ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ

ΥΠΟΛΟΓΙΣΤΩΝ

ΤΗΛΕΜΕΤΡΙΚΗ ΕΦΑΡΜΟΓΗ ΒΑΣΙΣΜΕΝΗ ΣΕ ΜΙΚΡΟΕΛΕΓΚΤΗ ΓΙΑ ΑΠΟΦΥΓΗ ΣΥΓΚΡΟΥΣΕΩΝ ΣΕ ΠΟΛΥΚΟΠΤΕΡΑ

Πρόκας Νικόλαος

Εξεταστική επιτροπή

Καθ. Απόστολος Δόλλας (ΗΜΜΥ)

Αναπλ. Καθ. Ιωάννης Παπαευσταθίου (ΗΜΜΥ)

Επικ. Καθ. Παναγιώτης Παρτσινέβελος (ΜΗΧΟΠ)

Χανιά, Φεβρουάριος 2016

Περίληψη

Την τελευταία δεκαετία, η μετάβαση της βιομηχανίας κατασκευής και αξιοποίησης μη επανδρωμένων ιπτάμενων οχημάτων (UAV - Unmanned Airborne Vehicles) από στρατιωτικές σε αμιγώς εμπορικές, βιομηχανικές και περιβαλλοντικές εφαρμογές έχει αυξήσει κατακόρυφα την διείσδυσή τους στην κοινωνία. Η συνεχής τεχνολογική βελτίωση των UAVs σε συνδυασμό με την μείωση του κόστους προμήθειάς τους, τα καθιστούν ιδιαίτερα δημοφιλή για τον απλό πολίτη.

Ωστόσο, τα περισσότερα από τα εμπορικά συστήματα χαμηλού κόστους που διατίθενται στην αγορά έχουν περιορισμένες τεχνικές δυνατότητες (π.χ. απλή λήψη φωτογραφιών) και η προσαρμογή σε αυτά νέων αισθητήρων καθίσταται ιδιαίτερα απαιτητική εάν όχι αδύνατη.

Στόχος της παρούσας εργασίας είναι η προσθήκη ενός ενσωματωμένου (embedded) συστήματος το οποίο θα επιτρέπει στον χρήστη να προσαρμόζει διαφορετικού τύπου αισθητήρες, ανάλογα με τις ανάγκες εφαρμογής του, βελτιώνοντας έτσι την χρηστικότητα και τη σχέση κόστους-αξίας (value-for-money) του UAV.

Προκειμένου να επιτευχθεί ο ανωτέρω στόχος, απαιτήθηκε η δημιουργία καινοτόμου πρωτοκόλλου επικοινωνίας, βασισμένου στο I2C, και το οποίο επιτρέπει την ταχεία και αξιόπιστη μεταφορά δεδομένων από το ενσωματωμένο σύστημα προς τον υπολογιστή ελέγχου πτήσης (flight controller) του UAV.

Τα ενθαρρυντικά αποτελέσματα της αξιοπιστίας του νέου αυτού πρωτοκόλλου επικοινωνίας, όπως προέκυψαν από ελέγχους σε πραγματικές συνθήκες πτήσης, οδήγησαν στην αξιοποίησή του και σε άλλο τομέα της πτήσης του UAV. Συγκεκριμένα, το ίδιο πρωτόκολλο χρησιμοποιήθηκε προκειμένου να γίνει μεταφορά δεδομένων αισθητήρων εγγύτητας στο UAV στα πλαίσια διαδικασιών αποφυγής συγκρούσεων με αντικείμενα ή/και άλλα UAV.

Η διαθεματική προσέγγιση (embedded systems, flight controller, μηχανολογικές κατασκευές, αεροδυναμική, ασφάλεια πτήσεων, διαχείριση σμήνους αεροσκαφών) που ακολουθήθηκε, οδήγησε στη δημιουργία ενός προϊόντος το οποίο μπορεί να χρησιμοποιηθεί τόσο από τις εταιρείες κατασκευής UAV, όσο και ανεξάρτητα, από τον κάθε πολίτη προκειμένου να κατασκευάσει ή παραμετροποιήσει μόνος του το UAV που επιθυμεί.

Abstract

In the last decade, the transition of the construction industry and exploitation of unmanned flying vehicles (UAVs) from military to purely commercial, industrial and environmental applications have dramatically increased their penetration in society. The continuous technological improvement of UAVs in conjunction with their cost reduction, make them very popular for the simple citizen.

However, most of the commercial low-cost systems available on the market have limited technical capabilities (e.g. image or video acquisition) and new sensor integration is a particularly challenging if not impossible task.

The objective of this diploma thesis is the addition of an embedded system which allows the user to customize their UAV with different types of sensors, depending on the application needs, thereby improving user-friendliness and value for money ratio.

In order to achieve this goal, an innovative communication protocol based on I2C has been implemented. This protocol enables the rapid and reliable transfer of data from the embedded system to the flight controller of the UAV.

The encouraging results of the reliability of this new communication protocol, resulting from tests in real flight conditions, led to its exploitation, in obstacle avoidance applications for UAVs. Specifically, the same protocol was used in order to transfer the proximity sensor data to the UAV in collision processes with objects and/or other UAVs.

The interdisciplinary approach (embedded systems, flight controller, mechanical engineering, aerodynamics, flight safety, aircraft fleet management), which was implemented, resulted in a product which can be used both by the UAV manufacturers, and independently by any citizen to build or customize their own UAV.

Περιεχόμενα

1 Εισαγωγή

1.1 Παρουσίαση προβλήματος

Την τελευταία δεκαετία η ανάπτυξη μη επανδρωμένων ιπτάμενων οχημάτων (UAV - Unmanned Airborne Vehicles) για εμπορικές εφαρμογές γνωρίζει ιδιαίτερη άνθηση με αποτέλεσμα να υπάρχει σημαντικός αριθμός κατασκευαστικών λύσεων και εφαρμογών. Η κατασκευή ενός UAV απαιτεί κατασκευή του σκελετού, ενσωμάτωση των κινητήρων προώθησης, του υλικού απαραίτητου (hardware) και του λογισμικού (software) στο σύστημα. Γενικά, η ανάπτυξη μιας εφαρμογής σε UAV απαιτεί αρκετό χρόνο και πιστοποίηση για τη λειτουργικότητα του συστήματος, αφού πολλές φορές το κόστος απόκτησης των εξαρτημάτων και ο χρόνος αποστολής τους (ειδικά για την Ελλάδα) κάνει την αστοχία και πτώση του UAV απαγορευτική. Οι περισσότεροι κίνδυνοι για το UAV εμφανίζονται στο hardware, το software καθώς και στην μεταξύ τους διασύνδεση. Αυτό, γιατί δεν υπάρχουν προσομοιωτές ώστε να πιστοποιηθεί η ορθή λειτουργία τους εκτός πτήσης, με αποτέλεσμα οτιδήποτε νέο σχεδιάζεται και υλοποιείται, να μπορεί να ελεγχθεί μόνο μέσω της διαδικασίας δοκιμής- σφάλματος (trial-and-error).

Έτσι, ο περισσότερος χρόνος για την υλοποίηση μιας εφαρμογής αναλώνεται στο σχεδιασμό, την κατασκευή, την υλοποίηση και την πιστοποίηση λειτουργίας του hardware και του software. Για παράδειγμα, στο Πανεπιστήμιο του Miami των Η.Π.Α., ενώ αρχικά επιχειρήθηκε η ενσωμάτωση τεσσάρων υπερηχητικών αισθητήρων σε ένα τετρακόπτερο για μια απλοϊκή αποφυγή συγκρούσεων, λόγω περιορισμών στο hardware και του διαθέσιμου χρόνου, ενσωματώθηκε επιτυχώς μόνο ο ένας αισθητήρας [1].

Ένα σημαντικό μέρος της διαδικασίας κατασκευής hardware και software για τα UAVs, είναι και η ενσωμάτωση αισθητήρων τόσο για τον έλεγχο της πτήσης όσο και για τη λήψη μετρήσεων για τις εφαρμογές για τις οποίες προορίζεται το UAV.

Σήμερα, προκειμένου να αλλάξει η χρήση του UAV πέραν της εφαρμογής για την οποία είχε αρχικά σχεδιαστεί, πρέπει (1) να γίνουν αλλαγές στο hardware και το software, και (2) να επανασχεδιαστεί και να πιστοποιηθεί η λειτουργία του πολυκοπτέρου για τη νέα αρχιτεκτονική του συστήματος με τους νέους αισθητήρες.

Το πρόβλημα αυτό έρχεται να επιλύσει η παρούσα διπλωματική εργασία όπως παρουσιάζεται στην επόμενη Ενότητα.

1.2 Σκοπός της διπλωματικής εργασίας

Όπως αναφέρθηκε προηγουμένως, κίνητρο για την εκπόνηση της παρούσας διπλωματικής εργασίας ήταν η αντιμετώπιση προβλημάτων που σχετίζονται με την ευκολία ενσωμάτωσης αισθητήρων στα μη επανδρωμένα ιπτάμενα οχήματα. Συνοπτικά, το πρόβλημα εντοπίζεται:

- στην ανυπαρξία εισόδων/εξόδων στο hardware συγκεκριμένων συστημάτων ελέγχου πτήσης (τα οποία αναλύονται διεξοδικά παρακάτω) ώστε να διευκολυνθεί η ενσωμάτωση των αισθητήρων,
- στον φόρτο (workload) του υπολογιστή ελέγχου πτήσης (flight controller), ο οποίος είναι ήδη πολύ μεγάλος, με αποτέλεσμα να μην είναι εύκολη η ενσωμάτωση "αργών" αισθητήρων σε σχέση με τις ταχύτητες λειτουργίας του συστήματος ελέγχου πτήσης, και
- στην ταχύτητα ενσωμάτωσης και πιστοποίησης της σωστής λειτουργίας του κάθε αισθητήρα ξεχωριστά, ώστε να εντοπίζεται για κάθε επιπλέον στοιχείο η πιθανότητα αστάθειας στο σύστημα.

Από τα παραπάνω συνάγεται ότι κύριο αντικείμενο της διπλωματικής αυτής εργασίας είναι η υλοποίηση ενός συστήματος το οποίο συλλέγει δεδομένα από αισθητήρες πάσης φύσεως και τα μεταβιβάζει στο σύστημα πλοήγησης του αεροσκάφους, με σκοπό εκείνο να τα μεταβιβάσει σε πραγματικό χρόνο ή εκ των υστέρων στο σταθμό βάσης. Δευτερεύον αντικείμενο αποτελεί η ενημέρωση του αεροσκάφους για δεδομένα αισθητήρων τα οποία είναι κρίσιμα για την ασφαλή πλοήγησή του στο χώρο και την αποφυγή συγκρούσεων.

Το υλοποιούμενο σύστημα αυτής της διπλωματικής θα μπορέσει να μειώσει το χρόνο και το κόστος υλοποίησης ορισμένων εφαρμογών, καθώς θα είναι δυνατή η αξιοποίηση αισθητήρων χαμηλού κόστους που μέχρι σήμερα δεν χρησιμοποιούνται στα UAV κυρίως λόγω της χαμηλής ταχύτητας απόκρισής τους.

Για την υλοποίηση της εργασίας τίθενται οι ακόλουθοι στόχοι:

- Στόχος 1 : Προσθήκη αισθητήρων στο UAV,
- Στόχος 2 : Μετάδοση μετρήσεων αισθητήρων στο σταθμό βάσης σε πραγματικό χρόνο,
- Στόχος 3 : Προσθήκη αισθητήρα εγγύτητας στο υλοποιούμενο σύστημα, μετάδοση των τιμών του στο UAV και αξιοποίηση των μετρήσεων του για πτήση με διατήρηση σταθερού υψομέτρου από το έδαφος,

- Στόχος 4 : Σχεδίαση διάταξης αισθητήρων εγγύτητας με την βοήθεια του υλοποιούμενου συστήματος και χρήση τους για αποφυγή συγκρούσεων

Το υλοποιούμενο σύστημα σχεδιάστηκε ώστε να πληροί τα ακόλουθα ποιοτικά και ποσοτικά χαρακτηριστικά:

- χαμηλή (< 200mW) κατανάλωση ενέργειας ώστε η επιβάρυνση του UAV από το επιπλέον σύστημα να είναι ελάχιστη,
- το βάρος πρέπει να είναι το ελάχιστο δυνατό (<100g),
- πρέπει να στηρίζει την επικοινωνία του με το UAV με ευρέως διαδεδομένα πρωτόκολλα ώστε να είναι εφικτή η εύκολη και γρήγορη διασύνδεσή του με διαφορετικά UAVs,
- πρέπει να υποστηρίζει πληθώρα διαφορετικών πρωτοκόλλων διασύνδεσης περιφερειακών, ούτως ώστε η διασύνδεση αισθητήρων να είναι μια απλή και εύκολη διαδικασία,
- οι τάσεις τροφοδοσίας να είναι ίδιες με τα υπόλοιπα στοιχεία του hardware του UAV,
- η ταχύτητα μετάδοσης από το υλοποιούμενο σύστημα στο UAV και από εκεί στο σταθμό βάσης (GCS-ground control station) να είναι της τάξης των μερικών milliseconds ώστε να μπορούν να θεωρηθούν πραγματικού χρόνου,
- η συχνότητα μετάδοσης των δεδομένων από το σύστημα στο UAV να είναι μεγαλύτερη ή τουλάχιστον ίση με αυτήν που ο flight controller έχει ενσωματωμένη (built-in) συμβατότητα

1.2 Οργάνωση Διπλωματικής

Μετά από την σύντομη αυτή εισαγωγή, η διάρθρωση και ανάπτυξη του κειμένου της διπλωματικής εργασίας έχει ως εξής: Στην δεύτερη Ενότητα, γίνεται βιβλιογραφική ανασκόπηση και συνοπτική παρουσίαση των εργαλείων hardware και software που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας. Ακολουθεί, στην τρίτη Ενότητα, η παρουσίαση της μοντελοποίησης και επιλογής της βέλτιστης αρχιτεκτονικής για κάθε επιμέρους τμήμα αλλά και για το ολοκληρωμένο ενσωματωμένο σύστημα και πρωτόκολλο επικοινωνίας. Η υλοποίηση της επιλεχθείσας αρχιτεκτονικής δίδεται στην τέταρτη Ενότητα ακολουθούμενη, στην πέμπτη Ενότητα, από την πιστοποίηση λειτουργίας και το πρωτόκολλο δοκιμών που οδήγησαν στην επιβεβαίωση της αποτελεσματικότητας του αναπτυχθέντος συστήματος σε πραγματικές συνθήκες πτήσης. Τέλος, η εργασία ολοκληρώνεται με παρουσίαση των κυριότερων συμπερασμάτων που εξήχθησαν κατά

την υλοποίησή της καθώς και προτάσεις για αξιοποίηση των αποτελεσμάτων της εργασίας σε μελλοντικές εφαρμογές.

2 Σχετική Έρευνα

2.1 Τμήματα μη επανδρωμένου αεροσκάφους

Παρακάτω παρουσιάζονται επιγραμματικά τα στοιχεία ενός μη επανδρωμένου οχήματος. Αυτή η διπλωματική εργασία πραγματεύεται κυρίως με τα hardware και software τμήματα ενός UAV. Εδώ θα δοθεί μια πλήρης δομή του αεροσκάφους ώστε να αποκτήσει ο αναγνώστης μια ολοκληρωμένη εικόνα του μοντέλου πάνω στο οποίο πραγματοποιήθηκαν τα πειράματα. Συνοπτικά εδώ αναφέρονται τα βασικά δομικά στοιχεία ενός UAV τα οποία είναι: το όχημα, το ωφέλιμο φορτίο και ο σταθμός βάσης.

2.1.1 Όχημα

Ως όχημα ορίζονται τα στοιχεία εκείνα του αεροσκάφους που χωρίς ένα από αυτά δεν μπορεί να πετάξει. Αυτά είναι :

- Σκελετός
- Σύστημα προώθησης
- Σύστημα ελέγχου πτήσης (Flight Controller Board)
- Πηγή τροφοδοσίας

Ο σκελετός αποτελεί τον κορμό του αεροσκάφους. Το σύστημα προώθησης, σε ένα πολυκόπτερο, συνήθως είναι ηλεκτροκινητήρες συνεχούς ρεύματος χωρίς ψήκτρες (brushless DC motors), που σε συνδυασμό με τις έλικες φροντίζουν για την προώθηση του οχήματος. Το flight controller board φροντίζει για τον έλεγχο των rpm (rounds per minute – στροφές ανά λεπτό) των ηλεκτροκινητήρων ώστε το πολυκόπτερο να ισορροπεί και να μπορεί κινείται στον αέρα (η λειτουργία του αναλύεται στο υποκεφάλαιο υπολογιστής ελέγχου πτήσης). Η πηγή τροφοδοσίας είναι συνήθως μπαταρία.

Σημείωση: η παραπάνω περιγραφή είναι για οχήματα όπως αυτά που χρησιμοποιήθηκαν για την παρούσα διπλωματική. Υπάρχουν οχήματα με προωθητικό σύστημα jet και οχήματα που λειτουργούν με βενζινοκινητήρες αλλά αυτά δεν θα απασχολήσουν την παρούσα εργασία.

2.1.2 Ωφέλιμο φορτίο

Ως ωφέλιμο φορτίο ορίζεται οτιδήποτε δεν είναι απαραίτητο για την ορθή πτήση του οχήματος αλλά είναι απαραίτητο για τη φύση της εφαρμογής που θα εκτελέσει. Ενδεικτικά

ακολουθούν παραδείγματα ωφέλιμου φορτίου που χρησιμοποιείται σε μη επανδρωμένα αεροσκάφη:

- Κάμερες
- Υπέρυθρα συστήματα
- Ραντάρ
- Αισθητήρες περιβάλλοντος
- Πακέτα με προϊόντα ή φάρμακα

Το υλοποιούμενο σύστημα της παρούσας διπλωματικής θα μπορούσε να θεωρηθεί ένα είδος τέτοιου ωφέλιμου φορτίου.

Τα UAVs λόγο της ικανότητας τους να μετακινούνται σε απρόσιτες περιοχές, πολλές φορές γίνονται το μέσο μεταφοράς του ωφέλιμου φορτίου καθώς το ωφέλιμο φορτίο είναι εκείνο που ορίζει την χρηστικότητα του αεροσκάφους.

2.1.3 Σταθμός ελέγχου εδάφους ή Βάση εδάφους ή GCS (Ground Control Station)

Ο σταθμός ελέγχου εδάφους είναι ο γενικός παρατηρητής και πολλές φορές χωρίς να αποτελεί τον κανόνα, χειριστής του αεροσκάφους. Θα μπορούσε να ονομαστεί και κέντρο επιχειρήσεων του UAV. Στο GCS είναι δυνατές οι παρακάτω λειτουργίες:

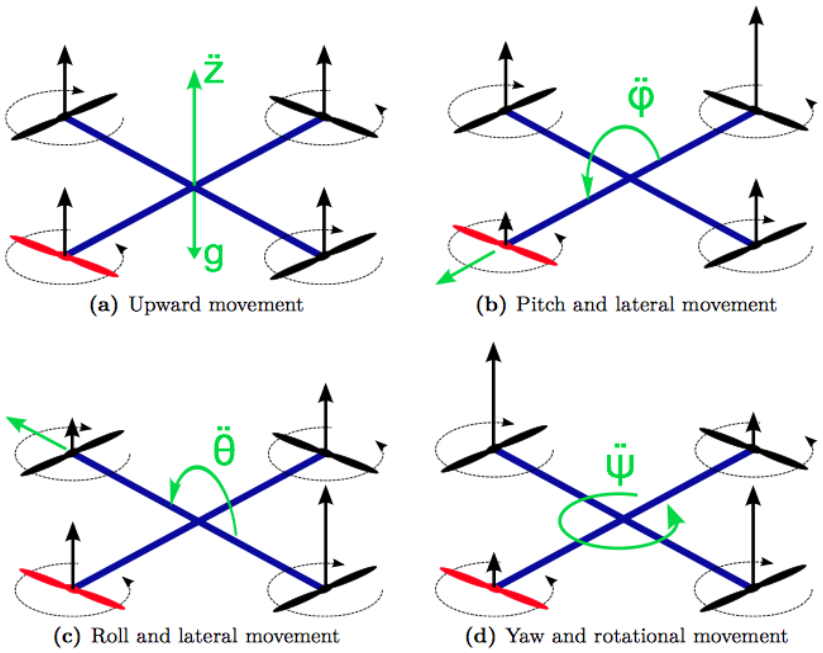
- Παρατήρηση σε πραγματικό χρόνο δεδομένων αισθητήρων και δεδομένων πτήσης
- Πλοήγηση αεροσκάφους
- Γραφική απεικόνιση σε χάρτη 2D/3D του σχεδίου πτήσης :
    - Πριν την πτήση
    - κατά τη διάρκεια σε πραγματικό χρόνο
    - μετά την προσγείωση στο έδαφος διαβάζοντας τα δεδομένα που συλλέχθηκαν
- Επικοινωνία κατά την πτήση και δυνατότητα αλλαγής σχεδίου πτήσης

2.1.4 Υπολογιστής Ελέγχου Πτήσης - Flight Controller Board

Το Flight Controller board είναι ο “πιλότος” του πολυκοπτέρου. Η χρηστικότητά του είναι απλή. Κανένας άνθρωπος δεν είναι ικανός να ελέγξει τις ταχύτητες περιστροφής τριών και πάνω κινητήρων την ίδια στιγμή με αρκετή ακρίβεια ώστε να ισορροπήσει ένα πολυκόπτερο. Εκεί επιστρατεύονται τα flight controller boards για να λύσουν το πρόβλημα. Η λειτουργία τους είναι να ελέγχουν τα rpm των κινητών του πολυκοπτέρου ώστε εκείνο αρχικά να ισορροπήσει και στη

συνέχεια να αλλάζει το rpm του κάθε κινητήρα ώστε το πολυκόπτερο να εκτελεί κινήσεις όπως pitch, yaw και roll τα οποία φαίνονται στο σχήμα 2.1.

Η κόκκινη έλικα δείχνει πάντα το μπροστινό μέρος του πολυκοπτέρου. Τα κάθετα διανύσματα στις έλικες δείχνουν τις προωθητικές δυνάμεις που ασκούν οι ίδιες για να ισορροπήσει το πολυκόπτερο σε συγκεκριμένο ύψος. Το πράσινο διάνυσμα στο κέντρο του πολυκοπτέρου δείχνει την κίνηση που εκτελεί το πολυκόπτερο. Στο σχήμα 2.1(a) το πολυκόπτερο προσπαθεί να κινηθεί προς τα πάνω. Παρατηρείται ότι το διάνυσμα z είναι λίγο μεγαλύτερο από διάνυσμα g. Το διάνυσμα z είναι η συνισταμένη των δυνάμεων που ασκείται στο πολυκόπτερο λόγω των προωθητικών δυνάμεων που ασκούνται από τις έλικες ενώ το g η δύναμη του βάρους. Στο σχήμα 2.1(b) το πολυκόπτερο κάνει κίνηση στον άξονα pitch, δηλαδή κίνηση προς τα εμπρός ή πίσω (στο σχήμα προφανώς κάνει μπροστά). Φαίνεται ότι έχει μειωθεί το διάνυσμα στην μπροστά έλικα ενώ έχει αυξηθεί στην πίσω. Στο 2.1(c) το πολυκόπτερο κάνει roll, δηλαδή κίνηση δεξιά ή αριστερά. Προφανώς οι προωθητικές δυνάμεις των κινητήρων μοιάζουν πολύ με εκείνες του pitch. Τέλος στο σχήμα 2.1(d) το πολυκόπτερο κάνει yaw, δηλαδή μειώνει την άνωση σε δύο έλικες και την αυξάνει σε άλλες δύο. Εδώ αξίζει τα σημειωθεί ότι το πολυκόπτερο κινείται προς τη φορά εκείνη που οι έλικες έχουν μειώσει τις προωθητικές δυνάμεις τους και όχι προς εκείνη που την έχουν αυξήσει.



(a) Upward movement (b) Pitch and lateral movement

(c) Roll and lateral movement (d) Yaw and rotational movement

Σχήμα 2.1 Δυνάμεις προώθησης κινητήρων πολυκοπτέρου ώστε εκείνο να εκτελεί κινήσεις (a) upward, (b) pitch, (c) roll, και (d) yaw

Η πλειοψηφία των flight controller boards αποτελούνται από έναν μικροεπεξεργαστή (microprocessor) και μια Μονάδα Μέτρησης Αδράνειας (Inertia Movement Unit - IMU). Η IMU αποτελείται από :

- Γυροσκόπιο τριών αξόνων
- Επιταχυνσιόμετρο τριών αξόνων
- Μαγνητόμετρο (πυξίδα)

Στα παραπάνω, συνήθως, προστίθεται και ένα βαρόμετρο για τα UAVs. Οι αισθητήρες της IMU βοηθούν τον μικροεπεξεργαστή να βελτιώσει τους υπολογισμούς του. Το γυροσκόπιο και το επιταχυνσιόμετρο δίνουν μετρήσεις για την περιστροφή του πολυκοπτέρου σε κάθε άξονα και το πολυκόπτερο μπορεί να ελέγχει την ισορροπία του. Το βαρόμετρο βοηθάει στην διατήρηση του ύψους, καθώς ο microcontroller μπορεί με απλές σχέσεις να μετατρέψει τη μέτρηση της ατμοσφαιρικής πίεσης σε υψόμετρο. Η πυξίδα είναι χρήσιμη στο yaw (το οποίο φαίνεται στο Σχήμα 2.1 d) καθώς η μέτρησή της, προσφέρει χρήσιμη πληροφορία για να παραμένει το πολυκόπτερο σε σταθερή ταχύτητα ή να τη μεταβάλλει.

Αυτή ήταν η πιο βασική λειτουργία του flight controller. Οι ανάγκες όμως για εξέλιξη δεν σταματούν ποτέ με αποτέλεσμα να δημιουργούνται flight controllers που είναι ικανοί για πολύ περισσότερα από τα παραπάνω. Αυτοί που χρησιμοποιήθηκαν σε αυτή τη διπλωματική είναι υπεύθυνοι για την τήρηση του σχεδίου πτήσης αυτόνομα και παράλληλα φέρουν ευθύνη για την ποιότητα της. Ειδικότερα, είναι εκείνο το στοιχείο του πολυκοπτέρου που απογειώνει και προσγειώνει αυτόνομα το πολυκόπτερο, αποθηκεύει και εκτελεί το σχέδιο πτήσης, ενώ συλλέγει και μεταδίδει δεδομένα προς τον σταθμό βάσης.

Προφανώς πολλά από τα παραπάνω δεν μπορούν να γίνουν πράξη χωρίς την προσθήκη και επιπλέον αισθητήρων όπως GPS.

Ο flight controller αυτής της διπλωματικής υποστηρίζει διαφορετικές καταστάσεις λειτουργίας (modes) τα οποία μπορεί να αλλάξει ο χειριστής από την τηλεκατεύθυνση. Μερικά από αυτά τα modes είναι GPS lock, altitude hold και manual controlling. Τα modes εξηγούνται διεξοδικά παρακάτω όπου αναλύεται το Megapirate NG firmware.

2.2 Έρευνα στη Βιβλιογραφία

Μετά από αναζήτηση στη βιβλιογραφία παρατηρήθηκε η έλλειψη μιας καθολικής λύσης γενικού σκοπού για την ενσωμάτωση αισθητήρων σε ένα πολυκόπτερο. Υπάρχουν πολλές

εργασίες πάνω σε πολυκόπτερα, οι οποίες εκμεταλλεύονται τους ήδη ενσωματωμένους αισθητήρες για συλλογή μετρήσεων ή ενσωματώνουν νέους αισθητήρες πάνω σε αυτά για την υλοποίηση μιας συγκεκριμένης εφαρμογής. Σε αυτή τη διπλωματική αυτό που παρουσιάζεται είναι μια έτοιμη διεπαφή ώστε οι μετρήσεις των αισθητήρων να φτάνουν σε πραγματικό χρόνο στον flight controller και στο σταθμό βάσης. Δυστυχώς βιβλιογραφικές αναφορές σε όμοιες διεπαφές με αυτή που θα παρουσιαστεί παρακάτω δεν βρέθηκαν. Παρόλα αυτά υπάρχουν πάρα πολλές δημοσιεύσεις που αφορούν τηλεμετρία με πολυκόπτερα. Μερικές από αυτές τις δημοσιεύσεις παρουσιάζονται παρακάτω, με σκοπό αρχικά να αποδειχθεί ότι τα μη επανδρωμένα οχήματα χρησιμοποιούνται κατά κόρων για τηλεμετρικές εφαρμογές όπου η προσέγγιση ανθρώπων για τη συλλογή των μετρήσεων κρίνεται επικίνδυνη και στη συνέχεια παρουσιάζονται κάποιες δημοσιεύσεις από τις οποίες εξάγονται χρήσιμα συμπεράσματα για το σχεδιασμό του συστήματος της παρούσας διπλωματικής.

2.2.1 Τηλεμετρία

Τα μη επανδρωμένα οχήματα είναι ικανά να εκτελέσουν πτήσεις σε δύσβατα σημεία. Τέτοιο σημείο αποτελούν και οι ηφαιστειακοί κρατήρες που υποστηρίζουν ότι εξέτασαν οι M.Bartholomai και Patrick Neumann [2], όπου ενσωματώνοντας σε ένα εμπορικό τετρακόπτερο, έναν καταλυτικό αισθητήρα με τον οποίο είχαν την δυνατότητα να μετράνε O_2, CO, H_2S, NH_3, CO_2, SO_2, PH_3, HCN, NO_2, Cl_2 , έναν αισθητήρα υγρασίας και έναν θερμοκρασίας, κατάφεραν να συλλέξουν μετρήσεις για τις ποσότητες SO_2 που εκκρίνονταν από τους κρατήρες των ηφαιστείων.

Το 2015 παρουσιάστηκε μία εφαρμογή (P.Croizé et al. [3]), όπου γίνεται χρήση ενός εμπορικού πολυκοπτέρου με σκοπό την εκτέλεση συγκεκριμένου σχεδίου πτήσης. Η μεταγωγή του οχήματος από το ένα σημείο του σχεδίου πτήσης στο άλλο συνοδευόταν με λήψη μετρήσεων H_2S (υδρόθειο) γύρω από το σημείο ενδιαφέροντος. Η συλλογή των δεδομένων από τον αισθητήρα H_2S έγινε με χρήση του ενσωματωμένου συστήματος Arduino. Στο Arduino επίσης ενσωματώθηκε και ένα GPS με σκοπό την πλοήγηση του πολυκοπτέρου, η οποία έγινε με την κατασκευή συγκεκριμένου πρωτοκόλλου για την διεπαφή Arduino-Πολυκοπτέρου. Μετά την πτήση, τα δεδομένα που συλλέχθηκαν, εισήχθησαν σε ειδικά σχεδιασμένο γραφικό περιβάλλον, όπου από τις μετρήσεις του GPS και του αισθητήρα H_2S, έγινε εντοπισμός των πηγών εκπομπής των συγκεκριμένων αερίων. Εδώ η προσοχή εστιάζεται σε δύο σημεία. Πρώτον, το μη επανδρωμένο όχημα αυτής της εφαρμογής χρησιμοποιήθηκε για τη συλλογή μετρήσεων αερίων

τα οποία καταδεικνύουν πιθανή εκκίνηση πυρκαγιάς, πράγμα που θα έκανε την λήψη μετρήσεων από άνθρωπο ιδιαίτερα επικίνδυνη. Δεύτερον, όταν οι συνθήκες της εκάστοτε εφαρμογής δεν ευνοούν την απευθείας σύνδεση αισθητήρων σε ένα μη επανδρωμένο όχημα, τότε η χρήση ενός δεύτερου συστήματος για τη συλλογή των μετρήσεων, ενδείκνυται.

Στο ίδιο μοτίβο με την προηγούμενη δημοσίευση, παρουσιάζεται και η παρούσα των M.El-Diwiny και A.H. El-Sayed [4] όπου η κατασκευή όλου του hardware και software του μη επανδρωμένου αεροσκάφους υλοποιήθηκε αποκλειστικά σε Arduino, πράγμα που αποδεικνύει ότι η μικρή υπολογιστική ισχύς που προσφέρει ένας μικροελεγκτής, όπως αυτός του Arduino, είναι αρκετός για τον έλεγχο ενός UAV. Εδώ το Arduino παίρνει τη θέση του flight controller σε ένα μη επανδρωμένο όχημα, πράγμα που θα γίνει και στην παρούσα διπλωματική.

2.2.2 Αποφυγή Συγκρούσεων

Παρόλο που τα UAVs μπορούν να προηγηθούν, σε εξωτερικούς χώρους, με τη χρήση GPS, το ίδιο δεν μπορεί να συμβεί σε εσωτερικούς χώρους (Nonami 2010 [5]). Κάποιες προσεγγίσεις, για την επίλυση συγκρούσεων, ενσωματώνουν σε πολυκόπτερα συστήματα με κάμερες, τα οποία με τη σειρά τους εισάγουν απαιτήσεις υπολογιστικής ισχύος και επιπλέον βάρος στο σύστημα πτήσης (N. Gageik 2012 [6]). Άλλες προσεγγίσεις, όπως αυτές που παρουσιάζονται στις παρακάτω παραγράφους, ενσωματώνουν αισθητήρες εγγύτητας στο σύστημα.

Οι αισθητήρες εγγύτητας μπορεί να είναι υπερηχητικοί ή υπέρυθροι. Μια εφαρμογή με δώδεκα υπέρυθρους αισθητήρες υλοποιήθηκε με σκοπό την αποφυγή συγκρούσεων (N. Gageik 2012 [6]). Ο flight controller κατασκευάστηκε από ένα Atmel Board με ATmega2560 και ένα IMU (Inertia Measurement Unit). Οι υπερηχητικοί αισθητήρες ενσωματώθηκαν στον μικροεπεξεργαστή μέσω του Atmel Board το οποίο προσέφερε πληθώρα εισόδων/εξόδων, κάτι που στους flight controllers αυτής της διπλωματικής δεν προσφέρεται. Η συγκεκριμένη διάταξη θα υλοποιηθεί διαφορετικά στην παρούσα διπλωματική. Στην παρούσα δημοσίευση εξηγείται εκτενώς το πρωτόκολλο αποφυγής συγκρούσεων που χρησιμοποιήθηκε, κάτι που θα φανεί ιδιαίτερα χρήσιμο και στην παρούσα υλοποίηση πρωτοκόλλου αποφυγής συγκρούσεων.

Παρόμοια εφαρμογή με την παραπάνω είναι η αποφυγή συγκρούσεων με χρήση τεσσάρων αισθητήρων απόστασης υπεριώδους ακτινοβολίας και ένας υπερήχου(J. F. Roberts et al. 2007 [7]). Σε αυτή την εφαρμογή, όπως και στην προηγούμενη, κατασκευάστηκε όλο το hardware του πολυκοπτέρου από την αρχή και ενσωματώθηκαν στη σχεδίαση οι αισθητήρες απόστασης. Ο

αισθητήρας υπερήχων έχει ενσωματωθεί κάτω από το πολυκόπτερο με σκοπό την διατήρηση σταθερού υψομέτρου κατά την πτήση. Οι αισθητήρες απόστασης ενσωματώνονται σε αναλογικές εισόδους του μικροελεγκτή της Atmel που χρησιμοποιήθηκε. Όλα τα πειράματα έγιναν σε εσωτερικούς χώρους. Από την παραπάνω δημοσίευση φαίνεται ότι οι αισθητήρες εγγύτητας μπορεί να είναι και infrared εκτός από υπερηχητικούς για αποφυγή συγκρούσεων. Παρατηρείται ότι για τη μέτρηση του υψομέτρου, όπου η απόσταση προς μέτρηση είναι μεγαλύτερη από την απόσταση για την αποφυγή συγκρούσεων, χρησιμοποιήθηκε υπερηχητικός αισθητήρας.

2.3 Επιλογή Διεπαφής επικοινωνίας μεταξύ υλοποιούμενου συστήματος-UAV

Το πρωτοκόλλου διεπαφής μεταξύ του UAV και του υλοποιούμενου συστήματος είναι το I2C. Η επιλογή έγινε μετά από μελέτη των διαθέσιμων πρωτοκόλλων που υποστηρίζονται από τα flight controller boards του εργαστηρίου (I2C, SPI, UART) και αξιολόγηση των πλεονεκτημάτων και μειονεκτημάτων του κάθε ενός. Το μεγαλύτερο πλεονέκτημα του I2C είναι ότι η σύνδεση ενός συστήματος σε αυτό δεν επιφέρει απώλεια εισόδων/εξόδων για τους flight controllers.

2.3.1 Η Επιλογή του I2C

Από τους flight controllers που χρησιμοποιούνται σε αυτή τη διπλωματική παρατηρείται ότι από τις 4 σειριακές θύρες οι 2 είναι ήδη σε χρήση για την τηλεμετρία (πομποδέκτης αποστολής μηνυμάτων MAVlink - παρουσιάζεται παρακάτω) και το GPS. Οπότε δεν είναι η καλύτερη επιλογή αφού η χρήση μιας ακόμα θύρας θα μειώσει επιπλέον τις εισόδους/εξόδους του flight controller.

Το SPI φαντάζει η καλύτερη επιλογή αφού δεν έχει διασυνδεθεί τίποτα με αυτό και οι ταχύτητες μετάδοσης του είναι αρκετά μεγαλύτερες από εκείνες του I2C. Το μειονέκτημα είναι ότι δεν έχει την ίδια επεκτασιμότητα με το I2C με αποτέλεσμα αν χρειαστεί να συνδεθούν πολλές συσκευές σε αυτό θα υπάρξει πρόβλημα λόγο της έλλειψης εισόδων/εξόδων μερικών από τους flight controllers. Το μεγαλύτερο πρόβλημα του SPI είναι ότι δεν έχουν όλοι οι flight controllers dataflash μνήμη. Δεδομένου ότι το πρωτότυπο του υλοποιούμενου συστήματος θα κατασκευαστεί στο Crius AIO v1.0, που δεν έχει dataflash μνήμη, μπορεί στους flight controllers με dataflash μνήμη να παρουσιαστούν προβλήματα throughput, αφού λόγω της επικοινωνίας μνήμης με μικροεπεξεργαστή, θα μειωθεί το εύρος του διαθέσιμου χρόνου επικοινωνίας.

Το πλεονέκτημα του Crius AIO v1.0 είναι ότι όλοι οι αισθητήρες του είναι I2C με αποτέλεσμα να χρησιμοποιεί το μέγιστο data rate που θα μπορούσαν να χρησιμοποιήσουν οι flight

controllers για την επικοινωνία τους με την IMU, οπότε αν το υλοποιούμενο σύστημα συνδεθεί σε άλλον εκτός του Crius AIO v1.0, flight controller, είναι βέβαιο ότι δεν θα συναντηθούν προβλήματα throughput.

Αυτά μας οδηγούν στην επιλογή του I2C. Αυτομάτως εξάγεται το συμπέρασμα ότι το υλοποιούμενο σύστημα αυτής της διπλωματικής θα είναι ένας I2C slave αφού οι flight controllers έχουν ήδη αισθητήρες με τους οποίους επικοινωνούν σαν I2C masters.

2.3.2 Micro Air Vehicle Link (MAVlink)

Το MAVlink είναι ένα lightweight πρωτόκολλο επικοινωνίας που χρησιμοποιείται σε μη επανδρωμένα οχήματα. Σχεδιάστηκε ως δίαυλος επικοινωνίας αποστολής μετρήσεων από μη επανδρωμένα οχήματα στο σταθμό βάσης αλλά έχει επεκταθεί και σε αποστολή εντολών για τον τηλεχειρισμό των οχημάτων. Το MAVlink πακετοποιεί δομές C και τις στέλνει μέσα από σειριακά κανάλια στο σταθμό βάσης. Η λειτουργία του έχει πιστοποιηθεί σε πολλές γνωστές πλατφόρμες όπως PX4, PIXHAWK, Ardupilot και Parrot AR Drone.

Το MAVlink εκδόθηκε το 2009 από τον Lorenz Meier με GNU Lesser General Public License (LGPL) license.

Επειδή το firmware που χρησιμοποιήθηκε σε αυτή τη διπλωματική είναι μια παραλλαγή του Ardupilot για οικονομικότερα ενσωματωμένα από τον Apm 2.6, είναι ήδη συμβατό με το MAVlink. Επειδή το MAVlink είναι ήδη κατασκευασμένο για να στέλνει δεδομένα τηλεμετρίας στο σταθμό βάσης, θα ήταν σπατάλη πόρων να χρησιμοποιεί νέο πρωτόκολλο ή νέο τρόπο διασύνδεσης του flight controller και του σταθμού βάσης προκειμένου να σταλούν τα δεδομένα στο σταθμό βάσης.

2.4 Προϊόντα που Χρησιμοποιήθηκαν

2.4.1 Megapirate NG Flight Controller Boards (MPNG)

Όλα τα flight controller boards που είναι συμβατά με το MPNG διέπονται από ένα κοινό χαρακτηριστικό. Είναι βασισμένα στον μικροελεγκτή ATmega2560 [8] της Atmel, δηλαδή έναν 8-bit μικροελεγκτή με τέσσερις σειριακές θύρες, I2C port, SPI port, 256KB flash memory, 4KB EEPROM και 8KB SRAM.

Τα υπόλοιπα χαρακτηριστικά ποικίλουν ανά board αλλά τα περισσότερα υποστηρίζουν 8 κανάλια τηλεχειρισμού και 8 εξόδους για την οδήγηση των ηλεκτροκινητήρων ώστε να

υποστηρίζουν μέχρι και οκτακόπτερο. Όλα έχουν IMU όπως αυτή περιγράφεται στο κεφάλαιο 2.1.4 Flight Controller Board και εισόδους/εξόδους ώστε να υποστηρίζονται οι λειτουργίες pitch, roll και trigger για την κίνηση της κάμερας και τη λήψη φωτογραφιών. Λίγα εκ' των boards υποστηρίζουν επιπλέον I/O για επεκτασιμότητα των ικανοτήτων του board με άλλα συστήματα. Μόνο το Crius V2 [9] έχει ενσωματωμένη dataflash για την αποθήκευση των δεδομένων πτήσης.

2.4.2 Arduino Pro Mini

Το Arduino Pro Mini [10] είναι το πιο ελαφρύ (2g) και μικρό Arduino που υπάρχει στην αγορά. Συγκρίνοντάς το και με άλλα γνωστά ενσωματωμένα που κυκλοφορούν εξάγεται το συμπέρασμα ότι έχει και την πιο μικρή κατανάλωση ενέργειας. Αυτός είναι και ο λόγος επιλογής του. Έχει δύο εκδόσεις με διαφορετικούς microcontrollers τους ATmega168 και ATmega328P. Δεν υπάρχει σχεδόν καμία διαφορά ανάμεσα στους 2 microcontrollers εκτός των μνημών όπου ο ATmega328P έχει διπλάσια SRAM(2Kbytes), διπλάσια EEPROM(1Kbyte) και διπλάσια Flash Program Memory (32Kbytes). Επειδή η τιμή των 2 ήταν ίδια επιλέχθηκε η έκδοση με τον ATmega328P.

Το Arduino Pro Mini είναι ένα board που αποτελείται από έναν ATmega328P, ένα πιεστικό διακόπτη (push button) για reset και μια μονάδα ρύθμισης τάσης (voltage regulator) ώστε τάσεις μεγαλύτερες των 5V να μπορούν να εισαχθούν στο Arduino χωρίς να ελλοχεύει κίνδυνος καταστροφής του board. Έχει 14 ψηφιακές εισόδους/εξόδους και 8 αναλογικές εισόδους. Υποστηρίζει μία UART TTL σειριακή θύρα μέσω των pins 0(RX) και 1(TX), εξωτερικές διακοπές (external interrupts) στα pins 2 και 3, PWM (pulse width modulation) εξόδους στα 3,5,6,9,10,11, SPI στα 10(SS), 11(MOSI), 12(MISO) και 13(SCK) και I2C στα A4(SDA), A5(SCL) όπου A4 και A5 τα αναλογικά pins 4 και 5.

2.4.3 3DR Radio Set V2

Το 3DR Radio Set [11] είναι ένα σετ κεραιών που αποτελούν ένα εύκολο τρόπο να στηθεί ένας δίαυλος τηλεμετρίας μεταξύ του flight controller και του σταθμού βάσης. Κάθε κεραία είναι ελαφριά, μικρή και σχετικά οικονομική, αν συνυπολογιστεί ότι μπορεί να πετύχει μεταδόσεις μέχρι 300m χωρίς επιπλέον κεραία. Μεταδίδει στα 433 MHz για την Ευρώπη και στα 915 MHz για την Αμερική. Επικοινωνεί με το controller board μέσω σειριακής θύρας και υποστηρίζει ρυθμούς μετάδοσης ως και 250kbps.

2.4.4 Sensors

DHT11

Ο DHT11 [12] είναι ένας χαμηλού κόστους, ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας. Χρησιμοποιεί χωρητικού τύπου αισθητήρα για τη μέτρηση της υγρασίας και θερμίστορ για τη μέτρηση της θερμοκρασίας. Εξάγει ψηφιακό σήμα για τη μετάδοση των δεδομένων. Το μειονέκτημα του συγκεκριμένου αισθητήρα, είναι η αργή του αντίδραση σε απότομες μεταβολές περιβάλλοντος.

BMP085

Ο BMP085 [13] είναι ένας, χαμηλού κόστους, αισθητήρας μέτρησης ατμοσφαιρικής πίεσης και θερμοκρασίας. Επειδή η ατμοσφαιρική πίεση μεταβάλλεται με μεταβολή του υψομέτρου είναι πολύ εύκολο να χρησιμοποιηθεί και για τη μέτρηση υψομέτρου, προφανώς σε μεγάλα υψόμετρα (από 10m και πάνω) όπου δεν απαιτείται μεγάλη ακρίβεια μέτρησης. Η επικοινωνία του αισθητήρα για την μετάδοση των δεδομένων γίνεται μέσω I2C πρωτοκόλλου.

HC-SR04

Ο συγκεκριμένος αισθητήρας μπορεί να χρησιμοποιηθεί για την μέτρηση αποστάσεων. Μπορεί να μετρήσει από δύο εκατοστά ως και τέσσερα μέτρα [14] με ακρίβεια εκατοστού για αντικείμενα επιφάνειας 50cm^2. Όταν ο αισθητήρας ελέγχθηκε για μεγαλύτερα ή μικρότερα αντικείμενα, η μέγιστη απόσταση μεγάλωσε πολύ περισσότερο από 4m. Για μέτρηση υψομέτρου, όπου το πάτωμα είναι ιδιαίτερα μεγάλο αντικείμενο, επετεύχθη μέτρηση μέχρι 10m με απόκλιση 5cm. Ο αισθητήρας έχει μία είσοδο (trigger) και μία έξοδο (echo). Στην είσοδο απαιτούνται παλμοί μήκους τουλάχιστον 10μs και στη έξοδο παρατηρούνται οι αποκρίσεις των ίδιων παλμών, μετά από εκπομπή τους από τον πομπό, ανάκλασή τους σε κάποιο αντικείμενο και ανίχνευση από το δέκτη. Από το χρόνο μεταγωγής του παλμού στον αέρα, υπολογίζεται η απόσταση μεταξύ του αισθητήρα και του αντικειμένου.

2.5 Open Source Εργαλεία

2.5.1 Megapirate NG

Το Megapirate NG(MPNG) είναι μια απομίμηση του firmware APM Autopilot Suite. Ο στόχος δημιουργίας του είναι η υποστήριξη οικονομικότερων board από τα APMs. Με αυτό τον προσανατολισμό υποστηρίζει τα παρακάτω flight controller boards:

- RCTimer Crius V2

- Crius V1
- HobbyKing AllInOne Pro boards
- HobbyKing MultiWii Pro board
- Black Vortex
- MultiWii PRO Ez3.0 Blacked MAG Editon Flight Controller
- PARIS v5 Mega iOSD
- APM 2.5, APM 2.6 κτλ

Το MPNG όπως και κάθε firmware πολυκοπτέρου υποστηρίζει modes [15] λειτουργίας τα οποία κάνουν το χειρισμό του πολυκοπτέρου ευκολότερο για τον χρήστη. Έτσι το MPNG υποστηρίζει τις ακόλουθες λειτουργίες (λεπτομέρειες δίνονται στο παράρτημα):

- Stabilize
- Alt Hold
- Loiter
- Land
- RTL (Return to Launch)
- Auto
- Acro
- Guided
- Circle
- Follow me

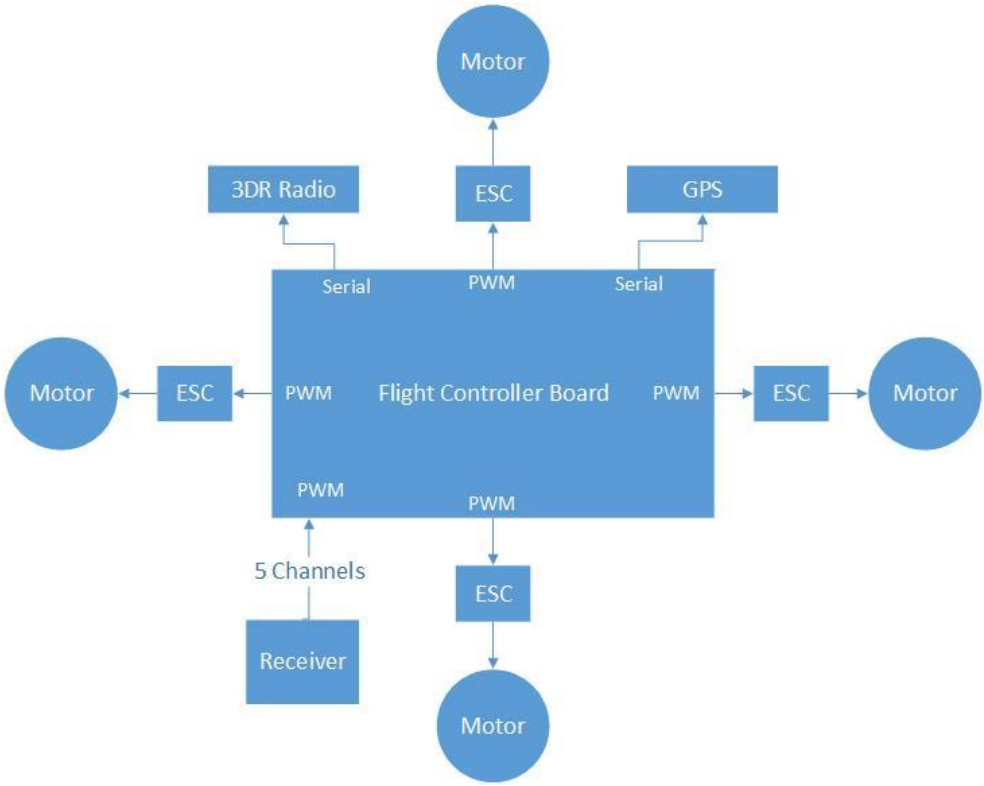
3 Μοντελοποίηση και Αρχιτεκτονική

Το Arduino(σύστημα μετρήσεων) που θα χρησιμοποιηθεί στα πλαίσια αυτής της διπλωματικής πρέπει να συλλέγει δεδομένα από αισθητήρες και με κάποιο τρόπο αυτά να φτάνουν στο flight controller και το σταθμό βάσης. Όπως αναφέρεται στο Πρώτο Κεφάλαιο, που έχουν οριστεί οι στόχοι του υλοποιούμενου συστήματος, αυτό πρέπει να μπορεί να γίνει σε πραγματικό χρόνο κατά τη διάρκεια της πτήσης.

3.1 Αρχική και τελική μορφή συνολικού συστήματος

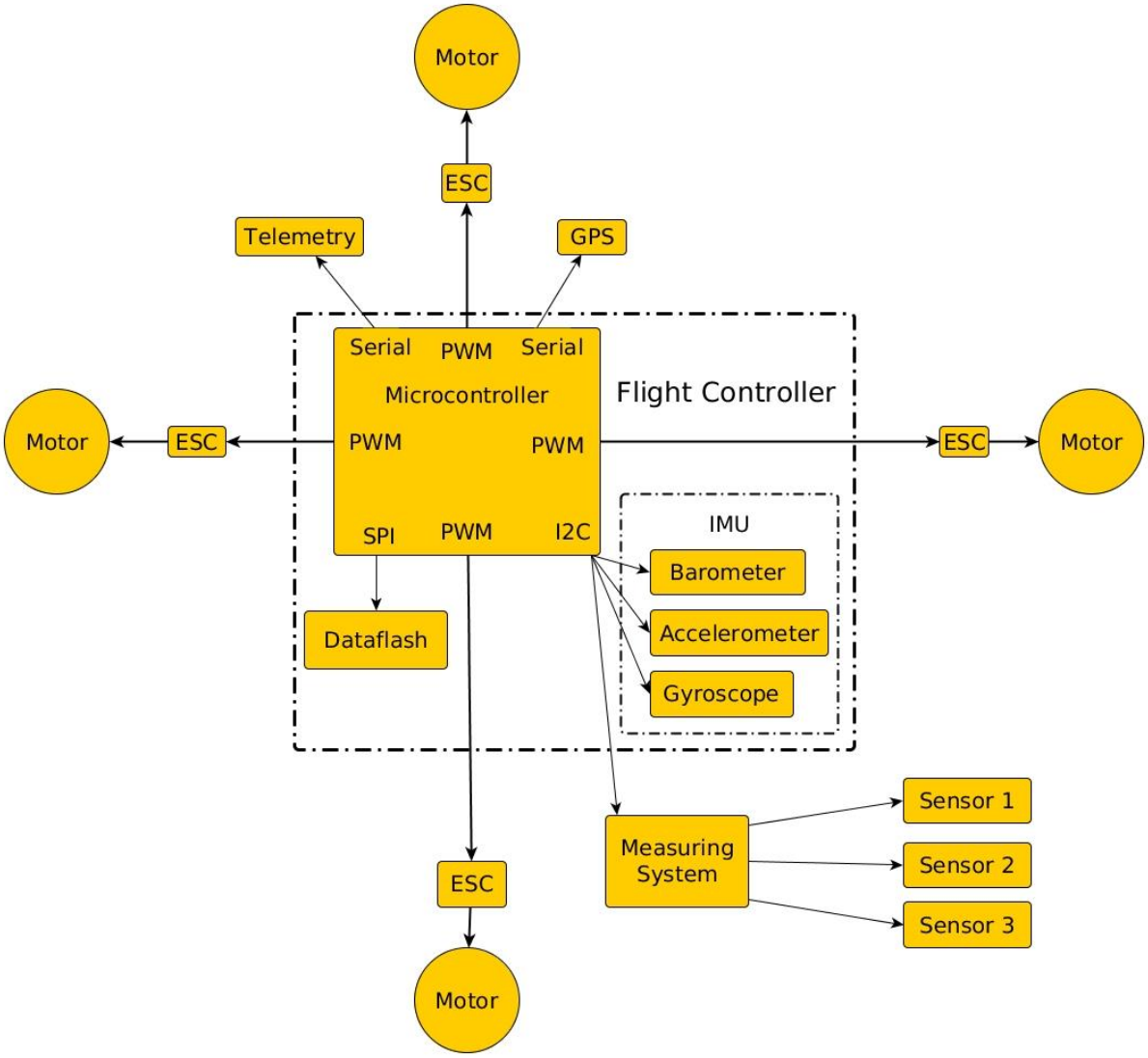
3.1.1 Αρχική μορφή συστήματος

Παρακάτω φαίνεται η αρχική μορφή του πολυκοπτέρου πριν γίνει οποιαδήποτε επέμβαση στα πλαίσια αυτής της διπλωματικής. Παρατηρείται ότι το πολυκόπτερο αποτελείται από τα βασικά δομικά στοιχεία που αναφέρθηκαν στα τμήματα του επανδρωμένου οχήματος δηλαδή τον flight controller, τα μοτέρ με τα speed controller τους και κάποια επιπλέον στοιχεία όπως 3DR Radio (τηλεμετρία) για μετάδοση των τιμών τηλεμετρίας, το δέκτη του χειριστηρίου και ένα GPS που μπορεί να προσφέρει αυτονομία σε συγκεκριμένα modes.



Σχήμα 3.1 Αρχιτεκτονική αρχικού συστήματος

3.1.2 Τελική μορφή πολυκοπτέρου



Σχήμα 3.2 Αρχιτεκτονική πολυκοπτέρου με το νέο σύστημα που κατασκευάζεται σε αυτή τη διπλωματική ενσωματωμένο

Στο τελικό σύστημα φαίνεται αναλυτικά η δομή μιας IMU εσωτερικά και τα επιμέρους στοιχεία της. Σε μεγαλύτερο επίπεδο φαίνεται η δομή ενός από τους flight controllers που χρησιμοποιήθηκαν στα πλαίσια αυτής της διπλωματικής. Στην τελική μορφή του πολυκοπτέρου φαίνεται πώς ενσωματώνεται το σύστημα μετρήσεων που κατασκευάζεται με το Arduino στο αρχικό σύστημα του πολυκοπτέρου.

Λόγο της φύσεως του πολυκοπτέρου η μετάδοση σε πραγματικό χρόνο απαιτεί προφανώς ασύρματη μετάδοση, δηλαδή κεραία στο σύστημα. Παρατηρώντας το παραπάνω block διάγραμμα

του πολυκοπτέρου του εργαστηρίου, παρατηρείται ότι μια κεραία υπάρχει ήδη εκεί και χρησιμοποιείται για την τηλεμετρία του πολυκοπτέρου, δηλαδή τη μετάδοση μηνυμάτων από το πολυκόπτερο στο σταθμό βάσης και αντίστροφα. Θα ήταν λοιπόν σπατάλη ενέργειας, αλλά και πόρων, να χρησιμοποιηθεί δεύτερη κεραία για μετάδοση δεδομένων ίδιου τύπου, αφού και από την κεραία τηλεμετρίας μεταδίδονται δεδομένα για την κατάσταση του UAV και των διαθέσιμων αισθητήρων. Έτσι για την μετάδοση των δεδομένων θα χρησιμοποιηθεί η ενσωματωμένη, στο σύστημα, κεραία τηλεμετρίας.

Όλα τα flight controllers, που χρησιμοποιούνται για αυτή τη διπλωματική, είναι σχεδιασμένα με τέτοιο τρόπο ώστε να είναι συμβατά στην δημοφιλή πλατφόρμα Arduino. Έτσι ο κώδικας του MPNG είναι δομημένος με τις γνωστές ρουτίνες του Arduino, setup() και loop() (αρχείο ArduCopter.pde). Οι σχεδιαστές του MPNG έχουν αποφασίσει ότι η πιο συχνή διεργασία για την πτήση του πολυκοπτέρου πρέπει να εκτελείται με συχνότητα 100Hz, με αποτέλεσμα να έχουν σχεδιάσει έτσι τον κώδικα, ούτως ώστε δύο διαδοχικές εκτελέσεις της loop() να απέχουν χρονικά 10 ms. Για να λειτουργήσει αυτό αποτελεσματικά, έχουν υλοποιήσει διεργασίες που δηλώνονται στο πρόγραμμα μαζί με τη συχνότητα και τον μέγιστο χρόνο εκτέλεσης τους. Έτσι ένας χρονοπρογραμματιστής (scheduler), ο οποίος καλείται στη loop(), ελέγχει τον εναπομείναντα χρόνο μέχρι την επόμενη εκτέλεσή της, και "καλεί" τις διεργασίες που πρέπει να "τρέξουν" σε αυτήν την επανάληψη. Έτσι διεργασίες που πρέπει να εκτελούνται με συχνότητα 50 Hz "καλούνται" ανά δύο εκτελέσεις της loop(), αυτές με συχνότητα 20 Hz ανά πέντε, κοκ. Προφανώς όλες οι διεργασίες, με διαφορετικές συχνότητες εκτέλεσης, που θα κληθούν στο ίδιο χρονικό "παράθυρο" των 10 ms δεν πρέπει να το ξεπεράσουν, όλες μαζί, καθώς τα όρια για τον επόμενο κύκλο είναι αυστηρά καθορισμένα. Διεργασίες, που καταχρώνται στο χρόνο που έχει δηλωθεί για εκείνες, μπορεί να αναγκαστούν να σταματήσουν την εκτέλεσή τους από το scheduler.

Σύμφωνα με τα παραπάνω, και εφόσον δεν υπάρχει χρόνος για αναδιαμόρφωση όλου του κώδικα του MPNG ώστε να επιτευχθούν συχνότερες κλήσεις της loop() η μεγαλύτερη συχνότητα δειγματοληψίας μεταξύ συστήματος και flight controller δεν θα ξεπεράσει τα 100Hz, αφού η συγγραφή μιας νέας διεργασίας στο σύστημα θα έχει ως μέγιστη συχνότητα δειγματοληψίας για κάθε αισθητήρα την προαναφερθείσα. Εξ' άλλου δεν υπάρχει λόγος για αναδιαμόρφωση του κώδικα, μιας και μία μέτρηση για κάθε αισθητήρα ανά 10ms είναι υπεραρκετή. Επίσης, η διάρκεια της νέας διεργασίας πρέπει να είναι η μικρότερη δυνατή, ώστε να μην γίνει σε καμία περίπτωση υπέρβαση του χρόνου των 10ms και δεν εκτελεστούν διεργασίες καίριας σημασίας για το

πολυκόπτερο, το οποίο θα οδηγούσε σε απρόβλεπτα αποτελέσματα. Η νέα διεργασία πρέπει να συλλέγει τα δεδομένα από τους αισθητήρες του υλοποιούμενου συστήματος και να τα μεταδίδει μέσω τηλεμετρίας.

Ελέγχοντας τον τρόπο μετάδοσης ενός μηνύματος μέσω τηλεμετρίας (αρχείο GCS_Mavlink.pde), παρατηρείται ότι για να επιτευχθεί μια σωστή μετάδοση πακέτου, απαιτούνται 500 μs, πράγμα που δεν μπορεί να παραληφθεί στην παρούσα σχεδίαση και ήδη γίνεται αντιληπτό, ότι αυτό μπορεί να αλλάξει δραματικά το ρυθμό μετάδοσης δεδομένων των αισθητήρων σε πραγματικό χρόνο, καθώς μια επιπλέον μετάδοση πακέτου θα αυξήσει πολύ το φόρτο εργασίας (workload) του flight controller, ίσως σε απαγορευτικό σημείο. Αυτό πρέπει να ελεγχθεί πειραματικά στην υλοποίηση και να βρεθεί ποιος είναι ο μέγιστος ρυθμός μετάδοσης που μπορεί να επιτευχθεί.

3.2 Διεπαφή I2C

Έχει αναφερθεί ήδη, ότι το σύστημα που υλοποιείται στα πλαίσια αυτής της διπλωματικής, θα μεταδίδει δεδομένα από ένα Arduino σε οποιονδήποτε flight controller είναι συμβατός με το MPNG. Το πρωτόκολλο που χρησιμοποιείται για τη μεταφορά των δεδομένων είναι το I2C, το οποίο είναι master-slave πρωτόκολλο. Το MPNG έχει ήδη I2C διεπαφή ως master, αφού σε πολλά flight controllers υπάρχουν ήδη ενσωματωμένοι αισθητήρες συμβατοί με το πρωτόκολλο αυτό. Έτσι το σύστημα υπό κατασκευή πρέπει κι εκείνο να μεταδίδει δεδομένα σαν I2C slave. Η απαίτηση, για την υπάρχουσα διπλωματική, είναι η μετάδοση μετρήσεων αισθητήρων, ανεξαρτήτου τύπου και αριθμού, από το σύστημα προς στον flight controller. Είναι γνωστό ότι ο master του I2C πρωτοκόλλου έχει το δικαίωμα να επικοινωνήσει με τους slaves αλλά όχι το αντίστροφο. Από την άλλη ο master δεν γνωρίζει τίποτα για τους αισθητήρες που είναι συνδεδεμένοι πάνω στον slave.

Πώς ο master διαβάζει δεδομένα όταν δεν γνωρίζει τι να ζητήσει από το slave;

Προφανώς, ο master στην εκκίνηση της επικοινωνίας, δεν ξέρει τι αισθητήρες είναι συνδεδεμένοι στο σύστημα, με αποτέλεσμα να μην γνωρίζει τι δεδομένα να ζητήσει. Πρέπει να ενημερωθεί από το slave για τους αισθητήρες που υπάρχουν και τους ρυθμούς με τους οποίους επιθυμεί ο σχεδιαστής, να ανανεώνονται οι μετρήσεις τους. Αλλά ούτε αυτό μπορεί να γίνει αφού ο slave δεν έχει δικαιοδοσία να γράψει στο κανάλι του I2C. Πρέπει με κάποιο τρόπο να του την παραχωρήσει ο master. Πρέπει λοιπόν, να βρεθεί ένας τρόπος ώστε ο master να διαβάζει, από το

slave, κάποια στοιχεία για τους αισθητήρες, ούτως ώστε στη συνέχεια να μπορεί να διαβάζει τα δεδομένα τους. Παρακάτω, τα απαραίτητα στοιχεία για κάθε αισθητήρα, ενσωματώνονται σε μία custom μεταβλητή ώστε να γίνει ευκολότερη η μεταβίβασή των δεδομένων τους από το master στο slave αλλά και η αποθήκευσή τους από το Arduino.

Ορισμός custom μεταβλητών

Η custom μεταβλητή, είναι ένα C enumeration, όπου αποθηκεύονται τα στοιχεία που απαιτούνται για κάθε αισθητήρα. Η μεταβλητή αυτή αποτελείται από τα παρακάτω στοιχεία.

Όνομα	Τύπος	Περιγραφή
name	char*	Ονομασία μεταβλητής-αισθητήρα
value	float	τιμή μέτρησης αισθητήρα
type	enum	τύπος μεταβλητής(οι τύποι μπορεί να είναι τηλεμετρία ή sonar)
sample rate	uint8_t	Ρυθμός δειγματοληψίας μέτρησης

Σχήμα 3.3 Ορίσματα δομής δεδομένων custom μεταβλητής

Για την τιμή του αισθητήρα δεν χρειάζεται να αναφερθούν πολλά. Το μόνο που θα αναφερθεί είναι ότι όλες οι μεταβλητές θα είναι τύπου floating point. Ο τύπος μεταβλητής είναι ένα enumeration, το οποίο παίρνει τιμές telemetry ή sonar. Με τον τύπο μεταβλητής προσδιορίζεται το πώς ο flight controller θα διαχειριστεί τη συγκεκριμένη μεταβλητή. Αν η μεταβλητή είναι τύπου τηλεμετρίας, τότε ο ρυθμός δειγματοληψίας προσδιορίζει το πόσο συχνά ο flight controller θα ζητάει ανανέωση στην τιμή της συγκεκριμένης μεταβλητής. Αν η μεταβλητή είναι τύπου sonar, τότε ο ρυθμός δειγματοληψίας δεν είναι μια πληροφορία που είναι απαραίτητη, καθώς ο flight controller διαβάζει την τιμή του sonar όταν τη χρειαστεί, για να έχει όσο το δυνατόν πιο πρόσφατη τιμή. Στην σχεδίαση υπάρχει και μια συμβολοσειρά για την ονομασία της κάθε μεταβλητής, με πλάνο αυτή να φτάνει στο σταθμό βάσης, ώστε να είναι πιο εύκολη η αναγνώριση της κάθε μεταβλητής από έναν εξωτερικό παρατηρητή.

Μετάδοση δεδομένων μεταξύ flight controller και Συστήματος Μέτρησης

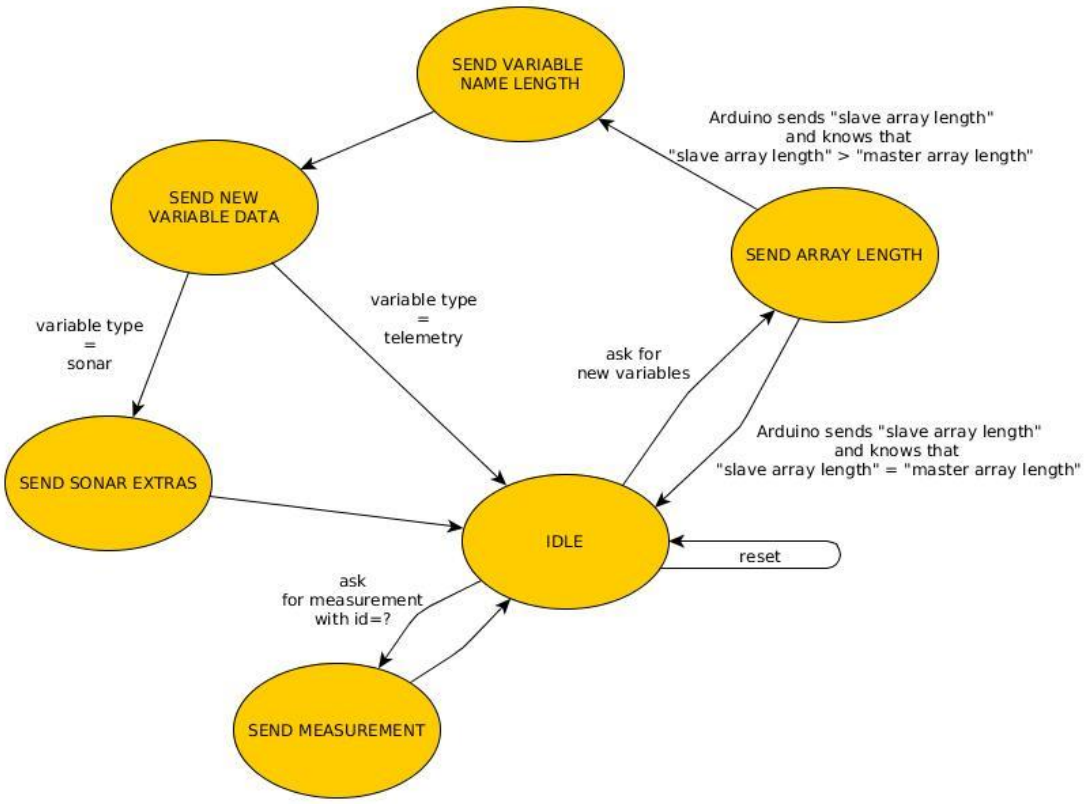
Ο flight controller και το σύστημα μετρήσεων κρατούν από έναν πίνακα το καθένα. Οι πίνακες αυτοί αποθηκεύουν τις custom μεταβλητές που περιγράφονται στο παραπάνω κεφάλαιο. Σκοπός αυτού του κεφαλαίου είναι να περιγράψει τον τρόπο με τον οποίο αυτοί οι δύο πίνακες

τελικά συγχρονίζονται.

Οι ροές μηνυμάτων από τον flight controller στο σύστημα μέτρησης και αντίστροφα είναι εγγραφές και αναγνώσεις byte(s), όπως αυτές παρουσιάζονται στην περιγραφή του I2C πρωτοκόλλου.

Το πρώτο byte κάθε επικοινωνίας είναι η εντολή που δίνει ο flight controller στο σύστημα μέτρησης κι εκείνο λαμβάνοντας την συγκεκριμένη εντολή θα αντιδράσει ανάλογα. Από την παρακάτω περιγραφή των εντολών, τις οποίες έχει ανάγκη ο flight controller για να συγχρονίζει τον πίνακά του με εκείνον του συστήματος μετρήσεων, φαίνεται η ανάγκη το σύστημα μετρήσεων να λειτουργεί σαν μηχανή πεπερασμένων καταστάσεων.

Εδώ δίνεται η πλήρης σχηματική απεικόνιση της μηχανής πεπερασμένων καταστάσεων σύμφωνα με την οποία θα λειτουργήσει το Arduino. Όπως φαίνεται και στο σχήμα 3.4 ο flight controller στέλνει τρείς τύπους μηνυμάτων στο Arduino, reset, ερώτηση μήκους πίνακα μεταβλητών(αισθητήρων), ερώτηση μέτρησης μεταβλητής(αισθητήρα). Το Arduino μετά τη μετάβαση από idle σε κάποια άλλη κατάσταση, αρχίζει να απαντά με πληροφορίες που περιμένει να διαβάσει ο flight controller από το ίδιο. Οπότε κάθε μετάβαση από κάποια κατάσταση εκτός της idle, γίνεται μετά από εκπομπή ενός μηνύματος από το Arduino στον flight controller. Η όλη διαδικασία βασίζεται στο ότι ο flight controller περιμένει να διαβάσει ένα συγκεκριμένο αριθμό από bytes. Προφανώς τον ίδιο αριθμό bytes μεταδίδει προς εκείνον το Arduino.



Σχήμα 3.4 Σχεδιάγραμμα Μηχανής Πεπερασμένων Καταστάσεων Arduino για την επικοινωνία με τον flight controller

Ο flight controller αρχικά στέλνει αιτήσεις για να μάθει το μέγεθος του πίνακα του Arduino και ενημερώνεται για τις μεταβλητές του. Οι αιτήσεις όπως και οτιδήποτε έχει να κάνει με την επικοινωνία Arduino-flight controller γίνεται με συχνότητα 100Hz. Όταν το Arduino συγχρονίσει με τον flight controller τις μεταβλητές του, εκείνος θα συνεχίσει να ρωτάει το μήκος του πίνακα μεταβλητών μέχρι να πάρει 100 φορές την ίδια απάντηση με το δικό του μήκος πίνακα. Αυτό συμβαίνει διότι το Arduino μπορεί να εκκινηθεί μετά τον flight controller με αποτέλεσμα να χρειαστεί περισσότερο χρόνο για να συγχρονίσει τις μεταβλητές του. Όταν ο flight controller συνεχίζει να στέλνει αιτήσεις για νέες μεταβλητές για ένα δευτερόλεπτο(100 φορές με συχνότητα 100Hz) είναι σίγουρο ότι το Arduino θα έχει εκκινηθεί και εκείνο σε αυτό το διάστημα. Όταν η διαδικασία αυτή ολοκληρωθεί ο flight controller θεωρεί ότι έχει συγχρονιστεί με το Arduino και σταματά να ρωτάει το μήκος του πίνακα μεταβλητών.

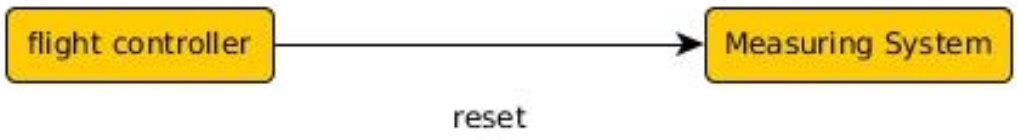
Μόλις τα δύο συστήματα συγχρονιστούν ο flight controller αρχίζει να στέλνει αιτήσεις για τις μεταβλητές που έχει συλλέξει ανάλογα με το ρυθμό δειγματοληψίας που έχει δηλωθεί από το

Arduino για αυτές. Αξίζει να σημειωθεί ότι ο ρυθμός δειγματοληψίας είναι αναγκαίο να είναι ακέραιος διαιρέτης του 100 αφού ανά 100Hz ελέγχονται οι μεταβλητές του πίνακα και αποφασίζεται για ποιες από εκείνες πρέπει να ανανεωθεί η μέτρηση. Αν η συχνότητα που έχει δηλωθεί δεν είναι ακέραιο πολλαπλάσιο του 100 τότε θα επιλεγεί ως συχνότητα δειγματοληψίας ο αμέσως μεγαλύτερος ακέραιος διαιρέτης. Αυτό συμβαίνει διότι ο χρονοπρογραμματιστής του flight controller εκτελεί ρουτίνες με συγκεκριμένες συχνότητες. Ο κώδικας για την επικοινωνία με το Arduino εισήχθη στην πιο συχνά καλούμενη ρουτίνα του flight controller.

Ο αριθμός των εντολών που ορίζονται στην μηχανή πεπερασμένων καταστάσεων είναι αρκετά μικρός(reset, ερώτηση μήκους πίνακα, ερώτηση μέτρησης) ώστε 2 bits να αρκούν για να οριστούν όλες. Επιλέγεται για ευκολία, και ίσως για μελλοντική επεκτασιμότητα του πρωτοκόλλου, η δέσμευση των τριών LSB(least significant bits) για τις εντολές του flight controller. Αυτά τα 3 bits θα αναφέρονται ως instruction id στη συνέχεια. Τα υπόλοιπα πέντε MSB(most significant bits) του πρώτου byte θα χρησιμοποιηθούν βοηθητικά στην εντολή που στέλνεται ως επιπλέον πληροφορία.

Reset

Όταν ο flight controller εκκινείται αυτόματα στέλνει reset στο σύστημα μέτρησης.

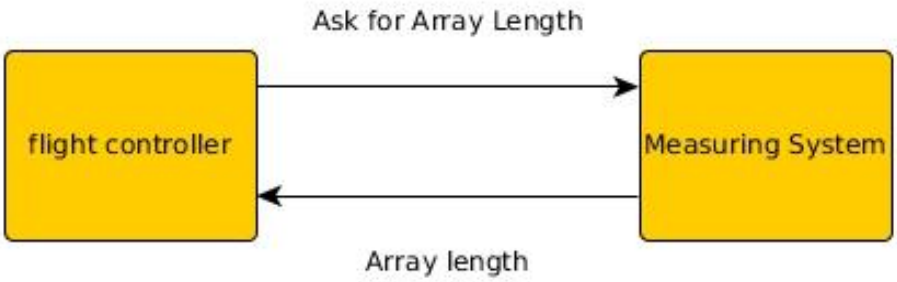


Σχήμα 3.5 Αποστολή Reset από τον flight controller στο σύστημα συλλογής μετρήσεων

Το σύστημα μετρήσεων κρατά έναν μετρητή ώστε ανά πάσα στιγμή να γνωρίζει το μέγεθος του πίνακα μεταβλητών του flight controller. Όταν ληφθεί reset(instruction id = 3), στο σύστημα μετρήσεων, αυτός ο μετρητής μηδενίζεται, με αποτέλεσμα το σύστημα μετρήσεων να αντιλαμβάνεται ότι ο flight controller δεν γνωρίζει τίποτα για τις μεταβλητές των αισθητήρων του.

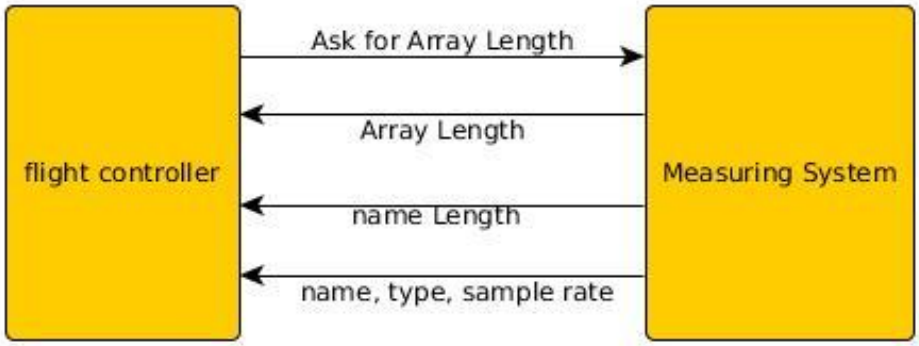
Ερώτηση μήκους πίνακα συστήματος μετρήσεων

Ο flight controller στέλνει στο σύστημα μετρήσεων ερώτηση(instruction id = 1) σχετικά με το μήκος του πίνακα μεταβλητών και εκείνο απαντάει. Ο flight controller συγκρίνει την απάντηση με το μήκος του δικού του πίνακα. Αν τα δύο μήκη είναι ίσα τότε η επικοινωνία σταματά εκεί (σχήμα 3.6).



Σχήμα 3.6 Ερωταπάντηση από τον flight controller προς το σύστημα μετρήσεων για το μήκος του πίνακά του

Αν τα δύο μήκη είναι διαφορετικά τότε ο flight controller θα επιχειρήσει να διαβάσει μια νέα μεταβλητή από το σύστημα (σχήμα 3.7).



Σχήμα 3.7 Σήματα που μεταδίδονται από τα δύο συστήματα κατά την μετάδοση νέας μεταβλητής

Αρχικά το σύστημα μετρήσεων στέλνει το μήκος του ονόματος ώστε να είναι γνωστό αργότερα στον flight controller πόσα bytes θα διαβάσει. Παραδείγματος χάρη, εάν το μήκος του ονόματος, της μεταβλητής, είναι 7 bytes τότε ο flight controller θα διαβάσει 9 bytes, όπου τα πρώτα 7 θα είναι η ονομασία, το επόμενο ο τύπος της μεταβλητής και θα ακολουθήσει ο ρυθμός μετάδοσης.

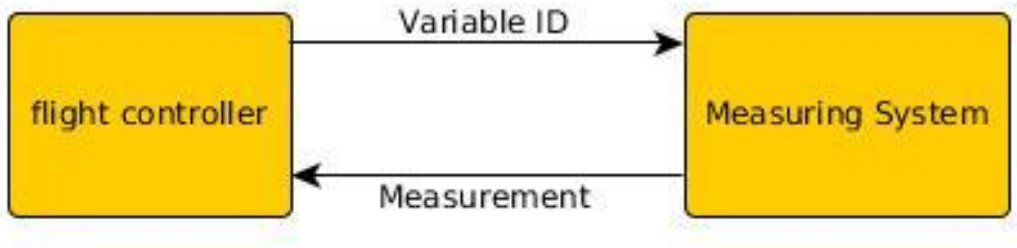
Ο flight controller και το σύστημα μετρήσεων δεν στέλνει επιπλέον εντολή για την ανάγνωση της νέας μεταβλητής. Όπως αναφέρεται και παραπάνω, ο flight controller λαμβάνοντας το μήκος του πίνακα μεταβλητών, του συστήματος μετρήσεων, το συγκρίνει με το μήκος του δικού του πίνακα. Στην περίπτωση που τα δύο μήκη δεν είναι ίσα, ο flight controller θα επιχειρήσει να διαβάσει τα στοιχεία μιας νέας για εκείνον μεταβλητής. Το σύστημα μετρήσεων γνωρίζοντας το μήκος του πίνακα μεταβλητών του flight controller, γνωρίζει ποια είναι η αμέσως επόμενη μεταβλητή που θα του γνωστοποιήσει. Μετά τη μετάδοση δεδομένων το σύστημα μετρήσεων θα

αυξήσει τη μεταβλητή για το μέγεθος του πίνακα του flight controller.

Ο flight controller θα συνεχίσει να ζητάει από το σύστημα μετρήσεων νέες μεταβλητές, μέχρις ότου τα δύο συστήματα να συγχρονιστούν.

Ερώτηση μέτρησης αισθητήρα

Οι πίνακες μεταβλητών των, flight controller και συστήματος μετρήσεων, συγχρονίζονται συνεχώς ώστε να διατηρούν στις ίδιες θέσεις, τις ίδιες μεταβλητές, με αποτέλεσμα να δημιουργείται μεγάλη ευκολία στη μετάδοση των μετρήσεων μεταξύ των δύο συστημάτων. Ο flight controller γνωρίζοντας το ρυθμό δειγματοληψίας, που επιθυμεί ο σχεδιαστής για κάθε αισθητήρα, διαβάζει με εκείνο το ρυθμό δεδομένα από το σύστημα μετρήσεων. Η ανάγνωση της μέτρησης για έναν αισθητήρα γίνεται με βάση τη θέση του στον πίνακα μεταβλητών. Ο flight controller στέλνει το πρώτο byte, το οποίο είναι η εντολή προς το σύστημα μετρήσεων, όπως και στις δύο προηγούμενες περιπτώσεις(instruction id = 0). Εδώ χρησιμοποιούνται τα 5 MSB του πρώτου byte για να γνωστοποιηθεί στο σύστημα μετρήσεων η μεταβλητή για την οποία, ο flight controller, ζητάει μέτρηση. Τα 5 MSB θα μπορούσαν να θεωρηθούν και ID της μεταβλητής. Αφού στέλνονται μέχρι 5 bits για αυτή τη διαδικασία, αυτόματα το σύστημα μετρήσεων περιορίζεται στην αποθήκευση μέχρι και 32(2^5) μεταβλητών. Το σύστημα μετρήσεων διαβάζοντας το αναγνωριστικό ID της μεταβλητής, θα κάνει μια ανάγνωση από το συγκεκριμένο αισθητήρα και θα μεταδώσει σε αυτόν το αποτέλεσμα της μέτρησης.



Σχήμα 3.8 Σήματα που μεταδίδονται από τον flight controller στο σύστημα μετρήσεων κατά τη διαδικασία ερωταπαντήσεων μετρήσεων αισθητήρων

3.3 Τηλεμετρία

Η μετάδοση των δεδομένων για την κατάσταση του πολυκοπτέρου γίνεται με τη βοήθεια του 3DR Radio και του πρωτοκόλλου MAVlink. Η δομή ενός μηνύματος MAVlink παρουσιάζεται

στο παράρτημα (υποκεφάλαιο Δομή ενός MAVlink πακέτου). Στο σχήμα 2.13 φαίνεται ο περιορισμός που υπάρχει για το ID ενός μηνύματος. Το ID του πακέτου είναι μη προσημασμένος ακέραιος μεγέθους 8 bit (uint8_t) με αποτέλεσμα να μπορεί να πάρει τιμές από 0 μέχρι και 255. Τα περισσότερα IDs είναι ήδη κατειλημμένα και για πολλά από αυτά που δεν είναι, υπάρχουν comments ότι θα καταληφθούν, στην επόμενη έκδοση του MAVlink, για μηνύματα που έχουν να κάνουν με την χρήση GoPro κάμερας. Έτσι δεν είναι εφικτή η ενσωμάτωση πολλών νέων πακέτων στο σύστημα.

Επιπλέον περιορισμό αποτελεί ο μικρός buffer ορισμένων radio modem. Ο μικρότερος buffer που υπάρχει στα εμπορικά radio modems είναι 64 bytes. Στην υπάρχουσα διπλωματική αυτό δεν αποτελεί περιορισμό αφού το 3DR Radio έχει buffer 120 byte για τη μετάδοση πακέτων.

Η πρώτη υλοποίηση που έγινε για την αποστολή των μετρήσεων ήταν η υλοποίηση δύο επιπλέον πακέτων για τη μετάδοση των ονομάτων των μεταβλητών αλλά και των μετρήσεων.

3.3.1 Χρήση δύο πακέτων

Πακέτο μετρήσεων

Είναι βέβαιο ότι τουλάχιστον ένα πακέτο είναι απαραίτητο, ώστε να είναι εφικτή η μετάδοση των μετρήσεων του νέου συστήματος. Αυτό το πακέτο είναι υπεραρκετό ούτως ώστε ένας σχεδιαστής, που κατασκεύασε ένα σύστημα συλλογής μετρήσεων αισθητήρων σε ένα Arduino, να παρακολουθεί, τα δεδομένα που συλλέγονται από το Arduino, και φτάνουν στο σταθμό βάσης, αφού η σειρά που αυτά θα εισαχθούν στο Arduino δεν θα μεταβληθεί μέχρι την άφιξή τους στο δέκτη(Από την παραπάνω περιγραφή για το συγχρονισμό του συστήματος μετρήσεων και του flight controller, φαίνεται ότι η σειρά των μεταβλητών δεν μεταβάλλεται). Το πακέτο θα μεταφέρει μόνο τις μετρήσεις των αισθητήρων, δηλαδή μόνο floating point μεταβλητές. Αυτό εισάγει περιορισμό στον αριθμό των μεταβλητών που μπορούν να μεταδοθούν με ένα πακέτο. Ο αριθμός των μεταβλητών περιορίζεται αυτόματα στις 14 αφού κάθε μέτρηση απαιτεί 4 bytes(4x14=56 bytes). Ο ρυθμός μετάδοσης των μετρήσεων είναι ο ρυθμός μετάδοσης της μεταβλητής με το μεγαλύτερο ρυθμό δειγματοληψίας. Το πακέτο παίρνει την ακόλουθη μορφή.

```
<message name="Telemetry Values" id="190">
  <description>Data packet, size 64</description>
  <field type="float" name="val1">value 1</field>
  <field type="float" name="val2">value 2</field>
  <field type="float" name="val3">value 3</field>
```

Κλπ.…
</message>

Πακέτο ονομάτων

Για να βελτιωθεί η απεικόνιση των μηνυμάτων, θα προστεθεί ακόμα ένα μήνυμα το οποίο θα μεταδίδει τις ονομασίες των μεταβλητών και γι' αυτό το λόγο θα δεσμευτεί ακόμα ένα από τα IDs για τα μηνύματα του MAVlink. Το νέο πακέτο ενσωματώνεται μόνο για ευκολότερη αντιστοίχιση των μετρήσεων με τις ονομασίες των ενσωματωμένων αισθητήρων του συστήματος. Επειδή το νέο πακέτο δεν είναι απολύτως απαραίτητο πρέπει να μην μειώσει το ρυθμό μετάδοσης των δεδομένων. Αυτό μπορεί επιτευχθεί αν το πακέτο με τα ονόματα μεταδοθεί μία φορά και μετά σταματήσει, μιας και τα ονόματα των μεταβλητών δεν μεταβάλλονται οπότε δεν υπάρχει και λόγος να επαναμεταδίδονται από το flight controller και να καταναλώνουν χρήσιμα bytes από τη μετάδοση (throughput).

Ένας περιορισμός που διέπει και το δεύτερο πακέτο είναι το μέγεθος του. Το μέγεθος και εδώ περιορίζεται στα 56 bytes με τη διαφορά ότι σε αυτό το πακέτο πρέπει να μεταδοθούν συμβολοσειρές(τα ονόματα είναι τύπου char*). Προφανώς το μέγεθος των συμβολοσειρών είναι πολύ μεγαλύτερο των 4 byte(που είναι οι floating point μετρήσεις) με αποτέλεσμα να μην υπάρχει αρκετός χώρος στο πακέτο για να ενσωματωθούν όλες οι συμβολοσειρές. Η επιλογή να κατασκευαστούν περισσότερα του ενός πακέτα για την μετάδοση των ονομάτων, των μεταβλητών, φαντάζει απαγορευτική λόγο των περιορισμένων IDs του MAVlink. Αρά πρέπει να βρεθεί μια πιο έξυπνη τακτική για τη μετάδοση αυτού του μηνύματος. Αυτό θα συμβεί μεταδίδοντας τρία ονόματα ανά πακέτο. Με αυτόν τον τρόπο το μέγεθος της συμβολοσειράς του ονόματος μπορεί να είναι μέχρι 17 bytes(πρακτικά μέχρι 17 χαρακτήρες), το οποίο είναι αρκετό για ονομασία μεταβλητής. Το 17 δεν προκύπτει τυχαία. Αν χρησιμοποιούνταν μια ονομασία ανά πακέτο θα μπορούσε να είναι μέχρι 56 bytes, το οποίο είναι υπερβολικό για ονομασία μεταβλητής. Αντίστοιχα, αν οι μεταβλητές ήταν 2 το μέγεθος της ονομασίας θα μπορούσε να είναι μέχρι 28 bytes(2x28=56), το οποίο είναι καλύτερο αλλά και πάλι φαντάζει υπερβολικό για ονομασία μεταβλητής. Έτσι η σχεδίαση κατέληξε στα 3 ονόματα ανά πακέτο, με μέγεθος ονόματος 18 bytes(3x17=53<56). Επειδή η μετάδοση των ονομάτων θα γίνεται στο ίδιο πακέτο, θα εισαχθεί ένα επιπλέον αναγνωριστικό ID στο πακέτο ώστε να είναι εύκολα προσδιοριστέο για ποιες θέσεις του πίνακα μεταβλητών πρόκειται. Για να συμβεί αυτό απαιτείται ένα επιπλέον byte για το πακέτο(54 bytes πλέον). Το μήκος επιτρεπόμενου ονόματος περιορίζεται στα 17 bytes και

εισάγεται στο πακέτο μία επιπλέον μεταβλητή ID. Τα ονόματα στην πρώτη, δεύτερη και τρίτη θέση του πίνακα μεταδίδονται με το ID ένα, στην τέταρτη, πέμπτη και έκτη θέση με το ID δύο κτλ. Από αυτό το πακέτο ο σταθμός βάσης μπορεί να αποθηκεύει τα ονόματα και να αναπαριστά την κάθε μέτρηση μαζί με το αναγνωριστικό όνομα του αισθητήρα της. Το όνομα μένει σταθερό για όλη τη διάρκεια λειτουργίας του συστήματος με αποτέλεσμα να είναι περιττή η αποστολή των ονομάτων περισσότερες από μία φορές. Το πακέτο παίρνει την ακόλουθη μορφή.

```
<message name="Telemetry Names" id="191">
 <description>Data packet, size 64</description>
 <field type="uint8_t" name="ID">ID field</field>
 <field type="char[17]" name="name1">var name 1</field>
 <field type="char[17]" name="name2">var name 2</field>
  <field type="char[17]" name="name3">var name 3</field>
</message>
```

Η σχεδίαση των δύο προηγουμένων παραγράφων έχει το κακό ότι μεταδίδονται μετρήσεις οι οποίες δεν έχουν μεταβληθεί. Επειδή ο ρυθμός δειγματοληψίας του πακέτου μεταβλητών είναι εκείνος της μεταβλητής με το μεγαλύτερο ρυθμό δειγματοληψίας οι υπόλοιπες μετρήσεις που δεν έχουν ανανεωθεί απλά μεταδίδονται αμετάβλητες. Παρακάτω δίνεται μια δεύτερη υλοποίηση η οποία χρησιμοποιεί ένα μόνο πακέτο

3.3.2 Χρήση ενός μόνο πακέτου

Μια διαφορετική σχεδίαση από την προηγούμενη είναι να χρησιμοποιηθεί ένα μόνο πακέτο MAVlink και να επανασχεδιαστεί ο τρόπος με τον οποίο μεταδίδονται ονομασίες και μεταβλητές.

```
<message name=" Telemetry Data" id="190">
  <description>Data packet for telemetry data</description>
  <field type="uint8_t" name="type">data type</field>
  <field type="uint8_t[55]" name="data">raw data</field>
</message>
```

Για να μεταδίδονται ονομασίες και μεταβλητές μέσα στο ίδιο πακέτο πρέπει να βρεθεί ένας τρόπος αναγνώρισης του περιεχομένου του πακέτου. Αυτό μπορεί να εισαχθεί στο πακέτο μέσω του type.

Οι τιμές του type από 0 ως 9 δεσμεύονται για να μεταβιβάζονται οι ονομασίες των μεταβλητών. Το ονόματα των μεταβλητών έχουν οριστεί να είναι το πολύ 18 χαρακτήρων οπότε

σε ένα πακέτο χωράνε 3 ονόματα μεταβλητών. Τα ονόματα εισάγονται στο πακέτο, στο όρισμα data, το ένα ακολουθούμενο από το επόμενο με διαχωρισμό ενός χαρακτήρα '\n' μεταξύ τους εκτός και αν το όνομα είναι ακριβώς 18 bytes οπότε ξεκινά το επόμενο χωρίς '\n' ανάμεσα στα 2 ονόματα. Άρα το πακέτο που στέλνεται μπορεί να μην είναι γεμάτο σε περίπτωση που οι ονομασίες των μεταβλητών δεν είναι 18 bytes η κάθε μία. Με 10 πακέτα όπου το καθένα μπορεί να περιέχει μόνο 3 μεταβλητές, ο συνολικός αριθμός μεταβλητών που μπορούν να σταλούν προς το σταθμό βάσης είναι 30.

Οι τιμές από 10,11 και 12 δεσμεύονται με σκοπό τη μετάδοση των μετρήσεων. Κάθε μεταβλητή μεταδίδεται με την αρίθμησή της στον πίνακα μεταβλητών και την τελευταία τιμή της. Το πακέτο που μεταδίδεται στην πρώτη θέση(θέση 0) του πίνακα data έχει την θέση την μεταβλητής στον πίνακα δεδομένων και στις 4 επόμενες την τιμή της μεταβλητής, στην επόμενη θέση(θέση 5) θα περιέχει τη θέση της επόμενης μεταβλητής και στις επόμενες 4 τη μέτρηση που μεταδίδεται για αυτήν κοκ. Με αυτό τον τρόπο μπορεί εύκολα να γίνει αντιστοίχιση των μετρήσεων στις ονομασίες των μεταβλητών τους.

Η παραπάνω σχεδίαση με ένα πακέτο έχει το πλεονέκτημα σε σχέση με την προηγούμενη ότι δεν μεταδίδονται μετρήσεις οι οποίες δεν έχουν ανανεωθεί στον flight controller διότι η δομή του πακέτου δεν το απαιτεί. Αν η εφαρμογή είναι συγκεκριμένη και δεν απαιτείται να γεμίσουν όλα τα bytes του πίνακα data(πχ μεταδίδονται μόνο 4 μεταβλητές ανά πακέτο, τότε μπορεί να εισαχθεί επιπλέον πληροφορία στο πακέτο που δεν αφορά τηλεμετρικά δεδομένα αλλά κάτι άλλο που θέλει να στείλει ο σχεδιαστής. Τα πακέτα ονομάτων όπως και στην προηγούμενη σχεδίαση μπορούν να μεταδοθούν στην εκκίνηση του flight controller και στη συνέχεια να μην επαναμεταδοθούν αφού οι ονομασίες των μεταβλητών είναι ήδη γνωστές.

3.4 Ενσωμάτωση Αισθητήρων εγγύτητας

Στο Megapirate NG μπορεί να ενσωματωθεί ένα sonar, συγκεκριμένου τύπου, για τη μέτρηση του υψομέτρου. Για το συγκεκριμένο sonar είναι ήδη υλοποιημένη η λειτουργία alt hold. Αφού δεν υπάρχει καμία υποστήριξη για άλλους αισθητήρες εγγύτητας στο Megapirate NG, θα πρέπει να υλοποιηθούν και οι λειτουργίες που αναμένεται να εκτελεί το πολυκόπτερο.

Επειδή η οικονομικότερη λύση για την υλοποίηση ήταν η χρήση υπερηχητικών αισθητήρων, οι αισθητήρες απόστασης/εγγύτητας αναφέρονται ως υπερηχητικοί αισθητήρες και αυτοί είναι που χρησιμοποιούνται και στην υλοποίηση. Πολύ εύκολα ένας υπερηχητικός

αισθητήρας θα μπορούσε να αντικατασταθεί από ένα infrared proximity sensor.

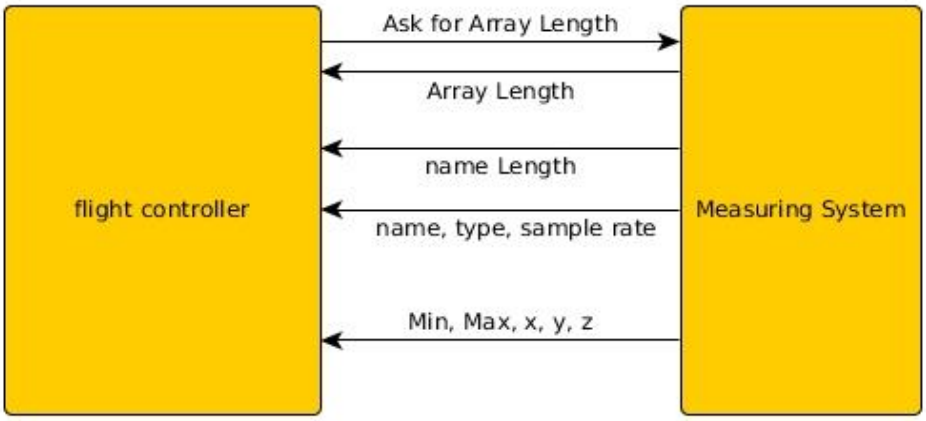
Παρακάτω παρουσιάζονται δύο τρόποι με τους οποίους μπορεί να ενσωματωθεί μια διάταξη με αισθητήρες εγγύτητας περιμετρικά του πολυκοπτέρου με σκοπό να γίνεται από το πολυκόπτερο βασική αποφυγή συγκρούσεων. Στο τέλος του κεφαλαίου θα αναλυθούν τα πλεονεκτήματα και μειονεκτήματα της κάθε μεθόδου.

3.4.1 Διαχείριση μετρήσεων sonar από τον flight controller

Παραπάνω αναφέρθηκαν οι τρόποι με τους οποίους θα διαδίδονται τα δεδομένα από το σύστημα μετρήσεων στο σταθμό βάσης. Με τον ίδιο τρόπο που διαδίδονται οι υπόλοιπες μετρήσεις, δηλαδή εκείνες των υπερηχητικών αισθητήρων. Οι υπερηχητικοί αισθητήρες όμως, είναι μια μέτρηση που δεν θα χρησιμοποιηθεί απλά για μετάδοση. Είναι μια μέτρηση που μπορεί να χρησιμοποιήσει ο flight controller για να βελτιώσει την ίδια του την επιχειρησιακή λειτουργία. Γι' αυτό το λόγο και παραπάνω, όπου ορίζονται οι custom μεταβλητές του συστήματος, ορίστηκε και ο τύπος sonar σαν χαρακτηριστικό της μεταβλητής. Όταν η μεταβλητή είναι τύπου sonar, αυτό που πρωτίστως ενδιαφέρει το flight controller, είναι η τοποθεσία του sonar πάνω στο πολυκόπτερο. Με άλλα λόγια πρέπει να προσδιοριστεί η θέση του υπερηχητικού αισθητήρα, για να υπάρξει αντίληψη ως προς το ποια απόσταση είναι αυτή που μετριέται. Διαφορετική πρέπει να είναι η αντίδραση του πολυκοπτέρου εάν μετρηθεί μεταβολή, στην μέτρηση του sonar που μετρά την απόσταση από το έδαφος και διαφορετική για sonar που έχει τοποθετηθεί με τέτοιο τρόπο πάνω στο πολυκόπτερο, ώστε να μετρά την απόστασή του πολυκοπτέρου από κάποιο αντικείμενο που βρίσκεται μπροστά του.

Έτσι αποφασίστηκε η επέκταση της εντολής δήλωσης μεταβλητής σε αυτή που φαίνεται στο σχήμα 3.8, παρακάτω. Εάν ο τύπος της μεταβλητής είναι sonar, τότε πρέπει να αποσταλούν από το σύστημα μετρήσεων κάποιες επιπλέον πληροφορίες. Αυτές είναι, η ελάχιστη και μέγιστη απόσταση που μπορεί να μετρηθεί από το sonar, ώστε ο flight controller να γνωρίζει πού πρέπει να σταματήσει να λαμβάνει υπόψιν τη μέτρησή του. Επίσης από το σύστημα μετρήσεων δηλώνονται τα x, y, z, τα οποία είναι παράμετροι που προσδιορίζουν πού βρίσκεται το sonar σε σχέση με το κέντρο βάρους του πολυκοπτέρου. Με αυτόν τον τρόπο επιχειρείται να προσδιοριστεί η θέση του sonar στο flight controller. Παραδείγματος χάρη το sonar με παραμέτρους Min = 2, Max = 500 , και [x y z] = [0 0 10] θα αναγνωριστεί από το flight controller σαν μέτρηση εγγύτητας του πολυκοπτέρου με το έδαφος, και μπορεί να μετρήσει τιμές από 2 ως 500 cm. Αν το sonar

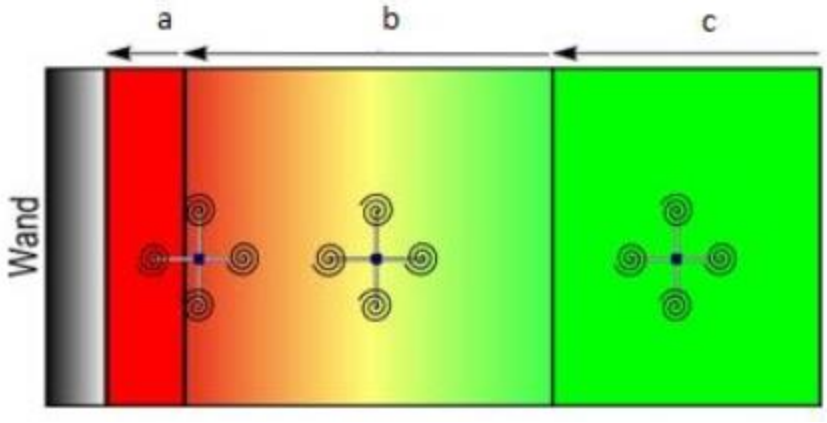
δηλωθεί με [x y z] = [0 0 -10] τότε ο flight controller θα θεωρήσει τη μέτρηση του sonar ως απόσταση από το ταβάνι ή ένα αντικείμενο που βρίσκεται πάνω από το πολυκόπτερο.



Σχήμα 3.9 Μετάδοση πληροφορίας για τη δήλωση νέας μεταβλητής στο σύστημα. Η νέα μεταβλητή είναι τύπου sonar

Όσον αφορά το sonar για την μέτρηση ύψους, το Megapirate NG υποστηρίζει την ενσωμάτωση του στον flight controller. Υποστηρίζονται μόνο συγκεκριμένοι τύποι sonar, από συγκεκριμένη εταιρία. Για να υποστηριχτούν όλων των ειδών οι αισθητήρες μέτρησης αποστάσεων, είτε πρόκειται για sonar, είτε για αισθητήρα που μετρά αποστάσεις μέσω της ανάκλασης της υπέρυθρης ακτινοβολίας(infrared proximity sensors), θα κατασκευαστεί μια επιπλέον βιβλιοθήκη στο Megapirate NG η οποία θα διαβάζει τις μετρήσεις εγγύτητας των αισθητήρων και ανάλογα με την τοποθέτηση τους στο πολυκόπτερο, ο flight controller θα τις διαχειρίζεται ανάλογα. Προφανώς η ανεξαρτησία του αισθητήρα εγγύτητας προέρχεται από το γεγονός ότι η ανάγνωση της μέτρησής του προέρχεται από το Arduino και στον flight controller γίνεται απλά ενημέρωση της μέτρησης. Οπότε ότι αισθητήρας εγγύτητας και να ενσωματωθεί στο Arduino, η αντιμετώπισή του από τον flight controller θα είναι η ίδια.

Για τη συμπεριφορά του πολυκοπτέρου όταν συναντά εμπόδιο θα υιοθετηθεί η προσέγγιση των N. Gageik et. al. [4] που περιγράφεται στο σχήμα 3.10. Η παρακάτω προσέγγιση προκαλεί διαφορετικές αντιδράσεις στο πολυκόπτερο ανάλογα με την απόσταση που εκτιμάται από ένα εμπόδιο.



Σχήμα 3.10 Σχεδίαση αποφυγής συγκρούσεων με κλίμακες

Για να εξηγηθεί καλύτερα το παραπάνω, θα αναλυθεί η συμπεριφορά του καναλιού roll, το οποίο ορίζει την κίνηση του πολυκοπτέρου δεξιά-αριστερά(Σχήμα 2.1-c). Το πολυκόπτερο για τιμές μεγαλύτερες του μηδενός τιμές θα κινηθεί δεξιά ενώ για τιμές μικρότερες του μηδενός θα κινηθεί αριστερά. Τότε οι τιμές που θα παίρνει το roll όταν συναντά εμπόδιο θα δίνονται από την ακόλουθη σχέση:

roll_out = roll_in +k* (sonar_left - sonar_right)

το k είναι ένας συντελεστής κλιμάκωσης για την επιτάχυνση του πολυκοπτέρου

Προφανώς θα χρησιμοποιηθούν δύο ή περισσότερες τέτοιες σχέσεις για την αποφυγή συγκρούσεων με διαφορετικά k=k1,k2… ώστε να είναι περισσότερο έντονη η αντίδραση του πολυκοπτέρου σε μικρότερες αποστάσεις από το εμπόδιο.

Η παραπάνω σχέση ορίζεται σε περιπτώσεις όπου και τα δύο sonar ανιχνεύουν αντικείμενο σε απόσταση μικρότερη κάποιας τιμής, αλλιώς η μέτρηση του sonar θα θεωρηθεί μηδέν.

Αν ένα εμπόδιο πλησιάζει προς το πολυκόπτερο, εκείνο θα αρχίσει να απομακρύνεται. Αν το εμπόδιο κινείται με μεγαλύτερη ταχύτητα και συνεχίσει να πλησιάζει το πολυκόπτερο θα αυξήσει την ταχύτητά του. Η παραπάνω σχέση είναι βολική για βασική αποφυγή εμποδίων, αφού αν πλησιάζουν δύο εμπόδια από αντίθετες κατευθύνσεις το πολυκόπτερο θα παραμείνει στο κέντρο των εμποδίων μέχρι να συγκρουστεί με αυτά. Για να αποφύγει τη σύγκρουση το πολυκόπτερο θα έπρεπε να κινηθεί μπροστά ή πίσω και αυτό θα μπορούσε να το κάνει υπό περιορισμούς. Η αποφυγή σύγκρουσης σε τέτοιες περιπτώσεις είναι πλέον θέμα τεχνητής νοημοσύνης πράγμα με το οποίο δεν θα απασχολήσει την παρούσα διπλωματική.

4 Σχεδίαση και Υλοποίηση

Σε αυτό το κεφάλαιο, αρχικά παρουσιάζεται η διαδικασία που ακολουθήθηκε, ώστε η επικοινωνία μεταξύ flight controller και συστήματος συλλογής μετρήσεων, να αποκτήσει ένα δικό της πρωτόκολλο επικοινωνίας, ιδικά σχεδιασμένο για τη μετάδοση μετρήσεων από αισθητήρες. Στη συνέχεια θα παρουσιαστεί η μέθοδος με την οποία τα δεδομένα μεταδίδονται από τον flight controller στο σταθμό βάσης, και ο τρόπος με τον οποίο χρειάστηκε να μεταβληθεί το λογισμικό του σταθμού βάσης ώστε να παρουσιάζει και τα δεδομένα που συλλέχθηκαν από το σύστημα μετρήσεων. Τέλος θα παρουσιαστεί η μηχανική και ηλεκτρονική ενσωμάτωση υπερηχητικών αισθητήρων στο σύστημα.

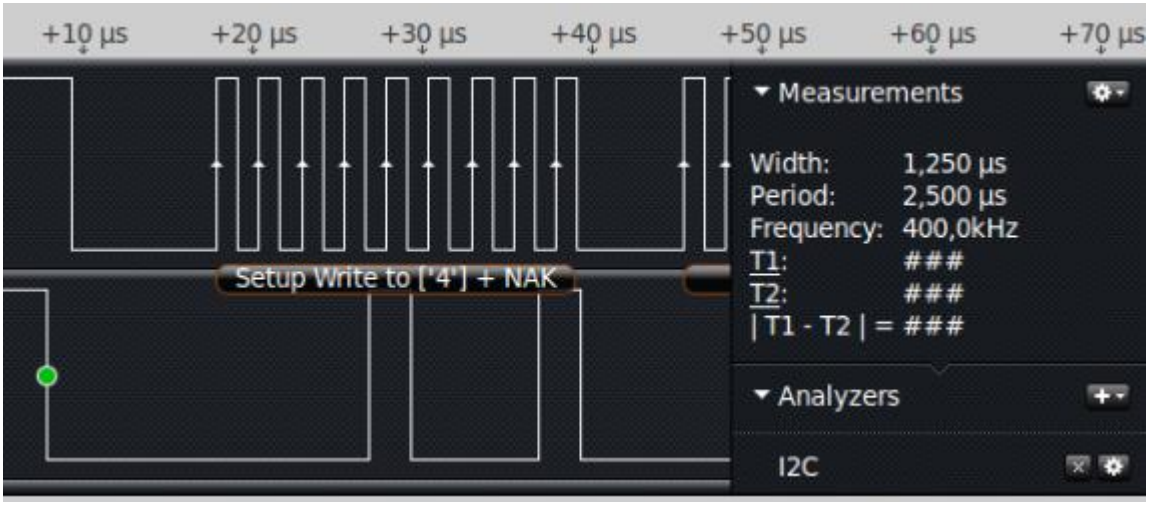
Για την παρούσα διπλωματική αποφασίστηκε η χρήση Arduino Mini, ως σύστημα συλλογής μετρήσεων, καθώς το μικρό του μέγεθος και βάρος, το καθιστούν ιδανικό για περιπτώσεις όπου η συλλογή μετρήσεων από αισθητήρες είναι ο αυτοσκοπός της εφαρμογής. Από το Arduino Mini απαιτείται η συλλογή μετρήσεων και η μετάδοσή τους στον flight controller σαν I2C slave.

4.1 Λειτουργία Arduino ως επέκταση του flight controller

Το Arduino λειτουργεί σαν I2C slave χρησιμοποιώντας την open source βιβλιοθήκη του Arduino Wire. Η Wire προσφέρει στον προγραμματιστή το πλεονέκτημα της σχεδίασης του Arduino σαν I2C master ή slave, αφαιρώντας την ανάγκη για εγγραφή σε καταχωρητές, υλοποίηση interrupt service routines για τα hardware interrupts του I2C και γενικώς εισάγει ένα επίπεδο αφαίρεσης από την υλοποίηση προγράμματος C ή Assembly για άμεση επικοινωνία με το hardware του μικροεπεξεργαστή. Αντιθέτως τα δεδομένα μπορούν να διαβαστούν και να σταλούν μεσώ έτοιμων συναρτήσεων της βιβλιοθήκης. Προφανώς και πάλι υπάρχουν interrupt service routines σε ένα υψηλότερο επίπεδο σχεδίασης με σκοπό την υλοποίηση των ερωταπαντήσεων μεταξύ master και slave, από τον προγραμματιστή.

Στο Arduino δόθηκε η διεύθυνση τέσσερα(4). Αυτό αποφασίστηκε μετά από αναζήτηση στις βιβλιοθήκες του MPNG και βεβαίωση ότι κανένας αισθητήρας από εκείνους που είναι ενσωματωμένοι στα boards που υποστηρίζει το MPNG, δεν έχει αυτή τη διεύθυνση. Στο σχήμα 4.1 φαίνεται μια εκκίνηση επικοινωνίας μεταξύ flight controller και Arduino. Στο σχήμα παρατηρείται ότι η συχνότητα του SCL(πάνω παλμός στο σχήμα 4.1) είναι 400 kHz. Για να συμβεί αυτό έπρεπε να πανωγραφεί μετά την αρχικοποίηση της Wire, ο καταχωρητής TWBR του

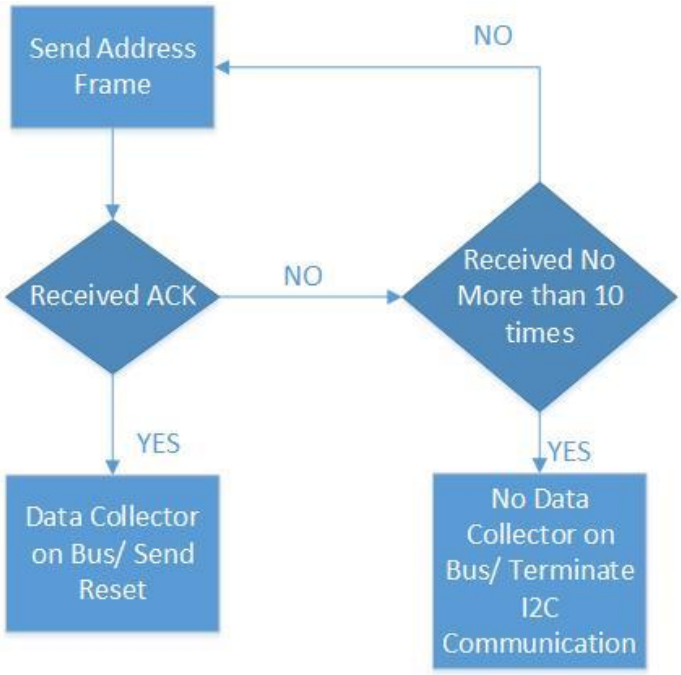
ATmega238P του Arduino. Στο σχήμα φαίνεται η εκκίνηση της επικοινωνίας με start condition όπως αυτό περιγράφεται στην ανάλυση των σημάτων στο I2C και στη συνέχεια φαίνεται η αποστολή του address frame ακολουθούμενο από ένα NACK. Η εικόνα του σχήματος όπως και οι επόμενες λήφθηκαν με logical analyzer.



Σχήμα 4.1 Εκκίνηση επικοινωνίας Arduino και flight controller

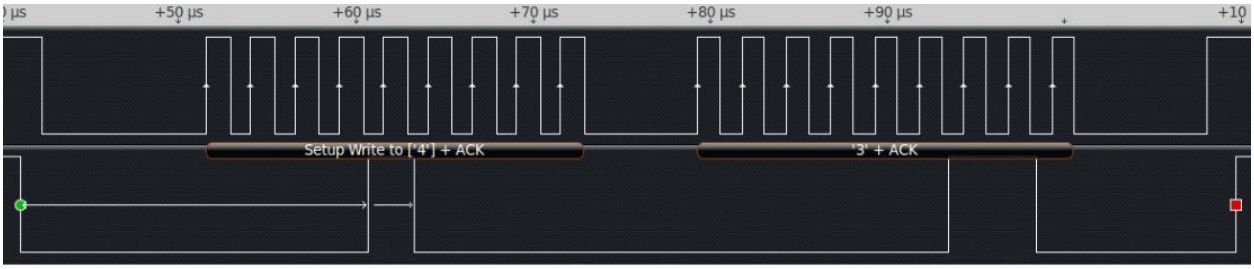
Το Arduino και ο flight controller εκκινούνται ταυτόχρονα. Ο flight controller κατά την εκκίνησή του, επιχειρεί να στείλει στο Arduino ένα μήνυμα reset. Στη σχετική θεωρία για το I2C, αναφέρεται ότι για κάθε byte που στέλνεται από ένα master του I2C σε ένα slave ή το αντίστροφο, ακολουθείται ένα ACK ή ένα NACK. Για να σταλεί το μήνυμα reset, πρέπει να γίνει μετάδοση της ακόλουθης σειράς πληροφορίας. Αρχικά πρέπει να εκδοθεί start condition, που στη συνέχεια θα ακολουθηθεί από μετάδοση ενός byte, όπου στα πρώτα 7 bits θα περιέχει το address frame και το 8^{ο} την πράξη που πρόκειται να ακολουθήσει(εγγραφή ή ανάγνωση), στην παρούσα περίπτωσή εγγραφή, αφού πρέπει να σταλεί το μήνυμα reset. Μετά από το προαναφερθέν byte θα ληφθεί ACK ή NACK ανάλογα με την ύπαρξη ή ανυπαρξία του Arduino στο κανάλι. Εάν το Arduino απαντήσει θετικά, δηλαδή ACK θα αποσταλεί ο αριθμός τρία(3) ως reset. Το Arduino σε αυτό το σημείο αναγκαστικά θα απαντήσει με ACK και ο flight controller θα εκδώσει stop condition με το οποίο και κλείνει η επικοινωνία. Στην περίπτωση που το πρώτο byte(address frame + εγγραφή\ανάγνωση) απαντηθεί με NACK, τότε ο flight controller έχει να αντιμετωπίσει τα ακόλουθα σενάρια. Το Arduino δεν είναι συνδεδεμένο στο σύστημα με αποτέλεσμα να πρέπει να συνεχιστεί η λειτουργία του, χωρίς τη χρήση δεδομένων τηλεμετρίας από Arduino ή το Arduino είναι στο σύστημα και έχει καθυστερήσει να εκκινηθεί. Για να υπάρξει η βεβαιότητα ότι το

Arduino δεν υπάρχει στο σύστημα πριν συνεχιστεί η λειτουργία του flight controller, ο τελευταίος θα επιχειρήσει να επανεκκινήσει την επικοινωνία του με το Arduino άλλες 9 φορές. Τα προαναφερθέντα φαίνονται σχηματικά στο παρακάτω διάγραμμα ροής.

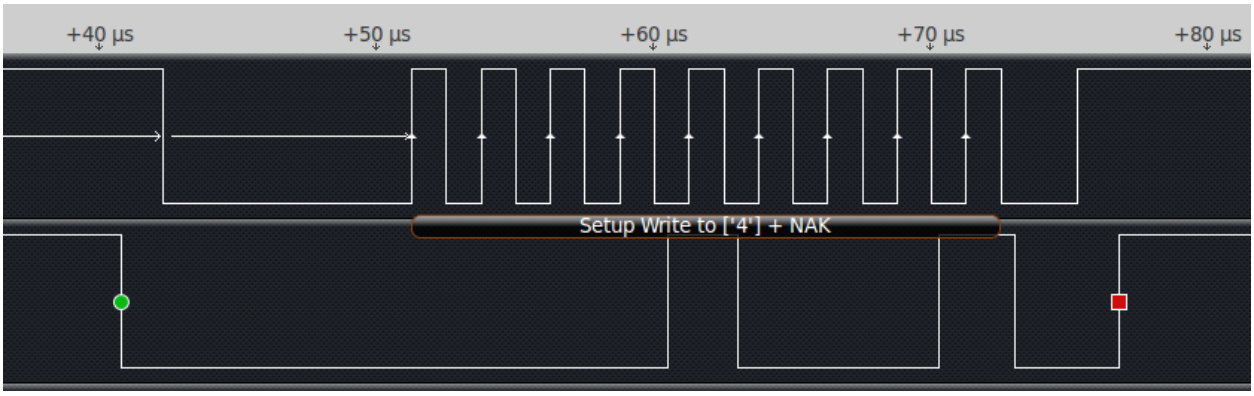


Σχήμα 4.2 Διάγραμμα ροής για την ανίχνευση του Arduino στο κανάλι

Παρακάτω φαίνεται ένα "επιτυχές" reset μεταξύ του flight controller και του Arduino και στη συνέχεια η αρχή μιάς ακολουθίας απο αποτυχιμένα resets.

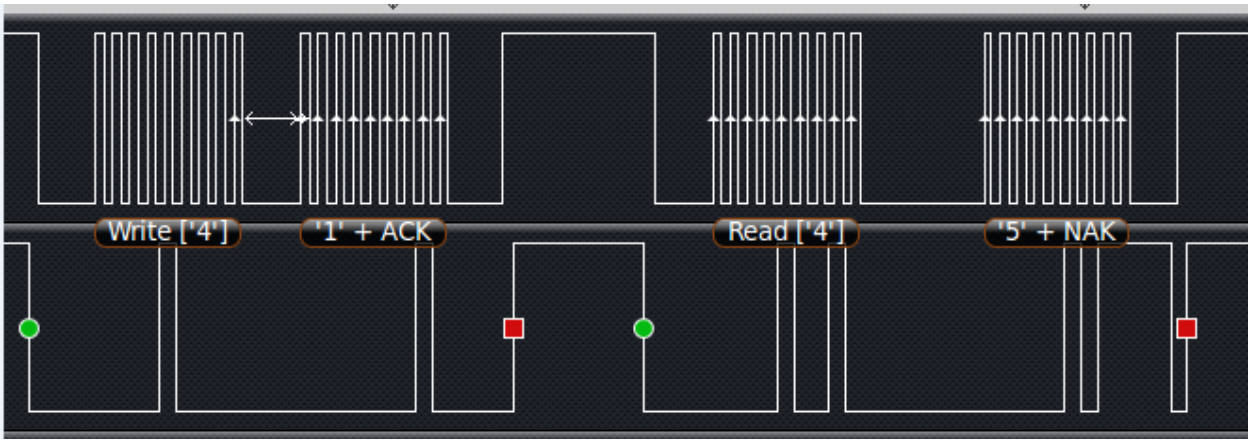


Σχήμα 4.3 Επιτυχημένο reset



Σχήμα 4.4 Αποτυχημένο reset

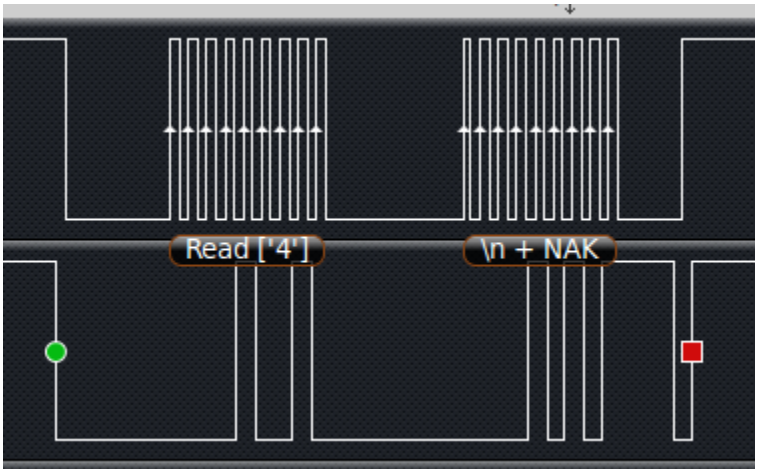
Στη συνέχεια παρουσιάζεται μια ανταλαγή δεδομένων για τη δήλωση μιας νέας μεταβήτής στον flight controller όπως αυτή σχεδιάστηκε στη μοντελοποίηση παραπάνω. Λόγο του μεγάλου μήκους της εικόνας εκείνη παρουσιάζεται σε μικρότερα τμήματα για τα οποία παρακάτω αναφέρονται πληροφορίες αναλυτικά. Στην πρώτη εικόνα ο flight controller κάνει αίτηση για εγγραφή στο στοιχείο του καναλιού με διεύθυνση τέσσερα (4) (δηλαδή το Arduino). Η αίτηση ακολουθείται απο θετική απόκριση και εν συνεχεία αποστολή μηνύματος για ανάγνωση του μήκους πίνακα μεταβλητών, η οποία γίνεται με αποστολή αναγνωριστικού ένα (1). Προφανώς για να διαβαστεί το μήκος πίνακα μεταβλητών, πρέπει αρχικά να γίνει αίτηση άναγνωσης απο το Arduino και μετά απο τη θετική απόκριση του ίδιου να διαβαστεί η τιμή του μήκους του πίνακα η οποία ακολουθείται απο NACK. NACK εκδίδεται πάντα στο τελευταίο byte που μεταδίδεται απο ένα I2C slave με σκοπό την έκδοση, μετά απο αυτό, stop condition απο το master.



Σχήμα 4.5 Ερώτηση για το μήκος πίνακα του arduino και απόκριση ότι μήκος = 5

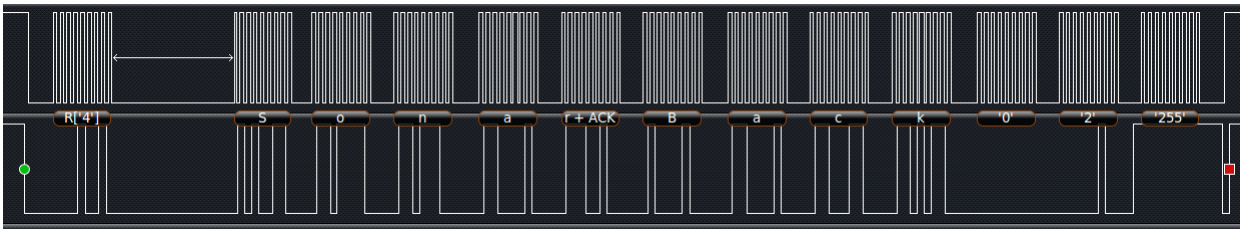
Στην περίπτωση που ο πίνακας μεταβλητών του Arduino είναι μεγαλύτερος από εκείνον

του flight controller, η ανταλαγή μηνυμάτων θα συνεχιστεί με σκοπό να διαβαστεί μία ακόμα μεταβλητή από το Arduino. Παρακάτω φαίνεται η συνέχιση της διαδικασίας αναγνωσης νέας μεταβλητής με την μετάδοση του μήκος ονομασίας της νέας μεταβλητής.



Σχήμα 4.6 Ανάγνωση μήκους ονομασίας νέας μεταβλητής

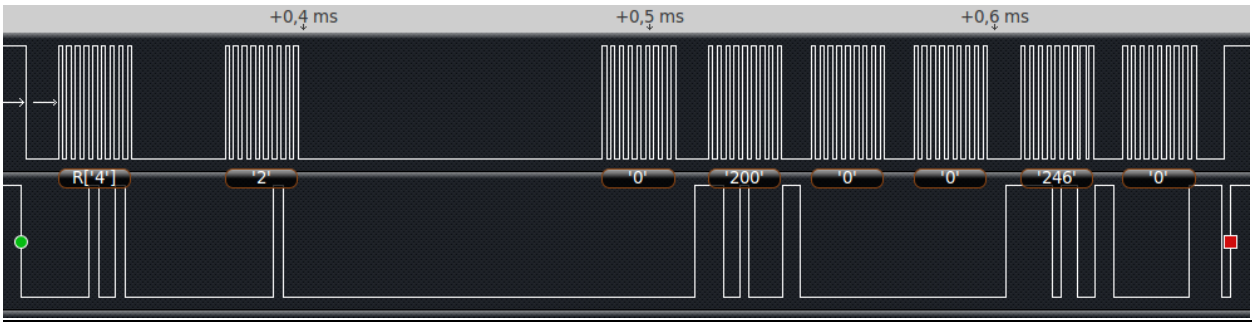
Ο χαρακτήρας \n, που στέλνεται από το Arduino δηλώνει το μήκος ονομασίας της μεταβλητής. Το \n είναι η αναπαράσταση του logical analyzer. Ουσιαστικά είναι η αναπαράσταση του αριθμού 10, που λόγο του κώδικα ascii, το logical analyzer αναπαριστά με το χαρακτήρα \n. Ακολουθείται η μετάδοση του ονόματος ακολουθούμενο από τον τύπο της μεταβλητής και το ρυθμό δειγματοληψίας της μεταβλητής.



Σχήμα 4.7 Μετάδοση του ονόματος της μεταβλητής ακολουθούμενο από τον τύπο και το ρυθμό δειγματοληψίας της

Όπως φαίνεται στην εικόνα 4.7 η μεταβλητή είναι τύπου sonar. Στην περίπτωση που η μεταβλητή ήταν τύπου τηλεμετρίας, τότε με θετικό ρυθμό δειγματοληψίας, θα υπήρχαν όλες οι πληροφορίες που χρειάζονταν προκειμένου να εκκινήθεί η συλλογή μετρήσεων για αυτή τη μεταβλητή. Η υποφαινόμενη μεταβλητή παραπάνω είναι εσκεμένα τύπου sonar ώστε να παρουσιαστούν όλες οι πιθανές περιπτώσεις σε ένα παράδειγμα. Για το sonar έχει ήδη γίνει

γνωστό ότι απαιτείται περισσότερη πληροφορία απο το Arduino, η οποία περιέχει τις συντεταγμένες στις οποίες είναι τοποθετημένο το sonar και την ελάχιστη και μέγιστη απόσταση που μπορεί να μετρήσει το ίδιο. Αυτή η πληροφορία φαίνεται στην παρακάτω εικόνα.



Σχήμα 4.8 Μετάδοση χαρακτηριστικών sonar

Η μετάδοση πληροφορίας είναι πλέον μια απλή διαδικασία δεδομένου ότι, η μεταβλητή για την οποία ο flight controller θέλει να ζητήσει δεδομένα, βρίσκεται στην ίδια θέση του πίνακα μεταβλητών στον ίδιο και στο arduino. Άρα το μόνο που χρειάζεται να κάνει για να διαβάσει δεδομένα είναι να γράψει στο Arduino τη θέση του πίνακα μεταβλητών απο την οποία θέλει να διαβαστούν τα δεδομένα και στη συνέχεια να κάνει μιά ανάγνωση για να τα ζητήσει.

4.2 Λειτουργία Arduino ως συλλογέας μετρήσεων

Δεδομένου ότι το Arduino έχει πλέον μια έτοιμη βιβλιοθήκη με την οποία μπορεί να εξυπηρετεί αιτήσεις για συγχρονισμό και δεδομένα από τον flight controller, μένει να παρουσιαστεί ο τρόπος με τον οποίο το Arduino μπορεί να συλλέγει δεδομένα από τους αισθητήρες που έχουν συνδεθεί στο ίδιο και να τις αποθηκεύει στον πίνακα μεταβλητών.

Το μέρος της συλλογής των μετρήσεων από τους αισθητήρες αποφασίστηκε να αφεθεί στην ευχέρεια του σχεδιαστή του εκάστοτε συστήματος. Αυτό συνέβη διότι δεν υπήρχε εύκολος τρόπος να εισάγεται κάθε αισθητήρας με μεθόδους plug & play στο Arduino και έπρεπε να βρεθεί ένας τρόπος ώστε να ελέγχεται η ορθή λειτουργία των αισθητήρων πριν την πτήση. Έτσι εξήχθη το συμπέρασμα ότι έπρεπε να υλοποιηθούν συναρτήσεις οι οποίες να προσφέρουν έναν εύκολο τρόπο εισαγωγής των αισθητήρων-μεταβλητών στο σύστημα μετάδοσής-συγχρονισμού μεταξύ του Arduino και του flight controller. Σε αυτό το σημείο λοιπόν λήφθηκε η απόφαση σχεδιασμού του πίνακα μεταβλητών-αισθητήρων και ο τρόπος συγχρονισμού τους με τον flight controller.

Οι συναρτήσεις που υλοποιήθηκαν έχουν σκοπό την εισαγωγή των απαραίτητων δεδομένων που απαιτεί ο flight controller από την εκάστοτε μεταβλητή ώστε να μπορεί να γίνει η μεταφορά των δεδομένων από το Arduino στο σταθμό βάσης. Ο σχεδιαστής του συστήματος, αρχικά, πρέπει να δηλώσει τα δεδομένα που απαιτούνται από τον flight controller προκειμένου να ζητήσει αργότερα τις μετρήσεις των αισθητήρων. Η συνάρτηση για τη δήλωση μιας μεταβλητής έχει ως σκοπό να αρχικοποιήσει την custom δομής μεταβλητής που φαίνεται στο σχήμα 3.2 με τη διαφορά της προσθήκης των επιπλέον τιμών που απαιτούνται για τους αισθητήρες εγγύτητας.

Συνάρτηση δήλωσης μεταβλητής τηλεμετρίας

```
int new_telemetry_variable (String name, uint8_t sample_rate);
```

Συνάρτηση δήλωσης μεταβλητής sonar

Προφανώς για τους υπερηχητικούς αισθητήρες η πληροφορία που απαιτείται είναι μεγαλύτερη μιας και απαιτείται πληροφορία για τη θέση του και την ελάχιστη-μέγιστη απόσταση που μπορεί να μετρήσει.

```
int new_sonar_variable (String name, uint16_t minimum, uint16_t maximum, int8_t x, int8_t y,
int8_t z);
```

Η εισαγωγή νέας μέτρησης μπορεί να γίνει με την παρακάτω συνάρτηση όπου δηλώνοντας την ονομασία του αισθητήρα και τη νέα τιμή μπορεί να γίνει ανανέωση της προηγούμενης.

```
int set_variable_value (String name, float val);
```

Με τον παραπάνω τρόπο ο σχεδιαστής του συστήματος μπορεί να συλλέξει τις μετρήσεις των αισθητήρων, να τις φιλτράρει, να τις μεταποιήσει ανάλογα με τις ανάγκες της εφαρμογής και μετά να γράψει τη μέτρηση του αισθητήρα στο σύστημα.

Εδώ έγινε η πρώτη πτήση του πολυκοπτέρου. Ουσιαστικά δηλώθηκαν 10 μεταβλητές με διαφορετικές συχνότητες δειγματοληψίας στις οποίες συνεχώς ανανεώνονταν με τυχαίες τιμές. Αυτό που έπρεπε να φανεί στη συγκεκριμένη πτήση ήταν ότι το νέο πρωτόκολλο που εισήχθη στον flight controller δεν επηρέαζε την υπάρχουσα λειτουργικότητα του πολυκοπτέρου.

4.3 Αισθητήρες Εγγύτητας – Υπερηχητικοί Αισθητήρες

Δεδομένου ότι το πρωτόκολλο που σχεδιάστηκε στα πλαίσια της παρούσας διπλωματικής, είναι ένα πρωτόκολλο για τη μεταφορά μετρήσεων αισθητήρων οποιοσδήποτε αισθητήρας εγγύτητας μπορεί να χρησιμοποιηθεί στο σύστημα και να μεταφέρονται οι μετρήσεις που

συλλέγει. Οι επιλογές που υπήρχαν στο εμπόριο για αισθητήρες εγγύτητας με αναλογίες μεγέθους που είναι εύκολα ενσωματώσιμες σε πολυκόπτερο ήταν υπερηχητικοί αισθητήρες και αισθητήρες υπεριώδους ακτινοβολίας. Τελικώς επιλέχθηκαν υπερηχητικοί αισθητήρες με βάση τα παρακάτω χαρακτηριστικά :

- Οι υπερηχητικοί αισθητήρες μπορούν να χρησιμοποιηθούν σε εσωτερικούς και εξωτερικούς χώρους χωρίς ιδιαίτερο πρόβλημα. Το μόνο πρόβλημα που παρουσιάζεται είναι σε μετρήσεις απόστασης από αντικείμενα που απορροφούν τον ήχο. Η ακρίβεια της μέτρησης μειώνεται δραστικά σε αυτήν την περίπτωση. Αντιθέτως οι μετρήσεις απόστασης αισθητήρων υπεριώδους ακτινοβολίας επηρεάζονται από το φως, το χρώμα της επιφάνειας που μετράται και τον καπνό. Από τα παραπάνω εξάγεται το συμπέρασμα ότι δεν μπορούν να χρησιμοποιηθούν κατά τη διάρκεια της μέρας, το οποίο είναι και ο μεγαλύτερος περιοριστικός παράγοντας ανάμεσα σε υπερηχητικούς αισθητήρες και αισθητήρες υπεριώδους ακτινοβολίας.
- Οι υπερηχητικοί αισθητήρες εγγύτητας είναι φθηνότεροι από τους αισθητήρες υπεριώδους ακτινοβολίας.

4.3.1 Μέτρηση υψομέτρου με υπερηχητικό αισθητήρα

Η μέτρηση υψομέτρου πριν την προσθήκη του αισθητήρα εγγύτητας γινόταν με μετατροπή της ατμοσφαιρικής πίεσης σε υψόμετρο. Για να βελτιωθεί αυτή η μέτρηση και να μην είναι εύκολα επιρεάσιμη από τον αέρα και το φως, απαιτείται προσθήκη αφρολέξ πάνω από το βαρόμετρο. Η μέτρηση και πάλι επηρεάζεται από τις δονήσεις που δέχεται ο flight controller. Παρακάτω παρατίθεται διάγραμμα μετρήσεων του βαρομέτρου την στιγμή που δόθηκε εντολή στον flight controller να κρατήσει σταθερό υψόμετρο. Η πράσινη γραμμή είναι το επιθυμητό υψόμετρο που θα έπρεπε να πετάει το πολυκόπτερο. Φαίνεται πως όταν ενεργοποιείται η λειτουργία AltHold η πράσινη γραμμή παίρνει μια τιμή η οποία είναι η τιμή του βαρομέτρου τη συγκεκριμένη χρονική στιγμή. Μετά από αυτή τη στιγμή το πολυκόπτερο προσπαθεί να κρατήσει σταθερό υψόμετρο αλλά όπως φαίνεται στο σχήμα η μέτρηση του βαρομέτρου είναι αρκετά κακή και έτσι να μην επέρχεται το επιθυμητό αποτέλεσμα πτήσης.

Σχήμα 4.9 με την πράσινη γραμμή γίνεται το επιθυμητό υψόμετρο ενώ με την κόκκινη το υψόμετρο μέτρησης του βαρομέτρου

Η παραπάνω μέτρηση έγινε σε εξωτερικό χώρο μονώνοντας το βαρόμετρο με αφρολέξ ώστε να μην υπάρχει επίδραση από το φως και τον άνεμο. Το πολυκόπτερο φάνηκε να αποκλίνει από το επιθυμητό υψόμετρο κατά 1m. Αυτό για πτήσεις σε χαμηλά υψόμετρα είναι υπερβολικά επικίνδυνο λόγο της μικρής απόστασης πολυκοπτέρου και εδάφους. Έτσι επιλέχθηκε η εισαγωγή υπερηχητικού αισθητήρα για τη μέτρηση του υψομέτρου. Η απόσταση μέτρησης των υπερηχητικών αισθητήρων περιορίζεται από κάποια άνω και κάτω όρια. Οι HCSR04 που επιλέχθηκαν για την παρούσα διπλωματική έχουν εύρος μέτρησης με ελάχιστο 2cm και μέγιστο το οποίο κυμαίνεται από το μέγεθος του εντοπιζόμενου αντικειμένου. Στο documentation(τεκμηρίωση) του αισθητήρα αναφέρεται ως μέγιστη απόσταση μέτρησης, τα 4m, αλλά δεν αναφέρονται στοιχεία για το μέγεθος του αντικειμένου με το οποίο έγιναν οι μετρήσεις. Ο ίδιος ο αισθητήρας μιας και δεν κάνει κάποιον έλεγχο για τη μέτρησή του αλλά είναι ένας αισθητήρας με trigger και echo, μπορεί να μετρήσει πολύ μεγαλύτερες αποστάσεις δεδομένου ότι το μόνο χρησιμοποιείται μεγαλύτερο αντικείμενο από εκείνο που χρησιμοποιήθηκε για το documentation. Προφανώς όταν το αντικείμενο είναι το έδαφος ο αισθητήρας μπορεί να μετρήσει πολύ μεγαλύτερη απόσταση από τα 4m. Έγιναν δύο μετρήσεις μετά τα 4m. Αυτό οφείλεται κυρίως στην έλλειψη χώρου ώστε να μπορεί να μετρηθεί ταυτόχρονα η απόσταση από δεύτερο όργανο μέτρησης ώστε να πιστοποιηθεί η μετρούμενη απόσταση. Έτσι μετρήθηκε το υψόμετρο από δύο διαδοχικούς ορόφους σε ένα κτήριο όπου το ύψος ήταν γνωστό εκ των προτέρων. Τα μετρούμενα υψόμετρα ήταν 7.5m και 10m με τον αισθητήρα να μην αποκλίνει από την πραγματική μέτρηση περισσότερο των 15cm. Έτσι θεωρήθηκε ως μέγιστη απόσταση μέτρησης τα 10m διότι μέχρι εκεί η μέτρηση του αισθητήρα θεωρήθηκε ότι είχε πολύ μικρότερη απόκλιση από εκείνη του βαρομέτρου, με αποτέλεσμα να έχει νόημα να λαμβάνεται υπόψιν η μέτρηση του υπερηχητικού

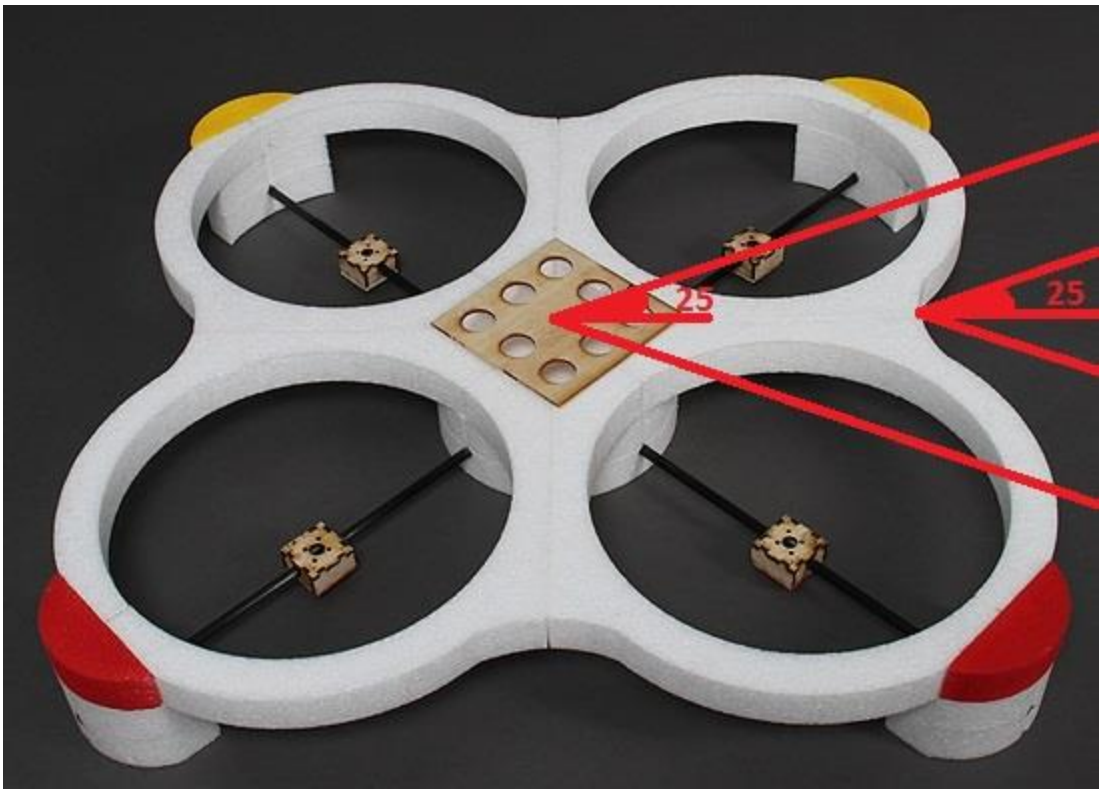
αισθητήρα.

4.3.2 Μέτρηση απόστασης περιμετρικά του πολυκοπτέρου με υπερηχητικούς αισθητήρες

Περιμετρικά του πολυκοπτέρου τοποθετήθηκαν αισθητήρες εγγύτητας με σκοπό να δημιουργηθεί ένα απλό πρωτόκολλο αποφυγής συγκρούσεων.

Αρχικά έγιναν πειραματικές δοκιμές ώστε οι αισθητήρες να τοποθετηθούν όσο πιο κοντά στο κέντρο του πολυκοπτέρου γινόταν. Με αυτόν τον τρόπο θα μεγιστοποιούνταν η εμβέλεια ανίχνευσης των υπερηχητικών αισθητήρων λόγω της αύξησης της αρχικής απόστασης από τα τοιχώματα του πολυκοπτέρου. Όπως φαίνεται και στην παρακάτω εικόνα, όταν ο υπερηχητικός αισθητήρας βρίσκεται κοντά στο κέντρο, ανιχνεύει με μεγαλύτερη εμβέλεια. Η τοποθέτηση των υπερηχητικών αισθητήρων με τέτοιο τρόπο ώστε στο ανιχνευτικό τους πεδίο να βρίσκονται οι προπέλες είναι γενικά μια διαδικασία που αντενδείκνυται λόγω του θορύβου που προκαλείται από την αναταραχή του αέρα (prop-wash) και του ακουστικού θορύβου (prop-noise) από τις προπέλες του πολυκοπτέρου, καθώς και του ηλεκτρομαγνητικού θορύβου που προκαλείται από το τριφασικό ρεύμα που απαιτείται για την κίνηση των μοτέρ (που κινούν τις προπέλες) [16]. Προφανώς κάτι τέτοιο έπρεπε να πιστοποιηθεί πειραματικά. Έτσι δένοντας το τετρακόπτερο στο έδαφος και αυξάνοντας σταδιακά το throttle (γκάζι) του, παρατηρήθηκε η έξοδος του υπερηχητικού αισθητήρα, κρατώντας σταθερά ένα αντικείμενο σε απόσταση 150cm από τον αισθητήρα. Παρακάτω δίνονται τιμές μέτρησης απόστασης που μετράει το πολυκόπτερο καθώς αυξάνεται το throttle. Παρατηρείται ότι όσο αυξάνεται το throttle, μειώνεται αντίστοιχα η απόσταση που μετρά ο αισθητήρας.

Τιμή αισθητήρα (cm)	Throttle (PWM value in μs)
150	1000
100	1134
83	1100
85	1325
77	1478
66	1595
55	1774



Σχήμα 4.10 Τυφλά σημεία ανάλογα με το πού τοποθετείται ο αισθητήρας

Με τα παραπάνω δεδομένα συμπεραίνεται ότι οι υπερηχητικοί αισθητήρες πρέπει να τοποθετηθούν όσο πιο μακριά γίνεται από τις προπέλες του πολυκοπτέρου ώστε να αποφευχθεί η επήρεια του θορύβου που προκαλείται από αυτές.

Έτσι οι αισθητήρες για την αποφυγή συγκρούσεων τοποθετήθηκαν στον εξωσκελετό του πολυκοπτέρου (στο προστατευτικό foam). Με αυτό το τρόπο μειώθηκε πολύ η ανιχνευτική εμβέλεια των αισθητήρων όπως φαίνεται και στην εικόνα 4.11 αλλά εξαφανίστηκε εξ ολοκλήρου η επίδραση του θορύβου από τις προπέλες του πολυκοπτέρου. Ο πίνακας με την αντιπαράθεση throttle-τιμής αισθητήρα δεν παρουσιάζεται εδώ διότι παρατηρήθηκε ότι κατά την αύξηση του throttle, η τιμή μέτρησης του αισθητήρα δεν μεταβάλλεται περισσότερο από 3 εκατοστά.

4.3.3 Μέτρηση απόστασης πάνω από το πολυκόπτερο

Η μέτρηση απόστασης πάνω από το πολυκόπτερο διέπεται από τις ίδιες αρχές με τη μέτρηση υψομέτρου με τη βασική διαφορά ότι εδώ οι αποστάσεις δεν είναι αναγκαίο να είναι ιδιαίτερα μεγάλες μιας και αυτός ο αισθητήρας χρησιμοποιείται για αποφυγή συγκρούσεων και όχι για διατήρηση υψομέτρου από μια επιφάνεια.

4.3.4 Λήψη μετρήσεων υπερηχητικών αισθητήρων από το Arduino

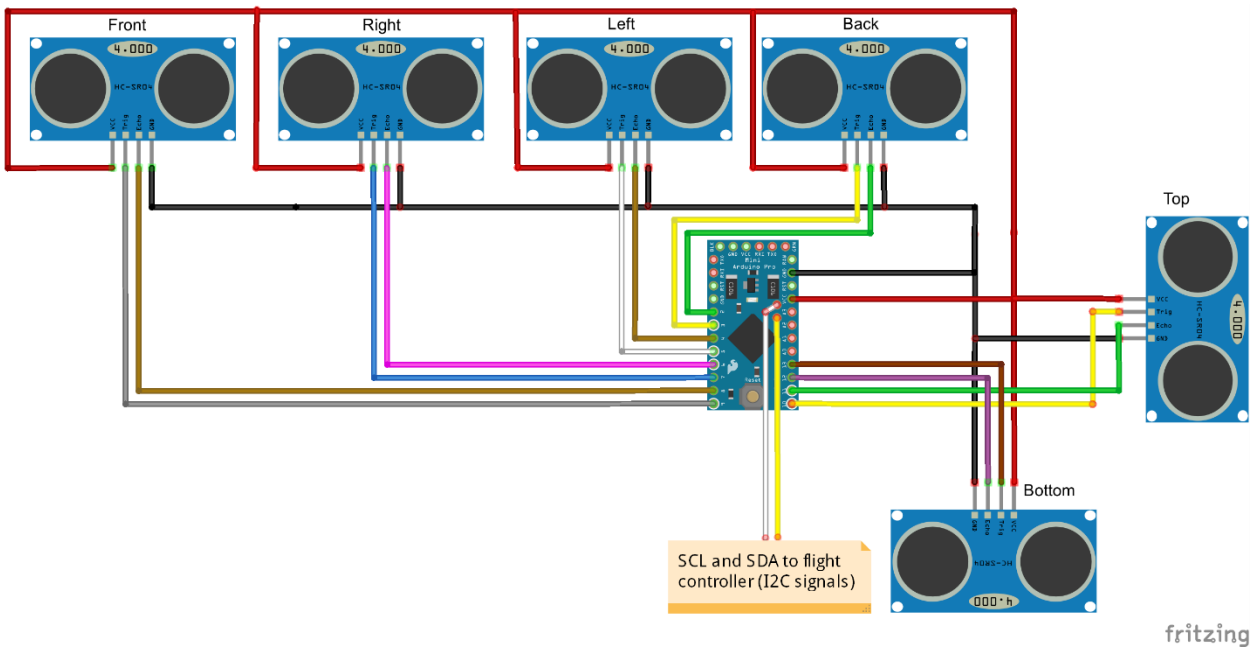
Οι αισθητήρες είναι trigger-echo, δηλαδή έχουν μια είσοδο(trigger) στην οποία πρέπει να σταλεί ένας παλμός διάρκειας τουλάχιστον 10μs. Έτσι από τον transmitter μεταδίδεται ένας υπερηχητικός παλμός συχνότητας 40kHz ο οποίος ανακλάται σε μια επιφάνεια και φτάνει στον δέκτη(receiver). Ο χρόνος μετάδοσης του παλμού στον αέρα μεταφράζεται στο echo με ένα παλμό διάρκειας ίσης με εκείνη που χρειάστηκε για να ταξιδέψει ο παλμός στον αέρα. Οπότε το μόνο που χρειάζεται είναι να σταλεί ένας παλμός διάρκειας 10μs στο trigger pin και να ληφθεί μέτρηση για τη διάρκεια του παλμού στο echo. Ο παλμός στο trigger pin είναι ένας απλός ψηφιακός παλμός, δηλαδή ένα σήμα που ενώ συνεχώς είναι LOW για 10μs γίνεται HIGH. Ομοίως ο παλμός που επιστρέφεται από τον υπερηχητικό αισθητήρα είναι ένα σήμα που από LOW μεταβαίνει σε HIGH και ο χρόνος που μένει σε HIGH είναι ο χρόνος μεταγωγής του σήματος στον αέρα.

Αρχικά έγινε προσπάθεια ώστε η λήψη μέτρησης παλμού στο echo pin να γίνεται ασύγχρονα με εξωτερικά(external) pin interrupts. External pin interrupts υποστηρίζονται μόνο από τα pins 2 και 3 του Arduino. Παρόλα αυτά σε όλα τα ports ενός ATmega328p όπως αυτός του Arduino mini υποστηρίζονται port change interrupts. Το port αποτελείται από 8 pins ενός μικροελεγκτή. Τα port change interrupts υποστηρίζονται ανά port, οπότε και η ανίχνευση γίνεται επίσης ανά port. Η εύρεση του pin που άλλαξε σε ένα port είναι ευθύνη του προγραμματιστή. Οπότε χρειαζόταν στο interrupt service routine να ανιχνευτεί ποιο pin του port έχει μεταβληθεί. Αυτό έγινε κρατώντας σε μια μεταβλητή την τελευταία κατάσταση του port. Κάθε φορά που καλείται το interrupt service routine λόγο μιας αλλαγής στο port συγκρίνεται η τελευταία κατάσταση του port και η τωρινή. Με αυτό τον τρόπο εντοπίζεται το pin στο οποίο προέκυψε η μεταβολή.

Έτσι αρχικά συνδέθηκε ένας αισθητήρας στο Arduino και επιχειρήθηκε να ληφθούν μετρήσεις. Για να εκκινηθεί η διαδικασία μέτρησης στον αισθητήρα χρησιμοποιήθηκε ένας timer. Στον timer ενεργοποιήθηκε interrupt για ένα συγκριτή (compare interrupt) και το interrupt υπερχείλισης (overflow interrupt). Ο συγκριτής ρυθμίστηκε με τέτοιο τρόπο ώστε να ενεργοποιηθεί το interrupt 10μs πριν ο timer εκδώσει overflow interrupt. Με αυτό τον τρόπο στο compare interrupt ενεργοποιείται το trigger pin σε HIGH ενώ στο overflow interrupt το trigger pin γίνεται LOW. Έτσι γίνεται η διαχείριση του trigger pin.

Για να επιτευχθεί η διαχείριση του trigger pin χρησιμοποιήθηκε ο timer0 (8 bit timer) του

ATmega328p του Arduino. Επειδή στα 16MHz ο timer παρήγαγε overflow ανά 16μs και ο χρόνος ανάμεσα σε 2 διαδοχικά counts του timer ήταν 0.625μs, το οποίο είναι πολύ μικρότερο από την απαιτούμενη ακρίβεια, κρίθηκε σκόπιμο να χρησιμοποιηθεί ο prescaler του timer. Η ακρίβεια που είναι ανεκτή για αυτή τη σχεδίαση είναι μια μεταβολή στην έξοδο του timer ανά ένα χρονικό διάστημα κοντά σε μικρό-second προκειμένου να προσδιοριστεί αυτή η τιμή του timer στον συγκριτή και να προκαλείται το compare interrupt 10μs πριν το overflow interrupt. Οι επιλογές για τον prescaler στους timer του ATmega328p είναι 8 και 64 και αντιπροσωπεύουν τους διαιρέτες της συχνότητας ρολογιού του μικροελεγκτή. Λειτουργώντας λοιπόν τον timer με prescaler 8, ο χρόνος ανάμεσα σε 2 διαδοχικά counts του timer ήταν 0.5μs (2MHz) ενώ με prescaler 64, ο χρόνος ανάμεσα σε 2 διαδοχικά counts του timer ήταν 4ms (250kHz) το οποίο προφανώς είναι πολύ αργό για την επιθυμητή ακρίβεια. Έτσι επιλέχθηκε prescaler 8. Το πρόβλημα με τον συγκεκριμένο prescaler είναι ότι ο timer θα κάνει overflow ανά 128μs (256*0.5μs) το οποίο είναι πολύ συχνό για τους υπερηχητικούς αισθητήρες αυτής της σχεδίασης. Η επιθυμητή συχνότητα δειγματοληψίας του συστήματος αυτού είναι 10Hz για κάθε αισθητήρα καθώς αυτή είναι η απαιτούμενη συχνότητα δειγματοληψίας για το βαρόμετρο του flight controller ώστε να παίρνει μέτρηση υψομέτρου. Για να επιτευχθεί αυτή η συχνότητα δειγματοληψίας από τον αισθητήρα εισήχθη μια μεταβλητή μέτρησης στο σύστημα, η οποία μετρά τα overflows. Όταν τα overflows ξεπεράσουν έναν αριθμό, τότε και μόνο τότε γίνεται ένα trigger στον αισθητήρα. Για να επιτευχθεί συχνότητα δειγματοληψίας 10Hz(100ms περίοδο), ο αριθμός αυτός πρέπει να είναι 100ms/128μs = 781. Άρα ανά 781 overflows γίνεται ένα trigger στον αισθητήρα για να ληφθεί μια μέτρηση.



Σχήμα 4.11 Συνδεσμολογία υπερηχητικών αισθητήρων στο Arduino

Το echo pin κάποια στιγμή θα γίνει HIGH και θα ενεργοποιηθεί το port change interrupt service routine. Τότε θα σωθεί η τιμή ρολογιού του Arduino σε μια μεταβλητή. Όταν το echo pin γίνει LOW θα ξαναενεργοποιηθεί το port change interrupt service routine και θα σωθεί και πάλι η τιμή του ρολογιού. Από την διαφορά των δύο τιμών μπορεί να υπολογιστεί ο χρόνος μεταγωγής του σήματος στον αέρα.

Στο datasheet του αισθητήρα αναφέρεται ότι είναι γραμμικός. Κάτι τέτοιο θεωρήθηκε καλό να επαληθευτεί στην πράξη. Έτσι μπήκε ένα εμπόδιο σε απόσταση ενός μέτρου από τον αισθητήρα και έγινε λήψη μέτρησης από τον ίδιο. Έτσι υπολογίστηκε ένας συντελεστής k με τον οποίο πρέπει να διαιρούνται οι χρονικές διάρκειες για να υπολογίζονται οι αποστάσεις.

(Χρονική διάρκεια) / k = (Απόσταση)

Έτσι υπολογίστηκε το k. Μετά μετακινήθηκε το αντικείμενο στα 2, 3 και 4 μέτρα και παρατηρήθηκε μια μέγιστη απόκλιση 5cm στα 3 και 4 μέτρα με αποτέλεσμα να θεωρηθεί αμελητέα. Έτσι ο γραμμικός συντελεστής k θεωρήθηκε έγκυρος σαν υπολογιστική μέθοδος και πιστοποιήθηκε ότι ο ίδιος ο αισθητήρας είναι γραμμικός.

Για να εισαχθούν και οι υπόλοιποι υπερηχητικοί αισθητήρες στο σύστημα αυτό που έγινε αρχικά ήταν να στέλνονται όλοι οι trigger παλμοί την ίδια στιγμή με τον ίδιο τρόπο που αναλύθηκε

παραπάνω. Αυτό δεν είχε το επιθυμητό αποτέλεσμα καθώς πολλά echo interrupts έπρεπε να κληθούν την ίδια χρονική στιγμή. Παρατηρήθηκε ότι πολλές μετρήσεις ήταν εσφαλμένες όταν το πολυκόπτερο βρισκόταν σε κλειστούς χώρους όπου λόγο ανακλάσεων των υπερήχων στις επιφάνειες αλλά και λόγο του φόρτου που αυξήθηκε από τα interrupts τα οποία έφταναν την ίδια χρονική στιγμή. Έτσι αποφασίστηκε να διαιρεθούν τα 781 overflows και να στέλνεται ένας παλμός trigger ανά 781/6=130 overflows σε έναν υπερηχητικό αισθητήρα. Οι αισθητήρες δέχονται παλμούς trigger κυκλικά με αλγόριθμο round robin. Με αυτό τον τρόπο έχει γίνει λήψη μιας μέτρησης από έναν αισθητήρα μέχρι να ενεργοποιηθεί ο επόμενος παλμός trigger με αποτέλεσμα να μην υπάρχει ούτε το φαινόμενο των ανακλάσεων που υπήρχε πριν αλλά ούτε και το φαινόμενο των echo pin interrupts να συμβαίνουν την ίδια χρονική στιγμή.

Όταν πιστοποιήθηκε ότι όλοι οι αισθητήρες λειτουργούν και μετρούν σωστά τιμές από τα αντικείμενα που κινούνταν γύρο από το πολυκόπτερο, εισήχθησαν οι συναρτήσεις που δηλώνουν τους υπερηχητικούς αισθητήρες στο πολυκόπτερο και μετά την λήψη μιας μέτρησης να γίνεται ανανέωση της τιμής εκείνης της μέτρησης στο πρωτόκολλο επικοινωνίας με το πολυκόπτερο.

Εκεί έγινε μια πτήση με το πολυκόπτερο προκειμένου να πιστοποιηθεί η ομαλή λειτουργία του. Σε αυτό το σημείο συναντήθηκε και το πρώτο πρόβλημα της υλοποίησης. Όταν το πολυκόπτερο απομακρύνθηκε ελάχιστα από το έδαφος τότε παρατηρήθηκε τελείως ασταθής συμπεριφορά. Φαινόταν δια γυμνού οφθαλμού ότι το πολυκόπτερο δεν μπορούσε να κρατήσει εύκολα την οριζόντια θέση του. Η πρώτη εκτίμηση για το πρόβλημα ήταν ότι λόγο της μεγάλης συχνότητας των interrupts που συνέβαιναν στο σύστημα από τους υπερηχητικούς αισθητήρες σε συνδυασμό με εκείνα του I2C, ο φόρτος του Arduino ανέβαινε πολύ περισσότερο από εκείνο που μπορούσε να εξυπηρετήσει, με αποτέλεσμα τα interrupts να συσσωρεύονται και ο flight controller να περιμένει απαντήσεις για ιδιαίτερα μεγάλα χρονικά διαστήματα λόγο της φύσης του I2C σαν πρωτόκολλο. Ο flight controller όταν περιμένει περισσότερο από ένα συγκεκριμένο χρονικό διάστημα για απάντηση από κάποιο slave τότε σταματά την επικοινωνία. Αν αυτό συμβαίνει επαναληπτικά όμως οι χρόνοι αναμονής του flight controller μεγαλώνουν πολύ με αποτέλεσμα να παρατηρείται η αστάθεια αυτή στο πολυκόπτερο.

Αλλάζοντας εξ’ ολοκλήρου την τεχνική ανάγνωσης των υπερηχητικών αισθητήρων, εφαρμόστηκε τεχνική polling για τη μέτρηση του παλμού echo και τα triggers στέλνονταν με σειριακό κώδικα και όχι με interrupts σε timer. Ουσιαστικά η τεχνική round robin δεν μεταβλήθηκε αλλά μεταβλήθηκε ο τρόπος αποστολής παλμού trigger και ο τρόπος μέτρησης του

παλμού echo.

Με αυτό τον τρόπο εξαλείφτηκαν τα όποια προβλήματα συναντήθηκαν με τον αυξημένο αριθμό interrupts στο Arduino αφού, εκτός της επικοινωνίας με τον flight controller, τα πάντα γίνονται σειριακά. Ξανακάνοντας πτήση δεν παρατηρήθηκε καμία αστάθεια αυτή τη φορά στο πολυκόπτερο με αποτέλεσμα να θεωρηθεί ότι πράγματι η αστάθεια ήταν αποτέλεσμα καθυστέρησης του Arduino.

4.3.5 Επέκταση λειτουργικοτήτων πολυκοπτέρου

Προσθήκη υπερηχητικού αισθητήρα κάτω από το πολυκόπτερο

Η προσθήκη αισθητήρα στο κάτω σημείο του πολυκοπτέρου βοήθησε στην ανάπτυξη λειτουργειών που με τη μέτρηση του βαρομέτρου δεν μπορούσαν σα γίνουν. Λόγω του ότι η μέτρηση του υπερηχητικού αισθητήρα εξαρτάται από την πραγματική απόσταση από το έδαφος και όχι από μια απόσταση που εκτιμάται μέσω του ανέμου, όταν η γεωμορφολογία του εδάφους αλλάζει, τότε αλλάζει και το υψόμετρο στο οποίο πετάει το πολυκόπτερο. Κάτι τέτοιο δεν ήταν εφικτό με τη μετατροπή της ατμοσφαιρικής πίεσης σε υψόμετρο. Οπότε εμμέσως εισήχθη στο πολυκόπτερο μια λειτουργία για terrain following σε χαμηλά υψόμετρα.

Η αλλαγή που έγινε ώστε να μπορεί το πολυκόπτερο να αναγνωρίσει τον υπερηχητικό αισθητήρα ήταν να εισαχθεί μια βιβλιοθήκη στο firmware του ώστε όταν δηλωθεί από το Arduino ότι έχει συνδεθεί ένας αισθητήρας για μέτρηση υψομέτρου, να γίνεται η ανάγνωση της μέτρησης από εκείνη τη βιβλιοθήκη και όχι από κάποια από εκείνες που υποστηρίζει ο flight controller για τους υπερηχητικούς αισθητήρες που υποστηρίζει ο ίδιος. Η ανάγνωση της μέτρησης του αισθητήρα σε αυτή τη βιβλιοθήκη γίνεται ακριβώς όπως θα γινόταν για οποιοδήποτε άλλο αισθητήρα του Arduino.

Προσθήκη υπερηχητικού αισθητήρα πάνω από το πολυκόπτερο

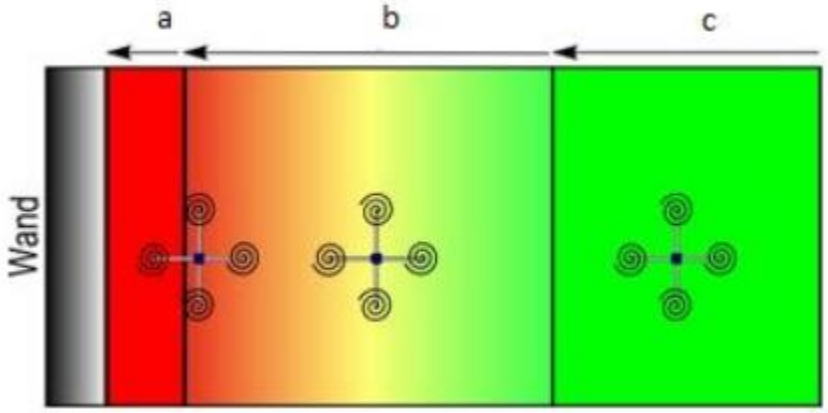
Με την προσθήκη υπερηχητικού αισθητήρα στο πάνω μέρος του πολυκοπτέρου δίνεται πλέον η δυνατότητα στο πολυκόπτερο να αποφεύγει εμπόδια τα οποία βρίσκονται από πάνω του. Αυτό ρυθμίστηκε να γίνεται μόνο σε modes στα οποία το πολυκόπτερο πρέπει να κρατήσει σταθερό υψόμετρο. Σε αυτά τα modes αν το πολυκόπτερο συναντήσει εμπόδιο από πάνω του θα προσπαθήσει να μην το πλησιάσει σε απόσταση μικρότερη των 50cm. Για να επιτευχθεί αυτή η απόσταση ο παρακάτω έλεγχος γίνεται κάθε φορά που ο flight controller επανεκτιμά το υψόμετρο στο οποίο πρέπει να παραμείνει. Αν όμως μειωθεί η απόσταση και στον αισθητήρα που μετρά το

υψόμετρο σε επίπεδα κάτω των 50cm τότε το πολυκόπτερο θα προσπαθήσει να μείνει στο κέντρο της απόστασης των δύο εμποδίων.

```
if(top_sonar->distance<=50 && bottom_sonar->distance>50){
        controller_desired_alt = controller_desired_alt - (50-top_sonar->distance);
else if(top_sonar->distance<=50 && bottom_sonar->distance<=50){
        controller_desired_alt = (top_sonar->distance + bottom_sonar->distance)/2;
```

4.3.5.1 Προσθήκη διάταξης υπερηχητικών αισθητήρων περιμετρικά του πολυκοπτέρου

Με αυτό τον τρόπο όταν το πολυκόπτερο πλοηγείτε σε ένα χώρο και συναντά εμπόδια επιχειρεί να τα αποφύγει. Στο παρακάτω σχήμα η απόσταση a ορίστηκε ως 50cm ενώ η b ως 150cm. Με αυτές τις ρυθμίσεις έγινε πτήση σε εξωτερικό χώρο με άνεμο ο οποίος έσπρωχνε το πολυκόπτερο προς ένα εμπόδιο(τοίχο). Παρατηρήθηκε ότι όταν το πολυκόπτερο πλησίαζε τον τοίχο ξεκινούσε να κινείται προς την αντίθετη κατεύθυνση και δεν πλησίασε ποτέ σε απόσταση μικρότερη του a οπότε δεν ενεργοποιήθηκε ποτέ μεγαλύτερο «γκάζι» από 600.



Σχήμα 4.12 Σχεδίαση αποφυγής συγκρούσεων με κλίμακες

```
if(front_sonar->distance<=50)
        control_pitch += 900;
else if(front_sonar->distance<=150)
        control_pitch += 600;
```

Όπως φαίνεται στην παραπάνω σχέση όταν το πολυκόπτερο κινείται προς ένα εμπόδιο και βρεθεί σε απόσταση μικρότερη του (a+b) τότε προστίθεται μια τιμή στο «γκάζι» του pitch. Είναι άξιο αναφοράς ότι η τιμή προστίθεται την υπάρχουσα τιμή του pitch αλλά δεν την πανωγράφει.

Αυτό βοήθησε πολύ στην πιστοποίηση λειτουργίας του συστήματος αφού ο χειριστής του πολυκοπτέρου μπορούσε κινώντας λίγο περισσότερο τα sticks του τηλεχειρισμού να πάρει το έλεγχο του συστήματος και να αποφευχθούν πολλές πτώσεις και συγκρούσεις με αντικείμενα. Παρακάτω δίνεται ο ψευδοκώδικας για την αντίδραση του πολυκοπτέρου όταν ένα αντικείμενο βρεθεί αντιδιαμετρικά του πρώτου. Οπότε όταν υπάρχει ένα αντικείμενο μπροστά στο πολυκόπτερο και ένα πίσω του σε αποστάσεις μικρότερες των 150 cm και τα δύο τότε το πολυκόπτερο θα προσπαθήσει να μείνει σε μια απόσταση περίπου στο κέντρο των δύο αντικειμένων αφού θα προστεθούν 600 στο control_pitch λόγο του μπροστινού αντικειμένου αλλά θα αφαιρεθούν λόγο του πίσω με αποτέλεσμα το πολυκόπτερο να μην κινηθεί στον άξονα pitch λόγο των εμποδίων.

```
if(back_sonar->distance<=50)
        control_pitch -= 900;
else if(back_sonar->distance<=150)
        control_pitch -= 600;
```

4.4 Μετάδοση μετρήσεων από τον flight controller προς το σταθμό βάσης

Για τη μετάδοση οποιασδήποτε πληροφορίας από το σταθμό βάσης προς τον flight controller και το αντίστροφο, χρησιμοποιούνται τα 3DR Radios που αναφέρονται στο κεφάλαιο 2. Αυτό που κάνουν τα συγκεκριμένα radios είναι να συνδέονται με 2 διαφορετικές συσκευές και να λειτουργούν σαν δίαυλος επικοινωνίας μεταξύ των 2. Κάθε radio επικοινωνεί με τη συσκευή που είναι συνδεμένο στα 57600 bps και μεταδίδει/λαμβάνει οποιαδήποτε πληροφορία διαδίδεται μέσω της σειριακής επικοινωνίας. Έτσι δύο συσκευές μπορούν μέσω σειριακής θύρας να επικοινωνήσουν ασύρματα. Το ένα radio συνδέεται στον flight controller ενώ το άλλο στη συσκευή που θα επικοινωνήσει με τον flight controller. Προφανώς η επικοινωνία μπορεί να γίνει και χωρίς τα 2 radios αν συνδεθούν ενσύρματα οι 2 συσκευές μέσω σειριακής θύρας στα 57600 bps.

Για να επιτευχθεί η επικοινωνία των δύο συσκευών με πακέτα μηνυμάτων έχει εισαχθεί ένα πρωτόκολλο επικοινωνίας για τη μεταφορά των δεδομένων, το MAVlink. Το MAVlink υποστηρίζει τη χρήση πακέτων για τη μεταφορά των δεδομένων και των εντολών που πρέπει να μεταφερθούν από τη μια συσκευή στην άλλη.

4.4.1 Χρήση δύο πακέτων

Η αρχική προσέγγιση για να μεταδοθούν τα δεδομένα ήταν να κατασκευαστούν δύο πακέτα όπου το ένα να αποτελείται από τα ονόματα των μεταβλητών και να μεταδίδεται μόνο στην εκκίνηση του flight controller. Με αυτό το πακέτο στο σταθμό βάσης εμφανίζονται οι ονομασίες των μεταβλητών, με σκοπό να είναι εύκολο στο χρήστη να αντιστοιχίσει τη μέτρηση του αισθητήρα με την ονομασία του. Το δεύτερο πακέτο προφανώς περιέχει τις μετρήσεις των αισθητήρων.

Όταν εκκινείται ο flight controller και θεωρήσει ότι ο συγχρονισμός με το Arduino έχει ολοκληρωθεί, ξεκινά να μεταδίδει τις ονομασίες των μεταβλητών. Επειδή ο flight controller λειτουργεί με scheduler και ρουτίνες που καλούνται με συγκεκριμένη συχνότητα, τα πακέτα επιλέχθηκε να στέλνονται με τον συχνότερο δυνατό τρόπο που προσφέρεται από τον scheduler, δηλαδή 100Hz.

Αρχικά έγινε πτήση προσπαθώντας να εντοπιστεί ο μέγιστος αριθμός πακέτων ονομάτων που μπορεί να σταλεί ανά 100Hz. Δοκιμάστηκε για ένα, δύο, τρία και τέσσερα πακέτα να γίνει πτήση και να φανεί η συμπεριφορά του αεροσκάφους. Παρατηρήθηκε ότι η αποστολή τριών και τεσσάρων πακέτων ονομάτων σε ένα κύκλο (από εκείνους των 100Hz) αύξανε το workload του flight controller υπερβολικά και του προκαλούσε αστάθειες κατά την πτήση. Οπότε εξήχθη το συμπέρασμα ότι το πολύ δύο πακέτα μπορούν να σταλούν ανά 100Hz.

Ως απόρροια των παραπάνω ένα πακέτο ονόματος στέλνεται ανά 100Hz και μέχρι να ολοκληρωθεί η αποστολή όλων των ονομάτων των μεταβλητών κανένα άλλο πακέτο δεν στέλνεται.

Τα πακέτα μεταβλητών ξεκινάνε να στέλνονται με ρυθμό εκείνο της μεταβλητής, με μεγαλύτερο ρυθμό δειγματοληψίας. Οι υπόλοιπες μεταβλητές απλά επαναμεταδίδουν τις μετρήσεις που υπάρχουν στον πίνακα μεταβλητών.

Στο δέκτη χρησιμοποιήθηκε ο open source σταθμός βάσης APM Planner με τον οποίο μπορούν να παρακολουθούνται οι μετρήσεις του πολυκοπτέρου. Σε αυτόν έγιναν όλες οι αλλαγές προκειμένου να είναι σε θέση να διαβάζει τα επιπλέον πακέτα που στέλνει ο flight controller και περιέχουν τα δεδομένα τηλεμετρίας του Arduino. Κάθε φορά που ένα πακέτο ονομάτων φτάνει στο σταθμό βάσης, ο σταθμός αποθηκεύει τα ονόματα σε ένα πίνακα με συμβολοσειρές. Κάθε φορά που ένα πακέτο μνημάτων φτάνει στο σταθμό βάσης εκείνος διαγράφει οτιδήποτε είναι αποθηκευμένο στην μεταβλητή ονόματος του πακέτου και γράφει στη θέση του οτιδήποτε έχει

σωθεί γι’ αυτή τη μεταβλητή στον πίνακα ονομάτων.

Για να γίνει αυτό περισσότερο κατανοητό θα δοθεί παρακάτω ένα παράδειγμα για να φανεί ο τρόπος λειτουργίας και κατασκευής του. Έστω ότι το ο flight controller θέλει να μεταδώσει τρείς μεταβλητές με ονόματα, temperature, humidity και pressure. Όταν ο σταθμός βάσης εκκινηθεί θα γεμίσει ένα πίνακα με παύλες (-) οι οποίες θα εμφανιστούν και στην οθόνη του. Εκεί θα φαίνονται οι τιμές τηλεμετρίας του Arduino να είναι μηδέν και οι ονομασίες με παύλες (-) ώστε ο χρήστης να αντιλαμβάνεται ότι καμία μέτρηση δεν έχει ληφθεί ακόμη από το Arduino. Η αρχική μορφή στην οθόνη λοιπόν είναι η παρακάτω.

- 0
- 0
- 0
- 0

….

Όταν ένα πακέτο φτάσει στο σταθμό βάσης διαβάζεται και αναδιαμορφώνεται από δομή C σε δομή Qt ώστε να εμφανιστεί στην οθόνη. Για το πακέτο ονομάτων αυτό δεν συμβαίνει. Το πακέτο εντοπίζεται και αντί να τυπωθεί όπως όλα τα υπόλοιπα πακέτα MAVlink, διαβάζονται οι τιμές του και εισάγονται σε έναν static πίνακα με συμβολοσειρές, ο οποίος δεν είναι άλλος από τον προαναφερθέντα πίνακα αποθήκευσης ονομάτων μεταβλητών. Τα παιδία του πακέτου μετρήσεων έχουν την ακόλουθη μορφή :

```
<field type="float" name="val1">value 1</field>
```

Η δομή του ίδιου πεδίου MAVlink που θα πάει να διαβαστεί και να μετατραπεί σε δομή Qt ώστε να εξαχθεί στην οθόνη έχει την ακόλουθη μορφή.

```
typedef struct __mavlink_field_info {
    const char *name;               // name of this field
    const char *print_format;       // printing format hint, or NULL
    mavlink_message_type_t type;    // type of this field
    unsigned int array_length;      // if non-zero, field is an array
    unsigned int wire_offset;       // offset of each field in the payload
    unsigned int structure_offset;  // offset in a C structure
} mavlink_field_info_t;
```

Άρα η μεταβλητή name στην παραπάνω δομή θα έχει την τιμή ‘val1’. Παρατηρείται ότι η μεταβλητή είναι δείκτης σε συμβολοσειρά οπότε μπορεί να ελευθερωθεί και στη συνέχεια να δεσμευτεί χώρος για την ονομασία που είναι αποθηκευμένη στον πίνακα μεταβλητών για αυτό το

field. Έτσι η ονομασία val1 μπορεί να πανωγραφεί από την ονομασία temperature. Οπότε στην οθόνη του Apm Planner αντί να εμφανιστεί

Val1 34
Val2 25
Val3 3
- 0

θα εμφανιστεί :

temperature 34
humidity 25
pressure 3
- 0

4.4.2 Χρήση ενός πακέτου

Η δεύτερη σχεδίαση που δοκιμάστηκε στα πλαίσια αυτής της διπλωματικής είχε σκοπό τη μείωση της αποστολής των μετρήσεων οι οποίες δεν είχαν ανανεωθεί. Όπως αναφέρθηκε και στην μοντελοποίηση για τη μεταφορά των δεδομένων μέσω τηλεμετρίας, για να επιτευχθεί η συγκεκριμένη υλοποίηση πρέπει να εισαχθούν αναγνωριστικά IDs τα οποία γράφονται στη μεταβλητή type του πακέτου. Όμοια με παραπάνω όταν ολοκληρωθεί ο συγχρονισμός ονομάτων με το Arduino, ο flight controller ξεκινά να στέλνει τα ονόματα των μεταβλητών στο σταθμό βάσης.

Λόγο του γεγονότος ότι πάνω από 2 πακέτα μεγέθους 64 bytes αυξάνουν υπερβολικά το workload του flight controller, θα γίνει αλλαγή της αρχικής σχεδίασης του πακέτου.

Αρχικά οι τιμές του type από 0 ως 7 δεσμεύονταν για να μεταβιβάζονται οι ονομασίες των μεταβλητών. Το ονόματα των μεταβλητών έχουν οριστεί να είναι το πολύ 20 χαρακτήρων οπότε σε ένα πακέτο χωράνε 3 ονόματα μεταβλητών. Τα ονόματα εισάγονται στο πακέτο, στο όρισμα data, το ένα ακολουθούμενο από το επόμενο με διαχωρισμό ενός χαρακτήρα ‘\n’ μεταξύ τους. Ο συνολικός αριθμός μεταβλητών που μπορούν να σταλούν προς το σταθμό βάσης είναι 21. Ο σταθμός βάσης λαμβάνει τα μηνύματα και τα εισάγει στον πίνακα ονομάτων των μεταβλητών σε θέσεις που ορίζονται από το type και τη θέση τους στο πακέτο. Στο πακέτο με type=0, η πρώτη συμβολοσειρά αντιστοιχίζεται στη θέση 0 του πίνακα ονομάτων μεταβλητών, η δεύτερη συμβολοσειρά αντιστοιχίζεται στη θέση 1 του πίνακα ονομάτων μεταβλητών κοκ. Προφανώς στο πακέτο με type=1, η πρώτη συμβολοσειρά εισάγεται τη θέση 3, η δεύτερη στην 4 κοκ.

Οι τιμές του type 8 και 9 δεσμεύονται με σκοπό τη μετάδοση των μετρήσεων. Κάθε μεταβλητή μεταδίδεται με την αρίθμησή της στον πίνακα μεταβλητών και την τελευταία τιμή της. Το πακέτο που μεταδίδεται στην πρώτη θέση(θέση 0) του πίνακα data έχει την θέση την μεταβλητής στον πίνακα δεδομένων και στις 4 επόμενες την τιμή της μεταβλητής, στην επόμενη θέση(θέση 5) θα περιέχει τη θέση της επόμενης μεταβλητής και στις επόμενες 4 τη μέτρηση που μεταδίδεται για αυτήν κοκ.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	Value 0				1	Value 1				2	Value 2				3	Value 3				…	

Με αυτό τον τρόπο μπορεί εύκολα να γίνει αντιστοίχιση των μετρήσεων στις ονομασίες των μεταβλητών τους.

Ο λόγος των παραπάνω αλλαγών είναι ο περιορισμός των 2 πακέτων ανά «κύκλο scheduler». Στα 2 πακέτα αυτής της σχεδίασης χωράνε 22 (2x11) μετρήσεις και γι’ αυτό τα types περιορίστηκαν σε 0-7(8x3=24). Παρατηρείται ότι αυτό είναι ο μεγαλύτερος περιοριστικός παράγοντας της υλοποίησης και μάλλον οι μετρήσεις που ένας σχεδιαστής θα είναι σε θέση να εισάγει στο σύστημα θα είναι 22.

Η παρουσίαση των δεδομένων από τον APM Planner εδώ είναι δυσκολότερη από ότι στην προηγούμενη περίπτωση αφού δεν υπάρχει πακέτο με την μορφή που θα ήταν επιθυμητή ώστε τα δεδομένα να μπορούν να παρουσιαστούν στο χρήστη, δηλαδή δεν υπάρχουν δομές με στοιχεία name και value όπως πριν. Αυτές οι δομές κατασκευάστηκαν. Δηλαδή όπως και πριν, όταν ένα πακέτο περιέχει ονομασίες, δηλαδή έχει type 0-7, τότε εκείνο δεν εκτυπώνεται από το APM Planner. Απλά τα ονόματα των μεταβλητών αποθηκεύονται σε έναν πίνακα. Εδώ αποθηκεύονται και οι μετρήσεις όταν αυτές φτάσουν με ένα πακέτο. Για να μπορέσουν τα δεδομένα να εκτυπωθούν στην οθόνη κατασκευάζονται οι δομές C όπως ακριβώς θα εξάγονταν από το MAVlink για το πακέτο Telemetry Values. Με αυτόν τον τρόπο γράφοντας στη δομή αυτή τις τιμές των ονομάτων και των μεταβλητών, εκείνες μπόρεσαν να εξαχθούν στην οθόνη μέσω του APM Planner.

5 Δοκιμές και Πιστοποίηση λειτουργίας

Επειδή η φύση των πολυκοπτέρων είναι τέτοια που τα ατυχήματα μπορεί να είναι πολλά λόγο των πολλών παραγόντων που πρέπει να ληφθούν κάθε φορά που κάτι νέο εισάγεται στο πολυκόπτερο αποφασίστηκε να γραφτεί ένα μικρό κεφάλαιο με τίτλο δοκιμές και πιστοποίηση λειτουργείας που να πραγματεύεται με τον τρόπο που έγιναν οι δοκιμές για να πιστοποιηθεί η λειτουργία του εκάστοτε συστήματος. Το κεφάλαιο κυρίως πραγματεύεται τον τρόπο με τον οποίο έγινε η προετοιμασία του πολυκοπτέρου, για να πιστοποιηθεί η λειτουργία του κάθε νέου συστήματος εν πτήση, και η πιστοποίηση λειτουργίας του ίδιου του νέου συστήματος.

5.1 Πιστοποίηση λειτουργίας συστήματος συλλογής μετρήσεων

Το σύστημα συλλογής μετρήσεων δεν παρουσίασε ιδιαίτερα προβλήματα κατά την πιστοποίηση λειτουργίας του εν πτήση καθώς δεν είναι κάτι που επηρεάζει τη λειτουργικότητα του flight controller. Ο flight controller λειτουργεί απλά σαν μεσάζοντας ανάμεσα στο Arduino και το σταθμό βάσης. Η πιστοποίηση λειτουργίας του συστήματος έγινε ελέγχοντας κάθε διεπαφή ξεχωριστά.

Αρχικά ελέγχθηκε η εισαγωγή και μετάδοση δεδομένων στο κανάλι του I2C και κατά πόσο ο flight controller λάμβανε σωστές μετρήσεις. Έτσι γράφτηκε μία μεταβλητή στο Arduino της οποίας ο ρυθμός δειγματοληψίας ήταν 100Hz και η τιμή της αυξανόταν κατά ένα σε κάθε ενημέρωση της τιμής της. Στο flight controller, εκτυπώνοντας τις τιμές που διάβαζε μέσω I2C στη σειριακή θύρα, παρατηρούνταν οι μεταβολές της τιμής της μέτρησης σε ένα τερματικό.

Στο επόμενο στάδιο ελέγχθηκε κατά πόσο οι μετρήσεις μεταδίδονται στο σταθμό βάσης. Ανοίγοντας το σταθμό βάσης και συνδέοντας τα radio της 3DR στον flight controller και σε έναν υπολογιστή ελέγχθηκε κατά πόσο οι μετρήσεις μεταδίδονται στο σταθμό βάσης. Αφού πιστοποιήθηκαν τα παραπάνω έγινε μία πτήση με το πολυκόπτερο προκειμένου να είναι βέβαιο ότι δεν έχει αυξηθεί το workload του flight controller και δεν επηρεάζεται η λειτουργικότητά του σε πτήση.

Αφού βεβαιώθηκε ότι όλα λειτούργησαν όπως έπρεπε, το επόμενο βήμα ήταν να βρεθεί ο μέγιστος αριθμός μεταβλητών που μπορούσαν να μεταφερθούν μέσω του καναλιού χωρίς να επηρεάζεται η λειτουργικότητα του flight controller εν πτήση και οι συχνότητες με τις οποίες μπορούσε κάτι τέτοιο να συμβεί. Αυξάνοντας σταδιακά τον αριθμό των μεταβλητών που εισάγονται στο Arduino και ελέγχοντας κατά πόσο ο flight controller είναι ακόμα λειτουργικός,

βρέθηκε ότι με παραπάνω των 15 μεταβλητών σε συχνότητα δειγματοληψίας 100Hz χάνονταν τιμές. Αυτό που γινόταν είναι να διαβάζονται οι τιμές και να εξάγονται μέσω της σειριακής θύρας στο τερματικό. Εκεί παρατηρήθηκε ότι οι τελευταίες μεταβλητές δεν ανανεώνουν τις τιμές τους. Ο λόγος που συνέβαινε αυτό ήταν ότι ο schedulertou flight controller έχει ένα προκαθορισμένο χρόνο τον οποίο θα τρέξει την κάθε ρουτίνα. Αυτός ο χρόνος είναι προκαθορισμένος και αν για οποιοδήποτε λόγο πάει να ξεπεραστεί το πολυκόπτερο δεν θα προλάβει να εκτελέσει σε καίριο χρόνο άλλες κύριες λειτουργίες του. Έτσι ο scheduler μπορεί να σταματήσει την εκτέλεση μιας ρουτίνας όταν τελειώσει το ποσοστό χρόνου που της αναλογεί. Επειδή εδώ το πρωτόκολλο με το οποίο επικοινωνεί ο flight controller με το Arduino είναι το I2C υπάρχει περίπτωση να σταματήσει μια μετάδοση στη μέση και το κανάλι να μείνει ανοιχτό χωρίς end condition στο Arduino. Αυτό είναι ιδιαίτερα επικίνδυνο διότι στο κανάλι υπάρχουν και άλλοι αισθητήρες με τους οποίου δεν θα καταφέρει να επικοινωνήσει ο flight controller. Έτσι δεν θα πάρει μετρήσεις οι οποίες είναι απαραίτητες για τη λειτουργία του πχ. Μέτρηση βαρομέτρου και θα υπάρξει είτε λάθος εκτίμηση σε μέτρηση ή ακόμα χειρότερα να κολλήσει σε ένα σημείο ο κώδικας περιμένοντας να πάρει τον έλεγχο του καναλιού και να πέσει το πολυκόπτερο.

Παρατηρήθηκε ότι για 10 μεταβλητές δεν υπήρχε πρόβλημα στη μετάδοση των μετρήσεων με αποτέλεσμα να θεωρηθεί ότι ο μέγιστος αριθμός μεταβλητών με συχνότητα ανανέωσης μέτρησης τα 100 Hz. Αυτό δεν σημαίνει ότι ο μέγιστος αριθμός μεταβλητών που μπορεί να εισαχθούν στο σύστημα είναι 10. Σημαίνει ότι ανά 10ms μπορεί να δειγματοληπτήσει 10. Όταν ο ρυθμός δειγματοληψίας είναι 10Hz τότε αν γίνει καλή κατανομή λήψεων των μετρήσεων σε διαφορετικά χρονικά διαστήματα, τότε στο σύστημα μπορούν να εισαχθούν μέχρι και 100 μετρήσεις. Εδώ αποφασίστηκε να κατανέμονται οι μεταβλητές που δεν έχουν ρυθμό δειγματοληψίας 100Hz σε διαφορετικούς «κύκλους» του scheduler.

Στη μετάδοση πακέτων μέσω MAVlink, όταν ο scheduler σταματήσει την εκτέλεση μιας ρουτίνας δεν υπάρχει περίπτωση να δημιουργηθούν προβλήματα στη λειτουργία του πολυκοπτέρου και πτώσεις. Απλά οι μετρήσεις δεν μεταδίδονται μέσω σειριακής στο 3DR radio και το πακέτο χάνεται. Επειδή η μετάδοση ενός πακέτου MAVlink είναι μια χρονοβόρα διαδικασία, παρατηρήθηκε ότι μόνο δύο πακέτα μπορούν να μεταδοθούν σε ένα “κύκλο” του scheduler. Αυτό πρακτικά σημαίνει ότι με την πρώτη υλοποίηση όπου στέλνονται όλες οι μετρήσεις ακόμα και όταν δεν έχουν ανανεωθεί μπορούν να σταλούν μέχρι 28(2x14 μετρήσεις ανά πακέτο) μετρήσεις ανά «κύκλο» ενώ στην δεύτερη όπου εισάγεται η θέση του πίνακα και

μετά η μέτρηση, μπορούν να σταλούν 22 (2x11 μετρήσεις ανά πακέτο). Αφού το πακέτο χωράει 11 ή 14 μετρήσεις ανάλογα με την υλοποίηση και το bottleneck της σχεδίασης ήταν το I2C κανάλι αποφασίστηκε να χρησιμοποιηθεί η δεύτερη σχεδίαση πακέτου για το MAVlink, διότι δεν στέλνει περιττή πληροφορία και σε περίπτωση που στο μέλλον αποφασιστεί η κατασκευή κάποιου πολυκοπτέρου ειδικού σκοπού που δεν χρησιμοποιεί όλο το μέγεθος του πακέτου για τα δεδομένα που στέλνει, αφήνεται χώρος στο πακέτο για δεδομένα που δεν ανήκουν στις μετρήσεις του Arduino.

Για να πιστοποιηθεί η λειτουργία των παραπάνω έγιναν δύο πτήσεις. Μια συνδέοντας στο Arduino έναν BMP085(Θερμοκρασία, Ατμοσφαιρική πίεση) με ρυθμό ανανέωσης 20Hz, έναν DHT11 (Θερμοκρασία, Υγρασία) με ρυθμό ανανέωσης 5Hz, ένα HC-SR04 που απλά έπαιρνε μετρήσεις με ρυθμό 20Hz και δηλώνοντας 5 μεταβλητές με ρυθμό ανανέωσης 100Hz που απλά έγραφαν τυχαίες τιμές σε κάθε ανανέωση. Μετά έγινε μια δεύτερη πτήση στην οποία και οι 10 μεταβλητές ήταν τυχαίες μετρήσεις συχνότητας 100Hz. Αφού πιστοποιήθηκε η ομαλή λειτουργία του πολυκοπτέρου εν πτήση και στις δύο περιπτώσεις η σχεδίαση θεωρήθηκε επιτυχής.

5.2 Χρήση υπερηχητικών αισθητήρων

5.2.1 Προσθήκη υπερηχητικού αισθητήρα για μέτρηση υψομέτρου

Για να πιστοποιηθεί η λειτουργία του υπερηχητικού αισθητήρα για τη μέτρηση του υψομέτρου αρχικά παρατηρήθηκε η μέτρηση του υψομέτρου χωρίς την εισαγωγή του αισθητήρα. Όπως έχει ήδη αναφερθεί παραπάνω ο αισθητήρας του βαρομέτρου είναι ευαίσθητος στις μεταβολές του ανέμου. Έτσι αφαιρέθηκε η μόνωσή του ώστε να αυξηθεί η ευαισθησία του αυτή. Παρατηρήθηκε ότι προκαλώντας ροή αέρα πάνω από το βαρόμετρο η μέτρηση του υψομέτρου μεταβάλλεται. Στη συνέχεια προστέθηκε ο υπερηχητικός αισθητήρας σύμφωνα με τη σχεδίαση που έχει αναφερθεί και επανελέγχθηκε η μέτρηση του υψομέτρου για να πιστοποιηθεί ότι τώρα το υψόμετρο που μετράται είναι εκείνο που μετρά ο υπερηχητικός αισθητήρας και όχι εκείνο που μετράται λόγο του βαρομέτρου. Παρατηρήθηκε ότι προκαλώντας ροή αέρα πάνω από το βαρόμετρο η μέτρηση του υψομέτρου παρέμεινε σταθερή. Όταν χωρίς να ενεργοποιήσουμε τα μοτέρ του πολυκοπτέρου αλλά κινώντας το πάνω κάτω η μέτρηση που έβλεπε το υψόμετρο ήταν εκείνη του υπερηχητικού αισθητήρα.

Αποφασίστηκε ότι όλα έβαιναν καλώς και εκεί έγινε η πτήση όπου το πολυκόπτερο σε λειτουργία alt hold κράτησε σταθερό υψόμετρο χωρίς να αλλάξει ύψος πάνω από 2cm. Μετά

ξεκίνησε ο πειραματισμός για να δούμε τις αντιδράσεις του πολυκοπτέρου. Αρχικά πετώντας το περίπου στο 1.5 μέτρο και θέτοντας το σε λειτουργία alt hold παρατηρούνταν το πολυκόπτερο να μένει σταθερό. Όταν κάτω από το πολυκόπτερο έμπαινε απότομα ένα τραπέζι ύψους 50cm, παρατηρούνταν το πολυκόπτερο να παίρνει κι εκείνο απότομα ύψος και να φτάνει στα 2m από το έδαφος, προσπαθώντας να κρατήσει σταθερή τη μέτρηση του υψομέτρου του. Μετά κατασκευάστηκε μία τεχνητή ξύλινη ράμπα, τέθηκε το πολυκόπτερο σε alt hold, και δόθηκε εντολή να κινηθεί το πολυκόπτερο εμπρός. Εμπρός βρισκόταν η ράμπα. Το πολυκόπτερο προσπαθώντας να κρατήσει σταθερό υψόμετρο άρχισε να ανεβαίνει τη ράμπα. Παρακάτω φαίνεται μια εικόνα με το πολυκόπτερο να ανεβαίνει τη ράμπα. Στη συνέχεια και βγαίνοντας σε εξωτερικό χώρο τέθηκε και πάλι σε alt hold το πολυκόπτερο και κινώντας το προς τα εμπρός επιχειρήθηκε να κάνει το πολυκόπτερο ακριβώς την ίδια κίνηση με πριν με τη διαφορά ότι αντί για ράμπα τώρα το πολυκόπτερο ανέβηκε σκαλοπάτια.

Σχήμα 5.1 Μια ράμπα εμφανίζεται στο κάτω μέρος του πολυκοπτέρου τη στιγμή που βρίσκεται σε alt hold και κινείται προς τα εμπρός. Το πολυκόπτερο μεταβάλλει το ύψος του ακολουθώντας τη γεωμορφολογία του εδάφους.

Αξίζει να σημειωθεί ότι όταν το πολυκόπτερο προσπάθησε να ανέβει τη ράμπα έχοντας την παραπάνω μορφή, άρχισε να αυξάνει το υψόμετρό του προσπαθώντας να μείνει σε σταθερό ύψος από το υποτιθέμενο έδαφος. Όταν τα ξύλα γυρίστηκαν από το κάτω μέρος τότε το πολυκόπτερο δεν αύξησε το ύψος του όπως αναμενόταν. Αυτό συνέβη διότι το ξύλο από το κάτω μέρος είναι πορώδες πράγμα που δεν το έκανε ιδιαίτερα ανακλαστικό στον ήχο. Έτσι ο υπέρηχος του αισθητήρα δεν ανακλάται με αποτέλεσμα να χαλάει η μέτρηση του και να κάνει το πολυκόπτερο ασταθές αφού ο αισθητήρας «βλέπει» πολύ θόρυβο. Στην πράξη φαίνεται ότι όταν το έδαφος δεν είναι ιδανικό για τη χρήση υπερηχητικού αισθητήρα τότε το πολυκόπτερο δεν είναι

σε θέση να υπολογίσει σωστά το ύψος του διότι κακή μέτρηση στον υπερηχητικό αισθητήρα κάνει το πολυκόπτερο να ακούει πότε τη μέτρηση του αισθητήρα(γιατί δεν είναι όλες κακές) και πότε εκείνη του βαρομέτρου. Η συνεχής απότομη μεταβολή μεταξύ των δύο μπορεί να κάνει το πολυκόπτερο ασταθές αφού προσπαθεί να κρατήσει σταθερό υψόμετρο μεταξύ δύο τιμών που «παίζουν» πολύ μεταξύ τους.

5.2.2 Προσθήκη διάταξης υπερηχητικών αισθητήρων περιμετρικά και πάνω από το πολυκόπτερο

Η διάταξη που εισήχθη στο πολυκόπτερο σαν έχει σκοπό την αποφυγή συγκρούσεων. Για να πιστοποιηθεί η λειτουργία της περιμετρικής διάταξης πριν την πτήση, παρατηρήθηκαν οι μετρήσεις των αισθητήρων αλλά και των μετρήσεων των καναλιών του πολυκοπτέρου. Όταν αντικείμενα κινούνταν γύρω από το πολυκόπτερο παρατηρήθηκε αυτό που έπρεπε, δηλαδή τα κανάλια του χειριστηρίου να παίρνουν τιμές ώστε το πολυκόπτερο να προσπαθεί να αποφύγει το εμπόδιο. Αρχικά δίνοντας μικρά «γκάζια» και στη συνέχεια μεγαλύτερα στο pitch και το roll παρατηρήθηκε η αποφυγή συγκρούσεων. Αρχικά για μικρές αποστάσεις αντίδρασης σε εσωτερικούς χώρους και στη συνέχεια σε εξωτερικούς πλησιάζοντας το πολυκόπτερο σε τοίχους. Λόγο του υλικού από το οποίο είναι φτιαγμένο το πολυκόπτερο που έγιναν οι δοκιμές δεν παρατηρήθηκαν φαινόμενα όπως εσφαλμένες μετρήσεις λόγο κραδασμών, λόγο ηλεκτρομαγνητικού θορύβου ή θορύβου από τις προπέλες του πολυκοπτέρου(μερικά από τα προβλήματα που αναφέρονται στην ιστοσελίδα της Maxbotix [16]). Αυτό δεν σημαίνει ότι τα προβλήματα αυτά δεν θα εμφανιστούν σε άλλα πολυκόπτερα. Εξάλλου προβλήματα τέτοιου τύπου ήταν η αίτια οι υπερηχητικοί αισθητήρες να μετακινηθούν εξωτερικά από την επίδραση των προπελών.

Για τον πάνω αισθητήρα ακολουθήθηκε η ίδια ακριβώς διαδικασία με τη διαφορά ότι τώρα δεν παρατηρήθηκε το throttle όπως θα περίμενε κανείς αλλά το επιθυμητό υψόμετρο που θέλει να κρατήσει το πολυκόπτερο όταν βρίσκεται σε λειτουργία alt hold. Όταν λοιπόν το πολυκόπτερο μπήκε σε λειτουργία alt hold και παρατηρούνταν η τιμή του επιθυμητού υψομέτρου ενώ κάτι πλησίαζε το πολυκόπτερο από πάνω αυτή η τιμή μειώθηκε ανάλογα. Τότε έγινε και πτήση με την οποία πλησίαζαν αντικείμενα από πάνω το πολυκόπτερο με σκοπό να παρατηρηθεί η μεταβολή του υψομέτρου του εν πτήση.

Επειδή οι αισθητήρες είναι υπερηχητικοί, συναντάται το πρόβλημα που ήταν αναμενόμενο

ότι θα συναντηθεί. Όταν οι επιφάνειες δεν είναι ανακλαστικές στον ήχο(π.χ. μάλλινα ρούχα), τότε ο υπερηχητικός αισθητήρας δεν δίνει αξιόπιστες μετρήσεις με αποτέλεσμα να μην είναι αξιόπιστη και η αντίδραση του πολυκοπτέρου.

6 Έλεγχος πολυκοπτέρου από ενσωματωμένο σύστημα αυξημένης υπολογιστικής ισχύος

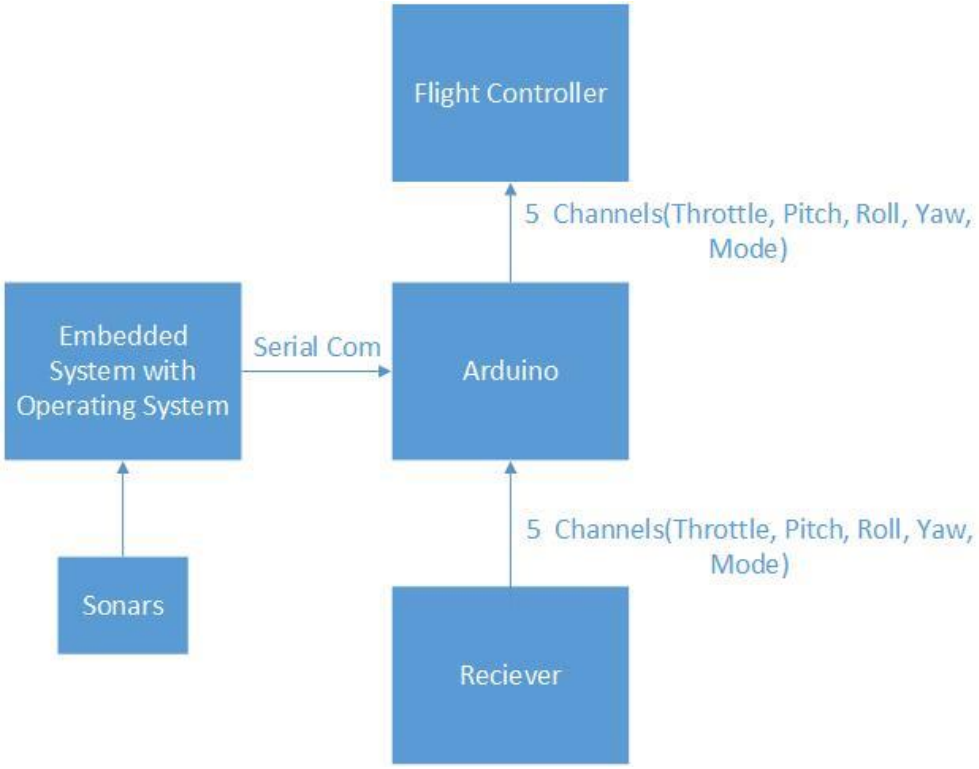
Για να μπορεί το πολυκόπτερο να εκμεταλλευτεί την υπολογιστική ισχύ συστημάτων όπως Raspberry Pi, Gumstix ή Odroid υλοποιήθηκε ένα σύστημα το οποίο να είναι σε θέση να δώσει τον έλεγχο του πολυκοπτέρου σε μια συσκευή αυξημένης υπολογιστικής ισχύος όπως οι παραπάνω.

6.1 Μοντελοποίηση και Αρχιτεκτονική Συστήματος

Τα κανάλια του δέκτη του χειριστηρίου και το εξωτερικό σύστημα συνδέονται με ένα Arduino και το Arduino συνδέεται με τον flight controller εξάγοντας τα ίδια ακριβώς κανάλια που έχει και ως εισόδους από τον δέκτη του χειριστηρίου. Το εξωτερικό σύστημα για να συνδεθεί με το Arduino χρησιμοποιεί σειριακή επικοινωνία. Το Arduino διαβάζει τα σήματα του χειριστηρίου και τα εξάγει αυτούσια προς το flight controller ενώ ανά τακτά χρονικά διαστήματα (100ms) ενημερώνει το εξωτερικό σύστημα για τις τιμές των καναλιών του δέκτη μέσω σειριακής θύρας. Έτσι το εξωτερικό σύστημα μπορεί να διαβάζει τις εισόδους του χειριστηρίου μέσω σειριακής επικοινωνίας. Αν το εξωτερικό σύστημα για κάποιο λόγο αποφασίσει να πάρει τον έλεγχο του πολυκοπτέρου και να το κινήσει μέσα σε ένα χώρο, μπορεί να στείλει εντολή στο Arduino ότι τα σήματα που θα εξάγει δεν θα είναι πλέον οι στάθμες του δέκτη αλλά εκείνες που στέλνονται από τη σειριακή θύρα.

Η αρχιτεκτονική που προτείνεται για την διαχείριση των αισθητήρων εγγύτητας και των σημάτων του χειριστηρίου παρατίθεται στο σχήμα 3.9 που φαίνεται παρακάτω όπου πρακτικά το εξωτερικό σύστημα μπορεί να δώσει εντολές στο πολυκόπτερο να κινηθεί. Οι αισθητήρες εγγύτητας εισήχθησαν ώστε να γίνει πιστοποίηση λειτουργίας του συστήματος και όχι ως αυτοσκοπός κάποιας εφαρμογής.

Το εξωτερικό σύστημα λειτουργεί εκτελώντας ένα ατέρμον βρόχο που σε κάθε επανάληψη κάνει μια μέτρηση για έναν υπερηχητικό αισθητήρα. Επειδή δεν ήταν αυτοσκοπός της εφαρμογής η ενσωμάτωση υπερηχητικών αισθητήρων στο εξωτερικό σύστημα δεν επιχειρήθηκαν μέθοδοι βελτιστοποίησης της ανάγνωσης των τιμών του αισθητήρα με interrupt handling. Για την Επίσης μπορεί να διαβάζει τα σήματα του δέκτη από το Arduino και έχει τη δυνατότητα να στείλει τιμές στο Arduino ώστε να πάρει τον έλεγχο και να κινήσει το πολυκόπτερο.



Σχήμα 6.1 Ένα Arduino συνδέεται ανάμεσα στον flight controller και το δέκτη του χειριστηρίου με σκοπό τη διαχείριση των σημάτων του δέκτη

6.2 Υλοποίηση Συστήματος

Σε αυτή την υλοποίηση επιχειρήθηκε η εισαγωγή ενός ενσωματωμένου συστήματος στο πολυκόπτερο το οποίο σκοπό έχει να παίρνει τον έλεγχο και να προσφέρει κάποιου είδους αυτονομία όταν κάτι τέτοιο είναι επιθυμητό.

Για να γίνει το παραπάνω ένα Arduino(external system) εισήχθη στο σύστημα που σκοπό είχε να διαβάζει τα σήματα δέκτη τηλεχειρισμού και να τα μεταδίδει στον flight controller. Ταυτοχρόνως ανά τακτά χρονικά(100ms) διαστήματα έχει την ευθύνη να μεταδίδει τα σήματα σε μια τρίτη συσκευή μέσω σειριακής θύρας. Η τρίτη συσκευή μπορεί όποτε κρίνει εκείνη απαραίτητο να στείλει εντολή στο Arduino ότι επιθυμεί να αναλάβει τον έλεγχο του πολυκοπτέρου. Από εκείνη τη στιγμή το τρίτο σύστημα έχει τον έλεγχο και στέλνει τιμές με τις οποίες ελέγχει τις κινήσεις του πολυκοπτέρου. Αν για οποιοδήποτε λόγο ο χειριστής του πολυκοπτέρου κινήσει κάποιο από τα sticks του χειριστηρίου αυτομάτως ξαναπαίρνει τον έλεγχο του συστήματος.

Με αυτό τον τρόπο ο χειριστής μπορεί να σχεδιάζει συστήματα τα οποία χρησιμοποιούν

τεχνητή νοημοσύνη και παίρνουν αποφάσεις για τις κινήσεις του πολυκοπτέρου αλλά αν κάτι δεν πάει καλά κατά την εκτέλεση του αλγορίθμου, μπορεί να ξαναπάρει τον έλεγχο οποιαδήποτε στιγμή.

6.2.1 Χειρισμός σημάτων PWM από το Arduino

Για να μετρηθούν τα σήματα του δέκτη του χειριστηρίου από το Arduino θεωρήθηκε καλή τεχνική να διαβαστεί ο κώδικας με τον οποίο ο flight controller διαβάζει τα σήματα του δέκτη. Το λογισμικό αυτός ήταν ήδη υλοποιημένο για την επιθυμητή σχεδίαση. Αυτό που το MPNG firmware έκανε για να διαβάσει τα σήματα του δέκτη δεν διέφερε πολύ από τον τρόπο με τον οποίο γινόταν για την ανάγνωση των σημάτων echo των υπερηχητικών αισθητήρων. Και το MPNG firmware χρησιμοποιεί port change interrupts και ανιχνεύει ποιο pin του port εκείνου μεταβλήθηκε. Έτσι για ένα pin, όταν το σήμα γίνεται HIGH λαμβάνεται μια μέτρηση για την τιμή του ρολογιού του μικροελεγκτή, ενώ ακόμη μία μέτρηση λαμβάνεται όταν το σήμα γίνεται LOW. Από τη διαφορά των δύο χρονικών στιγμών εξάγεται ο χρόνος παλμού του PWM σήματος του δέκτη. Η ίδια ακριβώς τεχνική ακολουθήθηκε και για την ανάγνωση των σημάτων από το Arduino.

Για να εξαχθούν τα σήματα του δέκτη του χειριστηρίου στον flight controller χρησιμοποιήθηκε η βιβλιοθήκη του Arduino, Servo η οποία χρησιμοποιεί τον timer 1 του ATmega 328 με σκοπό να εξάγει παλμούς PWM. Η βιβλιοθήκη αυτή θέτοντας τον prescaler του timer 1 σε 8 και χρησιμοποιώντας compare interrupt για να μεταβάλλει το σήμα εξόδου από HIGH σε LOW μπορεί να εξάγει σήμα PWM. Η τιμή του PWM δίνεται από τον προγραμματιστή σε microseconds και η Servo αναλαμβάνει όλη τη διαδικασία υπολογισμού της τιμής του συγκριτή ώστε με τον prescaler που η ίδια έχει θέσει να μπορεί να μεταβάλλει το σήμα του παλμού από HIGH σε LOW το κατάλληλο microsecond ώστε ο παλμός που εξάγεται να είναι ο επιθυμητός από τον προγραμματιστή.

6.2.2 Χειρισμός σειριακής θύρας από το Arduino

Το Arduino ανά 100ms στέλνει τις πιο πρόσφατες τιμές που έχει διαβάσει από τον δέκτη του χειριστηρίου στο τρίτο σύστημα το οποίο ακόμα δεν έχει πάρει τον έλεγχο. Για να υπολογίσει αυτή τη χρονική στιγμή, σε κάθε εκτέλεση της loop λαμβάνει μια μέτρηση για το χρόνο εκτέλεσης του προγράμματος. Αν η τελευταία στιγμή που στάλθηκαν μετρήσεις για τις τιμές του χειριστηρίου δεν διαφέρει περισσότερο από 100ms από την τωρινή στιγμή τότε το πρόγραμμα

συνεχίζει την εκτέλεσή του κανονικά, διαφορετικά στέλνει τις στάθμες του χειριστηρίου μέσω σειριακής θύρας και ανανεώνει ως τελευταία στιγμή που στάλθηκαν τα δεδομένα του χειριστηρίου μέσω σειριακής την τωρινή.

Η λήψη δεδομένων από τη σειριακή θύρα θα μπορούσε να γίνεται με ένα interrupt. Το πρόβλημα που παρουσιάζεται με τη χρήση interrupt είναι ότι ενεργοποιείται κάθε φορά που φτάνει ένα byte στη σειριακή θύρα. Ενα ολόκληρο πακέτο που στέλνεται από τη σειριακή θύρα έχει μέγεθος 6 bytes (το γιατί διαβάζονται 6 bytes εξηγείται παρακάτω). Αυτό σημαίνει 5 από τα 6 interrupts θα επιστρέφουν αμέσως επειδή δεν έχουν φτάσει όλα τα δεδομένα από τη σειριακή θύρα. Έτσι αποφασίστηκε σε κάθε εκτέλεση της loop να ελέγχεται αν έχουν φτάσει 6 bytes από τη σειριακή θύρα. Αν αυτά έχουν φτάσει και κάποιο από τα κανάλια του χειριστηρίου κινείται πλέον από το εξωτερικό σύστημα, τότε ένα flag θα ενεργοποιηθεί και οι τιμές που στέλνονται μέσω της Servo δεν είναι πια εκείνες που διαβάστηκαν από το δέκτη του χειριστηρίου αλλά εκείνες που διαβάστηκαν από το εξωτερικό σύστημα μέσω σειριακής θύρας.

6.2.3 Χειρισμός σειριακής θύρας από το εξωτερικό ενσωματωμένο σύστημα που συνδέεται στο Arduino

Για να είναι εύκολη η αποστολή και λήψη των τιμών των σημάτων από ένα σύστημα που έχει συνδεθεί μέσω σειριακής θύρας με το Arduino, κατασκευάστηκε μια βιβλιοθήκη σε python που να δέχεται απλά τις τιμές που επιθυμεί να στείλει το εξωτερικό σύστημα στο Arduino. Η βιβλιοθήκη γράφτηκε σε python για να μπορεί να συνδεθεί στο Arduino, οποιοδήποτε ενσωματωμένο σύστημα, ανεξαρτήτως λειτουργικού συστήματος. Η βιβλιοθήκη αυτό που κάνει είναι να συνδέεται με το Arduino όταν αρχικοποιείται (init_pilot_control_intarface). Στη συνέχεια εκκινεί ένα νήμα και εκείνο είναι υπεύθυνο για την ανάγνωση των τιμών που στέλνονται από το Arduino στο σύστημα σχετικά με τα σήματα που φτάνουν από το δέκτη του χειριστηρίου. Αυτό που κάνει το νήμα είναι ανά 100ms να ελέγχει αν έφτασαν 6 bytes από τη σειριακή θύρα. Αν έχουν φτάσει τα διαβάζει, ανανεώνει τις τιμές των σημάτων που αποθηκεύονται για το δέκτη και μετά κάνει yield διαφορετικά κάνει απλά yield. Αυτές οι τιμές αποθηκεύονται σε μεταβλητές και μπορούν οποιαδήποτε στιγμή να διαβαστούν από ένα πρόγραμμα που εκτελείται. Σε περίπτωση που το σύστημα θελήσει να πάρει τον έλεγχο του πολυκοπτέρου το μόνο που έχει να κάνει είναι να καλέσει μια συνάρτηση και να δηλώσει ποια κανάλια του χειριστηρίου θέλει να διαχειρίζεται.

6.2.4 Ανάλυση πληροφορίας που μεταδίδεται μέσω σειριακής θύρας

Για να μειωθεί ο όγκος πληροφορίας που ανταλλάσσεται μεταξύ των δύο συσκευών παρατηρήθηκαν οι τιμές των PWM καναλιών που πρέπει να στέλνονται. Το Arduino πρέπει να στέλνει τις τιμές των καναλιών του δέκτη. Οι τιμές αυτές κυμαίνονται από 1000 ως 1900 και αναφέρονται σε microseconds(microseconds που διαρκεί ένας παλμός). Άρα οι τιμές που στέλνονται από το Arduino μπορούν να έχουν εύρος από 0 ως 900 το οποίο απαιτεί 10 bit πληροφορίας για κάθε κανάλι(αφού με 9 bit μπορούν να σταλούν μέχρι 512 διαφορετικές τιμές ενώ με 10 bit μπορούν να σταλούν μέχρι 1024). Τα κανάλια πληροφορίας είναι 5 άρα συνολικά πρέπει να σταλούν 50 bits ανά ενημέρωση του εξωτερικού συστήματος. Αυτό δεν είναι απόλυτα αλήθεια αφού το 5^{ο} κανάλι είναι το mode και οι flight controllers υποστηρίζουν το πολύ έξι διαφορετικά modes. Άρα το 5^{ο} κανάλι αντί να αναπαρίσταται σαν τιμή παλμού σε μικρό-seconds μπορεί να αναπαρασταθεί με αρίθμηση από 1 ως 6 ανάλογα με το εύρος που βρίσκεται η τιμή στο κανάλι 5. Οπότε για το κανάλι 5 απαιτούνται το πολύ 3 bit με αποτέλεσμα να πέφτει ο αριθμός των bit από 50 σε 43(6 bytes).

Για να υπολογιστούν οι στάθμες του PWM που ο flight controller αλλάζει mode, παρατηρήθηκαν οι στάθμες εκείνες που αλλάζοντας τα modes στο χειριστήριο αλλάζει και το mode στον flight controller. Έτσι ορίστηκαν/αποθηκεύτηκαν στο Arduino οι στάθμες στις οποίες ο flight controller αλλάζει τιμή. Έτσι δίνεται η ευχέρεια στο Arduino να γνωρίζει το mode στο οποίο βρίσκεται ο flight controller απλά γνωρίζοντας τη στάθμη του PWM καναλιού.

Για τη μεταφορά των δεδομένων μέσω της σειριακής θύρας , για το σύστημα που τρέχει την python, ισχύει κάτι παρόμοιο με το Arduino, με τη διαφορά ότι εισάγονται 5 bit ενεργοποίησης(enable), ένα για κάθε κανάλι, ώστε όταν το σύστημα θελήσει να πάρει τον έλεγχο του πολυκοπτέρου, το Arduino να αφήσει οποιαδήποτε στάθμη δεν είναι ενεργοποιημένη από το εξωτερικό σύστημα να παίρνει τιμή, από την τρέχουσα τιμή του χειριστηρίου, ενώ όποια είναι ενεργοποιημένη, να διαβάζεται από το μήνυμα του εξωτερικού συστήματος. Όταν κάποιο από όλα τα sticks του χειριστηρίου βέβαια κινηθεί, τότε ο χειριστής αποκτά έλεγχο όλων των καναλιών και όχι μόνο εκείνου που κίνησε.

6.3 Πιστοποίηση λειτουργίας

Για να πιστοποιηθεί η λειτουργία του συστήματος σε πτήση συνδέθηκε ένας υπερηχητικός αισθητήρας μπροστά στο πολυκόπτερο και γράφτηκε ένα πρόγραμμα το οποίο να παίρνει το

έλεγχο του συστήματος όταν εκείνο μπει σε λειτουργία alt hold. Το πρόγραμμα κάνοντας polling παρακολουθεί το κανάλι mode του χειριστηρίου και περιμένει εκείνο να πάρει την τιμή 6(το 6^{ο} mode είχε δηλωθεί ως alt hold). Όταν ο χειριστής έκανε το mode alt hold, το ενσωματωμένο σύστημα ξεκινούσε να κινεί το πολυκόπτερο μπροστά ενεργοποιώντας το κανάλι pitch και δίνοντάς του τιμή 1400 (1470 ήταν η μέση για το συγκεκριμένο χειριστήριο. Για τιμές κάτω από αυτή κινείται μπροστά ενώ για τιμές πάνω από αυτή κινείται πίσω). Από εκείνη τη στιγμή το ενσωματωμένο σύστημα διαβάζει με polling συνεχώς τις μετρήσεις του υπερηχητικού αισθητήρα και περιμένει το πολυκόπτερο να πλησιάσει σε ένα εμπόδιο. Όταν αυτό συμβεί, μεταβάλλει την τιμή του σήματος σε 1550 ώστε να κινηθεί προς τα πίσω και να αποφύγει τη σύγκρουση. Όταν το εμπόδιο έχει απομακρυνθεί αρκετά, τότε δίνεται εντολή να αλλάξει το mode σε land και να προσγειωθεί το πολυκόπτερο.

Αρχικά έπρεπε να πιστοποιηθεί η πτήση του συστήματος όταν τα σήματα βγαίνουν από το Arduino και όχι απευθείας από το δέκτη. Έτσι κινώντας τα sticks του χειριστηρίου και εκτυπώνοντας τις μετρήσεις του flight controller για τα σήματα εισόδου πιστοποιήθηκε ότι οι τιμές του χειριστηρίου εξάγονται σωστά στο flight controller. Μετά κάνοντας πιστοποιήθηκε η ορθή λειτουργία του Arduino και του πολυκοπτέρου.

Στη συνέχεια δίνοντας εντολή στο πολυκόπτερο να κινηθεί ευθεία χωρίς να πετάει παρατηρήθηκε το σήμα εισόδου του flight controller και ελέγχθηκε κατά πόσο όλες οι κινήσεις που γίνονται στο χειριστήριο δίνουν πίσω τον έλεγχο στο χειριστή. Όταν τα sticks του χειριστηρίου κινούνται οι τιμές που διαβάζονται στον από το flight controller είναι εκείνες του χειριστηρίου με αποτέλεσμα να εξάγεται το συμπέρασμα το Arduino λειτουργεί σωστά.

Το πρόγραμμα σχεδιάστηκε με τέτοιο τρόπο ώστε να κάνει polling διαβάζοντας συνεχώς το κανάλι mode και να παίρνει τον έλεγχο του πολυκοπτέρου όταν ο χειριστής δώσει εντολή για alt hold. Εκεί δίνεται εντολή από το ενσωματωμένο σύστημα να κινήσει το πολυκόπτερο μπροστά δίνοντας στο σήμα pitch τιμή 1400. Πράγματι παρατηρήθηκε εν πτήση ότι όταν το πολυκόπτερο μπει σε λειτουργία alt hold αρχίζει να κινείται μπροστά μέχρι ο χειριστής να του πάρει τον έλεγχο.

Εδώ συνδέθηκε και ο υπερηχητικός αισθητήρας στο ενσωματωμένο σύστημα. Όταν πλησιάζει ένα αντικείμενο τον υπερηχητικό αισθητήρα το πρόγραμμα θα έπρεπε να πάρει τον έλεγχο και πάλι να πειράξει το σήμα pitch ώστε το πολυκόπτερο να κινηθεί πίσω. Όταν το αντικείμενο απομακρυνθεί, το σήμα θα έπρεπε να επανέλθει στην αρχική του τιμή και στη συνέχεια το ενσωματωμένο σύστημα να τεθεί σε mode land. Προφανώς τα παραπάνω

παρατηρήθηκαν εκ νέου μέσω σειριακής θύρας, εκτυπώνοντας τα σήματα εισόδου του flight controller. Αρχικά και πάλι το πολυκόπτερο τέθηκε σε alt hold και παρατηρήθηκε ότι το pitch έγινε 1400. Εκεί πλησίασε ένα αντικείμενο τον υπερηχητικό αισθητήρα και παρατηρήθηκε το σήμα της εισόδου του flight controller να γίνεται 1550. Όταν απομακρύνθηκε ξανά το αντικείμενο το πολυκόπτερο τέθηκε σε mode land. Έτσι ακριβώς συμπεριφέρθηκε και σε δοκιμαστικές πτήσεις προσέγγισης κάθετων τοίχων. Όταν το πολυκόπτερο μπήκε σε alt hold mode πλησίασε προς τον τοίχο, στη συνέχεια απομακρύνθηκε και μετά προσγειώθηκε.

7 Συμπεράσματα και Μελλοντική Έρευνα

7.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία αναπτύχθηκε λογισμικό που προσφέρει έναν εύκολο τρόπο ώστε συστήματα μετρήσεων που υλοποιούνται σε ένα Arduino, να έχουν μια έτοιμη βιβλιοθήκη, ώστε να επικοινωνήσουν με τον flight controller και το σταθμό βάσης. Με αυτό τον τρόπο μπορεί οποιοσδήποτε κατασκευαστής πολυκοπτέρων να εισάγει ένα σύστημα μικρού μεγέθους και κατανάλωσης ενέργειας στη σχεδίασή του, για να λαμβάνει μετρήσεις για τις μεταβλητές που τον απασχολούν. Η διαδικασία που πρέπει να ακολουθηθεί έχει υλοποιηθεί ώστε να παρέχει σημαντική απλοποίηση. Απλά εισάγεται μια βιβλιοθήκη στο ενσωματωμένο σύστημα του Arduino, μαζί με τις κατάλληλες συναρτήσεις στο πρόγραμμα για δήλωση μεταβλητών τηλεμετρίας, καθώς και για την ενημέρωση των τιμών των αισθητήρων τους.

Η συγκεκριμένη υλοποίηση μειονεκτεί καθώς το πρωτόκολλο I2C είναι αργό, κάτι που όμως δεν επηρεάζει την παρούσα σχεδίαση, καθώς ο υπολογιστής ελέγχου πτήσης ούτως ή άλλως δεν έχει την ικανότητα να λάβει πιο γρήγορα μετρήσεις. Έτσι, δεν παρατηρήθηκε ιδιαίτερα μεγάλη καθυστέρηση στο σταθμό βάσης σε σχέση με την ανανέωση της μέτρησης στο Arduino, με αποτέλεσμα να μην γίνουν πειράματα για να μετρηθεί ο χρόνος μεταγωγής μιας μέτρησης από το Arduino στο σταθμό βάσης. Αντιθέτως, σε υλοποιήσεις με γρηγορότερους μικροελεγκτές ίσως παρουσιαστεί πρόβλημα.

Επιπροσθέτως, στα πλαίσια της διπλωματικής εργασίας επεκτάθηκαν οι ικανότητες του Arduino στο να μην μεταφέρει απλά μετρήσεις στον υπολογιστή ελέγχου πτήσης, αλλά να μπορεί να προσδιορίσει ότι μέρος αυτών των μετρήσεων αφορούν τον ίδιο τον υπολογιστή ελέγχου πτήσης. Η εισαγωγή των υπερηχητικών αισθητήρων είναι ένα παράδειγμα με το οποίο το πολυκόπτερο μπορεί να εκμεταλλευτεί την ικανότητα του αισθητήρα για μέτρηση απόστασης και να βελτιώσει την πτητική του λειτουργία, είτε κρατώντας σταθερό υψόμετρο με μεγαλύτερη ακρίβεια από πριν, είτε αποφεύγοντας εμπόδια που πλησιάζουν από πάνω ή και περιμετρικά το μη επανδρωμένο όχημα. Το πρωτόκολλο μπορεί προφανώς να επεκταθεί και σε άλλους αισθητήρες με διαφορετικούς σκοπούς κάθε φορά.

Η εισαγωγή των αισθητήρων για την επέκταση των ικανοτήτων του πολυκοπτέρου, δεν μπορούσε να γίνει χωρίς την επέμβαση στο firmware του υπολογιστή ελέγχου πτήσης, ώστε να σχεδιαστούν οι αντιδράσεις του σε σχέση με το νέο αισθητήρα που εισάγεται.

Όταν εισάγονται στον υπολογιστή ελέγχου πτήσης νέοι αισθητήρες, αναγκαστικά

αυξάνονται και οι υπολογιστικές ανάγκες, προκειμένου να διαχειριστεί τις νέες διεπαφές που δημιουργούνται. Με τη χρήση του Arduino, η σχεδίαση των διεπαφών αφαιρείται, αφού προσφέρεται κατευθείαν από τη βιβλιοθήκη που σχεδιάστηκε στα πλαίσια της παρούσας διπλωματικής.

Ο μέγιστος αριθμός αισθητήρων που μπορούν να εισαχθούν στο σύστημα εξαρτάται από τον ρυθμό δειγματοληψίας τους. Ενδεικτικά αναφέρεται ότι αν ο επιθυμητός ρυθμός δειγματοληψίας είναι 100Hz τότε 10 μετρήσεις αισθητήρων είναι το μέγιστο που μπορεί σταλεί από το Arduino ως το σταθμό βάσης. Αντιθέτως αν μειωθεί ο ρυθμός δειγματοληψίας στα 50Hz οι αισθητήρες αυτομάτως διπλασιάζονται κοκ.

Είναι δύσκολος ο ακριβής υπολογισμός της κατανάλωσης ενέργειας των προαναφερθέντων, καθώς υπάρχει ο ευμετάβλητος παράγοντας του αριθμού των αισθητήρων που μπορούν να ενσωματωθούν στο σύστημα. Παρόλα αυτά παρουσιάζονται οι καταναλώσεις ενέργειας των συστημάτων με τα οποία έγινε η πιστοποίηση λειτουργίας των υλοποιήσεων, ώστε ο αναγνώστης να αποκτήσει μια γενική εικόνα της επιβάρυνσης του συστήματος του από την προσθήκη μιας τέτοιας κατασκευής συλλογής μετρήσεων.

Μετρώντας το ρεύμα της τροφοδοσίας του υπολογιστή ελέγχου πτήσης προς το Arduino, ενώ το Arduino είχε πάνω συνδεδεμένο έναν BMP085 (Θερμοκρασία, Ατμοσφαιρική πίεση) με ρυθμό ανανέωσης 20Hz, έναν DHT11 (Θερμοκρασία, Υγρασία) με ρυθμό ανανέωσης 5Hz, ένα HC-SR04 που απλά έπαιρνε μετρήσεις με ρυθμό 20Hz και δηλωμένες 5 μεταβλητές με ρυθμό ανανέωσης 100Hz που απλά έγραφαν τυχαίες τιμές σε κάθε ανανέωση, το πολύμετρο έδειξε τιμή 35mA. Άρα τη κατανάλωση του Arduino είναι 175mW (35mA x 5V). Η κατανάλωση του συστήματος αναμενόταν να είναι χαμηλότερη αλλά λόγο του φόρτου που εισάγεται στο Arduino αυξάνεται αναλογικά και η κατανάλωση του. Παρατηρείται ότι όση προσπάθεια και να γίνεται στη μείωση της κατανάλωσης των ενσωματωμένων συστημάτων στα πολυκόπτερα, αυτό δεν έχει ιδιαίτερη επίδραση αφού κάθε μοτέρ μπορεί να απαιτήσει μέγιστο ρεύμα 20Amps, με αποτέλεσμα τα 4 μοτέρ μαζί να απαιτούν 80Amps, δηλαδή μέγιστη κατανάλωση 888Watts, πράγμα που κάνει αμελητέα την οποιαδήποτε κατανάλωση μπορεί έχει ένα ενσωματωμένο σύστημα.

Ζυγίζοντας το σύστημα του Arduino Mini με τους αισθητήρες που αναφέρονται στην προηγούμενη παράγραφο, παρατηρήθηκε ότι το συνολικό βάρος του συστήματος είναι 17 γραμμάρια. Δεδομένου ότι το πολυκόπτερο στο οποίο έγιναν οι δοκιμές ζυγίζει 1.2kg θεωρήθηκε ότι η επίδραση του νέου συστήματος στο συνολικό βάρος του πολυκοπτέρου είναι αμελητέα.

Το υλοποιημένο σύστημα το οποίο επιτρέπει την πλοήγηση του μη επανδρωμένου οχήματος μέσω εντολών από ένα ξεχωριστό ενσωματωμένο σύστημα, είναι χρήσιμο σε εφαρμογές όπου ένα σύστημα με μεγαλύτερη υπολογιστική ισχύ είναι απαραίτητο. Στην παρούσα διπλωματική πιστοποιήθηκε υποτυπωδώς η ορθή λειτουργία του επιπρόσθετου ενσωματωμένου συστήματος για αποφυγή εμποδίων και ανακλαστική κίνηση προς την αντίθετη κατεύθυνση.

7.2 Μελλοντική έρευνα

Η προσθήκη ενσωματωμένου συστήματος μεγαλύτερης υπολογιστικής ισχύος προσφέρεται για πληθώρα εφαρμογών όπως:

- Προσγείωση σε ακριβές σημείο για αυτόματη φόρτιση.
- Τρισδιάστατη απεικόνιση χώρων με χρήση κάμερας και 360^{o} lidar (αισθητήρα υπέρυθρου φωτός για μέτρηση αποστάσεων).
- Πλοήγηση με 360^{o} lidar και αισθητήρες υπερήχων μέσα σε κλειστούς χώρους. Εκτέλεση αλγορίθμων τεχνητής νοημοσύνης όπως A*, BFS (Breadth first search), DFS (Depth first search) για να πλοηγείται το πολυκόπτερο από ένα σημείο στο επόμενο.
- Εντοπισμός πιθανών σημείων πυρκαγιάς με χρήση υπέρυθρης κάμερας και κίνηση προς την κατεύθυνση της φωτιάς με σκοπό τη λήψη μετρήσεων για αέρια παράγωγα της καύσης υλικών ώστε να αποφεύγονται οι εσφαλμένες ενδείξεις της κάμερας.
- Κατασκευή δυναμικού χάρτη ο οποίος να λαμβάνει τις μετρήσεις του πολυκοπτέρου και να τις αναπαριστά σε πραγματικό χρόνο.
- Ενσωμάτωση νέου mode στο firmware του flight controller με σκοπό την συγκράτηση σταθερής απόστασης από ένα αντικείμενο μπροστά από το πολυκόπτερο με τρόπο όμοιο με εκείνο που λειτουργεί το Alt Hold. Με αυτό τον τρόπο το πολυκόπτερο με χρήση κάμερας θα μπορεί να χρησιμοποιηθεί για παρακολούθηση φθορών σε κάθετες επιφάνειες όπως φαράγγια, σκελετούς ανεμογεννητριών κλπ.

8 Βιβλιογραφία

[1] S. Chaudhry, M. Agate και S. Markus, «Basic Sonar-based Collision Avoidance for a Quadcopter,» Miami, 2014.

[2] M. Bartholomai και P. Neumann, «Micro-Drone for Gas Measurement in Hazardous Scenarios via Remote Sensing,» Berlin, 2010.

[3] P. Croize, M. Archez, J.Boissonm, T.Roger και V.Monsegu, «Autonomous Measurement Drone for Remote Dangerous Source Location Mapping,» International Jornal of Enviromental Science and Development, 2015.

[4] M. A. El-Diwiny και A.-H. M.El-Sayed, «Intelligent control of Unmanned aerial vehicle using inertial guidance Algorithm,» *The Online Journal on Electronics and Electrical Engineering*, τόμ. 5, 2014.

[5] K. Nonami, F. Kendoul, S. Suzuki, W. Wang και D. & Nakazawa, Autonomous Flying Robots, 2010.

[6] N. Gageik, T. Müller και S. Montenegro, «Obstacle detection and collision avoidance using ultrasonic distance sensors for an autonomous quadrocopter,» University Of Würzburg, Aerospace Information Technology (Germany), Würzburg, 2012.

[7] J. F. Roberts, T. S. Stirling, J.-C. Zufferey και D. Floareano, «Quadrotor using minimal sensing for autonomous indoor flight,» Lausanne, 2007.

[8] «AtMega2560 Documentation,» pp. http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf.

[9] « Crius AIO V2 Guide for MegaPirateNG,» pp. http://yety-tech.cz/wp-content/uploads/2013/07/Crius_AIOP_V2_0_Guide_for_MegaPirateNG_Ver_1_0.pdf.

[10] «Arduino Pro Mini Specifications,» p. https://www.arduino.cc/en/Main/ArduinoBoardProMini.

[11] «3DR Radio Set - Quick Start Guide,» pp. https://3dr.com/wp-content/uploads/2013/10/3DR-Radio-V2-doc1.pdf.

[12] «DHT11 Documentation,» p. http://akizukidenshi.com/download/ds/aosong/DHT11.pdf.

[13] «BMP085 Digital Pressure Sensor,» pp. https://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf.

[14] «HC-SR 04 Documentation,» p. http://www.micropik.com/PDF/HCSR04.pdf.

[15] «Ardupilot Flight Modes,» pp. http://copter.ardupilot.com/wiki/flight-modes/.

[16] «MaxSonar Operation on a Multi-Copter,» p. http://www.maxbotix.com/articles/067.htm, 2013.

[17] N. Semiconductors, «I2C bus specifications and user manual,» (http://www.nxp.com/documents/user_manual/UM10204.pdf), 2014.

[18] «Detailed Introduction to I2C,» pp. http://www.i2c-bus.org/i2c-bus/.

[19] G. Donald και L. Meier, «Mavlink Documentation,» p. (http://qgroundcontrol.org/mavlink/start).

[20] «MegaPirateNG Documentation,» p. http://docs.megapirateng.com/.

Παράρτημα

Πρωτόκολλο I2C – Σχετική Θεωρία

Σύντομη Ιστορία

Το I2C σχεδιάστηκε το 1982 από τη Philips για διάφορα chips της ίδιας με σκοπό την ελαχιστοποίηση του αριθμού των καλωδίων των διαύλων επικοινωνίας ανάμεσα στα περιφερειακά και την κεντρική μονάδα επεξεργασίας(microcontroller unit-MCU). Η χρήση παράλληλων διευθύνσεων και καθώς και καλώδιο bus για τα δεδομένα σε massive production consumer electronics επιβάρυνε οικονομικά και την εταιρία αλλά και τους καταναλωτές. Η μείωση του αριθμού των καλωδιώσεων θα βοηθούσε στη μείωση του μεγέθους των περιφερειακών(μείωση pins για τη διεύθυνση), των PCBs, του κόστους παραγωγής και άρα του κόστους των devices. Έτσι η Philips κατέληξε στην κατασκευή ενός δισύρματου καλωδίου επικοινωνίας που το ονόμασε I2C.

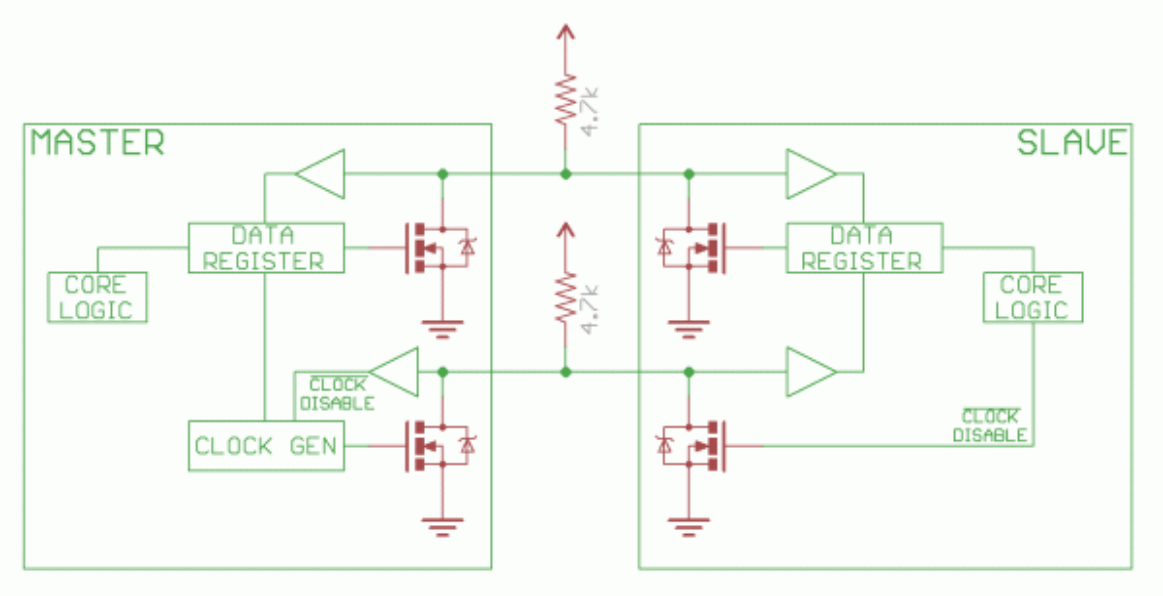
Η αρχική σχεδίαση του I2C λειτουργούσε μόνο για ταχύτητες 100kHz, με διευθύνσεις μόνο 7-bit και περιορισμένο αριθμό συσκευών (112 - αρκετές από τις διευθύνσεις είναι δεσμευμένες και δεν είναι έγκυρες διευθύνσεις I2C). Το 1992 το πρωτόκολλο επεκτάθηκε σε ταχύτητες 400kHz(fast mode) και σε διευθύνσεις 10-bit. Από τότε μέχρι σήμερα πολλά devices υποστηρίζουν I2C και έχουν αναπτυχθεί αρκετά διαφορετικά modes όπως το fast-mode plus στο 1 MHz, high-speed mode στα 3.4 MHz και ultra-fast mode στα 5 MHz.

Σήμερα το I2C bus χρησιμοποιείται σε πολλές εφαρμογές που απαιτούν μεταφορά δεδομένων κυρίως σε αισθητήρες αλλά όχι μόνο. Το I2C bus είναι γενικά αποδεκτό στην βιομηχανία και έχει υιοθετηθεί από πολλούς κατασκευαστές όπως STMicroelectronics, Intel, AMD, Texas Instruments, Atmel, Microchip Technology.

Hardware Level Architecture

Παρακάτω δίνεται ένα σχηματικό διάγραμμα του I2C driver ενός master και ενός slave. Όπως φαίνεται από το σχήμα οι drivers είναι open drain, το οποίο σημαίνει ότι μπορούν να οδηγήσουν το σήμα LOW, αλλά όχι και HIGH. Έτσι δεν μπορεί να υπάρξει διαμάχη στο κανάλι, δηλαδή ένα device προσπαθεί να κάνει το σήμα HIGH ενώ ένα άλλο LOW. Έτσι εξαλείφεται ο κίνδυνος να προκληθεί ζημιά στους drivers από υπερβολική διάχυση ισχύος στο σύστημα. Κάθε

σήμα έχει έναν pull-up resistor ούτως ώστε να επανέρχεται το σήμα σε HIGH όταν κανένα device δεν προσπαθεί να το κάνει LOW.



Σχήμα 2.6 Διάταξη I2C με pull up resistors και open drain διάταξη

Τα μηνύματα διαιρούνται σε δύο τύπους : τη διεύθυνση(address frame) όπου ο master υποδεικνύει τον slave στον οποίο θέλει να στείλει τα δεδομένα τα οποία είναι ένα η περισσότερα πακέτα δεδομένων(data frames) των 8-bit. Τα δεδομένα εισάγονται στο SDA όταν το SCL γίνει LOW και δειγματοληπτείται όταν το SCL γίνει HIGH. Ο χρόνος ανάμεσα στο SCL positive edge και στα read/write δεδομένα ορίζεται από τα devices στο bus και αλλάζει από chip σε chip.

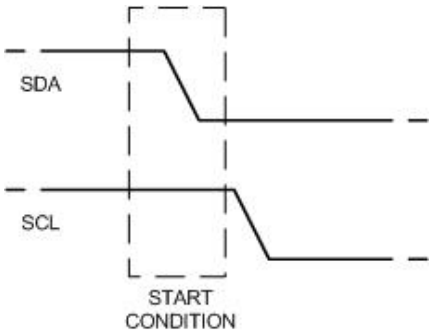
Ανάλυση Σημάτων

Start Condition

Για να εκκινηθεί μια ανταλλαγή δεδομένων μεταξύ του master και μιας συσκευής, ο master αναγκάζει το SDA σε μετάβαση από HIGH σε LOW ακολουθούμενη από μια μετάβαση του SCL από HIGH σε LOW. Έτσι τα devices-slaves αντιλαμβάνονται ότι μια επικοινωνία εκκινείται.

Αν δύο ή περισσότεροι masters θελήσουν να επικοινωνήσουν την ίδια στιγμή, εκείνος που θα θέσει το SDA σε LOW πρώτος θα καταφέρει να “εξουσιάσει” το bus.

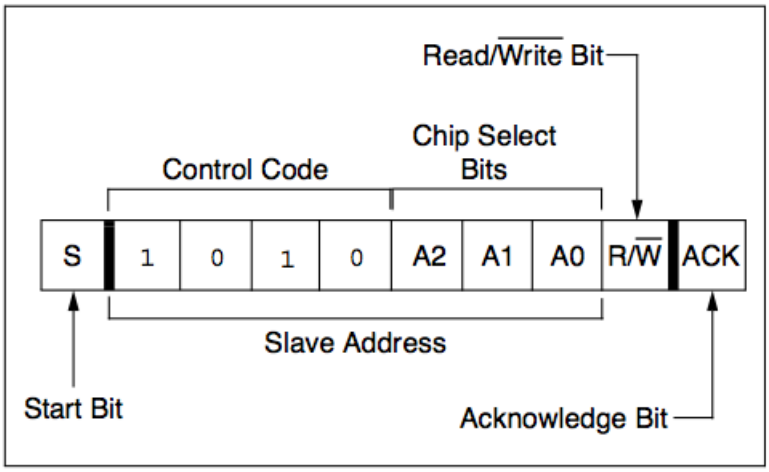
Είναι πιθανό ένας master να εκπέμπει επαναλαμβανόμενα start conditions ώστε να μην “παραδίδει” το bus σε άλλους masters.



Σχήμα 2.7 Start Condition

Address Frame

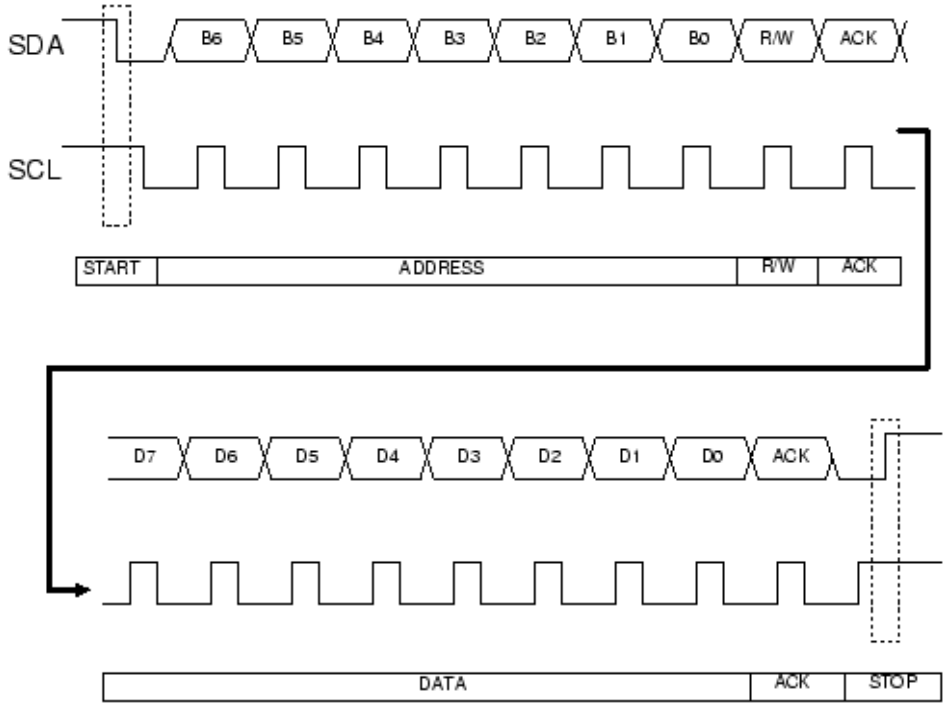
Το address frame είναι πάντα το πρώτο πακέτο σε κάθε επικοινωνία. Για 7-bit διευθύνσεις, το 8-bit πακέτο αποτελείται από τα πρώτα 7-bit (MSBs) τα οποία είναι η διεύθυνση ενώ το τελευταίο(LSB) είναι είτε read(1) είτε write(0). Το LSB υποδεικνύει στον slave τη διαδικασία που θα ακολουθήσει. Στο παρακάτω σχήμα παρατηρείται ότι τα bit πληροφορίας είναι 9 και όχι 8(το start bit είναι το start condition). Το 9ο bit του frame είναι είτε ACK (acknowledgment - επιβεβαίωση) είτε NACK (non-acknowledgment - αρνητική επιβεβαίωση) bit και συμβαίνει σε όλα τα frames. Όταν σταλούν τα πρώτα 8-bit, η συσκευή που λαμβάνει τα δεδομένα παίρνει τον έλεγχο του SDA. Αν το device που λαμβάνει τα δεδομένα(ουσιαστικά τη διεύθυνσή του) δεν θέσει το SDA σε LOW πριν τον 9ο SCL pulse, μπορεί να θεωρηθεί ότι το device που λαμβάνει την πληροφορία είτε δεν την έλαβε, είτε δεν είχε την ικανότητα να προσπελάσει τα δεδομένα. Σε αυτή την περίπτωση ο master αποφασίζει πως θα προχωρήσει.



Σχήμα 2.8 Address Frame with Acknowledgment

Data frames

Αφού σταλεί το address frame, ξεκινά η διαδικασία μετάδοσης δεδομένων. Ο master ή ο slave συνεχίζει να παράγει SCL pulses και τα δεδομένα θα τοποθετηθούν στο SDA, ανάλογα με το bit που έχει εκπέμψει ο master(Read/Write). Ο αριθμός των data frames δεν είναι οορισμένος(μπορούν να σταλούν όσα frames χρειάζεται) και τα περισσότερα slaves θα αυξάνουν τον εσωτερικό τους register για να στείλουν/διαβάσουν το επόμενο byte, αλλά αυτό δεν είναι πάντα ο κανόνας.



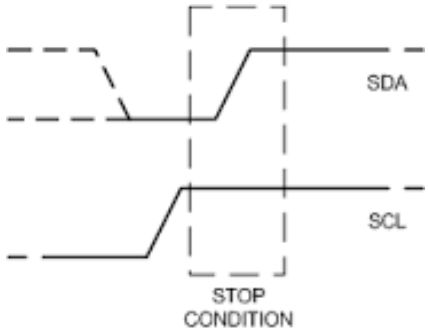
Σχήμα 2.9 Μια ολοκληρωμένη μεταφορά ενός byte

Stop Condition

Όταν όλα τα data frames σταλούν ο master

δημιουργεί ένα stop condition. Το stop condition ορίζεται ως μια μετάβαση 0->1 (από μηδέν σε ένα) στο SDA μετά από μια μετάβαση 0->1 στο SCL με το SCL να μένει στο 1 μετά τις μεταβάσεις. Σε διαδικασία εγγραφής ή ανάγνωσης το SDA δεν

πρέπει να μεταβάλλεται όταν το SCL είναι HIGH για να αποφεύγουμε λανθασμένα stop conditions.



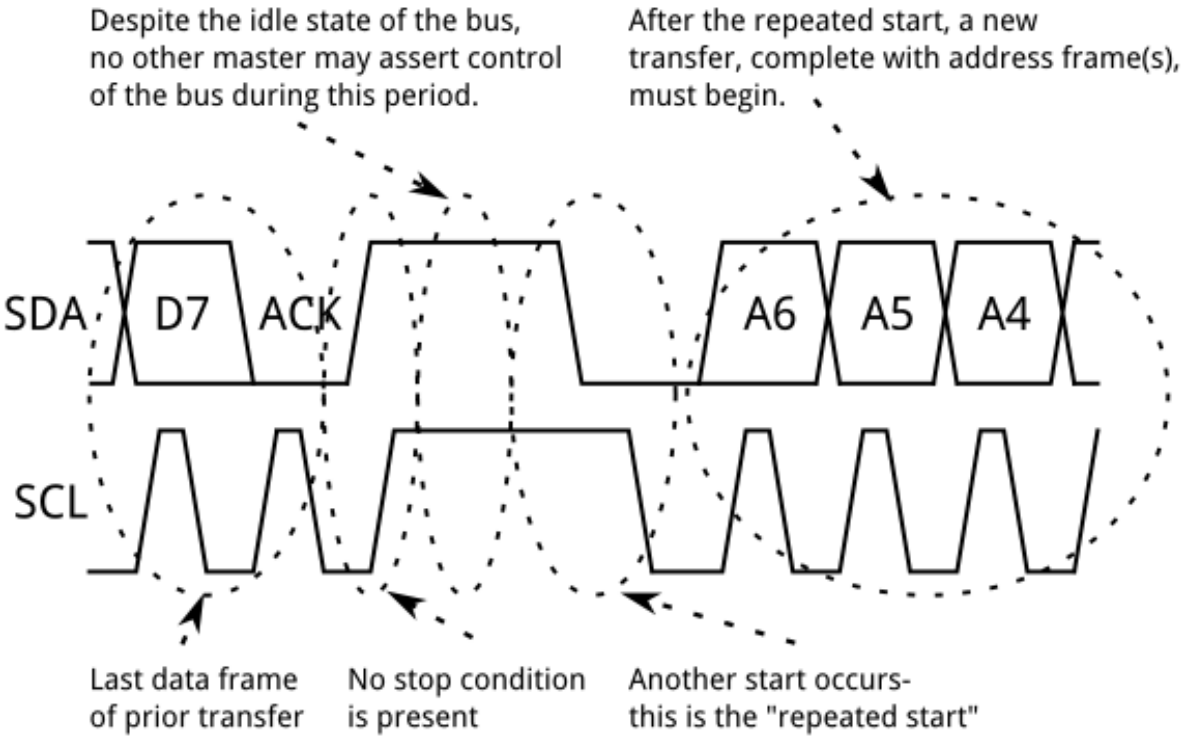
Σχήμα 2.10 Stop Condition

Repeated Start Condition

Μερικές φορές είναι σημαντικό ο master να ανταλλάξει περισσότερα του ενός data frames, με διαφορετικά devices, σε μία επικοινωνία(ανάμεσα σε ένα start - stop condition) χωρίς να αφήσει άλλους masters να παρέμβουν στο bus. Γι' αυτό το λόγο έχει οριστεί το repeated start condition.

Για να εφαρμοστεί repeated start operation, το SDA κάνει μετάβαση LOW->HIGH ενώ το SCL είναι LOW και μετά το SCL κάνει κι αυτό μετάβαση LOW->HIGH. Πρακτικά γίνεται η αντίθετη διαδικασία από αυτή που γίνεται στο stop condition με αποτέλεσμα η επικοινωνία να μην τερματίζεται και ο συγκεκριμένος master να μην χάνει τον έλεγχο του bus. Η διαδικασία φαίνεται στο παρακάτω σχήμα.

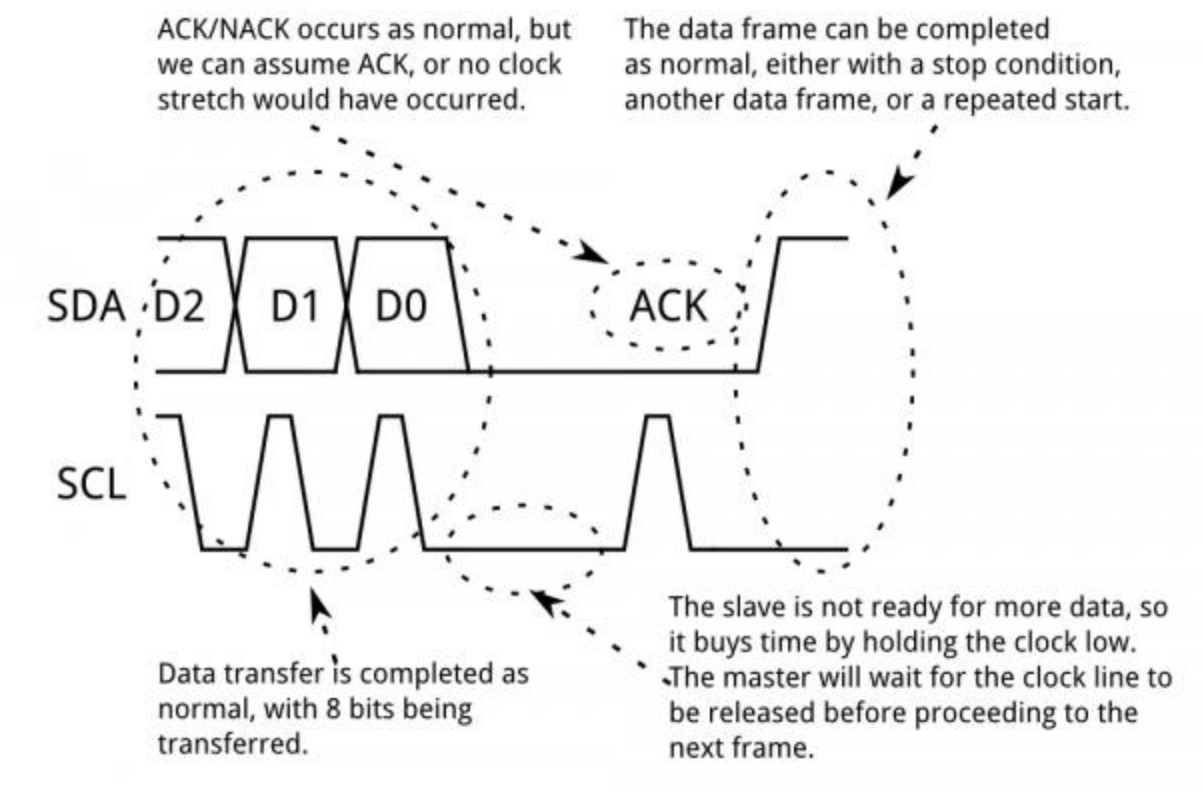
Σε αυτό το σημείο μπορεί να ξεκινήσει η νέα μετάδοση όπως κάθε άλλη αφού και τα δύο σήματα είναι HIGH. Οποιοσδήποτε αριθμός repeated starts είναι επιτρεπτός.



Σχήμα 2.11 Παράδειγμα Repeated Start Condition

Clock Stretching

Πολλές φορές, ο ρυθμός μετάδοσης των δεδομένων που έχει ορίσει ο master υπερβαίνει την ικανότητα του slave να επιστρέψει τα δεδομένα. Αυτό μπορεί να συμβεί επειδή τα δεδομένα δεν είναι έτοιμα ακόμα(π.χ. περιμένει έναν analog-to digital converter να ολοκληρώσει μία μέτρηση) ή κάποια διαδικασία δεν έχει ολοκληρωθεί(π.χ. δεν έχει τελειώσει το γράψιμο σε μια non-volatile memory και πρέπει να τελειώσει για να εξυπηρετηθούν άλλες λειτουργίες).



Σχήμα 2.12 Clock Stretching

Σε αυτές τις περιπτώσεις τα slave devices θα εκτελέσουν clock stretching. Το SCL οδηγείται από το master και τα slaves απλά γράφουν ή διαβάζουν δεδομένα από το SDA παρατηρώντας τους παλμούς του SCL, δηλαδή του master. Σε οποιοδήποτε σημείο της μετάδοσης ο slave μπορεί να κρατήσει το SCL LOW όταν ο master το αφήσει. Ο master δεν παράγει επιπλέον παλμούς μέχρι ο slave να απελευθερώσει το SCL.

Πηγές : [17], [18]

MAVlink

Δομή ενός MAVlink πακέτου

Το MAVlink μήνυμα είναι ένα stream από bytes τα οποία κωδικοποιούνται από τον πομπό, είτε αυτός είναι μη επανδρωμένο όχημα είτε είναι το Ground Control Station (GCS), και μεταδίδονται μέσω σειριακής θύρας. Η κωδικοποίηση του μηνύματος εδώ δεν σημαίνει κάποιας μορφής encryption άλλα την κατασκευή μιας δομής δεδομένων που περιλαμβάνει το μήνυμα προς αποστολή και κάποια επιπλέον bytes που φαίνονται στο σχήμα 2.13.

Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload "means" and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Σχήμα 2.13 Στοιχεία ενός MAVlink μηνύματος

Το MAVlink πακέτο αποτελείται από 6 bytes header(τα 6 πρώτα bytes του πακέτου), το μήνυμα/τα δεδομένα (payload) και στο τέλος του πακέτου δύο bytes checksum (CRC). Η δομή ενός πακέτου φαίνεται στο σχήμα 2.13.

Το πρώτο byte(0) δηλώνει την αρχή ενός νέου μηνύματος. Το δεύτερο byte(1) δηλώνει το μήκος του μηνύματος Το τρίτο byte(2) δηλώνει το ρυθμό αποστολής του πακέτου ώστε να είναι εύκολη η ανίχνευση χαμένων πακέτων. Το τέταρτο byte(3) δείχνει το ID της συσκευής που στέλνει το μήνυμα(άρα μπορούμε να έχουμε περισσότερα μη επανδρωμένα οχήματα και GCSs στο ίδιο δίκτυο). Το πέμπτο byte(4) δείχνει το Component ID (καθώς ο Controller ενός μη επανδρωμένου οχήματος δεν είναι ανάγκη να είναι ένα Board, όπως επίσης σε ένα board είναι πιθανό να δημιουργηθούν διάφορα software components με διαφορετικά IDs). Το έκτο byte(5) δηλώνει το ID του μηνύματος το οποίο είναι χρήσιμο στον δέκτη ούτως ώστε να διαχειριστεί τα bytes που ακολουθούν στο payload. Το payload είναι το μήνυμα το οποίο στέλνεται και θα αναλυθεί παρακάτω η μορφή στην οποία δηλώνεται ένα MAVlink μήνυμα(εδώ εννοούμε το payload) και θα γίνει περισσότερο κατανοητό γιατί είναι απαραίτητο Message ID.

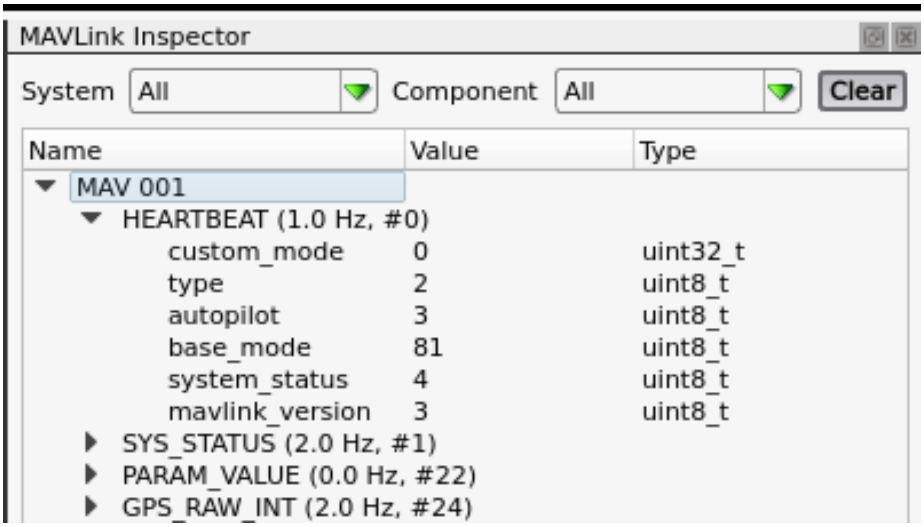
Δομή Δεδομένων (Data)

Παρακάτω δίνεται η μορφή του μηνύματος heartbeat όπως αυτό δηλώνεται στο MAVlink.

```
<message id="0" name="HEARTBEAT">
  <description>The heartbeat message</description>
  <field type="uint8_t" name="type">Type of the MAV </field>
  <field type="uint8_t" name="autopilot">Autopilot type / class </field>
  <field type="uint8_t" name="base_mode">System mode bit field </field>
  <field type="uint32_t" name="custom_mode">Navigation mode bit field </field>
  <field type="uint8_t" name="system_status">System status flag </field>
  <field type="uint8_t_mavlink_vers" name="mavlink_vers">MAVLink version</field>
</message>
```

Σχήμα 2.14 Δήλωση μεταβλητών ενός πακέτου MAVlink

Σύμφωνα με την παραπάνω μορφή είναι εύκολα κατανοητό ότι η ένα μήνυμα μπορεί να περιέχει αόριστο αριθμό πεδίων(fields), με διαφορετικού τύπου μεταβλητές το καθένα, οπότε το Message ID είναι απαραίτητο ούτως ώστε ο δέκτης του μηνύματος να γνωρίζει τη μορφή των δεδομένων που θα διαβάσει στο payload. Το μήνυμα heartbeat στο σταθμό βάσης έχει την παρακάτω μορφή.



Σχήμα 2.15 Στιγμιότυπο μέτρησης όπως εκείνη φαίνεται στο Σταθμό Βάσης

Πεδίο CRC (Cyclic Redundancy Check)

Η ακεραιότητα του μηνύματος διασφαλίζεται μέσω του CRC το οποίο αποτελεί τα 2 τελευταία bytes του αρχείου. Τα bytes του CRC είναι βοηθητικά ούτως ώστε πομπός και δέκτης να λειτουργούν με πακέτα ίδιας MAVlink έκδοσης και να αποκλείσουν τυχόν σφάλματα κατά τη μετάδοση του πακέτου. Το CRC υπολογίζεται κατά το generation του εκάστοτε πακέτου (δηλαδή όταν μετατρέπεται από xml σε δομή δεδομένων της εκάστοτε γλώσσας προγραμματισμού) και αποθηκεύεται σε πίνακα ώστε να μην επανυπολογίζεται από πομπούς και δέκτες κατά τη λειτουργία του μη επανδρωμένου οχήματος, με αποτέλεσμα να αυξάνεται η κατανάλωση ενέργειας. Το checksum είναι το ίδιο, με εκείνο που χρησιμοποιείται στα standards ITU X.25(CRC-16-CCITT) και SAE AS-4(SAE AS5669A).

Πηγή : [19]

Megapirate NG and APM firmware Modes

Stabilize

Στο Stabilize όλα ελέγχονται από τον χειριστή του πολυκοπτέρου. Σε περίπτωση που εκείνος αφήσει κάποια εκ' των pitch, roll και yaw το πολυκόπτερο θα προσπαθήσει να ισορροπήσει και να κρατήσει σταθερή κατεύθυνση. Το throttle που στέλνεται από το χειριστή είναι ο μέσος όρος rpm των motors που σημαίνει ότι το πολυκόπτερο δεν μένει σε σταθερό ύψος και χρειάζεται συνεχώς αναπροσαρμογές από τον χειριστή.

Να σημειωθεί ότι όταν τα roll, pitch και yaw μείνουν ακίνητα το πολυκόπτερο θα επιχειρήσει να σταθεροποιηθεί. Πλευρικός άνεμος θα κάνει το πολυκόπτερο να χάσει την θέση του. Το αεροσκάφος όμως θα ελέγξει την ισορροπία του και δεν θα στραφεί υπερβολικά με καταστροφικές για αυτό συνέπειες.

Alt Hold

Η λειτουργία για τα roll, pitch και yaw είναι ίδια με το Stabilize Mode αλλά το throttle ελέγχεται από το πολυκόπτερο. Το πολυκόπτερο εδώ κρατά σταθερό το ύψος του ελέγχοντας το βαρόμετρο του και μετατρέποντας την ατμοσφαιρική πίεση σε υψόμετρο.

Loiter

Το αεροσκάφος θα κάνει ότι ακριβώς και στο Alt Hold με τη διαφορά ότι εδώ το πολυκόπτερο θα κρατήσει τη θέση του σε περιπτώσεις όπου δέχεται ωθήσεις από το περιβάλλον του να τη χάσει(π.χ. πλευρικός άνεμος). Για να συμβεί το παραπάνω η χρήση GPS είναι απαραίτητη.

Land

Το αεροσκάφος θα μειώσει το υψόμετρό του ως τα 10m με προκαθορισμένη ταχύτητα ορισμένη από το σταθμό βάσης. Μετά θα συνεχίσει να κατεβαίνει με διαφορετική ταχύτητα και πάλι ορισμένη από το σταθμό βάσης. Όταν το πολυκόπτερο φτάνει στο έδαφος κλείνει τους κινητήρες

Το πολυκόπτερο αντιλαμβάνεται ότι έφτασε στο έδαφος όταν το τελευταίο δευτερόλεπτο οι κινητήρες του λειτουργούν στο minimum και ο ρυθμός μεταβολής του ύψους του δεν έχει μεταβληθεί περισσότερο από 20m/s.

RTL

Σε αυτό το mode το πολυκόπτερο αρχικά θα ανέβει σε ένα προκαθορισμένο υψόμετρο, ορισμένο από το σταθμό βάσης, και θα επιστρέψει στο σημείο στο οποίο ξεκίνησε την πτήση του

κρατώντας το ίδιο υψόμετρο. Εκεί θα κάνει Land μετά από προκαθορισμένο χρόνο, ορισμένο από το σταθμό βάσης.

Auto

Το Auto είναι mode του Αυτόματου πιλότου. Όταν το πολυκόπτερο κληθεί να λειτουργήσει σε Auto Mode τότε θα εκτελέσει την προσχεδιασμένη πτήση που δόθηκε από τον σταθμό βάσης.

Acro

Το Acro είναι mode που χρησιμοποιείται για ακροβατικά. Εδώ τα sticks του χειριστηρίου χρησιμοποιούνται για να χειριστεί κάποιος τις γωνιακές ταχύτητες του χειριστηρίου και όταν αφεθούν τα sticks το πολυκόπτερο κράτα το ύψος του αλλά δεν θα ισορροπήσει, θα κρατήσει το pitch, yaw και roll που του έχουν οριστεί.

Guided

Το συγκεκριμένο mode δεν ενεργοποιείται με switch του χειριστηρίου. Το mode αυτό χρησιμοποιείται με εφαρμογές του σταθμού βάσης όπου ο χειριστής έχει την ικανότητα να πατήσει διαδραστικά ένα σημείο στο χάρτη και το πολυκόπτερο θα μετακινηθεί εκεί πετώντας πάνω από το σημείο και περιμένοντας νέες εντολές.

Circle

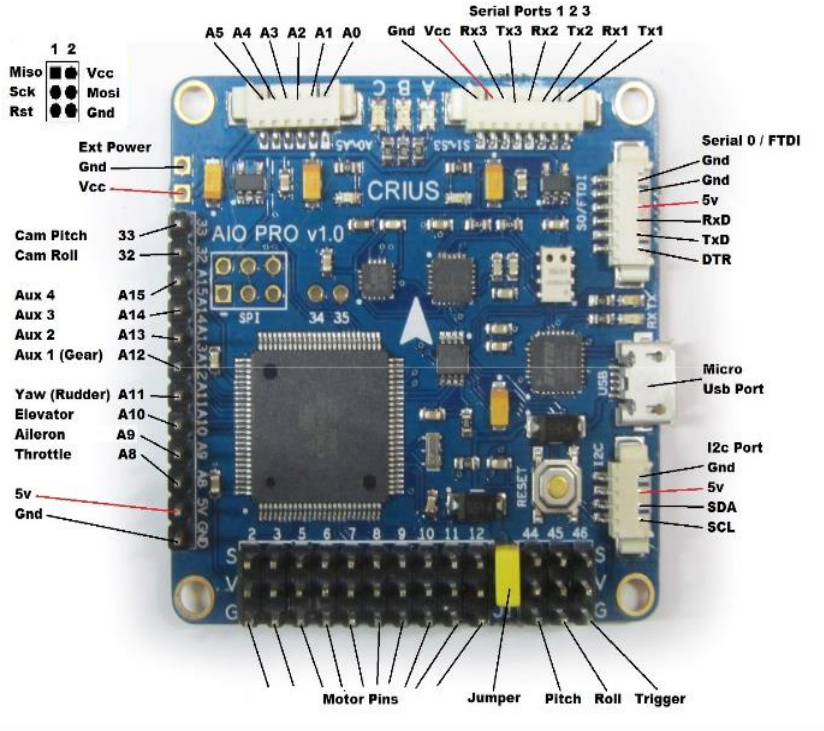
Το πολυκόπτερο θα αρχίσει να περιστρέφεται γύρω από ένα σημείο ενδιαφέροντος με την κατεύθυνσή του στραμμένη προς το σημείο εκείνο. Η ταχύτητα περιστροφής καθώς και η γωνία είναι παράμετροι που καθορίζονται από το σταθμό βάσης.

Follow Me

Το πολυκόπτερο θα ακολουθήσει το σταθμό βάσης αρκεί εκείνος να χρησιμοποιεί μια κεραία τηλεμετρίας και GPS.

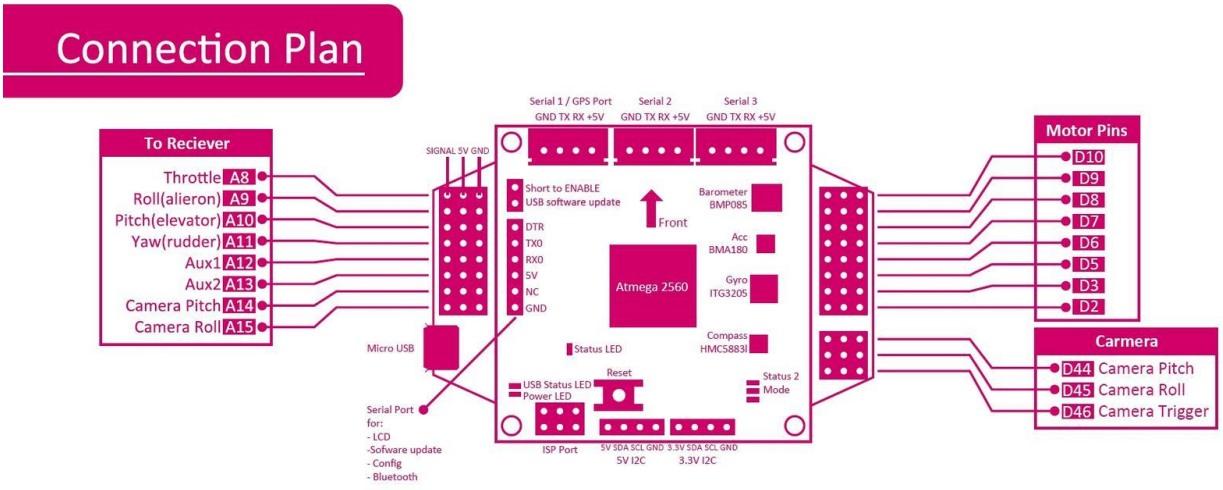
MPNG compatible Flight Controller Pinout

Παρακάτω φαίνεται το Crius V1 όπου φαίνονται όλα τα pinouts.



Σχήμα2.13 Crius V1.0 Pinout

Παρακάτω φαίνεται το HK Red MultiWii όπου φαίνονται όλα τα pinouts που περιγράφονται παραπάνω.



Σχήμα2.13 HK Red MultiWii Pinout