δεύτερη 14551 επιτρέπουν έτσι σε δύο διαφορετικά scripts να συνδέονται ταυτόχρονα στον Pixhawk για να ανταλλάσσουν δεδομένα. Τέλος, το --aircraft αποθηκεύει αυτόματα τα αρχεία της πτήσης σε έναν φάκελο.

3. Ενότητα [Install]

   Η παράμετρος WantedBy=multi-user.target κάνει την υπηρεσία να ξεκινά αυτόματα μόλις ανοίξει το Raspberry Pi.

Αφού ρυθμίστηκε το αρχείο απομένει η ενεργοποίηση της υπηρεσίας μέσω του systemctl CLI εργαλείου. Η διαδικασία πραγματοποιείται με τις παρακάτω εντολές:

```
aitos@raspberrypi:~ $ sudo systemctl enable mavproxy.service
aitos@raspberrypi:~ $ sudo systemctl daemon-reload
```

Συνοψίζοντας, το MAVProxy ρυθμίστηκε να λειτουργεί αυτόματα στο παρασκήνιο, λαμβάνοντας τα δεδομένα από τον ελεγκτή πτήσης και διαμοιράζοντάς τα σε δύο ανεξάρτητες θύρες. Έτσι, το σύστημα είναι έτοιμο να επικοινωνήσει ταυτόχρονα με τα δύο scripts που θα αναλυθούν στα επόμενα κεφάλαια.

Κεφάλαιο 5ο: Αλγόριθμος Αποφυγής Εμποδίων

5.1 Γενική Περιγραφή Αλγορίθμου Αποφυγής

Ο αλγόριθμος αποφυγής εμποδίων σχεδιάστηκε με κύριο σκοπό την αποτροπή συγκρούσεων κατά τη διάρκεια της πτήσης. Το σύστημα παρακολουθεί συνεχώς τον χώρο μπροστά από το μη επανδρωμένο όχημα μέσω του αισθητήρα LiDAR. Ο αισθητήρας αυτός παρέχει μετρήσεις απόστασης σε πραγματικό χρόνο.

Μόλις ανιχνευτεί κάποιο αντικείμενο κοντά στο drone, ενεργοποιείται αυτόματα η διαδικασία αποφυγής εμποδίων. Αρχικά, το drone στρέφει ελαφρώς την κατεύθυνσή του (yaw) προς τα δεξιά και προς τα αριστερά προκειμένου να ελέγξει για ελεύθερο χώρο. Αφού εντοπιστεί η κατάλληλη κατεύθυνση το drone δεν κινείται απευθείας προς τον ελεύθερο χώρο. Αντίθετα, στρέφεται ολόκληρο προς την πλευρά όπου εντόπισε τον ελεύθερο χώρο. Στη συνέχεια, κινείται παράλληλα με το εμπόδιο για κάποια προκαθορισμένα μέτρα που έχουν οριστεί στον αλγόριθμο. Έπειτα σταματάει και περιστρέφεται αντίστροφα για να επανέλθει στην αρχική του κατεύθυνση και ελέγχει ξανά τον χώρο μπροστά του. Εάν ο δρόμος είναι ελεύθερος, συνεχίζει την πορεία του ευθεία ώστε να προσπεράσει το εμπόδιο. Αν, ωστόσο διαπιστωθεί ότι υπάρχει ακόμα εμπόδιο μπροστά του και δεν μπορεί να προχωρήσει ευθεία το σύστημα επαναλαμβάνει ολόκληρη την αναζήτηση ελεύθερου χώρου από την αρχή.

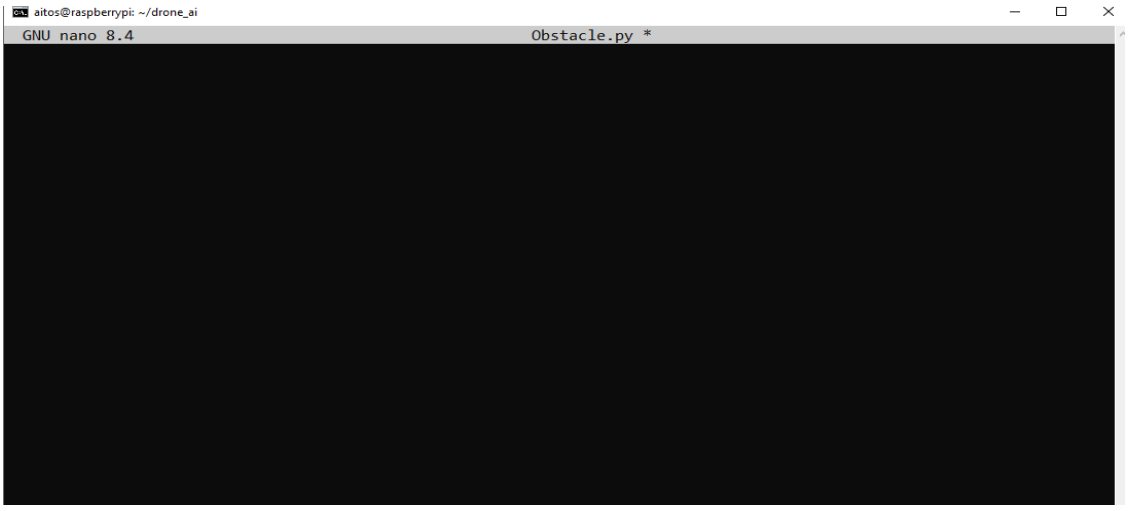
Επίσης, σε περίπτωση που κατά τη διαδικασία αναζήτησης ελεύθερου χώρου δεν εντοπιστεί επαρκές άνοιγμα για ασφαλή πορεία, ο αλγόριθμος επαναλαμβάνει την προσπάθεια. Το drone πραγματοποιεί μια μικρή κίνηση προς τα πίσω ώστε να αποκτήσει μεγαλύτερο οπτικό πεδίο και στη συνέχεια επαναλαμβάνει τον έλεγχο, αυξάνοντας το εύρος της σάρωσης σε κάθε προσπάθεια. Εάν, μετά από έναν προκαθορισμένο αριθμό διαδοχικών προσπαθειών εξακολουθεί να μην εντοπίζεται κατάλληλος ελεύθερος χώρος ενεργοποιείται η διαδικασία υποχώρησης. Σε αυτή τη φάση το drone απομακρύνεται προς τα πίσω, υποδεικνύοντας στον χειριστή ότι η αυτόματη αποφυγή του εμποδίου δεν είναι εφικτή. Το ίδιο ακριβώς ισχύει όταν το σκάφος επιχειρεί να κινηθεί παράλληλα με το εμπόδιο. Μόλις ο αισθητήρας διαπιστώσει απουσία ελεύθερου χώρου, το drone γυρίζει στην αρχική του κατεύθυνση, απομακρύνεται προς τα πίσω και ξεκινά μια νέα σάρωση.

Παράλληλα, μέσω ανίχνευσης αντικειμένων με χρήση τεχνητής νοημοσύνης το σύστημα αναγνωρίζει την παρουσία ανθρώπου. Όταν εντοπιστεί ανθρώπινη παρουσία, μειώνεται η ταχύτητα του UAV και αυξάνεται η απόσταση από το εμπόδιο. Έτσι, η αποφυγή γίνεται πιο ομαλά και με μεγαλύτερη ασφάλεια.

Για την υλοποίηση του παραπάνω αλγορίθμου κρίθηκε απαραίτητη η δημιουργία ενός script σε γλώσσα Python το οποίο αναλαμβάνει αποκλειστικά τον έλεγχο της αποφυγής εμποδίων. Το συγκεκριμένο αρχείο δημιουργήθηκε στον κατάλογο drone_ai του Raspberry Pi δίνοντας το όνομα Obstacle με κατάληξη py η οποία υποδηλώνει αρχείο κώδικα Python. Στον συγκεκριμένο φάκελο βρίσκεται ρυθμισμένο και το εικονικό περιβάλλον, ώστε το αρχείο να μπορεί να διαβάζει σωστά τις εγκατεστημένες βιβλιοθήκες. Η δημιουργία του πραγματοποιήθηκε μέσω του επεξεργαστή κειμένου nano εκτελώντας στο τερματικό την εντολή:

```
(venv) aitos@raspberrypi:~/drone_ai $ nano Obstacle.py
```

Με την εκτέλεση της παραπάνω εντολής ανοίγει το περιβάλλον του επεξεργαστή κειμένου nano, όπως φαίνεται στο σχήμα 5.1.



Σχήμα 5.1: Επεξεργαστής κειμένου nano

Στο συγκεκριμένο script συγκεντρώθηκαν όλες οι συναρτήσεις που αφορούν την ανάγνωση δεδομένων από τον LiDAR, την επεξεργασία εικόνας από την κάμερα, καθώς και την λογική λήψης αποφάσεων για την αποφυγή εμποδίων. Με αυτόν τον τρόπο επιτυγχάνεται η ενοποίηση όλων των λειτουργιών σε ένα ενιαίο σύστημα ελέγχου.

Στις επόμενες ενότητες αναλύεται η υλοποίηση του αλγορίθμου σε επίπεδο κώδικα. Αρχικά, αναλύεται η επικοινωνία του raspberry pi με το ελεγκτή πτήσης. Έπειτα, περιγράφεται ο τρόπος με τον οποίο χρησιμοποιείται ο αισθητήρας LiDAR για την ανίχνευση εμποδίων. Στη συνέχεια παρουσιάζεται η λειτουργία της κάμερας για την οπτική αναγνώριση. Τέλος, αναλύεται το πώς συνδυάζονται όλα τα επιμέρους μέρη στη συνολική εκτέλεση του αλγορίθμου.

5.2 Αρχικοποίηση Επικοινωνίας με τον Ελεγκτή Πτήσης

Για να πραγματοποιηθεί η επικοινωνία του ελεγκτή με το Raspberry Pi το πρώτο βήμα είναι η εισαγωγή της βιβλιοθήκης pymavlink στο script. Η προσθήκη αυτή πραγματοποιείται στις πρώτες γραμμές του αρχείου, όπως φαίνεται στο σχήμα 5.2, και παρέχει τα απαραίτητα εργαλεία για τη σύνδεση που ακολουθεί.

Σχήμα 5.2: Εισαγωγή βιβλιοθήκης στο script

Αφού οριστεί η βιβλιοθήκη, ακολουθεί η διαδικασία αρχικοποίησης της σύνδεσης. Όπως φαίνεται στο σχήμα 5.3, αρχικά ορίζεται η μεταβλητή connection_string με διεύθυνση UDP 127.0.0.1 και θύρα 14550 μέσω της οποίας όπως αναφέρθηκε σε προηγούμενο κεφάλαιο επιτρέπεται η αποστολή και λήψη δεδομένων από το MAVProxy. Στη συνέχεια καλείται η mavlink_connection η οποία αναλαμβάνει την έναρξη της επικοινωνίας μεταξύ των συσκευών. Επιπλέον καλείται η μέθοδος wait_heartbeat η οποία θέτει το πρόγραμμα σε αναμονή μέχρι να λάβει ένα αρχικό σήμα από τον ελεγκτή πτήσης επιβεβαιώνοντας έτσι ότι είναι ενεργοποιημένος. Ταυτόχρονα μέσα από αυτή την επικοινωνία αποθηκεύονται αυτόματα οι τιμές στο System ID και Component ID. Το System ID εκφράζει την

ταυτότητα ολόκληρου του drone ενώ το Component ID προσδιορίζει το εξάρτημα δηλαδή τον ελεγκτή. Η αποθήκευσή τους είναι απαραίτητη, καθώς εξασφαλίζει ότι οι μελλοντικές εντολές του αλγορίθμου θα κατευθυνθούν στον Pixhawk για να εκτελεστούν. Τέλος, χρησιμοποιείται και η εντολή print για να εμφανίζονται οι συγκεκριμένες αναγνωριστικές τιμές στην οθόνη.

```
# Connection to Pixhawk
connection_string = "udpin:127.0.0.1:14550"
print("Connecting to Pixhawk...")
m = mavutil.mavlink_connection(connection_string) # Start connection
m.wait_heartbeat() # Wait for signal
print("Heartbeat from System ID", m.target_system, "Component ID", m.target_component)
```

Σχήμα 5.3: Αρχικοποίηση της επικοινωνίας με τον ελεγκτή πτήσης

5.3 Ανίχνευση Εμποδίου με Χρήση του Αισθητήρα LiDAR

Όπως αναφέρθηκε στη γενική περιγραφή του αλγορίθμου το LiDAR αποτελεί την κύρια πηγή δεδομένων για την ανίχνευση και την αποφυγή εμποδίων. Στην παρούσα ενότητα παρουσιάζεται πρακτικά πώς αυτή η λειτουργία υλοποιείται στον κώδικα, από τη λήψη των μετρήσεων του αισθητήρα μέχρι την τελική εκτέλεση των ελιγμών αποφυγής.

Παρακάτω αναλύεται βήμα προς βήμα η προγραμματιστική υλοποίηση αυτής της διαδικασίας. Αρχικά, περιγράφεται η ενσωμάτωση του LiDAR εξηγώντας τον τρόπο που συνδέεται με το σύστημα ώστε να διαβάζει τα δεδομένα. Στη συνέχεια, αναλύεται ο τρόπος με τον οποίο ενεργοποιείται η αποφυγή εμποδίου. Έπειτα περιγράφεται πώς παίρνει την απόφαση ο αλγόριθμος ώστε να βρει την κατάλληλη κατεύθυνση, προκειμένου να μην συγκρουστεί με το αντικείμενο. Ακόμα, εξηγείται ο τρόπος με τον οποίο εκτελείται ο ελιγμός αποφυγής, ώστε το όχημα να απομακρυνθεί από το εμπόδιο με ασφάλεια. Τέλος, παρουσιάζεται η διαδικασία υποχώρησης, η οποία ενεργοποιείται όταν δεν εντοπίζεται ελεύθερος χώρος κατά τη σάρωση.

5.3.1 Ενσωμάτωση LiDAR στο Κώδικα

Για την επικοινωνία του συστήματος με τον αισθητήρα LiDAR χρησιμοποιήθηκε η βιβλιοθήκη pyserial. Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, η εγκατάστασή της πραγματοποιήθηκε στο εικονικό περιβάλλον που δημιουργήθηκε μέσω της εντολής:

```
(venv) aitos@raspberrypi:~/drone_ai $ pip install pyserial
```

Αφού ολοκληρώθηκε η εγκατάσταση, η βιβλιοθήκη serial προστέθηκε στο script ώστε να είναι δυνατή η πρόσβαση στη σειριακή θύρα και η λήψη των δεδομένων του αισθητήρα. Παράλληλα χρησιμοποιείται και η ενσωματωμένη βιβλιοθήκη struct η οποία είναι ήδη εγκατεστημένη στη Python. Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για τη μετατροπή των δυαδικών δεδομένων του LiDAR σε αριθμητικές τιμές, ώστε να μπορούν να επεξεργαστούν από τον αλγόριθμο. Η εισαγωγή των παραπάνω βιβλιοθηκών πραγματοποιήθηκε στις πρώτες γραμμές του αρχείου, όπως φαίνεται και στο σχήμα 5.4.

```
aitos@raspberrypi: ~/drone_ai
 GNU nano 8.4                                    Obstacle.py *
import struct
import serial
```

Σχήμα 5.4: Εισαγωγή των βιβλιοθηκών στο script

Αμέσως μετά ορίζονται οι βασικές παράμετροι λειτουργίας του LiDAR. Συγκεκριμένα ορίζεται η σειριακή θύρα που είναι συνδεδεμένος ο LiDAR καθώς και ο ρυθμός μετάδοσης δεδομένων στα 115200 baud. Η τιμή αυτή επιλέχθηκε επειδή προσφέρει γρήγορη και σταθερή μεταφορά δεδομένων, κάτι απαραίτητο για την άμεση αποφυγή εμποδίων. Στη συνέχεια προστίθεται η εντολή serial.Serial η οποία ανοίγει την επικοινωνία με τον αισθητήρα και η σύνδεση αποθηκεύεται στην μεταβλητή ser. Ταυτόχρονα, ορίζεται ένας χρόνος αναμονής 0,2 sec ώστε σε περίπτωση που χαθεί προσωρινά το σήμα, το πρόγραμμα να μην σταματήσει περιμένοντας δεδομένα, αλλά να επιτρέψει στο σύστημα να μεταβεί σε μια ασφαλή κατάσταση. Αμέσως μετά καλείται η συνάρτηση ser.reset_input_buffer(), η οποία καθαρίζει την προσωρινή μνήμη (buffer) ώστε το σύστημα να μην διαβάσει παλιές μετρήσεις κατά την εκκίνηση του script. Η υλοποίηση των παραπάνω ρυθμίσεων φαίνεται στο σχήμα 5.5.

```
# LiDAR
TF_PORT = '/dev/ttyAMA4'
TF_BAUD = 115200
ser = serial.Serial(TF_PORT, TF_BAUD, timeout=0.2)
ser.reset_input_buffer() #Clear buffer
```

Σχήμα 5.5: Διασύνδεση του αισθητήρα LiDAR με το Raspberry Pi

Αφού ορίστηκαν οι βασικές ρυθμίσεις επικοινωνίας, ακολουθεί η υλοποίηση των συναρτήσεων που διαχειρίζονται τα δεδομένα του αισθητήρα. Η πρώτη από αυτές είναι η συνάρτηση read_tfmini(), η οποία αναλαμβάνει την ανάγνωση και αποκωδικοποίηση των δεδομένων του αισθητήρα.

Στο σχήμα 5.6 παρουσιάζεται η υλοποίησή της ενώ η διαδικασία εκτελείται ως εξής:

1. Μέσω του βρόχου while διαβάζονται τα εισερχόμενα bytes. Ελέγχεται αν τα δύο πρώτα bytes αντιστοιχούν στην τιμή \x59 που δηλώνει την αρχή ενός έγκυρου πακέτου. Σε περίπτωση που δεν εντοπιστεί η τιμή \x59 η διαδικασία επαναλαμβάνεται. Όταν όμως παρέλθει το χρονικό όριο που έχει οριστεί, ο βρόχος while τερματίζεται και η συνάρτηση επιστρέφει κενή τιμή None.
2. Όταν εντοπιστεί σωστό πακέτο διαβάζονται τα 7 bytes (payload) τα οποία περιέχουν την βασική πληροφορία της μέτρησης. Σε αυτά περιλαμβάνονται η απόσταση, η ισχύς σήματος, η θερμοκρασία της συσκευής και το άθροισμα ελέγχου (checksum). Κάθε τιμή μεταδίδεται σε δύο bytes το Least significant και το Most significant byte. Αυτό συμβαίνει επειδή ένα μόνο byte μπορεί να αναπαραστήσει τιμές έως το 255. Επομένως, για να είναι εφικτή η καταγραφή μετρήσεων που ξεπερνούν αυτό το όριο, τα δύο αυτά μέρη συνδυάζονται ώστε να προκύψει ο τελικός αριθμός. Έπειτα τα δεδομένα αποκωδικοποιούνται με τη struct.unpack() σε αριθμητικές τιμές του δεκαδικού συστήματος. Στη συνέχεια πραγματοποιείται έλεγχος εγκυρότητας (checksum) ώστε να διασφαλιστεί ότι τα δεδομένα μεταφέρθηκαν σωστά. Στην υλοποίηση αξιοποιείται μόνο η απόσταση ενώ οι υπόλοιπες τιμές διαβάζονται αλλά δεν χρησιμοποιούνται.
3. Τέλος τα bytes της απόστασης (dL, dM) του LiDAR ενώνονται μέσω της πράξης (dM << 8 | dL) ώστε να προκύψει ο πλήρης ακέραιος αριθμός της μέτρησης σε εκατοστά. Στη συνέχεια η τιμή διαιρείται με το 100 για να μετατραπεί σε μέτρα, ώστε ο κώδικας να δουλεύει με μια ενιαία και πιο εύχρηστη μονάδα.

```python
def read_tfmini(max_time=0.3):
    start = time.time() # timer for timeout check
    while time.time() - start < max_time:
        b = ser.read(1)
        if not b: continue
        # Check for valid packet x59
        if b != b'\x59': continue
        b2 = ser.read(1)
        if b2 != b'\x59': continue
        # Read 7 bytes
        payload = ser.read(7)
        if len(payload) != 7: return None
        # Decode data
        dL, dM, sL, sM, tL, tM, checksum = struct.unpack('<BBBBBBB', payload)
        frame = b'\x59\x59' + payload
        if (sum(frame[:8]) & 0xFF) != checksum: return None
        # Union of distance bytes
        dist_cm = (dM << 8) | dL
        # Convert the meters
        return dist_cm / 100.0
    return None
```

Σχήμα 5.6: Συνάρτηση ανάγνωσης και αποκωδικοποίηση δεδομένων του αισθητήρα

Επιπλέον, ακολουθεί η συνάρτηση avg_distance() η οποία λειτουργεί σαν φίλτρο για τις μετρήσεις του αισθητήρα. Σκοπός της είναι η συλλογή πολλαπλών δειγμάτων και ο υπολογισμός του μέσου ορού τους ώστε να αποφευχθούν λανθασμένες ένδειξης.

Στο σχήμα 5.7 παρουσιάζεται η υλοποίηση της αξιολόγησης και του φιλτραρίσματος των μετρήσεων, κατά την οποία εκτελούνται τα εξής βήματα:

1. Η συνάρτηση αρχικοποιεί μια λίστα για την αποθήκευση των μετρήσεων και καταγραφεί τον χρόνο έναρξης. Έπειτα, ακολουθεί καθαρισμός του buffer της σειριακής θύρας, ώστε να ληφθούν μόνο πρόσφατα δεδομένα από τον αισθητήρα.

2. Μέσω του βρόχου while συλλέγονται διαδοχικές μετρήσεις καλώντας τη συνάρτηση read_tfmini(). Ο βρόχος ελέγχει δυο συνθήκες εάν έχει συλλεχθεί ο απαιτούμενος αριθμός εγκύρων δειγμάτων και εάν ο συνολικός χρόνος εκτέλεσης βρίσκεται εντός του επιτρεπτού ορίου. Αν ο μέγιστος χρόνος αναμονής ξεπεραστεί ο βρόχος τερματίζεται αποτρέποντας το ενδεχόμενο εγκλωβισμού του κώδικα.

3. Για κάθε έγκυρη μέτρηση που λαμβάνεται, πραγματοποιείται έλεγχος της τιμής. Στην περίπτωση που η απόσταση είναι 0, η τιμή αντικαθίσταται με τη μέγιστη εμβέλεια του αισθητήρα 12 μέτρα. Αυτό συμβαίνει διότι, σε ανοιχτό χώρο όταν δεν υπάρχει κοντινό εμπόδιο το LiDAR δεν εντοπίζει εμπόδια και επιστρέφει μηδέν. Με αυτή την αντικατάσταση αποτρέπεται το σύστημα από το να ερμηνεύσει λανθασμένα αυτό το 0 ως άμεση σύγκρουση.

4. Οι έγκυρες τιμές αποθηκεύονται στη λίστα ενώ αν δεν συλλεχθεί καμία μέτρηση η συνάρτηση επιστρέφει None.

5. Τέλος, υπολογίζεται και επιστρέφεται ο μέσος όρος των τιμών που αποθηκευτήκαν.

```
def avg_distance(samples=7, max_wait=0.6):
    # Initialize list
    vals = []
    t0 = time.time()
    ser.reset_input_buffer()
    time.sleep(0.01)
    while len(vals) < samples and (time.time() - t0) < max_wait:
        r = read_tfmini() # Read sensor measurment
        if r is None: continue
        d, s = r
        if d == 0.0: d = 12.0 # Reaplace 0 to 12
        vals.append(d) # save valid value
    if not vals: return None # Return None if empty
    return sum(vals) / len(vals) # Calculate and return average
```

Σχήμα 5.7: Συνάρτηση ανάγνωσης και αποκωδικοποίηση δεδομένων του αισθητήρα

5.3.2 Ενεργοποίηση Αποφυγής

Η υλοποίηση της αποφυγής εμποδίων βασίζεται σε ένα σύνολο συναρτήσεων, οι οποίες ενσωματώθηκαν στο script για να ρυθμίζουν τη συμπεριφορά του συστήματος και να ελέγχουν τη μετάβαση από την κανονική πτήση σε κατάσταση αποφυγής.

Αρχικά ξεκινάμε με τη συνάρτηση init_param() οπού όπως φαίνεται στο σχήμα 5.8 χρησιμοποιείται για τον ορισμό των βασικών παραμέτρων που απαιτούνται για τη λειτουργία της αποφυγής εμποδίου. Συγκεκριμένα αρχικοποιούνται οι μεταβλητές που σχετίζονται με αποστάσεις, ταχύτητες και χρονικές διάρκειες οι οποίες χρησιμοποιούνται κατά τη διάρκεια εκτέλεσης του προγράμματος.

Συγκεκριμένα, η μεταβλητή TRIGGER_DIST καθορίζει την απόσταση ενεργοποίησης της αποφυγής εμποδίων. Στη συνέχεια, η μεταβλητή YAW_SCAN_ANGLE καθορίζει σε μοίρες το άνοιγμα σάρωσης δεξιά και αριστερά εξασφαλίζοντας στον αισθητήρα LiDAR ένα ικανοποιητικό εύρος σάρωσης για να εντοπίσει κενό χώρο. Έπειτα, η παράμετρος CLEAR_DIST ορίζει το κενό χώρο που θα πρέπει να έχει η πλευρά ώστε να κατευθυνθεί προς τα εκεί. Παράλληλα ορίζεται η μεταβλητή FWD_VEL όπου ορίζει την ταχύτητα του drone όταν ενεργοποιείται η αποφυγή. Επίσης ορίστηκε μεταβλητή LATERAL_DIST η οποία καθορίζει τη απόσταση που θα κινηθεί παράλληλα με το εμπόδιο όταν στρίψει 90 μοίρες από την πλευρά με το κενό χώρο. Ακόμα ορίστηκε η μεταβλητή FWD_DIST η καθορίζει την απόσταση που πρέπει να κινηθεί, ώστε το drone να προσπεράσει τελείως το εμπόδιο. Ακόμα τέθηκε μεταβλητή MAX_TRIES όπου ορίζει το μέγιστο όριο προσπαθειών εύρεσης κενού χώρου αλλά και πλάγιων μετακινήσεων. Έπειτα, ορίστηκε η RESET_DIST η οποία επιτρέπει την οπισθοχώρηση του drone διευρύνοντας το οπτικό πεδίο για νέα αναζήτηση, όταν δεν εντοπίζεται κενό στην αρχική προσπάθεια. Η μεταβλητή RETREAT_DIST καθορίζει την απόσταση οπισθοχώρησης σε περίπτωση οριστικής αποτυχίας εύρεσης πορείας, λειτουργώντας ως δικλείδα ασφαλείας που απομακρύνει το drone από το εμπόδιο.

Στην υλοποίηση αυτή, το σύστημα δεν δίνει απευθείας εντολές απόστασης. Αντίθετα, αποστέλλει στον ελεγκτή πτήσης εντολές συγκεκριμένης χρονικής διάρκειας, προκειμένου το drone να καλύψει με ακρίβεια τα μέτρα που του έχουμε ορίσει. Για να εξασφαλιστεί αυτό το αποτέλεσμα, οι χρονικές παράμετροι δεν εισάγονται ως σταθερές τιμές, αλλά υπολογίζονται δυναμικά διαιρώντας την επιθυμητή απόσταση με την ταχύτητα πτήσης. Συγκεκριμένα, ορίστηκαν οι μεταβλητές FWD_TIME για την προσπέραση του εμποδίου, RETREAT_TIME για την οριστική οπισθοχώρηση και RESET_TIME για την διεύρυνση του οπτικού πεδίου.

Τέλος, οριστήκαν και παράμετροι για να μειώνουν την ταχύτητα στο μισό και αυξάνουν τις αποστάσεις, επιτρέποντας πιο προσεκτικούς ελιγμούς. Αξίζει να σημειωθεί ότι το σύνολο των τιμών της συνάρτησης έχουν επιλεγεί εμπειρικά κατόπιν δοκιμών. Το σύστημα ωστόσο παραμένει ευέλικτο, δίνοντας τη δυνατότητα στον χειριστή να τροποποιεί αυτές τις παραμέτρους.

```
def init_params():
    # Setup global flight parameter
    global TRIGGER_DIST, CLEAR_DIST, YAW_SCAN_ANGLE
    global FWD_VEL, FWD_DIST, LATERAL_DIST
    global FWD_TIME, RETREAT_TIME, RESET_TIME, MAX_TRIES
    global RETREAT_DIST, RESET_DIST
    global SLOW_FWD_VEL, SLOW_FWD_DIST, SLOW_LATERAL_DIST

    TRIGGER_DIST = 6.0  # Trigger for obstacle
    CLEAR_DIST   = 9.0  # Clear corridor threshold
    YAW_SCAN_ANGLE = 35 # Scan angle deg

    FWD_VEL      = 1.0 # Velocity
    FWD_DIST     = 5.0 # Avoid distance
    LATERAL_DIST = 3.0 #Paralell move


    FWD_TIME     = FWD_DIST / FWD_VEL # Avoidance flight time
    MAX_TRIES    = 3                  # Max attempts
    RETREAT_DIST = 3.0 # Final back when gave up
    RETREAT_TIME = RETREAT_DIST / FWD_VEL # Reatreat time
    RESET_DIST   = 2.0 # When you try to find a corner again
    RESET_TIME   = RESET_DIST / FWD_VEL # Duration for new distance

    # SLOW MODE
    SLOW_FWD_VEL      = FWD_VEL * 0.5 # Reduced velocity
    SLOW_FWD_DIST     = FWD_DIST * 2.0 # Double avoid distance
    SLOW_LATERAL_DIST = LATERAL_DIST * 2.0 # Double parallel move

init_params()
slow_mode = False
```

Σχήμα 5.8: Συνάρτηση αρχικοποίησης βασικών παραμέτρων

Συνεχίζουμε με τη συνάρτηση set_mode() οπού φτιάχτηκε προκειμένου ο αλγόριθμος αποφυγής να πάρει τον έλεγχο του σκάφους. Συγκεκριμένα η συνάρτηση αυτή είναι υπεύθυνη για την αλλαγή της κατάστασης πτήσης, διασφαλίζοντας την μετάβαση του ελεγκτή στην αυτόνομη λειτουργία όταν απαιτείται.

Στο σχήμα 5.9 παρουσιάζεται η ροή της συνάρτησης, η οποία ακολουθεί τα παρακάτω βήματα:

1. Η συνάρτηση λαμβάνει ως όρισμα το όνομα της επιθυμητής κατάστασης πτήσης από την συνάρτηση αποφυγής εμποδίων. Το αλφαριθμητικό (string) αυτό δεν μπορεί να χρησιμοποιηθεί απευθείας καθώς ο ελεγκτής πτήσης επικοινωνεί με αριθμητικές τιμές. Για αυτό, πριν την αποστολή προς το ελεγκτή πραγματοποιείται αντιστοίχιση του ονόματος στο κατάλληλο ακέραιο αναγνωριστικό κωδικό μέσω της συνάρτησης mode_mapping() της βιβλιοθήκης pymavlink.
2. Έχοντας πάρει το σωστό αναγνωριστικό κωδικό το αποστέλλει στο ελεγκτή πτήσης ,ώστε να γίνει η αλλαγή στο συγκεκριμένο mode μέσω της συνάρτησης set_mode_send.
3. Μετά την αποστολή της εντολής, ενεργοποιείται ένας βρόχος ελέγχου με μέγιστο χρονικό περιθώριο τα 5 δευτερόλεπτα, προκειμένου να βεβαιωθεί η αλλαγή του mode. Κατά τη διάρκεια αυτού του βρόχου το πρόγραμμα φιλτράρει τα εισερχόμενα πακέτα από τον ελεγκτή πτήσης χρησιμοποιώντας τη συνάρτηση recv_match με την παράμετρο type='HEARTBEAT'. Για να επιτευχθεί αυτό, η παράμετρος blocking=True αναλαμβάνει να σταματήσει την εκτέλεση του κώδικα μέχρι να ληφθεί το αναμενόμενο πακέτο, ενώ η παράμετρος timeout=1 θέτει ως μέγιστο χρόνο αναμονής το ένα δευτερόλεπτο. Αν αυτός ο χρόνος παρέλθει χωρίς τη λήψη κάποιου μηνύματος το πρόγραμμα προσπαθεί ξανά, επαναλαμβάνοντας τη διαδικασία μέχρι να ολοκληρωθεί ο συνολικός βρόχος των 5 δευτερολέπτων. Μέσω της διαδικασίας αυτής απομονώνονται διαδοχικά τα πακέτα αυτά, τα οποία περιέχουν βασικές πληροφορίες για την κατάσταση του ελεγκτή πτήσης όπως τον τύπο οχήματος, την κατάσταση όπλισης και το τρέχον mode λειτουργίας. Από το κάθε πακέτο εξάγεται η τρέχουσα κατάσταση μέσω της συνάρτησης mavutil.mode_string_v10() και παράλληλα μετατρέπεται από αριθμητική τιμή σε

αλφαριθμητικό. Η μετατροπή αυτή επιτρέπει τη σύγκριση της τρέχουσας κατάστασης με την επιθυμητή, ώστε να επιβεβαιωθεί η αλλαγή. Αν ο χρόνος εξαντληθεί χωρίς να επιβεβαιωθεί η αλλαγή, η συνάρτηση επιστρέφει False και η διαδικασία επαναλαμβάνεται.

```
def set_mode(mode_name):
    # Convert mode to integer
    mode_id = m.mode_mapping()[mode_name]
    # Send mode change
    m.mav.set_mode_send(
        m.target_system, # Target drone id
        mavutil.mavlink.MAV_MODE_FLAG_CUSTOM_MODE_ENABLED, # Enable custom modes
        mode_id # New flight mode id
    )
    print(f"Change mode to {mode_name}...")
    start = time.time()
    # Loop to confirm change
    while time.time() - start < 5:
    # Wait up to 1 second for a HEARTBEAT message from Pixhawk
        hb = m.recv_match(type='HEARTBEAT', blocking=True, timeout=1)
        if hb:
    # Get HEARTBEAT messages
            current = mavutil.mode_string_v10(hb)
    # Check if mode matches
            if current == mode_name:
                print(f"Mode : {mode_name}")
                return True
    print(f"No change confirmed in {mode_name}")
    return False
```

Σχήμα 5.9: Συνάρτηση αλλαγή κατάστασης πτήσης

Έπειτα, έχουμε τη συνάρτηση μετακίνησης send_body_velocity(), η υλοποίηση της οποίας φαίνεται στο σχήμα 5.10 και αναλαμβάνει την κίνηση του αεροσκάφους όταν βρίσκεται στη διαδικασία αποφυγής εμποδίου. Η συνάρτηση αξιοποιεί το τοπικό σύστημα συντεταγμένων του drone επιτρέποντας τη μετακίνηση του με βάση την κατεύθυνση που είναι στραμμένο εκείνη τη στιγμή.

Κατά την κλήση της συνάρτησης εκτελούνται τα εξής βήματα:

1. Αρχικά, η συνάρτηση λαμβάνει τις επιθυμητές ταχύτητες για τους τρεις άξονες κίνησης (vx,vy,vz) και τη χρονική διάρκεια της εντολής ως παραμέτρους. Έπειτα οι τιμές αυτές τυπώνονται στο τερματικό για την ενημέρωση του χρήστη.

2. Στη συνέχεια ξεκινάει ο βρόχος ελέγχου ο οποίος διατηρείται ενεργός για όσο χρονικό διάστημα η διαφορά της τρέχουσας ώρας από την ώρα εκκίνησης παραμένει μικρότερη από την ορισμένη διάρκεια.

3. Εντός του βρόχου, αποστέλλεται το πακέτο κίνησης προς τον ελεγκτή πτήσης μέσω της συνάρτησης set_position_target_local_ned_send(), χρησιμοποιώντας την παράμετρο MAV_FRAME_BODY_NED ώστε η πορεία του drone να υπολογίζεται με βάση τον τρέχοντα προσανατολισμό του και όχι τον γεωγραφικό βορρά. Παράλληλα, εφαρμόζεται η δυαδική μάσκα η οποία αποτελείται από δεκαέξι bits, όπου τα πρώτα τέσσερα δεν χρησιμοποιούνται, ενώ τα υπόλοιπα καθορίζουν ποια πεδία είναι ενεργά. Συγκεκριμένα, η τιμή 1 απενεργοποιεί τη θέση, την επιτάχυνση και την περιστροφή, ενώ η τιμή 0 ενεργοποιεί τις ταχύτητες. Στη μάσκα αυτή περιλαμβάνεται και το bit του force set, το οποίο ορίζει αν οι τιμές επιτάχυνσης θα θεωρηθούν ως δύναμη ή ως επιτάχυνση, χωρίς όμως να χρησιμοποιείται στην παρούσα υλοποίηση. Στη συνέχεια, ορίζονται οι έντεκα παράμετροι της εντολής, οι οποίες πρέπει να δοθούν σε συγκεκριμένη σειρά. Σημειώνεται ότι το force set δεν ορίζεται ως ξεχωριστή παράμετρος στη συνάρτηση, καθώς η πληροφορία του μεταφέρεται αποκλειστικά μέσω της μάσκας. Τα πεδία που δεν χρησιμοποιούνται συμπληρώνονται με μηδενικά, ενώ οι μεταβλητές vx, vy και vz τοποθετούνται στις αντίστοιχες θέσεις και καθορίζουν την ταχύτητα κίνησης του drone στους άξονες x, y και z.

4. Τέλος, για να αποφευχθεί η συνεχόμενη αποστολή εντολών προς τον ελεγκτή πτήσης, προστίθεται μια καθυστέρηση 10 ms με την εντολή time.sleep(0.01). Με αυτόν τον τρόπο μειώνεται ο φόρτος επικοινωνίας και αποτρέπεται η αποστολή πολλών πακέτων σε πολύ μικρό χρονικό διάστημα, εξασφαλίζοντας ομαλή και σταθερή ροή εντολών προς τον ελεγκτή πτήσης.

```
def send_body_velocity(vx, vy, vz, duration):
    print(f"Velocity vx={vx:.2f} vy={vy:.2f} vz={vz:.2f} for {duration:.2f}s")
    # Record start time
    start = time.time()
    # Loop until the specifies duration is reached
    while time.time() - start < duration:
        m.mav.set_position_target_local_ned_send(
            0, # time_boot_ms: 0 for instant execution with no delay
            m.target_system, # Target drone id
            m.target_component, # Target componet id
            mavutil.mavlink.MAV_FRAME_BODY_NED, # Movement based on drone nose not compass
            0b0000111111000111, # Bitmask: enable only velocity
            0, 0, 0, # Ignored position values
            vx, vy, vz, # Target velocity
            0, 0, 0, # Ignored acceleration
            0, 0 # Ignored Yaw
        )
        time.sleep(0.1)
```

Σχήμα 5.10: Συνάρτηση αποστολής εντολών κίνησης

Έπειτα συνεχίζουμε με την ανάλυση της κεντρικής συνάρτησης run_flight() η οποία ελέγχει τη βασική λειτουργία του συστήματος και αποφασίζει πότε πρέπει να ξεκινήσει η αποφυγή εμποδίων. Όπως διακρίνεται και στο σχήμα 5.11, η αρχιτεκτονική της αποτελείται από δύο βρόχους ελέγχου, διασφαλίζοντας την ομαλή μετάβαση του συστήματος από την κατάσταση αναμονής στη διαδικασία ανίχνευσης και αποφυγής.

Πιο συγκεκριμένα, ο πρώτος βρόχος λειτουργεί ως αρχικός έλεγχος πριν την ενεργοποίηση του συστήματος αποφυγής. Μέσω της συνάρτησης recv_match() το πρόγραμμα περιμένει να έρθει το αρχικό σήμα επικοινωνίας δηλαδή το πακέτο HEARTBEAT από το ελεγκτή πτήσης. Μέχρι να έρθει το πρώτο πακέτο, το πρόγραμμα τίθεται σε αναμονή, αποτρέποντας έτσι την εκτέλεση άλλων εντολών. Με την άφιξή του ελέγχεται ο οπλισμός των κινητήρων παίρνοντας τη σχετική πληροφορία από το πεδίο base_mode μέσω της σταθερά MAV_MODE_FLAG_SAFETY_ARMED που λειτουργεί ως φίλτρο για την απομόνωση αυτής της πληροφορίας. Όταν βεβαιωθεί ο οπλισμός των κινητήρων εκτελείται η εντολή break. Η εντολή αυτή τερματίζει τον πρώτο βρόχο, επιτρέποντας στο σύστημα να προχώρηση στη φάση επιτήρησης για τυχόν εμπόδια μπροστά.

Στο επόμενο στάδιο εκτελείται ο δεύτερος βρόχος ο οποίος επιβλέπει διαρκώς την κατάσταση του συστήματος περιμένοντας τη στιγμή που θα ενεργοποιήσει τη συνάρτηση αποφυγής. Τα βήματα που ακολουθεί είναι τα εξής:

1. Σε κάθε επανάληψη, το σύστημα αναζητά την επιβεβαίωση ότι το drone παραμένει οπλισμένο (ARMED). Εάν η κατάσταση αυτή δεν ανιχνευθεί πλέον, το πρόγραμμα αντιλαμβάνεται ότι οι κινητήρες έχουν σταματήσει και τερματίζει τη λειτουργία του μέσω της εντολής os._exit(0). Η ενέργεια αυτή λειτουργεί σαν δικλείδα ασφαλείας για την αποφυγή αποστολής εντολών μετά τον τερματισμό της πτήσης.

2. Στη συνέχεια καλείται η συνάρτηση avg_distance() για να υπολογιστεί η απόσταση από το εμπόδιο. Η τιμή που προκύπτει συγκρίνεται με το όριο ασφαλείας TRIGGER_DIST. Αν η απόσταση είναι μικρότερη ή ίση με το όριο καταγράφεται πιθανή ύπαρξη εμποδίου και αυξάνεται ο μετρητής trigger_count. Ωστόσο, για να επιβεβαιωθεί ότι δεν πρόκειται για τυχαία ένδειξη οι μετρήσεις πρέπει να είναι διαδοχικές. Αν ενδιάμεσα η απόσταση βρεθεί πάνω από

το όριο ασφαλείας ο μετρητής επανέρχεται στο μηδέν ακυρώνοντας τη διαδικασία ώστε να αγνοηθούν στιγμιαία σφάλματα του αισθητήρα.

3. Μόλις η συνθήκη ελέγχου ικανοποιηθεί έχοντας καταγράψει τέσσερις συνεχόμενες μετρήσεις κάτω από το όριο ασφαλείας, το εμπόδιο θεωρείται επιβεβαιωμένο. Στο σημείο αυτό, καλείται η συνάρτηση avoid_obstacle(), η οποία αναλαμβάνει τον πλήρη έλεγχο του σκάφους για την εκτέλεση των ελιγμών αποφυγής.

```python
# Main
def run_flight():

    trigger_count = 0 # Initialize counter
    print("Waiting for motors to be armed...")
    while True:
        # Wait up to 1 second for a HEARTBEAT message from Pixhawk
        hb = m.recv_match(type='HEARTBEAT', blocking=True, timeout=1)
        if not hb: continue # If no message received try again
        # Check the bit to see if the drone is ARMED
        armed = bool(hb.base_mode & mavutil.mavlink.MAV_MODE_FLAG_SAFETY_ARMED)
        if armed:
            print("ARMED motors detected. Starting obstacle avoidance system")
            break
    while True:
        # Wait up to 1 second for a HEARTBEAT message from Pixhawk
        hb = m.recv_match(type='HEARTBEAT', blocking=False)
        if hb:
            # Check the bit to see if the drone is ARMED
            armed = bool(hb.base_mode & mavutil.mavlink.MAV_MODE_FLAG_SAFETY_ARMED)
            if not armed: # If the drone is not ARMED (DISARMED)
                print("DISARMED, stopping.")
                os._exit(0) # stop script

        d = avg_distance() # Get the average distance from the LiDAR sensor
        if d is not None:
            print(f"Forward: {d:.2f} m")
            if d <= TRIGGER_DIST: # If an obstacle is closer than the limit
                trigger_count += 1 # Increment the counter
            else: # If the path is clear
                trigger_count = 0 # Reset the counter to zero

            if trigger_count >= 4: # If the obstacle is confirmed after 4 checks
                print("Obstacle detected, starting avoidance")
                avoid_obstacle() # Execute the avoidance meneuver
                trigger_count = 0 # Reset counter after the maneuver is done
        else:
            print("No LiDAR reading")
        time.sleep(0.2)
```

Σχήμα 5.11: Συνάρτηση για την ενεργοποίηση της αποφυγής εμποδίων.

Μόλις ενεργοποιηθεί η συνάρτηση avoid_obstacle() από την run_flight(), το σύστημα ξεκινά τη διαδικασία παράκαμψης του εμποδίου. Όπως φαίνεται στο σχήμα 5.12, το πρώτο στάδιο της συνάρτησης περιλαμβάνει τα εξής βήματα :

1. Η διαδικασία ξεκινά με το να πραγματοποιείται έλεγχος για τη μετάβαση του ελεγκτή πτήσης σε λειτουργία GUIDED μέσω της κλήσης της συνάρτησης set_mode("GUIDED"). Σε περίπτωση που η μετάβαση αυτή αποτύχει, καλείται η return όπου διακόπτει την εκτέλεση της συνάρτησης, επιστρέφοντας τη ροή του προγράμματος στην run_flight(). Εφόσον η μετάβαση είναι επιτυχής, το σκάφος παραμένει σε λειτουργία GUIDED καθ' όλη τη διάρκεια της διαδικασίας αποφυγής, ώστε να μπορεί να καθοδηγείται από το υπολογιστικό σύστημα για τη λήψη εντολών.

2. Έπειτα, εκτελείται η συνάρτηση send_body_velocity(0, 0, 0, 0.7) η οποία εφαρμόζει μηδενική ταχύτητα και στους τρεις άξονες κίνησης για χρονικό διάστημα 0,7 δευτερολέπτων. Σκοπός της ενέργειας αυτής είναι η ακινητοποίηση του drone, ώστε να εξαλειφθεί η ορμή του πριν την έναρξη του ελιγμού αποφυγής.

3. Έπειτα, καλείται η συνάρτηση check_human(check_time=0.3), η οποία πραγματοποιεί οπτικό έλεγχο διάρκειας 0,3 δευτερολέπτων για την ανίχνευση ανθρώπινης παρουσίας στο περιβάλλον. Η εσωτερική λειτουργία και ο τρόπος με τον οποίο το σύστημα αναγνωρίζει τον άνθρωπο θα παρουσιαστούν διεξοδικά σε επόμενη ενότητα.

```python
def avoid_obstacle():
    # Declare the global variables that will be used
    global slow_mode, MAX_TRIES
    global FWD_VEL, LATERAL_DIST, FWD_DIST, FWD_TIME
    global RESET_TIME, RETREAT_TIME

    # Switch flight mode to GUIDED and check if it succeeded
    if not set_mode("GUIDED"):
        print("Aborting avoidance.")
        return
    time.sleep(0.7)
    # Stop the drone and hover in place
    send_body_velocity(0, 0, 0, 0.7)
    # check for human
    check_human(check_time=0.3)
```

Σχήμα 5.12: Συνάρτηση αποφυγής εμποδίων

5.3.3 Έλεγχος Κατεύθυνσης

Εφόσον το σύστημα μεταβεί επιτυχώς σε κατάσταση GUIDED και ολοκληρωθεί η ακινητοποιήσει του σκάφους και ο οπτικός έλεγχος για ανθρώπινη παρουσία, η συνάρτηση avoid_obstacle() προχωρά στις υπόλοιπες εντολές στο στάδιο της αναζήτησης ασφαλούς διεξόδου από το εμπόδιο.

Αρχικά, ορίζονται ορισμένες βοηθητικές μεταβλητές στο script για την καταμέτρηση των πλάγιων προσπαθειών μετακίνησης και κίνησης προς τα πίσω με το μέγιστο όριο των προσπαθειών αυτών να καθορίζεται από την προκαθορισμένη σταθερά MAX_TRIES. Έπειτα, ορίζεται η λογική μεταβλητή failed, η οποία αρχικοποιείται ως False και λειτουργεί ως σημαία κατάστασης για τη διαχείριση της τελικής αποτυχίας του ελιγμού.

Στη συνέχεια όπως φαίνεται και στο σχήμα 5.13, η διαδικασία διεξόδου ξεκινά και εκτελείται μέσα σε έναν συνεχή βρόχο ελέγχου ο οποίος ακολουθεί τα παρακάτω λογικά βήματα:

1. Για αρχή καλείται η συνάρτηση avg_distance() για να διαβάσει την τρέχουσα απόσταση μπροστά από το αεροσκάφος. Έπειτα, μέσω ελέγχου ελέγχεται άμα υπάρχει μέτρηση από το LIDR άμα δεν υπάρχει ακυρώνεται η αποφυγή. Στη συνέχεια ελέγχεται εάν η τρέχουσα απόσταση είναι ίση ή μεγαλύτερη από το προκαθορισμένο όριο ασφαλείας. Εφόσον ισχύει αυτό, η διαδρομή θεωρείται ελεύθερη και ο ελιγμός τερματίζεται. Ακόμα ελέγχεται αν ο αριθμός των πλευρικών προσπαθειών έχει φτάσει το μέγιστο επιτρεπτό όριο. Αν συμβεί αυτό, ο βρόχος τερματίζεται με την εντολή break και η μεταβλητή failed γίνεται True, ώστε να ακολουθήσει η εναλλακτική διαδικασία ασφαλείας.

2. Εφόσον εντοπιστεί εμπόδιο ξεκινά η διαδικασία αναζήτησης ελεύθερου χώρου. Αρχικά υπολογίζεται η γωνία σάρωσης scan_angle η οποία ξεκινά από τις 35 μοίρες της σταθεράς YAW_SCAN_ANGLE. Η τιμή αυτή αυξάνεται κατά 5 μοίρες σε κάθε νέα προσπάθεια οπισθοχώρησης μέσω του τύπου retreat_attempts * 5 μπορώντας να φτάσει μέχρι και τις 45 μοίρες. Με αυτόν τον τρόπο το LiDAR έχει μεγαλύτερο εύρος να ψάξει για κενό χώρο σε κάθε προσπάθεια. Στη συνέχεια καλείται η συνάρτηση scan_sides() η οποία στρίβει το drone δεξιά και αριστερά για να βρεθούν οι αποστάσεις d_right και d_left και να εντοπιστεί η πλευρά με τον μεγαλύτερο διαθέσιμο χώρο.

3. Εάν το drone διαπιστώσει ότι δεν υπάρχει ελεύθερος χώρος ούτε δεξιά ούτε αριστερά ξεκινά η διαδικασία υποχώρησης. Εφόσον δεν έχει ξεπεραστεί το όριο των μέγιστων προσπαθειών, το UAV κινείται προς τα πίσω με αρνητική ταχύτητα για ένα χρονικό διάστημα το οποίο ορίζει και τα μέτρα που θα διανύσει μέσω της εντολής send_body_velocity(-FWD_VEL, 0.0, 0.0, RESET_TIME). Στη συνέχεια χρησιμοποιείται η εντολή send_body_velocity() για 0,8

δευτερόλεπτα ώστε να ακινητοποίηση το σκάφος στην νέα του θέση. Έπειτα αυξάνεται ο μετρητής προσπαθειών και επαναλαμβάνει τη σάρωση ξανά από την νέα θέση και με το νέο εύρος. Αν οι προσπάθειες συμπληρωθούν χωρίς αποτέλεσμα η διαδικασία τερματίζεται και η αποφυγή θεωρείται αποτυχημένη.

```
side_tries = 0
retreat_attempts = 0
failed = False # Flag for retreat

while True:
    # Read distance from LiDAR
    current_dist = avg_distance()
    # Check from sensor readings
    if current_dist is None:
        print("No LiDAR reading aborting avoidance.")
        failed = True
        break

    print(f"Forward distance = {current_dist:.2f} m")
    # If forward distance exceeds the safety threshold the path is clear
    if current_dist >= CLEAR_DIST:
        print("Path already clear.")
        break
    # Check if max side attempts reached
    if side_tries >= MAX_TRIES:
        print("Reached MAX_SIDE_TRIES, giving up.")
        failed = True
        break
    # Calculate and expand scan angle each attempt
    scan_angle = YAW_SCAN_ANGLE + retreat_attempts * 5
    # Execute scan on both sides
    d_right, d_left = scan_sides(scan_angle)
    # Check if no free space is detected on either side
    if (d_right is None or d_right < CLEAR_DIST) and \
       (d_left is None or d_left < CLEAR_DIST):
        print("No free space right/left.")
    # Retreat if attempts remain
        if retreat_attempts < MAX_TRIES:
            print(f"Retreat attempt {retreat_attempts+1}/{MAX_TRIES}")
    # Move back for more space and wire scan range
            send_body_velocity(-FWD_VEL, 0.0, 0.0, RESET_TIME)
            send_body_velocity(0, 0, 0, 0.8)
    # Increment retreat counter and restart the loop for a new scan
            retreat_attempts += 1
            continue
        else:
            failed = True # Mark as failed and move back
            break
```

Σχήμα 5.13: Συνέχεια του κώδικα της συνάρτησης αποφυγής εμποδίων.

Για την υλοποίηση της περιστροφής στο άξονα yaw, είτε δεξιά είτε αριστερά του drone χρησιμοποιείται η συνάρτηση yaw_relative(), η οποία παρουσιάζεται στο σχήμα 5.14. Όταν καλούμε τη συνάρτηση αυτή ορίζουμε τον επιθυμητό αριθμό μοιρών που θέλουμε να διανύσει το drone. Για την εκτέλεση της κίνησης είναι απαραίτητο να οριστεί και η ταχύτητα περιστροφής,επομένως η παράμετρος yaw_rate έχει προκαθοριστεί στις 30 μοίρες ανά δευτερόλεπτο.

Η διαδικασία εκτέλεσης της στροφής ολοκληρώνεται στα εξής τρία στάδια:

1. Αρχικά, πραγματοποιείται έλεγχος στο πρόσημο της επιθυμητής γωνίας (angle_deg) για να προσδιοριστεί αν η κίνηση θα είναι δεξιόστροφη ή αριστερόστροφη. Εάν ο αριθμός είναι θετικός η μεταβλητή direction λαμβάνει την τιμή 1, υποδεικνύοντας ότι το drone πρέπει να στρίψει προς τα δεξιά. Αντίθετα, εάν ο αριθμός είναι αρνητικός, η μεταβλητή λαμβάνει την τιμή -1, ορίζοντας ότι η στροφή θα γίνει προς τα αριστερά. Ο υπολογισμός αυτός είναι απαραίτητος, καθώς η τιμή της μεταβλητής direction θα χρησιμοποιηθεί στη συνέχεια ως υποχρεωτική παράμετρος στην εντολή της MAVLink, ώστε ο ελεγκτής πτήσης να γνωρίζει τη φορά της περιστροφής.

2. Στη συνέχεια, χρησιμοποιείται η εντολή command_long_send του πρωτοκόλλου MAVLink για την αποστολή των δεδομένων. Έπειτα ορίζουμε το drone που θα δεχτεί την εντολή και το συγκεκριμένο εξάρτημά του που θα την εκτελέσει δηλαδή το target_sytem και το target_component. Ακόμα βάζουμε και την παράμετρο MAV_CMD_CONDITION_YAW ότι

θα κινηθεί το UAV άξονα yaw. Στις συνέχεια προστίθεται η παραμέτρους οπού χρησιμοποιείται η συνάρτηση abs(angle_deg), η οποία μετατρέπει τη γωνία σε απόλυτη τιμή ώστε να καθοριστεί το μέγεθος της στροφής. Έπειτα προστίθενται η προκαθορισμένη ταχύτητα και η κατεύθυνση που υπολογίστηκε προηγουμένως. Τέλος τίθεται η παράμετρος 1 η οποία ενημερώνει το drone ότι η στροφή πρέπει να γίνει με βάση την τρέχουσα θέση του και όχι με βάση τον βορρά.

3. Αφού η εντολή σταλεί στον ελεγκτή πτήσης καλείται η εντολή time.sleep. Η συνάρτηση αυτή σταματά προσωρινά την εκτέλεση του κώδικα για ένα συγκεκριμένο χρονικό διάστημα. Η προσωρινή παύση της εκτέλεσης είναι αναγκαία ώστε ο ελεγκτής πτήσης να διαθέτει τον απαιτούμενο χρόνο για την ολοκλήρωση της περιστροφής. Έτσι αποτρέπεται η πρόωρη αποστολή νέων εντολών πριν το UAV ολοκληρώσει τη στροφή.

```
def yaw_relative(angle_deg, yaw_rate=30):
    # If positive number set direction 1 right else negative -1 for left
    direction = 1 if angle_deg >= 0 else -1
    m.mav.command_long_send(
        m.target_system, # Target drone id
        m.target_component, #Target component id
        mavutil.mavlink.MAV_CMD_CONDITION_YAW, # Sent Yaw command
        0, # Command confirmation set to first attempt
        abs(angle_deg), #Absolute value of degrees
        yaw_rate, # Angular speed
        direction,
        1, # Turn relative to current position
        0, 0, 0 # unused
    )
    time.sleep(abs(angle_deg) / yaw_rate + 0.7) # Wait for turn and extra safety time
```

Σχήμα 5.14: Συνάρτηση για περιστροφή στον άξονα yaw.

Για να αξιοποιηθεί πρακτικά η δυνατότητα περιστροφής του UAV κατά την αποφυγή ενός εμποδίου, χρησιμοποιείται η συνάρτηση scan_sides(). Η συνάρτηση αυτή αναλαμβάνει να οργανώσει τη διαδικασία της σάρωσης καλώντας τη συνάρτηση κίνησης yaw_relative που αναλύθηκε προηγουμένως.

Αφού κληθεί από τη συνάρτηση avoid_obstacle(), η οποία έχει ήδη υπολογίσει το απαιτούμενο εύρος σάρωσης, η τιμή αυτή μεταβιβάζεται στη scan_sides(). Η συνάρτηση scan_sides(), η οποία απεικονίζεται στο σχήμα 5.15, ξεκινά με τη σταθεροποίηση του αεροσκάφους στο σημείο όπου βρίσκεται. Αυτό επιτυγχάνεται με τον μηδενισμό των ταχυτήτων του μέσω της εντολής send_body_velocity(0, 0, 0, 0.9) για διάρκεια 0,9 δευτερολέπτων.

Μόλις το UAV σταθεροποιηθεί ακολουθεί ο έλεγχος του διαθέσιμου χώρου ο οποίος πραγματοποιείται μέσα από τρία διαδοχικά στάδια:

1. Για αρχή η συνάρτηση καλεί την yaw_relative δίνοντας της τη θετική τιμή της γωνίας μέσω της scan_angle ώστε το drone να περιστρέφει προς τα δεξιά. Μόλις το όχημα ολοκληρώσει τη στροφή η επόμενη εντολή μετράει τη διαθέσιμη απόσταση στο νέο σημείο μέσω της συνάρτησης avg_distance() και αποθηκεύει το αποτέλεσμα στη μεταβλητή d_right. Στη συνέχεια πραγματοποιείται έλεγχος σχετικά με την ταχύτητα πτήσης. Εάν το σύστημα δεν βρίσκεται ήδη σε κατάσταση αργής λειτουργίας λόγω ανθρώπινης παρουσίας στον χώρο, καλείται η συνάρτηση check_human() για να ελέγξει πιθανή ύπαρξη ανθρώπου. Διαφορετικά, εάν το όχημα έχει ήδη εντοπίσει ανθρώπινη παρουσία και βρίσκεται σε αργή λειτουργία το πρόγραμμα παρακάμπτει τον έλεγχο και τυπώνει ένα μήνυμα. Αμέσως μετά την ολοκλήρωση της δεξιάς σάρωσης χρησιμοποιείται η εντολή send_body_velocity για 0,8 δευτερόλεπτα ώστε να ακινητοποιήσει το drone στο σημείο που βρίσκεται.

2. Στη συνέχεια η συνάρτηση ελέγχει την αριστερή πλευρά. Για να το πετύχει αυτό καλεί ξανά την yaw_relative δίνοντάς της ως παράμετρο την αρνητική τιμή της γωνίας -2 * scan_angle ώστε το drone να επιστραφεί προς τα αριστερά. Ο διπλασιασμός της τιμής αυτής είναι απαραίτητος διότι το UAV βρίσκεται ήδη στραμμένο προς τη δεξιά πλευρά από προηγούμενη εντολή. Για να επιτευχθεί η σωστή περιστροφή η τιμή της γωνίας διπλασιάζεται ώστε το όχημα να καλύψει πρώτα την απόσταση μέχρι την ευθεία και έπειτα να στραφεί προς την αριστερή πλευρά. Μόλις το αεροσκάφος ολοκληρώσει τη στροφή η επόμενη εντολή μετράει τη διαθέσιμη απόσταση στο νέο σημείο μέσω της συνάρτησης avg_distance() και αποθηκεύει το αποτέλεσμα στη μεταβλητή d_left. Όπως και προηγουμένως πραγματοποιείται έλεγχος σχετικά με την ταχύτητα πτήσης. Εάν το σύστημα δεν πετάει ήδη αργά λόγω ανθρώπινης παρουσίας στον χώρο καλείται η συνάρτηση check_human() για να ελέγξει αν υπάρχει άνθρωπος. Διαφορετικά εάν το όχημα έχει ήδη εντοπίσει κάποιον και κινείται ήδη με χαμηλή ταχύτητα το πρόγραμμα παρακάμπτει τον έλεγχο και τυπώνει ένα μήνυμα.

3. Στο τελικό στάδιο το drone επιστρέφει στην αρχική του πορεία. Εκτελείται μία τελευταία κλήση της yaw_relative με θετική γωνία φέρνοντας τη μύτη του σκάφους να κοιτάζει και πάλι στην ευθεία που βρισκόταν πριν ξεκινήσει τη σάρωση. Η συνάρτηση ολοκληρώνει τη λειτουργία της επιστρέφοντας τις δύο αποστάσεις πίσω στον κύριο αλγόριθμο αποφυγής.

```
# SCAN SIDES

def scan_sides(scan_angle):
    print("Starting yaw scan for free space")
    send_body_velocity(0, 0, 0, 0.6)
    # Scan Right
    print(f"SCAN Yaw +{scan_angle}° (right side)...")
    # Turn right for scanning
    yaw_relative(+scan_angle)
    # Measure distance on the right
    d_right = avg_distance()
    print(f"SCAN Right distance {d_right}")
    # Check for humans if not already in slow mode
    if not slow_mode:
        check_human(check_time=0.6)
    else:
        print("Already in Slow Mode (Human seen) ")

    send_body_velocity(0, 0, 0, 0.7)

    # Scan Left
    print(f"SCAN Yaw -{2 * scan_angle}° (left side)...")
    # Turn double the angle to go from right to left
    yaw_relative(-2 * scan_angle)
    d_left = avg_distance()
    print(f"SCAN Left distance {d_left}")
    # Check for humans if not already in slow mode
    if not slow_mode:
        check_human(check_time=0.6)
    else:
        print("Already in Slow Mode (Human seen)")
    # Return to center
    print("SCAN Return to center...")
    yaw_relative(+scan_angle) # Turn back to center
    send_body_velocity(0, 0, 0, 0.7)
    # Return values
    return d_right, d_left
```

Σχήμα 5.15: Συνάρτηση για την εύρεση ελεύθερου χώρου δεξιά και αριστερά.

5.3.4 Εκτέλεση Ελιγμού Αποφυγής

Η διαδικασία συνεχίζεται με τον καθορισμό της κατεύθυνσης αποφυγής. Η κατεύθυνση προς την οποία θα στραφεί το όχημα ορίζεται από την εντολή yaw_angle = +90 if d_right > d_left else -90. Μέσω αυτής της συνθήκης, το drone επιλέγει να στρίψει κατά 90 μοίρες προς την πλευρά με τον περισσότερο διαθέσιμο χώρο ώστε στη συνέχεια να είναι σε θέση να κινηθεί παράλληλα με το εμπόδιο με σκοπό

παράκαμψή του. Συγκεκριμένα, εάν η δεξιά πλευρά διαθέτει μεγαλύτερο ελεύθερο χώρο το σκάφος επιλέγει να κινηθεί δεξιά ενώ σε άλλη περίπτωση επιλέγεται η αριστερή κατεύθυνση.

Έπειτα καλείται η συνάρτηση yaw_relative προκειμένου το drone να περιστραφεί προς την επιλεγμένη πλευρά. Αμέσως μετά εκτελείται η εντολή send_body_velocity με μηδενικές τιμές ταχύτητας με σκοπό την πλήρη ακινητοποίηση και σταθεροποίηση του για 1.4 δευτερόλεπτα. Στη συνέχεια, η συνάρτηση avg_distance καταγράφει τη διαθέσιμη απόσταση προς την κατεύθυνση που μόλις έστριψε το UAV αποθηκεύοντας την τιμή στη μεταβλητή d_side.

Μέσω αυτής της μέτρησης πραγματοποιείται έλεγχος από την νέα του θέση ώστε να βεβαιωθεί ότι δεν υπάρχει κάποιο εμπόδιο μπροστά του προτού ξεκινήσει την παράλληλη κίνηση. Εάν η απόσταση αυτή βρεθεί ανεπαρκής, δηλαδή μικρότερη από την απαιτούμενη τιμή LATERAL_DIST η πλευρική κίνηση ακυρώνεται. Το UAV περιστρέφεται ξανά γυρίζει στην αρχική του θέση και εφόσον δεν έχει υπερβεί το όριο προσπαθειών της μεταβλητής retreat_attempts πηγαίνει προς τα πίσω μέσω της εντολής send_body_velocity(). Η κίνηση αυτή προς τα πίσω επιτυγχάνεται χάρη στο αρνητικό πρόσημο - FWD_VEL. Μόλις ολοκληρωθεί η οπισθοχώρηση, η εντολή continue αναγκάζει το πρόγραμμα να διακόψει την τρέχουσα ροή και να ξεκινήσει από την αρχή τον βρόχο.

Εφόσον ο χώρος επαρκεί, ξεκινά η παράλληλη κίνηση. Ο χρόνος διαδρομής υπολογίζεται στη μεταβλητή lateral_time με βάση την απόσταση και την ταχύτητα, καθορίζονται πόσα μέτρα θα διανύσει το όχημα. Μόλις η πλάγια μετακίνηση ολοκληρωθεί το όχημα σταθεροποιείται στο νέο σημείο εκτελώντας την εντολή send_body_velocity με μηδενικές ταχύτητες για 1,3 δευτερόλεπτα. Έπειτα, επιστρέφει ξανά στην αρχική του κατεύθυνση καλώντας τη συνάρτηση yaw_relative, ώστε να κοιτάζει ξανά μπροστά και ακινητοποιείται ξανά για 1,5 δευτερόλεπτα. Επιπλέον πραγματοποιείται μέτρηση στην ευθεία πορεία μέσω της avg_distance και η τιμή αποθηκεύεται στη μεταβλητή d_forward. Εάν η μπροστινή μέτρηση δείξει απόσταση ίση ή μεγαλύτερη από την CLEAR_DIST επιβεβαιώνεται πως ο χώρος είναι ελεύθερος για ευθεία κίνηση. Σε αυτή την περίπτωση το όχημα μετακινείται προς τα εμπρός μέσω της εντολής send_body_velocity για χρονικό διάστημα ίσο με το FWD_TIME, η οποία ορίζει τα μέτρα της διαδρομής του. Μόλις τελειώσει αυτή η κίνηση ο βρόχος τερματίζεται με την εντολή break και το πρόγραμμα πηγαίνει στο τέλος της συνάρτησης. Διαφορετικά αν το drone δει εμπόδιο μπροστά του ακριβώς πριν ξεκινήσει να πηγαίνει προς τα εμπρός η προσπάθεια καταγράφεται ως αποτυχημένη και αυξάνεται η μεταβλητή side_tries. Στη συνέχεια εκτελείται η εντολή continue η οποία αναγκάζει το σύστημα να ξεκινήσει ολόκληρη τη διαδικασία από την αρχή του βρόχου χρησιμοποιώντας όμως ως αρχή τη νέα θέση στην οποία έχει φτάσει το drone.

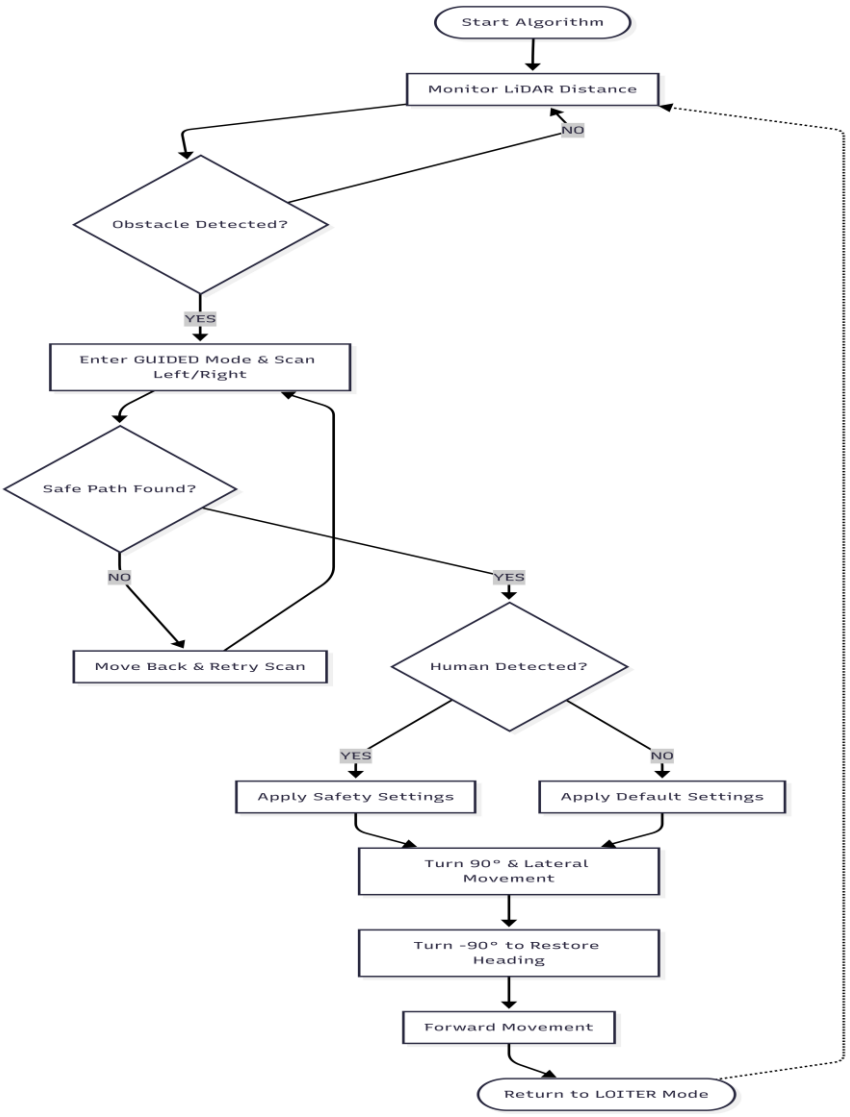
Τέλος, αν η μεταβλητή failed είναι αληθής μετά από πολλαπλές αποτυχημένες προσπάθειες, το drone οπισθοχωρεί για κάποια μέτρα βάσει του RETREAT_TIME καλώντας τη συνάρτηση send_body_velocity. Έπειτα, το όχημα μεταβαίνει σε λειτουργία LOITER ώστε να αναλάβει τον έλεγχο ο χειριστής. Στη συνέχεια εάν η μεταβλητή slow_mode έχει γίνει True ενεργοποιώντας την αργή κατάσταση πτήσης και τις διπλάσιες αποστάσεις το σύστημα επαναφέρεται στις κανονικές του ταχύτητες και αποστάσεις μέσω της εντολής init_params() ώστε το αεροσκάφος να συνεχίσει με τις αρχικές ρυθμίσεις. Ο αλγόριθμος αποφυγής ολοκληρώνεται με μια παύση από την time.sleep ώστε να περιμένει ο κώδικας προτού επιστρέψει στη συνάρτηση run_flight.

Στο σχήμα 5.16 που ακολουθεί παρουσιάζεται συγκεντρωτικά ο αλγόριθμος για τον οποίο έγινε η ανάλυση του παραπάνω κειμένου.

```
        # Choose clearer side for 90 deg turn
        yaw_angle = +90 if (d_right or 0) > (d_left or 0) else -90
        # Display on the terminal the direction and angle it will turn
        turn_side = "Right" if yaw_angle == 90 else "Left"
        print(f"Turning {turn_side} ({yaw_angle}°) for avoidance...")
        # Rotate to chosen direction
        yaw_relative(yaw_angle)
        send_body_velocity(0, 0, 0, 1.4)
        # Measure distance to confirm  clear path
        d_side= avg_distance()
        print(f"Obstacle distance after rotation: {d_side}m")
        if d_side is None or d_side < LATERAL_DIST:
            print(f"Path blocked ({d_side}m) after turn. Aborting.")
        # Revert turn
            yaw_relative(-yaw_angle)
        # Move back if attempts are available
            if retreat_attempts < MAX_TRIES:
                send_body_velocity(-FWD_VEL, 0.0, 0.0, RESET_TIME)
                retreat_attempts += 1
            continue
        # Calculate time needed to cover the lateral distance
        lateral_time = LATERAL_DIST / FWD_VEL
        print(f"Moving Lateral for {LATERAL_DIST}m (Time: {lateral_time:.2f}s)")
        send_body_velocity(FWD_VEL, 0, 0, lateral_time)
        send_body_velocity(0, 0, 0, 1.3)
        # Face forward again
        print("Restoring forward heading...")
        yaw_relative(-yaw_angle)
        send_body_velocity(0, 0, 0, 1.5)
        # Measure forward distance drom new position
        d_forward = avg_distance()
        print(f"Distance ahead: {d_forward}m")
        if d_forward is not None and d_forward >= CLEAR_DIST:
            print(f"Moving Forward for {FWD_DIST}m (Time: {FWD_TIME:.2f}s)")
        # Move forward to pass obstacle
            send_body_velocity(FWD_VEL, 0, 0, FWD_TIME)
            break
        # Obstacle still ahead
        else:
            print("Blocked after lateral move. Trying again.")
            side_tries += 1
            continue
    # Reatreat due failure to avoid obstacle
    if failed:
        print("Could not find safe corridor, retreating.")
        send_body_velocity(-FWD_VEL, 0.0, 0.0, RETREAT_TIME)

    set_mode("LOITER")

    if slow_mode:
        print("Resetting to NORMAL SPEED & DISTANCES.")
        slow_mode = False
        init_params()

    time.sleep(1.2)
```

Σχήμα 5.16: Τελικό στάδιο της διαδικασίας αποφυγής εμποδίου.

Το Σχήμα 5.17 παρουσιάζει συνοπτικά το διάγραμμα ροής του αλγορίθμου αποφυγής εμποδίων. Απεικονίζει τη διαδικασία λήψης αποφάσεων βάσει του αισθητήρα LiDAR για την εκτέλεση του ελιγμού, την επιστροφή σε λειτουργία LOITER, καθώς και τη χρήση της οπτικής αναγνώρισης για την ενεργοποίηση ρυθμίσεων κατάλληλων ρυθμίσεων ασφάλειας σε περίπτωση ανίχνευσης ανθρώπου.



Σχήμα 5.17: Διάγραμμα ροής του αλγορίθμου αποφυγής εμποδίων

5.4 Οπτική Αναγνώριση Μέσω Κάμερας

Η ενσωμάτωση της κάμερας επιτρέπει στο σύστημα να αντιλαμβάνεται τον χώρο στο οποίο κινείται. Μέσω της επεξεργασίας της οπτικής πληροφορίας το UAV αποκτά την ικανότητα να αναγνωρίζει την ανθρώπινη παρουσία προσφέροντας ένα κρίσιμο επίπεδο νοημοσύνης που ενισχύει σημαντικά την ασφάλεια των πτήσεων. Με αυτόν τον τρόπο το αεροσκάφος μπορεί να κατανοεί τι συμβαίνει γύρω του και να αντιδρά κατάλληλα εξασφαλίζοντας μια πιο ελεγχόμενη και ασφαλή πλοήγηση.

Παρακάτω περιγράφεται η προγραμματιστική προσέγγιση αυτής της διαδικασίας. Αρχικά, αναφέρεται ο τρόπος με τον οποίο ενσωματώθηκε το μοντέλο YOLO στο script, δίνοντας στο σύστημα τη δυνατότητα της τεχνητής νοημοσύνης για την αναγνώριση αντικειμένων. Επίσης εξηγείται πώς το drone μεταβαίνει από την απλή εκτέλεση του κώδικα σε μια πιο σύνθετη λειτουργία που λαμβάνει υπόψη την ανθρώπινη παρουσία προσαρμόζοντας τις κίνησής του για μεγαλύτερη ασφάλεια.

5.4.1 Ενσωμάτωση YOLO

Για την προσθήκη της ικανότητας οπτικής αναγνώρισης ανθρώπων μέσω της κάμερας που έχει προστεθεί χρησιμοποιήθηκαν οι βιβλιοθήκες ultralytics και opencv-python. Όπως έγινε αναφορά και

σε προηγούμενο κεφάλαιο η εγκατάστασή τους πραγματοποιήθηκε στο εικονικό περιβάλλον που δημιουργήθηκε μέσω της εντολής:

```
(venv) aitos@raspberrypi:~/drone_ai $ pip install ultralytics opencv-python
```

Μετά την ολοκλήρωση της εγκατάστασης οι βιβλιοθήκες προστέθηκαν στον βασικό κώδικα αναλαμβάνοντας ξεχωριστούς ρόλους. Αρχικά η βιβλιοθήκη cv2 του λογισμικού OpenCV λειτουργεί ως η όραση του συστήματος καθώς παίρνει το βίντεο από την κάμερα και το μετατρέπει σε ψηφιακά δεδομένα. Στη συνέχεια η βιβλιοθήκη YOLO από το πακέτο της ultralytics αναλαμβάνει να παραλάβει τα δεδομένα της εικόνας από το OpenCV και χρησιμοποιεί τεχνητή νοημοσύνη για να αναλύσει τον χώρο και να εντοπίσει τα αντικείμενα. Η εισαγωγή των παραπάνω βιβλιοθηκών μαζί με την ενσωματωμένη βιβλιοθήκη threading για την παράλληλη εκτέλεση δύο διεργασιών πραγματοποιήθηκε στις πρώτες γραμμές του κώδικα όπως αποτυπώνεται στο σχήμα 5.18.

```
  GNU nano 8.4                                          Obstacle.py *
import time
import struct
import serial
from pymavlink import mavutil
import cv2
import threading
from ultralytics import YOLO
import os
```

Σχήμα 5.18: Προσθήκη βιβλιοθηκών στο script

Στη συνέχεια, πραγματοποιείται η ρύθμιση των βασικών παραμέτρων για τη λειτουργία του συστήματος αναγνώρισης, όπως αυτές απεικονίζονται στο σχήμα 5.19. Αρχικά ορίζεται η μεταβλητή STREAM_URL με την τοπική διεύθυνση 127.0.0.1 και θύρα 8888 μέσω της οποίας ο κώδικας λαμβάνει τα δεδομένα της εικόνας από την κάμερα προκειμένου να τα χρησιμοποιήσει για επεξεργασία. Η διαδικασία ρύθμισης της κάμερας ώστε να μεταδίδει το βίντεο στο σύστημα θα παρουσιαστεί αναλυτικά στη συνέχεια. Έπειτα, χρησιμοποιείται η μεταβλητή MODEL_PATH με το αρχείο yolov8n.pt, όπου από το γράμμα n προκύπτει ότι χρησιμοποιείται η έκδοση nano. Η συγκεκριμένη έκδοση επιλέχθηκε διότι αποτελεί ένα ελαφρύ μοντέλο του YOLO καθιστώντας το ιδανικό για τις ανάγκες του Raspberry Pi. Στη συνέχεια, η μεταβλητή YOLO_IMG_SIZE ορίζεται στα 320 pixel. Η παράμετρος αυτή καθορίζει την ανάλυση εισόδου του νευρωνικού δικτύου δηλαδή το μέγεθος της εικόνας που επεξεργάζεται ο αλγόριθμος για να εντοπίσει το αντικείμενο. Η επιλογή της συγκεκριμένης τιμής έγινε με σκοπό να μειωθεί ο όγκος των δεδομένων που πρέπει να επεξεργαστεί σε κάθε καρέ. Με αυτόν τον τρόπο μειώνεται η απαιτούμενη υπολογιστική ισχύς και αυξάνεται η ταχύτητα απόκρισης ώστε η αναγνώριση να γίνεται σε πραγματικό χρόνο χωρίς καθυστερήσεις.

```
# YOLO and CAMERA

STREAM_URL = "tcp://127.0.0.1:8888" # Video stream source address
MODEL_PATH = "yolov8n.pt" # YOLO detection model file
YOLO_IMG_SIZE = 320 # Input image resolution
JPEG_QUALITY = 30
```

Σχήμα 5.19: Ορισμός μεταβλητών

Έπειτα, όπως φαίνεται και από το σχήμα 5.20, χρησιμοποιείται η εντολή YOLO() με όρισμα τη διαδρομή MODEL_PATH, η οποία αναλαμβάνει τη φόρτωση του εκπαιδευμένου αρχείου. Το αποτέλεσμα αυτής της διαδικασίας εκχωρείται στη μεταβλητή model. Αυτό επιτρέπει στο πρόγραμμα να έχει άμεση πρόσβαση στις οδηγίες ανίχνευσης χωρίς να χρειάζεται να διαβάζει το αρχείο κάθε φορά που επεξεργάζεται μια νέα εικόνα. Στη συνέχεια, ορίζονται οι μεταβλητές yolo_lock και frame_lock

μέσω της threading.Lock(). Ο ρόλος τους είναι να προστατεύουν τα δεδομένα αποτρέποντας την ταυτόχρονη εγγραφή νέων καρέ από την κάμερα και την ανάλυση από το YOLO στον ίδιο χώρο μνήμης. Για τη διαχείριση της ροής των δεδομένων, ορίζονται οι μεταβλητές latest_raw_frame και annotated_frame η οποίες αποθηκεύουν την καθαρή εικόνα και το αποτέλεσμα της ανίχνευσης αντίστοιχα. Ορίζονται ως None για να αποφευχθούν σφάλματα πριν την εκκίνηση της κάμερας. Επιπλέον, ορίζεται η μεταβλητή is_human_detected η οποία τίθεται ως False ώστε το πρόγραμμα να ξεκινά θεωρώντας ότι δεν έχει εντοπιστεί ακόμα κανένας άνθρωπος. Επιπλέον, ορίζεται η μεταβλητή slow_mode και τίθεται ως False, ώστε το drone να ξεκινάει με τις κανονικές του ταχύτητες και ρυθμίσεις, μέχρι να εντοπίσει άνθρωπο και να αλλάξει η τιμή της για να τεθεί σε αργή λειτουργία.

```
model = YOLO(MODEL_PATH) # Initialize YOLO model for detection
yolo_lock = threading.Lock() # Lock for thread safe model inference
frame_lock = threading.Lock() # Lock for protecting shared frame buffer
latest_raw_frame = None
annotated_frame = None
is_human_detected = False
slow_mode = False
```

Σχήμα 5.20: Ορισμός μεταβλητών

Αφού οριστούν οι απαραίτητες μεταβλητές και φορτωθεί το μοντέλο YOLO, το πρόγραμμα προχωρά στην υλοποίηση των συναρτήσεων που απαιτούνται για την οπτική αναγνώριση. Προκειμένου το σύστημα να λειτουργεί σε πραγματικό χρόνο χωρίς καθυστερήσεις η λήψη της εικόνας και η ταυτόχρονη ανάλυσή της εκτελούνται παράλληλα. Αυτό επιτυγχάνεται μέσω της χρήσης της βιβλιοθήκης threading. Η παράλληλη εκτέλεση επιτρέπει στην κάμερα να καταγράφει συνεχώς νέα δεδομένα την ίδια ακριβώς στιγμή που ο αλγόριθμος αναλύει μια εικόνα.

Αρχικά, υλοποιείται η συνάρτηση camera_reader η οποία αναλαμβάνει τη συνεχή λήψη και ανανέωση των εικόνων από την κάμερα του drone ώστε το σύστημα να έχει πάντα την πιο πρόσφατη εικόνα. Ο κώδικας της συγκεκριμένης λειτουργίας παρατίθεται στο σχήμα 5.21, ενώ τα στάδια της εκτέλεσης αναλύονται παρακάτω

1. Η διαδικασία ξεκινά με τη δημιουργία της σύνδεσης με την κάμερα μέσω της εντολής VideoCapture της βιβλιοθήκης cv2 της OpenCV. Η εντολή αυτή υλοποιεί τη διεπαφή επικοινωνίας με την πηγή του βίντεο και εξασφαλίζει την πρόσβαση στη ροή δεδομένων. Στην εντολή αυτή ορίζονται η διεύθυνση της ροής δεδομένων το STREAM_URL και η παράμετρος CAP_FFMPEG. Το FFmpeg είναι ένα multimedia framework που αναλαμβάνει την αποκωδικοποίηση του βίντεο. Συγκεκριμένα λαμβάνει τη συμπιεσμένη πληροφορία από τη κάμερα και την αποκωδικοποιεί σε εικόνα. Ο σκοπός του είναι να μετατρέψει τα δεδομένα της κάμερας σε κατάλληλη μορφή ώστε το πρόγραμμα και ο αλγόριθμος να μπορούν να τα διαβάζουν και να τα επεξεργάζονται σωστά. Η σύνδεση αυτή αποθηκεύεται στη μεταβλητή cap από την οποία το πρόγραμμα λαμβάνει κάθε καρέ που έρχεται από την κάμερα. Έπειτα, μέσω της μεταβλητής cap η μέθοδος set ρυθμίζει την ιδιότητα cv2.CAP_PROP_BUFFERSIZE η οποία καθορίζει το μέγιστο αριθμό καρέ που μπορούν να αποθηκευτούν στο buffer. Ορίζοντας τη συγκεκριμένη ιδιότητα στην τιμή 1 εξασφαλίζεται ότι έχει το πιο πρόσφατο καρέ και αποφεύγεται οποιαδήποτε καθυστέρηση στην εικόνα.

2. Ξεκινάει ένας συνεχής βρόχος επαναλήψεων ο οποίος είναι απαραίτητος για τη συνεχή λήψη δεδομένων από την κάμερα. Μέσα σε αυτόν τον βρόχο χρησιμοποιείται η μέθοδος read της μεταβλητής cap. Η συγκεκριμένη λειτουργία επιστρέφει δύο στοιχεία. Το πρώτο είναι μια λογική μεταβλητή κατάστασης η οποία ονομάζεται success και επιβεβαιώνει την επιτυχία της

λήψης. Το δεύτερο στοιχείο είναι ο πίνακας της εικόνας που αποθηκεύεται προσωρινά στη μεταβλητή frame.

3. Ακολουθεί ένας έλεγχος ο οποίος εξετάζει τη μεταβλητή success η οποία λειτουργεί ως ένδειξη για το εάν η λήψη της εικόνας ολοκληρώθηκε με επιτυχία. Εάν η λήψη της εικόνας από την κάμερα αποτύχει η σύνδεση τερματίζεται με την εντολή release() ώστε να κλείσει την αποτυχημένη επικοινωνία. Στη συνέχεια, το σύστημα κάνει μια νέα προσπάθεια σύνδεσης δημιουργώντας από την αρχή το αντικείμενο VideoCapture και εκτελώντας ξανά την set. Συνεχίζοντας η εντολή continue η εκτέλεση επιστρέφει αμέσως στην αρχή του βρόχου ώστε να πραγματοποιηθεί μια νέα προσπάθεια λήψης. Σε περίπτωση που η λήψη της εικόνας ολοκληρωθεί με επιτυχία εκτελείται η εντολή with σε συνδυασμό με τη μεταβλητή frame_lock η οποία έχει οριστεί στην αρχή του προγράμματος ως μηχανισμός κλειδώματος μέσω του threading.Lock. Η εντολή with αναλαμβάνει να κλειδώσει την πρόσβαση στη μνήμη ακριβώς πριν ξεκινήσει η αλλαγή του δείκτη των δεδομένων και να την ανοίξει μόλις αυτή η ενημέρωση ολοκληρωθεί. Μέσα σε αυτό το προστατευμένο περιβάλλον πραγματοποιείται η ανάθεση της εικόνας από την μεταβλητή frame στην μεταβλητή latest_raw_frame.

```
# Camera Reader
def camera_reader():
    global latest_raw_frame
    cap = cv2.VideoCapture(STREAM_URL, cv2.CAP_FFMPEG) # Initialize the incoming video stream
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 1) # Defines the maximum number of frames to be stored in the buffer
    while True:
        success, frame = cap.read() # Capture the next video frame and status
        if not success: # Check if the connection to the camera was lost
            cap.release() # Free the camera resources
            time.sleep(0.5)
            cap = cv2.VideoCapture(STREAM_URL, cv2.CAP_FFMPEG)
            continue
        with frame_lock: # Apply a thread lock to prevent simultaneous read or write access
            latest_raw_frame = frame
```

Σχήμα 5.21: Συνάρτηση για την λήψη εικόνας

Ακολουθεί η συνάρτηση yolo_processor η οποία αναλαμβάνει την ανάλυση των οπτικών δεδομένων. Η συγκεκριμένη συνάρτηση επεξεργάζεται συνεχώς την πιο πρόσφατη εικόνα που λαμβάνει από την κάμερα. Στη συνέχεια εφαρμόζει το μοντέλο YOLO για τον εντοπισμό ανθρώπων. Ο κώδικας της συγκεκριμένης λειτουργίας παρατίθεται στο σχήμα 5.22, ενώ η αναλυτική επεξήγησή του παρουσιάζεται παρακάτω:

1. Η συνάρτηση αρχίζει με έναν βρόχο στον οποίο εκτελούνται επαναλαμβανόμενα όλες οι λειτουργίες της. Έπειτα γίνεται έλεγχος στη μεταβλητή latest_raw_frame για να διαπιστωθεί αν έχει αποθηκευτεί κάποια εικόνα. Αν η μεταβλητή είναι κενή η εκτέλεση καθυστερεί για 0.01 δευτερόλεπτα και ο βρόχος ξεκινάει πάλι από την αρχή. Όταν υπάρχει διαθέσιμη εικόνα εντός του βρόχου εκτελείται η εντολή with σε συνδυασμό με τη μεταβλητή frame_lock η οποία λειτουργεί ως μηχανισμός threading lock. Με αυτόν τον τρόπο κλειδώνεται στιγμιαία η πρόσβαση στη μνήμη και δημιουργείται ένα αντίγραφο της εικόνας εκτελώντας τη μέθοδο latest_raw_frame.copy(), η οποία αποθηκεύει το αντίγραφο στη μεταβλητή frame. Το κλείδωμα της μνήμης χρησιμοποιείται για να προστατεύσει τη διαδικασία της αντιγραφής. Επειδή η κάμερα και η επεξεργασία εικόνας τρέχουν παράλληλα υπάρχει πιθανότητα η κάμερα να γράψει νέα δεδομένα την ίδια στιγμή που η latest_raw_frame προσπαθεί να αντιγράψει τα δεδομένα στη μεταβλητή frame. Με το κλείδωμα αποφεύγεται αυτή η σύγκρουση και η εικόνα αντιγράφεται με απόλυτη ασφάλεια.

2. Αμέσως μετά ξεκινά η ανάλυση της εικόνας εκτελώντας την εντολή model.predict, μέσω της οποίας το προεκπαιδευμένο νευρωνικό δίκτυο αναλαμβάνει να κάνει την αναγνώριση. Η διαδικασία αυτή ρυθμίζεται μέσω συγκεκριμένων παραμέτρων που καθορίζουν τη συμπεριφορά του μοντέλου. Αρχικά, η εικόνα περνάει στον αλγόριθμο μέσω της παραμέτρου source για ανάλυση. Έπειτα ορίζεται η παράμετρος imgsz η οποία καθορίζει σε πόσα pixel θα εκτελεστεί η ανάλυση. Ακολουθεί η παράμετρος conf η οποία λειτουργεί ως ένα όριο αξιοπιστίας. Η συγκεκριμένη παράμετρος ρυθμίζεται στο 0.4 προκειμένου το σύστημα να κρατήσει μόνο τα αποτελέσματα για τα οποία η βεβαιότητα της αναγνώρισης ξεπερνά το 40 τοις εκατό. Συνεχίζουμε με την παράμετρο classes στην οποία ορίζεται η τιμή μηδέν ώστε η αναζήτηση να περιορίζεται αποκλειστικά στον εντοπισμό ανθρώπινης παρουσίας. Τέλος, η παράμετρος verbose ορίζεται ως False για να αποτραπεί η περιττή εκτύπωση μηνυμάτων στο σύστημα. Τα συνολικά δεδομένα της αναγνώρισης αποθηκεύονται στη μεταβλητή results σε μορφή λίστας. Στη συνέχεια, μέσω της results[0] επιλέγεται η τρέχουσα εικόνα και κάνοντας χρήση της ιδιότητας boxes εξάγονται τα ορθογώνια περιγράμματα των στόχων. Έπειτα η συνάρτηση len τα καταμετρά και αν υπερβαίνουν το μηδέν η μεταβλητή found λαμβάνει την τιμή True επιβεβαιώνοντας την παρουσία ανθρώπου αλλιώς παραμένει False. Το αποτέλεσμα αυτό χρησιμοποιείται ώστε το σύστημα όταν εντοπίζει ανθρώπινη παρουσία να μειώνει την ταχύτητά του και να αυξάνει την απόσταση αποφυγής από το αντικείμενο. Από το results[0] λαμβάνουμε την επεξεργασμένη εικόνα του καρέ και με τη συνάρτηση plot() το YOLO σχεδιάζει τα περιγράμματα που αντιστοιχούν στις συντεταγμένες των εντοπισμένων αντικειμένων. Το τελικό οπτικό αποτέλεσμα αποθηκεύεται στη μεταβλητή temp_annotated και στη συνέχεια χρησιμοποιείται για την εμφάνιση της εικόνας στην οθόνη ώστε ο χρήστης να βλέπει τα σημεία όπου εντοπίστηκαν αντικείμενα.

3. Στη συνέχεια του κώδικα χρησιμοποιείται η εντολή with μαζί με τη μεταβλητή frame_lock για να κλειδώσει στιγμιαία η πρόσβαση στη μνήμη. Όσο η μνήμη παραμένει κλειστή η εικόνα από την temp_annotated εκχωρείται στη μεταβλητή annotated_frame και έπειτα η λογική τιμή found η οποία υποδηλώνει τον επιτυχή εντοπισμό περνάει στην is_human_detected. Αυτό το κλείδωμα αποτρέπει τη σύγκρουση δεδομένων μεταξύ των παράλληλων λειτουργιών του συστήματος.

```
# YOLO Processor
def yolo_processor():
    global latest_raw_frame, annotated_frame, is_human_detected
    while True:
        # Check if the camera captured a frame yet
        if latest_raw_frame is None:
            time.sleep(0.01)
            continue
        with frame_lock: # Lock resources to ensure the safe transfer of image data
            frame = latest_raw_frame.copy() # Copy to new variable
        results = model.predict(source=frame, imgsz=YOLO_IMG_SIZE, conf=0.4, classes=[0], verbose=False) # Run YOLO to detect humans
        found = len(results[0].boxes) > 0 # Check if a human is detected in the frame
        temp_annotated = results[0].plot() # Draw the bounding boxes on the image

    with frame_lock:
        annotated_frame = temp_annotated
        is_human_detected = found
    time.sleep(0.01)
```

Σχήμα 5.22: Συνάρτηση για την ανίχνευση ανθρώπινης παρουσίας

Προκειμένου η λειτουργία της κάμερας και η αναγνώριση του αλγορίθμου να εκτελούνται ταυτόχρονα και αδιάκοπα αξιοποιείται η βιβλιοθήκη threading. Πιο συγκεκριμένα, με τη χρήση δύο ξεχωριστών εντολών threading.Thread δημιουργούνται δύο διαφορετικά threads. Η χρήση των threads είναι

απαραίτητη ώστε η κάμερα να τροφοδοτεί συνεχώς το σύστημα με εικόνες και το YOLO να επεξεργάζεται την πιο πρόσφατη από αυτές. Με αυτόν τον τρόπο η λήψη εικόνων και η ανάλυση γίνονται παράλληλα και η μία διεργασία δεν περιμένει την άλλη. Για τη σωστή λειτουργία τους ορίζονται οι κατάλληλες παράμετροι. Αρχικά, η παράμετρος target καθορίζει τη συγκεκριμένη εργασία που αναλαμβάνει κάθε thread διαχωρίζοντας τη λήψη από την ανάλυση. Έπειτα η παράμετρος daemon ορίζεται ως αληθής επιτρέποντας στις διεργασίες να τρέχουν στο παρασκήνιο και να τερματίζονται αυτόματα με το κλείσιμο του κύριου προγράμματος. Τέλος, καλείται η μέθοδος start στο τέλος της κάθε εντολής η οποία ενεργοποιεί το αντίστοιχο thread. Η συνολική υλοποίηση του παραπάνω κώδικα φαίνεται στο σχήμα 5.23.

```
# Satrt the backround threads
threading.Thread(target=camera_reader, daemon=True).start() # Thread 1 for camera
threading.Thread(target=yolo_processor, daemon=True).start() # Thread 2 for YOLO
```

Σχήμα 5.23: Ορισμός των threads για την παράλληλη λήψη και ανάλυση της εικόνας.

Έπειτα, ακολουθεί η συνάρτηση detect_person_once, όπως φαίνεται και από το σχήμα 5.24, η οποία μέσω της μεταβλητής is_human_detected αναλαμβάνει να ενημερώσει το πρόγραμμα για την κατάσταση της ανίχνευσης ανθρώπου επιστρέφοντας την αντίστοιχη τιμή αληθείας. Όταν το αποτέλεσμα επιβεβαιώνεται ως αληθές ενεργοποιούνται αμέσως τα απαραίτητα μέτρα προστασίας και πρόληψης.

```
# Return the current detection state
def detect_person_once():
    return is_human_detected
```

Σχήμα 5.24: Συνάρτηση για την επιστροφή της κατάστασης ανίχνευσης.

5.4.2 Προσαρμογή Συμπεριφοράς Πτήσης

Συνεχίζουμε με τη συνάρτηση check_human της οποίας η κύρια λειτουργία είναι να ελέγχει αν έχει εντοπιστεί άνθρωπος στο οπτικό πεδίο. Εκτελείται τη στιγμή που απαιτείται ο έλεγχος, ώστε να διαπιστωθεί αν το σύστημα αναγνώρισης έχει εντοπίσει ανθρώπινη παρουσία και να ενεργοποιηθεί η ασφαλής και αργή λειτουργία.

Η υλοποίηση της πραγματοποιείται μέσα από την ακόλουθη διαδικασία, όπως φαίνεται και στο σχήμα 5.25:

1. Η συνάρτηση ξεκινά ελέγχοντας τη μεταβλητή slow_mode. Αν η αργή λειτουργία είναι ήδη ενεργή από προηγούμενη εκτέλεση, δηλαδή αν το drone έχει ήδη μειώσει την ταχύτητά του και έχει αυξήσει την απόσταση ασφαλείας από το εμπόδιο, η διαδικασία σταματά αμέσως με την εντολή return και το πρόγραμμα βγαίνει από τη συνάρτηση. Αν δεν έχει ενεργοποιηθεί ακόμη καταγράφεται η τρέχουσα χρονική στιγμή μέσω της time.time() και αποθηκεύεται στη μεταβλητή start, η οποία χρησιμοποιείται παρακάτω στον βρόχο επανάληψης για τον υπολογισμό της διάρκειας του ελέγχου.
2. Στη συνέχεια, ξεκινά ένας βρόχος επανάληψης ο οποίος ελέγχει συνεχώς την πάροδο του χρόνου. Σε κάθε κύκλο αυτής της επανάληψης, το πρόγραμμα λαμβάνει την ακριβή τρέχουσα ώρα μέσω της εντολής time.time() και αφαιρεί την τιμή της μεταβλητής start που αποθηκεύσαμε στο προηγούμενο βήμα. Το αποτέλεσμα αυτής της αφαίρεσης μάς δείχνει με ακρίβεια πόσα δευτερόλεπτα έχουν περάσει από τη στιγμή που ξεκίνησε ο βρόχος επανάληψης.

Όσο αυτός ο χρόνος που έχει περάσει παραμένει μικρότερος από την παράμετρο check_time, η διαδικασία συνεχίζεται κανονικά. Με αυτόν τον τρόπο, δημιουργείται ένα χρονόμετρο που εξασφαλίζει ότι το drone θα ελέγχει αν υπάρχει ανθρώπινη παρουσία μόνο για τον χρόνο που έχουμε ορίσει στην check_time αποτρέποντας το σύστημα από το να εγκλωβιστεί σε μια ατελείωτη αναμονή.

3. Στον βρόχο πραγματοποιείται έλεγχος μέσω της detect_person_once() για το αν υπάρχει άνθρωπος στο περιβάλλον. Εφόσον υπάρχει η μεταβλητή slow_mode τίθεται σε True. Στη συνέχεια, γίνεται αντικατάσταση των βασικών παραμέτρων κίνησης, ώστε το drone να κινηθεί πιο αργά και να διανύσει μεγαλύτερη απόσταση κατά την αποφυγή από το εμπόδιο. Συγκεκριμένα, στην ταχύτητα FWD_VEL εκχωρείται η μειωμένη τιμή της SLOW_FWD_VEL, ενώ οι αποστάσεις FWD_DIST και LATERAL_DIST λαμβάνουν τις αυξημένες τιμές των SLOW_FWD_DIST και SLOW_LATERAL_DIST αντίστοιχα, ώστε να διατηρείται μεγαλύτερη απόσταση από το εμπόδιο κατά την αποφυγή. Έπειτα, ο χρόνος κίνησης FWD_TIME υπολογίζεται ξανά από τη σχέση απόστασης προς ταχύτητα ώστε να καλυφθεί η διπλάσια απόσταση. Αντίστοιχα, οι χρόνοι RETREAT_TIME και RESET_TIME προσαρμόζονται στη μειωμένη ταχύτητα, έτσι ώστε να παραμείνουν σταθερές οι αποστάσεις τους. Τέλος, η εντολή return τερματίζει τη συνάρτηση εφαρμόζοντας τις νέες ρυθμίσεις.

```python
def check_human(check_time):
    # Global parameter
    global slow_mode, FWD_VEL, FWD_DIST, LATERAL_DIST
    global FWD_TIME, RETREAT_TIME, RESET_TIME
    # Check if it is True
    if slow_mode:
        return
    start = time.time() # Record scan start time
    # Loop continuously for the given check_time duration
    while time.time() - start < check_time:
        if detect_person_once():
            print("Human detected Enabling SLOW MODE")
            slow_mode = True
            # Overwrite base variables
            FWD_VEL      = SLOW_FWD_VEL
            FWD_DIST     = SLOW_FWD_DIST
            LATERAL_DIST = SLOW_LATERAL_DIST
            FWD_TIME     = FWD_DIST / FWD_VEL # Recalculate forward time for the newly increased distance
            # Adjust times to keep distances constant
            RETREAT_TIME = RETREAT_DIST / FWD_VEL
            RESET_TIME   = RESET_DIST / FWD_VEL

            print("SLOW MODE ACTIVE:", "FWD_VEL =", FWD_VEL, "FWD_DIST =", FWD_DIST, "LATERAL_DIST =", LATERAL_DIST)
            return # Exit and apply new settings
        time.sleep(0.1)
```

Σχήμα 5.25: Συνάρτηση ενεργοποίησης της ασφαλούς λειτουργίας

5.5 Διαδικασία Εκτέλεσης του Αλγορίθμου

Η διαδικασία εκτέλεσης του συστήματος ελέγχου και αποφυγής εμποδίων αποτελείται από δύο στάδια. Το πρώτο στάδιο αφορά την αποστολή της εικόνας από την κάμερα στο script. Το δεύτερο στάδιο περιλαμβάνει την εκκίνηση του κύριου κώδικα, ο οποίος αναλαμβάνει τη σύνδεση με τον ελεγκτή πτήσης, την ανάλυση των οπτικών δεδομένων μέσω του μοντέλου YOLO, τη διαχείριση του αισθητήρα LiDAR και τη μετάδοση της εικόνας.

Για την υλοποίηση του πρώτου σταδίου ανοίγεται αρχικά ένα τερματικό και εκτελείται η παρακάτω εντολή για τη διαμόρφωση και ενεργοποίηση της κάμερας:

```
aitos@raspberrypi:~ $ rpicam-vid -t 0 --width 800 --height 600 --framerate 10 --inline --listen --intra 15 -o tcp://0.0.0.0:8888 --rotation 180
```

Η εντολή αυτή χρησιμοποιεί όπως έχουμε αναφέρει το rpicam για να ενεργοποιήσει την κάμερα. Μέσω της παραμέτρου -o tcp://0.0.0.0:8888, η ροή εκπέμπεται σε όλες τις διαθέσιμες διευθύνσεις IP μέσω της πόρτας 8888 επιτρέποντας στον κύριο κώδικα να συνδεθεί και να λάβει την εικόνα. Στον πίνακα 5.1 παρατηρείται η σημασία της κάθε παραμέτρου.

Πίνακας 5.1: Παράμετροι εκκίνησης κάμερας

Όνομα παραμέτρου	Τιμή	Σημασία
-t	0	Χρόνος εκτέλεσης. Η τιμή 0 δηλώνει συνεχή λειτουργία
-width	800	Πλάτος βίντεο
-height	600	Ύψος βίντεο
-framerate	10	Ρυθμός καρέ ανά δευτερόλεπτο (FPS)
-inline	-	Ενσωμάτωση κεφαλίδων αποκωδικοποίησης
-listen	-	Αναμονή σύνδεσης για έναρξη της ροής
-intra	15	Συχνότητα πλήρων καρέ
-o	tcp://0.0.0.0:8888	Ορισμός προορισμού της ζωντανής ροής
-rotation	180	Περιστροφή της εικόνας κατά 180 μοίρες

Επίσης να σημειωθεί ότι η παράμετρος --rotation 180 χρησιμοποιείται λόγω της φυσικής τοποθέτησης της κάμερας στο drone. Επειδή η κάμερα έχει τοποθετηθεί ανάποδα, η εντολή αυτή περιστρέφει το βίντεο μέσω λογισμικού ώστε η εικόνα να εμφανίζεται με τον σωστό προσανατολισμό. Με αυτόν τον τρόπο διασφαλίζεται ότι η εικόνα φτάνει στον αλγόριθμο.

Μετά την εκτέλεση της εντολής η κάμερα ενεργοποιείται και αρχίζει να καταγράφει τα πρώτα καρέ μέχρι να γεμίσει την προσωρινή μνήμη, όπως φαίνεται και στο σχήμα 5.26. Σε αυτό το σημείο, η διαδικασία σταματάει προσωρινά. Λόγω της παραμέτρου --listen, η κάμερα δεν στέλνει την εικόνα στο δίκτυο αλλά μπαίνει σε κατάσταση αναμονής. Το σύστημα περιμένει να τρέξει το script με τον αλγόριθμο. Όταν ο κώδικας ανοίξει τη σύνδεση και ζητήσει τη ροή από την κάμερα, τότε η καταγραφή συνεχίζεται.

```
aitos@raspberrypi:~ $ rpicam-vid -t 0 --width 800 --height 600 --framerate 10 --inline --listen --intra 15 -o tcp://0.0.0.0:8888 --rotation 180
[0:02:28.959394616] [1063]  INFO Camera camera_manager.cpp:330 libcamera v0.5.2+99-bfd68f78
[0:02:28.973076485] [1068]  INFO RPI pisp.cpp:720 libpisp version 1.3.0
[0:02:28.991698301] [1068]  INFO IPAProxy ipa_proxy.cpp:180 Using tuning file /usr/share/libcamera/ipa/rpi/pisp/imx219.json
[0:02:29.002706783] [1068]  INFO Camera camera_manager.cpp:220 Adding camera '/base/axi/pcie@1000120000/rp1/i2c@88000/imx219@10' for pipeline handler rpi/pisp
[0:02:29.002749853] [1068]  INFO RPI pisp.cpp:1179 Registered camera /base/axi/pcie@1000120000/rp1/i2c@88000/imx219@10 to CFE device /dev/media0 and ISP device /dev/media2 using PiSP variant
 BCM2712_D0
Made X/EGL preview window
Made DRM preview window
Preview window unavailable
Mode selection for 800:600:12:P(10)
    SRGGB10_CSI2P,640x480/103.327 - Score: 1560
    SRGGB10_CSI2P,1640x1232/41.8515 - Score: 1374.49
    SRGGB10_CSI2P,1920x1080/47.5737 - Score: 1566.67
    SRGGB10_CSI2P,3280x2464/21.1941 - Score: 2092.49
    SRGGB8,640x480/103.327 - Score: 2560
    SRGGB8,1640x1232/41.8515 - Score: 2374.49
    SRGGB8,1920x1080/47.5737 - Score: 2566.67
    SRGGB8,3280x2464/21.1941 - Score: 3092.49
Stream configuration adjusted
[0:02:29.189897166] [1063]  INFO Camera camera.cpp:1215 configuring streams: (0) 800x600-YUV420/SMPTE170M (1) 1640x1232-RGGB_PISP_COMP1/RAW
[0:02:29.189994419] [1068]  INFO RPI pisp.cpp:1483 Sensor: /base/axi/pcie@1000120000/rp1/i2c@88000/imx219@10 - Selected sensor format: 1640x1232-SRGGB10_1X10/RAW - Selected CFE format: 1640x
1232-PC1R/RAW
[libx264 @ 0x55561e3afd20] using cpu capabilities: ARMv8 NEON
[libx264 @ 0x55561e3afd20] profile High, level 3.1, 4:2:0, 8-bit
Output #0, h264, to 'tcp://0.0.0.0:8888?listen=1':
  Stream #0:0: Video: h264, yuv420p(tv, smpte170m/smpte170m/bt709), 800x600, q=2-31, 10 fps, 10 tbr, 1000k tbn
      Side data:
        cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: N/A
#7 (0.00 fps) exp 62608.00 ag 4.00 dg 1.00
#8 (10.02 fps) exp 61738.00 ag 4.00 dg 1.00
#9 (10.02 fps) exp 61625.00 ag 4.00 dg 1.00
#10 (10.02 fps) exp 61474.00 ag 4.00 dg 1.00
#11 (10.02 fps) exp 61341.00 ag 4.00 dg 1.00
#12 (10.02 fps) exp 59621.00 ag 4.00 dg 1.00
#13 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#14 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#15 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#16 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#17 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#18 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#19 (10.02 fps) exp 59508.00 ag 4.00 dg 1.00
#20 (10.02 fps) exp 60113.00 ag 4.00 dg 1.00
#21 (10.02 fps) exp 60113.00 ag 4.00 dg 1.00
```

Σχήμα 5.26: Εκτέλεση εντολής για ενεργοποίηση της κάμερας.

Στη συνέχεια, ανοίγεται ένα δεύτερο τερματικό, πραγματοποιείται μετάβαση στον φάκελο όπου έχει αποθηκευτεί το αρχείο Obstacle.py και εκτελείται με την παρακάτω εντολή:

```
(venv) aitos@raspberrypi:~/drone_ai $ python3 Obstacle.py
```

Με την εκτέλεση του αρχείου Obstacle.py, το script ξεκινά και εμφανίζει στο τερματικό τα μηνύματα που έχουν οριστεί στον κώδικα. Παράλληλα, το πρόγραμμα συνδέεται στη θύρα 8888, τερματίζοντας την κατάσταση αναμονής της κάμερας που είχε οριστεί νωρίτερα, και συνεχίζει την καταγραφή.

Όπως απεικονίζει και το σχήμα 5.27, αρχικά ξεκινά με το μήνυμα Connecting to Pixhawk όπου προσπαθεί να συνδεθεί στον ελεγκτή πτήσης. Όταν η σύνδεση είναι επιτυχής εμφανίζεται το μήνυμα “ from System ID 1 Component ID 0” όπου σημαίνει ότι συνδέθηκε στον εκλεκτή.

Έπειτα, περιμένει να μπουν σε λειτουργία οι κινητήρες για να ξεκινήσει ο αλγόριθμος να λειτουργεί και βγάζει το μήνυμα “Waiting for motors to be armed”. Επιπλέον, εμφανίζει τα μηνύματα για την εκκίνηση του τοπικού διακομιστή Flask προβάλλοντας τις διαθέσιμες διευθύνσεις IP. Ο διακομιστής αυτός χρησιμοποιείται για να βλέπουμε ζωντανά την εικόνα της κάμερας με τα αποτελέσματα της ανάλυσης από το μοντέλο YOLO. Ο τρόπος που λειτουργεί αυτός ο διακομιστής και ο κώδικάς του θα εξηγηθούν αναλυτικά στο επόμενο κεφάλαιο.

```
(venv) aitos@raspberrypi:~/drone_ai $ python3 Obstacle.py
Connecting to Pixhawk...
Heartbeat from System ID 1 Component ID 0
Waiting for motors to be armed...
 * Serving Flask app 'Obstacle'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.79:5000
Press CTRL+C to quit
```

Σχήμα 5.27: Εκκίνηση του αλγορίθμου

Με τον οπλισμό των κινητήρων εμφανίζεται το μήνυμα “ARMED motors detected” και ξεκινά η συνεχής παρακολούθηση της μπροστινής απόστασης μέσω του αισθητήρα LiDAR, με την εμφάνιση των μετρήσεων της μορφής “Forward”.

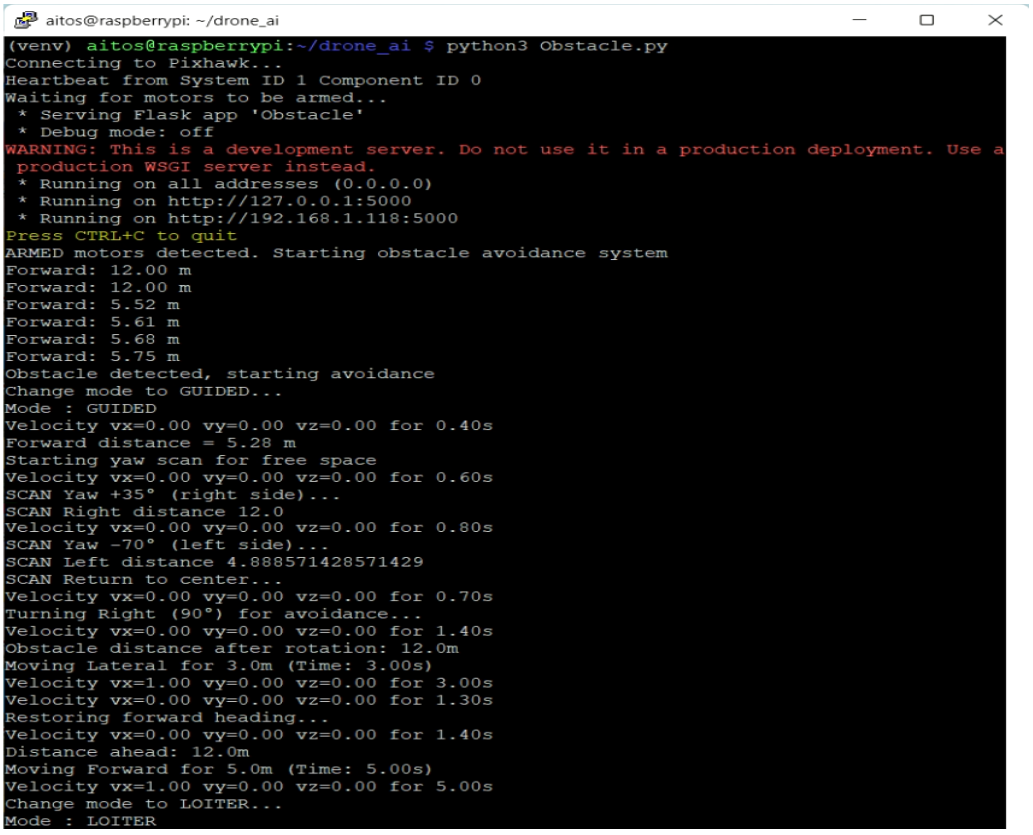
Όταν οι μετρήσεις δείξουν ότι το drone βρίσκεται σε απόσταση μικρότερη από το προκαθορισμένο όριο ασφαλείας από κάποιο εμπόδιο, εμφανίζεται το μήνυμα “Obstacle detected, starting avoidance”, το οποίο ενεργοποιεί τη διαδικασία αυτόνομης αποφυγής.

Αμέσως μετά, το σύστημα μεταβαίνει σε λειτουργία GUIDED, όπως φαίνεται από το μήνυμα “Change mode to GUIDED”, επιτρέποντας στο υπολογιστικό σύστημα να αναλάβει τον έλεγχο του drone.

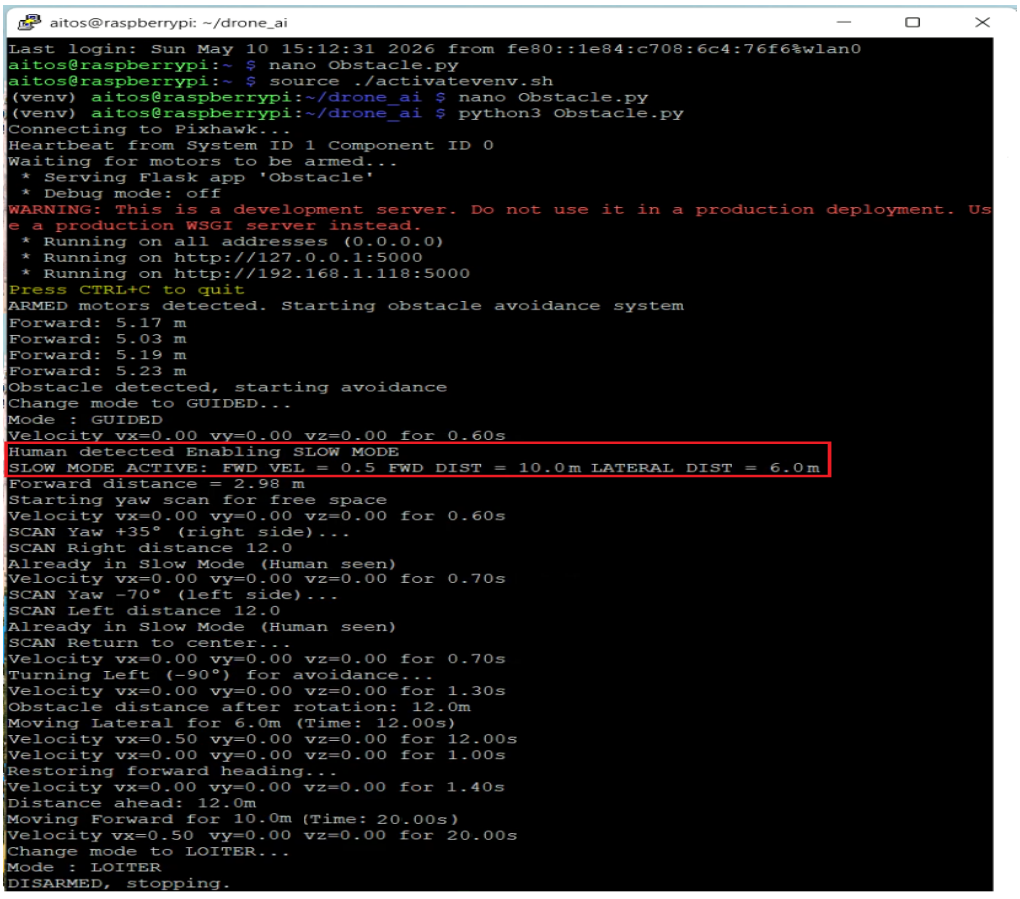
Στη συνέχεια ξεκινά η διαδικασία σάρωσης του περιβάλλοντος χώρου, εμφανίζοντας το μήνυμα “Starting yaw scan for free space”. Το drone περιστρέφεται αρχικά προς τα δεξιά κατά 35 μοίρες και έπειτα προς τα αριστερά κατά 70 μοίρες, καταγράφοντας τις αντίστοιχες αποστάσεις μέσω των μηνυμάτων “SCAN Right distance” και “SCAN Left distance”. Με τις μετρήσεις αυτές, επιλέγεται η κατεύθυνση με τον μεγαλύτερο διαθέσιμο ελεύθερο χώρο. Σε περίπτωση που οι δύο πλευρές έχουν ίση απόσταση από εμπόδιο, επιλέγεται η αριστερή πλευρά. Αν οι αποστάσεις είναι ίσες επιλέγει την αριστερή πλευρά. Στη συνέχεια, το drone στρέφεται κατά 90 μοίρες προς την επιλεγμένη πλευρά, προβάλλοντας το αντίστοιχο μήνυμα στο τερματικό, και εκτελεί την πλευρική μετατόπιση, όπως φαίνεται από το μήνυμα “Moving Lateral”. Αφού ολοκληρωθεί ο ελιγμός, το drone συνεχίζει την εμπρόσθια κίνηση εμφανίζοντας το μήνυμα “Moving Forward” προσπερνώντας το εμπόδιο. Ολόκληρη η παραπάνω διαδικασία απεικονίζεται αναλυτικά στο σχήμα 5.28.

Έπειτα, όταν ολοκληρώσει τη διαδικασία, εμφανίζεται το μήνυμα “Mode : LOITER” για να ξανά λάβει τον έλεγχο ο χειριστής.

Τέλος, σε περίπτωση ανίχνευσης ανθρώπινης παρουσίας κατά τη διάρκεια της σάρωσης, στο τερματικό εμφανίζεται το μήνυμα “Human detected Enabling SLOW MODE”, όπως φαίνεται και στο σχήμα 5.29. Τότε ενεργοποιείται αυτόματα η αργή λειτουργία πτήσης και εμφανίζεται η ένδειξη “SLOW MODE ACTIVE”. Σε αυτή την κατάσταση, η ταχύτητα του drone μειώνεται στο μισό, ενώ οι αποστάσεις αποφυγής γίνονται διπλάσιες. Στη συνέχεια, ο ελιγμός αποφυγής συνεχίζεται με τον ίδιο ακριβώς τρόπο, απλώς με τις νέες παραμέτρους, μέχρι το σύστημα να επιστρέψει με ασφάλεια σε λειτουργία LOITER.



Σχήμα 5.28: Μηνύματα τερματικού κατά την έναρξη της κανονικής αποφυγής εμποδίου



Σχήμα 5.29: Μηνύματα τερματικού κατά την αποφυγή εμποδίου παρουσία ανθρώπου

Κεφάλαιο 6ο: Ζωντανή Μετάδοση Εικόνας και Τηλεμετρίας

6.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο παρουσιάζεται η ανάπτυξη του γραφικού περιβάλλοντος διεπαφής (Dashboard) και του συστήματος ζωντανής προβολής. Η δημιουργία αυτών των εργαλείων επιτρέπουν στο χειριστή να παρακολουθεί σε πραγματικό χρόνο τόσο τα αποτελέσματα της επεξεργασίας της εικόνας από την κάμερα του drone όσο και βασικές πληροφορίες σχετικά με την κατάσταση της πτήσης.

Αρχικά θα εξηγηθεί η λειτουργία της ζωντανής μετάδοσης εικόνας. Θα περιγραφεί ο τρόπος με τον οποίο προβάλλεται η ροή από την κάμερα του drone, επιτρέποντας στον χειριστή να βλέπει το τελικό οπτικό αποτέλεσμα. Έτσι, ο χρήστης μπορεί να παρακολουθεί ζωντανά τι ακριβώς έχει εντοπίσει το σύστημα μετά τη διαδικασία της ανάλυσης και της ανίχνευσης. Μετέπειτα, θα αναλυθεί η δημιουργία του Dashboard. Σκοπός της ενότητας αυτής είναι να παρουσιάσει τον τρόπο με τον οποίο λαμβάνονται και προβάλλονται ζωντανά οι βασικές πληροφορίες όπως η γεωγραφική του θέση, το υψόμετρο, η στάθμη της μπαταρίας και άλλες χρήσιμες ένδειξης πτήσης. Μέσω αυτού του περιβάλλοντος ο χειριστής μπορεί να έχει συνεχή παρακολούθηση της κατάστασης του αεροσκάφους καθ' όλη τη διάρκεια λειτουργίας του.

6.2 Ζωντανή Μετάδοση Εικόνας

Στο προηγούμενο κεφάλαιο αναλύθηκε η λειτουργία του αλγορίθμου αποφυγής και η συνολική εκτέλεση του προγράμματος. Όπως αναφέρθηκε, παράλληλα ενεργοποιείται ένας τοπικός διακομιστής flask για τη ζωντανή μετάδοση της κάμερας. Στην παρούσα ενότητα εξετάζεται ο κώδικας αυτού του διακομιστή και εξηγείται πώς η επεξεργασμένη εικόνα προβάλλεται.

Για την επίτευξη της ζωντανής μετάδοσης, χρησιμοποιήθηκε η βιβλιοθήκη Flask της python. Προκειμένου να χρησιμοποιηθεί η συγκεκριμένη βιβλιοθήκη, απαιτήθηκε αρχικά η εγκατάστασή της στο σύστημα. Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, η εγκατάσταση πραγματοποιήθηκε εντός του εικονικού περιβάλλοντος που έχει δημιουργηθεί, μέσω της εντολής:

```
(venv) aitos@raspberrypi:~/drone_ai $ pip install flask
```

Μετά την ολοκλήρωση της εγκατάστασης, η βιβλιοθήκη Flask ενσωματώθηκε στο script της αποφυγής εμποδίων, λειτουργώντας ως ο διακομιστής (server) του συστήματος για τη μετάδοση της εικόνας. Η εισαγωγή της βιβλιοθήκης όπως φαίνεται και στο σχήμα 6.1, πραγματοποιήθηκε στις πρώτες γραμμές του κώδικα, μαζί με τις υπόλοιπες βιβλιοθήκες.

```
aitos@raspberrypi: ~/drone_ai
  GNU nano 8.4
import time
import struct
import serial
from pymavlink import mavutil
import cv2
import threading
from ultralytics import YOLO
from flask import Flask, Response
import os
```

Σχήμα 6.1: Εισαγωγή βιβλιοθήκης flask

Για την υλοποίηση της ζωντανής μετάδοσης εικόνας δημιουργήθηκε αρχικά η συνάρτηση generate_frames, η οποία αναλαμβάνει τη μετατροπή και προετοιμασία των εικόνων. Η συγκεκριμένη

συνάρτηση επεξεργάζεται συνεχώς την πιο πρόσφατη εικόνα, στην οποία έχουν ήδη σχεδιαστεί τα πλαίσια αναγνώρισης του YOLO και τη διαμορφώνει στην κατάλληλη μορφή ώστε να είναι έτοιμη για τη ζωντανή μετάδοση. Ο κώδικας της συγκεκριμένης συνάρτησης φαίνεται στο σχήμα 6.2, ενώ ο τρόπος με τον οποίο λειτουργεί παρουσιάζεται αναλυτικά παρακάτω:

1. Η συνάρτηση ξεκινά με έναν βρόχο. Εντός του βρόχου, εκτελείται η εντολή with σε συνδυασμό με τη μεταβλητή frame_lock η οποία λειτουργεί ως μηχανισμός κλειδώματος της μνήμης. Όσο η πρόσβαση είναι κλειδωμένη, ελέγχεται η μεταβλητή annotated_frame. Εάν υπάρχουν δεδομένα, δημιουργείται ένα αντίγραφο εκτελώντας τη μέθοδο copy() και αποθηκεύεται στη μεταβλητή frame. Η χρήση του κλειδώματος είναι απαραίτητη για να εξασφαλιστεί ότι η εικόνα αντιγράφεται με ασφάλεια, χωρίς να υπάρξει αλλοίωση σε περίπτωση που η μεταβλητή annotated_frame προσπαθήσει να ανανεώσει τα δεδομένα την ίδια ακριβώς στιγμή.

2. Ακολουθεί ένας έλεγχος συνθήκης για να διαπιστωθεί αν η μεταβλητή frame είναι κενή. Αν η μεταβλητή δεν περιέχει δεδομένα εκτελείται η εντολή continue η οποία αναγκάζει τον βρόχο να παρακάμψει τις επόμενες εντολές και να ξεκινήσει πάλι από την αρχή.

3. Αφού επιβεβαιωθεί ότι η εικόνα είναι έγκυρη, ακολουθεί η διαδικασία συμπίεσης και η μετατροπής. Αρχικά, ορίζονται οι ρυθμίσεις συμπίεσης μέσω της μεταβλητής encode_param. Στη συνέχεια, χρησιμοποιείται η παράμετρος cv2.IMWRITE_JPEG_QUALITY, η οποία καθορίζει την ποιότητα της εικόνας κατά την αποθήκευσή της σε μορφή JPEG. Η τιμή αυτής της παραμέτρου ορίζεται μέσω της μεταβλητής JPEG_QUALITY η οποία στη συγκεκριμένη υλοποίηση έχει οριστεί στο 30. Η επιλογή της συγκεκριμένης τιμής έγινε με σκοπό να μειωθεί το μέγεθος της εικόνας, επιτρέποντας την ταχύτερη μετάδοση της. Έπειτα, εκτελείται η εντολή cv2.imencode για την πραγματοποίηση της μετατροπής σε μορφή JPEG. Η συγκεκριμένη μορφή επιλέχθηκε διότι προσφέρει την απαραίτητη συμπίεση για ομαλή ροή βίντεο. Στη συνέχεια η εντολή αυτή δέχεται ως παραμέτρους την επιθυμητή μορφή εξαγωγής .jpg, τα δεδομένα της εικόνας από τη μεταβλητή frame και τις ρυθμίσεις συμπίεσης από τη encode_param. Τα αποτελέσματα της διαδικασίας επιστρέφονται στη μεταβλητή ret, η οποία δηλώνει την επιτυχία ή την αποτυχία της ενέργειας, καθώς και στη μεταβλητή jpeg η οποία περιλαμβάνει τα δεδομένα της συμπιεσμένης εικόνας. Ελέγχεται η μεταβλητή ret για να διαπιστωθεί αν η συμπίεση ολοκληρώθηκε σωστά. Αν η τιμή της είναι False, εκτελείται η εντολή continue, ώστε να μην σταλεί η τρέχουσα εικόνα και να συνεχίσει με το επόμενο καρέ.

4. Τέλος, η αποστολή της εικόνας πραγματοποιείται βάσει του προτύπου MJPEG (Motion JPEG) μιας τεχνικής όπου η ροή του βίντεο δημιουργείται μεταδίδοντας συνεχόμενα διαδοχικές εικόνες μορφής JPEG. Η διαδικασία αυτή υλοποιείται με τη χρήση της εντολής yield, η οποία κρατά ενεργή τη συνάρτηση και επιτρέπει τη συνεχή αποστολή των δεδομένων χωρίς να σταματά. Το πακέτο που αποστέλλεται ξεκινά με το διαχωριστικό b'--frame\r\n', το οποίο δηλώνει την έναρξη μιας νέας εικόνας. Ακολουθεί η επικεφαλίδα Content-Type: image/jpeg, που ενημερώνει τον παραλήπτη (client) ότι τα δεδομένα που ακολουθούν είναι εικόνα σε μορφή JPEG. Στη συνέχεια προστίθεται η ακολουθία \r\n\r\n, η οποία δημιουργεί αλλαγή γραμμής και διαχωρίζει την επικεφαλίδα από τα δεδομένα της εικόνας. Έπειτα προστίθενται τα δεδομένα της εικόνας μέσω του jpeg.tobytes() όπου η μεταβλητή JPEG περιέχει τη συμπιεσμένη εικόνα και η μέθοδος tobytes() τη μετατρέπει σε δυαδική μορφή κατάλληλη για μετάδοση. Το πακέτο ολοκληρώνεται με το b'\r\n' το οποίο δηλώνει ότι το καρέ έχει τελειώσει.

```python
# Flask Server
app= Flask(__name__) # Initializes the flask
JPEG_QUALITY = 30 # Image quality
def generate_frames():
    global annotated_frame
    while True:
        # Lock the frame momentarily to safely copy it
        with frame_lock:
            frame = None if annotated_frame is None else annotated_frame.copy()
        if frame is None:
            time.sleep(0.01)
            continue
        # Sets the parameters for JPEG compression
        encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), JPEG_QUALITY]
        # Encodes the image into JPEG format
        ret, jpeg = cv2.imencode('.jpg', frame, encode_param)
        if not ret: continue
        # Yields the encoded image in the correct byte format for MJPEG video streaming
        yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n')
```

Σχήμα 6.2: Συνάρτηση για την δημιουργία της ζωντανής ροή βίντεο

Αρχικά, χρησιμοποιείται η μεταβλητή app στην οποία τίθεται η εντολή Flask(__name__). Η ενέργεια αυτή είναι απαραίτητη για να δημιουργηθεί το αντικείμενο app, το οποίο μας δίνει τη δυνατότητα να χρησιμοποιήσουμε όλες τις λειτουργίες της βιβλιοθήκης Flask.

Όπως αποτυπώνεται στο σχήμα 6.3, ορίζεται ο διακοσμητής app.route() με την παράμετρο /video η οποία καθορίζει ως το σημείο πρόσβασης (endpoint) της εφαρμογής. Ο ρόλος του είναι να δέχεται τα αιτήματα στο συγκεκριμένο URL (Uniform Resource Locator) και να ενεργοποιεί τη συνάρτηση video(). Η συνάρτηση αυτή επιστρέφει ένα αντικείμενο Response, το οποίο χρησιμοποιείται για τη συνεχή μετάδοση της εικόνας στον browser. Μέσω της διαδικασίας αυτής δημιουργείται μια συνεχής ροή δεδομένων από τη συνάρτηση generate_frames(), ενώ η παράμετρος mimetype καθορίζει τον τρόπο με τον οποίο αποστέλλονται τα καρέ της εικόνας. Με αυτόν τον τρόπο επιτυγχάνεται η ζωντανή προβολή του βίντεο σε πραγματικό χρόνο.

```python
# Creats the endpoints where the live video can be viewed
@app.route('/video')
def video():
    # Returns the continuous stream of images
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Σχήμα 6.3: Ορισμός endpoint και της συνάρτησης διαχείρισης για τη συνεχή αναμετάδοση της ροής δεδομένων

Στη συνέχεια, ορίζεται η συνάρτηση run_flask(), όπως φαίνεται και στο σχήμα 6.4, η οποία περιλαμβάνει την εντολή app.run(). Μέσα στην εντολή αυτή ορίζεται η παράμετρος host με διεύθυνση 0.0.0.0, η οποία επιτρέπει στον διακομιστή να δέχεται συνδέσεις από όλες τις διαθέσιμες διευθύνσεις IP του συστήματος, καθιστώντας τη ροή προσβάσιμη από οποιαδήποτε άλλη συσκευή βρίσκεται στο ίδιο τοπικό δίκτυο. Έπειτα, ορίζεται η port η οποία με τιμή 5000 καθορίζει τη θύρα επικοινωνίας. Ορίζουμε την επιλογή threaded με τιμή True ώστε να επιτρέπεται την ταυτόχρονη εξυπηρέτηση πολλαπλών αιτημάτων, κάτι που είναι απαραίτητο για τη συνεχή μετάδοση των καρέ χωρίς διακοπές. Παράλληλα προστίθεται και η παράμετρος debug με τιμή False, η οποία απενεργοποιεί τις λειτουργίες ανάπτυξης ώστε να μην εμφανίζονται τεχνικά μηνύματα στον χρήστη.

```python
def run_flask():
    # Starts the local server for video streaming
    app.run(host="0.0.0.0", port=5000, threaded=True, debug=False)
```

Σχήμα 6.4: Εκκίνηση του διακομιστή με τις ρυθμίσεις για τη ροή του βίντεο

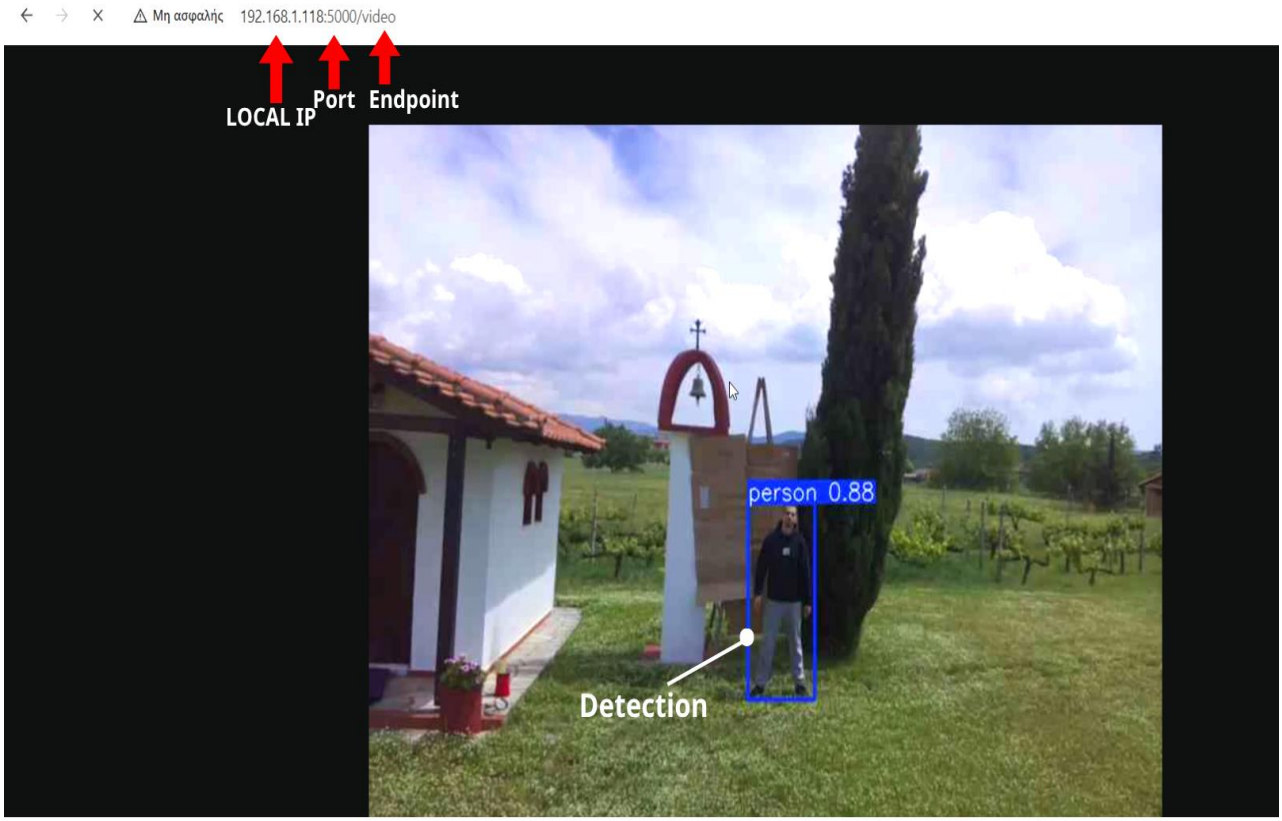
Στη συνέχεια, όπως φαίνεται στο σχήμα 6.5, δημιουργείται η μεταβλητή flask_thread μέσω της βιβλιοθήκης threading για την εκτέλεση ενός νέου νήματος. Στο νήμα αυτό ανατίθεται η εκτέλεση της

συνάρτησης run_flask() ενώ παράλληλα ορίζεται ώστε να τερματίζεται αυτόματα μαζί με το κύριο πρόγραμμα. Έπειτα ενεργοποιείται η εκτέλεση του νήματος μέσω της μεθόδου start(). Με αυτόν τον τρόπο ο Flask server λειτουργεί ταυτόχρονα με την ανίχνευση YOLO, επιτρέποντας τη συνεχή και ομαλή μετάδοση του βίντεο χωρίς διακοπές.

```
# Thread Flask
flask_thread = threading.Thread(target=run_flask, daemon=True)
flask_thread.start()
```

Σχήμα 6.5: Δημιουργία και εκκίνηση νήματος flask

Αφού πραγματοποιηθεί η εκτέλεση του κώδικα, σύμφωνα με τα βήματα που έχουν αναφερθεί σε προηγούμενο κεφάλαιο, ενεργοποιείται ο server και ο χρήστης μπορεί πλέον να παρακολουθήσει τη ζωντανή μετάδοση. Η πρόσβαση στη ροή βίντεο επιτυγχάνεται μέσω ενός περιηγητή ιστού (browser), από οποιαδήποτε συσκευή είναι συνδεδεμένη στο ίδιο τοπικό δίκτυο με το Raspberry Pi. Στη συνέχεια, απαιτείται η πληκτρολόγηση της διεύθυνσης IP του Raspberry Pi στο δίκτυο, ακολουθούμενη από τη θύρα 5000 και την κατάληξη /video. Η πλήρης διεύθυνση σύνδεσης έχει τη μορφή http://IP-Raspberry:5000/video. Μέσω αυτής της διεύθυνσης, το σύστημα προβάλλει σε πραγματικό χρόνο τις επεξεργασμένες εικόνες σε ροή βίντεο, εμφανίζοντας τα πλαίσια αναγνώρισης του αλγορίθμου YOLO γύρω από τα αντικείμενα μαζί με το αντίστοιχο ποσοστό πιθανότητας ότι πρόκειται για άνθρωπο. Η ζωντανή μετάδοση και η προβολή των αποτελεσμάτων ανίχνευσης του μοντέλου YOLO παρουσιάζονται στο σχήμα 6.6.



Σχήμα 6.6: Ζωντανή προβολή αποτελεσμάτων ανίχνευσης ανθρώπου

6.3 Σχεδιασμός Dashboard

6.3.1 Εγκατάσταση Dashboard Node-Red

Το κεντρικό λογισμικό που επιλέχθηκε για την ανάπτυξη του συστήματος τηλεμετρίας είναι το Node-RED μέσω του οποίου υλοποιήθηκε το γραφικό περιβάλλον παρακολούθησης του drone.

Η εγκατάστασή του πραγματοποιήθηκε μέσω του τερματικού στο Raspberry Pi χρησιμοποιώντας την παρακάτω εντολή:

```
aitos@raspberrypi:~ $ sudo npm install -g --unsafe-perm node red
```

Μετά την ολοκλήρωση της εγκατάστασης, το Node-RED κατέστη διαθέσιμο στο σύστημα για τη δημιουργία του γραφικού περιβάλλοντος παρακολούθησης. Στη συνέχεια, πραγματοποιήθηκε η ενεργοποίησή του ως υπηρεσία του συστήματος ώστε να εκκινείται αυτόματα όταν ενεργοποιείται το Raspberry Pi. Η ρύθμιση αυτή πραγματοποιήθηκε μέσω του τερματικού με την εντολή:

```
aitos@raspberrypi:~ $ sudo systemctl enable nodered.service
```

Με αυτόν τον τρόπο, το Node-RED παραμένει συνεχώς ενεργό και μπορεί να χρησιμοποιηθεί για τη λήψη και προβολή των δεδομένων τηλεμετρίας που αποστέλλονται.

Για να καταστεί δυνατή η συνεχής αποστολή των δεδομένων προς το Dashboard απαιτείται η εγκατάσταση κάποιου ενδιάμεσου λογισμικού το οποίο θα αναλάβει τη διαχείριση της επικοινωνίας. Για την κάλυψη αυτής της ανάγκης στο συγκεκριμένο σύστημα επιλέχθηκε ο συνδυασμός του διακομιστή Mosquitto και του πρωτοκόλλου MQTT.

Αρχικά, χρησιμοποιήθηκε ο Mosquitto Broker, ο οποίος λειτουργεί ως ενδιάμεσος διακομιστής επικοινωνίας για τη διαχείριση και προώθηση των δεδομένων. Πιο συγκεκριμένα, αναλαμβάνει να λαμβάνει τα μηνύματα που αποστέλλονται και να τα προωθεί στα αντίστοιχα κανάλια επικοινωνίας (topics), ώστε να μπορούν να διαβαστούν από άλλες εφαρμογές όπως το Node-RED. Με αυτόν τον τρόπο επιτυγχάνεται η συνεχής ανταλλαγή δεδομένων σε πραγματικό χρόνο. Η εγκατάστασή του πραγματοποιήθηκε μέσω του τερματικού με την εντολή:

```
aitos@raspberrypi:~ $ sudo apt install mosquitto mosquitto-clients
```

Στη συνέχεια, χρησιμοποιήθηκε το πρωτόκολλο MQTT (Message Queuing Telemetry Transport), το οποίο αποτελεί ένα ελαφρύ πρωτόκολλο μεταφοράς δεδομένων και χρησιμοποιείται για την αποστολή πληροφοριών μεταξύ διαφορετικών εφαρμογών μέσω δικτύου. Η λειτουργία του βασίζεται στο μοντέλο δημοσίευσης (publish) και εγγραφής (subscribe) όπου τα δεδομένα αποστέλλονται σε συγκεκριμένα κανάλια επικοινωνίας που ονομάζονται topics. Με αυτόν τον τρόπο, κάθε κατηγορία δεδομένων μπορεί να μεταδίδεται ξεχωριστά στο αντίστοιχο topic.

Το MQTT συνεργάζεται με τον Mosquitto Broker, ο οποίος αναλαμβάνει να παραλαμβάνει τα δεδομένα που δημοσιεύονται στα topics και να τα προωθεί στο Node-RED Dashboard για την εμφάνισή τους. Χάρη στην ελαφριά του σχεδίαση, το MQTT χρησιμοποιείται σε εφαρμογές όπου απαιτείται γρήγορη και συνεχής μεταφορά δεδομένων.

Προκειμένου να χρησιμοποιηθεί το πρωτόκολλο MQTT για την αποστολή δεδομένων στο Node-RED εγκαταστάθηκε η βιβλιοθήκη paho-mqtt η οποία παρέχει τις απαραίτητες λειτουργίες σύνδεσης με τον

broker. Η εγκατάσταση πραγματοποιήθηκε στο εικονικό περιβάλλον που έχει δημιουργηθεί για τις ανάγκες της εργασίας:

```
(venv) aitos@raspberrypi:~/drone_ai $  pip install paho-mqtt
```

6.3.2 Ανάπτυξη Κώδικα Αποστολής Δεδομένων

Το αρχείο που αναλαμβάνει τη μεταφορά των δεδομένων από τον ελεγκτή πτήσης προς το Node-RED αναπτύχθηκε σε γλώσσα προγραμματισμού Python. Αυτό το αρχείο δημιουργήθηκε μέσα στον φάκελο που σχεδιάστηκε για την παρούσα εργασία, όπου υπάρχει ήδη το εικονικό περιβάλλον ώστε να αναγνωρίζονται οι βιβλιοθήκες που εγκαταστάθηκαν. Η κατασκευή του αρχείου έγινε μέσω του επεξεργαστή κειμένου nano με τη χρήση της παρακάτω εντολής:

```
(venv) aitos@raspberrypi:~/drone_ai $ nano telemetry.py
```

Αφού ολοκληρώθηκε η δημιουργία του αρχείου προστέθηκαν στην αρχή του κώδικα οι βιβλιοθήκες που θα χρησιμοποιηθούν. Αρχικά προστέθηκε η βιβλιοθήκη time η οποία χρησιμοποιείται για τη διαχείριση του χρόνου. Έπειτα ενσωματώθηκε η βιβλιοθήκη json (JavaScript object notation) η οποία αναλαμβάνει τη μορφοποίηση των συντεταγμένων σε κατάλληλη μορφή ώστε να μεταδίδονται ως ένα ενιαίο πακέτο πληροφορίας. Επιπλέον για την επικοινωνία με τον ελεγκτή πτήσης προστέθηκε η βιβλιοθήκη pymavlink μέσω της οποίας επιτυγχάνεται η σύνδεση και η αποκωδικοποίηση των εισερχόμενων μηνυμάτων. Παράλληλα για την αποστολή αυτών των μετρήσεων στο Dashboard απαιτείται η εισαγωγή της βιβλιοθήκης paho.mqtt η οποία διαχειρίζεται τη σύνδεση με τον ενδιάμεσο διακομιστή και τη δημοσίευση των δεδομένων στα αντίστοιχα κανάλια επικοινωνίας. Η εισαγωγή όλων των παραπάνω βιβλιοθηκών πραγματοποιήθηκε στις πρώτες γραμμές του κώδικα, όπως φαίνεται και στο σχήμα 6.7

```
aitos@raspberrypi: ~/drone_ai
 GNU nano 8.4                                                telemetry.py
import time
import json
from pymavlink import mavutil
import paho.mqtt.client as mqtt
```

Σχήμα 6.7: Εισαγωγή βιβλιοθηκών στο script

Μετά την εισαγωγή των βιβλιοθηκών ακολουθεί η διαδικασία αρχικοποίησης της σύνδεσης με τον ελεγκτή πτήσης. Αρχικά ορίζεται η μεταβλητή connection_string η οποία περιέχει το πρόθεμα udpin ακολουθούμενο από την τοπική διεύθυνση 127.0.0.1 και τη θύρα 14551. Η χρήση του όρου udpin υποδηλώνει ότι το λογισμικό ρυθμίζεται αποκλειστικά για ακρόαση. Το συγκεκριμένο πρόγραμμα λειτουργεί μόνο ως δέκτης περιμένοντας να διαβάσει τα εισερχόμενα μηνύματα του ελεγκτή πτήσης. Η συγκεκριμένη γραμμή κώδικα αποτυπώνεται αναλυτικά στο σχήμα 6.8.

```
# Connect to PIXHAWK
connection_string = "udpin:127.0.0.1:14551"
print(f"Connecting to Pixhawk")
```

Σχήμα 6.8: Μεταβλητή σύνδεσης με τον ελεγκτή πτήσης

Ακολουθεί η χρήση της εντολής try η οποία επιτρέπει την ασφαλή εκτέλεση των βασικών εντολών επικοινωνίας, ώστε να εντοπίζεται άμεσα οποιοδήποτε πρόβλημα κατά τη λειτουργία του script. Μέσα

σε αυτήν ορίζεται η μεταβλητή m, η οποία καλεί τη συνάρτηση mavutil.mavlink_connection με παράμετρο την connection_string προκειμένου να δημιουργηθεί η σύνδεση επικοινωνίας με τον εκλεκτής πτήσης μέσω της διεύθυνσης που έχει οριστεί.

Στη συνέχεια εκτελείται η μεθοδος wait_heartbeat() μέσω της μεταβλητης m η οποία θέτει το πρόγραμμα σε κατάσταση αναμονής μέχρι να ληφθεί το πρώτο μήνυμα heartbeat από τον ελεγκτή πτήσης. Το μήνυμα αυτό λειτουργεί ως επιβεβαίωση ότι ο εκλεκτής είναι ενεργός και ότι η επικοινωνία μεταξύ των συσκευών έχει πραγματοποιηθεί επιτυχώς. Αμέσως εμφανίζεται το μνήμα ότι λήφθηκε το heartbeat επιβεβαιώνοντας τη σύνδεση. Σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα κατά τη σύνδεση, ενεργοποιείται το τμήμα except το οποίο εμφανίζει το κατάλληλο μήνυμα σφάλματος μέσω της εντολής print και στη συνέχεια τερματίζει την εκτέλεση του προγράμματος με την εντολή exit(1). Στο σχήμα 6.9 παρουσιάζεται το τμήμα του κώδικα που υλοποιεί τη διαδικασία σύνδεσης και ελέγχου μέσω του πρωτοκόλλου MAVLink.

```
# Starts error checking
try:
    # Connect to Pixhawk
    m = mavutil.mavlink_connection(connection_string)
    # Wait the error
    m.wait_heartbeat()
    print("Heartbeat OK")
# Handle the error
except Exception as e:
    print(f"Error connecting to Pixhawk: {e}")
# Exit the script
    exit(1)
```

Σχήμα 6.9: Σύνδεση με τον ελεγκτής πτήσης

Αμέσως μετά χρησιμοποιείται η μέθοδος request_data_stream_send(), όπως φαίνεται στο σχήμα 6.10, μέσω της οποίας αποστέλλεται αίτημα για την έναρξη της μετάδοσης των δεδομένων τηλεμετρίας. Η κλήση αυτή αξιοποιεί την ιδιότητα mav της μεταβλητής m, η οποία αναλαμβάνει να δημιουργεί και να μορφοποιεί τα μηνύματα σύμφωνα με το πρωτόκολλο MAVLink. Με αυτόν τον τρόπο το πρόγραμμα ζητά από τον ελεγκτή πτήσης να ξεκινήσει τη συνεχή αποστολή των διαθέσιμων δεδομένων προς το Raspberry Pi.

Μέσα στη μέθοδο ορίζονται οι απαραίτητες παράμετροι για την εκτέλεση της εντολής. Αρχικά μέσω των παραμέτρων m.target_system και m.target_component καθορίζεται το σύστημα και το εξάρτημα προς το οποίο θα αποσταλεί το αίτημα. Οι συγκεκριμένες τιμές έχουν αποθηκευτεί αυτόματα κατά τη διαδικασία σύνδεσης και επιτρέπουν στο πρόγραμμα να επικοινωνεί με τον Pixhawk. Στη συνέχεια χρησιμοποιείται η παράμετρος mavutil.mavlink.MAV_DATA_STREAM_ALL η οποία δηλώνει ότι ζητείται η αποστολή όλων των διαθέσιμων δεδομένων από τους αισθητήρες του συστήματος. Με αυτόν τον τρόπο το πρόγραμμα μπορεί να λαμβάνει πληροφορίες όπως δεδομένα GPS, ταχύτητα, υψόμετρο και κατάσταση της μπαταρίας. Έπειτα ορίζεται η τιμή 4 η οποία καθορίζει τον ρυθμό αποστολής των δεδομένων. Πιο συγκεκριμένα ο ελεγκτής πτήσης στέλνει νέα δεδομένα τέσσερις φορές ανά δευτερόλεπτο ώστε το σύστημα τηλεμετρίας να ενημερώνεται συνεχώς. Τέλος, ορίζεται η τελευταία παράμετρος στην τιμή 1 για να ξεκινήσει η συνεχής αποστολή των δεδομένων προς το πρόγραμμα.

```
# Requests to receive data
m.mav.request_data_stream_send(
    m.target_system, # Target drone id
    m.target_component, # Target componet id
    mavutil.mavlink.MAV_DATA_STREAM_ALL,
    4,   # Rate (Hz)
    1    # Start
)
print("Data Stream Requested (All streams enabled)")
```

Σχήμα 6.10: Αίτημα για συνεχή αποστολή δεδομένων τηλεμετρίας

Έπειτα, ορίζονται οι παράμετροι επικοινωνίας για το πρωτόκολλο MQTT. Συγκεκριμένα η μεταβλητή MQTT_HOST λαμβάνει την τιμή 127.0.0.1 η οποία καθορίζει την τοπική διεύθυνση του συστήματος. Η συγκεκριμένη τιμή διασφαλίζει ότι η επικοινωνία παραμένει αποκλειστικά στο εσωτερικό του Raspberry Pi εξασφαλίζοντας την ταχύτατη και ασφαλή μεταφορά των δεδομένων. Στη συνέχεια ορίζεται η μεταβλητή MQTT_PORT η οποία καθορίζει τη θύρα 1883 επιτρέποντας την απρόσκοπτη σύνδεση με τον διακομιστή MQTT, όπως αποτυπώνεται και στο σχήμα 6.11.

```
# Settings for MQTT
MQTT_HOST = "127.0.0.1" # Defines the local network address of the system
MQTT_PORT = 1883 # Sets the default communication port
```

Σχήμα 6.11: Επικοινωνία με το MQTT

Στη συνέχεια, ορίζονται τα topics του πρωτοκόλλου MQTT, όπως φαίνεται και στο σχήμα 6.12. στα οποία θα αποστέλλονται τα δεδομένα τηλεμετρίας. Οι μεταβλητές αυτές λειτουργούν ως ξεχωριστά κανάλια επικοινωνίας εξασφαλίζοντας τον ορθό διαχωρισμό των πληροφοριών ώστε να αναγνωρίζονται σωστά από το Node-RED.

Αρχικά, η μεταβλητή TOPIC_GPS ορίζει την ετικέτα drone/gps για τη μετάδοση του γεωγραφικού στίγματος. Στη συνέχεια η μεταβλητή TOPIC_SPEED καθορίζει την ετικέτα drone/speed για την αποστολή της ταχύτητας πτήσης. Έπειτα η μεταβλητή TOPIC_BAT ορίζει την ετικέτα drone/battery μεταφέροντας τα δεδομένα για την τάση της μπαταρίας. Παράλληλα η παράμετρος TOPIC_ALT περιέχει την ετικέτα drone/alt για την αποστολή του υψομέτρου. Επίσης η μεταβλητή TOPIC_VERTICAL ορίζει την ετικέτα drone/vertical για τη μετάδοση της κατακόρυφης ταχύτητας του οχήματος. Η χρήση του προθέματος drone σε όλα τα topics πραγματοποιείται ώστε όλα τα δεδομένα που αφορούν το drone να ανήκουν στην ίδια κατηγορία επικοινωνίας.

```
# Definition of all the topics used for telemetry transmission
TOPIC_GPS = "drone/gps"
TOPIC_SPEED = "drone/speed"
TOPIC_BAT = "drone/battery"
TOPIC_ALT = "drone/alt"
TOPIC_VERTICAL = "drone/vertical"
```

Σχήμα 6.12: Ορισμός των topic του MQTT

Στη συνέχεια πραγματοποιείται η διαδικασία σύνδεσης με τον MQTT Broker, όπως φαίνεται και στο σχήμα 6.13. Αρχικά, εμφανίζεται μήνυμα ενημέρωσης ότι ξεκινά η προσπάθεια σύνδεσης με τον broker. Έπειτα η εντολή mqtt.Client αναλαμβάνει την επικοινωνία του κώδικα με το πρωτόκολλο MQTT. Η εντολή αυτή αποθηκεύεται στη μεταβλητή client και χρησιμοποιείται για τις λειτουργίες αποστολής των δεδομένων. Στη συνέχεια χρησιμοποιείται η client.connect() όπου ορίζονται οι βασικές παράμετροι της σύνδεσης, όπως η διεύθυνση του MQTT Broker και η θύρα επικοινωνίας. Επιπλέον, ορίζεται και ο χρόνος αναμονής στα 60 δευτερόλεπτα. Το συγκεκριμένο χρονικό διάστημα καθορίζει το μέγιστο όριο κατά το οποίο η επικοινωνία παραμένει ενεργή πριν το σύστημα θεωρήσει ότι η σύνδεση έχει διακοπεί. Αμέσως μετά ακολουθεί η εντολή client.loop_start() η οποία ενεργοποιεί τον εσωτερικό μηχανισμό λειτουργίας του MQTT στο παρασκήνιο. Με αυτόν τον τρόπο το πρόγραμμα μπορεί να συνεχίσει την εκτέλεση του υπόλοιπου κώδικα ενώ παράλληλα διατηρείται συνεχώς ενεργή η επικοινωνία . Τέλος ως τελική διαδικασία εμφανίζεται μήνυμα επιβεβαίωσης ότι η σύνδεση με τον διακομιστή MQTT ολοκληρώθηκε επιτυχώς.

```
# MQTT
print("Connecting to MQTT Broker...")
client = mqtt.Client() # Creates the mechanism that handles the communication
client.connect(MQTT_HOST, MQTT_PORT, 60) # Connects to the local IP address
client.loop_start() # Starts running the MQTT in the background
print("MQTT Connected!")
```

Σχήμα 6.13: Σύνδεση στο MQTT

Στο σχήμα 6.14 παρουσιάζεται το τμήμα του κώδικα που είναι υπεύθυνο για τη συνεχή λήψη των δεδομένων τηλεμετρίας από τον ελεγκτή πτήσης. Για τον σκοπό αυτό χρησιμοποιείται ένας συνεχόμενος βρόχος while ο οποίος εκτελείται συνεχώς όσο το πρόγραμμα βρίσκεται σε λειτουργία επιτρέποντας τη συνεχή συλλογή νέων δεδομένων από τον Pixhawk.

Στη συνέχεια χρησιμοποιείται η συνάρτηση λήψης recv_match η οποία αναλαμβάνει να συλλέγει και να λαμβάνει τα δεδομένα MAVLink τα οποία αποστέλλονται από τον ελεγκτή πτήσης. Κάθε φορά που εκτελείται η συγκεκριμένη εντολή ανακτά το πρώτο διαθέσιμο πακέτο το οποίο βρίσκεται στη μνήμη εκείνη ακριβώς τη χρονική στιγμή. Έπειτα, ορίζεται και η παράμετρος blocking με την τιμή False ώστε το πρόγραμμα να μην παραμένει σε αναμονή μέχρι να φτάσει κάποιο νέο μήνυμα αλλά να συνεχίζει κανονικά την εκτέλεση του υπόλοιπου κώδικα.

Η επόμενη εντολή πραγματοποιεί έλεγχος για το αν λήφθηκε κάποιο μήνυμα. Στην περίπτωση που η μεταβλητή είναι κενή ενεργοποιείται η εντολή continue η οποία αναγκάζει τον βρόχο να ξεκινήσει από την αρχή νέα αναζήτηση.

Στη συνέχεια χρησιμοποιείται η time.time() μέσω της οποίας καταγράφεται ο ακριβής τρέχων χρόνος και αποθηκεύεται στη μεταβλητή time. Η συγκεκριμένη εντολή δημιουργεί το χρονικό σημείο αναφοράς για τους χρονικούς υπολογισμούς που πραγματοποιούνται κατα την εκτέλεση του βρόχου. Η επόμενη εντολή get_type() εξετάζει το μήνυμα που έχει αποθηκευτεί στη μεταβλητή msg, αναγνωρίζει σε ποια κατηγορία τηλεμετρίας ανήκουν τα δεδομένα που περιέχει και αποθηκεύει αυτή την πληροφορία στη μεταβλητή msg_type.

```
last_global_pos = 0 # Initialize timestamp for the last global posistion
last_bat = 0 # Initialize timestamp for the last battery


print("Starting Main Loop... Waiting for messages.")

while True:
    msg = m.recv_match(blocking=False) # Read incoming data
    if not msg:
        time.sleep(0.01)
        continue # Reset of the loop

    time = time.time()
    msg_type = msg.get_type() # Read the message name
```

Σχήμα 6.14: Έναρξη της συνεχούς λήψης δεδομένων.

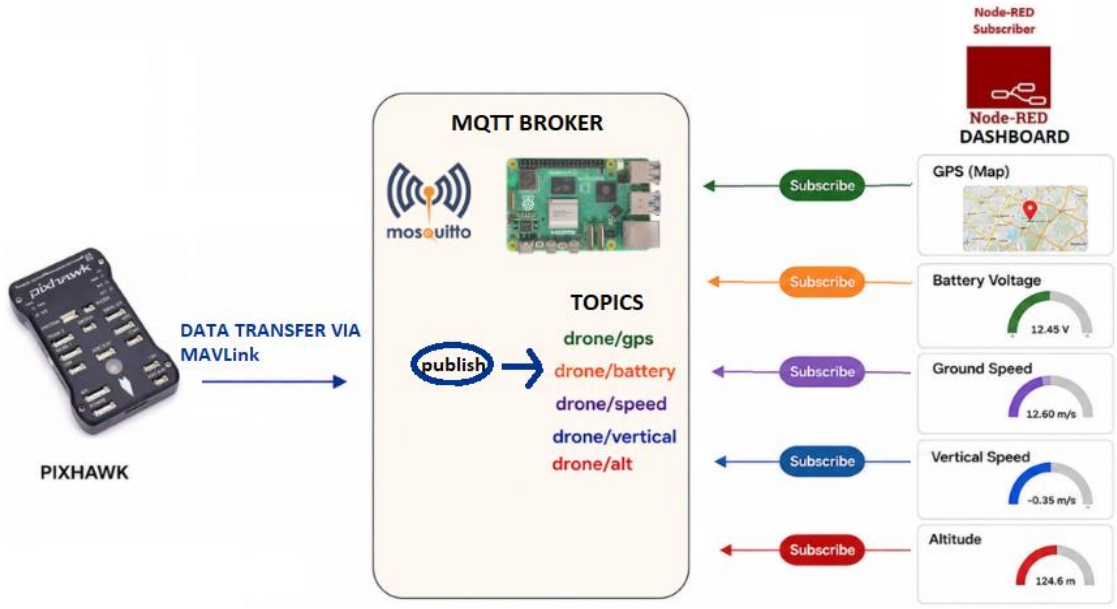
Στο υπόλοιπο κομμάτι του βρόχου, όπως απεικονίζεται και στο σχήμα 6.15, το πρόγραμμα εξετάζει κάθε εισερχόμενο μήνυμα και αναγνωρίζει σε ποια κατηγορία ανήκει. Ανάλογα με το αν αφορά την κατάσταση της μπαταρίας ή τη γεωγραφική θέση ή την ταχύτητα, ενεργοποιείται η αντίστοιχη διαδικασία επεξεργασίας. Ο διαχωρισμός αυτός πραγματοποιείται μέσα από τα παρακάτω βήματα:

1. Διαχείριση δεδομένων μπαταρίας. Αρχικά πραγματοποιείται έλεγχος αν η τιμή της μεταβλητής msg_type είναι ίδια με το τύπο μηνύματος SYS_STATUS η όποια περιέχει την τάση της μπαταρίας. Παράλληλα ελέγχεται αν η διαφορά του τρέχοντος χρόνου time από την προηγούμενη καταγραφή last_bat η οποία έχει αρχικοποιηθεί στην τιμή μηδέν κατά την έναρξη του προγράμματος είναι μεγαλύτερη από ένα δευτερόλεπτο. Η συνθήκη αυτή λειτουργεί ως ένας μηχανισμός καθυστέρησης με σκοπό να αποτρέψει την υπερφόρτωση του δικτύου καθώς η τάση δεν μεταβάλλεται συνέχεια. Εφόσον οι προϋποθέσεις ικανοποιηθούν η τιμή voltage_battery διαιρείται με το χίλια. Η διαίρεση γίνεται με σκοπό η τιμή που είναι σε ακέραιο millivolt να μετατραπεί σε δεκαδικό σε volt. Αμέσως μετά η εντολή print αναλαμβάνει να εμφανίσει την υπολογισμένη τάση. Στη συνέχεια καλείται η μέθοδος publish του αντικειμένου client η οποία υλοποιεί την τελική αποστολή των δεδομένων. Η μέθοδος χρησιμοποιεί την παράμετρο TOPIC_BAT για να ορίσει το κανάλι στο οποίο θα σταλούν τα δεδομένα. Αμέσως μετά η εντολή str μετατρέπει την τιμή της τάσης σε αλφαριθμητική μορφή string ώστε να είναι συμβατή με το πρωτόκολλο επικοινωνίας MQTT. Στο τελευταίο στάδιο η μεταβλητή last_bat ενημερώνεται με τον τρέχοντα χρόνο για τον επόμενο έλεγχο.
2. Διαχείριση δεδομένων θέσης. Στο πρώτο στάδιο πραγματοποιείται έλεγχος αν η τιμή της μεταβλητής msg_type είναι ίδια με τον τύπο μηνύματος GLOBAL_POSITION_INT το οποίο περιέχει τις πληροφορίες θέσης. Ταυτόχρονα γίνεται έλεγχος ώστε η διαφορά ανάμεσα στον τρέχοντα χρόνο και το last_global_pos να υπερβαίνει τα 0.2 δευτερόλεπτα προκειμένου να αποφευχθεί η συνεχής αποστολή δεδομένων. Εφόσον ισχύουν οι προϋποθέσεις οι τιμές lat (γεωγραφικό πλάτος) και lon (γεωγραφικό μήκος) διαιρούνται με το δέκα εκατομμύρια ώστε οι αριθμοί να μετατραπούν σε συντεταγμένες. Αμέσως μετά όλα τα δεδομένα των συντεταγμένων αποθηκεύονται στη μεταβλητή gps_payload. Ακολουθεί, η μέθοδος publish του client αποστέλλει τα δεδομένα στο topic TOPIC_GPS ενώ η εντολή json.dumps τα μετατρέπει ταυτόχρονα σε κείμενο εξασφαλίζοντας την πλήρη συμβατότητα με το πρωτόκολλο MQTT. Στη συνέχεια του κώδικα, μέσα στον ίδιο έλεγχο οι ταχύτητες vx και vy διαιρούνται με το εκατό ώστε να μετατραπούν σε μέτρα ανά δευτερόλεπτο. Έπειτα πραγματοποιείται η κατάλληλη πράξη ώστε οι δύο επιμέρους ταχύτητες να συνδυαστούν και το τελικό αποτέλεσμα της συνολικής ταχύτητας εδάφους του drone να αποθηκευτεί στη μεταβλητή speed. Αμέσως μετά η τιμή αυτή μετατρέπεται σε κείμενο μέσω της εντολής str και δημοσιεύεται στο TOPIC_SPEED. Στην επόμενη γραμμή του κώδικα πραγματοποιείται ο υπολογισμός της κατακόρυφης ταχύτητας του drone μέσω της μεταβλητής vz. Η τιμή διαιρείται με το εκατό ώστε να μετατραπεί από εκατοστά ανά δευτερόλεπτο σε μέτρα ανά δευτερόλεπτο, ενώ χρησιμοποιείται και το σύμβολο μείον ώστε η φορά της κίνησης να εμφανίζεται σωστά. Η τιμή μετατρέπεται σε κείμενο μέσω της str() και αποστέλλεται στο TOPIC_VERTICAL χρησιμοποιώντας την client.publish().Έπειτα πραγματοποιείται η λήψη του σχετικού υψομέτρου του drone μέσω της μεταβλητής relative_alt. Αμέσως μετά η τιμή διαιρείται με το χίλια ώστε να μετατραπεί από χιλιοστά σε μέτρα και αποθηκεύεται στη μεταβλητή alt_m. Η τιμή μετατρέπεται σε κείμενο και αποστέλλεται στο TOPIC_ALT. Στο τελευταίο στάδιο, η μεταβλητή last_global_pos ενημερώνεται με τον τρέχοντα χρόνο για τον επόμενο έλεγχο.

```python
# Battery (Volt)
if msg_type == "SYS_STATUS" and (time - last_bat) > 1.0:
    voltage = msg.voltage_battery / 1000.0 # Convert battery voltage from millivolts to volts
    print(f"Battery Voltage: {voltage:.2f} V")
    client.publish(TOPIC_BAT, str(voltage)) # Send voltage to broker
    last_bat = time # Update the battery timestamp

# GPS & SPEED
if msg_type == "GLOBAL_POSITION_INT" and (time - last_gps) > 0.2:
    lat = msg.lat / 1e7 # Convert latitude to decimal degrees
    lon = msg.lon / 1e7 # Convert longitude to decimal degrees
    alt = msg.relative_alt / 1000.0  # Relative altitude (m)
    print(f"GPS: Lat={lat}, Lon={lon}")
    gps_payload = {"lat": lat, "lon": lon} # Create a JSON with the GPS data
    client.publish(TOPIC_GPS, json.dumps(gps_payload)) # Publish GPS data as JSON

    # Ground speed
    vx = msg.vx / 100.0 # Convert X axis velocity from cm/s to m/s
    vy = msg.vy / 100.0 # Convert Y axis velocity from cm/s to m/s
    speed = (vx*vx + vy*vy) ** 0.5 # Calculate total ground speed
    client.publish(TOPIC_SPEED, str(speed)) # Convert calculated speed to string and publish via MQTT
    # Vertical
    vz= -(msg.vz / 100.0) # Convert Z axis velocity to m/s
    client.publish(TOPIC_VERTICAL, str(vz)) # Convert vertical speed to string and publish via MQTT

    # alt
    alt_m= msg.relative_alt / 1000.0 # Convert relative altitude from millimeters to meters
    client.publish(TOPIC_ALT, str(alt_m)) # Convert altitude to string and publish via MQTT
    last_gps = time # Update the last global timestamp
```

Σχήμα 6.15: Κώδικας επεξεργασίας και μετάδοσης των δεδομένων πτήσης

Στο σχήμα 6.16 αποτυπώνεται η διαδικασία επικοινωνίας μεταξύ του Pixhawk, του MQTT Broker και του Node-RED Dashboard για τη μετάδοση και προβολή των δεδομένων τηλεμετρίας.



Σχήμα 6.16: Διάγραμμα τηλεμετρίας του drone προς το Node-RED

Προκειμένου το script τηλεμετρίας να εκτελείται αυτόματα χωρίς χειροκίνητη εκκίνηση, μετατράπηκε σε service. Με αυτόν τον τρόπο το πρόγραμμα ξεκινά με την ενεργοποίηση του Raspberry Pi και παραμένει ενεργό στο παρασκήνιο αναλαμβάνοντας την αποστολή των δεδομένων προς το Node-RED Dashboard. Η διαδικασία πραγματοποιήθηκε με τη δημιουργία του αρχείου:

```
(venv) aitos@raspberrypi:~/drone_ai $ sudo nano /etc/systemd/system/drone_node.service
```

Εντός του αρχείου, όπως φαίνεται και στο σχήμα 6.17, ορίστηκαν οι βασικές παράμετροι εκτέλεσης του script, όπως ο χρήστης λειτουργίας, ο φάκελος εργασίας καθώς και η εντολή εκκίνησης του Python προγράμματος Επίσης, μέσω της παραμέτρου After ορίστηκε η τιμή mavproxy.service εξασφαλίζοντας ότι το script θα ξεκινά μετά την εκκίνηση του MAVProxy. Η ρύθμιση αυτή είναι απαραίτητη ώστε να έχει προηγουμένως δημιουργηθεί η επικοινωνία με τον Pixhawk πριν ξεκινήσει η λήψη και αποστολή των δεδομένων τηλεμετρίας. Κλείνοντας, η παράμετρος Restart ρυθμίστηκε στην τιμή always ώστε να εξασφαλίζεται η αυτόματη επανεκκίνηση του script σε περίπτωση σφάλματος.

```
  GNU nano 8.4                                    /etc/systemd/system/drone_node.service *
[Unit]
Description=Drone Telemetry
After=mavproxy.service

[Service]
Type=simple
User=aitos
WorkingDirectory=/home/aitos/drone_ai


ExecStart=/home/aitos/drone_ai/venv/bin/python3 /home/aitos/drone_ai/telemetry.py


Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Σχήμα 6.17: Αρχείο ρυθμίσεων της υπηρεσίας του συστήματος

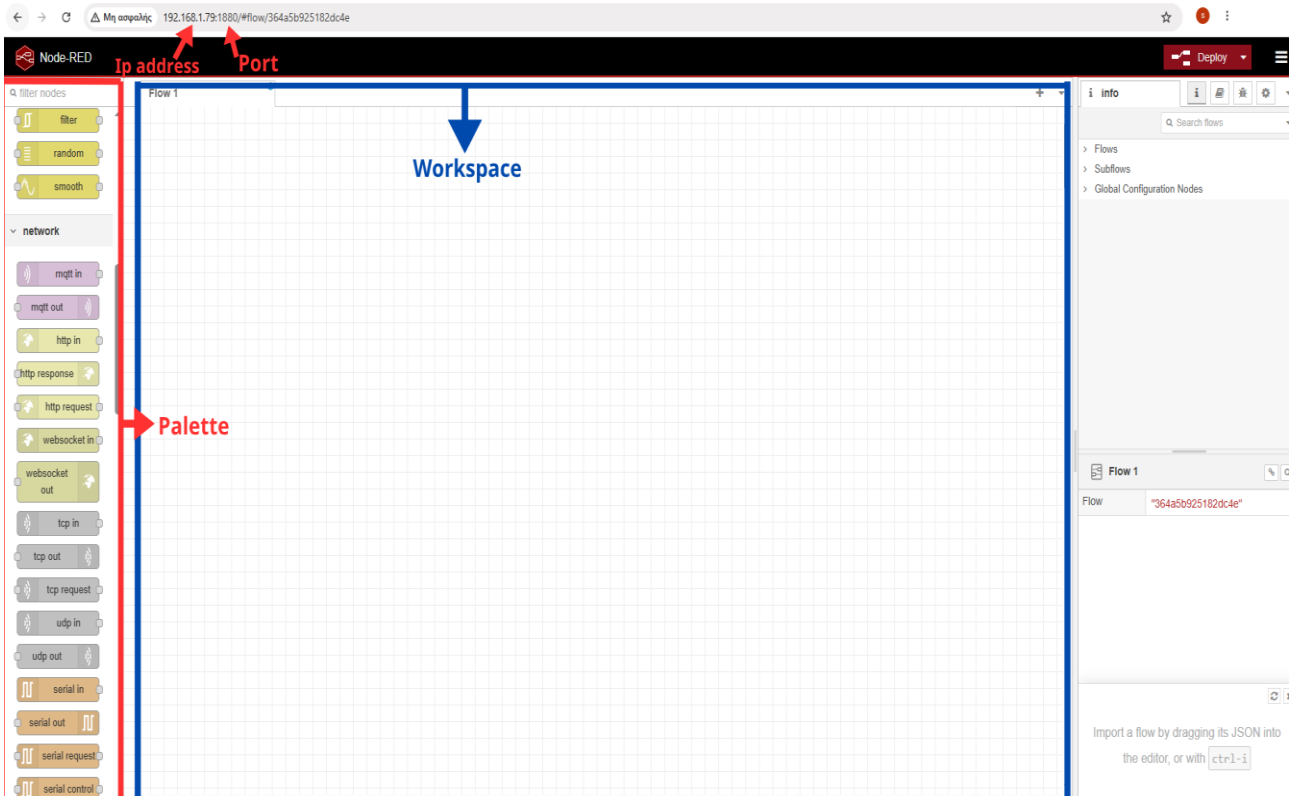
Αφού ρυθμίστηκε το αρχείο ενεργοποιήθηκε η υπηρεσία μέσω του systemctl CLI εργαλείου. Η διαδικασία πραγματοποιείται με τις παρακάτω εντολές:

```
(venv) aitos@raspberrypi:~/drone_ai $ sudo systemctl enable drone_node.service
(venv) aitos@raspberrypi:~/drone_ai $ sudo systemctl daemon-reload
```

6.3.3 Δημιουργία του Dashboard στο Node-RED

Μετά την υλοποίηση του script τηλεμετρίας και την αποστολή των δεδομένων μέσω MQTT, ακολούθησε η δημιουργία του Dashboard στο περιβάλλονανάπτυξης του Node-RED.

Αρχικά πραγματοποιήθηκε πρόσβαση στο Node-Red μέσω του browser χρησιμοποιώντας την τοπική διεύθυνση IP του Raspberry Pi και τη θύρα 1880. Μέσα από το περιβάλλον αυτό πραγματοποιήθηκε η σχεδίαση του Dashboard με τη χρήση των διαθέσιμων nodes από την παλέτα εργαλείων που βρίσκεται στο αριστερό τμήμα της οθόνης, όπως εμφανίζεται και στο σχήμα 6.18.



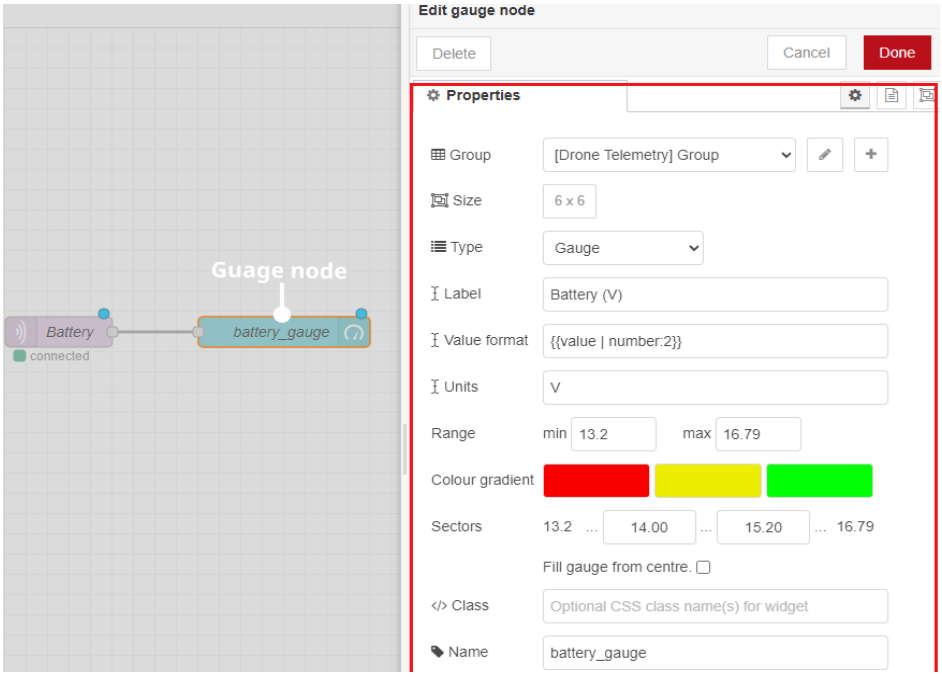
Σχήμα 6.18: Περιβάλλον ανάπτυξης Node-Red

Το πρώτο στάδιο αφορά τη λήψη των δεδομένων από τα αντίστοιχα MQTT topics του Mosquitto Broker. Για τον σκοπό αυτό χρησιμοποιήθηκαν κόμβοι τύπου MQTT in, οι οποίοι αναλαμβάνουν τη λήψη των δεδομένων τηλεμετρίας από τα αντίστοιχα topics. Συνολικά χρησιμοποιήθηκαν πέντε διαφορετικά MQTT in nodes. Κάθε MQTT in node ρυθμίστηκε ώστε να συνδέεται με τη διεύθυνση localhost και τη θύρα 1883 καθώς ο Mosquitto Broker εκτελείται τοπικά στο ίδιο Raspberry Pi με το Node-RED. Παράλληλα, σε κάθε node ορίστηκε το αντίστοιχο topic τηλεμετρίας στο οποίο πραγματοποιείται εγγραφή για τη λήψη των δεδομένων. Χρησιμοποιήθηκε το drone/battery για την προβολή της τάσης της μπαταρίας, το drone/vertical για την προβολή της κατακόρυφης ταχύτητας, το drone/alt για την προβολή του υψομέτρου, το drone/speed για την ταχύτητα εδάφους και το drone/gps για την προβολή των γεωγραφικών συντεταγμένων.

Σχήμα 6.19: Ενδεικτική ρύθμιση παραμέτρων κόμβου MQTT in

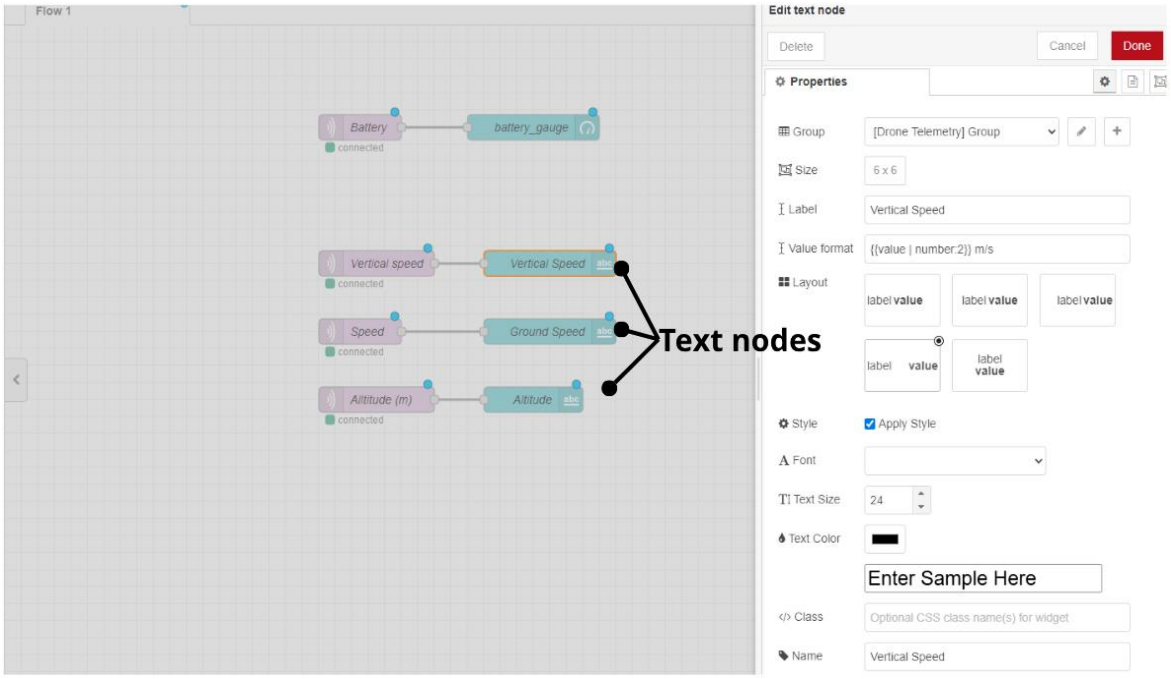
Στη συνέχεια, οι κόμβοι εισόδου συνδέθηκαν με γραφικά στοιχεία. Για την εμφάνιση των τιμών της μπα

ταρίας χρησιμοποιήθηκε ένας gauge node ο οποίος ρυθμίστηκε ώστε να εμφανίζει την τάση της μπαταρίας σε volt μαζί με τα αντίστοιχα όρια και τις χρωματικές ενδείξεις όπως αποτυπώνεται και στο σχήμα 6.20.



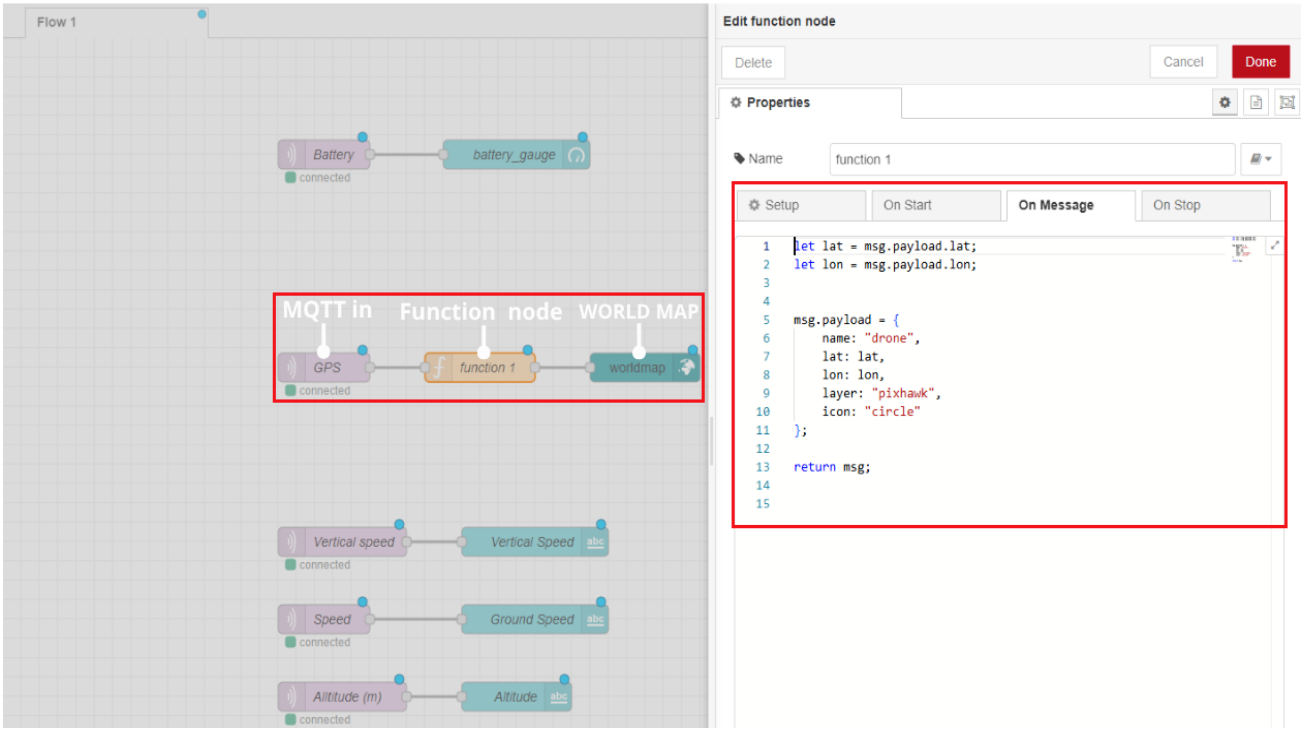
Σχήμα 6.20: Παραμετροποίηση Gauge node για την ένδειξη τάσης της μπαταρίας

Αντίστοιχα, για την εμφάνιση των υπόλοιπων δεδομένων τηλεμετρίας όπως η ταχύτητα, το ύψος και η κατακόρυφη ταχύτητα χρησιμοποιήθηκαν text nodes τα οποία προβάλλουν σε πραγματικό χρόνο τις τιμές που λαμβάνονται από το drone.



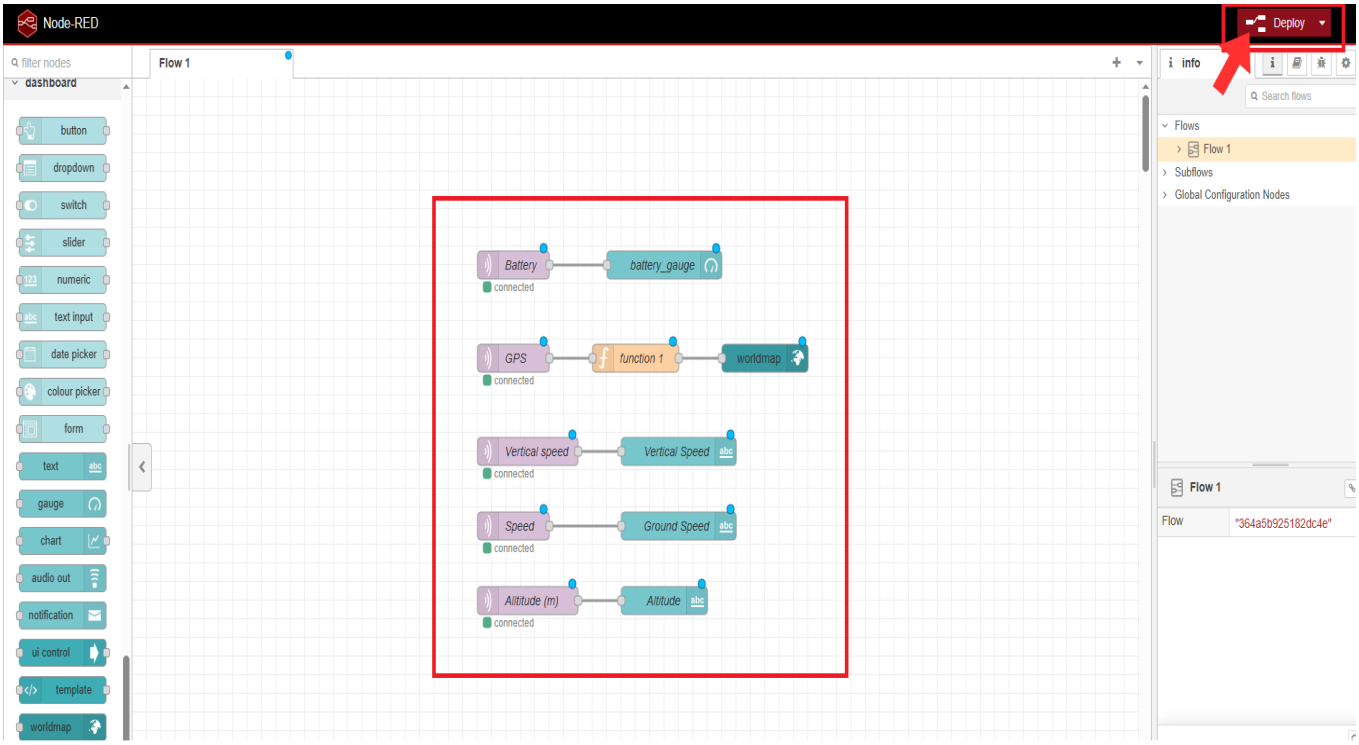
Σχήμα 6.21: Ενδεικτική ρύθμιση text node για την εμφάνιση δεδομένων τηλεμετρίας.

Έπειτα, για την προβολή της θέσης του drone χρησιμοποιήθηκε το worldmap node, το οποίο δίνει τη δυνατότητα εμφάνισης των δεδομένων GPS πάνω σε χάρτη. Πριν από το worldmap τοποθετήθηκε ένα function node με σκοπό την κατάλληλη επεξεργασία των δεδομένων που λαμβάνονται από το topic drone/gps. Εντός του function node αρχικά εξάγονται οι συντεταγμένες του γεωγραφικού πλάτους και μήκους από τα δεδομένα GPS που λαμβάνονται, διαδικασία η οποία απεικονίζεται στο σχήμα 6.22. Στη συνέχεια δημιουργούνται οι κατάλληλες πληροφορίες που χρειάζεται το worldmap node, όπως οι συντεταγμένες, το όνομα του στίγματος, το επίπεδο εμφάνισης και το εικονίδιο του χάρτη. Με αυτόν τον τρόπο το worldmap μπορεί να ενημερώνει τη θέση πάνω στον χάρτη χωρίς να δημιουργείται νέο σημείο σε κάθε μέτρηση GPS. Τέλος, το επεξεργασμένο πακέτο προωθείται από το function node προς το worldmap node όπου πραγματοποιείται η εμφάνιση της θέσης στο χάρτη.



Σχήμα 6.22: Κώδικας μορφοποίησης των δεδομένων GPS εντός του Function node

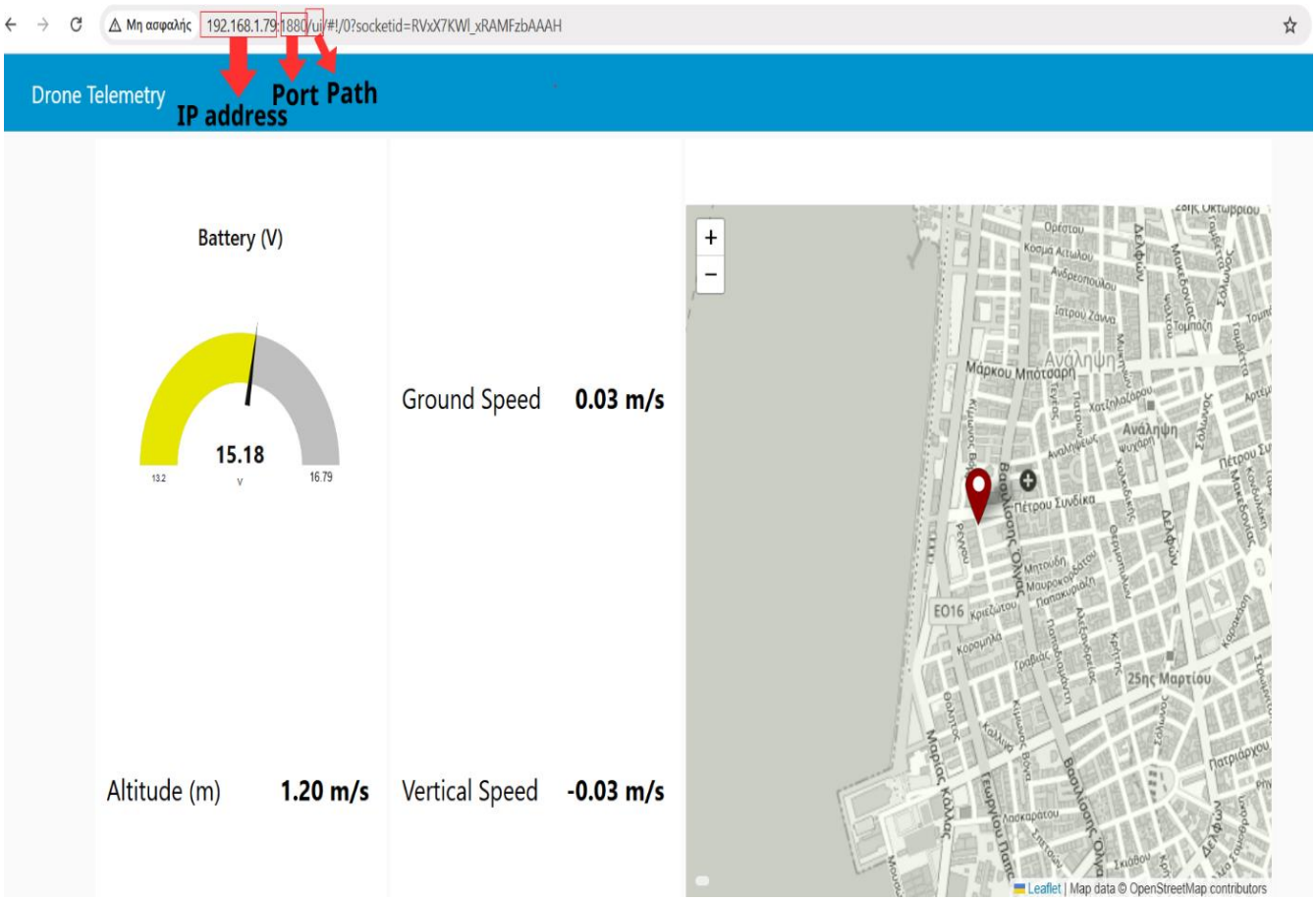
Αφού ολοκληρώθηκαν οι συνδέσεις και οι ρυθμίσεις από όλους τους κόμβους, το Dashboard τέθηκε σε λειτουργία με την επιλογή deploy, όπως φαίνεται και από το σχήμα 6.23.



Σχήμα 6.23: Ενεργοποίηση της ροής μέσω της επιλογής Deploy.

Αμέσως μετά την ολοκλήρωση της ανάπτυξης χρησιμοποιήθηκε η ίδια διεύθυνση IP του Node-RED με την προσθήκη της κατάληξης /ui στο τέλος της διεύθυνσης. Η ενέργεια αυτή ανοίγει στον browser το

τελικό γραφικό περιβάλλον όπου πραγματοποιείται η ζωντανή παρακολούθηση της τηλεμετρίας του drone, όπως φαίνεται και από το σχήμα 6.24.



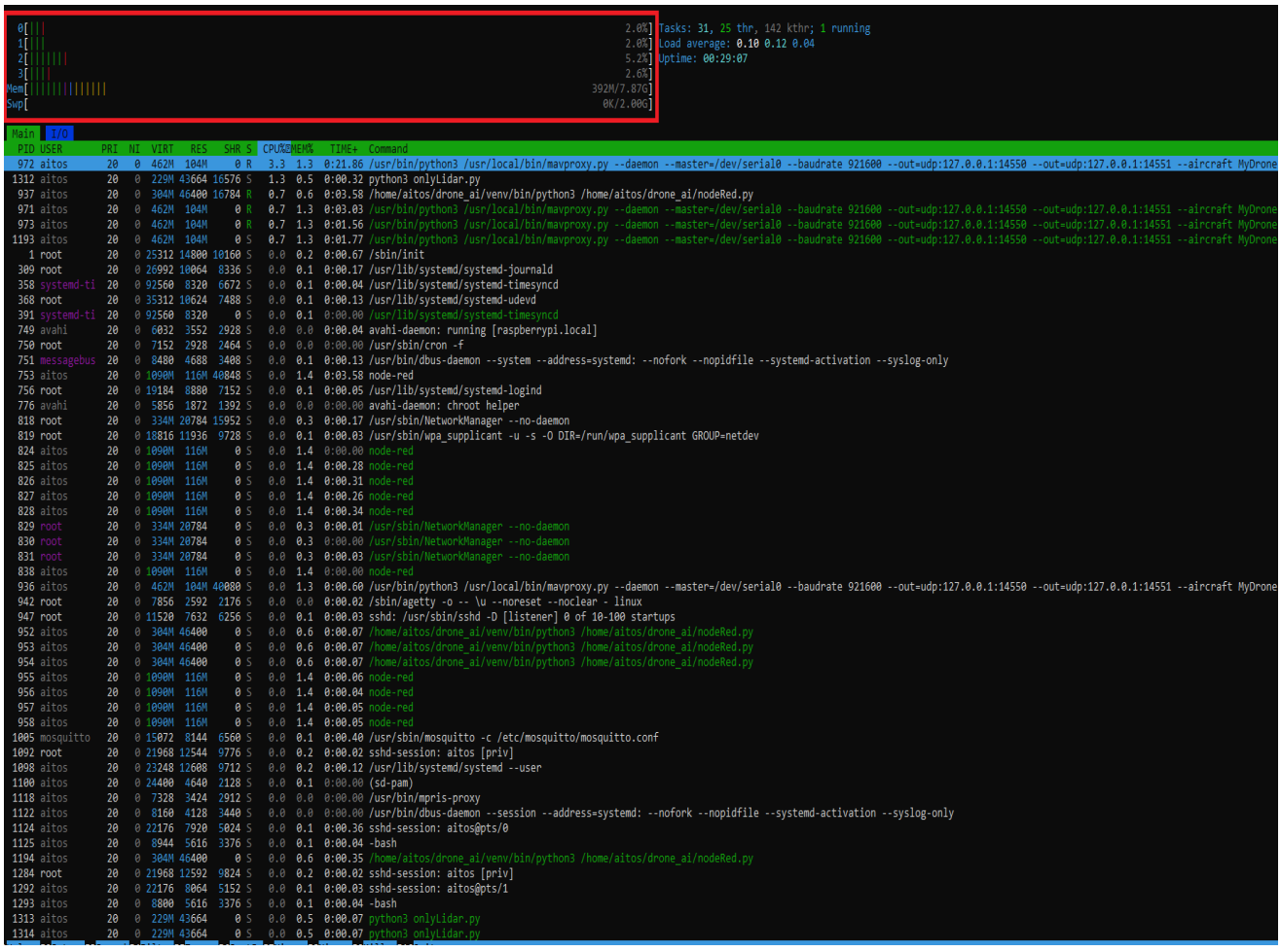
Σχήμα 6.24: Γραφικό περιβάλλον τηλεμετρίας

Κεφάλαιο 7ο: Πειραματικά Αποτελέσματα και Σύγκριση

7.1 Δοκιμές με Χρήση Αισθητήρα LiDAR

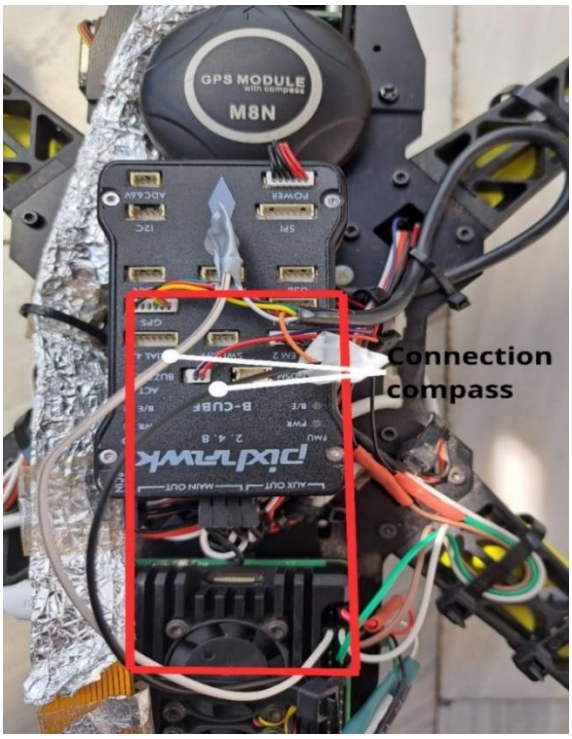
Στο πρώτο στάδιο των πειραματικών δοκιμών το σύστημα αξιολογήθηκε χρησιμοποιώντας μόνο τον αισθητήρα LiDAR. Ο αλγόριθμος αποφυγής βασιζόταν μόνο στις μετρήσεις απόστασης του αισθητήρα χωρίς δυνατότητα αναγνώρισης του είδους του εμποδίου που εντοπίζει.

Κατά το στάδιο αυτό, η εκτέλεση του κώδικα ήταν ιδιαίτερα άμεση, καθώς το Raspberry Pi λαμβάνει αποκλειστικά τις μετρήσεις του LiDAR χωρίς να εκτελεί πρόσθετες διεργασίες, όπως η επεξεργασία ή η μετάδοση εικόνας. Ως εκ τούτου, η απαιτούμενη επεξεργαστική ισχύς ήταν ελάχιστη, όπως απεικονίζεται και στο σχήμα 7.1, η επικοινωνία με τον Pixhawk γινόταν σε πραγματικό χρόνο.



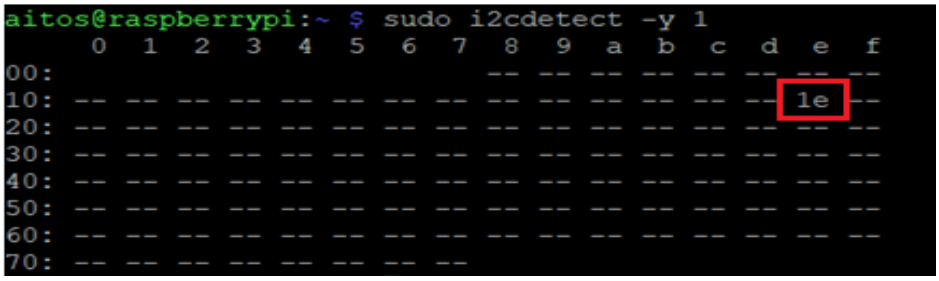
Σχήμα 7.1: Χρήση CPU σε λειτουργία αποφυγής με LiDAR

Ωστόσο, στην πορεία των δοκιμών παρουσιάστηκε πρόβλημα στην επικοινωνία της εξωτερικής πυξίδας με τον ελεγκτή πτήσης. Προκειμένου να διαπιστωθεί αν το πρόβλημα προέρχεται από την ίδια την πυξίδα ή από τον ελεγκτή πτήσης πραγματοποιήθηκε δοκιμή σύνδεσης της πυξίδας στις υποδοχές I2C του Raspberry Pi όπως φαίνεται και στο σχήμα 7.2.



Σχήμα 7.2: Σύνδεση πυξίδας στο Raspberry Pi

Μέσω της εντολής “i2cdetect -y 1” ,όπως παρατηρείται στο σχήμα 7.3 ,πραγματοποιήθηκε έλεγχος για την ανίχνευση της πυξίδας. Κατά τη δοκιμή το Raspberry Pi αναγνώρισε την πυξίδα στην διεύθυνση 1e γεγονός που σηματοδοτεί ότι η πυξίδα μπορεί να μεταδώσει δεδομένα, όπως απεικονίζεται και στο σχήμα 7.3.



Σχήμα 7.3: Εντολή ανίχνευσης της πυξίδας στο Raspberry Pi

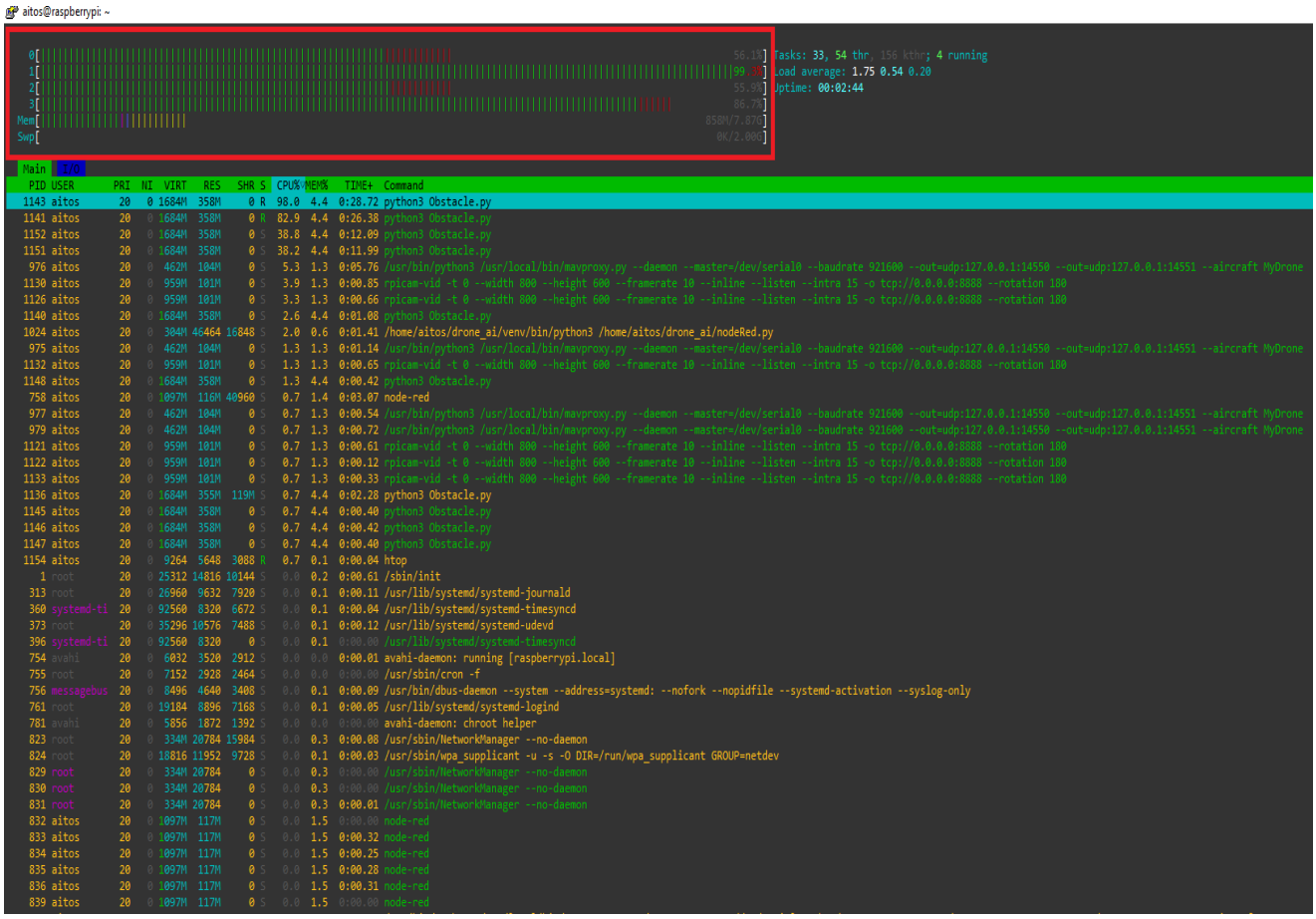
Με βάση τα αποτελέσματα των δοκιμών, εκτιμήθηκε ότι το πρόβλημα προερχόταν από τη θύρα επικοινωνίας I2C του συγκεκριμένου ελεγκτή πτήσης. Ο ελεγκτής που χρησιμοποιήθηκε αποτελεί έναν κλώνο του αυθεντικού Pixhawk και δεν διαθέτει την ίδια ποιότητα κατασκευής και αξιοπιστία, γεγονός που συνέβαλε στην εμφάνιση της δυσλειτουργίας.

Λόγω αυτού του περιορισμού, το σύστημα λειτούργησε στη συνέχεια χρησιμοποιώντας την εσωτερική πυξίδα του flight controller αντί της εξωτερικής. Η εσωτερική πυξίδα βρίσκεται κοντά στα ηλεκτρονικά κυκλώματα και στα καλώδια τροφοδοσίας του drone, με αποτέλεσμα να επηρεάζεται πιο εύκολα από ηλεκτρομαγνητικές παρεμβολές. Οι παρεμβολές αυτές προκαλούσαν αποκλίσεις στον προσανατολισμό του drone κατά τη διάρκεια της πτήσης.

7.2 Δοκιμές με Χρήση Αισθητήρα LiDAR και Κάμερας

Έπειτα, στις επόμενες δοκιμές ενσωματώθηκε και η κάμερα μαζί με τον αισθητήρα LiDAR. Με αυτόν τον τρόπο το σύστημα μπορούσε πλέον να αποφεύγει εμπόδια και ταυτόχρονα να εντοπίζει ανθρώπους μέσω του αλγορίθμου YOLO. Το σύστημα δεν βασιζόταν μόνο στις μετρήσεις απόστασης, αλλά επεξεργαζόταν ταυτόχρονα και εικόνα σε πραγματικό χρόνο από την κάμερα.

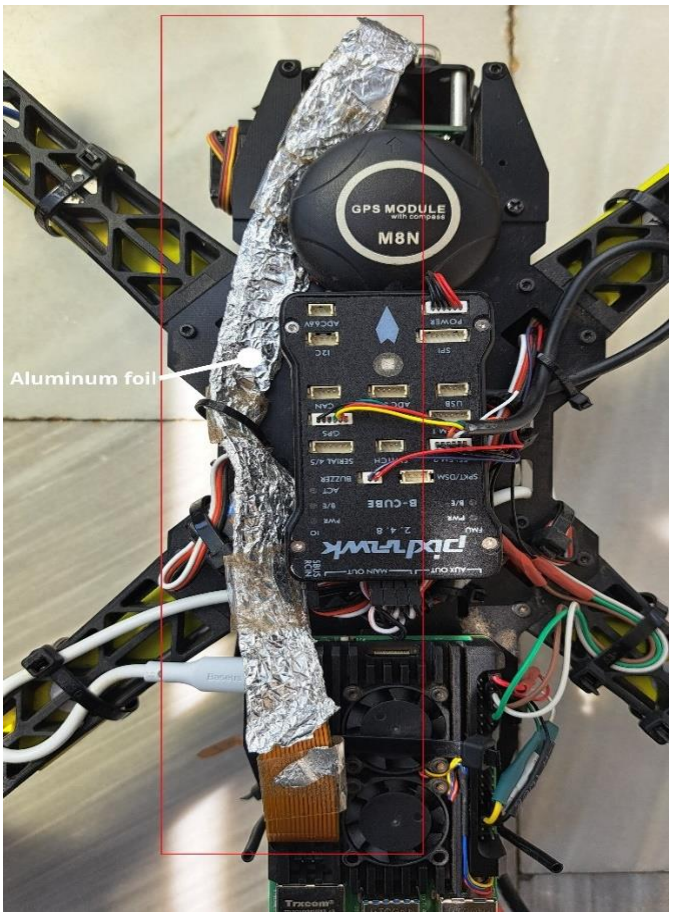
Η προσθήκη όμως της κάμερας αύξησε σημαντικά το φόρτο του επεξεργαστή, όπως μπορούμε να δούμε και από το σχήμα 7.4, λόγω της παράλληλης εκτέλεσης πολλαπλών διεργασιών. Σε αυτό περιλαμβανόταν η λήψη της εικόνας, η επεξεργασία των καρέ, η εκτέλεση του YOLO, το livestreaming και ο αλγόριθμος αποφυγής. Ως αποτέλεσμα ορισμένες εντολές αποφυγής καθυστερούν να εκτελεστούν προκαλώντας όχι τόσο ομαλή κίνηση του drone κατά τη διάρκεια της αποφυγής εμποδίων.



Σχήμα 7.4: Χρήση CPU σε λειτουργία αποφυγής με κάμερα και LiDAR

Επιπλέον, λόγω της λειτουργίας με την εσωτερική πυξίδα του flight controller, οι ήδη υπάρχουσες ηλεκτρομαγνητικές παρεμβολές έγιναν πιο έντονες μετά την προσθήκη της κάμερας. Ως αποτέλεσμα, σε ορισμένες δοκιμές το drone παρουσίαζε αποκλίσεις στον προσανατολισμό του.

Παράλληλα, κατά τη μετάδοση της εικόνας παρατηρούνταν ορισμένες περιπτώσεις όπου η ροή βίντεο πάγωνε προσωρινά. Το φαινόμενο αυτό αποδόθηκε σε ηλεκτρομαγνητικές παρεμβολές που επηρέαζαν το καλώδιο σύνδεσης της κάμερας. Για τη μείωση του προβλήματος πραγματοποιήθηκε θωράκιση της καλωδιοταινίας με αλουμινόχαρτο γεγονός που βελτίωσε τη σταθερότητα της εικόνας, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 7.5: Ηλεκτρομαγνητική θωράκιση καλωδίου με φύλλο αλουμινίου.

Παρόλα αυτά, η χρήση της κάμερας πρόσθεσε σημαντικά πλεονεκτήματα στο σύστημα αποφυγής. Με το YOLO έγινε δυνατή η αναγνώριση ανθρώπων, επιτρέποντας στο drone να μειώνει ταχύτητα και να κρατά μεγαλύτερη απόσταση ασφαλείας κάτι που δεν ήταν εφικτό στη λειτουργία μόνο με LiDAR, όπου όλα τα εμπόδια αντιμετωπίζονταν με τον ίδιο τρόπο.

7.3 Συγκριτική Αξιολόγηση

Συνοψίζοντας, από τις πειραματικές δοκιμές προέκυψαν σημαντικές διαφορές μεταξύ της λειτουργίας μόνο με LiDAR και της λειτουργίας με LiDAR και κάμερα. Στην πρώτη περίπτωση, το Raspberry Pi επεξεργαζόταν αποκλειστικά δεδομένα αποστάσεων από το LiDAR με αποτέλεσμα η χρήση των υπολογιστικών πόρων να παραμένει χαμηλή. Όπως φαίνεται και στον πίνακα 7.1 βάσει των μετρήσεων από το σχήμα 7.1, το σύστημα παρουσίαζε πιο σταθερή λειτουργία και πιο ομαλή συμπεριφορά κατά την πτήση, καθώς οι εντολές αποφυγής εκτελούνταν άμεσα χωρίς σημαντικές καθυστερήσεις.

Από την άλλη, με την προσθήκη του αλγορίθμου YOLO το σύστημα απέκτησε τη δυνατότητα αναγνώρισης ανθρώπων και επομένως πιο ευφυή συμπεριφορά αποφυγής. Ωστόσο, η ταυτόχρονη λειτουργία του LiDAR, της επεξεργασίας εικόνας μέσω του YOLO και του livestreaming της εικόνας από την κάμερα αύξησε σημαντικά τον φόρτο του επεξεργαστή. Όπως παρουσιάζεται και στον πίνακα 7.1 βάσει των μετρήσεων από το σχήμα 7.4, η λειτουργία αυτή απαιτούσε σημαντικά περισσότερους υπολογιστικούς πόρους γεγονός που επηρέαζε τη σταθερότητα του συστήματος κατά τη διάρκεια της πτήσης.

Συνεπώς, από τις πειραματικές δοκιμές προέκυψε ότι ο αλγόριθμος αποφυγής λειτουργεί επιτυχώς και στις δύο περιπτώσεις όμως η συνολική απόδοση του συστήματος επηρεάζεται άμεσα από τους περιορισμούς του χρησιμοποιούμενου hardware. Το αυξημένο υπολογιστικό φορτίο της εκτέλεσης αλγορίθμων τεχνητής νοημοσύνης σε πραγματικό χρόνο σε συνδυασμό με τις ηλεκτρομαγνητικές παρεμβολές που επηρέαζαν την εσωτερική πυξίδα του flight controller επηρέασαν τη σταθερότητα και τη συνολική λειτουργία του UAV κατά τη διάρκεια της πτήσης.

Πίνακας 7.1: Συγκριτική αξιολόγηση πόρων του συστήματος

Παράμετρος	Lidar	Lidar και YOLO
Χρήση CPU	2 έως 6%	55 έως 98%
Χρήση RAM	392 MB	858 MB
Livestreaming	Όχι	Ναι
Επεξεργασία εικόνας	Όχι	Ναι
Σταθερότητα πτήσης	Υψηλή	Μειωμένη

Κεφάλαιο 8ο: Συμπεράσματα

8.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία πραγματοποιήθηκε η υλοποίηση ενός drone με δυνατότητα αποφυγής εμποδίων μέσω αισθητήρα LiDAR και υπολογιστικής όρασης. Η παρούσα διπλωματική εργασία δεν έμεινε μόνο στην ανάπτυξη του λογισμικού. Επίσης περιελάβανε την κατασκευή και τη συναρμολόγηση του ίδιου του UAV καθώς και τη διασύνδεση όλων των απαραίτητων ηλεκτρονικών υποσυστημάτων. Με τον τρόπο αυτό αναπτύχθηκε μία ολοκληρωμένη υλοποίηση η οποία συνδυάζει τόσο το υλικό όσο και το λογισμικό ενός αυτόνομου συστήματος πτήσης.

Η χρήση του αισθητήρα έδωσε στο σύστημα τη δυνατότητα μέτρησης αποστάσεων και αντίδρασης απέναντι σε εμπόδια σε πραγματικό χρόνο ενώ η ενσωμάτωση κάμερας και αλγορίθμου YOLO επέτρεψε την αναγνώριση ανθρώπινης παρουσίας και την πιο έξυπνη προσαρμογή της συμπεριφοράς του drone κατά την πτήση. Παράλληλα, υλοποιήθηκε σύστημα τηλεμετρίας και ζωντανής μετάδοσης εικόνας παρέχοντας τη δυνατότητα συνεχούς παρακολούθησης της κατάστασης του UAV και της εικόνας κατά τη διάρκεια λειτουργίας του.

Μέσα από τις πειραματικές δοκιμές διαπιστώθηκε ότι η λειτουργία μόνο με LiDAR παρουσίαζε μεγαλύτερη σταθερότητα και χαμηλότερη κατανάλωση υπολογιστικών πόρων. Αντίθετα, η ταυτόχρονη λειτουργία LiDAR, και αλγορίθμων τεχνητής νοημοσύνης αύξησε σημαντικά το υπολογιστικό φορτίο του Raspberry Pi επηρεάζοντας τη συνολική απόκριση του συστήματος. Επιπλέον, οι ηλεκτρομαγνητικές παρεμβολές που επηρέαζαν την εσωτερική πυξίδα του flight controller συνέβαλαν σε αποκλίσεις κατά τον προσανατολισμό του UAV.

Συνοψίζοντας, η παρούσα διπλωματική εργασία έδειξε ότι ο συνδυασμός αισθητήρων, υπολογιστικής όρασης και ενσωματωμένων συστημάτων μπορεί να συμβάλει στη δημιουργία έξυπνων και αυτόνομων εφαρμογών πτήσης. Παράλληλα, αναδείχθηκαν και οι περιορισμοί που προκύπτουν από το διαθέσιμο hardware όταν απαιτείται επεξεργασία δεδομένων και τεχνητή νοημοσύνη σε πραγματικό χρόνο.

8.2 Προτάσεις Βελτίωσης

Η υλοποίηση του συστήματος αποτελεί μια ολοκληρωμένη λύση. Ωστόσο λόγω του εύρους των τεχνολογιών που εμπλέκονται υπάρχουν αρκετά σημεία που μπορούν να εξελιχθούν περαιτέρω. Οι βασικές προτάσεις βελτιώσεις είναι οι εξής:

1) Εκπαίδευση μοντέλων για την αναγνώριση πολλαπλών αντικειμένων. Τα μοντέλα που χρησιμοποιούνται στην τωρινή υλοποίηση είναι ρυθμισμένα αποκλειστικά για το εντοπισμό ανθρώπινης παρουσίας. Παρότι αυτό καλύπτει έναν κρίσιμο στόχο παραμένει μια περιορισμένη προσέγγιση. Η ενσωμάτωση επιπλέον μοντέλων αναγνώρισης θα μπορούσε να επιτρέψει στο UAV να εντοπίζει και άλλα είδη εμποδίων βελτιώνοντας τη συνολική λειτουργία αποφυγής.

2) Εξωτερική επεξεργασία της υπολογιστικής όρασης. Η επεξεργασία εικόνας γίνεται απευθείας στο Raspberry Pi κάτι που αυξάνει τον υπολογιστικό φόρτο. Η χρήση εξωτερικών επιταχυντών όπως το Coral USB Accelerator θα επέτρεπε πολύ υψηλότερους ρυθμούς ανάλυσης εικόνας και θα μείωνε τον υπολογιστικό φόρτο του συστήματος.

3) Αναβάθμιση αισθητήρων για τρισδιάστατη χαρτογράφηση. Η τωρινή λύση βασίζεται σε έναν απλό LiDAR ο οποίος λειτουργεί αξιόπιστα αλλά δεν προσφέρει τρισδιάστατη αντίληψη του

χώρου. Η χρήση 3D LiDAR θα παρείχε πλήρη τρισδιάστατα δεδομένα και θα επέτρεπε πιο αποδοτικό σχεδιασμό διαδρομών.

4) Μετάβαση σε πλήρως αυτόνομη πλοήγηση. Στην παρούσα μορφή, το UAV λειτουργεί ημιαυτόνομα, με τον αλγόριθμο να παρεμβαίνει μόνο διορθωτικά. Σε μελλοντική υλοποίηση, το drone θα μπορούσε να αποκτήσει πλήρως αυτόνομη πλοήγηση από ένα σημείο σε ένα άλλο χωρίς παρέμβαση χειριστή.

5) Δυναμική παράκαμψη και αποκλεισμός περιοχών. Η τωρινή υλοποίηση επικεντρώνεται στο να εντοπίζει έναν άνθρωπο και προσαρμόζει την ταχύτητα και την απόσταση πτήσης. Σε μελλοντική επέκταση, η λειτουργία αυτή μπορεί να εξελιχθεί ώστε το drone να αποκλείει την περιοχή όπου εντοπίστηκε το αντικείμενο. Στη συνέχεια, θα μπορεί να υπολογίζει και να ακολουθεί μια νέα εναλλακτική διαδρομή, παρακάμπτοντας το εμπόδιο με πιο ασφαλή και αποδοτικό τρόπο.

Συνοψίζοντας, αυτά αποτελούν τα σημαντικότερα σημεία που μπορούν να βελτιωθούν χωρίς να είναι τα μοναδικά. Ένα σύστημα που συνδυάζει τεχνητή νοημοσύνη, τηλεμετρία και εναέρια πλοήγηση εξελίσσεται συνεχώς.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Pang, B., & Wang, C. H. J. (2021). *Framework of Level-of-Autonomy-based Concept of Operations:* UAS Capabilities. 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC).

[2] Ariante, G., & Del Core, G. (2025). Unmanned Aircraft Systems (UASs): Current State, Emerging Technologies, and Future Trends. Drones, 9.

[3] Merei, A., et al. (2025). A Survey on Obstacle Detection and Avoidance Methods for UAVs. Drones, 9(203).

[4] Alsadik, Bashar & Nex, Francesco. (2021). The Rise in UAV Inspections for Civil Infrastructure https://www.gim-international.com/content/article/the-rise-in-uav-inspections-for-civil-infrastructure. GIM International.

[5] N. Gageik, P. Benz and S. Montenegro, "Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors," in IEEE Access, vol. 3, pp. 599-609, 2015.

[6] K. Gao and K. Xu, "Research on Autonomous Obstacle Avoidance Flight Path Planning of Rotating Wing UAV," 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC), Dalian, China, 2017, pp. 1200-1203.

[7] Illmann, B., et al. (2020). "Investigation of Low-Cost LiDAR Sensors for Mobile Robotics." 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).

[8] Y. Ren, Y. Cai, H. Li, N. Chen, F. Zhu, L. Yin, F. Kong, R. Li, and F. Zhang, “A Survey on LiDAR-Based Autonomous Aerial Vehicles,” IEEE/ASME Transactions on Mechatronics, pp. 1–17, 2025, doi: 10.1109/TMECH.2025.3599633.

[9] B. Ruf, S. Monka, M. Kollmann, and M. Grinberg, “Real-time on-board obstacle avoidance for UAVs based on embedded stereo vision,” The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 363–370, Sep. 2018.

[10] Y. Yu, W. Tingting, C. Long and Z. Weiwei, "Stereo vision based obstacle avoidance strategy for quadcopter UAV," 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 2018, pp. 490-494.

[11] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013, pp. 1750-1757.

[12] H. Yu, F. Zhang, P. Huang, C. Wang and L. Yuanhao, "Autonomous Obstacle Avoidance for UAV based on Fusion of Radar and Monocular Camera," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 5954-5961.

[13] Z. Xu, X. Zhan, B. Chen, Y. Xiu, C. Yang and K. Shimada, "A real-time dynamic obstacle tracking and mapping system for UAV navigation and collision avoidance with an RGB-D camera," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 10645-10651.

[14] Wang, D.; Li, W.; Liu, X.; Li, N.; Zhang, C. UAV environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution. Comput. Electron. Agric. 2020, 175, 105523.

[15] A. Ghaddar and A. Merei, “EAOA: Energy-Aware Grid-Based 3D-Obstacle Avoidance in Coverage Path Planning for UAVs,” Future Internet, vol. 12, no. 2, p. 29, 2020, doi: 10.3390/fi12020029.

[16] Park, B., & Oh, H. (2020). "Vision-Based Obstacle Avoidance for UAVs via Imitation Learning with Sequential Neural Networks". International Journal of Aeronautical and Space Sciences, 21, 768–779.

[17] A. N and U. S. V, "Custom Based Obstacle Detection Using Yolo v3 for Low Flying Drones," 2021 International Conference on Circuits, Controls and Communications (CCUBE), Bangalore, India, 2021, pp. 1-6, doi: 10.1109/CCUBE53681.2021.9702739.

[18] A. Sonny, S. R. Yeduri, and L. R. Cenkeramaddi, “Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance,” Applied Soft Computing, vol. 147, 2023, Art. no. 110773. doi: 10.1016/j.asoc.2023.110773.

[19] Q. Geng, H. Shuai and Q. Hu, "Obstacle avoidance approaches for quadrotor UAV based on backstepping technique," 2013 25th Chinese Control and Decision Conference (CCDC), Guiyang, China, 2013, pp. 3613-3617, doi: 10.1109/CCDC.2013.6561575.

[20] E. Ferrera, A. Alcántara, J. Capitán, A. R. Castaño, P. J. Marrón, and A. Ollero, “Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments,” Sensors, vol. 18, no. 12, p. 4101, 2018, doi: 10.3390/s18124101.

[21] A. Moffatt, E. G. Platt, B. Mondragon, A. Kwok, D. Uryeu, and S. Bhandari, “Obstacle Detection and Avoidance System for Small UAVs using a LiDAR,” in Proc. 2020 Int. Conf. Unmanned Aircraft Systems (ICUAS), 2020, pp. 633–640.

[22] S. K. NT, G. U. Sai, P. K. Duba and P. Rajalakshmi, "Real Time Vision Based Obstacle Avoidance for UAV using YOLO in GPS Denied Environment," 2023 OITS International Conference on Information Technology (OCIT), Raipur, India, 2023, pp. 586-591.

[23] Z. Zhang, M. Xiong and H. Xiong, "Monocular Depth Estimation for UAV Obstacle Avoidance," 2019 4th International Conference on Cloud Computing and Internet of Things (CCIOT), Changchun, China, 2019, pp. 43-47.

[24] R. Ait-Jellal and A. Zell, "Outdoor obstacle avoidance based on hybrid visual stereo SLAM for an autonomous quadrotor MAV," 2017 European Conference on Mobile Robots (ECMR), Paris, France, 2017, pp. 1-8, doi: 10.1109/ECMR.2017.8098686.

[25] Brother Hobby, “Returner R3 2207 Brushless Motor Specifications,” BrotherHobby Store. [Online]. Available: https://www.brotherhobbystore.com/returner-r3-2207-motor-p0023.html.

[26] ArduPilot Dev Team, “Pixhawk Hardware” ArduPilot Copter Documentation. Available: https://ardupilot.org/copter/docs/common-pixhawk-overview.html.

[27] ArduPilot Dev Team, “GPS and Compass Modules,” ArduPilot Copter Documentation. Available: https://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html.

[28] FlySky, “FS-i6 Transmitter and Receiver User Manual,” FlySky RC. Available: https://www.flysky-cn.com/fsi6.

[29] FlySky, “FS-CVT01 Voltage Sensor Technical Specifications,” FlySky RC. Available: https://www.flysky-cn.com/cvt01-canshu.

[30] Raspberry Pi Ltd, “Raspberry Pi 5 Board Computer Product Specifications,” Raspberry Pi. Available: https://www.raspberrypi.com/products/raspberry-pi-5.

[31] Raspberry Pi Ltd, “Raspberry Pi Active Cooler” Raspberry Pi Assets. Available: https://pip-assets.raspberrypi.com/categories/993-raspberry-pi-active-cooler.

[32] Benewake, “TFmini Plus LiDAR ToF Distance Sensor Overview,” Benewake Site. Available: https://en.benewake.com/TFminiPlus/index.html.

[33] Raspberry Pi Ltd, “Raspberry Pi Camera Module v2” Raspberry Pi. Available: https://www.raspberrypi.com/products/camera-module-v2.

[34] Adafruit Industries, “Adafruit Raspberry Pi Camera Board Case Specifications,” Available: https://www.adafruit.com/product/3253#technical-details.

[35] ArduPilot Dev Team, “Connect ESCs and Motors to Flight Controller,” ArduPilot Copter Documentation. Available: https://ardupilot.org/copter/docs/connect-escs-and-motors.html.

[36] Geeetech, “Pixhawk Compass Calibration Diagram” Available: https://www.geeetech.com/Documents/Pixhawk%20%20user%20maual%20.pdf.

[37] ArduPilot Dev, “Navigation Architecture Diagram,” Available: https://ardupilot.org/dev/docs/apmcopter-programming-attitude-control-2.html.