ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΤΗΛΕΧΕΙΡΙΖΟΜΕΝΟ ΠΛΟΙΟ ΓΙΑ ΤΗ ΧΡΗΣΗ ΨΑΡΕΜΑΤΟΣ»

Των φοιτητών

Ιωσηφίδης Μιχαήλ, ΑΜ: 513199

Χονδρόπουλος Κων/νος, ΑΜ: 513191

Επιβλέπων

Γιακουμής Άγγελος

Λέκτορας

Ημερομηνία: 14/9/2021

Τίτλος Δ.Ε. : Τηλεχειριζόμενο πλοίο για τη χρήση ψαρέματος
Κωδικός Δ.Ε.: 19163
Ονοματεπώνυμο φοιτητών : Ιωσηφίδης Μιχαήλ, Χονδρόπουλος Κων/νος
Ονοματεπώνυμο εισηγητή: Γιακουμής Άγγελος
Ημερομηνία ανάληψης Δ.Ε. : 18/11/2019
Ημερομηνία περάτωσης Δ.Ε. : 14/9/2021

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

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Για την εκπόνηση της πτυχιακής μας εργασίας αναλάβαμε την κατασκευή ενός τηλεχειριζόμενου πλοίου για την χρήση ψαρέματος. Η αγάπη για το ψάρεμα και την δημιουργία μάς οδήγησε στην αναζήτηση πληροφοριών μέχρι την ολοκλήρωση του δικού μας τηλεχειριζόμενου σκάφους.

Το ψάρεμα. εκτός από επάγγελμα, είναι και ένα είδος ψυχαγωγίας και πάνω σε αυτό το κομμάτι της ψυχαγωγίας βασίστηκε η εργασία. Στην αγορά τα συγκεκριμένα σκαφάκια πωλούνται συνήθως από χομπίστες. Δεν υπάρχουν συγκεκριμένες προδιαγραφές για αυτά τα σκαφάκια, καθώς προσθέτει ό,τι θέλει ο καθένας, σε αντίθεση δε με τα drone που δεν μπορούν να εξυπηρετήσουν τόσο καλά αυτό το σκοπό.

Στόχος της παρούσας εργασίας είναι η ανάπτυξη ενός σκάφους που θα έχει την δυνατότητα να πηγαίνει έως 500 μέτρα και να ρίχνει το αγκίστρι μέσα. Επίσης, θα έχει υποδοχή όπου θα μπορούμε να βάζουμε την μαλάγρα και να την ρίχνουμε εκεί που θα ρίχνουμε και το αγκίστρι. Το τηλεχειριστήριο το οποίο θα διαθέτουμε θα μάς δείχνει τις συντεταγμένες στην lcd οθόνη για το πού βρίσκεται το σκάφος, την απόσταση του σκάφους από την στεριά και την ταχύτητα με την οποία κινείται. Επίσης, θα έχει φως ώστε να μπορούμε να το εντοπίσουμε την νύχτα. Τέλος, όταν υπάρχει κίνδυνος να ξεμείνει από μπαταρία, ή να ξεπεράσει την απόσταση που ορίσαμε, θα γυρίζει την αρχική του θέση.

Περίληψη

Η εργασία βασίζεται στην μελέτη και έπειτα στην κατασκευή ενός τηλεχειριζόμενου σκάφους για την χρήση ψαρέματος. Αρχικά, μέσα από μία ιστορική αναδρομή δίνονται πληροφορίες για διάφορους τρόπους ψαρέματος σε διάφορες χώρες, από την αρχαιότητα έως σήμερα. Συνεχίζοντας με την βιβλιογραφική αναφορά επικεντρώνεται σε πληροφορίες σχετικά με τα υλικά που χρησιμοποιήθηκαν για την δική μας κατασκευή, δηλαδή τον τρόπο χρήσης και λειτουργίας τους.

Η βασική ιδέα του τηλεχειριζόμενου σκάφους είναι ένας ερασιτέχνης ψαράς να βάζει την πετονιά του πίσω στον απαγκιστρωτή, να το πηγαίνει μέχρι 500 μέτρα, να πετάει την πετονιά μέσα στο νερό, να ρίχνει την μαλάγρα. Έπειτα, να δει τις συντεταγμένες τις οποίες άφησε το σκάφος και να το γυρίσει πίσω. Οι συνταγμένες βοηθάνε, ώστε αν πιάσει ψάρι να γνωρίζει το μέρος για να ξαναπάει.

Η λειτουργία του ηλεκτρονικού μέρος βασίζεται στον μικροελεγκτή Atmega328P – PU. Όλη η διαχείριση του κυκλώματος και στο τηλεχειριστήριο και στο πλοίο βασίζεται πάνω σε αυτόν. Μόλις επεξεργάζεται τα δεδομένα, θα τα δίνει στο lora μέσω του πρωτόκολλου επικοινωνίας SPI. Έπειτα, τα lora, τα οποία λειτουργούν ως πομποδέκτες, θα στέλνουν εναλλάξ τα πακέτα.

Ολοκληρώνοντας την εργασία, διαπιστώσαμε ότι η κατασκευή ενός τέτοιου σκάφους είναι περίπλοκη. Χρειάζεται αρκετό χρόνο, μελέτη και λεπτομέρεια σε πολλά στοιχεία τα οποία αλληλεπιδρούν όλα μεταξύ τους, από το ηλεκτρονικό μέχρι και το μηχανικό κομμάτι.

Λέξεις – κλειδιά: Μικροελεγκτής, LORA

Abstract

The study is based on the design and construction of a remote-control boat for fishing use. Initially, through a historical background, information is given about different ways of fishing in different countries, from antiquity to the present day. Continuing with the bibliographic report, it focuses on information about the materials used for our construction and how they are used and operated.

The basic idea of the remote-control boat is for an amateur fisherman to put his line behind the forbidden one, to go up to 500 meters, to throw the line in the water, to throw the malaga. Then to see the new coordinates he left the boat and turned it back. The recipes help to catch fish to get to know the place to go again.

The operation of the electronic part is based on the Atmega328P-PU microcontroller. All circuit management on both the remote control and the ship is based on it. Once the data is processed, they will give it to lora via the SPI communication protocol. Then the lora which act as transceivers, will send the packets alternately.

Concluding the study, we found that the construction of such a boat is complicated. It takes a lot of time, study and detail on many elements that interact with each other, from the electronic to the mechanical part.

Keywords: Microcontroller, LORA

Ευχαριστίες

Με το πέρας αυτής της εργασίας θα θέλαμε να ευχαριστήσουμε θερμά τον επιβλέποντα καθηγητή κ. Γιακουμή Άγγελο, για την άμεση ανταπόκριση κάθε φορά που είχαμε οποιαδήποτε απορία. Επίσης, οφείλουμε να ευχαριστήσουμε τον κ. Βογιατζόγλου Δημήτριο, που έφτιαξε το εξωτερικό μέρος του σκάφους. Θα θέλαμε να ευχαριστήσουμε και τον Βουτυρά Γιώργο από την Hobbyland, που μάς έδωσε συμβουλές για το μηχανικό κομμάτι του σκάφους. Τέλος, θα θέλαμε να ευχαριστήσουμε τον φίλο μας Μηνά Τάσο, που μάς έδωσε υλικά και μάς βοήθησε στην εκτύπωση του τηλεχειριστηρίου με το 3D printer.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή στο Ψάρεμα

1.1 Εισαγωγή

Στην σημερινή εποχή το ψάρεμα είναι ένας τρόπος διαφυγής, χαλάρωσης από την καθημερινότητα. Ως αλιεία ή ψάρεμα χαρακτηρίζεται τόσο το κυνήγι, όσο και ο τρόπος που χρησιμοποιείται για την σύλληψη ψαριών και άλλων υδρόβιων ζώων. Το ψάρεμα είναι μια αρχαία δραστηριότητα που σήμερα χρησιμοποιείται παγκόσμια και συνεισφέρει τροφή, θέσεις εργασίας, άθληση και ψυχαγωγία. Η εργασία επικεντρώνεται στο κομμάτι της ψυχαγωγίας.

Ο στόχος του κάθε ψαρά είναι να μπορέσει να ψαρέψει σε όσο μεγαλύτερο βάθος γίνεται, εκεί που υπάρχει και η μεγαλύτερη δραστηριότητα από ψάρια. Ένας ερασιτέχνης ψαράς με δυσκολία θα πετάξει το καλάμι του πάνω από 100 μέτρα, ενώ, επίσης, δεν θα έχει την δυνατότητα να διαθέτει ένα βαρκάκι για οικονομικούς λόγους. Στο παρελθόν ήταν αδύνατον να ψαρέψεις στα βαθιά χωρίς να διαθέτεις βάρκα. Τώρα ξεκίνησαν νέοι τρόποι που μας δίνουν την δυνατότητα για ψάρεμα σε μεγάλη απόσταση, ένας δημοφιλής τρόπος είναι τα drone και ο άλλος τρόπος είναι αυτός που θα αναλύσουμε, το τηλεχειριζόμενο σκαφάκι.

Η εργασία έχει διαιρεθεί σε επιμέρους κεφάλαια, κάθε ένα από τα οποία πραγματεύεται ένα διαφορετικό δομικό κομμάτι της. Συγκεκριμένα, στο πρώτο κεφάλαιο γίνεται μία γενική μελέτη του ψαρέματος, ως προ την ιστορία του και σταδιακά φτάνουμε να δούμε πως εξελίχθηκε ο τρόπος ψαρέματος μέχρι σήμερα.

Στο δεύτερο κεφάλαιο γίνεται μια θεωρητική παρουσίαση του κάθε εξαρτήματος που χρησιμοποιήσαμε, με σκοπό να καλυφθεί το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της λειτουργίας του κάθε εξαρτήματος.

Το τρίτο κεφάλαιο αφορά το τηλεχειριστήριο. Πιο συγκεκριμένα αναφέρεται στη διαδικασία που χρησιμοποιήσαμε για την σχεδίαση των βαθμίδων και παράλληλα στην υλοποίηση του.

Στο τέταρτο κεφάλαιο γίνεται μια ανάλυση για το καραβάκι. Συγκεκριμένα αναλύονται οι βαθμίδες από τις οποίες αποτελείται και στη συνέχεια εστιάζεται το κεφάλαιο στην υλοποίηση του.

Στο πέμπτο και τελευταίο κεφάλαιο αναφέρονται κάποια συμπεράσματα, που αφορούν τη λειτουργικότητα της τελικής κατασκευής και προτείνονται τρόποι οι οποίοι θα οδηγούσαν στην βελτίωση της.

1.2 Ιστορική αναδρομή

Προϊστορία

Η αλιεία είναι μια πανάρχαια πρακτική που χρονολογείται τουλάχιστον από την Ανώτερη Παλαιολιθική Εποχή, πριν περίπου 40.000 χρόνια. Αρχαιολογικά ευρήματα κοχυλιών και υπολειμμάτων από ψαροκόκαλα, καθώς και σχετικές βραχογραφίες σε σπήλαια δείχνουν ότι τα αλιεύματα αποτελούσαν σημαντική σε ποσότητα πηγή τροφής, αλλά και είδος ανταλλαγής για τους ανθρώπους της περιόδου. Μάλιστα, ενώ η πλειοψηφία των ανθρώπινων φυλών της εποχής ζούσαν μια

νομαδική ζωή κυνηγών - τροφοσυλλεκτών που αναγκαστικά μετακινούνταν σχεδόν συνεχώς, υπήρχαν παραδείγματα φυλών με σχετικά μόνιμη εγκατάσταση, όπως π.χ. αυτή στο Lepenski Vir, που πάντοτε συσχετίζονταν με μια σχετικά σταθερή κύρια πηγή αλιευμάτων που παρείχαν την κύρια πηγή τροφής των αλιευτικών αυτών φυλών. Μάλιστα χρησιμοποιούσαν την περίσσεια αλιευμάτων και κοχύλια για ανταλλαγές με άλλα προϊόντα με άλλες φυλές με τις οποίες είχαν επαφή και δεν διέθεταν αυτά τα είδη.

Κατά τη Νεολιθική Εποχή μεταξύ των μεγάλων νέων τεχνολογικών επιτευγμάτων της εποχής περιλαμβάνονταν και πολλές βασικές τεχνικές αλιείας, που πολλές απ' αυτές χρησιμοποιούνται παρόμοια ως τις μέρες μας [1].

Αίγυπτος

Οι Αρχαίοι Αιγύπτιοι είχαν ανακαλύψει αρκετούς τρόπους ψαρέματος και εργαλεία. Όλα αυτά τα είχαν εικονογραφήσει στις ταφικές τους τοιχογραφίες και σε πάπυρους. Ο ποταμός Νείλος στην αρχαία εποχή ήταν πολύ πλούσιος σε ψάρια. Ψάρευαν με δίχτυα και πετονιές στην κοίτη του. Όταν αποσύρονταν το νερό ήταν πολύ εύκολο να τα πιάσουν.

Έπιαναν τόσο απίθανες ποσότητες ψαριών μ' αυτό το τρόπο, ώστε στην συνέχεια έκαναν εξαγωγή σε άλλες χώρες αφού τα ξέραιναν απλά στον ήλιο. Παρόλο που οι λαοί της Νότιας Ευρώπης και οι Λατίνοι δεν είχαν έλλειψη ψαριών, τους άρεσε ιδιαίτερα το ξεραμένο ψάρι που προερχόταν από την Αίγυπτο. Το ψάρι αποτέλεσε ίσως την βασική τροφή των φτωχών του εσωτερικού της χώρας οι οποίοι δεν είχαν σχεδόν καμιά δυνατότητα να προμηθευτούν κρέας με την εκτροφή ζώων ή με το κυνήγι [1][2].

Σχήμα 1.1: Αιγύπτιοι φέρνουν ψάρια και τα ετοιμάζουν για πάστωμα [1].

Ινδία

Το κλασσικό δραβιδικό βασίλειο των Ταμίλ της Πανδύας, στην Ινδία, ήταν γνωστό για την αλιεία μαργαριταριών από τον 1ο π.Χ. αιώνα. Το λιμάνι του Τουτικόριν ήταν γνωστό για την αλιεία μαργαριταριών από το βυθό της θάλασσας με χρήση δυτών. Οι Παραβάς (paravas), μια ειδική κάστα από τους Ταμίλ, είχε έδρα της το Τουτικόριν και εκεί αναπτύχθηκε μια πλούσια κοινότητα εξαιτίας της αλιείας και του εμπορίου των μαργαριταριών, των ναυτικών της γνώσεων και των αλιευτικών της σκαφών [1].

Αρχαία Ελλάδα

Αλιευτικές σκηνές σχετικά σπάνια παρουσιάζονται στην Αρχαία Ελληνική τέχνη, μια αντανάκλαση του σχετικά χαμηλού κοινωνικού επιπέδου των ιχθυέων στην Αρχαία Ελληνική Κοινωνία. Έχει βρεθεί πάντως ένας κρατήρας χρονολογημένος γύρω στα 510 - 500 π.Χ., που εικονίζει ένα αγόρι να σκαρφαλώνει πάνω σε ένα βράχο με ένα ράβδο ψάρια στο κεφάλι του και ένα καλάθι στα αριστερά του. Στο νερό κάτω, ένα στρογγυλεμένο αντικείμενο από το ίδιο υλικό με ένα άνοιγμα από πάνω. Αυτό θεωρείται ότι ήταν μια ιχθυοπαγίδα που διατηρούσε τα ψάρια ζωντανά. Σίγουρα δεν είναι από δίχτυ. Αυτό το αντικείμενο βρίσκεται στο Μουσείο Καλών Τεχνών της Βοστώνης, Η.Π.Α.

Ο Έλληνας ιστορικός Πολύβιος περιγράφει την αλιεία ξιφία χρησιμοποιώντας καμάκι με κατάληξη οδοντωτή και μυτερή. Ο Οππιανός της Κόρυκας (Oppian of Corycus), ένας Έλληνας συγγραφέας, συνέγραψε (177 – 180) μια μεγάλη μελέτη για τη θαλάσσια αλιεία, την «Αλιευτική». Είναι η αρχαιότερη γνωστή γραπτή μελέτη για το θέμα που έχει επιβιώσει ως τις μέρες μας. Ο Οππιανός περιγράφει αρκετά μέσα αλιείας, που περιλαμβάνουν τη χρήση διχτυών από αλιευτικά σκάφη, παγιδευτικά (scoop) δίχτυα, συγκρατημένα με στεφάνη, καμάκια και τρίαινες, καθώς και πολλές αλιευτικές παγίδες, «...που λειτουργούν ενώ οι ιδιοκτήτες τους κοιμούνται...». Η περιγραφή του Οππιανού για ψάρεμα με «ακίνητο» δίχτυ είναι επίσης ενδιαφέρουσα: «Οι αλιείς τοποθετούν πολύ ελαφρά δίχτυα από κατεργασμένο λινάρι κυκλικά, καθώς χτυπούν βίαια την επιφάνεια της θάλασσας με τα κουπιά τους και σχηματίζουν μια δίνη γύρω από τις άκρες του κύκλου με το δίχτυ. Από τα χτυπήματα των κουπιών και το σαματά που κάνουν τα ψάρια τρομοκρατούνται και τρέχουν πάνω στο στάσιμο δίχτυ, νομίζοντας ότι είναι καταφύγιο: ανόητα ψάρια που τρομαγμένα από το θόρυβο, εισέρχονται στις πύλες του χαμού τους. Έπειτα οι ψαράδες τραβούν το δίχτυ και από τις δυο πλευρές με σχοινιά και τα οδηγούν στην ακτή.»

Από αρχαίες αναπαραστάσεις και γραπτές πηγές είναι φανερό ότι τα αλιευτικά σκάφη είναι τυπικά μικρά και χωρίς κατάρτι ή ιστίο, κατάλληλα επομένως μόνο για παράκτια αλιεία με σχετική ασφάλεια [1].

Σχήμα 1.2: Γλυπτό του θεού Ποσειδώνα, προστάτη των ναυτικών και των αλιέων, στο λιμάνι της Κοπεγχάγης στη Δανία [1].

Ρώμη

Ενδείξεις Ρωμαϊκής αλιείας βρέθηκαν με τη μορφή μωσαϊκών που δείχνουν αλιεία από βάρκες με πετονιά και με δίχτυα. Αρκετά υδρόβια είδη που αλιεύονταν εικονίζονται, όπως γόγγροι, αστακοί, αχινοί, χταπόδια και σουπιές.[9]. Μια παρωδία αλιείας ήταν ένας τύπος μονομάχου που ονομαζόταν «retiarius» και εμφανίζονταν οπλισμένος με μια τρίαινα κι ένα δίκτυ. Μονομαχούσε συνήθως μ' έναν άλλο τύπο μονομάχου, τον «murmillo», που εμφανίζονταν οπλισμένος με κοντό ξίφος, μικρή ασπίδα και περικεφαλαία με απεικόνιση ενός ψαριού εμπρός [1].

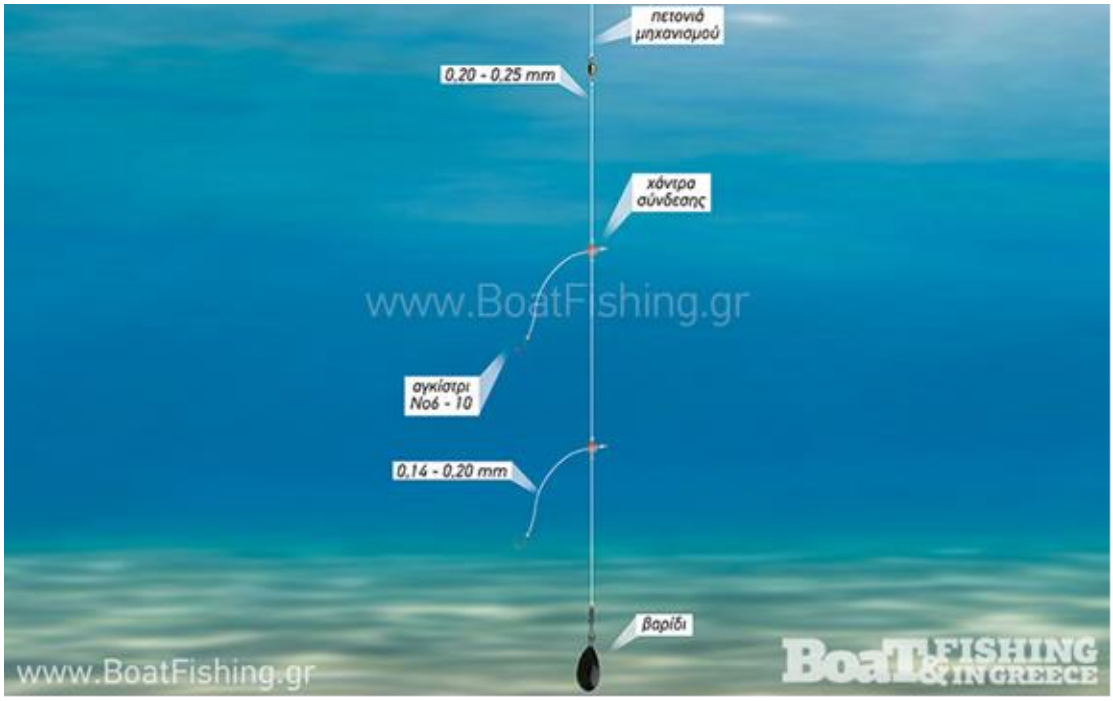
1.3 Τεχνικές ψαρέματος από ακτή

Οι περισσότεροι ψαράδες αδυνατούν να ψαρέψουν από την βάρκα, οπότε συμβιβάζονται με το ψάρεμα από την ακτή. Το ψάρεμα από την ακτή μπορεί να αποδειχθεί ως ένας από τους αποδοτικότερους τρόπους ψαρέματος, αρκεί να επιλέξουμε τον σωστό τρόπο και να ακολουθήσουμε τις σχετικές διαδικασίες. Θα αναφερθούν μερικοί τρόποι με τους οποίους μπορούμε να ψαρέψουμε. Ποιο συγκεκριμένα θα αναφέρουμε τα πράγματα τα οποία θα χρειαστούν για να ετοιμάσουμε το καλάμι.

Casting

Η πλειοψηφία των ερασιτεχνών ψαράδων χρησιμοποιεί αυτή την τεχνική. Τα πράγματα για να υλοποιηθεί αυτή η τεχνική είναι ένα καλάμι με μηχανισμό, μια αρματωσιά, ένα βαρίδιο και δόλωμα.

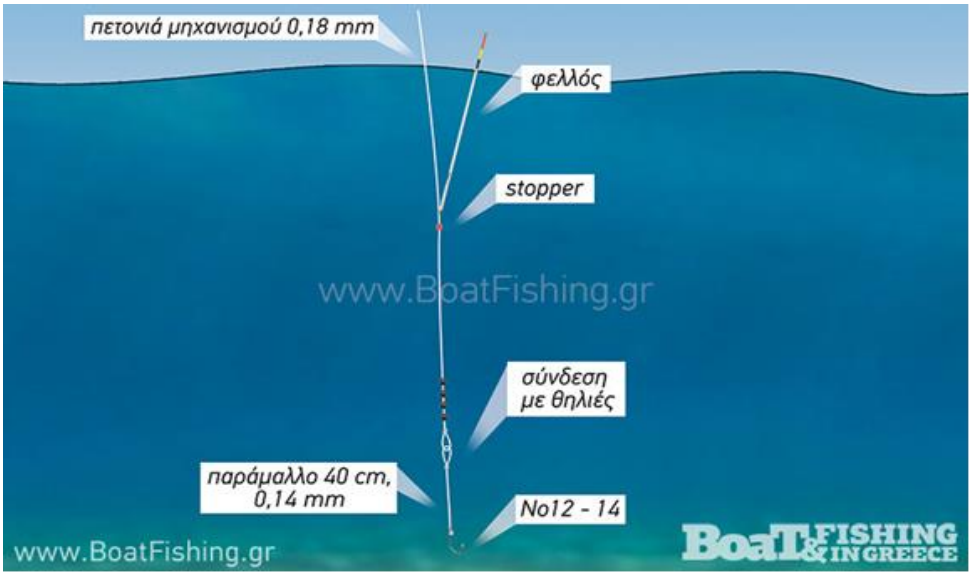
Η σωστή αρματωσιά είναι το μέσο που θα μας δώσει τα πολλά τσιμπήματα. Αλλά και θα καταφέρει να αγκιστρώσει τα ψάρια. Λεπτά γενικά εργαλεία, πετονιές διαμέτρου 0,20 – 0,25 χιλιοστά για μάνα αρματωσιάς και 0,14 – 0,20 χιλιοστά για παράμαλλο, είναι μια καλή επιλογή. Τα αγκίστρια μας καλό είναι να έχουν λεπτό στέλεχος και να χρησιμοποιούνται μικρά σχετικά νούμερα, όπως για παράδειγμα No 6 – 10. Η σύνδεση παράμαλλων - μάνας μμπορεί να γίνει με κόμπο ή με διάφανη χάντρα – σταυρό [3].



Σχήμα 1.3: Αρματωσιά για casting [3].

Εγγλέζικο

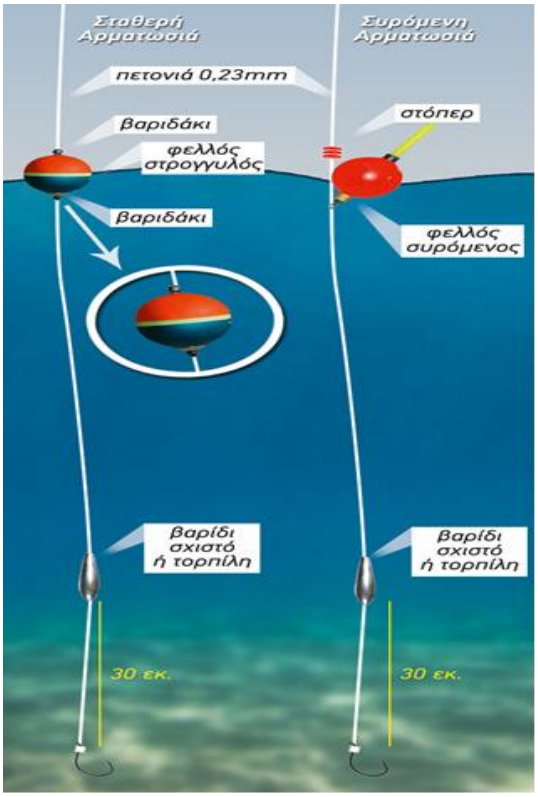
Ένας πρώτης τάξεως τρόπος για να παίρνουμε τσιμπιές και ψάρια. Τα λεπτά εργαλεία δεν τρομάζουν και δεν πονηρεύουν τα ψάρια, και έτσι τσιμπούν με όρεξη τα δολώματά μας. Τον βασικό εξοπλισμό τον οποίο θα χρειαστούμε είναι ένα καλάμι με casting weight 5 – 20 γραμμάρια, μηχανισμό 1000 – 2000, πετονιά μηχανισμού διαμέτρου 0.14 – 0.18 χιλιοστών, αρματωσιά. [3].



Σχήμα 1.4: Αρματωσιά για Εγγλέζικο [3].

Μπολονέζ

Αυτός ο τρόπος ψαρέματος θα μας δώσει εύκολα και σε σύντομο χρονικό διάστημα, πολλά κι εκλεκτά ψάρια. Μπορεί να πραγματοποιηθεί νύχτα σε λιμάνια και ημέρα στα βράχια [4].



Σχήμα 1.5: Αρματωσιά για Μπολονέζ [4].

Spinning

Το spinning είναι ένας εύκολος τρόπος ψαρέματος. Δεν χρειάζεται δόλωμα και περίπλοκο εξοπλισμό. Το μόνο που χρειάζεται είναι ένα καλάμι με μηχανισμό γεμισμένο με νήμα η πετονιά και ένα τεχνητό δόλωμα [3].

1.4 Βοηθητικοί μηχανισμοί ψαρέματος

Μία νέα καινοτομία μηχανημάτων έφτασε στην αγορά και είχε δραματική επίδραση στο τρόπο που ψαρεύουμε σήμερα. Αυτές οι ανακαλύψεις κάνανε πιο προσιτό και ευκολότερο το ψάρεμα για έναν ερασιτέχνη ψαρά.

Τα drone έχουν πολλές χρήσεις. Μερικές από τις χρήσεις τους είναι για στρατιωτικούς (αναγνωριστικούς, κατασκοπευτικούς ή και για μεταφορά βομβών), για ερευνητικούς σκοπούς, για

delivery, για διάσωση, φωτογραφίες και τώρα ξεκίνησε να χρησιμοποιείται και για ψάρεμα. Μερικά από τα πλεονεκτήματα των drone είναι πως έχουν την δυνατότητα να πετάνε σε αρκετά μεγάλες αποστάσεις, δηλαδή πάνω από 300 με 400 μέτρα απόσταση. Αυτό μάς δίνει την δυνατότητα, αν έχουμε μεγάλη πετονιά, να μεταφέρουμε την αρματωσιά σε εξαιρετικά μεγάλες αποστάσεις. Το όφελος στην απόσταση είναι πως θα έχουμε την δυνατότητα να ψαρέψουμε σε μεγαλύτερο βάθος.

Όπως κάθε μηχανισμός, εκτός από τα θετικά του έχει και τα αρνητικά του. Τα αρνητικά του είναι:

1. Οι μπαταρίες στα drone τελειώνουν γρήγορα.
2. Σε περίπτωση που έχει δυνατό αέρα δεν συνιστάται να χρησιμοποιούνται. Παρόλο που υπάρχουν drone που αντέχουν έως 30km/h η μπαταρία τους θα εξαντληθεί ακόμα πιο γρήγορα.
3. Ανάλογα με το drone που διαθέτει ο καθένας μπορεί να σηκώσει και το ανάλογο βάρος. Αν κάποιος ψαρεύει με την μέθοδο casting θα πρέπει να ελέγξει πρώτα το βαρίδιο που θα βάλει αν μπορεί να το σηκώσει το drone [5][6].

Ένας άλλο μηχανισμός για ψάρεμα είναι το rc fishing boat (remote control fishing boat) ή αλλιώς τηλεχειριζόμενο σκάφος για ψάρεμα. Αρκετοί ερασιτέχνες ψαράδες αρχικά αγοράζανε ένα rc boat και προσθέτανε κάτι το οποίο θα ρίχνει την αρματωσιά. Στην συνέχεια κάποιοι χομπίστες ξεκίνησαν να σχεδιάζουν δικά τους σκάφη που θα είναι ειδικά για αυτό το σκοπό. Θα πρέπει να διευκρινιστεί ότι και τα drone και τα rc fishing boat δεν χρησιμοποιούνται για να πιάσουν το ψάρι δεν το επιτρέπει ο νόμος, χρησιμοποιούνται για να μπορέσει ο ερασιτέχνης ψαράς να φτάσει την αρματωσιά σε ένα επιθυμητό βάθος και να το αφήσει. Τα πλεονεκτήματα ενός τέτοιου σκάφους είναι:

1. Μπορείς να το μεταφέρεις την αρματωσιά σε μεγάλο βάθος.
2. Έχουν την δυνατότητα να κουβαλήσουν μεγαλύτερο βάρος σε βαρίδιο απ’ ότι τα drone.
3. Δεν έχουν περιορισμούς, το κάθε σκάφος μπορεί να έχει όσες δυνατότητες θέλει αυτός που το κατασκευάζει. Επειδή φτιάχνετε και πουλιέται κυρίως από ερασιτέχνες μπορεί ο καθένας να βρει αρκετά διαφορετικά σκάφη με διαφορετικές δυνατότητες.

Μερικά από τα μειονεκτήματά του είναι:

1. Σε περίπτωση που πιαστεί κάποιο ψάρι ενώ το σκάφος δεν έχει αφήσει ακόμα την αρματωσιά να πέσει στο νερό, το ψάρι πιθανό να βουλιάξει το σκάφος.
2. Στα περισσότερα σκάφη αν χαθεί από το οπτικό σου πεδίο είναι δύσκολο να το επαναφέρεις στην αρχική του θέση [7].

Κεφάλαιο 2ο: Θεωρητική Προσέγγιση Εξαρτημάτων

Για να μπορέσουμε να κατανοήσουμε την λειτουργία των κυκλωμάτων στα επόμενα κεφάλαια, θα πρέπει να γνωρίζουμε μερικά στοιχειώδη πράγματα για το κάθε εξάρτημα που χρησιμοποιείται σε αυτή την εργασία. Σε αυτό το κεφάλαιο θα εστιάσουμε στο θεωρητικό κομμάτι του κάθε εξαρτήματος.

2.1. Μπαταρίες Li – ion

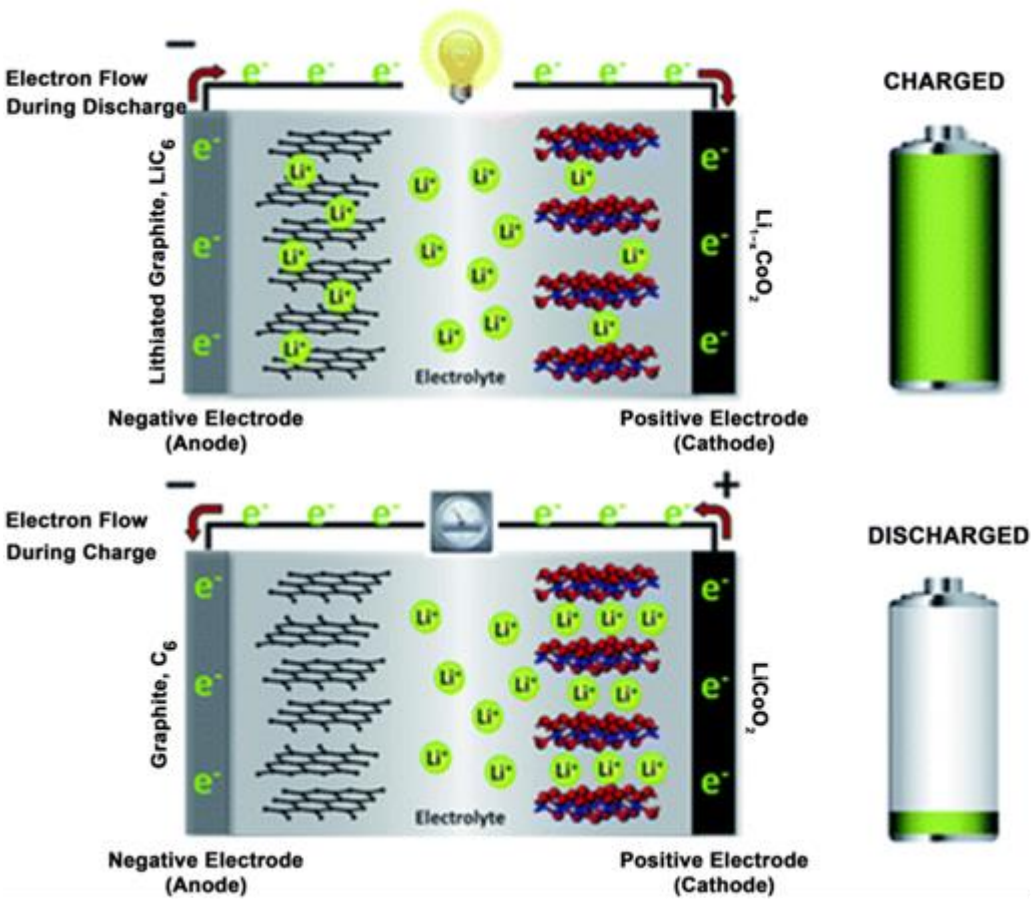
Η μπαταρία ή συσσωρευτής ιόντων λιθίου (αγγλ., lithium – ion battery, Li – ion battery ή LIB) είναι ένας τύπος επαναφορτιζόμενης μπαταρίας στην οποία τα ιόντα λιθίου κινούνται από το αρνητικό ηλεκτρόδιο προς το θετικό ηλεκτρόδιο κατά τη διάρκεια της εκφόρτισης και αντίστροφα κατά τη φόρτιση. Οι μπαταρίες ιόντων λιθίου χρησιμοποιούν μια παρεμβαλλόμενη ένωση του λιθίου ως υλικό του ενός ηλεκτροδίου, συγκρινόμενες με το μεταλλικό λίθιο που χρησιμοποιείται σε μια μη επαναφορτιζόμενη μπαταρία λιθίου. Ο ηλεκτρολύτης, που επιτρέπει την ιονική μετακίνηση και τα δύο ηλεκτρόδια είναι τα συστατικά του στοιχείου μπαταρίας ιόντων λιθίου [8 – 10].

Σχήμα 2.1: Μπαταρίες Li – ion.

Οι μπαταρίες ιόντων λιθίου είναι συνηθισμένες στα οικιακά ηλεκτρονικά. Είναι ένας από τους πιο δημοφιλείς τύπους επαναφορτιζόμενων μπαταριών για φορητά ηλεκτρονικά, με υψηλή ενεργειακή πυκνότητα, πολύ μικρό φαινόμενο μνήμης και χαμηλή αυτοεκφόρτιση. Επίσης πρέπει να προσέχουμε ιδιαίτερα αυτές τις μπαταρίες να μην τις φορτίζουμε υπερβολικά γρήγορα γιατί μπορεί να προκαλέσει βραχυκύκλωμα που θα οδηγήσει σε έκρηξη η πυρκαγιά. Αυτό συμβαίνει επειδή διαθέτουν έναν εύφλεκτο ηλεκτρολύτη. Τέλος, συνιστάται με αυτές τις μπαταρίες να χρησιμοποιούμε μία ασφάλεια.

Κατά τη διάρκεια της εκφόρτισης, τα ιόντα λιθίου (Li+) μεταφέρουν το ρεύμα μες τη μπαταρία από το αρνητικό προς το θετικό ηλεκτρόδιο, μέσω του μη-υδατικού ηλεκτρολύτη και του διαχωριστικού διαφράγματος.

Κατά τη διάρκεια της φόρτισης, μια εξωτερική πηγή ηλεκτρικής ισχύος (το κύκλωμα φόρτισης) εφαρμόζει υπέρταση (υψηλότερη τάση από όσο παράγει η μπαταρία ίδιας πολικότητας), εξαναγκάζοντας το ρεύμα φόρτισης να ρέει μέσα στη μπαταρία από το θετικό προς το αρνητικό ηλεκτρόδιο, δηλαδή κατά αντίθετη κατεύθυνση από το ρεύμα εκφόρτισης σε κανονικές συνθήκες. Τα ιόντα λιθίου τότε μεταναστεύουν από το θετικό προς το αρνητικό ηλεκτρόδιο, όπου ενσωματώνονται στο πορώδες υλικό του ηλεκτροδίου σε μια διεργασία γνωστή ως παρεμβολή [8 – 10].



Σχήμα 2.2: Λειτουργία li – ion [11].

2.1.1 Μπαταρίες LI – PO

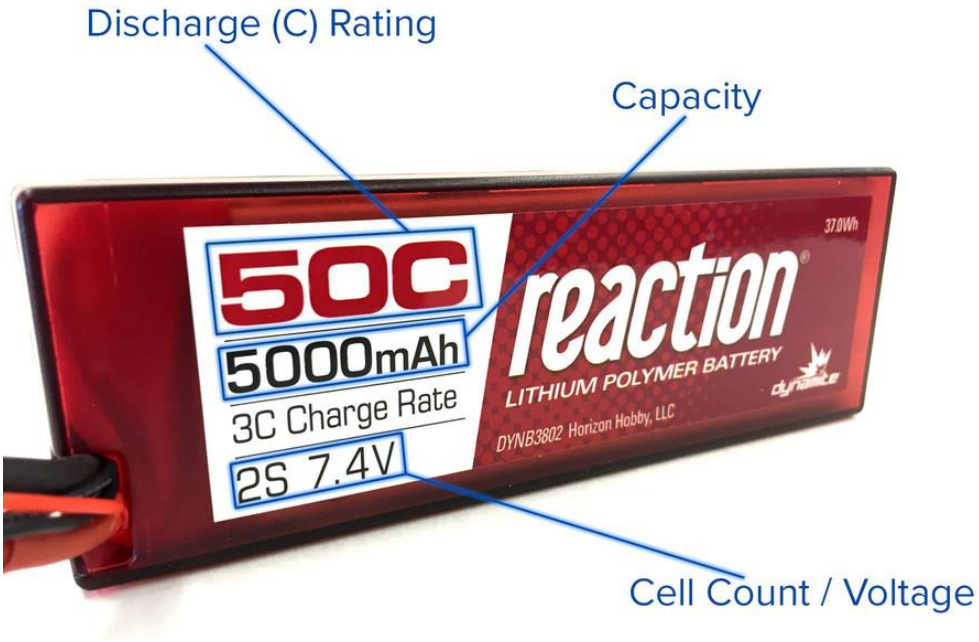
Η li – po είναι μια επαναφορτιζόμενη μπαταρία τεχνολογίας ιόντων λιθίου χρησιμοποιώντας πολυμερή ηλεκτρολύτη αντί υγρού ηλεκτρολύτη. Τα ημιστερεά πολυμερή υψηλής αγωγιμότητας σχηματίζουν αυτόν τον ηλεκτρολύτη. Η πλήρη φόρτιση μίας μπαταρίας li – po είναι 4.2V και η

ελάχιστη επιτρεπόμενη τάση που μπορεί να φτάσει εξαρτάται από τον κατασκευαστή, άλλοτε είναι στα 2.7V η στα 3V [7].

Όταν αγοράσουμε μία μπαταρία li-po θα έχει περίπου την μέση τάση, δηλαδή 3.7V. Πρέπει να είμαστε ιδιαίτερα προσεκτικοί με το χειρισμό αυτών των μπαταριών γιατί μπορούν να εκραγούν. Τα στοιχεία li – po πρέπει να τα φορτίζουμε μόνο με ειδικούς φορτιστές για τέτοιες μπαταρίες. Οι φορτιστές αυτοί είναι χρήσιμοι γιατί τα προστατεύουν κατά την φόρτιση. Η ασφάλεια που προσφέρει είναι:

- Προσέχει να μην ξεπεράσει την μέγιστη τάση δηλαδή 4.2V.
- Αν η τάση είναι κάτω από 2.7V δεν τα φορτίζει γιατί σημαίνει ότι καταστράφηκε η μπαταρία.
- Αν ξεπεράσει τον προβλεπόμενο χρόνο φόρτισης και δεν έχει φτάσει στο μέγιστο η τάση, σταματάει την φόρτιση και θεωρεί την μπαταρία χαλασμένη.
- Μπορούμε να ελέγξουμε αν έχουμε σε μπαταρίες π.χ. 2S αν έχουν την ίδια τάση και οι 2 ώστε να φορτίσουν ίσα [12 – 14].

Οι μπαταρίες li – po αναγράφουν εξωτερικά κάποιους αριθμούς, που βάσει αυτών των αριθμών μπορούμε να αξιολογήσουμε αν μας κάνει αυτή η μπαταρία.



Σχήμα 2.3: Μπαταρία li – po [14].

Όπως αναφέραμε πιο πάνω, η μέση τάση μιας μπαταρίας li – po είναι 3.7V. Το 2S που αναγράφει στην έξοδο σημαίνει ότι είναι 2 μπαταρίες σε σειρά, άρα:

$$3{,}7V + 3{,}7V = 7{,}4V.$$

Ένα ακόμα στοιχείο που μάς δίνει η μπαταρία είναι το ρεύμα που μπορεί να μας δώσει η μπαταρία μέσα σε μία ώρα. Στο Σχήμα 2.3 το ρεύμα αυτό είναι 5000 mAh ή 5 A την ώρα. Όσο μεγαλύτερο ρεύμα διαθέτει μία μπαταρία li-po τόσο μεγαλύτερο θα είναι το μέγεθος και το βάρος της.

Η τελευταία πληροφορία που μας δίνει είναι πόσο γρήγορα μπορεί να αποφορτιστεί η μπαταρία με ασφάλεια χωρίς να χαλάσει. Αυτό μας το δείχνει το 50C, αλλά για να μπορέσουμε να κατανοήσουμε πόσο είναι το μέγιστο φορτίο που μπορεί να χειριστεί το βρίσκουμε με μία μαθηματική πράξη. Αυτή η μαθηματική πράξη ορίζεται ως εξής:

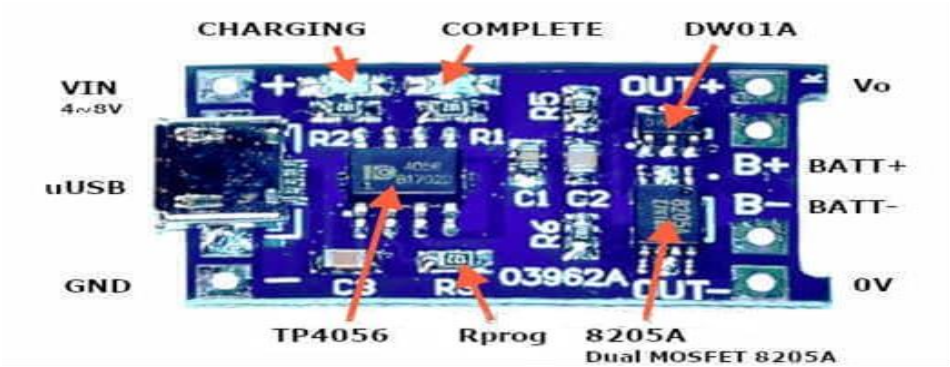
$$50\mathrm{C} = 50 \times \mathrm{C(A)}$$

$$50 \times 5 = 250\mathrm{A}$$

Ο αριθμός που προκύπτει από αυτή την μαθηματική πράξη αντιπροσωπεύει πως αυτή η μπαταρία μπορεί να χειριστεί με ασφάλεια 250 A φορτίο. Σε περίπτωση που βάλουμε παραπάνω από 250 A τότε η μπαταρία στην καλύτερη περίπτωση απλά θα τελειώσει πολύ γρήγορα, αλλιώς πιθανό να πάρει φωτιά [14].

2.2 TP4056

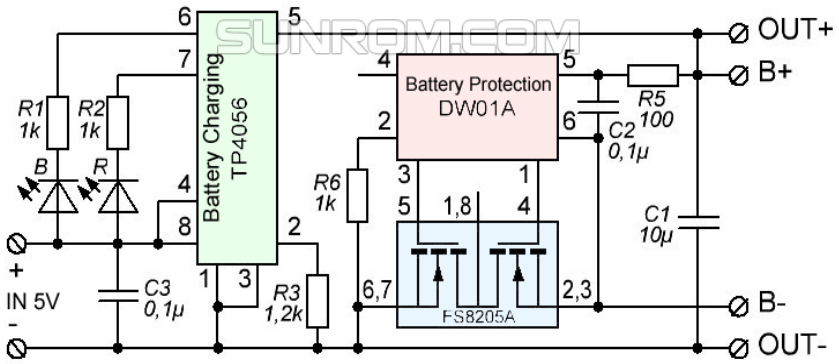
Το TP4056 χρησιμοποιείται για την ασφάλεια των μπαταριών li – ion. Η κύρια δουλειά του είναι να προσέχει να μην αποφορτίσει η μπαταρία, δηλαδή περίπου στα 2.4v αυτόματα θα κόψει την ισχύ εξόδου από την μπαταρία μέχρι να φορτιστεί ξανά η τάση της μπαταρίας πάνω από τα 2.9V. Εάν η τάση της μπαταρίας είναι μικρότερη από 2.9V, η μονάδα θα χρησιμοποιήσει ένα ρεύμα φόρτισης 130mA έως ότου η τάση της μπαταρίας φτάσει τα 2.9V, οπότε το ρεύμα φόρτισης θα αυξηθεί γραμμικά. Επίσης άλλη δυνατότητα που διαθέτει είναι να προσέχει κατά την φόρτιση να σταματήσει την τάση στα 4.2V, ώστε μην πάθει ζημιά η μπαταρία. Άλλη δυνατότητα που διαθέτει είναι να περιορίζει το ρεύμα φόρτισης στο 1 A. Οι μπαταρίες li – ion όπως αναφέραμε αν φορτίσουν με παραπάνω τάση η ρεύμα υπάρχει κίνδυνος έκρηξης η να πάρει φωτιά. Η μονάδα θα κόψει την έξοδο από την μπαταρία εάν ο ρυθμός εκφόρτισης υπερβεί τα 3A ή εάν συμβεί βραχυκύκλωμα. Τέλος, όταν φορτίζουμε την μπαταρία δεν πρέπει ταυτόχρονα να τροφοδοτούμε το κύκλωμα [15].



Σχήμα 2.4: Πλακέτα TP4056 [15].

Η συνδεσμολογία της μπαταρίας με την πλακέτα γίνεται αρκετά εύκολα. Αρχικά κολλάμε ένα καλώδιο στο θετικό μέρος της μπαταρίας και ένα στο αρνητικό. Στην συνέχεια το θετικό μέρος το

κολλάμε στο B+ και το αρνητικό στο B-. Στο OUT+ από εκεί βγαίνει η τάση ώστε να τροφοδοτήσουμε το κύκλωμα το οποίο θέλουμε και το OUT- η γείωση. Στο IN συνδέουμε τον φορτιστή όταν θέλουμε να φορτίσουμε την μπαταρία, εκτός αν διαθέτουμε micro usb οπότε μπορούμε απευθείας να το τροφοδοτήσουμε από εκεί [15].

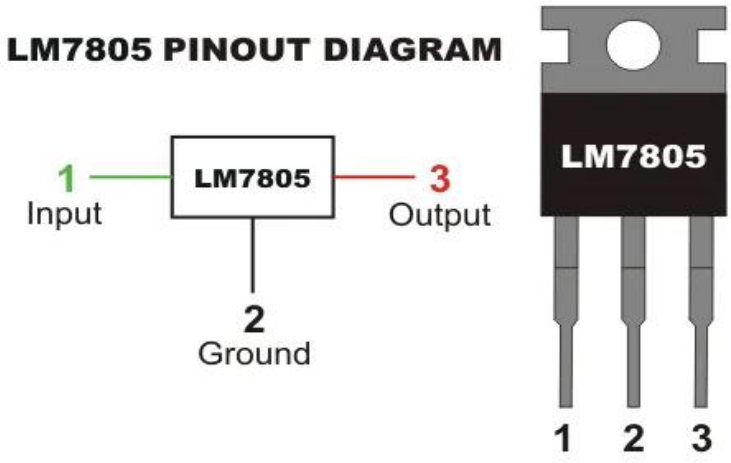


Σχήμα 2.5: Κύκλωμα TP4056 [16].

2.3 Voltage regulator

Το voltage regulator είναι ένα σύστημα που έχει σχεδιαστεί για να διατηρεί αυτόματα μια σταθερή τάση. Τα ολοκληρωμένα κυκλώματα που χρησιμοποιούνται για την ρύθμιση της τάσης ονομάζονται ICs.

Ένα παράδειγμα voltage regulator είναι ο ICs 7805. Δέχεται στην είσοδό του από 7 έως 35V και στην έξοδό του βγάζει 5V [17].



Σχήμα 2.6: Voltage regulator [17].

2.4. Κινητήρες

Στην σημερινή εποχή υπάρχουν δύο διαφορετικά είδη ηλεκτρικών κινητήρων. Τα δύο είδη είναι οι ηλεκτρικοί κινητήρες εναλλασσόμενου ρεύματος AC (Alternate Current) και οι κινητήρες συνεχούς ρεύματος DC (Direct Current).

Οι ηλεκτρικοί κινητήρες εναλλασσόμενου χωρίζονται σε δύο υποκατηγορίες. Τα σύγχρονα και τα ασύγχρονα. Η τροφοδοσία των AC κινητήρων γίνεται με 220V ή 380V.

Οι υποκατηγορίες που χωρίζονται οι DC κινητήρες είναι:

- Απλοί ηλεκτροκινητήρες με ψήκτρες DC.
- Βηματικοί κινητήρες (Stepper motors).
- Ηλεκτροκινητήρες χωρίς ψήκτρες (BrushLess DC) [18].

2.4.1. Ηλεκτροκινητήρες με ψήκτρες DC

Οι ηλεκτρικοί κινητήρες δεν είναι τίποτα άλλο από μία διάταξη που χρησιμοποιείται για την μετατροπή της ηλεκτρικής ενέργειας σε μηχανική ενέργεια. Η αρχή λειτουργίας των περισσότερων ηλεκτροκινητήρων βασίζεται στην αλληλεπίδραση ανάμεσα σε δυο φορείς ηλεκτρομαγνητικών πεδίων που έχουν την τάση να προσανατολίζονται μεταξύ τους. Το αποτέλεσμα αυτής της αλληλεπίδρασης είναι η δημιουργία δύναμης και στην συνέχεια ροπής πάνω στον άξονα του ηλεκτροκινητήρα.

Όταν ένας αγωγός μήκους L διαρρέεται από ρεύμα I, ενώ αυτός βρίσκεται μέσα σε μαγνητικό πεδίο B, τότε παράγεται μια δύναμη F σύμφωνα με το σχήμα και τον κανόνα του δεξιού χεριού κάθετη προς το ρεύμα και προς το μαγνητικό πεδίο. Η δύναμη αυτή είναι ανάλογη του μήκους του αγωγού, της έντασης του ρεύματος και της έντασης του μαγνητικού πεδίου σύμφωνα με τη σχέση:

$$F = I \times L \times B \times \eta\mu\theta \tag{2.1}$$

Αυτοί οι κινητήρες επιλέγονται επειδή είναι αρκετά φθηνοί και επειδή είναι ο εύκολος έλεγχος της ταχύτητας τους αυξομειώνοντας την τάση τροφοδοσίας. Τα μειονεκτήματα ενός τέτοιου κινητήρα είναι οι ψήκτρες τους φθείρονται και απαιτείται περιοδικά αντικατάσταση. Επίσης άλλο ένα μειονέκτημα είναι πως στις χαμηλές ταχύτητες είναι δύσκολος ο έλεγχος τους [19][20].

2.4.2 Βηματικοί κινητήρες (Stepper motors)

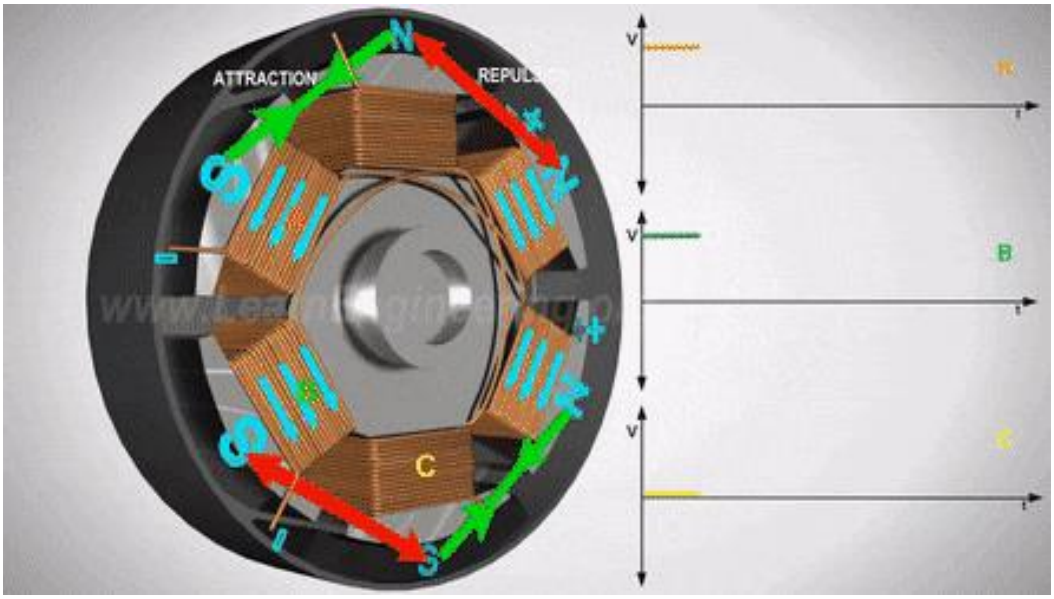
Οι βηματικοί κινητήρες είναι ένας κινητήρας επαγωγής ο οποίος είναι σταδιακών βημάτων. Για να μπορέσει να λειτουργήσει χρειάζεται ένα κύκλωμα το οποίο θα το τροφοδοτεί με παλμούς. Ο κάθε παλμός που θα δέχεται ανάλογα με την λειτουργία που θα κάνει μισό ή ένα βήμα. Το κύριο χαρακτηριστικό ενός τέτοιου κινητήρα είναι ότι δεν κινείται ανεξέλεγκτα αλλά σε διακριτές θέσεις και η μονάδα μέτρησης των θέσεων είναι τα βήματα. Ένας κινητήρας που μπορεί να κάνει περισσότερα βήματα είναι πιο ακριβής. Δηλαδή π.χ. έχουμε έναν κινητήρα 600 θέσεων, θα είναι πιο ακριβής από έναν των 100 θέσεων. Όταν αναφέρουμε βήματα εννοούμε, πόσα βήματα θα κάνει ο ρότορας για μία πλήρη περιστροφή.

Τα πλεονεκτήματα των βηματικών κινητήρων είναι ότι ο έλεγχός τους είναι σχετικά εύκολος και φθηνός. Μία πλακέτα Arduino θα μπορούσε να κάνει την οδήγηση αυτού του κινητήρα. Άλλο πλεονέκτημα είναι ότι δεν χρειάζεται φρένο, μπορούμε να το σταματήσουμε αμέσως με την διακοπή του παλμού. Τέλος, κατά την εκκίνηση και στις χαμηλές ταχύτητες παράγουν σχετικά μεγάλη ροπή. Τα μειονεκτήματα ενός τέτοιο κινητήρα είναι ο θόρυβος που κάνει κατά την λειτουργία του. Σε σύγκριση με τους DC κινητήρες έχουν περιορισμένη ταχύτητα. Γενικά, οι κινητήρες αυτοί χρησιμοποιούνται μόνο σε εφαρμογές ελέγχου θέσης χαμηλής ισχύος [21].

2.4.3 Ηλεκτροκινητήρες χωρίς ψήκτρες (BrushLess DC)

Τα τελευταία χρόνια ξεκίνησαν να αναπτύσσονται οι ηλεκτρονικοί κινητήρες χωρίς ψήκτρα. Αυτό συνέβη λόγο των προβλημάτων που δημιουργούταν με τις ψήκτρες στους η DC κινητήρες με ψήκτρες . Η λειτουργία ενός brushless κινητήρα είναι περίπου κοινή με έναν κινητήρα με ψήκτρα. Η μόνη διαφορά τους είναι ο τρόπος εναλλαγής της πολικότητας. Ένας κινητήρας DC χρησιμοποιεί ψήκτρες για αυτό το σκοπό, αλλά φθείρονται γρήγορα. Έτσι αναπτύχθηκαν οι κινητήρες χωρίς ψήκτρες οι οποίοι για να κάνουν την εναλλαγή πολικότητας χρησιμοποιούνται αισθητήρες Hall για την αίσθηση της θέσης του ρότορα και μιας πλακέτας κυκλώματος που λειτουργεί ως διακόπτης. Τα σήματα των αισθητήρων υποβάλλονται σε επεξεργασία από την ψηφιακό κύκλωμα, το οποίο ελέγχει με ακρίβεια τη σωστή στιγμή εναλλαγής καθώς ο ρότορας περιστρέφεται.

Οι ηλεκτρικοί κινητήρες brushless δεν χρειάζονται συντήρηση, έχουν πολύ καλύτερο βαθμό απόδοσης από τους κινητήρες DC. Για να μπορέσουν να λειτουργήσουν χρειάζονται έναν οδηγό ESC (Electronic Speed Controller) και PWM που μπορούμε να το πάρουμε από έναν μικροελεγκτή. Αυτό το σήμα PWM θα δίνει το σήμα στο ESC και από εκεί θα ελέγχει την ταχύτητα του κινητήρα [18][22][23].



Σχήμα 2.7: Λειτουργία BrushLess κινητήρα και σήμα ESC [24].

Οι κινητήρες brushless έχουν αρκετά πλεονεκτήματα σε σχέση με τους άλλους κινητήρες. Τα πλεονεκτήματα αυτά είναι:

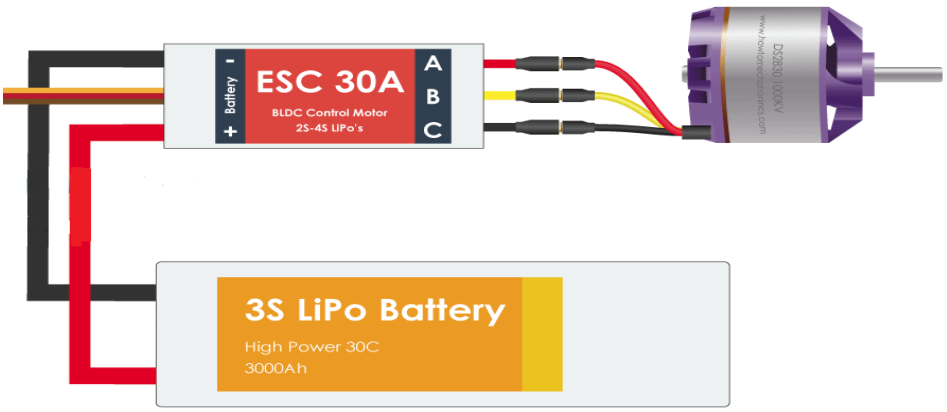
1. Επιτυγχάνουν πολύ υψηλές ταχύτητες.
2. Πιο ανθεκτικό, καθώς δεν χρειάζεται τόση συντήρηση.
3. Οι κινητήρες χωρίς ψήκτρες είναι πιο αποδοτικοί καθώς η ταχύτητά τους καθορίζεται από τη συχνότητα με την οποία τροφοδοτείται ρεύμα, όχι από την τάση.
4. Δεν υπάρχει σπινθήρισμα και πολύ λιγότερο θόρυβος κατά τη λειτουργία.
5. Οι κινητήρες brushless επιταχύνουν και επιβραδύνουν εύκολα καθώς έχουν χαμηλή αδράνεια του ρότορα.
6. Οι κινητήρες brushless επιταχύνουν και επιβραδύνουν εύκολα καθώς έχουν χαμηλή αδράνεια του ρότορα.

Ενώ ορισμένα μειονεκτήματά τους είναι τα παρακάτω:

1. Για να το ελέγξετε, θα χρειαστείτε προγράμματα οδήγησης ή ελεγκτές, ώστε να μπορείτε να ελέγχετε την περιστροφή. Είναι αδύνατο να το κάνουμε χειροκίνητα όπως και σε άλλες περιπτώσεις.
2. Είναι ακριβοί [18][23].

2.5. Electronic Speed Control (ESC)

Το ESC (Electronic speed control) η αλλιώς έλεγχος ταχύτητας είναι μία συσκευή που ρυθμίζει την ισχύ ενός ηλεκτροκινητήρα, επιτρέποντας του να ρυθμίσει το γκάζι από 0% έως 100%. Ένα esc αποτελείται από τρία μέρη. Το πρώτο μέρος είναι ρυθμιστής BEC (Battery Elimination Circuit), το BEC το χρησιμοποιούμε για να μπορέσει να τροφοδοτήσει το ESC τον δέκτη. Η ισχύς που βγάζουν τα BEC είναι 5V και 1Ah. Το δεύτερο μέρος είναι ένας επεξεργαστής που η δουλειά του είναι να μεταφράζει τις πληροφορίες που τού δίνονται από τον δέκτη και να εναλλάσσει στο τρίτο μέρος, δηλαδή τα FETs ώστε να ρυθμίζουν την ισχύ του κινητήρα. Με λίγα λόγια τα FET είναι ένας ηλεκτρονικός διακόπτης που μειώνει τη ροή ηλεκτρικής ενέργειας που με τη σειρά του πάει στον κινητήρα [25].

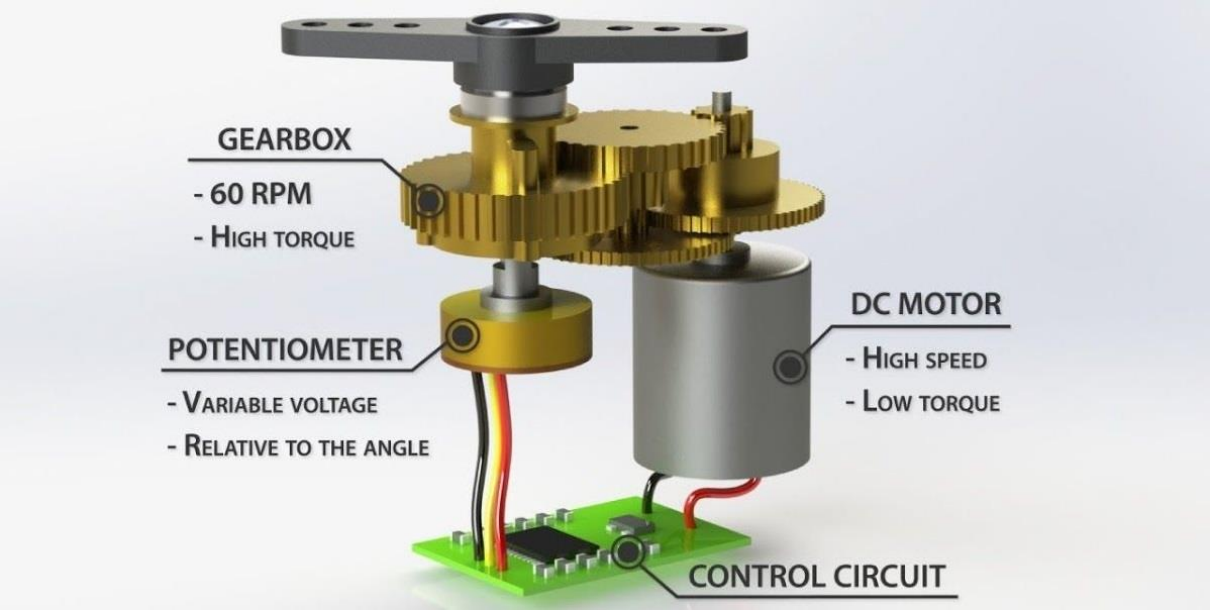


Σχήμα 2.8: Συνδεσμολογία ESC με μπαταρία και κινητήρα [26].

2.6. Σερβοκινητήρας

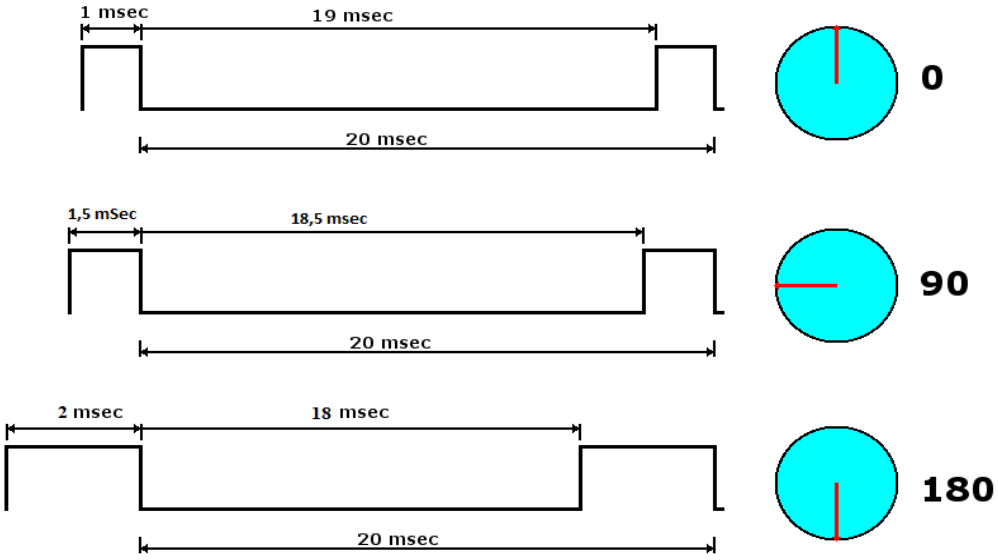
Οι σερβοκινητήρες είναι ένας μηχανισμός που μπορεί να γυρίζει με ακρίβεια έναν άξονα από μηδέν έως 180 μοίρες. Οι σερβοκινητήρες υπάρχουν εδώ και πολύ καιρό και χρησιμοποιούνται σε πολλές εφαρμογές. Χρησιμοποιούνται για τηλεχειριζόμενα παιχνίδια, ρομπότ, drone και σε βιομηχανικές εφαρμογές.

Στο εσωτερικό του σερβοκινητήρα υπάρχει ένας μικρός κινητήρας DC , ποτενσιόμετρο και ένα κύκλωμα ελέγχου. Ο κινητήρας συνδέεται με τα γρανάζια στον τροχό ελέγχου. Καθώς ο κινητήρας περιστρέφεται, η αντίσταση του ποτενσιόμετρου αλλάζει , έτσι το κύκλωμα ελέγχου μπορεί να ρυθμίσει με ακρίβεια πόση κίνηση υπάρχει και σε ποια κατεύθυνση [27][28].



Σχήμα 2.9: Ο σερβοκινητήρας εσωτερικά [27].

Κάθε σερβοκινητήρας διαθέτει τρία καλώδια. Το κόκκινο είναι το VCC το μαύρο η γείωση και το πορτοκαλί / κίτρινο θα συνδέεται στον επεξεργαστή ο οποίος θα του δίνει το σήμα PWM. Για να γίνει ο έλεγχος του σερβοκινητήρα θα πρέπει ο επεξεργαστής να στέλνει ένα επαναλαμβανόμενο παλμό κάθε 20msec. Ανάλογα με το μήκος του παλμού ρυθμίζεται η γωνία άξονα.



Σχήμα 2.10: Αντιστοιχία παλμών σε μοίρες [29].

2.7. LORA

Η τεχνολογία LoRa (LongRange) αναπτύχθηκε από τον Cycleo της Γκρενόμπλ της Γαλλίας και αποκτήθηκε από τον Semtech και αποτελεί ένα νέο ασύρματο πρωτόκολλο σχεδιασμένο για επικοινωνία μεγάλης εμβέλειας και χαμηλής ισχύος. Η LoRa είναι μια τεχνολογία chirp-based διασποράς φάσματος (spread-spectrum), με μεγαλύτερο εύρος ζώνης από αυτό των ανταγωνιστών της. Λόγω της τεχνικής διαμόρφωσης και της ενσωματωμένης δυνατότητας διόρθωσης σφαλμάτων (forward error correcting, FEC), το σήμα LoRa μπορεί να μεταδώσει δεδομένα με ισχύ σήματος αρκετά κάτω από το επίπεδο θορύβου περιβάλλοντος (noise floor), επιτρέποντας με αυτόν τον τρόπο πολύ μεγάλες αποστάσεις επικοινωνίας. Ακόμα, προσφέρει αποτελεσματική αμφίδρομη λειτουργικότητα. Έτσι ως λύση είναι αποτελεσματική για τη λήψη μηνυμάτων από τελικά σημεία (endpoints), αλλά και για την αποστολή μηνυμάτων από σταθμούς βάσης σε τελικά σημεία (όπως για εφαρμογές εντολών και ελέγχου) [30 – 32].

Σχήμα 2.11: Λογότυπο Lora [33].

Το LoRa χρησιμοποιεί ζώνες ραδιοσυχνοτήτων που δεν χρειάζονται άδεια όπως το 433MHZ, 868MHZ (Ευρώπη), 915MHZ (Αυστραλία και Βόρεια Αμερική), 865 MHz έως 867 MHz (Ινδία) και 923 MHz (Ασία) με ρυθμό μετάδοσης που κυμαίνεται από 0.25 kbps έως 50kbps. Η αμφίδρομη επικοινωνία παρέχεται από την τεχνική διαμόρφωσης, που είναι παράγωγο της chirp διασποράς φάσματος (Chirp Spread Spectrum, CSS), που απλώνει ένα σήμα στενής ζώνης σε ένα μεγαλύτερο εύρος ζώνης καναλιού [30].

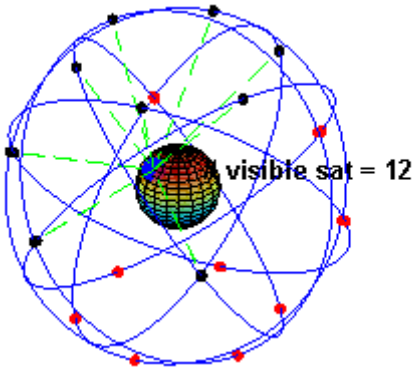
2.8 GPS

Η ονομασία GPS προέρχεται από τα αρχικά της φράσης "Global Positioning System", που μεταφράζεται ως σύστημα παγκόσμιας τοποθέτησης. Είναι γνωστό επίσης ως Navstar GPS ή απλά Navstar. Το GPS project ξεκίνησε να αναπτύσσεται το 1973, από το Αμερικανικό Υπουργείο Άμυνας. Στόχος του project ήταν να ξεπεράσει τους περιορισμούς των προηγούμενων επίγειων συστημάτων, χρησιμοποιώντας δορυφόρους για την πλοήγηση. Ο Αμερικανός επιστήμονας Roger Lee Easton ήταν ο επικεφαλής εφευρέτης και σχεδιαστής, και θεωρείται ο πατέρας του GPS. Για την ανάπτυξη του συστήματος συνεργάστηκε με τους Ivan A. Getting και Bradford Parkinson. Ο πρώτος πειραματικός δορυφόρος του συστήματος, NAVSTAR 1, εκτοξεύτηκε στις 22/2/1978, με ένα πύραυλο τύπου Atlas F. Το Δεκέμβριο του 1993 συμπληρώθηκαν 24 δορυφόροι σε τροχιά. Αυτός είναι και ο ελάχιστος αριθμός δορυφόρων που είναι απαραίτητοι για την πλοήγηση GPS.

Όταν μια συσκευή μπαίνει στο gps π.χ. το κινητό, η επικοινωνία είναι μονόδρομη. Δηλαδή η συσκευή λαμβάνει μόνο από τους δορυφόρους αλλά δεν μπορεί να στείλει πίσω δεδομένα. Αν θέλαμε να φτιάξουμε κάτι το οποίο θα κάνει αμφίδρομη επικοινωνία τότε θα χρειαζόταν πολύ μεγαλύτερη κεραία, όπως αυτές που έχουν τα δορυφορικά τηλέφωνα [34][35].

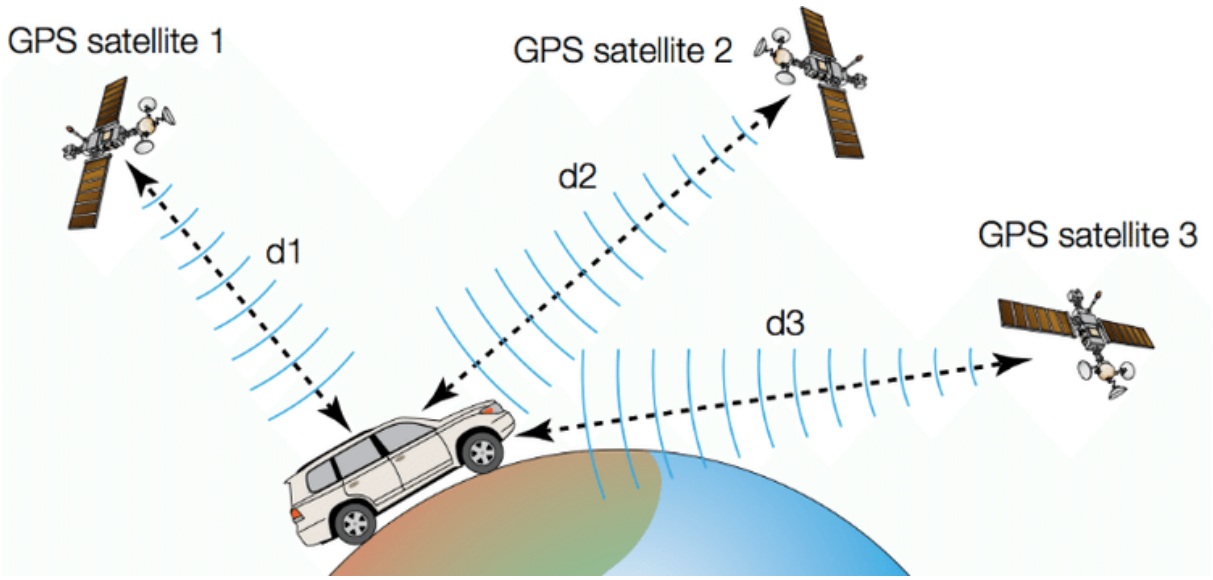
Σχήμα 2.12: Δορυφορικά τηλέφωνα [34].

Οι δορυφόροι του gps έχουν τοποθετηθεί σε πολύ συγκεκριμένα σημεία, ώστε ανά πάσα στιγμή να είναι ορατοί τουλάχιστον τέσσερις σε οποιοδήποτε σημείο της γης. Κάθε δορυφόρος εκπέμπει διαρκώς ραδιοκύματα προς την γη. Οι πληροφορίες που λαμβάνουμε είναι τη θέση του δορυφόρου και την ώρα την οποία στάλθηκε το σήμα.



Σχήμα 2.13: Αναπαράσταση του αρχικού σχεδίου του συστήματος GPS [35].

Το κινητό μας ή η συσκευή GPS συγκρίνει την ώρα που έλαβε το σήμα με την ώρα που έστειλε ο δορυφόρος. Από αυτή τη διαφορά ώρας, και γνωρίζοντας την ακριβή ταχύτητα των ραδιοκυμάτων στην ατμόσφαιρα, ο δέκτης μπορεί να υπολογίσει την ακριβή του απόσταση από καθέναν από τους δορυφόρους. Με αυτή την πληροφορία, ο δέκτης GPS μπορεί να βρει ακριβώς που βρίσκεται, με τη μέθοδο του τριπλευρισμού. Ο τριπλευρισμός (trilateration) είναι μία μέθοδος της Γεωμετρίας, χωρίς την οποία δεν θα ήταν δυνατή η πλοήγηση GPS [34].



Σχήμα 2.14: Εντοπισμός θέσης gps [34].

2.9 Μικροελεγκτής

Με τον όρο μικροελεγκτή εννοείται ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (π.χ. τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.

Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους απαντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard,

η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (π.χ. οι σειρές AVR από την Atmel και PIC από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου φον Νόιμαν.

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (π.χ. I^2C, SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC, ISDN, ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.
- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS – 232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.
- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, π.χ. κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση [36].

Κεφάλαιο 3ο: Ανάλυση Τηλεχειριστηρίου

Σε αυτό το κεφάλαιο θα ασχοληθούμε με το ηλεκτρονικό μέρος του τηλεχειριστηρίου, δηλαδή την πλακέτα που φτιάξαμε μαζί με τα εξαρτήματα που χρησιμοποιήσαμε. Η ανάλυση θα γίνει σε μικρές βαθμίδες μέχρι να φτάσουμε στο τελικό κύκλωμα.

3.1 Τροφοδοσία

Το κύκλωμα του τηλεχειριστηρίου αποτελείται από τρείς διαφορετικές τάσεις. Η κύρια τροφοδοσία του κυκλώματος γίνεται από δύο μπαταρίες Li – ion. Χρησιμοποιήσαμε μπαταρίες Li – ion επειδή δίνουν πολύ περισσότερο ρεύμα απ' ότι χρειάζεται το κύκλωμα μας και επίσης είναι φθηνές και επαναφορτιζόμενες. Αρχικά συνδέσαμε την κάθε μία μπαταρία με την ασφάλεια TP4056, έτσι η μπαταρία όταν πέσει κάτω από 2.6V η ασφάλεια θα σταματήσει την τροφοδοσία του κυκλώματος. Επίσης κατά την φόρτιση δεν θα χρειάζονται συχνή παρακολούθηση, το TP4056 έχει δύο led πάνω του που δείχνουν σε τι κατάσταση βρίσκεται, όταν δεν είναι φορτισμένες πλήρως είναι αναμμένο το κόκκινο led. Όταν είναι 100% η φόρτιση τότε ανάβει το μπλε led και στην συνέχεια αφού φτάσει 100% θα διακόψει την φόρτιση.

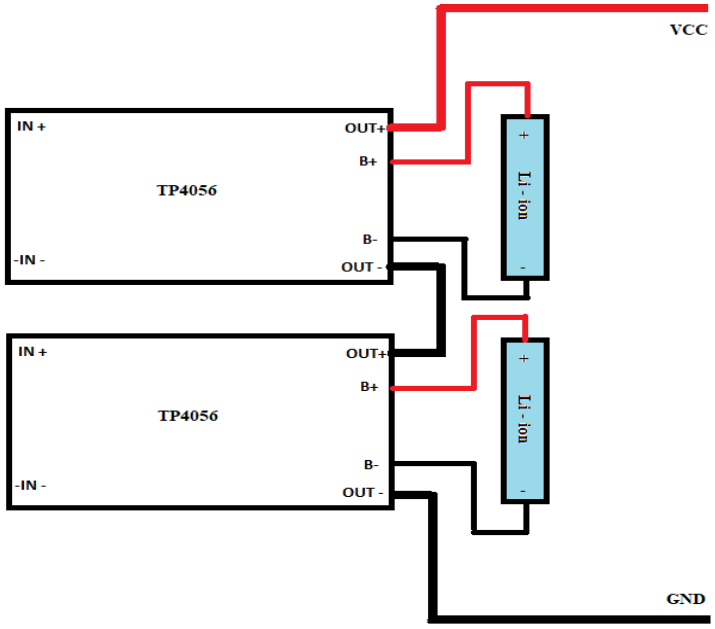
Μετά την σύνδεση των Li – ion με τα TP4056 παίρνουμε από την μία ασφάλεια το OUT- και το συνδέουμε σε σειρά με την άλλη ασφάλεια το OUT+. Όπως γνωρίζουμε δύο μπαταρίες όταν είναι σε σειρά η τάση προστίθεται, άρα η μέγιστη τάση που θα λαμβάνει η πλακέτα μας είναι:

$$4.2V + 4.2V = 8.4V$$

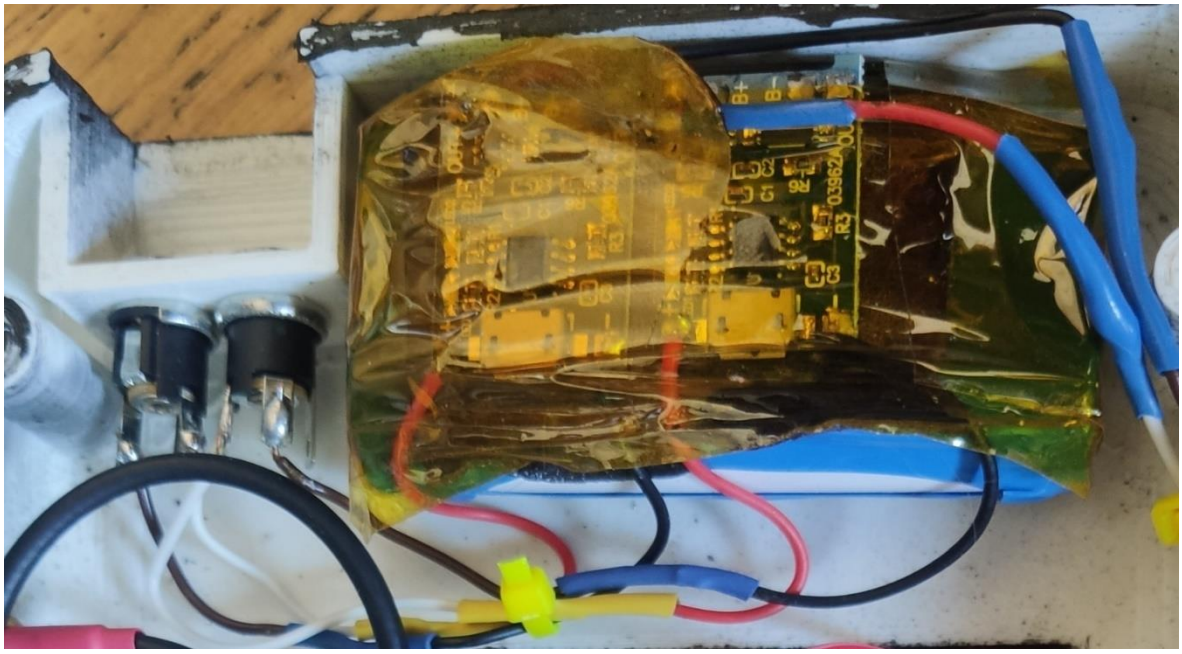
και η ελάχιστη τάση είναι:

$$2.6V + 2.6V = 5.2V$$

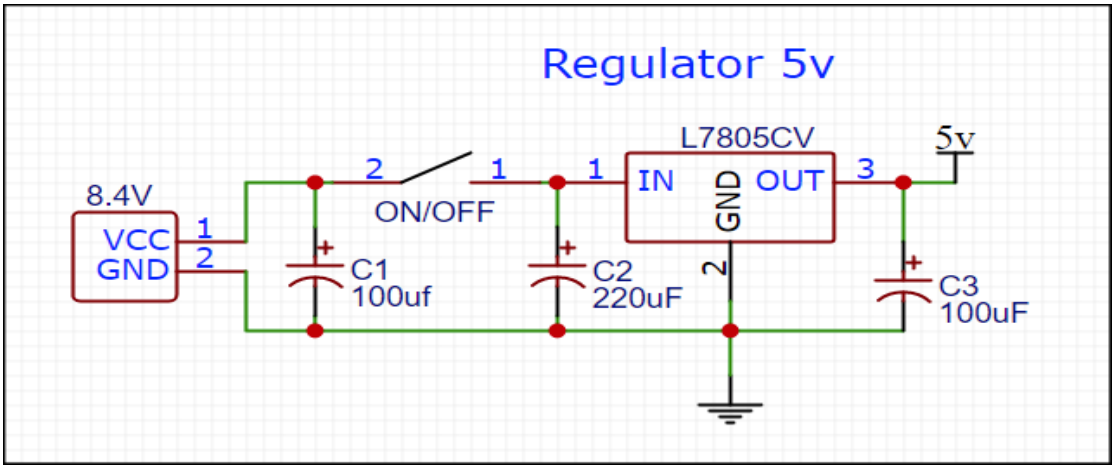
Με την ολοκλήρωση της συνδεσμολογίας συνδέσαμε το VCC στο + της πλακέτας και το GND στην γείωση της πλακέτας. Επίσης συνδέσαμε το IN+ και IN- σε θηλυκά φισάκια ώστε να υπάρχει η δυνατότητα να φορτίζουμε τις μπαταρίες χωρίς να ανοίγουμε το κουτί. Όλο το κύκλωμα με τις μπαταρίες και το TP4056 το τυλίξαμε με θερμοσυστελλόμενο υλικό για να αποφύγουμε μελλοντικά βραχυκυκλώματα. Το θερμοσυστελλόμενο είναι ένα υλικό το οποίο όσο το ζεσταίνουμε αυτό κλείνει και χρησιμοποιείται για να μονώσει γυμνά σημεία σε καλώδια ή σε πλακέτες.



Σχήμα 3.1: Συνδεσμολογία TP4056 με μπαταρίες Li – ion.



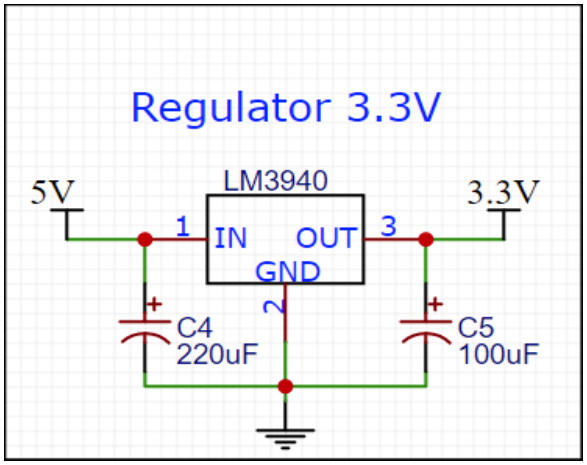
Σχήμα 3.2: Τελικό αποτέλεσμα τροφοδοσίας.



Σχήμα 3.3: Σταθεροποιητής 5V.

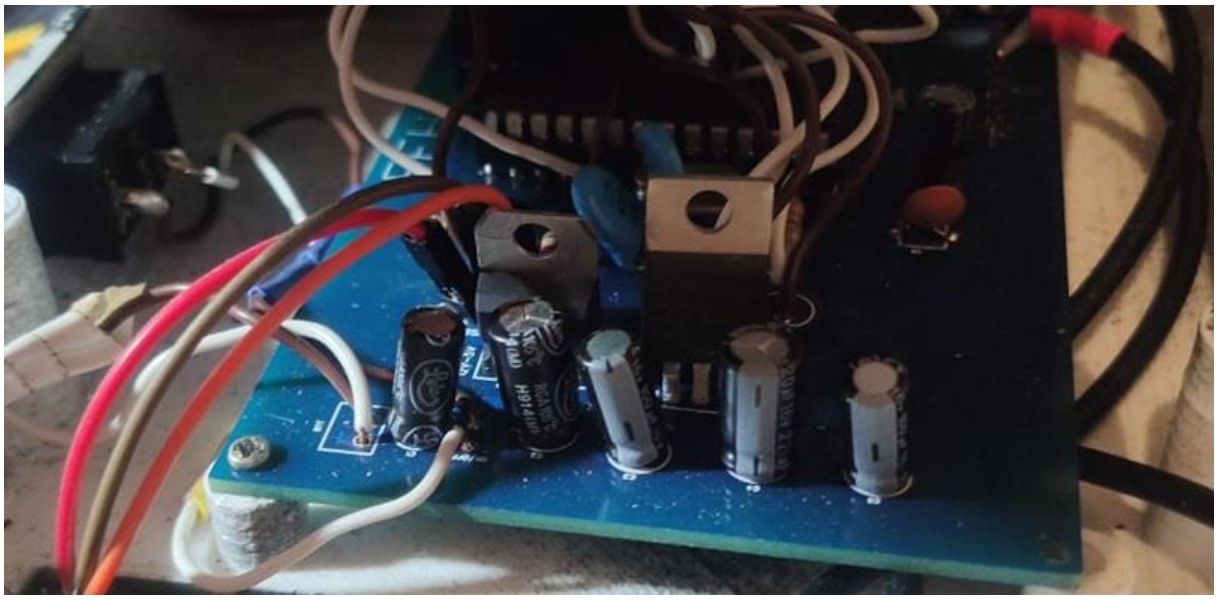
Μετά την τροφοδοσία του κυκλώματος χρησιμοποιήσαμε παράλληλα έναν ηλεκτρολυτικό πυκνωτή C1 100μF. Αυτός ο πυκνωτής χρησιμοποιείται γιατί έχουμε βραχυκυκλωτήρες και ο κάθε βραχυκυκλωτήρας περιέχει μία εσωτερική αντίσταση. Έτσι υπάρχει περίπτωση να μην μπορέσουμε να πάρουμε άμεσο ρεύμα και βάζοντας αυτό το πυκνωτή λύνεται αυτό το πρόβλημα. Στην συνέχεια τοποθετήθηκε ένας διακόπτης με την ένδειξη ON/OFF. Όταν είναι OFF η τάση θα μείνει στην άκρη του διακόπτη, ενώ όταν γίνει ON θα περάσει κανονικά άρα θα λειτουργεί το κύκλωμα.

Η τάση μετά τον διακόπτη θα καταλήξει σε έναν σταθεροποιητή τάσης τον L7805CV ο οποίος στην έξοδο θα μας δώσει 5V. Ο σταθεροποιητής χρησιμοποιεί δύο ηλεκτρολυτικούς πυκνωτές, έναν στην είσοδο C2 220μF και έναν στην έξοδο C3 100μF. Ο πυκνωτής C2 ουσιαστικά χρησιμοποιείται για να γειώσει αν πάει να μπει οποιοσδήποτε θόρυβος και καθαρίζει την είσοδο του σταθεροποιητή. Ο πυκνωτής C3 καθαρίζει περαιτέρω το σήμα DC, ώστε να έχουμε μια καθαρή πηγή 5V. Έτσι δεν μας επηρεάζει η αυξομείωση των μπαταριών Li – ion στην είσοδο του σταθεροποιητή, ο σταθεροποιητής και στην μέγιστη και στην ελάχιστη τάση στην είσοδο θα μας δώσει 5V.



Σχήμα 3.4: Σταθεροποιητής 3.3V.

Η τελευταία βαθμίδα της τροφοδοσίας είναι ο σταθεροποιητής τάσης LM3940. Αυτός ο σταθεροποιητής έχει την δυνατότητα να δέχεται στην είσοδο από 4.5V έως 5.5 και στην έξοδο του μας δίνει 3,3V. Η είσοδος του αυτού του σταθεροποιητή θα συνδέεται με την έξοδο του L7805CV. Στην είσοδο ο ηλεκτρολυτικός πυκνωτής C4 220μF και στην έξοδο ο ηλεκτρολυτικός πυκνωτής C5 100μF κάνουν ακριβώς την ίδια δουλειά με τον προηγούμενο σταθεροποιητή.



Σχήμα 3.5: Τελικό κύκλωμα τροφοδοσίας.

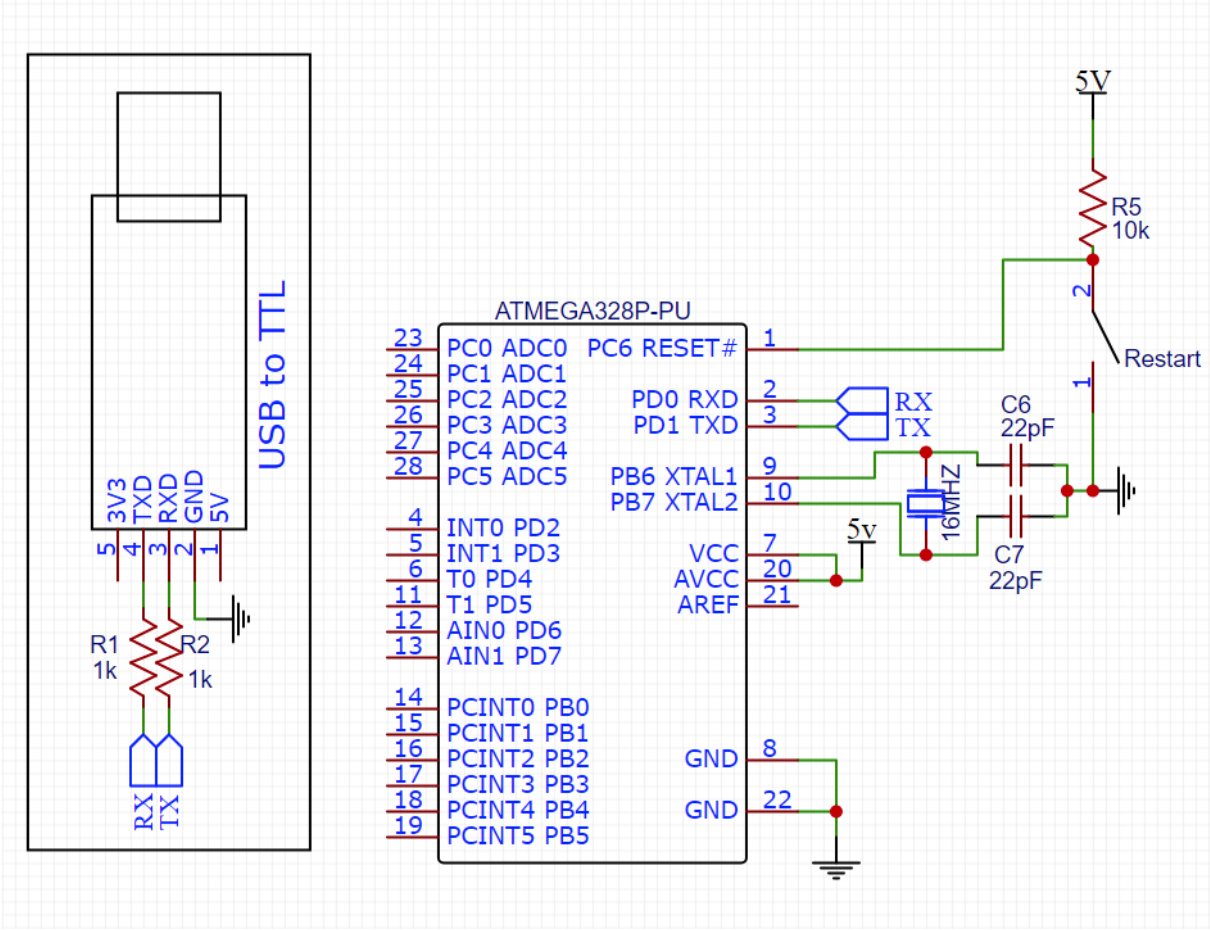
3.2 Atmega328P – PU

Το βασικό μέρος του τηλεχειριστηρίου είναι ο μικροελεγκτής. Ο μικροελεγκτής θα μπορούσαμε να πούμε ότι είναι η καρδιά του συστήματος μας, αφού όλες οι λειτουργίες του τηλεχειριστηρίου βασίζονται σε αυτόν.

Ο μικροελεγκτής ο οποίος χρησιμοποιήθηκε για να υλοποιηθεί αυτή η εργασία είναι ο Atmega328P – PU. Η τροφοδοσία που χρειάζεται για να λειτουργήσει είναι 5V. Επίσης, χρησιμοποιήθηκε κρύσταλλος 16MHZ, καθώς είναι το μέγιστο που μπορεί να δεχτεί. Όπως αναφέρονται στα datasheets, για κρύσταλλο 16MHZ πρέπει να βάλουμε δύο πυκνωτές 22pF. Τα χαρακτηριστικά που διαθέτει αυτός ο μικροελεγκτής είναι:

- 6 PWM,
- 14 ψηφιακά ποδαράκια,
- 6 αναλογικές εισόδους,
- Μνήμη 32kB,
- Ταχύτητα ρολογιού 16MHZ,
- Ρεύμα εξόδου 40mA.

Τέλος, ο προγραμματισμός του γίνεται με μία συσκευή που λέγεται USB TO TTL. Με αυτή την συσκευή μπορούμε να περάσουμε το πρόγραμμα συνδέοντας σειριακά με το Amtega328P – PU. Το συγκεκριμένο USB TO TTL που έχουμε δεν διαθέτει το ποδαράκι DTR. Το DTR ανεβάζει αυτόματα το πρόγραμμα στον μικροελεγκτή. Αφού δεν διαθέτουμε το ποδαράκι αυτό, θα χρειαστεί στο ποδαράκι ένα του Atmega328 να συνδέσουμε μία αντίσταση την R5 10KΩ, την οποία θα την συνδέσουμε στην τροφοδοσία 5V. Έπειτα θα βάλουμε ένα button που θα πάει στην γείωση. Όταν θα πατάμε να ανεβάσουμε το πρόγραμμα θα πρέπει να πατήσουμε το button να κάνει restart το μικροελεγκτή για να καταφέρει να γραφτεί στον μικροελεγκτή. Η γλώσσα προγραμματισμού η οποία χρησιμοποιήθηκε είναι C Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++). Το περιβάλλον το οποίο χρησιμοποιήθηκε για να προγραμματιστεί είναι το Arduino.



Σχήμα 3.6: Συνδεσμολογία USB TO TTL με το Atmega328P – PU.

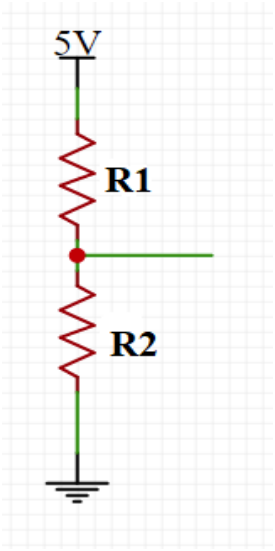
3.3. Ποτενσιόμετρα

Το τηλεχειριστήριο διαθέτει ένα joystick. Το joystick εσωτερικά αποτελείται από δύο ποτενσιόμετρα. Το ένα ποτενσιόμετρο αλλάζει η τιμή του όταν κάνουμε πάνω κάτω το joystick, αυτός είναι ο Y άξονας. Το δεύτερο ποτενσιόμετρο αλλάζει την τιμή του όταν το κάνουμε δεξιά αριστερά, αυτό είναι το X άξονας. Μία τελευταία δυνατότητα που διαθέτει το joystick είναι ένας εσωτερικό

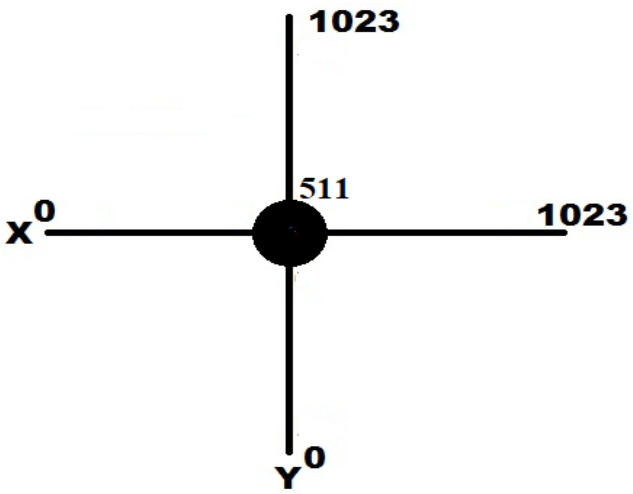
διακόπτης. Τα άλλα δύο ποδαράκια που έχει πάνε τροφοδοσία, δηλαδή στα 5V και το άλλο στην γείωση.

Το ποτενσιόμετρο είναι ένα αναλογικό εξάρτημα με τρία ποδαράκια. Το ένα άκρο πάει τροφοδοσία, το άλλο άκρο γείωση και η μέση στο μέρος που θέλουμε να τροφοδοτήσουμε. Το ποτενσιόμετρο ουσιαστικά είναι ένας διαιρέτης τάσης που όσο το κουνάμε μεταβάλετε η τάση από 0 V έως VCC, στην συγκεκριμένη περίπτωση από 0 έως 5V. Ο τύπος ο οποίος μας δίνει την τάση που θα βγάλει το μεσαίο ποδαράκι στο ποτενσιόμετρο είναι:

$$V = \frac{R2}{R1+R2} \tag{3.1}$$



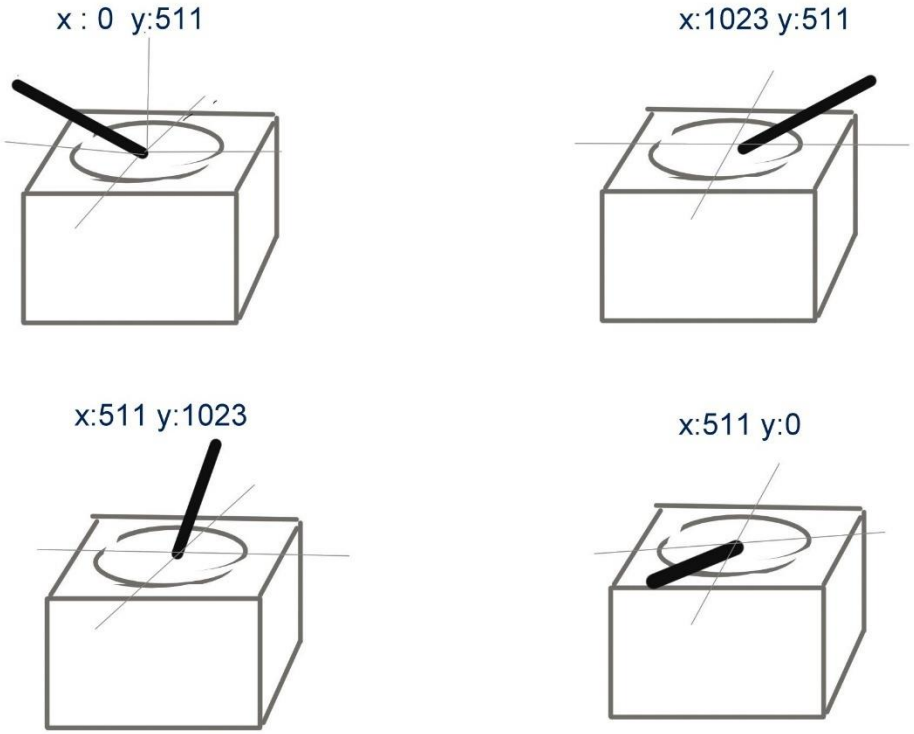
Σχήμα 3.7: Διαιρέτης τάσης.



Σχήμα 3.8: Joystick αναλογικές τιμές.

Το ποδαράκι Y του joystick συνδέεται με το analog to digital converter(ADC3) του μικροελεγκτή. Ο ADC3 θα διαβάσει την αναλογική τάση από το ποτενσιόμετρο και θα το μετατρέψει σε 10 bit αριθμό. Αν υποθέσουμε ότι κάνουμε πάνω το joystick τότε ο ADC3 θα διαβάσει την αναλογική τιμή 5V και θα τον αντιστοιχίσει με τον ακέραιο αριθμό 1023. Όταν θα κατεβάσουμε το joystick τότε η αναλογική τιμή που θα διαβάσει είναι 0, άρα ο ADC3 θα το αντιστοιχήσει με την τιμή 0. Στο κύκλωμα μας όταν θα ανεβάζουμε πάνω το joystick θα αλλάζει την φωτεινότητα του led στο σκάφος. Αρχικά θα ξεκινάει το led με την τιμή 0. Η τιμή 0 ορίζει ότι το led είναι σβησμένο. Κάθε φορά που θα ανεβάζουμε πάνω το led θα ανεβάζει ένα επίπεδο φωτεινότητας μέχρι να φτάσει το μέγιστο που είναι το 3, όταν θα φτάσει στο 3 θα παραμείνει εκεί όσο και να το ανεβάζουμε. Όταν θέλουμε να κατεβάσουμε την φωτεινότητα τότε θα πρέπει να κατεβάσουμε κάτω το joystick. Κάθε φορά που θα το κατεβάζουμε θα αφαιρεί ένα επίπεδο μέχρι να πέσει στο 0. Όταν θα πέσει στο 0 θα παραμείνει εκεί.

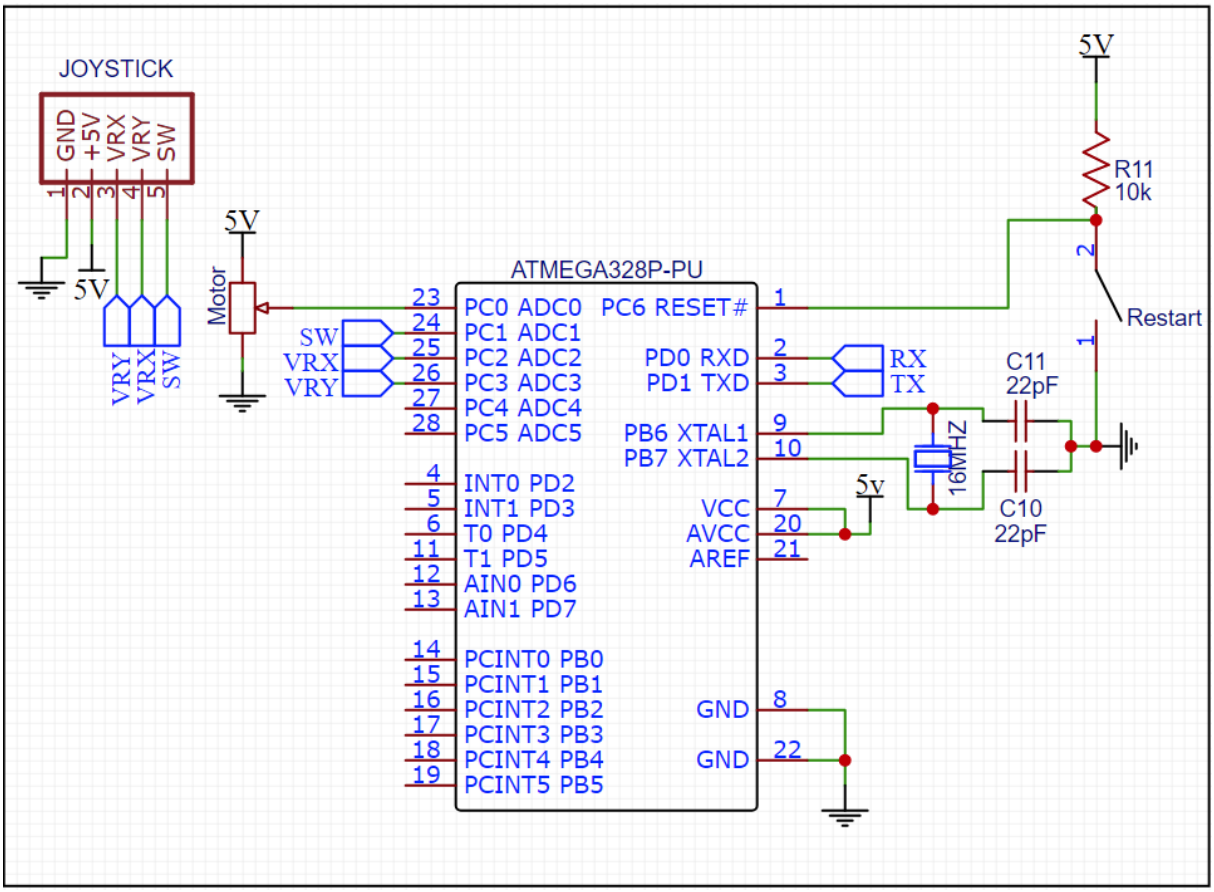
Το ποδαράκι X του joystick συνδέεται με το ADC2 του μικροελεγκτή. Ο X άξονας του joystick χρησιμοποιείται για να πει στο σκαφάκι προς ποια κατεύθυνση θα πρέπει να στρίψει. Όταν θα στρίψει το joystick αριστερά θα στρίψει και το σκαφάκι αριστερά, όταν στρίψουμε δεξιά το Joystick θα στρίψει δεξιά το σκαφάκι.



Σχήμα 3.9: Τιμές joystick σε διάφορες πλευρές [37].

Η τελευταία λειτουργία του joystick είναι ο διακόπτης. Το switch του joystick συνδέεται στο ποδαράκι ADC1. Επειδή δεν θα χρειαστεί να διαβάσουμε αναλογική τάση θα το λειτουργήσουμε σαν ψηφιακό ποδαράκι, δηλαδή θα διαβάζει μόνο High η LOW δηλαδή 0 ή 1. Όταν θα πατήσουμε το joystick μέσα ο μικροελεγκτής μας θα διαβάσει την τιμή 1. Αν το ξαναπατήσουμε θα διαβάσει την τιμή 0. Το πρακτικό κομμάτι αυτής της λειτουργίας στο σκαφάκι θα είναι με το που πατήσουμε το joystick, το σκάφος θα γυρίσει περίπου 40 μέτρα από την αρχική αρχική του θέση.

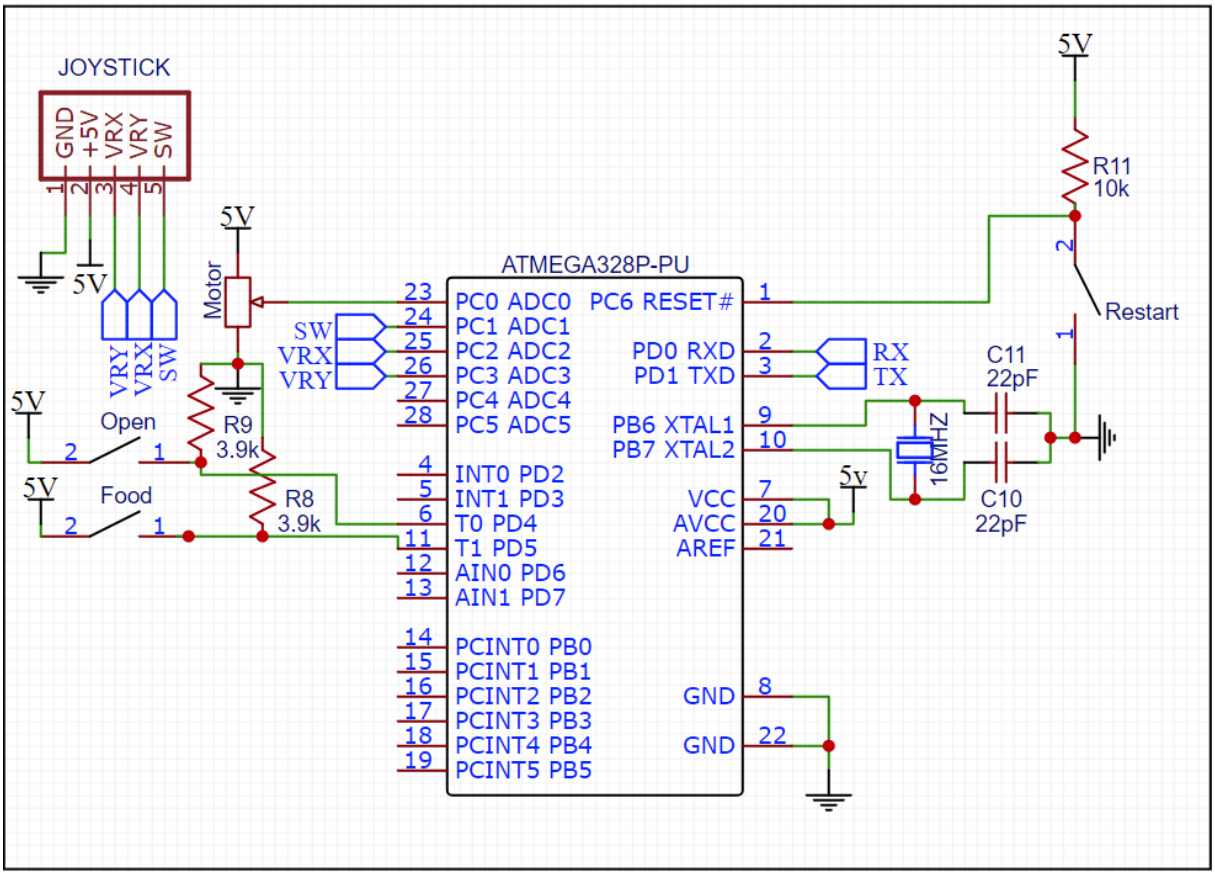
Τέλος, το τηλεχειριστήριο διαθέτει ένα ακόμα ποτενσιόμετρο. Το ποτενσιόμετρο αυτό χρησιμοποιείται για να μπορέσουμε να ρυθμίσουμε τον κινητήρα με τι ταχύτητα θα πηγαίνει. Το ποτενσιόμετρο συνδέεται στο ποδαράκι ADC0 του μικροελεγκτή. Η λειτουργία του θα είναι ίδια με το joystick, δηλαδή με το που δώσουμε 5V με το ποτενσιόμετρο θα το μετατρέψει ο ADC0 στην τιμή 1023, όταν θα δώσουμε 2,5V θα διαβάσει την τιμή 511. Στο πρόγραμμα μέσα έχουμε κάνει μια αντιστοιχία μεταξύ του 0-1023 και του 0-180. Αυτό το κάναμε για να μην στείλουμε τον αριθμό 1023 που είναι 10bit και στέλνουμε απευθείας τις τιμές που διαβάζει ο κινητήρας, δηλαδή 180 που είναι 8 bit. Έτσι εξοικονομούμε χώρο στο πακέτο που στέλνουμε.



Σχήμα 3.10: Συνδεσμολογία joystick και ποτενσιόμετρο στον μικροελεγκτή.

3.4 Button

Το button είναι ένα αναλογικό εξάρτημα που όταν το πατάμε βραχυκυκλώνει τις δύο άκρες του, όταν σταματάμε να το πατάμε σταματάει να βραχυκυκλώνει. Στο τηλεχειριστήριο χρησιμοποιούμε τρία button. Το πρώτο όπως αναφέραμε πιο πάνω είναι το reset, με το που το πατάμε θα κάνει restart ο μικροελεγκτής. Το δεύτερο button συνδέεται στο ποδαράκι 6 του μικροελεγκτή δηλαδή το digital 4. Το ένα άκρο του button θα είναι στην τροφοδοσία και το άλλο άκρο του θα είναι σε μία αντίσταση pull-down 3.9ΚΩ και στο μικροελεγκτή. Όταν θα πατήσουμε το button θα περάσει η τάση και θα δώσει 5V στον μικροελεγκτή, άρα το digital ποδαράκι που διαβάζει 0 και 1 ή LOW και HIGH , θα διαβάσει HIGH. Αυτό θα ισχύει για όσο πατάμε το button, με το που το αφήσουμε θα διαβάσει LOW. Ο λόγος που βάλαμε μία pull-down αντίσταση είναι διότι όταν δεν πατάμε το button το πόδι του θα παραμένει στον αέρα, άρα μπορεί να εισάγει θόρυβο, ακόμα και μερικά μV αν πάνε στο ψηφιακό ποδαράκι του μικροελεγκτή μπορεί να το θεωρήσει HIGH. Έτσι αυτή η αντίσταση όταν εισάγει θόρυβο το ποδαράκι του button θα το γειώνει. Το button αυτό θα ρυθμίζει πότε ο απαγκιστρωτής στο σκαφάκι θα αφήσει την πετονιά στην θάλασσα. Το τρίτο button συνδέεται στο ποδαράκι 11 του μικροελεγκτή δηλαδή στο digital 5. Αυτό το button Θα ελέγχει πότε θα ρίξει την μαλάγρα το σκαφάκι. Η λειτουργία του είναι ακριβώς ίδια με το προηγούμενο button.



Σχήμα 3.11: Σύνδεση button με τον μικροελεγκτή.

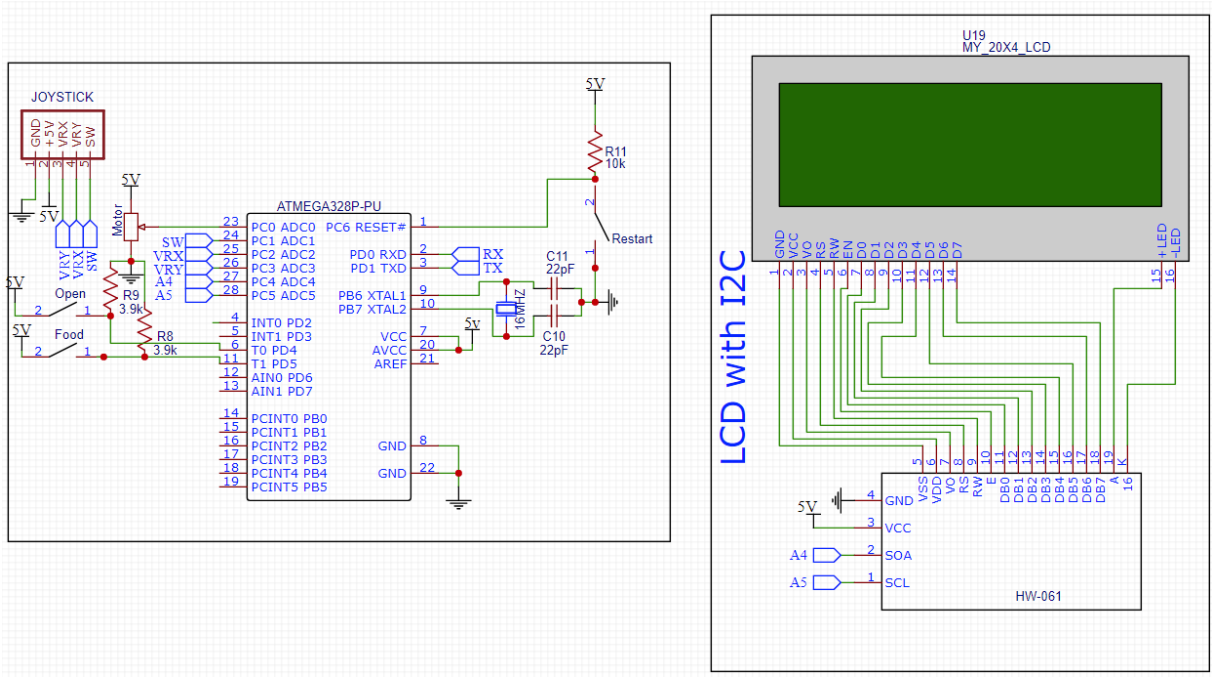
3.5 LCD

Στο τηλεχειριστήριο χρησιμοποιείται μία LCD 20x4. Όταν λέμε ότι μία LCD έχει 20x4 εννοούμε 20 θέσεις 4 σειρών. Η LCD θα μας δέσμευε πολλά ποδαράκια στον μικροελεγκτή, οπότε χρησιμοποιήσαμε έναν I2C.

Σχήμα 3.12: I2C [39].

Η συνδεσμολογία γίνεται πολύ πιο απλή. Η συσκευή αυτή χρησιμοποιεί το πρωτόκολλο I2C. Οι συσκευές i2c χρησιμοποιούν μόνο δύο ακροδέκτες για τη μεταφορά δεδομένων, το ένα είναι το SCL(Serial Clock) που συγχρονίζει τη μεταφορά δεδομένων και το δεύτερο είναι το SDA (Serial Data) για μεταφορά δεδομένων. Το I2C έχει ένα ποτενσιόμετρο πίσω του, είναι για να ρυθμίσουμε την φωτεινότητα της οθόνης. Τέλος, έχει δύο γυμνά ποδαράκια που αν τα βραχυκυκλώσουμε θα ανάψει το blacklight της οθόνης.

Η συνδεσμολογία του I2C με τον μικροελεγκτή είναι απλή, αρχικά τα δύο ποδαράκια του I2C είναι τροφοδοσία 5V και γείωση. Το ποδαράκι SDA θα συνδεθεί με το ADC4 του μικροελεγκτή και το ποδαράκι SCL θα συνδεθεί με το ποδαράκι ADC5. Η σύνδεση I2C LCD γίνεται άμεσα, βάζουμε τα pins του I2C στις θέσεις της LCD. Έχουν ίδια σειρά και ίδιες ονομασίες οπότε είναι απλή η σύνδεση τους.

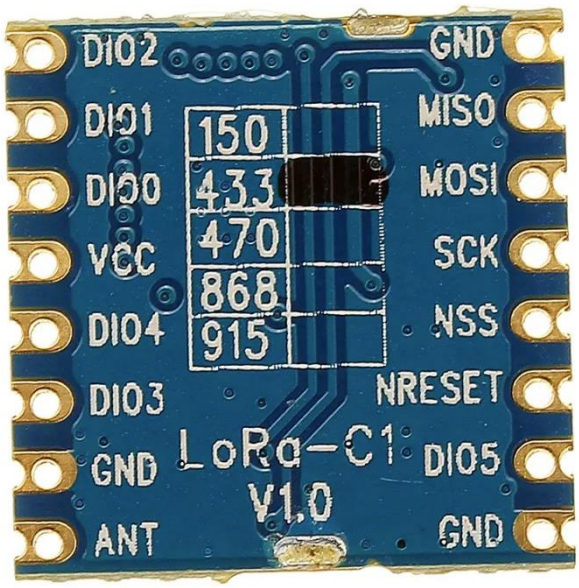


Σχήμα 3.13: Σύνδεση LCD I2C με μικροελεγκτή.

Η LCD σε αυτή την εργασία χρησιμοποιείται για να μπορούμε να βλέπουμε διάφορες λειτουργίες. Αρχικά θα μπορούμε να δούμε το επίπεδο φωτεινότητας. Όταν πατάμε τα button θα μας δείχνει μία κλειδαριά να ανοίγει, ώστε να είμαστε σίγουροι ότι δόθηκε σωστά η εντολή. Όταν στρίβουμε με το joystick θα μας δείχνει το αντίστοιχο βελάκι στην οθόνη, δηλαδή στρίβουμε δεξιά το joystick και μας δείχνει ένα βελάκι δεξιά στην οθόνη, όπως και αν στρίψουμε αριστερά θα εμφανίσει αριστερό βελάκι στην οθόνη. Όταν πατήσουμε μέσα το joystick θα βγάλει στην οθόνη το σύμβολο ένα σύμβολο που θα δείχνει ότι το σκάφος θα γυρίσει πίσω. Θα μπορούμε να δούμε το επίπεδο της μπαταρίας που έχει το σκαφάκι, όπως και τις συντεταγμένες τις οποίες έχει. Επιπλέον θα δείχνει με τι ταχύτητα πάει το σκαφάκι και σε τι απόσταση βρίσκεται όπως και πόσους δορυφόρους πιάνει.

3.6. LORA

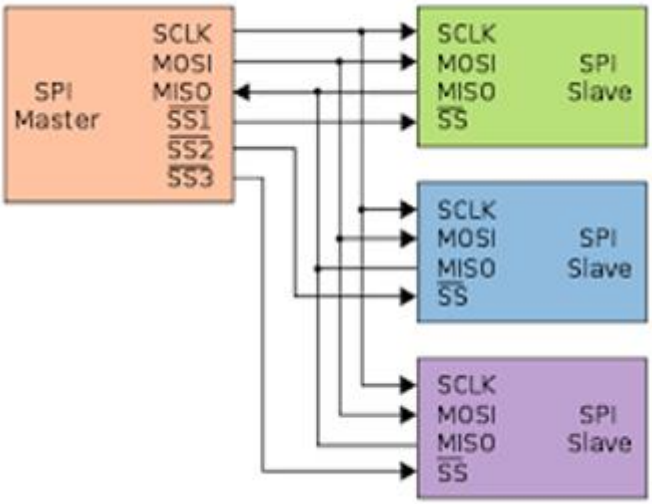
Η επικοινωνία μεταξύ του τηλεχειριστήριου και του σκάφους γίνεται με δύο Lora. Αρχικά όταν πήραμε το Lora ήταν σε μορφή SMD οπότε χρειάστηκε για τα πρώτα πειράματα να περαστεί σε πλακέτα που θα βγάζει pin ώστε να μπαίνουν σε ράστερ. Χρησιμοποιήθηκαν δύο πυκνωτές έναν κεραμικό 10nF και έναν ηλεκτρολυτικό 1μF. Στην συνέχεια τα συνδέσαμε παράλληλα στα πόδια της τροφοδοσίας ώστε να πνίγουν το rf να μην πηγαίνει προς τα πίσω στα υπόλοιπα εξαρτήματα. Το Lora που χρησιμοποιούμε είναι της Semtech το μοντέλο SX1278. Λειτουργεί σε συχνότητα 433, επιλέχτηκε αυτή η συχνότητα επειδή είναι η ελεύθερη ζώνη τηλεχειρισμού της Ευρώπης. Η επικοινωνία του SX1278 με τον μικροελεγκτή γίνεται με το πρωτόκολλο επικοινωνίας SPI.



Σχήμα 3.14*:* Lora SMD.

Το SPI (Serial Peripheral Interface) πρωτόκολλο επιτρέπει στον μικροελεγκτή να επικοινωνεί σειριακά και σύγχρονα με τις περιφερειακές συσκευές, που σημαίνει ότι χρησιμοποιεί ξεχωριστές γραμμές για τα δεδομένα και το σήμα ρολογιού, με αποτέλεσμα οι SPI συσκευές να έχουν τέλειο συγχρονισμό. Οι SPI συσκευές χρησιμοποιούν τέσσερα σήματα (ακροδέκτες) τα οποία είναι τα εξής: SCK, MISO, MOSI, SS. Αυτός που στέλνει τα δεδομένα ονομάζεται master και αυτός που τα δέχεται slave.

- Το Serial clock (SCK). Το SCK σήμα ρολογιού είναι η έξοδος του master και η είσοδος του slave. Χρησιμοποιείται για να το συγχρονισμό της μεταφοράς δεδομένων μεταξύ master και slave.
- Master in, slave out (MISO). Αυτό το σήμα είναι έξοδος από τον slave και η είσοδος του master. Χρησιμοποιείται για τη μεταφορά δεδομένων σειριακά από τον slave στον master. Ο ακροδέκτης MISO τίθεται σε κατάσταση υψηλής αντίστασης, όταν η SPI συσκευή slave δεν είναι επιλεγμένη.
- Master out, slave in (MOSI). Αυτό το σήμα είναι έξοδος από τον master και είσοδος για τους slave συσκευές. Χρησιμοποιείται για τη μεταφορά δεδομένων σειριακά από τον master στο slave
- Slave select (SS). Καθώς η SPI συσκευή διαμορφώνεται ως slave, αυτός ο ακροδέκτης είναι πάντα είσοδος. Στον slave η είσοδος SS ελέγχεται μόνο από το SPI κύκλωμα και δεν μπορεί να ελεγχθεί από κώδικα. Όταν η είσοδος SS της SPI slave συσκευής οδηγείται σε κατάσταση high ο slave απενεργοποιείται. Ο SPI slave ενεργοποιείται και γίνεται μεταφορά δεδομένα μ' αυτόν, όταν ο ακροδέκτης εισόδου SS οδηγείται σε κατάσταση low. Ο ακροδέκτης Slave select (SS) στον master πρέπει να διαμορφώνεται ως έξοδος, αφού επιλέγει τον slave για SPI επικοινωνία και αν έχουμε πολλαπλούς slave θα πρέπει να υπάρχουν πολλές γραμμές SS, οι οποίες ελέγχονται από τον κώδικα που θα γράψουμε [38].



Σχήμα 3.15: SPI επικοινωνία μεταξύ master και slave [38].

Πίνακας 3.1: Συνδεσμολογία μεταξύ Lora και Atmega328O – PU.

Atmega328P-PU	**SX1278**
MOSI, ποδαράκι 17	MOSI
MISO, ποδαράκι 18	MISO
VCC = 3,3V	5V
SCK, ποδαράκι 19	SCK
Ποδαράκι 15	NRESET
Ποδαράκι 16	NSS
Ποδαράκι 4	DIO0

Ο παραπάνω πίνακας δείχνει την συνδεσμολογία που έγινε μεταξύ Lora και Atmega328P – PU. Το DIO0 το χρησιμοποιούμε για interrupt. Το interrupt είναι μία μέθοδος που διακόπτει την κύρια ροή του προγράμματος και βάζει να ασχοληθεί με κάτι συγκεκριμένο που ορίσαμε εμείς. ΟΙ διακοπές μπορεί να είναι είτε εξωτερικές είτε εσωτερικές. Εμείς χρησιμοποιούμε εσωτερική διακοπή. Ουσιαστικά όταν λάβουμε κάποιο πακέτο γίνεται HIGH το DIO0 και έτσι σταματάει ότι κάνει ο μικροελεγκτής και διαβάζει το πακέτο. Επίσης για να σιγουρέψουμε ότι δεν θα κάνει τίποτα εάν λάβει ένα άκυρο πακέτο, προσθέσαμε στο πακέτο ένα κωδικό. Αν όταν λάβει το πακέτο ελέγξει και δει ίδιο κωδικό τότε θα προχωρήσει και στα άλλα, αλλιώς θα πάει στο επόμενο πακέτο.

Το Lora διαθέτει μερικές δυνατότητες που μπορούμε να τα προσαρμόσουμε. Μία από τις δυνατότητες αυτές είναι το bandwidth. Το bandwidth (BW) είναι η χωρητικότητα που έχει ένα δίκτυο για την μεταφορά των δεδομένων. Τα bandwidth που έχουμε σαν επιλογή είναι:

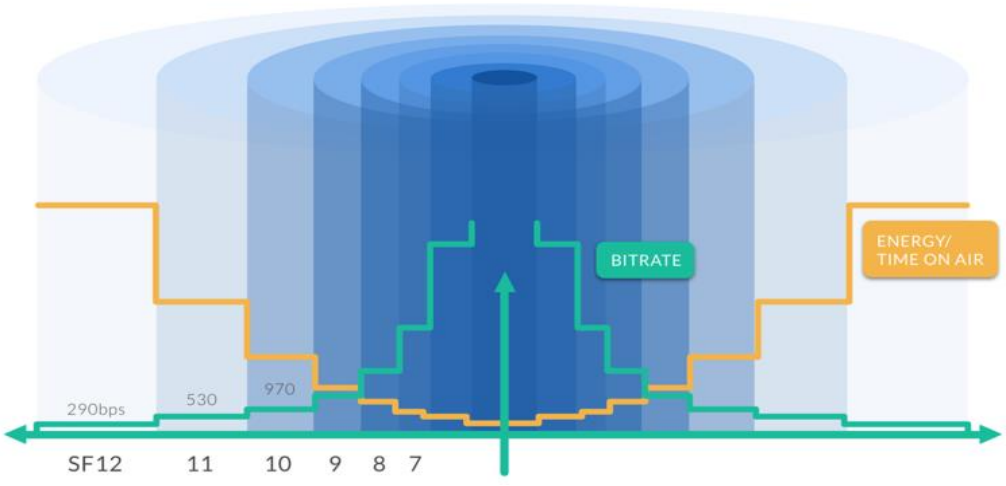
- 7.80 kHz
- 10.40 kHz
- 15.60 kHz
- 20.80 kHz
- 31.25 kHz
- 41.70 kHz
- 62.50 kHz
- 125.00 kHz
- 250.00 kHz
- 500.00 kHz

Όσο υψηλότερο BW δίνει υψηλότερο ρυθμό δεδομένων, αλλά χάνει σε απόσταση. Ένα χαμηλότερο BW δίνει μεγαλύτερη απόσταση επικοινωνίας, αλλά χαμηλότερο ποσοστό δεδομένων.

Το Spreading Factor (SF) ορίζεται από τον τύπο :

$$SF = log_2(\frac{Rc}{Rs}) \tag{3.2}$$

Όπου το Rs (symbol rate) είναι ο ρυθμός συμβόλων (symbols/s) και το Rc (chip rate) είναι ο αριθμός των παλμών ανά δευτερόλεπτο στους οποίους μεταδίδεται ή λαμβάνεται ο κωδικός. Ο ρυθμός του chip rate είναι μεγαλύτερος από τον ρυθμό συμβόλων. Οι επιλογές που έχουμε για το SF είναι από το 6 έως το 12. Το SF όσο μεγαλύτερος αριθμός είναι τότε αυξάνεται η απόσταση επικοινωνίας αλλά θα έχει λιγότερα δεδομένα για κωδικοποίηση ανά δευτερόλεπτο. Για τον ίδιο όγκο δεδομένου με υψηλό SF θα κάνει περισσότερη ώρα μετάδοσης. Περισσότερο χρόνος μετάδοσης σημαίνει περισσότερη κατανάλωση ενέργειας [40 – 42].



Σχήμα 3.16: Συμπεριφορά SF για κάθε αριθμό [40].

Το Coding rate (CR) περιγράφει την αναλογία των πραγματικών δεδομένων προς τα δεδομένα που προστίθενται για τη διόρθωση σφαλμάτων. Οι τιμές που μπορεί να πάρει είναι:

- $\frac{4}{5}$,
- $\frac{4}{6}$,
- $\frac{4}{7}$,
- $\frac{4}{8}$.

Όσο μικρότερο είναι το CR τόσο υψηλότερος είναι ο χρόνος μετάδοσης, οπότε θα κάνει περισσότερη ώρα ώστε να στείλει ένα πακέτο και θα αυξήσει την κατανάλωσή της μπαταρίας.

Με όλα τα παραπάνω στοιχεία μπορούμε να βρούμε το Data Rate δηλαδή τον ρυθμό δεδομένων, Ο τύπος ο οποίος μας δίνει την ταχύτητα των δεδομένων είναι:

$$Data\ rate = SF\ \times \frac{BW}{2^{SF}} \times CR \tag{3.3}$$

Το αποτέλεσμα του data rate είναι η ταχύτητα με την οποία τα δεδομένα μεταφέρονται από τη μια συσκευή στην άλλη. Το data rate το μετράμε σε bps.

Άλλη ρύθμιση την οποία μπορεί να κάνει κανείς είναι η λειτουργία sync. Αυτή η λειτουργία είναι μία τιμή ενός byte που χρησιμοποιείται για τη διαφοροποίηση των δικτύων Lora που χρησιμοποιούν τις ίδιες ζώνες συχνοτήτων. Υποδεικνύει το μέγεθος του ωφέλιμου φορτίου (σε byte), το code rate που χρησιμοποιείται και αν υπάρχει η όχι ένα CRC. Μπορεί να δεχτεί από 0 έως 0XFF.

Τέλος, υπάρχουν άλλες τέσσερις λειτουργίες. Η ισχύς εξόδου που μπορεί να πάρει τιμές από 2 dBm έως 17dBm δηλαδή από 1.5mW έως 50mW. Επίσης έχει λειτουργία που βάζουμε όριο στο ρεύμα, οι τιμές που μπορεί να πάρει είναι από 45 – 240 mA. Αν βάλουμε την τιμή 0 τότε απενεργοποιείται αυτή η λειτουργία. Τα τελευταία δύο είναι το preamble length που οι τιμές που δέχεται είναι 6 – 65535 και το κέρδος του ενισχυτή είναι 1-6 όπου το 1 είναι το μέγιστο κέρδος. Το 0 είναι αυτόματο έλεγχο κέρδους [41][42].

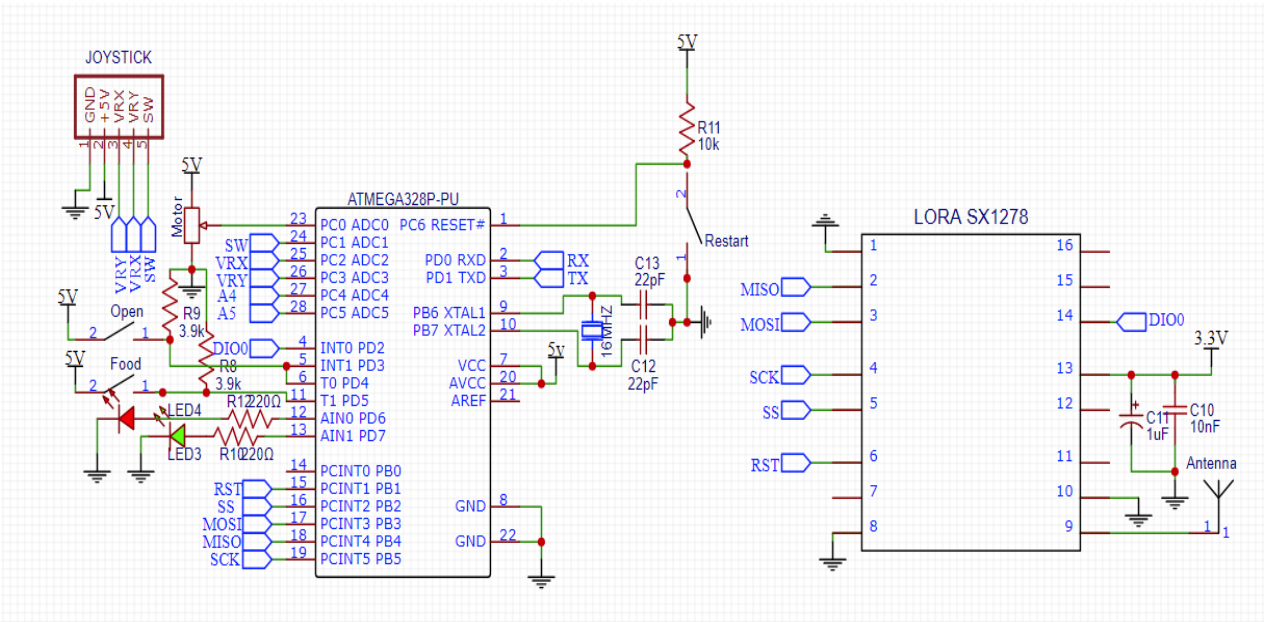
Στο τηλεχειριστήριο οι τιμές που ορίσαμε είναι BW: 500kHZ, SF: 8 , CR: $\frac{4}{7}$, sync: 0x18, ισχύ εξόδου 14dBm, ασφάλεια στα 100mA, preamble lengt: 20, κέρδος ενισχυτή: 0.

$$Data\ rate = 8 \times \frac{500kHz}{2^8} \times \frac{4}{7} = 8.929 \text{ kbps}$$

Δόθηκε περισσότερο βάση στην ταχύτητα τα δεδομένων και όχι στην απόσταση. Η απόσταση θέλουμε να είναι το πολύ στα 500μέτρα. Τέλος, χρησιμοποιήθηκε μια κεραία σε συχνότητα λειτουργίας 433MHz, με κέρδος 2dBi.

Σχήμα 3.17: Κεραία τηλεχειριστηρίου.

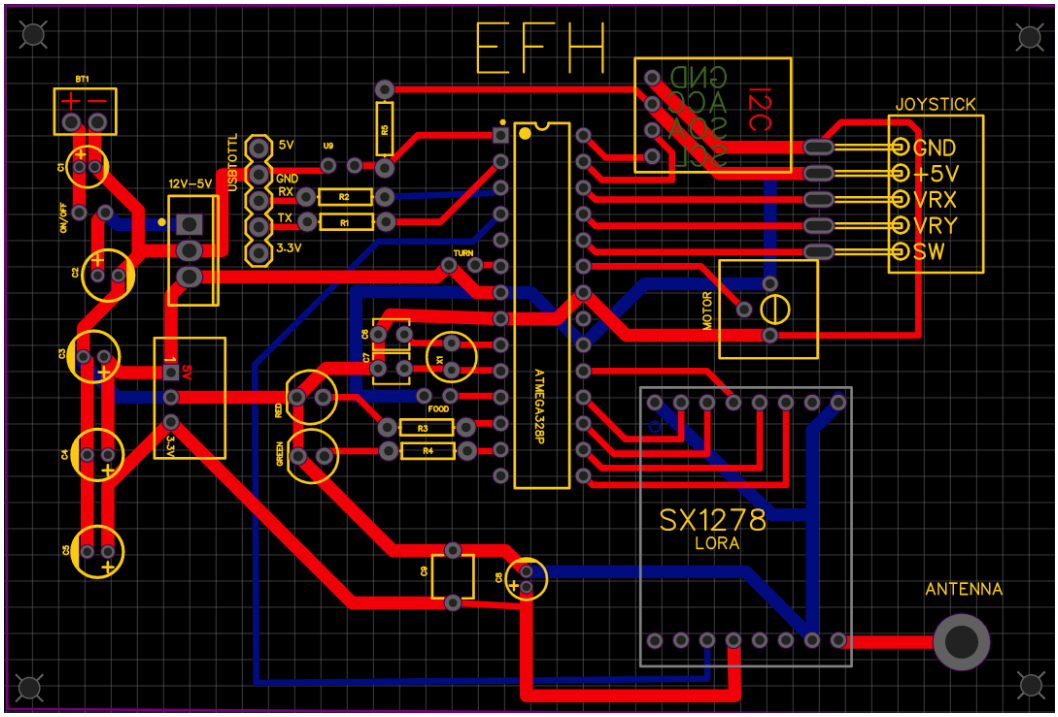
Τα τελευταία εξαρτήματα που τοποθετήθηκαν στο τηλεχειριστήριο είναι τα δύο led, ένα πράσινο και ένα κόκκινο. Το πράσινο led όταν ανάβει σημαίνει ότι το Lora λειτουργεί σωστά, δηλαδή η συχνότητα του είναι όσο πρέπει, το BW κ.λπ. Όταν ανάβει το κόκκινο και ταυτόχρονα το πράσινο είναι επειδή υπήρξε κάποιο πρόβλημα με το CRC στο πακέτο. Όταν κλείσει το πράσινο φως και μείνει μόνο το κόκκινο ανοιχτό σημαίνει ότι υπάρχει πρόβλημα το οποίο μόνο με restart ίσως διορθωθεί. Αν πάλι δεν γίνει σημαίνει κάτι δεν λειτουργεί σωστά και υπάρχει ηλεκτρονικό πρόβλημα με την πλακέτα η το Lora.



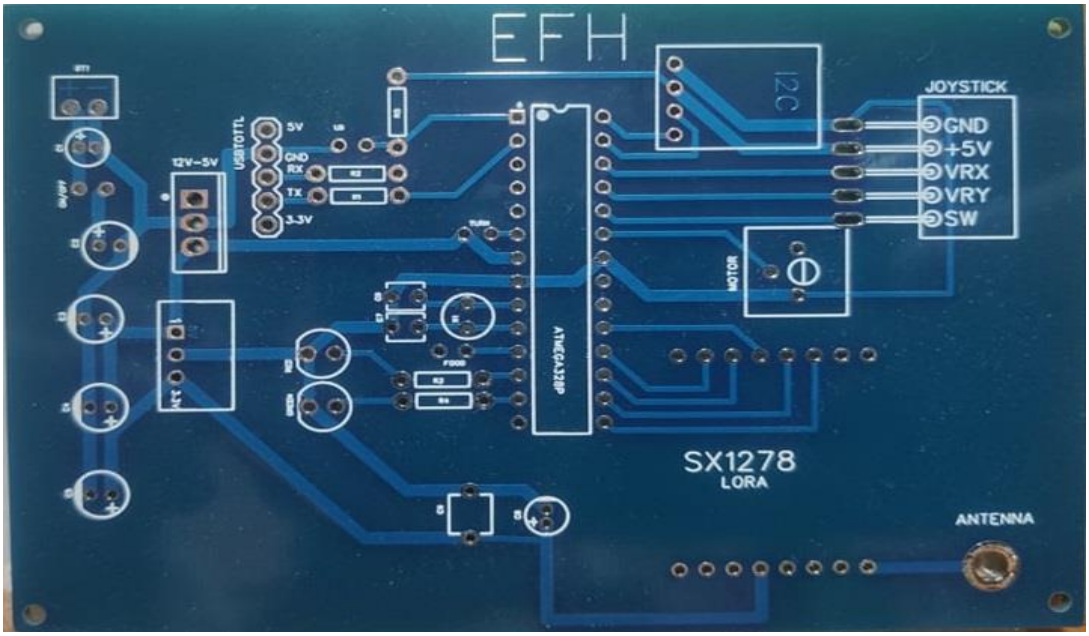
Σχήμα 3.18: Σύνδεση Lora και leds με μικροελεγκτή.

3.6 Κατασκευή

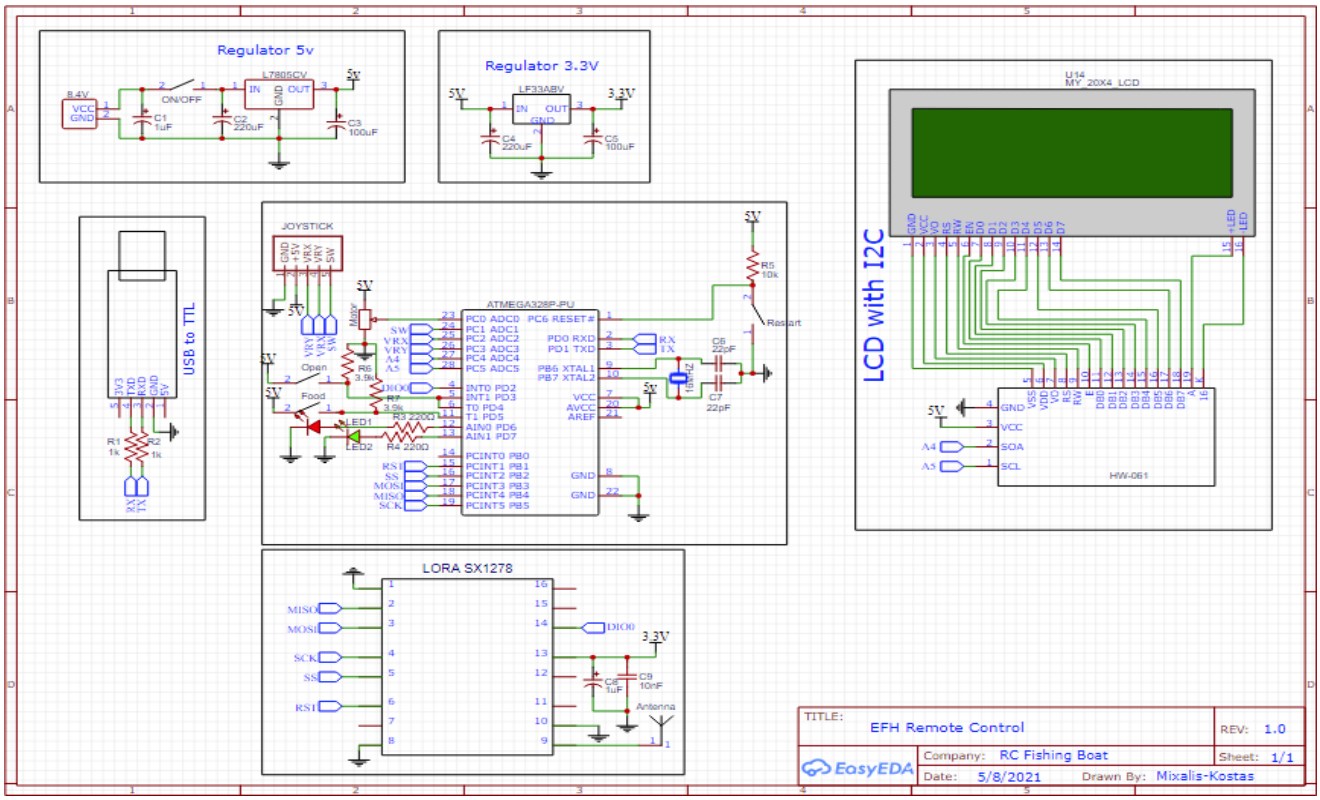
Το πρόγραμμα που χρησιμοποιήθηκε για την σχεδίαση και την απεικόνιση του κυκλώματος είναι το EasyEDA. Το πάχος των αγωγών οι οποίοι τροφοδοτούσαν είναι 50mil, ενώ τα υπόλοιπα 25mil.



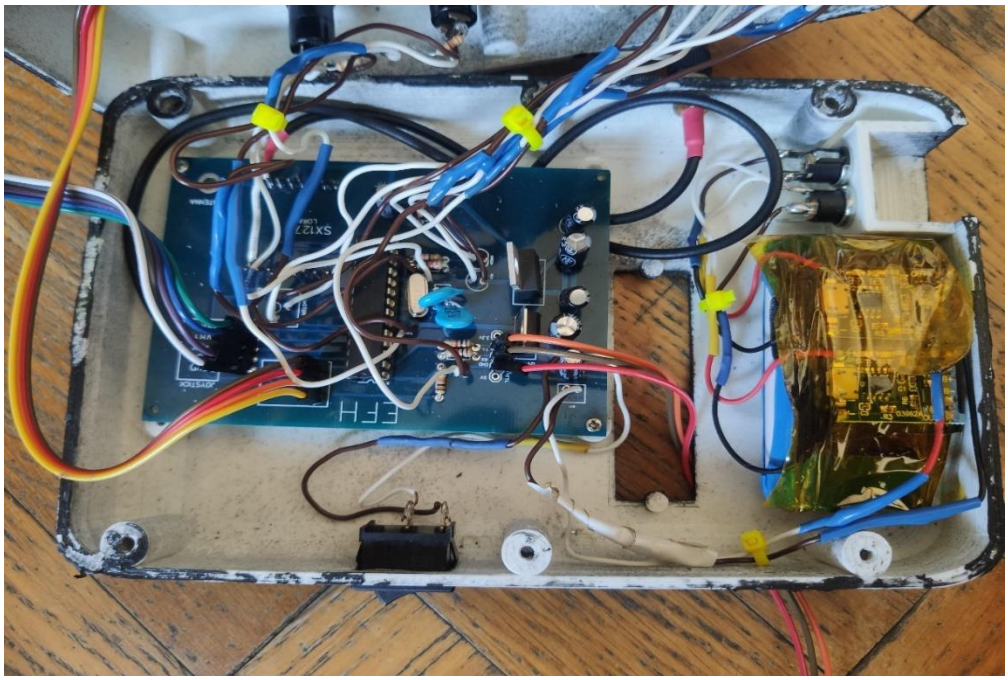
Σχήμα 3.19: Σχέδιο πλακέτας.



Σχήμα 3.20: Πλακέτα.



Σχήμα 3.21: Τελικό κύκλωμα.



Σχήμα 3.22: Τελικό κύκλωμα.

Το εξωτερικό μέρος του τηλεχειριστηρίου φτιάχτηκε με 3D printer.

Σχήμα 3.23: Αρχικό αποτέλεσμα.

Σχήμα 3.24: Τελικό αποτέλεσμα.

Κεφάλαιο 4ο: Λειτουργία Σκάφους

Σε αυτό το κεφάλαιο θα αναλύσουμε την λειτουργία του σκάφους. Υλικά τα οποία χρησιμοποιήθηκαν και στο τηλεχειριστήριο, όπως για παράδειγμα το Lora, δεν θα τα αναλύσουμε για να μην επαναλαμβανόμαστε.

4.1 Τροφοδοσία σκάφους

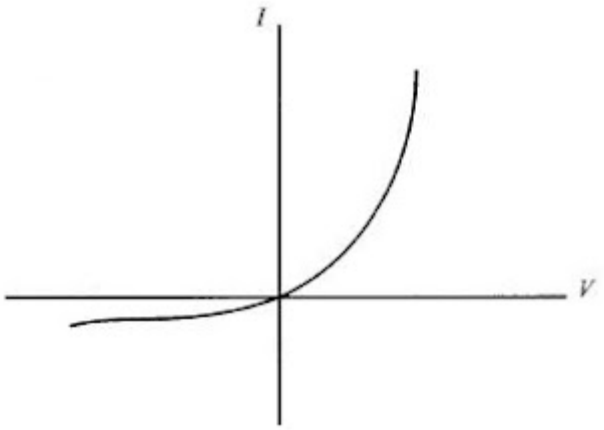
Η τροφοδοσία του σκάφους βασίζεται σε μπαταρίες Li – po. Η τάση από τις μπαταρίες είναι 11.1V δηλαδή είναι τρεις μπαταρίες σε σειρά (3S). Επειδή το ρεύμα που μας παρέχει μία μπαταρία 3S είναι λίγο (4500mAh), χρησιμοποιήσαμε δύο μπαταρίες 3S και τις συνδέσαμε παράλληλα. Έτσι αυξήσαμε το ρεύμα τους και το κάναμε 9000mAh. Η κάθε μία μπαταρία έχει την δυνατότητα να χειριστεί φορτίο 202.5A.

$$45 \times 4.5 = 202.5\ A$$

Σχήμα 4.1: Δύο μπαταρίες Li – po 3S συνδεδεμένες παράλληλα.

Οι μπαταρίες Li – po συνδέονται στον electronic speed controller (ESC). Ο ESC όπως έχουμε αναφέρει βγάζει τρία καλώδια. Ένα του PWM και δύο για να τροφοδοτεί. Αυτό το σύστημα τροφοδοσίας ονομάζεται BEC. Εμείς χρησιμοποιήσαμε το BEC για να τροφοδοτήσουμε την πλακέτα. Το BEC μας βγάζει 5.5V. Έτσι, προσθέσαμε μία δίοδο Schottky για να ρίξουμε την τάση και ταυτόχρονα, εκτός από το ESC, θα μάς δίνει ασφάλεια και η δίοδος.

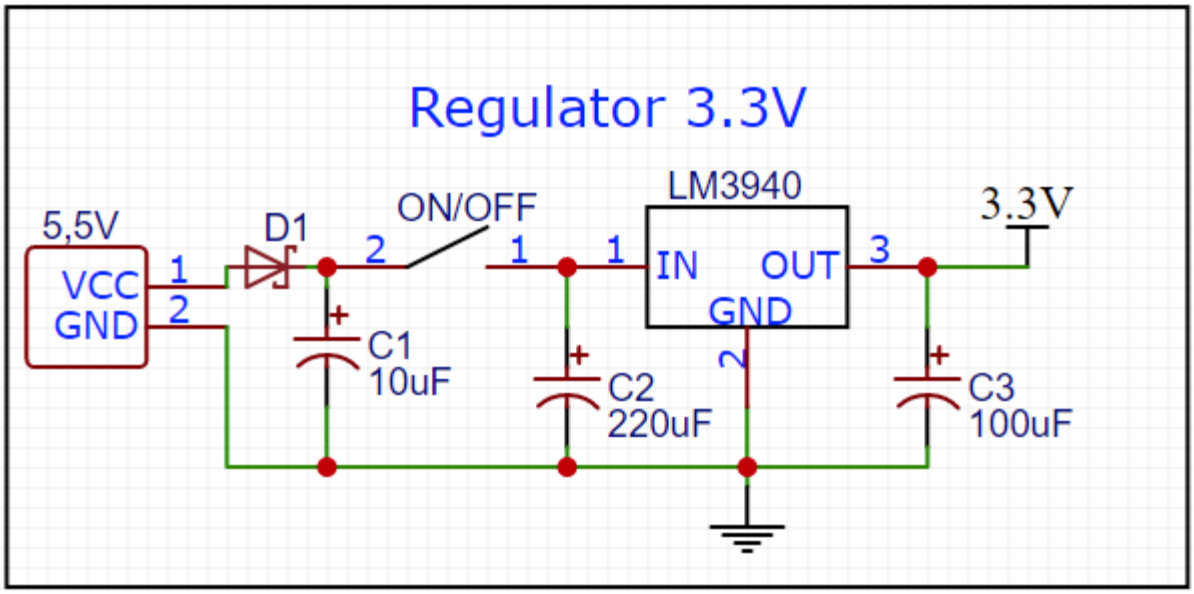
Η δίοδος Schottky είναι μια πιο εξειδικευμένη δίοδος που αποτελείται από ένα συνδυασμό μετάλλου και ημιαγωγού. Στο ένα μέρος της επαφής υπάρχει μέταλλο, χρυσός, λευκόχρυσος ή άργυρος. Στην άλλη επαφή θα παρατηρήσουμε σώμα πυριτίου. Πριν πολωθεί η δίοδος αυτή, τα ελεύθερα ηλεκτρόνια του πυριτίου βρίσκονται σε χαμηλότερα επίπεδα ενέργειας από αυτά των ηλεκτρονίων του μετάλλου. Αυτή η διαφορά στάθμης ονομάζεται φράγμα Schottky. Μόλις έρθει η δίοδος σε ορθή πόλωση, τα ελεύθερα ηλεκτρόνια του πυριτίου ανεβαίνουν ενεργειακά, φτάνοντας σε επίπεδο που να μπορούν να εισέλθουν στο μέταλλο και να επιτευχθεί μεγάλο ρεύμα. Η επαναφορά σε κατάσταση αποκοπής γίνεται σχεδόν ακαριαία εν συγκρίσει με τις υπόλοιπες διόδους [43].



Σχήμα 4.2: Χαρακτηριστική καμπύλη διόδου Schottky [9].

Σχήμα 4.3: ESC.

Στην συνέχεια, μετά την δίοδο Schottky έχουμε έναν ηλεκτρολυτικό πυκνωτή 10μF για να μάς δώσει άμεσο ρεύμα και έπειτα καταλήγει στον σταθεροποιητή τάσης τον LM3940 που θα μας ρίξει την τάση στα 3.3V. Ο σταθεροποιητής LM3940 χρησιμοποιεί δύο ηλεκτρολυτικούς πυκνωτές 22μF και 100μF όπως τους άλλους δύο σταθεροποιητές που χρησιμοποιούμε στο τηλεχειριστήριο.



Σχήμα 4.4: Κύκλωμα τροφοδοσίας.

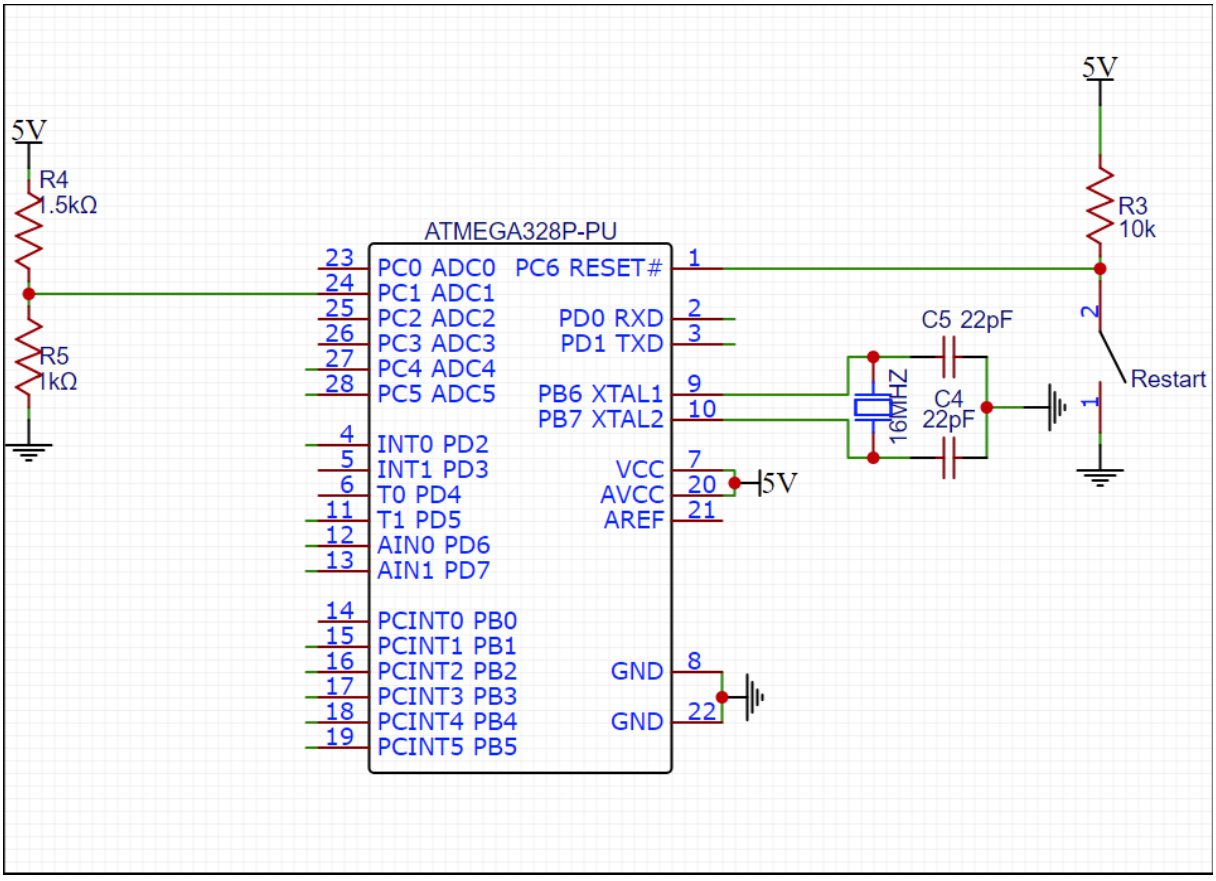
4.2 Δεδομένα αποστολής

Το σκαφάκι στέλνει έξι δεδομένα. Με βάση αυτά τα δεδομένα θα αναλύσουμε και τις αντίστοιχες λειτουργίες.

Το πρώτο μέρος είναι ο μετρητής των μπαταριών. Όταν λέμε μετρητή εννοούμε πως θα μας δείχνει πόσο τις εκατό είναι η μπαταρία ώστε να γνωρίζουμε αν θα ξεκινήσουμε το σκαφάκι. Η λειτουργία του είναι απλή, κάναμε ένα διαιρέτη τάσης ο οποίος θα αντιστοιχεί τα 12.6V με 5V. Ο λόγος που θέλουμε να πάει στα 5V είναι επειδή τόσο μπορεί να δεχτεί ο μικροελεγκτής. Επειδή είναι απίθανο να πετύχουμε ακριβός 5V, θα βάλουμε τιμές αντιστάσεων που θα μας φέρει όσο πιο κοντά γίνεται. Οι τιμές που βάλαμε είναι 1ΚΩ στην γείωση και 1.5ΚΩ στην τροφοδοσία. Οπότε αν κάνουμε την πράξη

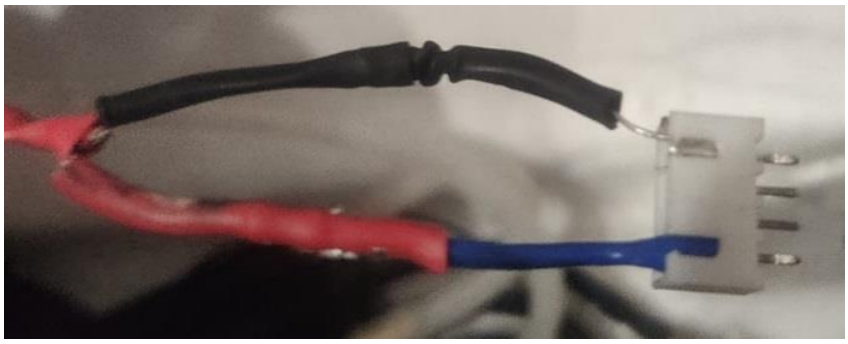
$$\frac{1k}{1k+1,5k} \times 5V = 5.04V.$$

Επειδή οι αντιστάσεις δεν έχουν ακριβώς ίδια τιμή, το αποτέλεσμα στο διαιρέτη βγαίνει γύρω στο 4.8V. Για να πιάσουμε όσο το δυνατό πιο κοντά στην πραγματική τάση της μπαταρίας προσθέσαμε στο τελικό αποτέλεσμα έναν αριθμό, τον οποίο μπορούμε να τον θεωρήσουμε αριθμό σφάλματος. Η σύνδεση του διαιρέτη τάσης έγινε με την αναλογική πόρτα ADC1.

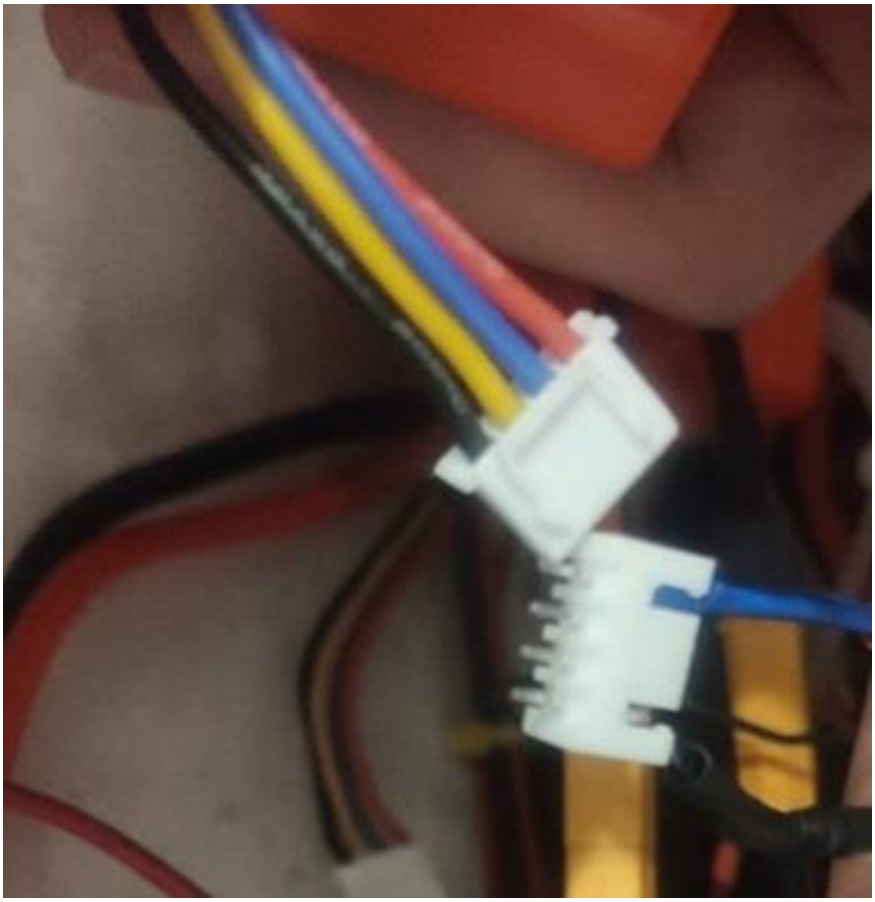


Σχήμα 4.5: Σύνδεση μετρητή μπαταρίας στο μικροελεγκτή.

Επειδή στο κύκλωμα παίρνουμε τάση από το BEC δηλαδή σταθερή τάση, δεν θα μπορούσαμε να καταλάβουμε την τάση που έχει η μπαταρία, κάναμε μία πατέντα την οποία θα τα συνδέουμε κάθε φορά με την μπαταρία αλλά όχι από την πλευρά που δίνουμε τροφοδοσία στον ESC αλλά από την πλευρά που έχουν τα καλώδια οι Li – po μπαταρίες ώστε να μπορείς να μετρήσεις την κάθε μπαταρία που έχει ξεχωριστά. Εμείς μετράμε την συνολική τάση, άρα βάζουμε στο μαύρο και στο κόκκινο.



Σχήμα 4.6: Μετρητής μπαταρίας.



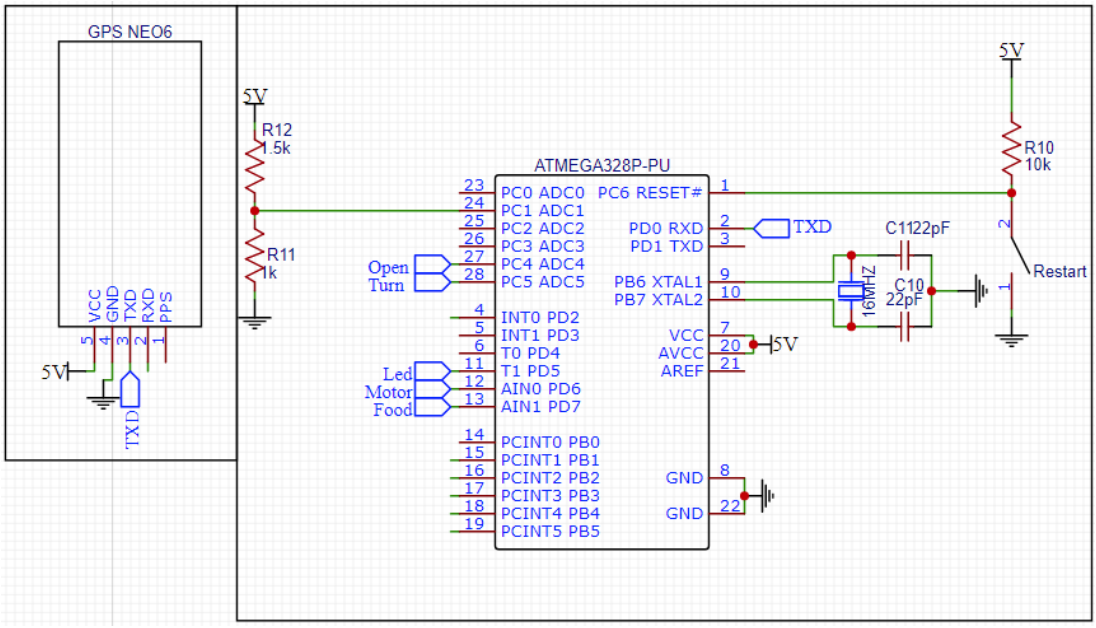
Σχήμα 4.7: Σύνδεση διαιρέτη με την μπαταρία.

Τα επόμενα δεδομένα που θα στέλνει το σκαφάκι είναι τις συντεταγμένες από το GPS. Επίσης εκτός από τις συντεταγμένες θα στέλνει την ταχύτητα από το σκαφάκι και την απόσταση που βρίσκεται. Για να μπορέσει να υπολογίσει την απόσταση και την ταχύτητα θα χρειαστεί να γνωρίζει την αρχική συντεταγμένη. Όπως αναφέραμε στο τηλεχειριστήριο όταν πατήσουμε το button μέσα το σκαφάκι θα αποθηκεύσει την συντεταγμένη στην οποία βρίσκεται εκείνη την στιγμή. Έτσι χάρης την βιβλιοθήκη tinygps++ θα μπορέσουμε να υπολογίσουμε την απόσταση και την ταχύτητα.

Στην συνδεσμολογία του gps υπήρξε πρόβλημα γιατί δεν γινόταν να κάνουμε μέσω software άλλα pins σειριακά. Αυτό συνέβη επειδή χρησιμοποιεί τον Timer1 όπως και οι σερβοκινητήρες. Οπότε αναγκαστικά το RX και το TX του μικροελεγκτή όταν το χρησιμοποιούσαμε για προγραμματισμό βγάζαμε το gps.

Το GPS που χρησιμοποιήθηκε είναι το NEO 6. Το VCC του πάει στα 5V το GND γείωση και το TX του στο RX του μικροελεγκτή.

Σχήμα 4.8: GPS.



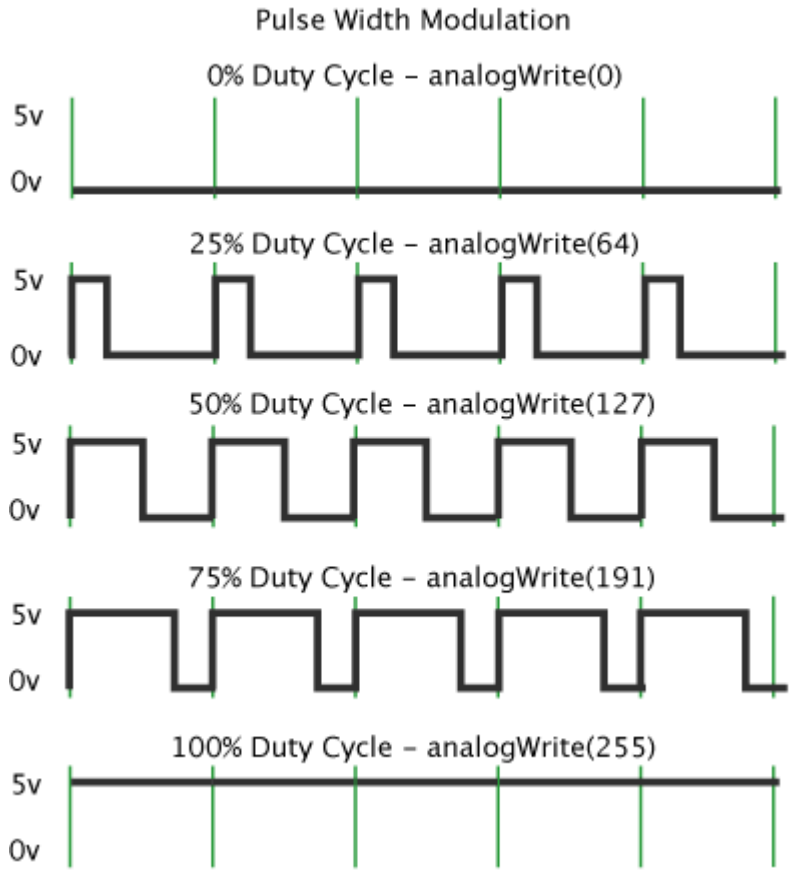
Σχήμα 4.9: Σύνδεση GPS μικροελεγκτή.

Τέλος, και το σκαφάκι στέλνει πίσω ένα κωδικό. Το τηλεχειριστήριο θα το ελέγξει και αν είναι σωστό θα γράψει τα δεδομένα που δέχτηκε, αλλιώς θα προχωρήσει κανονικά στο επόμενο πακέτο χωρίς να κάνει τίποτα.

4.3. Εισερχόμενα δεδομένα

Τα δεδομένα τα οποία δέχεται το σκαφάκι είναι αριθμοί με τους οποίους απεικονίζουμε τι θέλουμε να κάνουμε. Ένα παράδειγμα είναι αντί να στείλουμε τον σερβοκινητήρα "στρίψε 180 μοίρες" του στέλνουμε "1". Αυτό γίνεται για να μην χρειάζεται να στείλουμε μεγάλα πακέτα, όσο μικρότερο είναι το πακέτο τόσο πιο σίγουροι είμαστε ότι θα φτάσει χωρίς να αλλοιωθεί .

Στο σκαφάκι χρησιμοποιείται ένα led 3watt. Το led αυτό χρησιμοποιείται για να μπορούμε να δούμε την νύχτα που βρίσκεται το σκαφάκι. Ο έλεγχος του led γίνεται με PWM. Το PWM δίνει τιμές από 0 έως 255.



Σχήμα 4.10: Αναλογικές τιμές αντιστοιχία με Duty Cycle.

Όπως αναφέραμε στο τηλεχειριστήριο, το led θα έχει τρεις κλίμακες, δηλαδή τρία επίπεδα φωτεινότητας. Όταν θα φτάνει ο αριθμός ένα θα το μετατρέψουμε σε τιμή PWM δηλαδή 85 και θα ανάψει λίγο το φως. Όταν στείλουμε το δύο θα δώσει το PWM 170 άρα θα είναι η μεσαία φωτεινότητα και όταν στείλουμε 3 θα πάει στο τέρμα δηλαδή 255. Το led έχει τρία ποδαράκια. Το ένα η τροφοδοσία το οποίο πάει στα 5V, το άλλο είναι η γείωση και το τρίτο είναι το σήμα που συνδέεται στο ποδαράκι 11 digital 5 του μικροελεγκτή.

Σχήμα 4.11: LED 3W.

Οι σερβοκινητήρες ανάλογα με τον παλμό που θα δεχτούν θα πάρουν τις αντίστοιχες μοίρες. Ας υποθέσουμε ότι πατάμε το button που ρίχνει την μαλάγρα, αυτό θα στείλει τον αριθμό ένα στο σκαφάκι. Το σκαφάκι μέσω του κώδικα θα δει ότι ορίσαμε ως "1" τις 180 μοίρες, άρα θα δώσει 2ms παλμό και θα ρίξει την μαλάγρα. Όταν θα αφήσουμε το button θα στείλει τον αριθμό μηδέν. Όταν το λάβει το σκαφάκι θα ελέγξει ότι το ορίσαμε όταν λάβει μηδέν να πάει στις 0 μοίρες, αρά θα πάρει 1ms παλμό. Ο σερβοκινητήρας αυτός συνδέεται στο ποδαράκι 13 του μικροελεγκτή δηλαδή στο digital 7. Στα 5V έχει την δύναμη να σηκώσει 13 κιλά.

Σχήμα 4.12: Ο σερβοκινητήρας που ρίχνει την μαλάγρα.

Στο σερβοκινητήρα που ξεαγκιστρώνει λειτουργεί αντίθετα. Όταν θα λάβει τον αριθμό μηδέν τότε θα ελέγξει ότι το ορίσαμε 180 μοίρες και θα πάρει 2ms παλμό και θα κλείσει την δαγκάνα. Όταν λάβει τον αριθμό ένα θα δει ότι πρέπει να πάει 0 μοίρες και θα ανοίξει την δαγκάνα. Ο σερβοκινητήρας αυτός είναι συνδεδεμένος στο ποδαράκι 27 του μικροελεγκτή. Στα 5V έχει την δύναμη να σηκώσει 2.2κιλά.

Σχήμα 4.13: Ο σερβοκινητήρας που χρησιμοποιεί η δαγκάνα.

Ο σερβοκινητήρας που στρίβει δέχεται τρεις αριθμούς. Όταν πάμε αριστερά το joystick θα στείλει τον αριθμό μηδέν και θα στρίψει ο σερβοκινητήρας 90 μοίρες δηλαδή αριστερά. Όταν αφήσουμε το joystick στην μέση θα στείλει τον αριθμό ένα και ο σερβοκινητήρας θα μείνει στην μέση δηλαδή 57 μοίρες. Τέλος όταν στρίψουμε δεξιά το joystick θα στείλει τον αριθμό δύο και θα πάει ο σερβοκινητήρας στις 30 μοίρες. Ο σερβοκινητήρας συνδέθηκε στο ποδαράκι 28. Χρησιμοποιήσαμε ίδιο σερβοκινητήρα με αυτόν που ρίχνει την μαλάγρα.

Ο τρόπος που θα λειτουργήσουμε τον ESC είναι εξίσου ίδιος με τους σερβοκινητήρες, η μόνη διαφοροποίηση είναι ότι πρέπει να ορίσουμε εξαρχής ότι 1ms είναι ο ελάχιστος παλμός και 2ms ο μέγιστος παλμός που θα δεχτεί. Έτσι θα αντιστοιχήσει τις τιμές 0 έως 180 που θα του λένε με τι ταχύτητα πρέπει να τρέξει. Το καλώδιο που παίρνει το σήμα το βάλαμε στο ποδαράκι 12 του μικροελεγκτή. Οι δυνατότητες που έχει ο κινητήρας μας είναι:

- KV (RPM / Volt): 4200KV. Αυτό σημαίνει ότι για κάθε 1V, μπορεί να κάνει 4200 στροφές.
- Οι μέγιστες στροφές που μπορεί να πάρει είναι 60.000 RPM.
- Μέγιστο ρεύμα που μπορεί να δεχτεί : 100A.
- Μπορεί να αντέξει έως 1400Watt.
- Μέγιστη τάση 14V.

Η μέγιστη τάση που μπορεί να πάρει ο κινητήρας είναι 12.6V άρα και οι μέγιστες στροφές είναι:

12 X 4200 = 50.400 RPM

Συν 2.100 στροφές για το 0.6 V που δεν υπολογίσαμε, βγαίνουν 52.500RPM.

Και οι ελάχιστες στροφές είναι:

$$10\ X\ 4200 = 42.000RPM.$$

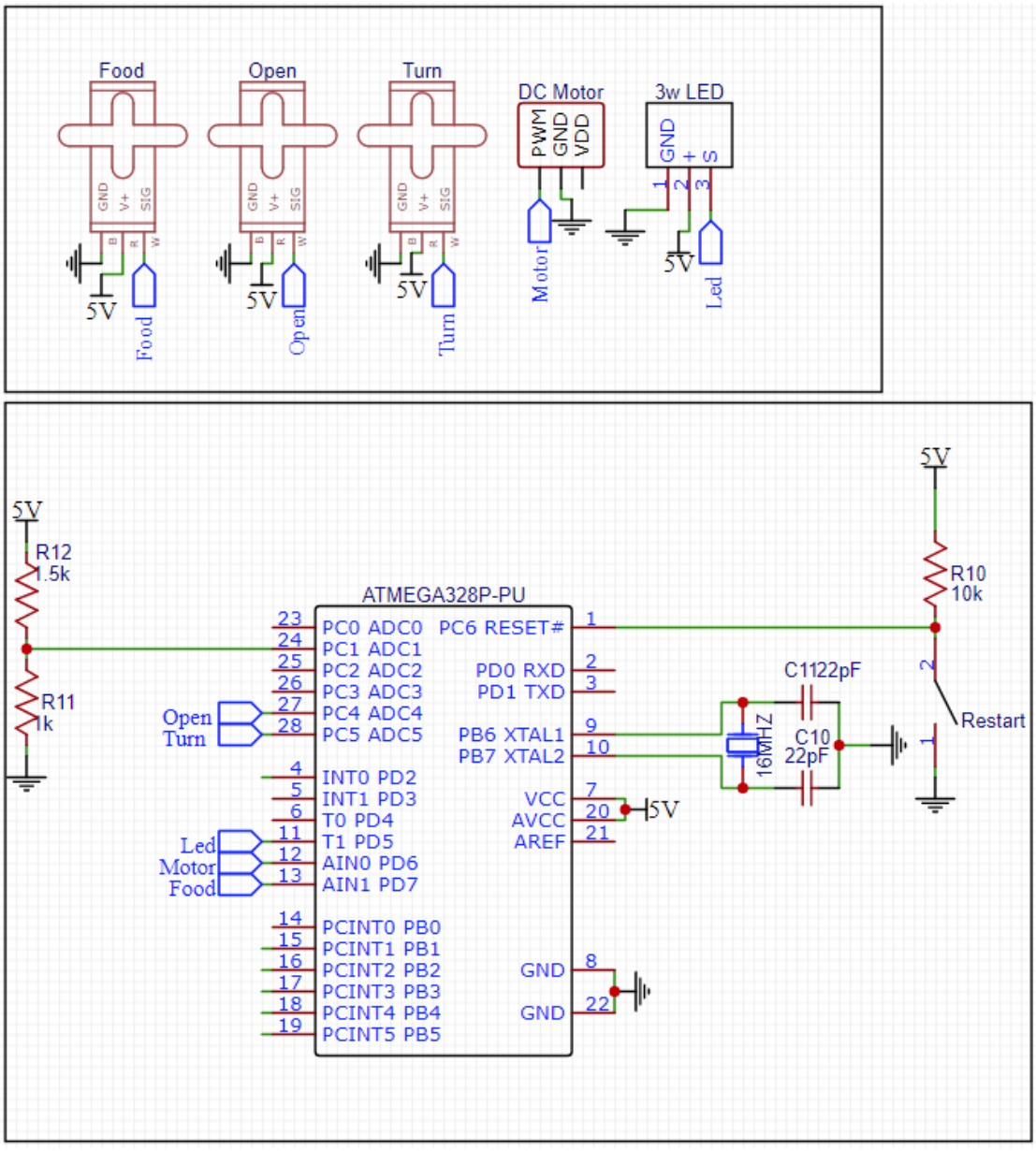
Επειδή, όπως βλέπουμε, ανεβάζει πολλές στροφές, άρα και θερμοκρασία, χρησιμοποιούμε υδρόψυξη. Το νερό περνάει γύρω από τον κινητήρα, έπειτα πάει στο ESC και από εκεί έχουμε μία έξοδο που βγάζει το νερό έξω από το σκαφάκι.

Σχήμα 4.14: Κινητήρας.

Το ESC έχει δικιά του μενού. Όταν βάλουμε μπροστά το κύκλωμα θα αρχίσει να χτυπάει κάποιες κόρνες ο ESC. Η κάθε κόρνα θα αντιστοιχεί σε κάποια λειτουργία. Η ρύθμιση του γίνεται ως εξής, αρχικά ανοίγουμε πρώτα το τηλεχειριστήριο και βάζουμε την μέγιστη τιμή στο ποτενσιόμετρο που ρυθμίζει το γκάζι. Έπειτα ενεργοποιούμε τον διακόπτη στο σκαφάκι και θα κάνει ένα θόρυβο ο ESC που θα μας επιβεβαιώνει ότι μπήκε στο μενού του. Στην συνέχεια θα αρχίσει να ακούγεται ένα μπιπ ανά λίγα δευτερόλεπτα, αυτό θα ακουστεί τέσσερις φορές. Αυτό είναι η πρώτη επιλογή στο μενού , αν κάνει δύο διαδοχικά μπιπ και μετά παύση για λίγα δευτερόλεπτα πάει στην δεύτερη επιλογή του μενού. Το μενού έχει τις επιλογές:

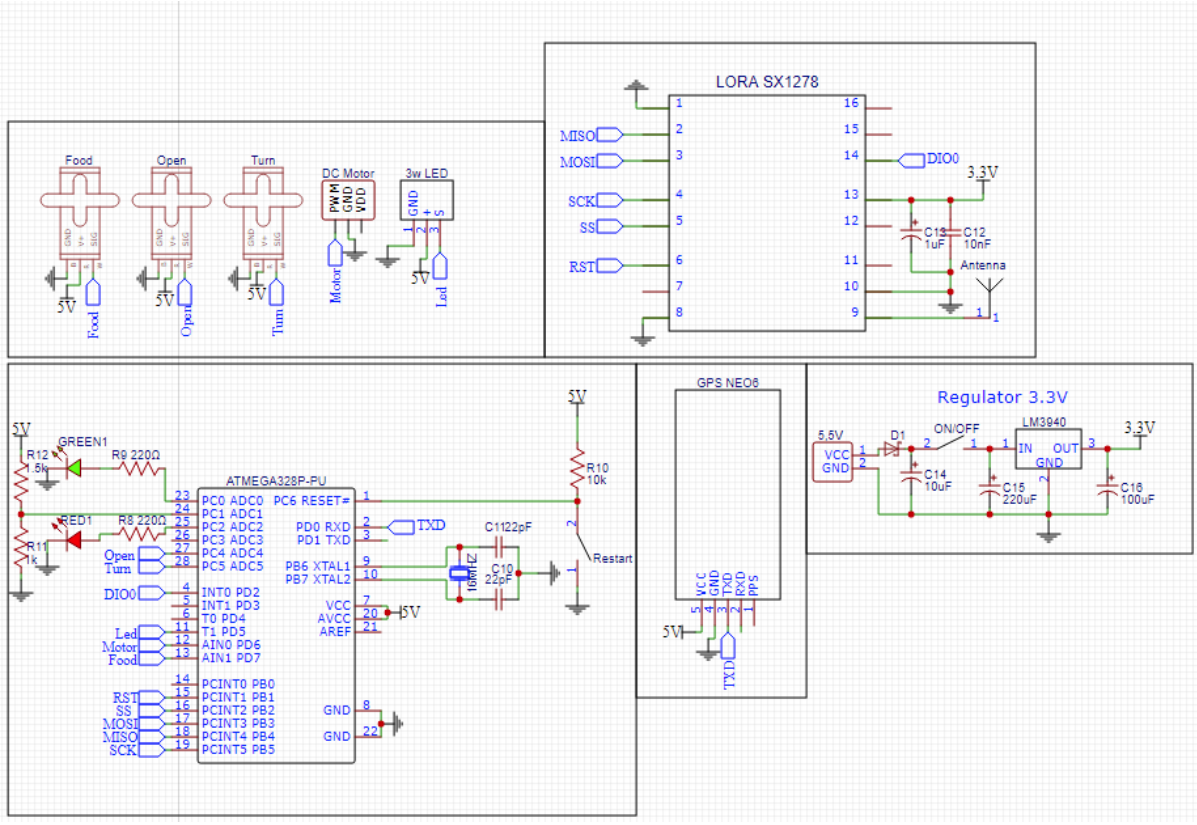
- Λειτουργίες φρένου.
- Τύπος μπαταριών.
- Voltage cutoff threshold.
- Επαναφορά ρυθμίσεων.
- Motor timing.
- Από ποια πλευρά θα γυρνάει ο κινητήρας.
- Κατά την εκκίνηση με τι ταχύτητα να ξεκινήσει.
- Low voltage cutoff type.

Από το μενού η μόνη λειτουργία που χρειαστήκαμε είναι κατά την εκκίνηση με τι ταχύτητα να ξεκινήσει. Για να το διαλέξουμε ακολουθήσαμε την εξής διαδικασία, αρχικά περιμέναμε να δούμε πότε θα ακουστούν τα μπιπ που δείχνουν ότι φτάσαμε σε αυτή την επιλογή. Όταν φτάσει αυτό το σημείο τότε κατεβάζουμε το ποτενσιόμετρο στην ελάχιστη τιμή δηλαδή μηδενίζουμε το γκάζι. Μετά βγάζουμε τις μπαταρίες από το σκαφάκι και τις ξαναβάζουμε για να περάσουν οι καινούργιες ρυθμίσεις. Όταν ανοίγουμε το σκαφάκι θα πρέπει από το τηλεχειριστήριο να μην έχουμε την μέγιστη τιμή στο γκάζι για να μην μπει στο μενού ο ESC.

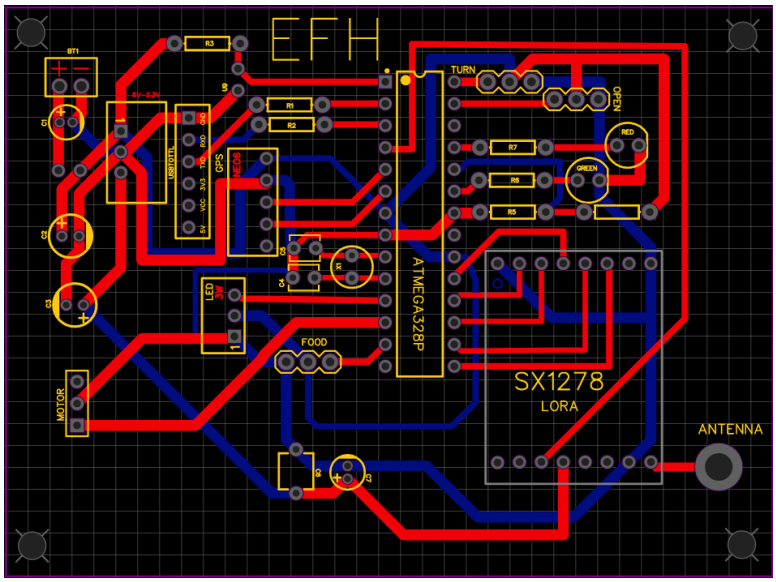


Σχήμα 4.15: Σύνδεση σερβοκινητήρα, κινητήρα και led με τον μικροελεγκτή.

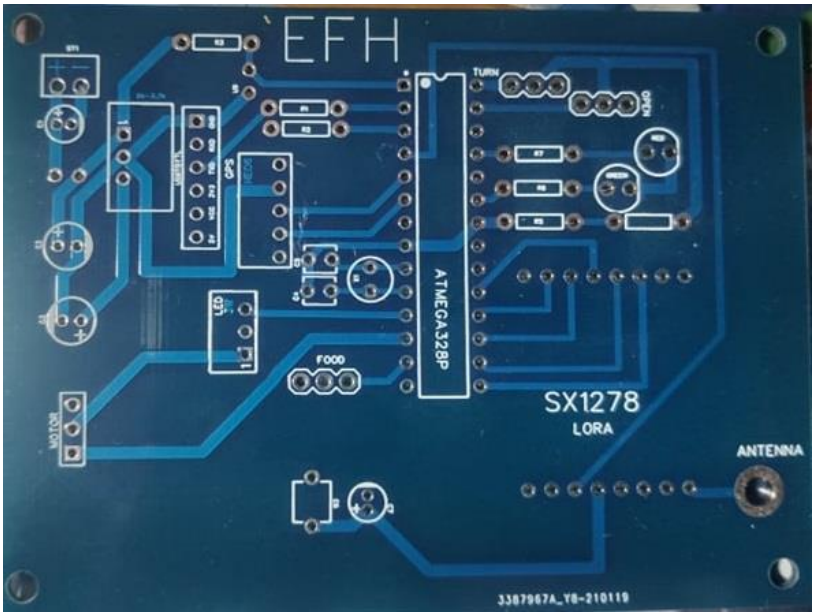
Το Lora κάνει ακριβώς τις ίδιες λειτουργίες με το τηλεχειριστήριο. Οι τιμές στην συχνότητα στο BW, SF κ.λπ. είναι ίδιες. Η μόνη διαφορά είναι ότι τώρα στέλνει float αριθμούς και δέχεται int. Επίσης, τα led τα χρησιμοποιούσαμε για το πειραματικό στάδιο. Στο πρακτικό κομμάτι δεν βοηθάνε πουθενά γιατί δεν θα μπορούμε να τα δούμε.



Σχήμα 4.16: Τελικό κύκλωμα.



Σχήμα 4.17: Σχεδίαση πλακέτας.



Σχήμα 4.18: Πλακέτα σκάφους.

4.4 Εξωτερικό μέρος πλοίου

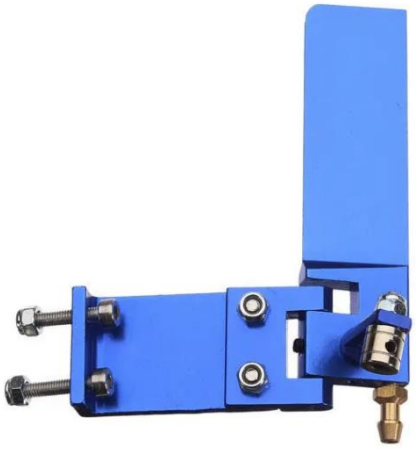
Το σκαφάκι κατασκευάστηκε με πολυεστέρα. Για να μπορούμε να έχουμε πρόσβαση μέσα χρησιμοποιήθηκαν δύο υδραυλικά καπάκια. Αυτά τα καπάκια κουμπώνουν και ξεκουμπώνουν. Η τοποθέτηση τους έγινε καρφώνοντας με βίδες και τοποθετήσαμε σιλικόνη από κάτω ώστε να μην περάσει νερό. Χρησιμοποιήθηκαν δύο πλαστικά κουτάκια τα οποία τα βγάλαμε από άλλη κατασκευή και τα τοποθετήσαμε πάνω στο πλοίο, το ένα για να καλύπτει το led και το άλλο για να μπορεί να βλέπει η κεραία από το gps στον έξω χώρο, χωρίς να είναι εκτεθειμένη στο νερό. Η σύνδεση του κινητήρα με τον άξονα έγινε με ένα συνδετικό σπαστό. Αν βάζαμε συνδετικό που δεν σπάει θα έπρεπε ο κινητήρας με τον άξονα να είναι ευθεία διότι αν είχε ελάχιστη απόκλιση θα τριβόταν μέσα ο άξονας.

Σχήμα 4.19: Συνδετικό σπαστό, σύνδεση άξονα – κινητήρα.

Σχήμα 4.20: Άξονας.

Ο άξονας περιέχει μία ράβδο μέσα του και έτσι γυρνάει. Η μικρή του άκρη συνδέεται με το συνδετικό και έπειτα στο κινητήρα. Η άλλη άκρη έχει ένα κούμπωμα που το βγάζουμε βάζουμε τον έλικα και το ξανακουμπώνουμε. Μέσα στον άξονα βάλαμε γράσο ώστε να γυρνάει πιο εύκολα και για να μην αφήνει να μπαίνει θαλασσινό νερό. Αυτό το γράσο θα ανανεώνεται ανά κάποιο χρονικό διάστημα.

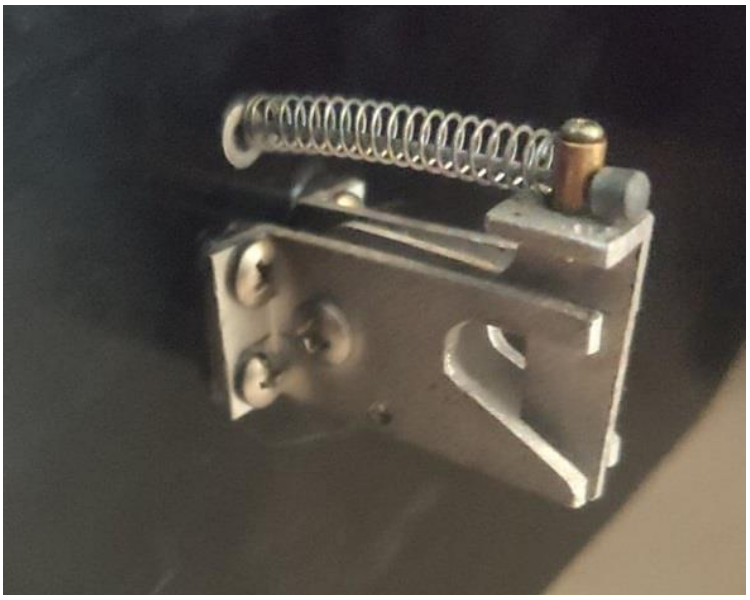
Το πηδάλιο που χρησιμοποιήθηκε είναι μεταλλικό και μπορούμε να το βάλουμε σαν είσοδο για την υδρόψυξη, αλλά δεν θα ήταν πρακτικό γιατί το σκαφάκι δεν δημιουργήθηκε για να τρέχει, άρα θα έχει μικρή πίεση το νερό και δεν θα κυλάει. Για αυτό τοποθετήθηκε ο σωλήνας μπροστά από τον έλικα. Ο έλικας όταν έχει μεγάλες στροφές θα σπρώχνει το νερό προς τον σωλήνα, οπότε θα έχει αρκετή πίεση το νερό ώστε να κυλάει μέσα στους σωλήνες. Το πηδάλιο γυρνάει με τον σερβοκινητήρα. Έχει ο σερβοκινητήρας μια σιδερένια ράβδο που σπρώχνει τον έλικα.



Σχήμα 4.21: Πηδάλιο.

Σχήμα 4.22: Πίσω μέρος του σκάφους.

Αυτό που θα απαγκιστρώνει είναι συνδεδεμένο με ένα σερβοκινητήρα μέσο ενός συρματόσχοινου. Στην μία περίπτωση ο σερβοκινητήρας θα τραβάει το συρματόσχοινο και θα ανοίγει το πάνω μέρος του απαγκιστρωτής και θα αφήνει να πέφτει το αγκίστρι κάτω. Στην άλλη περίπτωση θα αφήνει ελεύθερο το συρματόσχοινο και θα κλείσει.



Σχήμα 4.23: Απαγκιστρωτής.

Ο τρόπος που θα ρίχνει την μαλάγρα είναι αντίθετος. Όταν το συρματόσχοινο είναι ελεύθερο τότε θα ρίχνει την μαλάγρα. Όταν ο σερβοκινητήρας θα τραβάει το συρματόσχοινο θα επανέρχεται στην αρχική του θέση.

Σχήμα 4.24: Το εξάρτημα που θα ρίχνει την μαλάγρα.

Τέλος, για να διασφαλίσουμε την στεγανοποίηση στο σκάφος βάλαμε σε κάθε τρύπα σιλικόνη και η ένωση του πάνω με το κάτω μέρος θα γίνει επίσης με σιλικόνη. Η σιλικόνη που πήραμε είναι ιδική για νερό, έτσι θα σιγουρέψουμε ότι δεν θα υπάρξει πρόσβαση νερού εντός του σκάφους.

Σχήμα 4.25: Εσωτερικό μέρος σκάφους.

Κεφάλαιο 5^{ο}: Συμπεράσματα και Βελτιώσεις

Ο στόχος της εργασίας αυτής ήταν να δείξει την πρακτικότητα ενός τέτοιου σκάφους καθώς και την διαδικασία που χρησιμοποιήθηκε για να υλοποιηθεί. Στο τέλος της κατασκευής προέκυψαν μερικές παρατηρήσεις οι οποίες θα βελτίωναν την απόδοση του σκάφους.

Η πρώτη παρατήρησή μας αφορά τον μικροελεγκτή. Ο μικροελεγκτής μας διαθέτει πολλές λειτουργίες σε κάθε ποδαράκι. Έτσι όταν χρησιμοποιήσαμε το SPI πρωτόκολλο, ξοδέψαμε σχεδόν όλα τα ποδαράκια PWM. Έτσι για να ξεπεραστεί αυτό το πρόβλημα χρησιμοποιήθηκε software που κάνουμε και πόδια που δεν δίνουν PWM να δώσουν το σήμα που χρειάζεται.

Η τάση που μας βγάζει το BEC του ESC δεν το κατεβάσαμε ακριβώς στα 5V ακόμα και με την δίοδο. Η δίοδος μας το πήγε στα 5.15V. Παρόλο που είναι στα όρια όλων των εξαρτημάτων που χρησιμοποιούμε, δεν συνίσταται. Οι επιλογές που θα είχαμε αν δεν το κάναμε αυτό είναι να βάλουμε δίοδο που θα μας το κατέβαζε 0.7V άρα θα πέφταμε κάτω από τα 5V και θα έχαναν δύναμη οι σερβοκινητήρες. Η τελευταία επιλογή θα ήταν να χρησιμοποιούμε σταθεροποιητή στα 5V. Ο σταθεροποιητής δεν θα ήταν επαρκής για το ρεύμα που θα μας ζητούσε το κύκλωμα σε πλήρη λειτουργία με φορτίο.

Μια πολύ σημαντική παρατήρηση είναι όταν πειράξαμε τις ρυθμίσεις του lora, δηλαδή το SF, BW, CR κ.λπ. διαπιστώθηκε ότι με λάθος αριθμούς το πακέτο έφτανε παραμορφωμένο ακόμα και σε κοντινή απόσταση. Επίσης διαπιστώθηκε με ένα υψηλό BW και ένα μέτριο SR πόσο γρήγορη ανταπόκριση ξεκίνησε να έχει το lora.

Μια τελευταία παρατήρηση είναι πόσο πολύ επηρεάζει το κάθε δευτερόλεπτο στην αποστολή και την λήψη. Η lcd χρειάζεται κάποια ms για να γράψει η να διαγράψει κάτι. Σαν αποτέλεσμα είχε ότι μας κολλούσε το lora. Ήταν στιγμές που έστελνε 8 δευτερόλεπτα και 4 δευτερόλεπτα δεν ανταποκρινόταν καθόλου. Αυτό το έκανε όση ώρα και να το είχαμε ανοιχτό. Για να λυθεί το πρόβλημα έπρεπε να κάνουμε την lcd να γράφει μόνο όταν είναι απαραίτητο ώστε να σταματήσει να κολλάει.

Με βάση τα παραπάνω καταλήξαμε ότι για την επιτυχή υλοποίηση ενός τηλεχειριζόμενο σκάφους πρέπει να προσέξουμε ξεχωριστά αρκετές παραμέτρους που η κάθε μία παίζει το ρόλο της. Η εργασία είναι επιτυχής από την στιγμή που καταφέραμε να εκπληρώσουμε τους στόχους που τέθηκαν.

Έχοντας ολοκληρώσει την εργασία και αφού τέθηκε σε λειτουργία το όργανο θα μπορούσε να προστεθεί για την πιο αποτελεσματική λειτουργία ένας άλλος μικροελεγκτής με περισσότερα ποδαράκια, ώστε να καλύπτει τις απαιτήσεις του κυκλώματος χωρίς την προσθήκη επιπλέον software.

Τέλος κάτι ακόμα που θα μπορούσε να βελτιώσει όλη την κατασκευή είναι ένα βυθόμετρο. Το βυθόμετρο θα μας βοηθούσε να επιλέγουμε το βάθος που θα ψαρέψουμε.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Site

[1] https://el.wikipedia.org/wiki/Αλιεία

[2] https://4fishingtackle.gr/2017/10/07/το-ψάρεμα-στην-αίγυπτο/

[3] https://www.boatfishing.gr/article-post/afieroma-technikes-gia-kalokairina-psaremata-apo-akti/

[4] https://www.drososfishing.gr/mpolonez-sta-vrachia-me-monokommati-armatosia/

[5] https://fishingbooker.com/blog/drone-fishing-is-it-really-fishing/

[6] https://dronefishing.com/blogs/news/what-is-drone-fishing

[7] https://en.m.wikipedia.org/wiki/Remote_control_fishing?fbclid=IwAR1Vk6an0dZxl-YR8EPUk_EgFHkny9QvPFOvWPNncHwIV3ikNQ1xqsYnR3U

[8] https://www.amperorio.gr/index.php/el/texnika-themata-syssvreutwn/item/351-mpataries-ionton-lithiou-typoi-kai-xriseis.html

[9] https://www.cei.washington.edu/education/science-of-solar/battery-technology/

[10] https://el.wikipedia.org/wiki/Μπαταρία_ιόντων_λιθίου

[11] https://link.springer.com/article/10.1007/s12209-020-00236-w

[12] https://www.cnydrones.org/lipo-batteries-and-safety-for-beginners/

[13] https://www.skafakigiapsarema.gr/index.php/blog/118-lipo-batts

[14] https://rogershobbycenter.com/lipoguide

[15] https://www.best-microcontroller-projects.com/tp4056.html

[16] https://www.sunrom.com/p/lithum-battery-charger-with-protection-microusb

[17] https://www.electrical4u.com/voltage-regulator-7805/

[18] https://www.tokalo.gr/BLDC_motors.html

[19] https://el.wikipedia.org/wiki/Ηλεκτρικός_κινητήρας

[20] https://www.tokalo.gr/motors.html

[21] https://www.getcert.gr/pos-leitourgei-to-stepper-motor/

[22] http://gr.hanzelmotor.org/news/brushless-dc-motors-low-inertia-fast-respons-10997147.html

[23] https://www.hwlibre.com/el/motor-brushless/

[24] https://gfycat.com/faithfulcrispivorybilledwoodpecker

[25] https://www.modelflight.com.au/blog/electronic-speed-controllers

[26] https://howtomechatronics.com/tutorials/arduino/arduino-brushless-motor-control-tutorial-esc-bldc/

[27] https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

[28] https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html

[29] https://alexkaltsas.wordpress.com/2012/03/18/avr-gcc-σερβοκινητήρες-μοντελισμού-aka-servos/

[30] https://en.wikipedia.org/wiki/LoRa

[31] https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[32] http://web.luxresearchinc.com/hubfs/Insight_Breakdown_of_LPWAN_Technologies.pdf

[33] https://psihogios.com/technologia-lora-long-range-iot/

[34] https://www.pcsteps.gr/1412-πως-λειτουργεί-η-πλοήγηση-gps-στο-κινητό/

[35] https://el.wikipedia.org/wiki/Global_Positioning_System

[36] https://el.wikipedia.org/wiki/Μικροελεγκτής

[37] https://exploreembedded.com/wiki/Analog_JoyStick_with_Arduino

[38] https://learnelectronics.gr/το-πρωτόκολλο-επικοινωνίας-spi/

[39] https://www.14core.com/wiring-i2c-module-on-16x2-lcd-with-sclsda/

[40] https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life

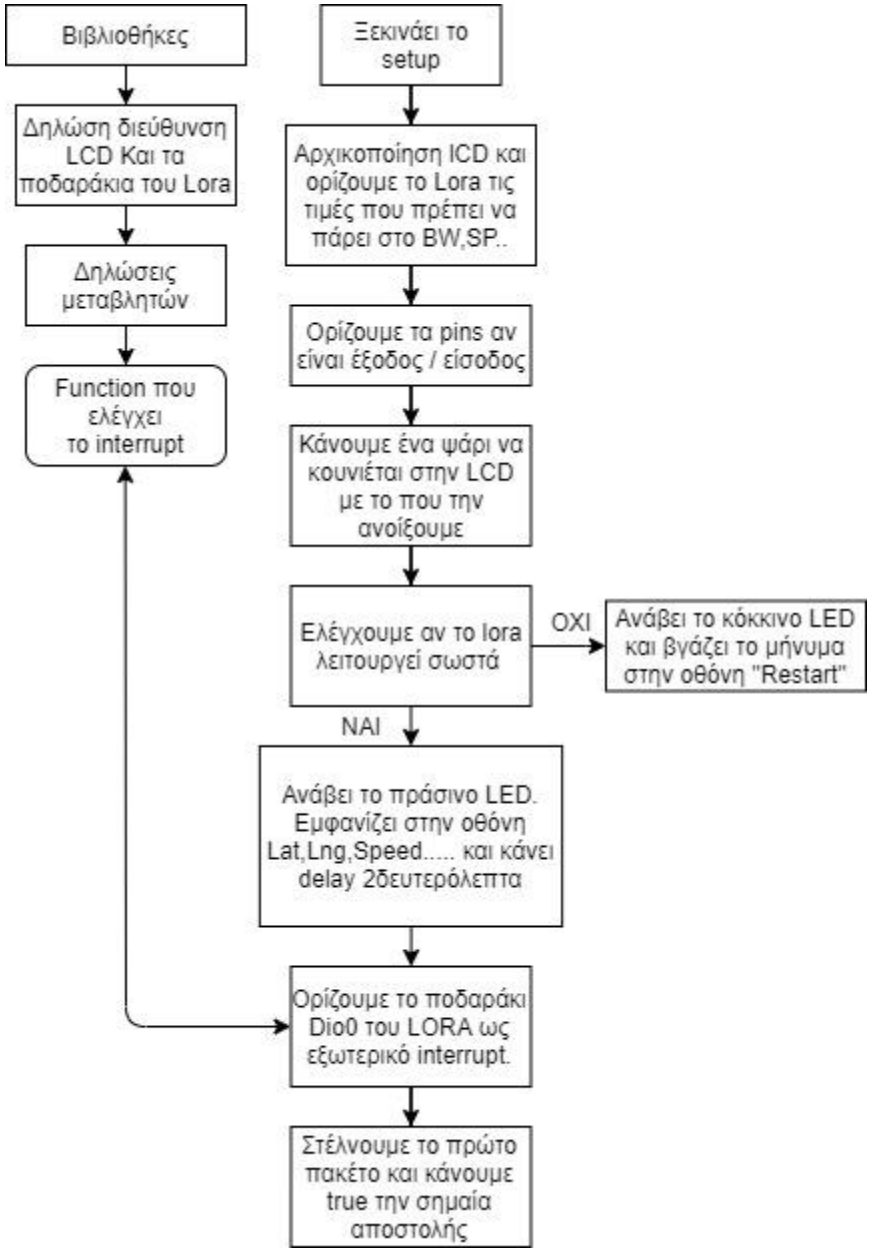
[41] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7070984/

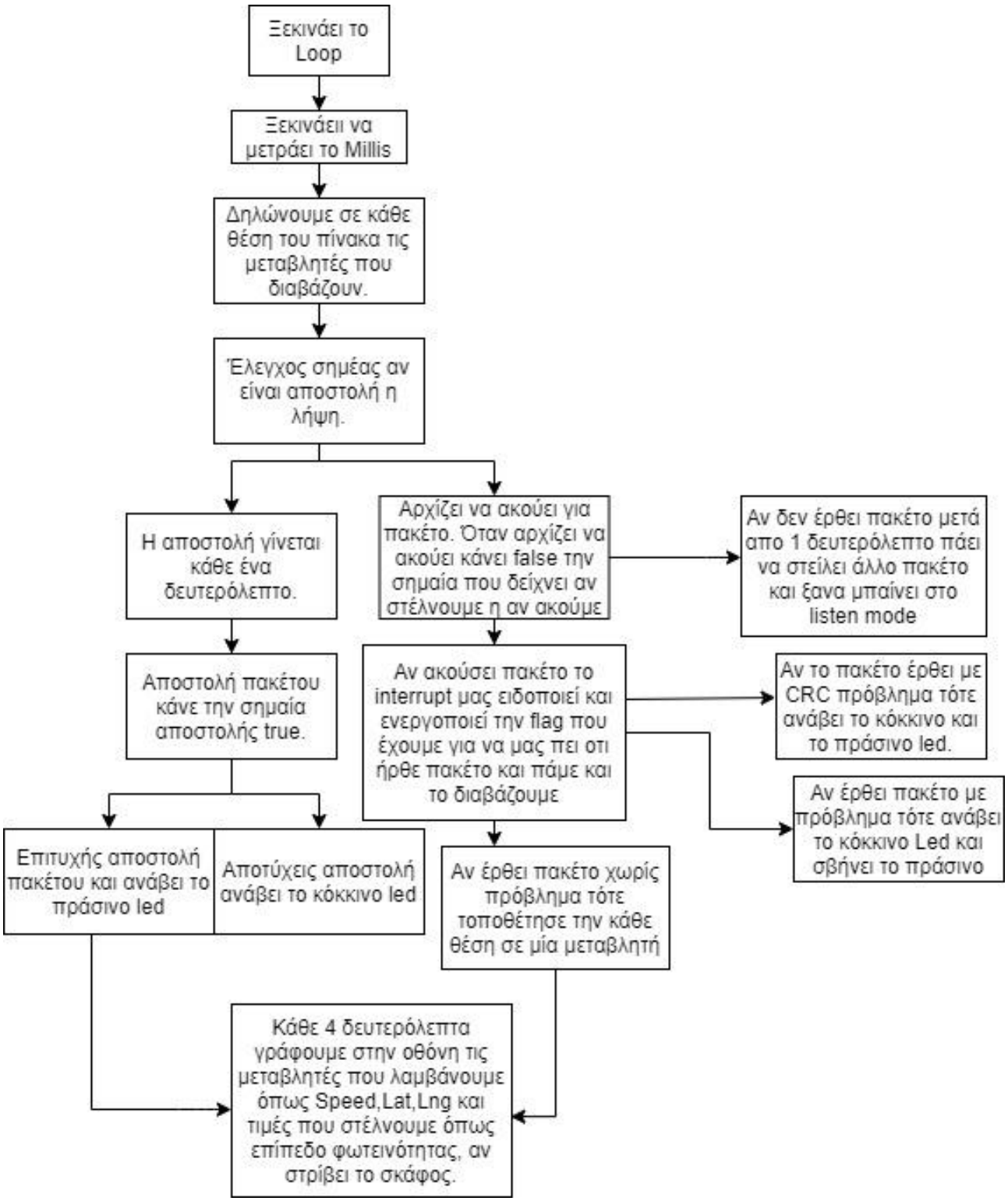
[42] https://eprints.lancs.ac.uk/id/eprint/85515/4/lora_tps_r1342.pdf

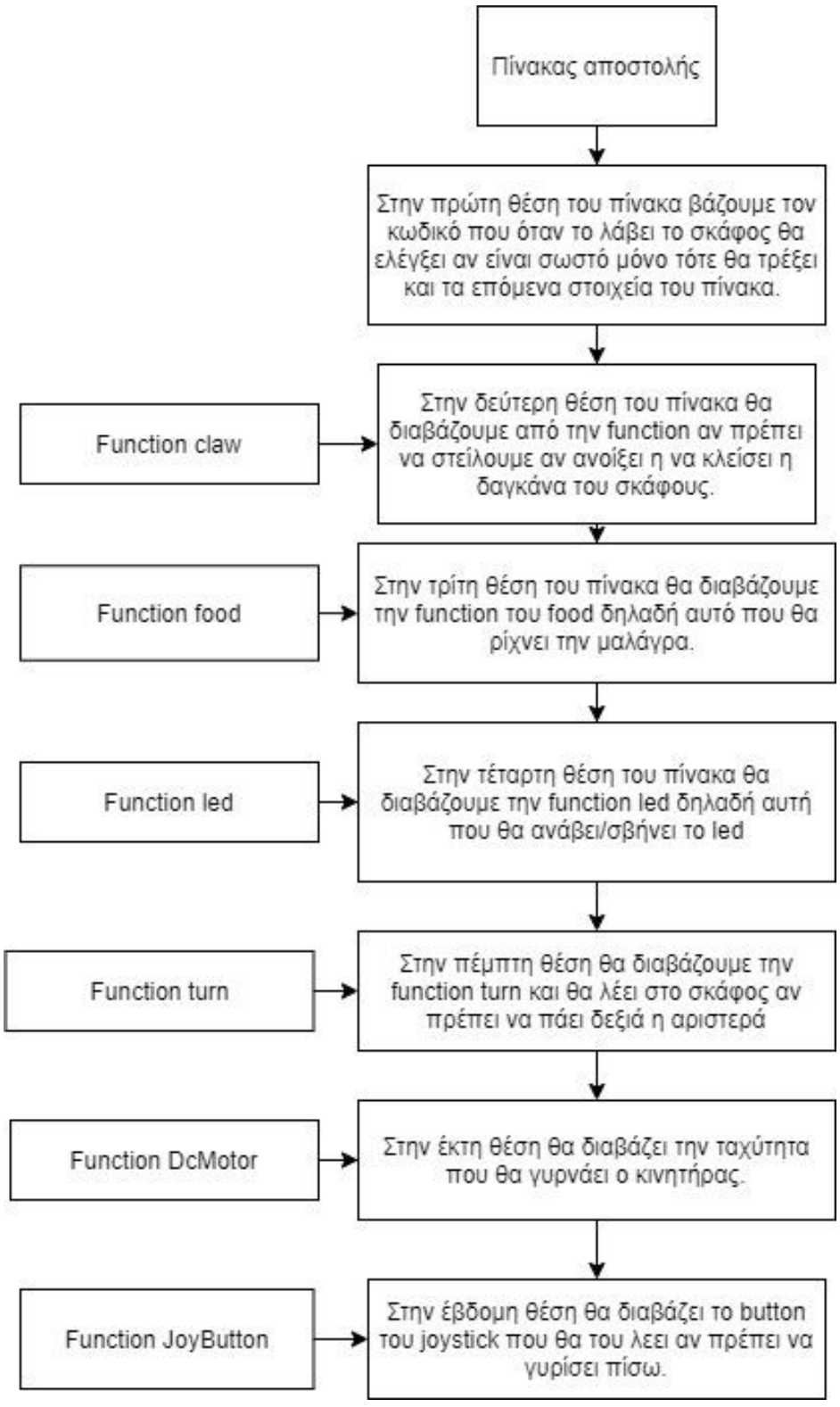
[43] http://e-lektronikos.blogspot.com/2013/04/diodos-zener-vs-diodos-schottky.html

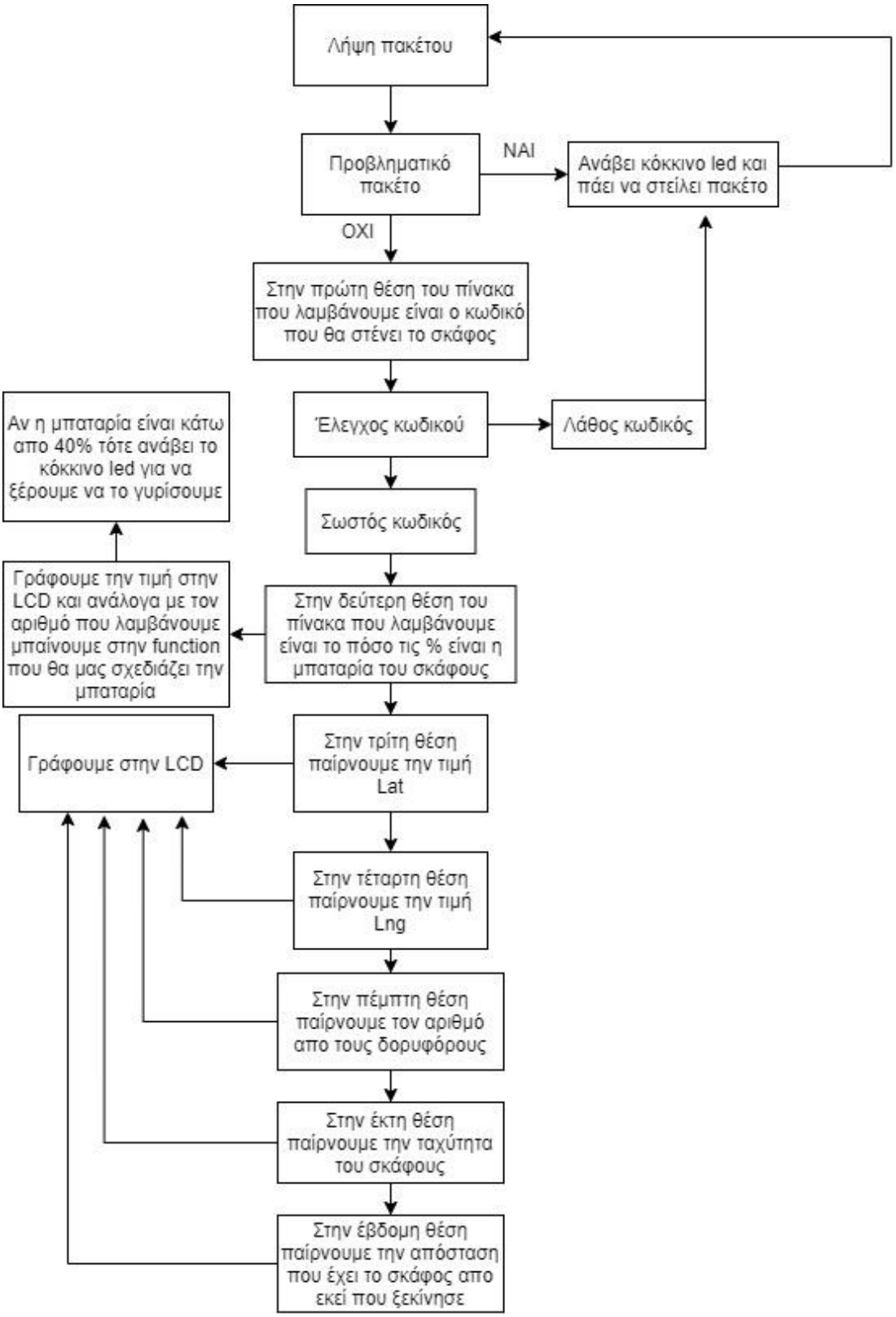
ΠΑΡΑΡΤΗΜΑ Α

Σε αυτό το παράρτημα θα παρουσιάσουμε το διάγραμμα ροής του κώδικα στο τηλεχειριστήριο.




Ξεκινάει το Loop
Ξεκινάειι να μετράει το Millis
Δηλώνουμε σε κάθε θέση του πίνακα τις μεταβλητές που διαβάζουν.
Έλεγχος σημέας αν είναι αποστολή η λήψη.
Η αποστολή γίνεται κάθε ένα δευτερόλεπτο.
Αποστολή πακέτου κάνε την σημαία αποστολής true.
Επιτυχής αποστολή πακέτου και ανάβει το πράσινο led
Αποτύχεις αποστολή ανάβει το κόκκινο led
Αρχίζει να ακούει για πακέτο. Όταν αρχίζει να ακούει κάνει false την σημαία που δείχνει αν στέλνουμε η αν ακούμε
Αν δεν έρθει πακέτο μετά απο 1 δευτερόλεπτο πάει να στείλει άλλο πακέτο και ξανα μπαίνει στο listen mode
Αν ακούσει πακέτο το interrupt μας ειδοποιεί και ενεργοποιεί την flag που έχουμε για να μας πει οτι ήρθε πακέτο και πάμε και το διαβάζουμε
Αν το πακέτο έρθει με CRC πρόβλημα τότε ανάβει το κόκκινο και το πράσινο led.
Αν έρθει πακέτο με πρόβλημα τότε ανάβει το κόκκινο Led και σβήνει το πράσινο
Αν έρθει πακέτο χωρίς πρόβλημα τότε τοποθέτησε την κάθε θέση σε μία μεταβλητή
Κάθε 4 δευτερόλεπτα γράφουμε στην οθόνη τις μεταβλητές που λαμβάνουμε όπως Speed,Lat,Lng και τιμές που στέλνουμε όπως επίπεδο φωτεινότητας, αν στρίβει το σκάφος.


Πίνακας αποστολής
Στην πρώτη θέση του πίνακα βάζουμε τον κωδικό που όταν το λάβει το σκάφος θα ελέγξει αν είναι σωστό μόνο τότε θα τρέξει και τα επόμενα στοιχεία του πίνακα.
Function claw
Στην δεύτερη θέση του πίνακα θα διαβάζουμε από την function αν πρέπει να στείλουμε αν ανοίξει η να κλείσει η δαγκάνα του σκάφους.
Function food
Στην τρίτη θέση του πίνακα θα διαβάζουμε την function του food δηλαδή αυτό που θα ρίχνει την μαλάγρα.
Function led
Στην τέταρτη θέση του πίνακα θα διαβάζουμε την function led δηλαδή αυτή που θα ανάβει/σβήνει το led
Function turn
Στην πέμπτη θέση θα διαβάζουμε την function turn και θα λέει στο σκάφος αν πρέπει να πάει δεξιά η αριστερά
Function DcMotor
Στην έκτη θέση θα διαβάζει την ταχύτητα που θα γυρνάει ο κινητήρας.
Function JoyButton
Στην έβδομη θέση θα διαβάζει το button του joystick που θα του λεει αν πρέπει να γυρίσει πίσω.


Λήψη πακέτου
Προβληματικό πακέτο
ΝΑΙ
Ανάβει κόκκινο led και πάει να στείλει πακέτο
ΟΧΙ
Στην πρώτη θέση του πίνακα που λαμβάνουμε είναι ο κωδικό που θα στένει το σκάφος
Έλεγχος κωδικού
Λάθος κωδικός
Σωστός κωδικός
Στην δεύτερη θέση του πίνακα που λαμβάνουμε είναι το πόσο τις % είναι η μπαταρία του σκάφους
Γράφουμε την τιμή στην LCD και ανάλογα με τον αριθμό που λαμβάνουμε μπαίνουμε στην function που θα μας σχεδιάζει την μπαταρία
Αν η μπαταρία είναι κάτω απο 40% τότε ανάβει το κόκκινο led για να ξέρουμε να το γυρίσουμε
Στην τρίτη θέση παίρνουμε την τιμή Lat
Γράφουμε στην LCD
Στην τέταρτη θέση παίρνουμε την τιμή Lng
Στην πέμπτη θέση παίρνουμε τον αριθμό απο τους δορυφόρους
Στην έκτη θέση παίρνουμε την ταχύτητα του σκάφους
Στην έβδομη θέση παίρνουμε την απόσταση που έχει το σκάφος απο εκεί που ξεκίνησε

```
/******************** Βιβλιοθήκες ********************/

#include <RadioLib.h>          // Βιβλιοθήκη για το Lora.
#include <LiquidCrystal_I2C.h>  // Βιβλιοθήκη για την LCD.
#include <Wire.h>              // Βιβλιοθήκη για το I2C

/******************** Ορισμός LCD-Lora ********************/

LiquidCrystal_I2C lcd(0x27, 20, 4);  //Δηλώνω  πως η LCD ειναι 20 χαρακτήρων 4 σειρών. Το 0χ27 υποδηλώνει την διεύθυνση του I2C.
SX1278 radio = new Module(10, 2, 9);  //Δηλώνω στο digital 10 του atmel συνδέεται το NSS του LORA, digital 2 το DIO0 (Interrupt), digital 9 το RESET.

/******************** Σχηματισμός στην LCD ********************/

byte fish[8] =   // Δημιουργεί ένα ψάρι.
{
  B00000,
  B00000,
  B10110,
  B11001,
  B11001,
  B10110,
  B00000,
  B00000
};

byte lock[8] =  // Δημιουργεί μία κλειδαριά .
{
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11111,
  B11111
};

byte LockOpen[8] // Δημιουργεί μία ανοιχτή κλειδαριά.
{
  B00110,
  B00001,
  B00001,
  B11111,
  B11011,
  B11011,
  B11111,
  B11111
};

byte ArrowRight[8] = // Δημιουργεί ένα βέλος δεξιά.
{
  B00000,
  B00100,
```

```
  B00010,
  B11111,
  B00010,
  B00100,
  B00000,
  B00000
};

byte ArrowLeft[8] =  // Δημιουργεί ένα βέλος αριστερά.
{
  B00000,
  B00100,
  B01000,
  B11111,
  B01000,
  B00100,
  B00000,
  B00000
};

byte Ret[8] = // Δημιουργεί ένα σήμα που δείχνει οτι επιστρέφει.
{
  B00100,
  B01000,
  B01111,
  B01001,
  B10101,
  B10001,
  B11111,
  B00000
};
/******************** Flags/Interrupts ********************/

bool transmitFlag = false;            // Flag που μας δείχνει αν είναι αποστολή η λήψη.
volatile bool enableInterrupt = true;  // Αυτή η μεταβλητή ενεργοποιεί και απενεργοποιεί την διακοπή.
volatile bool operationDone = false;   // Αυτή η μεταβλητή μας δείχνει πότε στάλθηκε/λήφθηκε ένα πακέτο.

/******************** Μεταβλητές ********************/

typedef struct remote   // Για να γλιτώσουμε τα περιττά bit χρησιμοποιούμε το struct.
{
  uint16_t Pot           : 10;   // Τιμές 0-1023.
  uint16_t JoystickY     : 10;   // Τιμές 0-1023.
  uint16_t JoystickX     : 10;   // Τιμές 0-1023.
  uint8_t  motor;                // Τιμές 0-180.
  uint8_t  Degrees       : 3;    // Τιμές 0-5.
  uint8_t  LedW = 0;     : 2;    // Τιμές 0-3.
  uint8_t  Meter = 0;    : 2;    // Τιμές 0-3.
  uint8_t satellite;
  uint8_t  Pin = 231;
  uint8_t PinSend;
  uint8_t  Food          : 1;    // Τιμές 0 ή 1.
  uint8_t  ReadFood      : 1;    // Τιμές 0 ή 1.
  uint8_t  Dagana;       : 1;    // Τιμές 0 ή 1.
```

```
  uint8_t  ReadDagana      : 1;    // Τιμές 0 ή 1.
  uint8_t RedLed = 6;      : 1;    // Τιμές 0 ή 1. Επίσης δηλώνω το κόκκινο ποδαράκι είναι το Digital 6.
  uint8_t GreenLed = 7;    : 1;    // Τιμές 0 ή 1. Επίσης δηλώνω το κόκκινο ποδαράκι είναι το Digital 7.
  uint8_t Button           : 1;
};

struct remote Control;

float Lng;     // Μεταβλητή για το lng.
float Lat;     // Μεταβλητή για το Lat.
float Distance;
float Run;
int timh = 0;  // Μεταβλητή για την μπαταρία.

uint16_t sendPacket[7]; // Αυτός ο πίνακας θα αποθηκεύει τις μεταβλητές που θα στέλνει.
float recePacket[7];    // Αυτός ο πίνακας θα αποθηκεύει τις μεταβλητές που θα λαμβάνει.

unsigned long SendTime = 1000;
unsigned long previousSend = 0;
unsigned long LcdTime = 4000;
unsigned long previousLcd = 0;

int TraState = ERR_NONE;   // Αυτή η μεταβλητή ελέγχει την επικοινωνία της αποστολής.
int ReceState = ERR_NONE;  // Αυτή η μεταβλητή ελέγχει την επικοινωνία δέκτη.

/******************** Έλεγχος Interrupt ********************/

void setFlag(void)
{
  // Έλεγχος αν το interrupt είναι ενεργοποιημένο.
  if (!enableInterrupt)
  {
    return;
  }
  operationDone = true;  // Στείλαμε η λάβαμε πακέτο, κάνε true το                flag.
}

/******************** Ξεκινάει το πρόγραμμα ********************/

void setup()
{
  Serial.begin(9600);
  lcd.init();              // Αρχικοποιήση LCD

  radio.begin(433.5, 500.0, 8, 7, 0x18, 17, 20, 0 );
  //Frequency        : 433.5 MHz
  //Bandwidth        : 500 kHz
  //Spreading factor : 8
  //Coding rate      : 7
  //Sync word        : 0x18
  //Output power     : 14
  //Preamble length  : 20
  //Amplifier gain   : 0
```

```
  pinMode(Control.RedLed, OUTPUT);   // Δηλώνουμε το pin που ειναι συνδεδεμένο το κόκκινο led έξοδο.
  pinMode(Control.GreenLed, OUTPUT); // Δηλώνουμε το pin που ειναι συνδεδεμένο το πράσινο led έξοδο.
  pinMode(5, INPUT);                 // Δηλώνουμε το digital 5 είσοδο.
  pinMode(4, INPUT);                 // Δηλώνουμε το digital 4 είσοδο.
  pinMode(A1, INPUT);

  /***************** ΞΕΚΙΝΑΕΙ Η LCD ****************/


  lcd.backlight();              // Ανοίγει το backlight της οθόνης.
  lcd.setCursor(8, 0);          // Γράφει στην LCD στην πρώτη σειρά έκτη στήλη.
  lcd.print("E.F.H");           // Εκτύπωση μηνύματος.
  lcd.createChar(0, fish); // Φτιάχνει τον χαρακτήρα.

  for ( int i = 0; i < 19; i++) // Θα μετακινήσει το ψάρι που βρίσκεται στην δεύτερη σειρά.
  {
    lcd.setCursor(i, 1);        // Αλλάζει την στήλη.
    lcd.write(((byte)0));       // Εμφανίζεται στην κενούργια στήλη ο χαρακτήρας.
    delay(100);
    lcd.setCursor(i, 1);
    lcd.write(' ');             // Διαγράφει τον χαρακτήρα.
  }

  /***************** ΕΛΕΓΧΟΣ ΑΝ ΥΠΑΡΧΕΙ ΣΦΑΛΜΑ ΣΤΟ lORA  *****************/
  int j = 0;

  while (radio.begin() != ERR_NONE)    // Σε περίπτωση που το Lora.begin έχει κάποια σφάλμα μπαίνει σε αυτή την while.
  {
    for ( int i = 0; i < 15 ; i++)   // Σε αυτή την for θα μετακινεί το ψάρι στην οθόνη όσο κάνει έλεγχο την Lora.begin.
    {
      lcd.setCursor(i, 1);  // Ελέγχει σε ποια θέση θα είναι το ψάρι.
      lcd.write(((byte)0)); // Δημιουργεί το ψάρι.
      delay(100);
      lcd.setCursor(i, 1);
      lcd.write(' ');      // Διαγράφει το ψάρι.
    }
    delay(100);
    j++;
    digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
    digitalWrite(Control.RedLed, HIGH);   // Ανάβει το κόκκινο led.

    if ( j > 10)    // Όταν κάνει 10 φορές τον έλεγχο αν δεν διορθωθεί θα μπεί σε αυτη την for και θα μας δείξει η lcd
    { // για να κάνουμε restart το τηλεχειριστήριο.
      lcd.clear();
      lcd.setCursor(4, 0);

      for (;;)
      {
        lcd.print("restart");
        digitalWrite(Control.RedLed, HIGH); // Ανοίγει το κόκκινο Led.
```

```
    }
   }

  }//end while
  lcd.clear();  //Καθαρίζει την LCD.
  if (radio.setCurrentLimit(100) == ERR_INVALID_CURRENT_LIMIT) // Ορίζουμε 100mA ρεύμα.
  {
    digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
    digitalWrite(Control.RedLed, HIGH);   // Ανοίγει το κόκκινο led.
  }
  if (radio.setCRC(true) == ERR_INVALID_CRC_CONFIGURATION)  // Ενεργοποιούμε το CRC.
  {
    digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
    digitalWrite(Control.RedLed, HIGH);   // Ανοίγει το κόκκινο led.
  }
  digitalWrite(Control.GreenLed, HIGH);   // Αν δεν έχει κάποιο το Lora.begin τότε ανάβει το πράσινο
led/
  digitalWrite(Control.RedLed, LOW);      // Κλείνει το κόκκινο led.

  lcd.setCursor(15, 1);   // Ορίζουμε θέση.
  lcd.print("LED:");      // Εκτύπωνει LED:
  lcd.setCursor(0, 0);
  lcd.print(F("LAT:"));
  lcd.setCursor(0, 1);
  lcd.print(F("LNG:"));
  lcd.setCursor(0, 2);
  lcd.print(F("SPEED:"));
  lcd.setCursor(0, 3);
  lcd.print("DISTANCE:");
  delay(2000);          // Καθυστέρησε 2 δευτερόλεπτα.


  /**************** Τέλος ελέγχου ****************/
  radio.setDio0Action(setFlag);   // Δηλώνω το ποδαράκι του lora Dio0 να δηιαβάζει την fuction
setFlag.

  TraState = radio.startTransmit((uint8_t*)sendPacket, sizeof(sendPacket)); //Στέλνουμε το πρώτο
πακέτο.
  transmitFlag = true; // Μόλις στείλουμε το πρώτο πακέτο το κάνουμε true για να μπει στην
λειτουργία λήψης.
}

void loop()
{
  unsigned long currentMillis = millis(); // Αυτή η μεταβλητή θα μετράει μόνιμα.

  sendPacket[0] = Control.Pin;  // Δηλώνουμε κάθε θέση του πίνακα να παίρνει μία μεταβλητή.
  sendPacket[1] = claw();
  sendPacket[2] = food();
  sendPacket[4] = turn();
  sendPacket[5] = DcMotor();

  if ( currentMillis - previousSend >= SendTime)
  {
    previousSend = currentMillis;
```

```
   sendPacket[3] = led();  // Η θέση τρία του πίνακα θα παίρνει την τιμή του led Κάθε ένα δευτερόλεπτο.
   sendPacket[6] = JoyButton();
   TraState = radio.startTransmit((uint8_t*)sendPacket, sizeof(sendPacket));
   transmitFlag = true;    // Μόλις στείλουμε το πρώτο πακέτο το κάνουμε true για να μπει στην λειτουργία λήψης.
  }

 if (operationDone)  //Στείλαμε η λάβαμε πακέτο άρα γίνεται true και μπαίνει στο if.
 {
  enableInterrupt = false; // Κλείνουμε το Interrupt μέχρι να κάνουμε την δουλειά που χρειάζεται.
  operationDone = false;   // Κάνουμε reset το flag

  if (transmitFlag)       // Αν είχαμε στείλει πριν πακέτο και είναι η σημαία true μπαίνει στο if.
    // Αν ήταν στο receive mode τότε συνεχίζουμε τον κώδικα χωρίς να μπούμε σε αυτό το if.
  {
    if (TraState == ERR_NONE)
    {
      digitalWrite(Control.GreenLed, HIGH);   // Αν στάλθηκε με επιτυχία  τότε ανάβει το πράσινο led.
      digitalWrite(Control.RedLed, LOW);     // Κλείνει το κόκκινο.
    }
    else
    {
      digitalWrite(Control.GreenLed, LOW);   // Αν υπήρξε σφάλμα κλείνει το πράσινο LED.
      digitalWrite(Control.RedLed, HIGH);    // Ανάβει το κόκκινο.
    }
    radio.startReceive();                  // Πάμε στο Listen mode
    transmitFlag = false;                   // Όταν είμαστε στο listen mode το trasmitFlag πρέπει να είναι false.
  }

  else
  {
   ReceState = radio.readData((byte*)&recePacket,7*sizeof(float)); // Έρχεται ειδοποιήση απο το interrupt οτι δέχτηκε η στάλθηκε πακέτο,τότε ελέγχει ο μικροελεγκτής την σημαία transmitFlag αν είναι false ξέρει οτι ήρθε πακέτο και πρέπει να διαβάσει τα δεδομένα.

   if (ReceState == ERR_NONE)  // Αν το πακέτο είναι εντάξει τότε μπές στο If αλλιώς τρέξε το else.
   {
     Control.PinSend = recePacket[0]; // Λέμε στις μεταβλητές να διαβάσουν τις τιμές απο το πακέτο που λάβαμε.
     timh = recePacket[1];
     Lat = recePacket[2];
     Lng = recePacket[3];
     Control.satellite= recePacket[4];
     Run = recePacket[5];
     Distance = recePacket[6];
     Battery();       // Καλούμε την function.
     digitalWrite(Control.GreenLed, HIGH);
     digitalWrite(Control.RedLed, LOW);
   }
   else if(ReceState == ERR_CRC_MISMATCH)
   {
    digitalWrite(Control.GreenLed, HIGH); // Ανάβει το πράσινο led.
```

```
     digitalWrite(Control.RedLed, HIGH);   // Ανάβει το κόκκινο led.
    }
    else
    {
     timh = 0;
     digitalWrite(Control.GreenLed, LOW); // Κλείνει το πράσινο led.
     digitalWrite(Control.RedLed, HIGH);  // Ανάβει το κόκκινο led.
    }
   }
   enableInterrupt = true;    // Ενεργοποιούμε το interrupt.
 }

 if(timh < 40 )
 {
   digitalWrite(Control.GreenLed, HIGH); // Ανάβει το πράσινο led.
   digitalWrite(Control.RedLed, HIGH);  // Ανάβει το κόκκινο led.
 }

if(currentMillis - previousLcd >= LcdTime)
{
  previousLcd = currentMillis;
 if(Control.PinSend == 213)
 {
  lcd.setCursor(15, 0);  // Στην πρώτη σειρά στην θέση 16.
  lcd.print(timh);       // Εκτυπώνουμε την μεταβλητή.
  lcd.print(F("%"));
  lcd.print("");
  lcd.setCursor(4, 0);  // Στην πρώτη σειρά στην θέση 16.
  lcd.print(Lat,6);       // Εκτυπώνουμε την μεταβλητή.
  lcd.print("  ");
  lcd.setCursor(4,1);
  lcd.print(Lng,6);
  lcd.print("  ");
  lcd.setCursor(18,3);
  lcd.print(Control.satellite);
  lcd.print(" ");
  lcd.setCursor(6,2);
  lcd.print(Run);
  lcd.print(" ");
  lcd.setCursor(9,3);
  lcd.print(Distance);
  lcd.print("  ");
 }
 }
}

/****************** Ρίχνει το αγκίστρι ******************/

int claw()
{
  static unsigned int Lock = 0; // Χρησιμοποιούμαι αυτή την μεταβλητή ώστε να μην γράφουμε όλη την ώρα στην οθόνη το ίδιο σύμβολο.
  static unsigned int Open = 0; // Χρησιμοποιούμαι αυτή την μεταβλητή ώστε να μην γράφουμε όλη την ώρα στην οθόνη το ίδιο σύμβολο.
```

```
  Control.ReadDagana = digitalRead(5);   // Σε αυτή την function Θα ελέγχει το ποδαράκι digital 5 το button αν είναι low η high.

  if ( Control.ReadDagana == LOW ) // Αν είναι low επιστρέφτει την τιμή 0 και εκτυπώνει μία κλειδαριά στην LCD.
  {
     Control.Dagana = 0;

    if( Lock == 0)
    {

   lcd.createChar(5 , lock);
   lcd.setCursor(17, 2);
   lcd.write(byte(5));
   }
    Lock = 1;
    Open = 0;
  }
  else
  {
    Control.Dagana = 1;  // Αν είναι High τότε επιστρέφει την τιμή 1 και  εκτυπώνει μία ανοιχτεί κλειδαριά.
    if(Open == 0)
    {
     lcd.createChar(6 , LockOpen);
     lcd.setCursor(17, 2);
     lcd.write(byte(6));
    }
    Lock = 0;
    Open = 1;
  }
  return Control.Dagana;
}

/******************* Ρίχνει την μαλάγρα ******************/

int food()
{
  static unsigned int Lock = 0; // Χρησιμοποιούμαι αυτή την μεταβλητή ώστε να μην γράφουμε όλη την ώρα στην οθόνη το ίδιο σύμβολο.
  static unsigned int Open = 0; // Χρησιμοποιούμαι αυτή την μεταβλητή ώστε να μην γράφουμε όλη την ώρα στην οθόνη το ίδιο σύμβολο.
  Control.ReadFood = digitalRead(4);  // Ελέγχει το button αν ειναι high η low μέσο του ποδαράκι digital 4.

  if ( Control.ReadFood == LOW ) // Αν είναι low επιστρέφτει την τιμή 0 και εκτυπώνει μία κλειδαριά στην LCD.
  {
    Control.Food = 0;
    if( Lock == 0)
    {
      lcd.createChar(5 , lock);
      lcd.setCursor(18, 2);
      lcd.write(byte(5));
    }
```

```
   Lock = 1;
   Open = 0;
  }
  else                  // Αν είναι High τότε επιστρέφει την τιμή 1 και  εκτυπώνει μία ανοιχτεί κλειδαριά.
  {
   Control.Food = 1;
   if(Open == 0)
   {
    lcd.createChar(6 , LockOpen);
    lcd.setCursor(18, 2);
    lcd.write(byte(6));
   }
   Open = 1;
   Lock = 0;
  }
  return Control.Food;
}

/******************** Έλεγχος led ********************/

int led()
{

  Control.JoystickX  = analogRead(A2); // Η αναλογική πόρτα A2 διαβάζει τον X άξονα του joystick.
  Control.JoystickY = analogRead(A3); // Η αναλογική πόρτα A2 διαβάζει τον Y άξονα του joystick.

  if ((Control.JoystickY > 800) && (Control.JoystickX > 400 ) && ( Control.JoystickX < 750)) // Αν
πάει πάνω το joystick ανέβασε την φωτεινότητα +1 αν φτάσει το 3 τότε stop.
  {
   if (Control.Meter == 0)
   {
    Control.LedW = 0;
    Control.Meter = 1;
   }
   else if (Control.Meter  == 1)
   {
    Control.LedW = 1;
    Control.Meter = 2;
   }
   else if (Control.Meter  == 2)
   {
    Control.LedW = 2;
    Control.Meter = 3;
   }
   else if (Control.Meter  == 3)
   {
    Control.LedW = 3;
   }
  }// Τέλος if++

  else if ((Control.JoystickY < 300) && (Control.JoystickX > 400 ) && ( Control.JoystickX < 750)) //
Αν πάει κάτω το Joystick κατέβασε την φωτεινότητα -1 αν φτάσει 0 τότε stop.
  {
   if (Control.Meter == 0)
   {
```

```
    Control.LedW = 0;
   }
   else if (Control.Meter  == 1)
   {
    Control.LedW = 1;
    Control.Meter = 0;
   }
   else if (Control.Meter == 2)
   {
    Control.LedW = 2;
    Control.Meter = 1;
   }
   else if (Control.Meter  == 3)
   {
    Control.LedW = 3;
    Control.Meter = 2;
   }
 }// Τέλος if --
 lcd.setCursor(19, 1);        // Εκτύπωσε το επίπεδο φωτεινότητας στην θέσει 19 στην δεύτερη σειρά.
 lcd.print(Control.LedW);
 lcd.print("");

 return Control.LedW;
}

/******************** Στροφή πλοίου ********************/
int turn()
{
 /* Ελέγχει αν είναι το joystick δεξιά, αριστερά η στην μέση. Ανάλογα με την θέση του πάει στο
ανάλογο If.
  *  Αν ειναι δεξιά στέλνει έναν αριθμό και σχηματίζει ενα βελάκι δεξιά.
  */
  static unsigned int right =0 ;
  static unsigned int left = 0;
 Control.JoystickX  = analogRead(A2); // Η αναλογική πόρτα A2 διαβάζει τον X άξονα του joystick.
 Control.JoystickY  = analogRead(A3); // Η αναλογική πόρτα A3 διαβάζει τον Y άξονα του joystick.

 if ((Control.JoystickX <= 1023 ) && ( Control.JoystickX >= 750) && ( Control.JoystickY > 400)
&& ( Control.JoystickY < 800))
 {
  Control.Degrees = 0;
  if(right == 0)
  {
   lcd.createChar(2 , ArrowRight);
   lcd.setCursor(19, 2);
   lcd.write(byte(2));
  }
 right = 1;
 left = 0;
 }
 else if ((Control.JoystickX > 400 ) && ( Control.JoystickX < 750) && ( Control.JoystickY > 400)
&& ( Control.JoystickY < 800))
 {
  Control.Degrees = 1;
  lcd.setCursor(19, 2);
```

```
    lcd.write(' ');
    lcd.setCursor(16, 2);
    lcd.write(' ');
    right  = 0;
    left = 0;
  }
  else if ((Control.JoystickX <= 400 ) && ( Control.JoystickX >= 0) && ( Control.JoystickY > 400) && ( Control.JoystickY < 800) )
  {
    Control.Degrees = 2;
    if (left == 0)
    {
      lcd.createChar(3 , ArrowLeft);
      lcd.setCursor(16, 2);
      lcd.write(byte(3));
    }
    left = 1;
    right = 0;
  }
  return Control.Degrees;
}
/****************** Έλεγχος στροφή κινητήρα ******************/

int DcMotor()
{

  Control.Pot = analogRead(A0);                    // Η αναλογική είσοδος A0 διαβάζει την τάση που θα δέχεται απο το ποτενσιόμετρο.
  Control.motor = map(Control.Pot , 0 , 1023 , 0 , 180); // Μετατρέπουμε τις τιμές 0-1023 σε 0-180 μοίρες.

  return  Control.motor;                           // Επιστρέφει την τιμή της μεγαλητής.
}

/****************** Σχηματίζει την μπαταρία ανάλογα με την τάση  ******************/
void Battery()
{

  if (timh <= 100 && timh > 90) // Βλέπει τι αριθμός θα φτάσει απο το σκάφος και το τοποθετεί στο ανάλογο If και σχεδιάζει την μπαταρία με την κατάλληλη τάση.
  {
    byte batlevel[8] =
    {
      B01110,
      B11111,
      B10101,
      B10001,
      B11011,
      B11011,
      B11111,
      B11111,
    };
    lcd.createChar(1 , batlevel);
    lcd.setCursor(19, 0);
    lcd.write(byte(1));
```

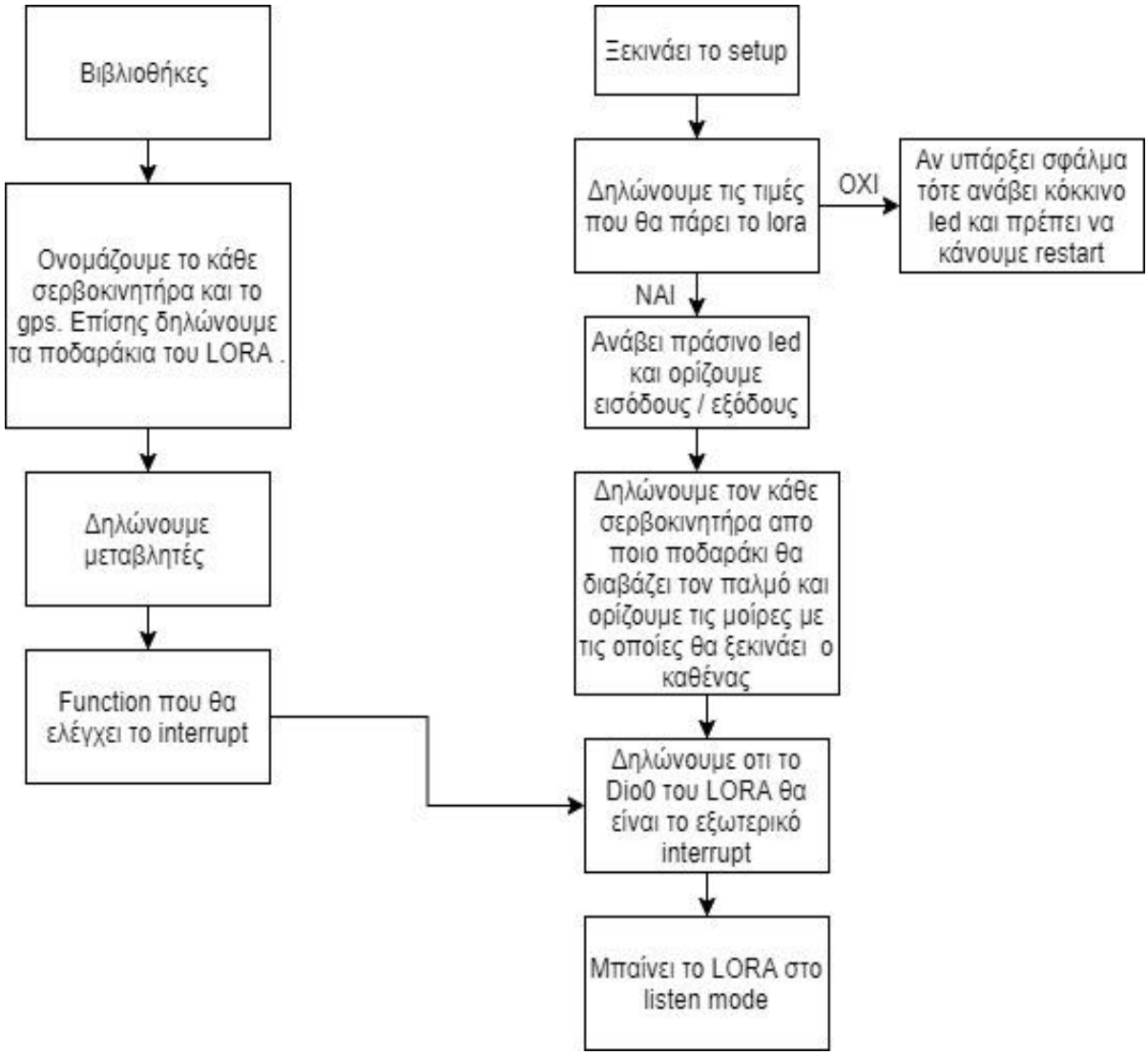
```
}
if (timh <= 90 && timh > 80)
{
  byte batlevel[8] =
  {
    B01110,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
  };
  lcd.createChar(1 , batlevel);
  lcd.setCursor(19, 0);
  lcd.write(byte(1));
}
if (timh <= 80 && timh > 70)
{
  byte batlevel[8] =
  {
    B01110,
    B10001,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
  };
  lcd.createChar(1 , batlevel);
  lcd.setCursor(19, 0);
  lcd.write(byte(1));
}
if (timh <= 70 && timh > 60)
{
  byte batlevel[8] = {
    B01110,
    B10001,
    B10001,
    B11111,
    B11111,
    B11111,
    B11111,
    B11111,
  };
  lcd.createChar(1 , batlevel);
  lcd.setCursor(19, 0);
  lcd.write(byte(1));
}
if (timh <= 60 && timh > 40)
{
  byte batlevel[8] =
  {
```

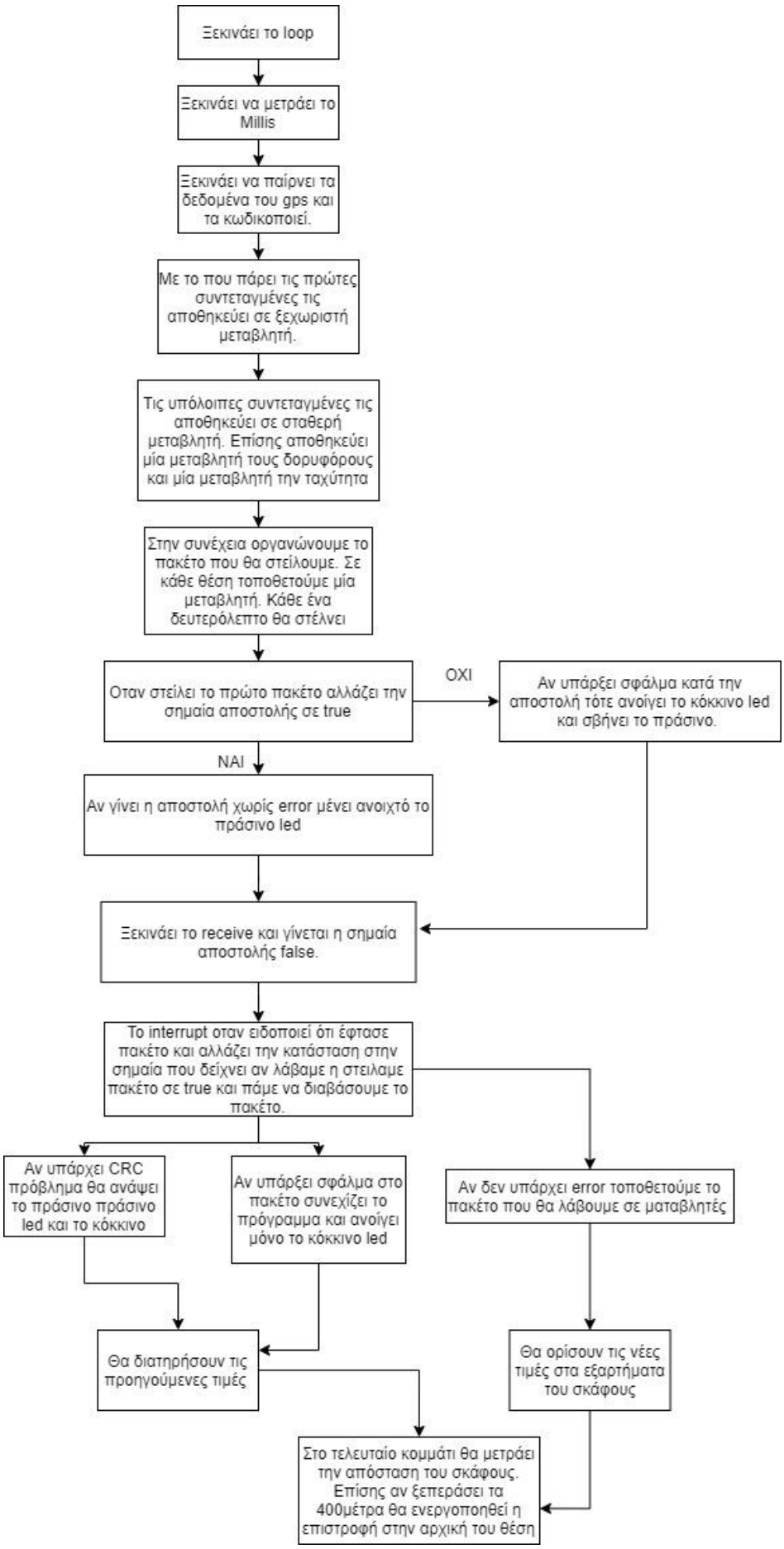
```
    B01110,
    B10001,
    B10001,
    B10001,
    B11111,
    B11111,
    B11111,
    B11111,
   };
   lcd.createChar(1 , batlevel);
   lcd.setCursor(19, 0);
   lcd.write(byte(1));
}
if (timh <= 40 && timh > 30)
{
   byte batlevel[8] =
   {
    B01110,
    B10001,
    B10001,
    B10001,
    B10001,
    B11111,
    B11111,
    B11111,
   };
   lcd.createChar(1 , batlevel);
   lcd.setCursor(19, 0);
   lcd.write(byte(1));
}
if (timh <= 30 && timh > 10)
{
   byte batlevel[8] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B10001,
    B10001,
    B11111,
    B11111,
   };
   lcd.createChar(1 , batlevel);
   lcd.setCursor(19, 0);
   lcd.write(byte(1));
}
if (timh <= 10)
{
   byte batlevel[8] = {
    B01110,
    B10001,
    B10001,
    B10001,
    B10001,
    B10001,
```

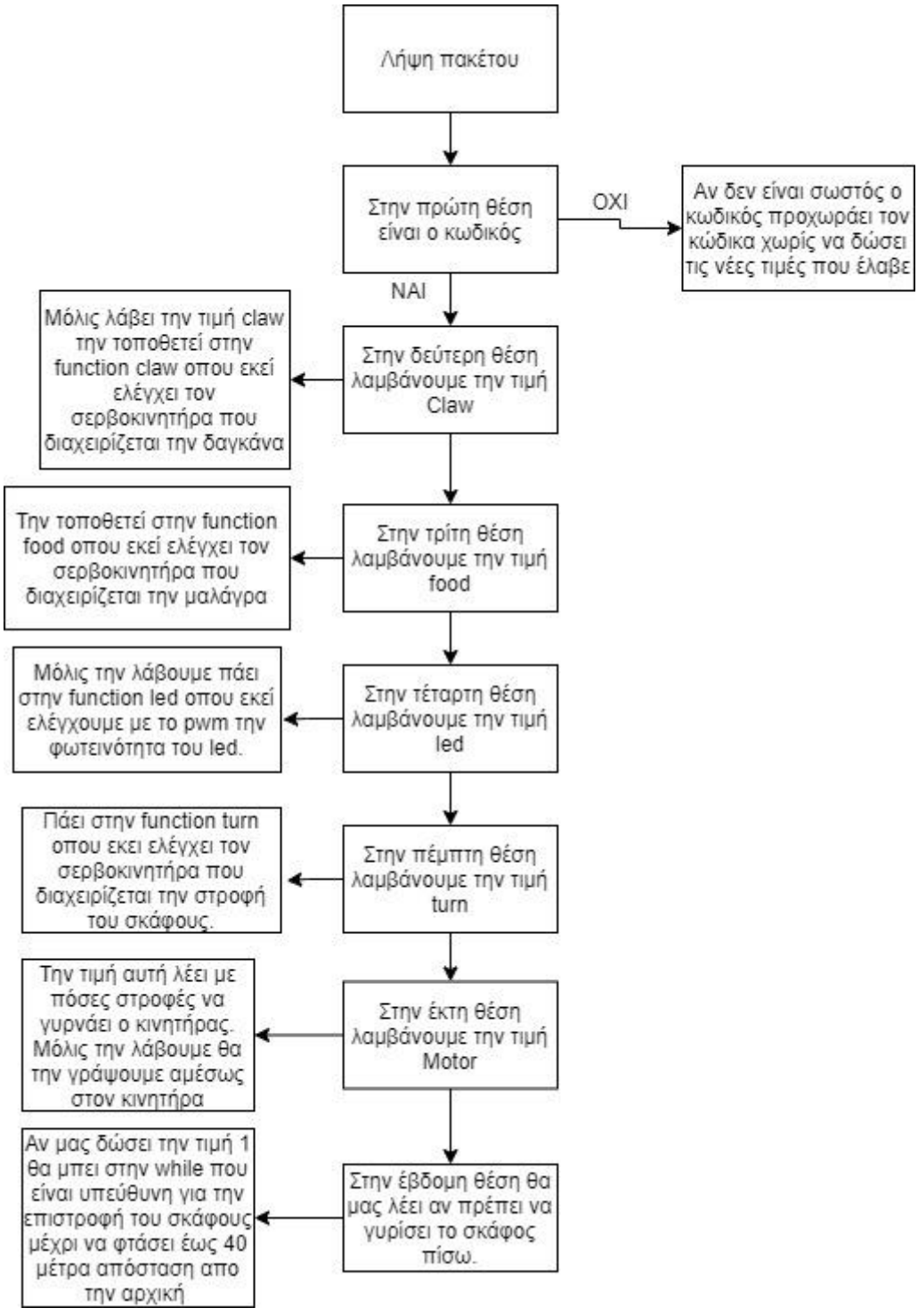
```
      B10001,
      B11111,
    };
    lcd.createChar(1 , batlevel);
    lcd.setCursor(19, 0);
    lcd.write(byte(1));
  }
}
int JoyButton()
{
  static int JoyButton = 0;
  static int Return = 0;
  Control.Button = digitalRead(A1);

  if ( Control.Button == 0)
  {
   if(JoyButton == 0)
   {
    JoyButton = 1;
    if(Return == 0)
    {
      lcd.createChar(9 , Ret);
      lcd.setCursor(15, 2);
      lcd.write(byte(9));
    }
    Return = 1;
  }
   else if(JoyButton == 1)
    {
     JoyButton = 0;
     lcd.setCursor(15, 2);
     lcd.write(' ');
     Return = 0;
    }
  }
return JoyButton;
}
```

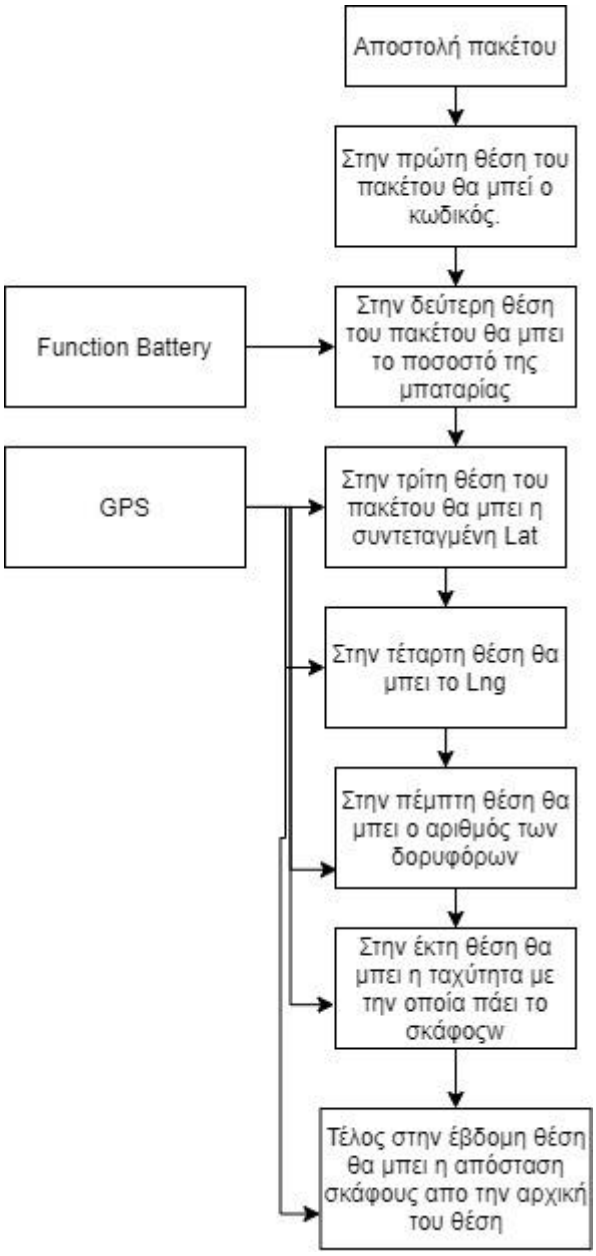
ΠΑΡΑΡΤΗΜΑ Β

Σε αυτό το παράρτημα θα παρουσιάσουμε το διάγραμμα του κώδικα που χρησιμοποιήθηκε για την υλοποίηση του σκάφους.




Ξεκινάει το loop
Ξεκινάει να μετράει το Millis
Ξεκινάει να παίρνει τα δεδομένα του gps και τα κωδικοποιεί.
Με το που πάρει τις πρώτες συντεταγμένες τις αποθηκεύει σε ξεχωριστή μεταβλητή.
Τις υπόλοιπες συντεταγμένες τις αποθηκεύει σε σταθερή μεταβλητή. Επίσης αποθηκεύει μία μεταβλητή τους δορυφόρους και μία μεταβλητή την ταχύτητα
Στην συνέχεια οργανώνουμε το πακέτο που θα στείλουμε. Σε κάθε θέση τοποθετούμε μία μεταβλητή. Κάθε ένα δευτερόλεπτο θα στέλνει
Οταν στείλει το πρώτο πακέτο αλλάζει την σημαία αποστολής σε true
ΟΧΙ
Αν υπάρξει σφάλμα κατά την αποστολή τότε ανοίγει το κόκκινο led και σβήνει το πράσινο.
ΝΑΙ
Αν γίνει η αποστολή χωρίς error μένει ανοιχτό το πράσινο led
Ξεκινάει το receive και γίνεται η σημαία αποστολής false.
Το interrupt οταν ειδοποιεί ότι έφτασε πακέτο και αλλάζει την κατάσταση στην σημαία που δείχνει αν λάβαμε η στειλαμε πακέτο σε true και πάμε να διαβάσουμε το πακέτο.
Αν υπάρχει CRC πρόβλημα θα ανάψει το πράσινο πράσινο led και το κόκκινο
Αν υπάρξει σφάλμα στο πακέτο συνεχίζει το πρόγραμμα και ανοίγει μόνο το κόκκινο led
Αν δεν υπάρχει error τοποθετούμε το πακέτο που θα λάβουμε σε ματαβλητές
Θα διατηρήσουν τις προηγούμενες τιμές
Θα ορίσουν τις νέες τιμές στα εξαρτήματα του σκάφους
Στο τελευταίο κομμάτι θα μετράει την απόσταση του σκάφους. Επίσης αν ξεπεράσει τα 400μέτρα θα ενεργοποιηθεί η επιστροφή στην αρχική του θέση


Λήψη πακέτου
Στην πρώτη θέση είναι ο κωδικός
ΟΧΙ
Αν δεν είναι σωστός ο κωδικός προχωράει τον κώδικα χωρίς να δώσει τις νέες τιμές που έλαβε
ΝΑΙ
Στην δεύτερη θέση λαμβάνουμε την τιμή Claw
Μόλις λάβει την τιμή claw την τοποθετεί στην function claw οπου εκεί ελέγχει τον σερβοκινητήρα που διαχειρίζεται την δαγκάνα
Στην τρίτη θέση λαμβάνουμε την τιμή food
Την τοποθετεί στην function food οπου εκεί ελέγχει τον σερβοκινητήρα που διαχειρίζεται την μαλάγρα
Στην τέταρτη θέση λαμβάνουμε την τιμή led
Μόλις την λάβουμε πάει στην function led οπου εκεί ελέγχουμε με το pwm την φωτεινότητα του led.
Στην πέμπτη θέση λαμβάνουμε την τιμή turn
Πάει στην function turn οπου εκει ελέγχει τον σερβοκινητήρα που διαχειρίζεται την στροφή του σκάφους.
Στην έκτη θέση λαμβάνουμε την τιμή Motor
Την τιμή αυτή λέει με πόσες στροφές να γυρνάει ο κινητήρας. Μόλις την λάβουμε θα την γράψουμε αμέσως στον κινητήρα
Στην έβδομη θέση θα μας λέει αν πρέπει να γυρίσει το σκάφος πίσω.
Αν μας δώσει την τιμή 1 θα μπει στην while που είναι υπεύθυνη για την επιστροφή του σκάφους μέχρι να φτάσει έως 40 μέτρα απόσταση απο την αρχική


Αποστολή πακέτου
Στην πρώτη θέση του πακέτου θα μπεί ο κωδικός.
Function Battery
Στην δεύτερη θέση του πακέτου θα μπει το ποσοστό της μπαταρίας
GPS
Στην τρίτη θέση του πακέτου θα μπει η συντεταγμένη Lat
Στην τέταρτη θέση θα μπει το Lng
Στην πέμπτη θέση θα μπει ο αριθμός των δορυφόρων
Στην έκτη θέση θα μπει η ταχύτητα με την οποία πάει το σκάφοςw
Τέλος στην έβδομη θέση θα μπει η απόσταση σκάφους απο την αρχική του θέση

```
/******************** Βιβλιοθήκες ********************/

#include <RadioLib.h> // Βιβλιοθήκη Lora.
#include <Servo.h>    // Βιβλιοθήκη σερβοκινητήρα.
#include <TinyGPS++.h> // Βιβιοθήκη GPS.

/******************** Ορισμός Servo-Lora ********************/

Servo claw; // Ονομάζουμε τον ένα σερβοκινητήρα claw.
Servo food; // Ονομάζουμε τον άλλον σερβοκινητήρα food.
Servo turn; // Ονομάζουμε τον άλλον σερβοκινητήρα turn.
Servo motor;// Ονομάζουμε τον άλλον σερβοκινητήρα motor.

SX1278 radio = new Module(10, 2, 9); //Δηλώνω στο digital 10 του atmel συνδέεται το NSS του LORA, digital 2 το DIO0 (Interrupt), digital 9 το RESET.
TinyGPSPlus gps;  // gps object
/******************** Flags/Interrupts ********************/

bool transmitFlag = false;              // Flag που μας δείχνει αν είναι αποστολή η λήψη.
volatile bool enableInterrupt = true;  // Αυτή η μεταβλητή ενεργοποιεί και απενεργοποιεί την διακοπή.
volatile bool operationDone = false;   // Αυτή η μεταβλητή μας δείχνει πότε στάλθηκε/λήφθηκε ένα πακέτο.

boolean validGPS = false; // Μεταβλητή για να δείχνει αν είναι έγκυρα τα gps data.
static unsigned int flag = 0; // Σημαία που δείχνει αν πρέπει να αποθηκεύσει τις πρώτες συντεταγμενές η τις συνεχείς.

/******************** Μεταβλητές ********************/

typedef struct remote // Για να γλιτώσουμε τα περιττά bit χρησιμοποιούμε το struct.
{
  uint8_t Motor;              // Τιμές 0-180.
  uint8_t Pin;               // Κωδικός 231.
  uint8_t PinSend = 213;      // Κωδικός 213.
  uint8_t Led = 5;            // Τιμές 0-255.
  uint8_t satellite;
  uint8_t Turn          : 3; // Τιμές 0-5.
  uint8_t LedW          : 2; // Τιμές 0-3.
  uint8_t JoyButton     : 1; // Τιμές 0 ή 1.
  uint8_t Food          : 1; // Τιμές 0 ή 1.
  uint8_t Claw          : 1; // Τιμές 0 ή 1.
  uint8_t RedLed = A2;   : 1; // Τιμές 0 ή 1.
  uint8_t GreenLed = A0; : 1; // Τιμές 0 ή 1.
};

struct remote Control;

unsigned long SendTime = 1000;        // Η μεταβλητή αυτή θα ελέγχει κάθε πότε θα στέλνει δεδομένα.
static unsigned long previousSend = 0; // Κρατάει την τιμή της προηγούμενης αποστολής.
unsigned long GpsTime = 4000;         // Κάθε πότε θα διαβάζει δεδομένα το gps.
static unsigned long previousGps = 0;  // Κρατάει την τιμή απο την προηγούμενη φορά που διάβασε δεδομένα το gps.
```

```
float Lat;
float Lng;
float FirstLat;
float FirstLng;
float Run;
float distanceToDestination;
double courseToDestination;
int courseChangeNeeded;

int TraState = ERR_NONE;   // Αυτή η μεταβλητή ελέγχει την επικοινωνία της αποστολής.
int ReceState = ERR_NONE;  // Αυτή η μεταβλητή ελέγχει την επικοινωνία δέκτη.

uint16_t recePacket[7]; // Αυτός ο πίνακας θα αποθηκεύει τις μεταβλητές που θα λαμβάνει.
float sendPacket[7];    // Αυτός ο πίνακας θα αποθηκεύει τις μεταβλητές που θα στέλνει.

/******************** Έλεγχος Interrupt ********************/

void setFlag(void)
{
 if (!enableInterrupt)
   // Έλεγχος αν το interrupt είναι ενεργοποιημένο.
 {
   return;
 }
 operationDone = true; // Στείλαμε η λάβαμε πακέτο, κάνε true το flag.
}

/******************** Ξεκινάει το πρόγραμμα ********************/

void setup()
{
 Serial.begin(9600);

 pinMode(Control.RedLed, OUTPUT);   // Δηλώνουμε το pin που ειναι συνδεδεμένο το κόκκινο led
έξοδο.
 pinMode(Control.GreenLed, OUTPUT); // Δηλώνουμε το pin που ειναι συνδεδεμένο το πράσινο led
έξοδο.

 radio.begin(433.5, 500.0, 8, 7, 0x18, 17, 20, 0);

 while (radio.begin() != ERR_NONE)
 {
   digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
   digitalWrite(Control.RedLed, HIGH);   // Ανάβει το κόκκινο led.
 }
 if (radio.setCurrentLimit(100) == ERR_INVALID_CURRENT_LIMIT)
 {
   digitalWrite(Control.RedLed, HIGH); // Ανάβει το κόκκινο led.
   digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
 }
 if (radio.setCRC(true) == ERR_INVALID_CRC_CONFIGURATION)  // Ενεργοποιούμε το CRC.
 {
   digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
   digitalWrite(Control.RedLed, HIGH);   // Ανοίγει το κόκκινο led.
 }
```

```
  digitalWrite(Control.GreenLed, HIGH); // Ανάβει το πράσινο led.
  digitalWrite(Control.RedLed, LOW);    // Κλείνει το κόκκινο led.

  claw.attach(A4);
  claw.write(180);
  food.attach(7);
  food.write(0);
  turn.attach(A5);  // Δηλώνω πως ο σερβοκινητήρας που θα γυρνάει το πλοίο θα παιρνει απο το πόδι
A5  το σήμα PWM.
  turn.write(40);
  motor.attach(6, 1000, 2000); // Δηλώνω στο πόδι digital 6 Θα παίρνει ο κινητήρας το σήμα και πως ο
ελάχιστος παλμός που θα δέχεται είναι 1000 και ο μέγιστος 2000.

  radio.setDio0Action(setFlag); // Δηλώνω το ποδαράκι του lora Dio0 να δηιαβάζει την fuction
setFlag.

  radio.startReceive();
}

void loop()
{
  unsigned long currentMillis = millis();

  while (Serial.available()) // Ακούει τα data του gps.
  {
    validGPS = gps.encode(Serial.read()); //Ανάγνωση και κωδικοποίηση δεδομένων gps.
  }

  if (validGPS && gps.location.isUpdated())
  {
    if ( flag == 0)
    {
      FirstLat = gps.location.lat(); // Κρατάνε τις πρώτες συντεταγμένες
      FirstLng = gps.location.lng(); // Κρατάνε τις πρώτες συντεταγμένες
      flag = 1;
    }
    else if (flag == 1)
    {
      Lat = gps.location.lat(); // Κρατάει την τελευταία συντεταγμένη.
      Lng = gps.location.lng(); // Κρατάει την τελευταία συντεταγμένη.
      Control.satellite = gps.satellites.value(); // Κρατάει τον αριθμό των δορυφόρων.
      Run = gps.speed.kmph(); // Αυτή η μεταβλητή θα δείχνει την ταχύτητα του σκάφους.
    }
  }

  if (currentMillis - previousGps >= GpsTime)
  {
    previousGps = currentMillis;

    distanceToDestination = TinyGPSPlus::distanceBetween(Lat, Lng, FirstLat, FirstLng); //
Απόσταση
    courseToDestination = TinyGPSPlus::courseTo(Lat, Lng, FirstLat, FirstLng);
    const char *directionToDestination = TinyGPSPlus::cardinal(courseToDestination);
    courseChangeNeeded = (int)(360 + courseToDestination - gps.course.deg()) % 360;
  }
```

```
if (currentMillis - previousSend >= SendTime)
{
  previousSend = currentMillis;
  sendPacket[0] = Control.PinSend;
  sendPacket[1] = Battery();
  sendPacket[2] = Lat;
  sendPacket[3] = Lng;
  sendPacket[4] = Control.satellite;
  sendPacket[5] = Run;
  sendPacket[6] = distanceToDestination;

  TraState = radio.startTransmit((byte*)&sendPacket, 7 * sizeof(float));
  transmitFlag = true;
}

if (operationDone)
{
  enableInterrupt = false;
  operationDone = false;

  if (transmitFlag)
  {
    if (TraState == ERR_NONE)
    {
      digitalWrite(Control.GreenLed, HIGH);  // Ανάβει το πράσινο led.
      digitalWrite(Control.RedLed, LOW);   // Κλείνει το κόκκινο led.
    }
    else
    {
      digitalWrite(Control.GreenLed, LOW);  // Ανάβει το πράσινο led.
      digitalWrite(Control.RedLed, LOW);   // Κλείνει το κόκκινο led.
    }

    radio.startReceive();
    transmitFlag = false;
  }

  else
  {
    ReceState = radio.readData((uint8_t*)recePacket, sizeof(recePacket));

    if (ReceState == ERR_NONE)
    {
      Control.Pin   = recePacket[0];
      Control.Claw  = recePacket[1];
      Control.Food  = recePacket[2];
      Control.LedW  = recePacket[3];
      Control.Turn  = recePacket[4];
      Control.Motor = recePacket[5];
      Control.JoyButton = recePacket[6];

      digitalWrite(Control.GreenLed, HIGH); // Ανάβει το πράσινο led.
      digitalWrite(Control.RedLed, LOW);   // Κλείνει το κόκκινο led.
    }
```

```
    else if (ReceState == ERR_CRC_MISMATCH)
    {
      digitalWrite(Control.GreenLed, HIGH); // Ανάβει το πράσινο led.
      digitalWrite(Control.RedLed, HIGH);   // Ανάβει το κόκκινο led.
    }
    else
    {
      digitalWrite(Control.GreenLed, LOW);  // Κλείνει το πράσινο led.
      digitalWrite(Control.RedLed, HIGH);   // Ανάβει το κόκκινο led.
    }
   }
   enableInterrupt = true;
 }
 if (Control.Pin == 231)
 {
   CLAW();
   FOOD();
   LEDW();
   TURN();
   motor.write(Control.Motor);
 }

 while((distanceToDestination >= 400.0) || (Control.JoyButton == 1))
 {
  if(distanceToDestination >= 40)
  {
     motor.write(Control.Motor);
   if (courseChangeNeeded >= 345 || courseChangeNeeded < 15)      // Πάνε ευθεία
     turn.write(40);
    else if (courseChangeNeeded >= 315 && courseChangeNeeded < 345) // Λίγο αριστερά
     turn.write(25);
    else if (courseChangeNeeded >= 15 && courseChangeNeeded < 45)   // Λίγο δεξιά
     turn.write(55);
    else if (courseChangeNeeded >= 255 && courseChangeNeeded < 315)  // Αριστερά
      turn.write(10);
    else if (courseChangeNeeded >= 45 && courseChangeNeeded < 105)  // Δεξιά
      turn.write(70);
    else
     turn.write(70);
     delay(1000);
  }
    else
    {
     Control.JoyButton == 0;
     break;
    }
 }

}


/******************** Έλεγχος δαγκάνας ********************/

void CLAW()
{
```

```
  if (Control.Claw == 0)
  {
    claw.write(180);
  }
  else if (Control.Claw == 1)
  {
    claw.write(0);
  }
}

/******************** Έλεγχος μαλάγρα ********************/

void FOOD()
{
  if (Control.Food == 0)
  {
    food.write(0);
  }
  else if (Control.Food == 1)
  {
    food.write(180);
  }
}

/******************** Έλεγχος led ********************/

void LEDW()
{
  if (Control.LedW == 0)
  {
    analogWrite(Control.Led, 0);
  }
  else if (Control.LedW == 1)
  {
    analogWrite(Control.Led, 85);
  }
  else if (Control.LedW == 2)
  {
    analogWrite(Control.Led, 170);
  }
  else if (Control.LedW == 3)
  {
    analogWrite(Control.Led, 255);
  }
}

/******************** Στροφή πλοίου ********************/

void TURN()
{
  if (Control.Turn == 0)
  {
    turn.write(70);
  }
  else if (Control.Turn == 1)
```

```
  {
    turn.write(40);
  }
  else if (Control.Turn == 2)
  {
    turn.write(10);
  }
}

/******************** Τάση μπαταρίας  ********************/

int Battery()
{
  static int BatteryVolt = 0;
  int readAnalogVolt = analogRead(A1);

  float Value = ((readAnalogVolt * 5) / 1024) + 0.7 ;
  float Volt = Value / 0.394;

  if (Volt >= 12.5)
  {
    BatteryVolt = 100;
  }

  else if ((Volt < 12.5) && (Volt >= 12.2))
  {
    BatteryVolt = 90;
  }
  else if ((Volt < 12.2) && (Volt >= 11.9))
  {
    BatteryVolt = 80;
  }

  else if ((Volt < 11.9) && (Volt >= 11.6))
  {
    BatteryVolt = 70;
  }

  else if ((Volt < 11.6) && (Volt >= 11.3))
  {
    BatteryVolt = 60;
  }
  else if ((Volt < 11.3) && (Volt >= 11))
  {
    BatteryVolt = 50;
  }

  else if ((Volt < 11) && (Volt >= 10.8))
  {
    BatteryVolt = 40;
  }

  else if  ((Volt < 10.8) && (Volt >= 10.6))
  {
    BatteryVolt = 30;
```

```
  }

  else if (Volt < 10.6 && Volt >= 10.5)
  {
    BatteryVolt = 20;
  }

  else if (Volt < 10.5 && Volt >= 10.3)
  {
    BatteryVolt = 10;
  }
  else
  {
    BatteryVolt = 0;
  }

  return BatteryVolt;
}
```