



ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΘΕΜΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

**ΚΑΤΕΥΘΥΝΟΜΕΝΟ ΣΚΑΦΟΣ ΨΑΡΕΜΑΤΟΣ ΜΕ ΧΡΗΣΗ ΚΙΝΗΤΟΥ
ANDROID, ΕΝΤΟΠΙΣΜΟΥ ΘΕΣΗΣ ΚΑΙ ΑΥΤΟΜΑΤΗΣ ΔΙΟΡΘΩΣΗΣ
ΠΟΡΕΙΑΣ**

ΟΝΟΜΑ ΦΟΙΤΗΤΗ:

ΛΑΒΔΑΣ ΠΑΝΑΓΙΩΤΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΗΣ

ΑΙΓΑΛΕΩ, ΙΟΥΛΙΟΣ 2018

Περιεχόμενα

1. Συνοπτικός τρόπος λειτουργίας	4
2. Πρωτόκολλα και τεχνικές	5
2.1 Σειριακή επικοινωνία	5
2.2 Πρωτόκολλο I ² C	6
Χαρακτηριστικά του I ² C Bus	6
2.3 Διαμόρφωση εύρους παλμών (PWM)	8
2.4 Μετατροπείς αναλογικού σήματος σε ψηφιακό (ADC)	9
2.5 Πρωτόκολλο επικοινωνίας 802.15.1(Bluetooth)	10
2.6 Xbee 802.15.4 (Series 2)	10
2.7 Ελεγκτής PID	11
3. ΣΚΑΦΟΣ ΚΑΙ ΒΑΣΗ	13
3.1 Η βάση του σκάφους.....	13
3.2 Το σκάφος	15
4. Κύκλωμα και συναρτήσεις	18
4.1 Buzzer	18
4.2 Έλεγχος επίπεδου μπαταρίας.....	19
4.3 Έλεγχος θερμοκρασίας Speed Controller (ESC)	20
4.4 Αντλία νερού	21
4.5 Reset	22
4.6 LED Light	22
4.7 Τροφοδοτικά	23
4.8 Ταχύτητα κινητήρα και διεύθυνση πηδαλίου.....	24
4.8.1 Κινητήρας	24
4.8.2 Πηδάλιο	25
4.9 Κύκλωμα ATMEGA 2560.....	26
5. Αυτόματος πιλότος	27
6. Κύκλωμα στη βάση και τρόπος λειτουργίας.....	35
7. Εφαρμογή android	37
7.1 Γενικά.....	37
7.2 Το layout (.xml) της εφαρμογής	37
7.3 ο κώδικας της εφαρμογής, xml	39
7.4 ο κώδικας της εφαρμογής, Java	44
8. ΒΙΒΛΙΟΓΡΑΦΙΑ	56

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/α Ναΐβδα Νικολαΐδου Χαραλαμπίδου φοιτητής του Τμήματος Αυτοματιδολογίας του Πανεπιστημίου Δυτικής Αττικής, πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε, ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα, σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασή της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση Π.Ε με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού δμήνου από την ημερομηνία ανάθεσής της.

Ο Δηλών



Ημερομηνία

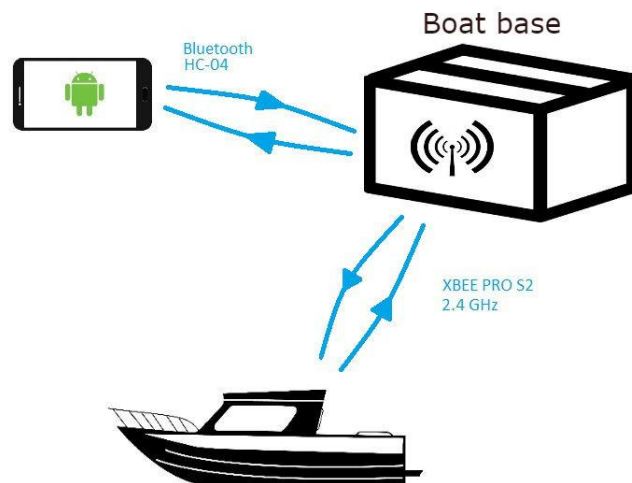
3/7/2018

1. Συνοπτικός τρόπος λειτουργίας

Το σκάφος είναι κατασκευασμένο για ψάρεμα από ακτή ή βάρκα, το λεγόμενο παραγάδι.

Ο λόγος κατασκευής του κατευθυνόμενου αυτού σκάφους είναι για εξοικονόμηση χρημάτων και χρόνου γιατί σε διαφορετική περίπτωση η αντίστοιχη διαδικασία γίνεται με βάρκα.

Το σκάφος αυτό κατευθύνεται με εφαρμογή κινητού android. Πιο συγκεκριμένα, η εφαρμογή στέλνει δεδομένα μέσω Bluetooth στη βάση του σκάφους και το κύκλωμα της βάσης στέλνει τα δεδομένα μέσω RF στα 2.4GHz στο σκάφος και αντίστροφα.



Από την εφαρμογή μπορούμε να διαχειριστούμε την διεύθυνση, την ταχύτητα τα φώτα, καθώς και να ελέγχουμε τη στάθμη της μπαταρίας και την θερμοκρασία του speed controller του σκάφους. Επίσης, μπορούμε να το κλειδώσουμε σε μια πορεία και αυτό να διορθώνει αυτόματα την πορεία του από τυχόν εξωτερικούς παράγοντες (π.χ. αέρας, κύμα).

Διαθέτει επίσης σύστημα G.P.S το οποίο μας δίνει την δυνατότητα να εντοπίζουμε τη θέση του σκάφους.

2. Πρωτόκολλα και τεχνικές

2.1 Σειριακή επικοινωνία

Ο μικροελεγκτής μπορεί να ανταλλάσει δεδομένα με τον έξω κόσμο σειριακά. Οι ακροδέκτες σειριακής λήψης και αποστολής συμβολίζονται με Rx και Tx αντίστοιχα. Όταν ο μικροελεγκτής θέλει να στείλει κάτι προς τη εξωτερική συσκευή εμφανίζει κάθε ένα από τα bits στον ακροδέκτη αποστολής Tx επί το χρονικό διάστημα T_b , το οποίο ονομάζεται διάρκεια bit. Το ίδιο γίνεται και όταν η εξωτερική συσκευή στέλνει κάτι στο μικροελεγκτή. Τα bits διαβάζονται μέσω του ακροδέκτη Rx.

Στη μονάδα του χρόνου (sec) μεταδίδονται $r_b = 1/T_b$ bits. Το μέγεθος αυτό ονομάζεται ρυθμός μετάδοσης bits (bit rate) και μετριέται σε bit per second (bps), kilobit per second (kbps), megabits per second (mbps) κ.ο.κ., όταν η διάρκεια bit T_b μετριέται σε sec, msec, μ sec κ.ο.κ. Άλλη μια ονομασία γι αυτό το μέγεθος είναι το baud rate, δηλαδή ο ρυθμός μετάδοσης των μεταβολών της στάθμης του δυναμικού (τάσης) της γραμμής επικοινωνίας. Οι χρησιμοποιημένες τιμές του baud rate είναι συμφωνημένες διεθνώς και είναι 300, 600, 1.200, 2.400, 4.800, 9.600, 14.400, 19.200, 28.800, 38.400, 57.600, 115.200. Στη συγκεκριμένη εφαρμογή επιλέχθηκε baud rate 115.200.

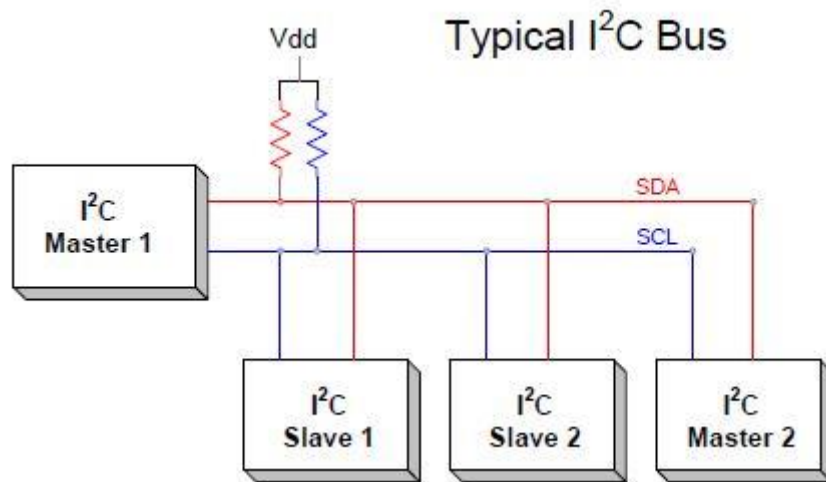
Η αποστολή και λήψη ενός δεδομένου γίνεται με πρώτο το λιγότερο σημαντικό bit (LSB) και τελευταίο το περισσότερο σημαντικό bit (MSB). Της αποστολής του LSB προηγείται η αποστολή ενός bit αρχής (start bit) που έχει διάρκεια T_b και έχει πάντα τιμή 0 (LOW). Επίσης, μετά την αποστολή (MSB) ακολουθεί η αποστολή ενός bit τέλους που έχει πάντα την τιμή 1 (HIGH). Ο αποστολέας μέχρι να στείλει το επόμενο σειριακό δεδομένο κρατά τη γραμμή επικοινωνίας σε υψηλή τιμή.

2.2 Πρωτόκολλο I²C

Το I²C Bus αποτελεί ένα πρωτόκολλο που χρησιμοποιείται για την διασύνδεση μεταξύ διαφορετικών ολοκληρωμένων, όπως των μικροελεγκτών και των υπόλοιπων ολοκληρωμένων περιφερειακών όπως EEPROMs, A/D μετατροπείς, LCD drivers, αισθητήρες αλλά και με άλλους μικροελεγκτές. Είναι ένα πρωτόκολλο που μεταδίδει δεδομένα σειριακά. Με το I²C αποφεύγεται η χρησιμοποίηση ενός παράλληλου διαύλου δεδομένων που εισάγει μεγάλη πολυπλοκότητα στη σχεδίαση αλλά και μεγαλύτερο κόστος. Βρίσκει πολλές εφαρμογές στα σύγχρονα ηλεκτρονικά συστήματα όπως σε συσκευές εικόνας και ήχου, σε τηλεφωνικές συσκευές, modems, dip switches, embedded microprocessor boards αλλά και στην επικοινωνία αισθητήρων θερμοκρασίας με τις οθόνες όπου παρουσιάζονται τα αποτελέσματα των μετρήσεων. Οι ταχύτητες τις οποίες επιτυγχάνει μπορούν να φτάσουν μέχρι και 3.4 Mbps, ταχύτητες ικανές για την ανταλλαγή δεδομένων ανάμεσα στους κόμβους και το μήκος του καλωδίου μερικές δεκάδες μέτρα, μήκος αρκετά ικανό αν θεωρήσουμε ότι οι αποστάσεις συνήθως δεν ξεπερνάνε τα μερικά μέτρα.

Χαρακτηριστικά του I²C Bus

Το I²C υλοποιείται με την χρήση δύο καλωδίων διπλής κατεύθυνσης. Τα δύο καλώδια αυτά είναι το SDA, Serial Data, που χρησιμοποιείται για την μεταφορά δεδομένων και το SCL, Serial Clock, για το ρολόι και είναι συνδεδεμένα πάντα σε θετική τροφοδοσία μέσω pull-up αντιστάσεων. Κάθε συσκευή πάνω στο bus έχει τη δική της μοναδική διεύθυνση, καθώς επίσης και το δικαίωμα αποστολής και λήψης δεδομένων από το δίαυλο. Το μήκος του καλωδίου(bus) μπορεί να φτάσει τα 3 με 4 μέτρα αλλά μπορεί και να αυξηθεί με τους λεγόμενους bus extenders έως και 100m. Επιπλέον κάθε συσκευή πάνω στο bus μπορεί να λειτουργεί είτε ως Master, οπότε αποφασίζει για τις λειτουργίες που επιτελούνται πάνω στο bus, είτε ως slave, οπότε ανταποκρίνεται στις αιτήσεις του Master.



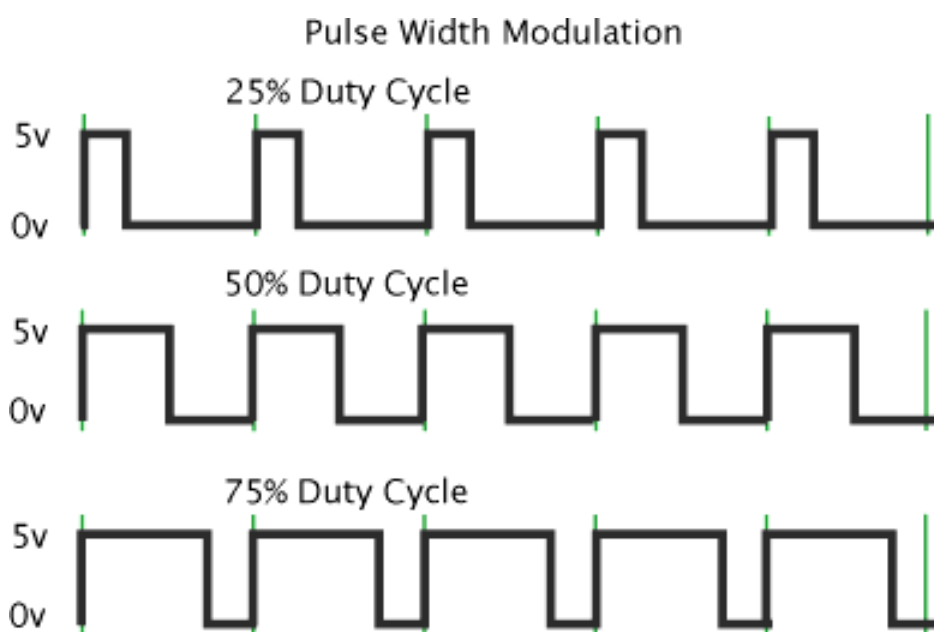
Εξαιτίας της αμφίδρομης φύσης των καλωδίων, SDA και SCL, τα στάδια εξόδου των ολοκληρωμένων για να μπορούν να αντεπεξέρχονται σε τυχόν συγκρούσεις (collisions) χωρίς να καταστρέφονται, έχουν στάδια εξόδου ανοικτού συλλέκτη (open collector/drain) έτσι ώστε να διεκπεραιώνουν την καλωδιωμένη- AND(wired-AND) λογική του καλωδίου. Το στάδιο εξόδου του ανοικτού συλλέκτη, όπως φαίνεται στο επόμενο σχήμα, δεν έχει pull-up αντίσταση συνεπώς δεν μπορεί να οδηγήσει την γραμμή σε υψηλή στάθμη. Οι εξωτερικές pull-up αντιστάσεις οδηγούν τις γραμμές SDA, SCL οποτεδήποτε οι γραμμές αυτές δεν οδηγούνται από το τρανζίστορ εξόδου σε υψηλή στάθμη.

Από την παραπάνω τοπολογία συμπεραίνουμε ότι όταν το bus βρίσκεται σε αδρανή κατάσταση οι δυο γραμμές SDA, SCL, είναι σε υψηλή στάθμη. Το κύκλωμα οδήγησης των διαφόρων συσκευών μπορεί μόνο να θέσει σε μηδενική στάθμη τις γραμμές SDA, SCL ενώ η υψηλή στάθμη δίνεται από τις εξωτερικές pull-up αντιστάσεις. Η ακριβής τιμή των αντιστάσεων δεν παίζει καθοριστικό ρόλο στην λειτουργία ενός συστήματος επικοινωνίας μέσω του πρωτοκόλλου I²C. Οι τιμές των αντιστάσεων μπορούν να κυμαίνονται από 1k8 (1800 ohms) έως 47k (47.000 ohms). Η πιο συχνά χρησιμοποιούμενες τιμές είναι 1k8, 4k7 και 10k. Η απουσία όμως αυτών των αντιστάσεων θα είχε ως αποτέλεσμα οι γραμμές των SCL, SDA να «κρατιούνται» πάντα σε χαμηλό επίπεδο - κοντά στα 0 volts - και το I²C να μην λειτουργεί. Η τιμή των pull-up αντιστάσεων μαζί με την χωρητικότητα των γραμμών και των εισόδων των συσκευών καθορίζουν την καθυστέρηση στην άνοδο του παλμού και για αυτό

θα πρέπει το γινόμενο τους να είναι αρκετά μικρό ώστε να είναι ευδιάκριτο το μέτωπο του παλμού. Για το λόγο αυτό η τιμή των pull-up αντιστάσεων θα πρέπει να είναι αρκετά μικρή.

2.3 Διαμόρφωση εύρους παλμών (PWM)

Τα αρχικά PWM προέρχονται από τις λέξεις Pulse Width Modulation που σημαίνει διαμόρφωση εύρους παλμών. Μία PWM κυματομορφή στην πραγματικότητα αποτελεί μία περιοδική κυματομορφή η οποία έχει δύο τμήματα. Το τμήμα ON στο οποίο η κυματομορφή έχει την μέγιστη τιμή της και το τμήμα OFF στο οποίο έχει την τιμή μηδέν. Το ON τμήμα ονομάζεται Duty Cycle και μετριέται είτε σε μονάδες χρόνου (ms, us κλπ) είτε σε ποσοστό (%) επί της περιόδου.



Στην παραπάνω εικόνα βλέπουμε 3 κυματομορφές PWM με duty cycle 25%, 50%, 75%. Εφαρμόζοντας μία PWM κυματομορφή μπορούμε να ελέγξουμε το ποσοστό της ισχύος που πέφτει πάνω σε ένα φορτίο, να ελέγξουμε σέρβο κινητήρες, στροφές κινητήρων κ.α. μερικά παραδείγματα χρήσης PWM που χρησιμοποιήθηκαν σε αυτή την εργασία θα δούμε στο επόμενο κεφάλαιο.

2.4 Μετατροπείς αναλογικού σήματος σε ψηφιακό (ADC)

Οι μετατροπείς αναλογικού σήματος σε ψηφιακό (ADC) μετατρέπουν τα αναλογικά σήματα σε ψηφιακά ώστε οι λαμβανόμενες τιμές μέτρησης να μπορούν να τύχουν περεταίρω επεξεργασίας από έναν υπολογιστή. Υπάρχουν τρεις τρόποι ταξινόμησης των μετατροπέων αναλογικού σήματος σε ψηφιακό:

Ανάλογα με το αν είναι προγραμματιζόμενοι ή μη προγραμματιζόμενοι. Στους προγραμματιζόμενους ADCs η μετατροπή εκτελείται σε χρονικά δοσμένο αριθμό βημάτων.

Ανάλογα με το αν είναι τύπου ανατροφοδότησης ή ανοικτού βρόχου. Στους ADCs ανοικτού βρόχου γίνεται απευθείας σύγκριση μεταξύ της αναλογικής εισόδου και μιας τάσης αναφοράς. Στους ADCs ανατροφοδότησης η αναλογική τάση εισόδου συγκρίνεται με μια αναλογική τάση που δημιουργείται εσωτερικά ως συνάρτηση της ψηφιακής εξόδου και ανατροφοδοτεί την μια είσοδο του συγκριτή.

Ανάλογα με το αν χρησιμοποιούν μεθόδους σύγκρισης τάσεων ή ολοκλήρωσης. κάθε μια από αυτές τις δυο μεθόδους μετατροπής πραγματοποιείται με τρεις τουλάχιστον διαφορετικές τεχνικές.

Μέθοδος	Ταχύτητα	Τεχνική	12-bit fs	8-bit fs
Σύγκριση Τάσης	1 ms-10ns	Διαχρονική προσέγγιση	500 kHz	8 MHz
		Ανίχνευση	250 Hz	4 kHz
		Ταυτόχρονα		100 MHz
Ολοκλήρωση	0.4s-1ms	Διπλή κλίση	100 Hz	1.6 kHz
		Εξισορρόπηση φορτίου	60 Hz	1 kHz
		Μετατροπή τάσης συχνότητας	2.5 Hz	40 Hz

2.5 Πρωτόκολλο επικοινωνίας 802.15.1(Bluetooth)

Το Bluetooth είναι σχεδιασμένο για μικρού μήκους ρυθμού μετάδοσης δεδομένων, μειωμένης κατανάλωσης ισχύος και μικρού κόστους. Η συχνότητα στην οποία δουλεύει το Bluetooth είναι από 2,4Ghz έως 2.485 Ghz , εντός εύρους περίπου 10 μέτρων. Η συχνότητα αυτή ανήκει στη ζώνη ελεύθερης χρήσης δηλαδή δεν απαιτεί κάποιου είδους αδείας από την ΕΕΤΤ. Γι αυτό και στη συχνότητα αυτή μεταδίδουν πάρα πολλές συσκευές όπως τα wifi, Xbee κ.α. Για να αποφευχθούν οι παρεμβολές από συσκευές που έχουν την ίδια συχνότητα, τα Bluetooth έχουν την ιδιότητα να αλλάζουν τη συχνότητα τους 1600 φορές ανά δευτερόλεπτο. Το Bluetooth μπορεί να δέχεται και να λαμβάνει δεδομένα ωστόσο η ταχύτητα δεν είναι ιδιαίτερα υψηλή, βέβαια για την μετάδοση ομιλίας, μουσικής είναι ικανοποιητική. Ένα ακόμα προσόν είναι και η χαμηλή κατανάλωση ενεργείας.

2.6 Xbee 802.15.4 (Series 2)

Τα Xbee S2 pro είναι ασύρματοι πομποδέκτες, μπορούν να στέλνουν και να λαμβάνουν δεδομένα μέσω μια σειριακής θύρας, πράγμα που σημαίνει ότι μπορούν να συνδεθούν με πάρα πολλές συσκευές όπως υπολογιστές μικροεπεξεργαστές κ.α. Μπορούν να χρησιμοποιηθούν ως ένα ζεύγος ανταλλαγής δεδομένων ή σαν δικτυωτά δίκτυα από δεκάδες Xbees.

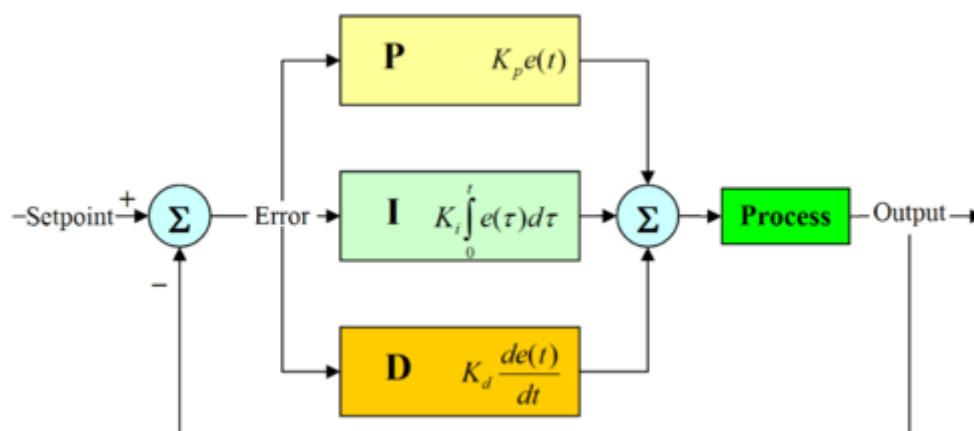


Το Xbee pro S2 έχει συχνότητα μετάδοσης 2.4 Ghz και η μέγιστη απόσταση είναι περίπου 3 χιλιόμετρα. Ο ρυθμός μετάδοσης δεδομένων είναι 250 kbps και η τάση λειτουργίας είναι από 2.7-3.6 Volt.

Για να προγραμματίσουμε το Xbee χρειαζόμαστε ένα USB to Serial με επίπεδο λειτουργίας 3.3V και το πρόγραμμα XCTU.

2.7 Ελεγκτής PID

Ένας αναλογικός-ολοκληρωτικός-παραγωγικός ελεγκτής (ελεγκτής PID) είναι ένας γενικός μηχανισμός με ανατροφοδότηση βρόχων ελέγχου που χρησιμοποιείται ευρέως στα βιομηχανικά συστήματα ελέγχου. Ένας ελεγκτής PID προσπαθεί να διορθώσει το λάθος μεταξύ μιας μετρημένης μεταβλητής-διαδικασίας (ProcessValue) και ενός επιθυμητού σημείου λειτουργίας (setpoint) με τον υπολογισμό και έπειτα με την έξοδο μιας διορθωτικής δράσης που μπορεί να ρυθμίσει τη διαδικασία αναλόγως.



Ο υπολογισμός της εξόδου του ελεγκτή PID (αλγόριθμος) περιλαμβάνει τρεις ξεχωριστούς όρους. Τον αναλογικό, ολοκληρωτικό και παραγωγικό όρο. Το αναλογικό κέρδος καθορίζει την αντίδραση στο τρέχον λάθος, το ολοκλήρωμα καθορίζει την αντίδραση βασισμένη στο άθροισμα των λαθών και η παράγωγος καθορίζει την αντίδραση βάση του ποσοστού στο οποίο το λάθος έχει αλλάξει. Το σταθμισμένο ποσό αυτών των τριών ενεργειών χρησιμοποιείται για να ρυθμίσει τη διαδικασία μέσω ενός στοιχείου ελέγχου όπως η θέση μιας βαλβίδας ελέγχου ή η παροχή ηλεκτρικού ρεύματος ενός στοιχείου θέρμανσης, κινητήρα κτλ.

Με τον “συντονισμό” των τριών σταθερών στον αλγόριθμο ελεγκτών PID, το PID μπορεί να παρέχει τη δράση ελέγχου που χρειάζεται για τις συγκεκριμένες απαιτήσεις της διαδικασίας. Η έξοδος του ελεγκτή μπορεί να περιγραφεί από τον τρόπο της απόκρισης του ελεγκτή σε ένα λάθος, ο βαθμός στον οποίο ο ελεγκτής κάνει υπερανύψωση από το setpoint και το βαθμό των ταλαντώσεων του συστήματος. Σημειώστε ότι η χρήση του αλγορίθμου PID για τον έλεγχο δεν εγγυάται το βέλτιστο έλεγχο του συστήματος.

Μερικές εφαρμογές μπορούν να απαιτήσουν μόνο έναν ή δύο όρους για να παρέχουν τον κατάλληλο έλεγχο συστημάτων. Αυτό επιτυγχάνεται με τον καθορισμό του κέρδους των ανεπιθύμητων όρων ελέγχου σε μηδέν. Ένας ελεγκτής PID θα λέγεται ελεγκτής pi , pd , p ή i ανάλογα με την έλλειψη των αντίστοιχων όρων ελέγχου. Οι ελεγκτές pi είναι ιδιαίτερα κοινοί, δεδομένου ότι η δράση του παραγωγικού όρου είναι πολύ ευαίσθητη στο θόρυβο μέτρησης.

3. ΣΚΑΦΟΣ ΚΑΙ ΒΑΣΗ

3.1 Η βάση του σκάφους

Η βάση του σκάφους είναι φτιαγμένη από ξύλο για την προστασία του σκάφους από χτυπήματα στη μεταφορά. Κατά μήκος της βάσης υπάρχουν δυο σχοινιά(λαβες) για διευκόλυνση στη μεταφορά. Το επάνω μέρος είναι ανοιγόμενο όπως φαίνεται και στην εικόνα 1.



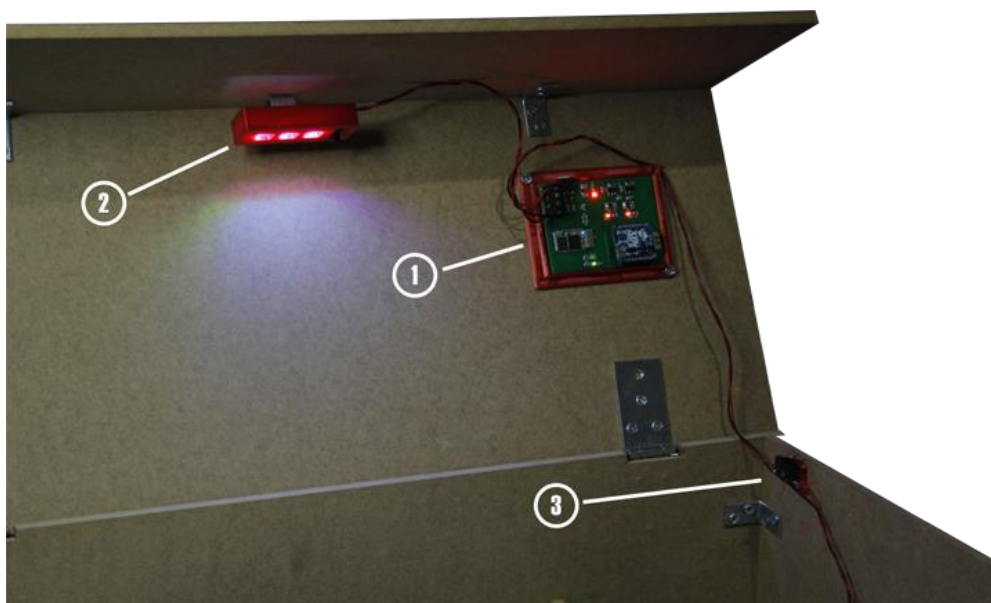
Εικόνα 1

Στη βάση υπάρχει η πλακέτα που μετατρέπει το σήμα του bluetooth που λαμβάνει από το android smartphone σε RF μέσω του Xbee, όπου η λειτουργία του εξηγείται αναλυτικά στο κεφάλαιο 5.. Η πλακέτα είναι τοποθετημένη στο άνω μέρος της βάσης για την καλύτερη δυνατή ποιότητα σήματος. Στη δεξιά πλευρά της βάσης υπάρχει ένας τερματοδιακοπτής για την αυτόματη ενεργοποίηση του κυκλώματος της πλακέτας και της λυχνίας τη στιγμή που ανοίγει το άνω μέρος της βάσης. Επίσης υπάρχει μια λυχνία led που τροφοδοτείται όπως και το

κύκλωμα από μια επαναφορτιζόμενη μπαταρία μολύβδου οξέος 12 volt και 2200 mah.

Στην εικόνα 2 υποδεικνύονται τα ακόλουθα :

1. Πλακέτα μετατροπής σήματος
2. Λυχνία Led
3. Τερματοδιακοπής



Εικόνα 2

Για την βάση χρησιμοποιήθηκε ξύλο τύπου MDF πάχους 8mm με τις ακόλουθες διαστάσεις:

1. 61.7cm x 26cm (x2)
2. 61.7cm x 30cm (x1)
3. 23.6cm x 30cm (x2)
4. 23.6cm x 15cm (x2)
5. 61.7cm x 17cm (x1)
6. 61.7cm x 13cm (x1)

3.2 Το σκάφος

Το σκάφκι είναι κατευθυνόμενο από κινητό android, είναι υδρόψυκτο με αυτόματη ενεργοποίηση της αντλίας νερού όταν η θερμοκρασία ξεπεράσει την προκαθορισμένη τιμή. Το επίπεδο μπαταρίας καθώς και θερμοκρασίας απεικονίζονται στην εφαρμογή του κινητού. Στη δεξιά και αριστερή πλευρά του σκάφους έχουν τοποθετηθεί από 2 λυχνίες Led στην κάθε πλευρά διαφορετικού χρώματος για να υποδεικνύουν την κατεύθυνση του σκάφους στο σκοτάδι. Η προπέλα που έχει επιδεχθεί δίνει τη δυνατότητα το σκάφος να μπορεί να τραβήξει φορτίο, δηλαδή σχετικά χαμηλή ταχύτητα με πολλές στροφές στον κινητήρα. Διαθέτει επίσης σύστημα αυτομάτου πιλότου για την διατήρηση του σε σταθερή πορεία και σύστημα G.P. για τον εντοπισμό της ακριβούς θέσης του.

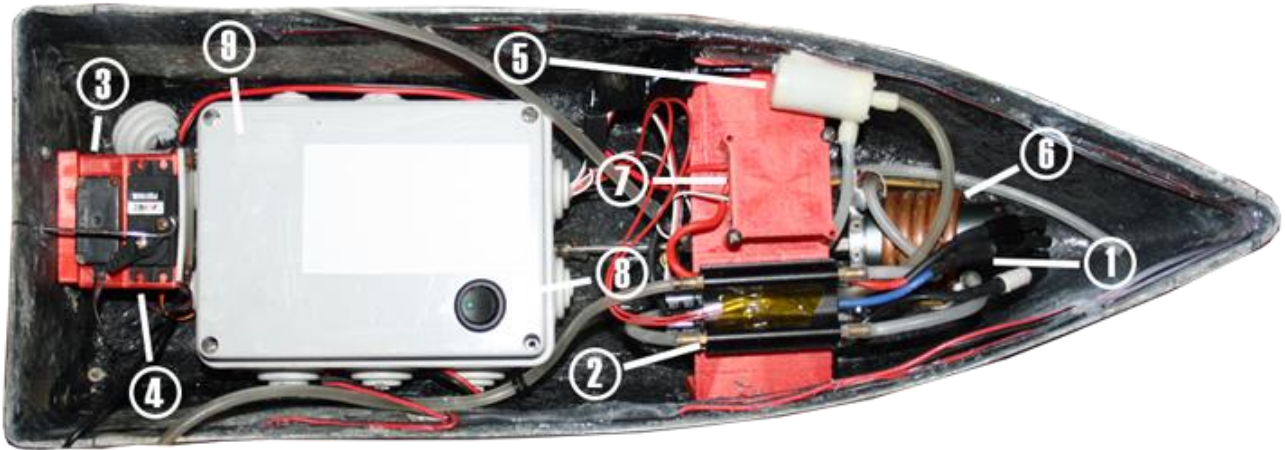


Εικόνα 3

Το πλαίσιο είναι κατασκευασμένο από πολυεστέρα. Στα τοιχώματα δεξιά και αριστερά στο κάτω μέρος διαγράφεται μια καμπυλότητα, στο εσωτερικό της καμπυλότητας αυτής υπάρχει αφρός πολυουρεθάνης. Αυτό έχει ως αποτέλεσμα εάν πάρει νερά το σκάφος στο εσωτερικό του για τον οποιοδήποτε λόγο, το σκάφος να παραμείνει στην επιφάνεια της θάλασσας. Το καλούπι κατασκευαστικό από

Λεπτό ξύλο το οποίο και αφαιρέθηκε μετά το πέρας της διαδικασίας του στεγνώματος του πολυεστέρα. Εσωτερικά και εξωτερικά έχουν κατασκευαστεί βάσεις και κουτιά στήριξης από 3d εκτυπωτή, με το σχεδιαστικό πρόγραμμα inventor.

Στην εικόνα 4 υποδεικνύονται τα εσωτερικά μέρη του σκάφους



Εικόνα 4

1. Κινητήραςxk2040-4500kv Brushless Inrunner
2. Ρυθμιστής ταχύτητας 70A Boat ESC1AUBEC
3. Σερβοκινητήρας πηδάλιου
4. Σερβοκινητήρας απαγκίστρωσης
- 5.Αντλία νερού
6. σωληνάκι ψύξης κινητήρα
7. Κύκλωμα επανεκκίνησης ρυθμιστή ταχύτητας
8. Αδιάβροχος διακόπτης ON/OFF
9. Αδιάβροχο κουτί θωράκισης πλακέτας, μπαταριάς

Στην εικόνα 5 υποδεικνύονται τα εξωτερικά μέρη του σκάφους.

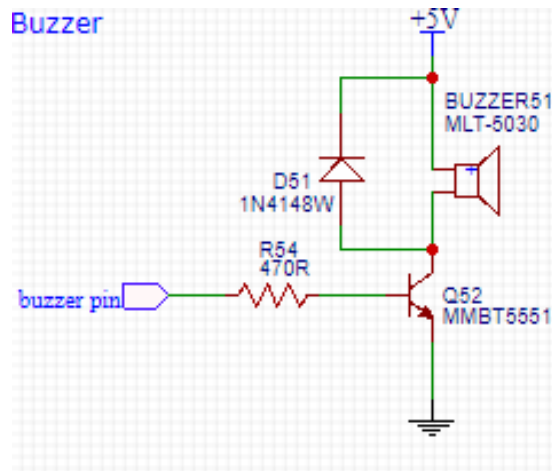


Εικόνα 5

1. Εισαγωγή-εξαγωγή νερού για ψύξη, κινητήρα και ρυθμιστή ταχύτητας
2. Απαγκιστρωτής
3. ΛυχνίαLed

4. Κύκλωμα και συναρτήσεις

4.1 Buzzer



Η συνάρτηση buzzercorrect χρησιμοποιείται κυρίως για να δείξει αν η σύνδεση με το Bluetooth, το xbee είναι επιτυχής και η μετάδοση δεδομένων έχει ξεκινήσει . Αν το σήμα έχει χαθεί ή κάτι πήγε λανθασμένα για κάποιο λόγο τότε η συνάρτηση buzzerincorrect ενεργοποιείται

```
#define buzzer 6
```

```
void buzzercorrect()
```

```
{  
    digitalWrite(buzzer,HIGH);  
    delay(120);  
    digitalWrite(buzzer,LOW);  
    delay(50);  
    digitalWrite(buzzer,HIGH);  
    delay(120);  
    digitalWrite(buzzer,LOW);  
}
```

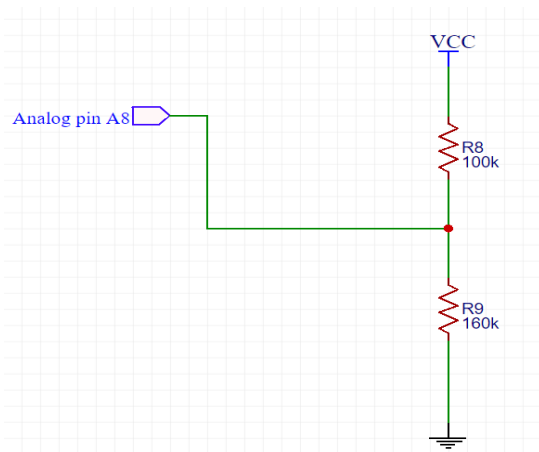
```
void buzzerincorrect()
```

```
{  
    digitalWrite(Led, HIGH);  
    digitalWrite(buzzer,HIGH);  
    delay(500);  
    digitalWrite(buzzer,LOW);  
    delay(250);  
    digitalWrite(buzzer,HIGH);  
    digitalWrite(Led, LOW);  
    delay(500);  
    digitalWrite(buzzer,LOW);  
}
```

4.2 Έλεγχος επίπεδου μπαταρίας

Το ADC του μικροελεγκτή μπορεί να διαβάσει σήματα από 0-5V και μπορεί να δώσει τιμές από 0-1.024 (2^{10}) δηλαδή 10bit ADC. Για να μετατρέψουμε την τάση από τα 13V(μεγίστη τιμή της μπαταρίας)στα 5V αρκεί ένας απλός διαιρέτης τάσης. Ο μαθηματικός τύπος για τον διαιρέτη τάσης είναι ο εξής:

$$V_{out} = \frac{R2}{R1 + R2} V_{cc}$$



συμπεραίνουμε ότι οι αντιστάσεις R1,R2 θα πρέπει να είναι 100KΩ,180KΩ αντίστοιχα, όπως φαίνεται και στην παραπάνω εικόνα.

Επειδή ο Speed controller (ESC) μπορεί να δουλεύει με τάση από 10-13V, πρέπει το ποσοστό της μπαταρίας που θα παίρνουμε να μετατρέπεται από 10-13V σε 0-100%. Αυτό επιτυγχάνεται

με τη χρήση της εντολής map όπως φαίνεται και στον κώδικα από κάτω.

```
#define battery A8
```

```
int Batterylevel()
```

```
{
```

```
int pinakas[5];
```

```
int i,sum,mo;
```

```
for (i=0;i<=5;i++)
```

```
{
```

```
pinakas[i]=analogRead(battery);
```

```
delay(5);
```

```
}
```

```
for (i=0;i<=5;i++)
```

```
{
```

```
sum=sum+pinakas[i];
```

```
delay(5);
```

```
}
```

```
mo=sum/6;
```

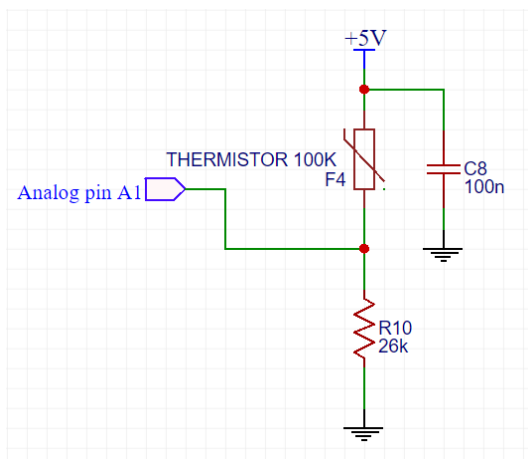
```
mo=map(mo,765,985,0,100);
```

```
return mo;
```

```
}
```

4.3 Έλεγχος θερμοκρασίας Speed Controller (ESC)

Το ESC είναι αυτό που μετατρέπει την τάση από συνεχή σε τριφασική για να δουλέψει ο ηλεκτροκινητήρας του σκάφους. Ο έλεγχος των στροφών γίνεται με PWM από τον μικροελεγκτή διάμεσου της επαφής signal του speed controller. Ο ESC έχει υψηλές θερμοκρασίες με αποτέλεσμα να πρέπει να ψύχεται ανά διαστήματα, όταν, δηλαδή, η θερμοκρασία είναι υψηλή.



Για να μπορέσουμε να ελέγξουμε την θερμοκρασία χρησιμοποιήσαμε το θερμίστορ το οποίο είναι μια αντίσταση η οποία μεταβάλλεται ανάλογα με την θερμοκρασία. Υπάρχουν 2 ειδών θερμίστορ τα NTC, PTC.

Τα NTC (Negative Temperature Coefficient), στα οποία η αντίσταση μειώνεται καθώς αυξάνεται η θερμοκρασία.

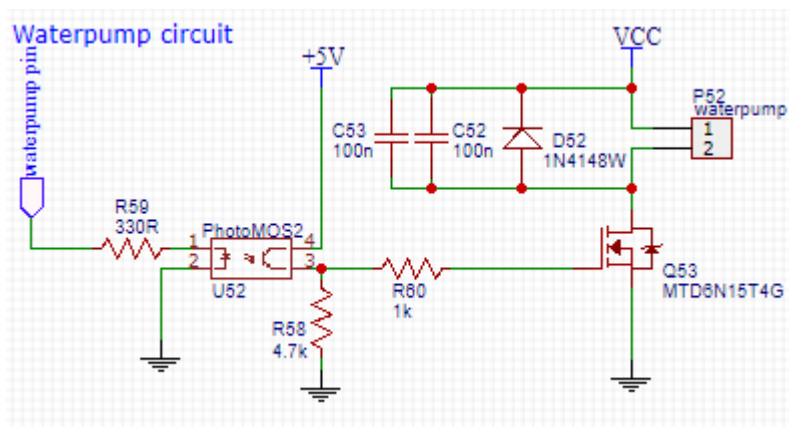
Τα PTC (Positive Temperature Coefficient), στα οποία η αντίσταση αυξάνεται καθώς αυξάνεται η θερμοκρασία.

Για την συγκεκριμένη εφαρμογή χρησιμοποιήθηκε θερμίστορ NTC 100KΩ στους 25°C.

Με κλίμακα από 0-100°C και θεωρώντας γνωστό ότι το ADC μπορεί να ψηφιοποιήσει 0-5V με τιμές 0-1024. Επομένως, στους 25°C η τάση στα άκρα της αναλογικής ακίδας του μικροελεγκτή πρέπει να είναι 1.25V. Συμφώνα με τη συνδεσμολογία της παραπάνω εικόνας και τη μαθηματική έκφραση του διαιρέτη τάσης επιλέχτηκε $R10=26K\Omega$.

4.4 Αντλία νερού

Όταν η θερμοκρασία του speed controller ξεπεράσει κάποιους βαθμούς, τότε ο ελεγκτής δίνει σήμα στην αντλία νερού, η αντλία ενεργοποιείται και έτσι ψύχεται ο κινητήρας και ο speed controller.



Για να διασφαλίσουμε την ασφάλεια του κυκλώματος από κάποιο πιθανό βραχυκύκλωμα έχει τοποθετηθεί ορτοcoupler. Αυτό χρησιμεύει ώστε οποιαδήποτε βλάβη και αν προκληθεί στο κύκλωμα της αντλίας νερού να μην επηρεάσει τα άλλα μέρη της πλακέτας.

```
int Temp()
```

```
{  
    int pinakas[5];  
    int i,sum,mo;  
    for (i=0;i<=5;i++) {  
        pinakas[i]=analogRead(thermistor);  
        delay(5); }  
    for (i=0;i<=5;i++) {  
        sum=sum+pinakas[i];  
        delay(5);  
    }  
}
```

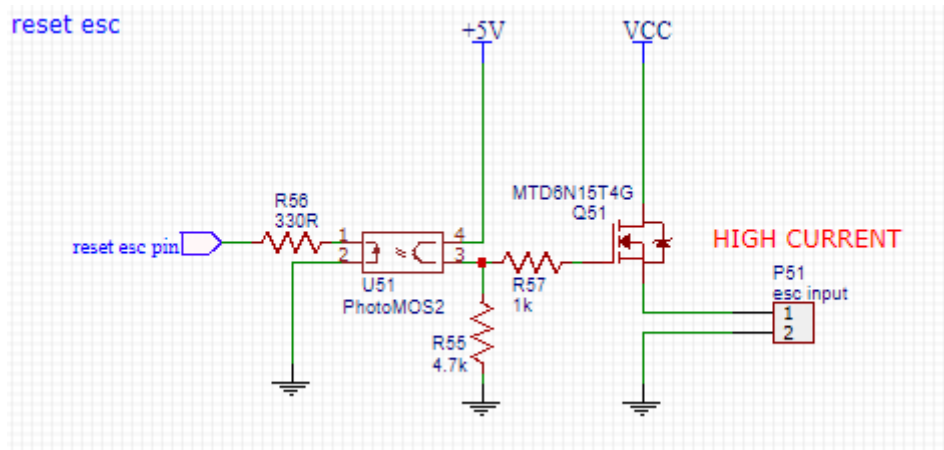
```
    mo=sum/6;  
    mo=map(mo,0,1023,0,100);  
    return mo;  
}
```

```
void pumpspeed(int temp)
```

```
{  
    if (temp>28){  
        digitalWrite(waterpump,HIGH);  
    } else {  
        digitalWrite(waterpump,LOW);  
    }  
}
```

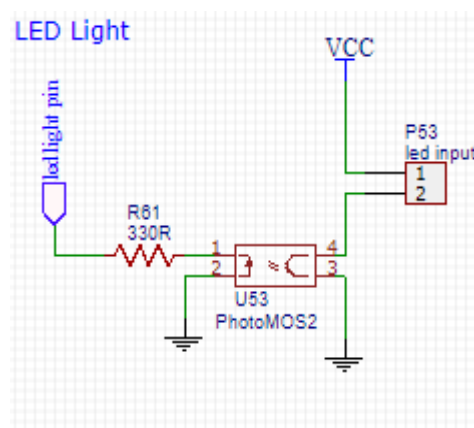
4.5 Reset

Το κύκλωμα αυτό απενεργοποιεί τον speed controller και ενεργοποιείται ξανά μετά από 3s. Επίσης μέσα από την εφαρμογή αρχικοποιούνται κάποιες μεταβλητές που έχουν να κάνουν με το speed controller.



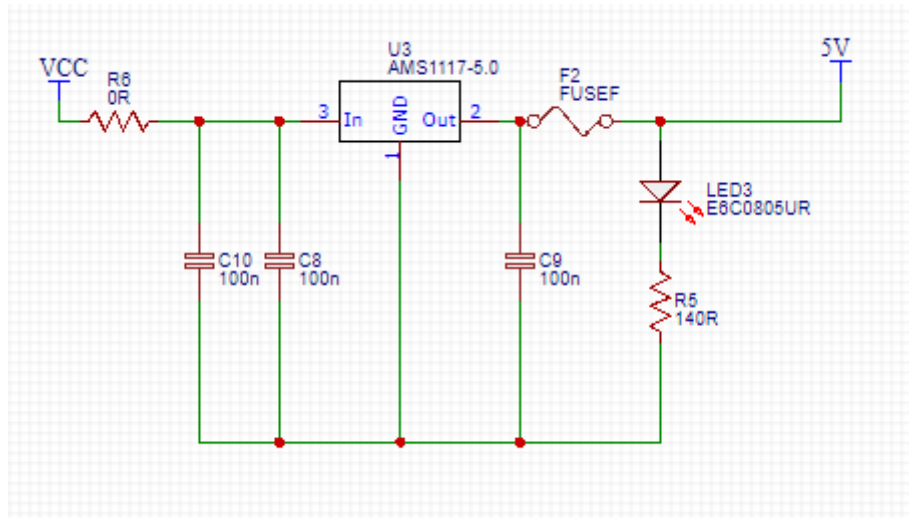
4.6 LED Light

Στο σκάφος υπάρχουν 2X3 led κόκκινα στο πίσω μέρος και 2X3 άσπρα στο μπροστά για να μπορούμε να διακρίνουμε την πορεία του σκάφους από μεγαλύτερη απόσταση.

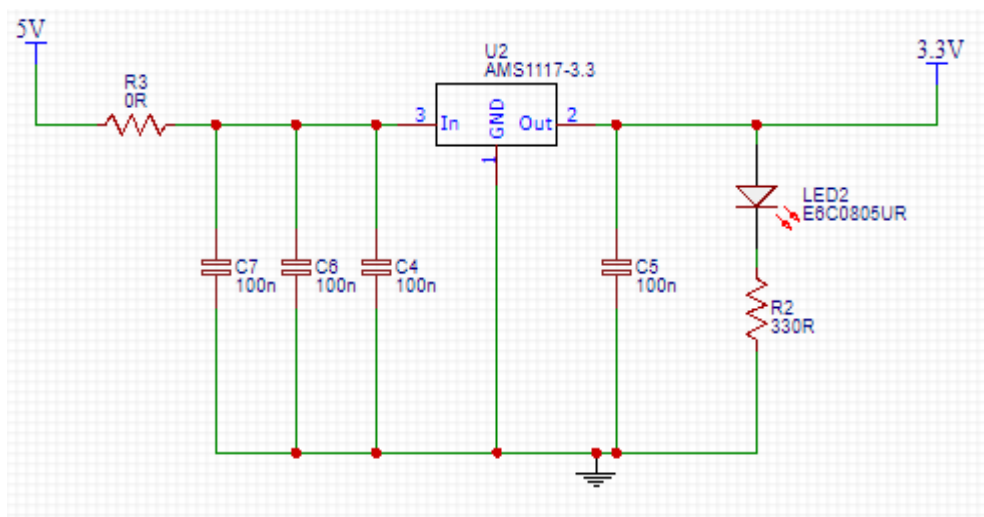


4.7 Τροφοδοτικά

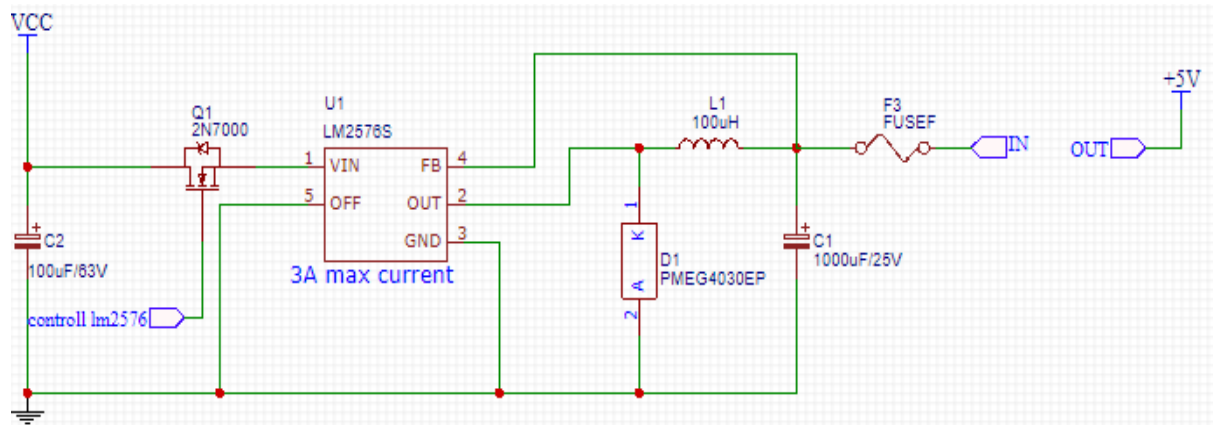
Στο σκάφος υπάρχουν 3 τροφοδοτικά. Ένα γραμμικό των 5V 1A για τον ελεγκτή και τα περιφερικά κυκλώματα, ένα γραμμικό 3,3V 1A για την τροφοδότηση του xbee και ένα παλμικό 5V 3A για τον τροφοδότηση των σέρβο κινητήρων.



α) Γραμμικό τροφοδοτικό 5v



β) Γραμμικό τροφοδοτικό 3,3V



γ)παλμικό τροφοδοτικό 5V

4.8 Ταχύτητα κινητήρα και διεύθυνση πηδαλίου

4.8.1 Κινητήρας

Για να λειτουργήσει ο κινητήρας και να μπορούμε να ελέγξουμε την ταχύτητα του, έχει τοποθετηθεί ο Speed controller (ESC). Το σήμα που δέχεται ο ρυθμιστής ταχύτητας από τον μικροελεγκτή είναι σήμα PWM σε μικροδευτερόλεπτα και σύμφωνα με το datasheet του speed controller για κίνηση εμπρός τα μικροδευτερόλεπτα είναι από 1185us-1875us και για κίνηση προς τα πίσω είναι από 1185us-1000us.

```
void escspeed(int taxitita)
```

```
{
if ((taxitita>=1000)&&(taxitita<=1030)){
taxitita2 = map(taxitita, 1000, 1030,1185,1875);
esc.writeMicroseconds(taxitita2); }
else if ((taxitita>=1200)&&(taxitita<=1230)) {
taxitita2 = map(taxitita, 1200, 1230,1185,1000);
esc.writeMicroseconds(taxitita2);
} }
```

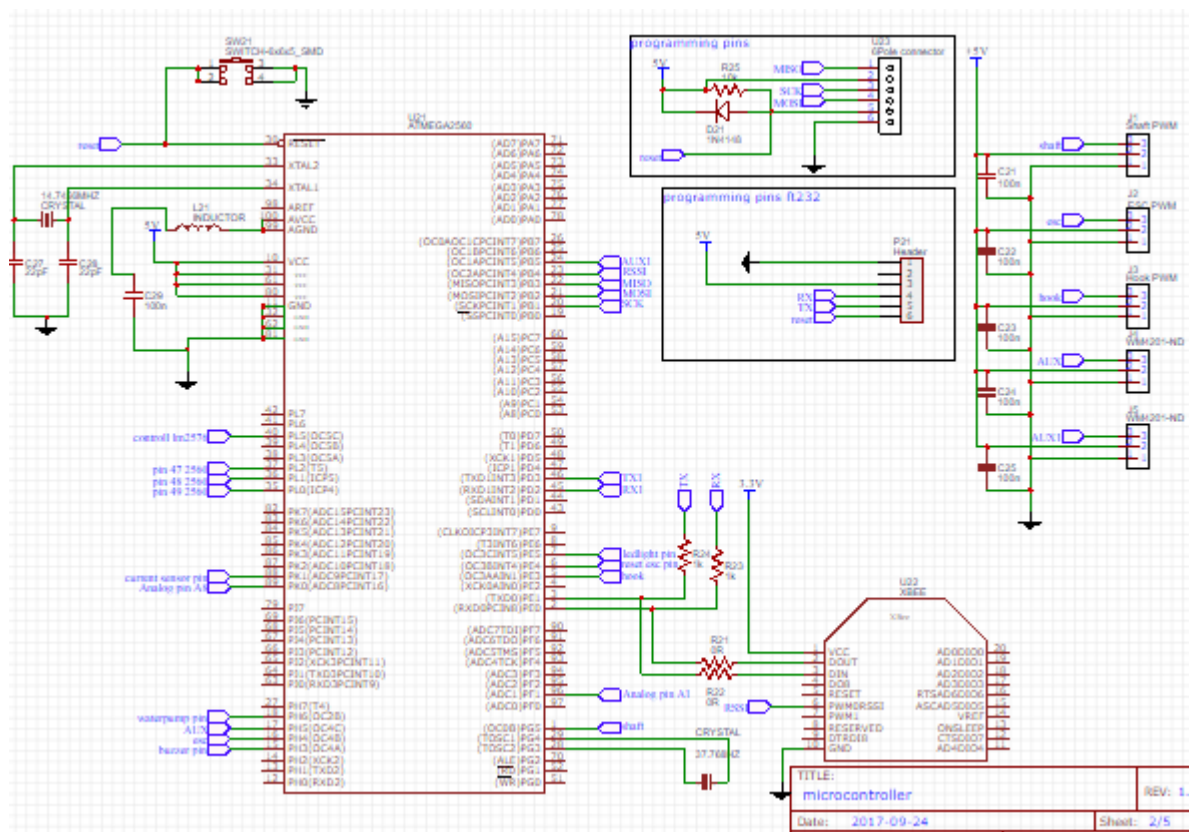
4.8.2 Πηδάλιο

Το πηδάλιο είναι ένας αεροκινητήρας ο οποίος βρίσκεται στο πίσω μέρος του σκάφους και από αυτό εξαρτάται η διεύθυνση του. Ο σέρβο κινητήρας μπορεί να περιστραφεί έως 180 μοίρες δηλαδή 90 μοίρες δεξιά και 90 αριστερά. Από τις δοκιμές που έγιναν παρατηρήθηκε ότι έχει την καλύτερη απόκριση στην διεύθυνση του έως τις 35 μοίρες και γι αυτό το λόγο επιλέχθηκε η γωνία στρέψεις του πηδαλίου να είναι 35 μοίρες δεξιά και 35 αριστερά.

```
void rudder1(int pidalio)
```

```
{  
  if ((pidalio>134)&&(pidalio<170)) {  
    pidalio1=map(pidalio,135,170,90,125);  
  }  
  if ((pidalio>99)&&(pidalio<136)) {  
    pidalio1=map(pidalio,135,100,90,55);  
  }  
  rudder.write(pidalio1);  
}
```

4.9 Κύκλωμα ATMEGA 2560

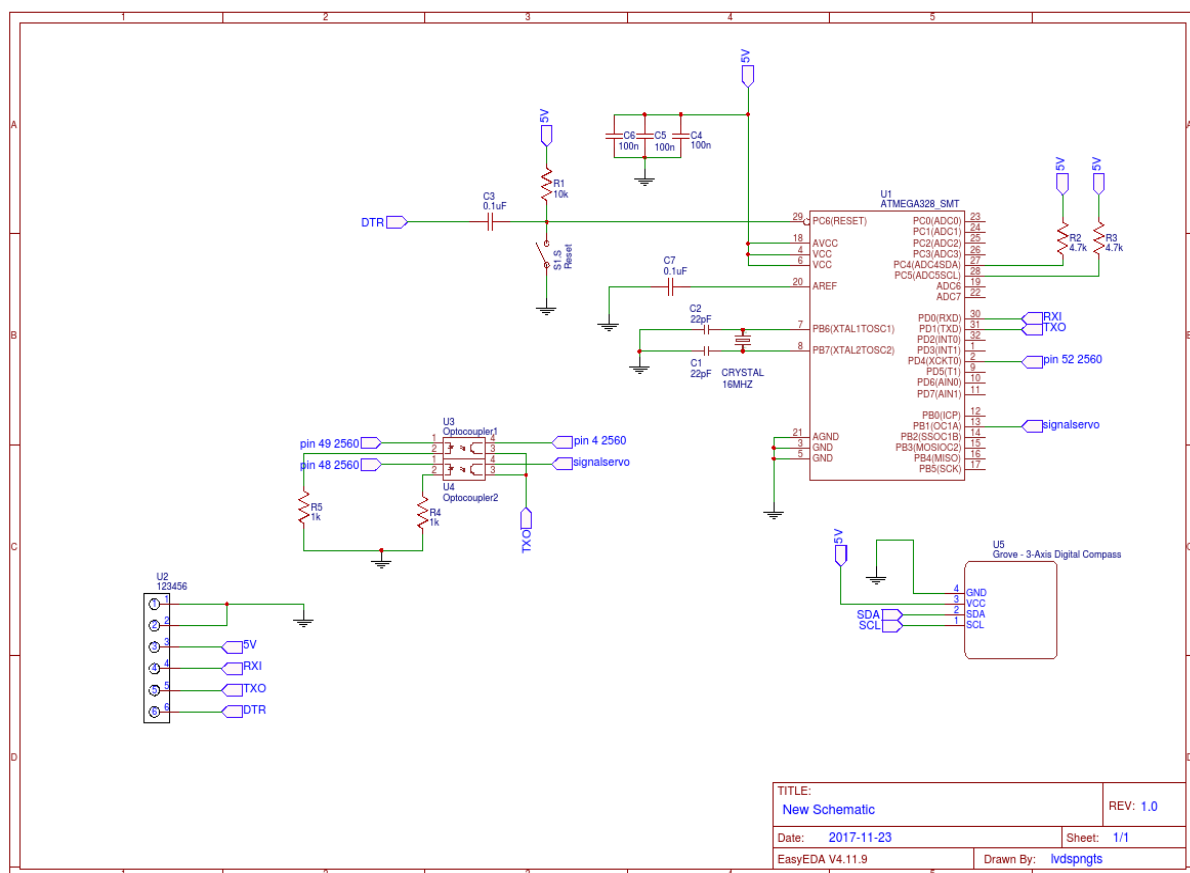


5 Αυτόματος πιλότος

Για την λειτουργία του αυτομάτου πιλότου χρησιμοποιήθηκε ο ελεγκτής PI. Για να μπορέσουμε να βρούμε τους όρους K_c , T_i χρησιμοποιήθηκε μια από τις πιο διαδεδομένες μεθόδους βαθμονόμησης, η μέθοδος Ziegler-Nichols, η οποία στηρίζεται στον προσδιορισμό των κρίσιμων χαρακτηριστικών ενός συστήματος κρίσιμη περίοδο P_{cr} , και κρίσιμη ενίσχυση/κερδος K_{cr} για τον υπολογισμό των παραμέτρων του ελεγκτή. Κρίσιμη ενίσχυση K_{cr} ονομάζεται η τιμή του κέρδους του αναλογικού ελεγκτή P η οποία προκαλεί αμείωτη ταλάντωση της εξόδου του συστήματος με περίοδο ίση με την κρίσιμη περίοδο P_{cr} .

Πίνακας Ziegler-Nichols			
	k_c	t_i	t_d
P	$K_{cr}/2$		
PI	$K_{cr}/2.2$	$P_{cr}/1.2$	
PID	$K_{cr}/1.7$	$P_{cr}/2$	$P_{cr}/8$

Με βάση τον πίνακα Ziegler-Nichols επιλέχτηκαν οι κατάλληλες τιμές P, I



#include <Wire.h>

#include <Servo.h>

Servo rudder;

#define on 4

bool set;

float setpoint;

int j = 1;

bool enable = false;

//////////PID//////////

float pid p = 0;

float pid i = 0;

float pid d = 0;

//////////PID CONSTANTS//////

double kp = 1.3;

double ki = 0.002;

double kd = 0;

float elapsedTime, time, timePrev;

float error, previous_error;

```

float compass()
{
    // put your main code here, to run repeatedly:

    int x, y, z; //triple axis data


    //Tell the HMC5883 where to begin reading data
    Wire.beginTransmission(0x1E);
    Wire.write(0x03); //select register 3, X MSB register
    Wire.endTransmission();

    //Read data from each axis, 2 registers per axis
    Wire.requestFrom(0x1E, 6);
    if (6 <= Wire.available()) {
        x = Wire.read() << 8; //X msb
        x |= Wire.read(); //X lsb
        z = Wire.read() << 8; //Z msb
        z |= Wire.read(); //Z lsb
        y = Wire.read() << 8; //Y msb
        y |= Wire.read(); //Y lsb
    }


    //Print out values of each axis

    float heading = atan2(y, x);

```

```

    if (set == true){
        setpoint = heading;
        set = false;
    }

    heading = heading - setpoint;

    if (heading < 0){
        heading += 2 * PI;
    }

    // Check for wrap due to addition of declination.
    if (heading > 2 * PI)
        heading -= 2 * PI;

    // Convert radians to degrees for readability.
    float headingDegrees = heading * 180 / M_PI;

    return headingDegrees;
}

float PIDcontroller(float realpoint)
{
    float error;

    if (realpoint > 180)

```

```

    {
        realpoint = map(realpoint, 359, 180, -1, -180);
    }

    timePrev = time; // the previous time is stored before the actual time
read

    time = millis(); // actual time read

    elapsedTime = (time - timePrev) / 1000;

    error = setpoint - realpoint;

    pid p = kp * error;

    if (-3 < error < 3)
    {
        pid i = pid i + (ki * error);
    }

    pid d = kd * ((error - previous_error) / elapsedTime);

    float PID = pid p + pid i + pid d;

    return PID;
}

void autopilot(float PID){

```

```

int servoPos;

if (PID <= 90) {
    servoPos = 90 + PID;
}
else{
    servoPos = 90 - PID;
}

if (servoPos > 125) {
    servoPos = 125;
}

else if (servoPos < 55) {
    servoPos = 55;
}

rudder.write(servoPos);
}

void setup() {
    pinMode(on, INPUT);
    rudder.attach(9);
    Wire.begin();
    Wire.beginTransmission(0x1E); //open communication with HMC5883
    Wire.write(0x02); //select mode register

```

```

Wire.write(0x00); //continuous measurement mode

Wire.endTransmission();

delay(1000);

}

void loop() {

  if (enable == HIGH){

    float postion;

    float pid;

    if (j == 1) {

      set = true;

      j = 2;

    }

    postion = compass();

    pid = PIDcontroller(postion);

    autopilot(pid);

  }

  else

  {

    j = 1;

    pid p = 0;

    pid i = 0;

```

pid d = 0;

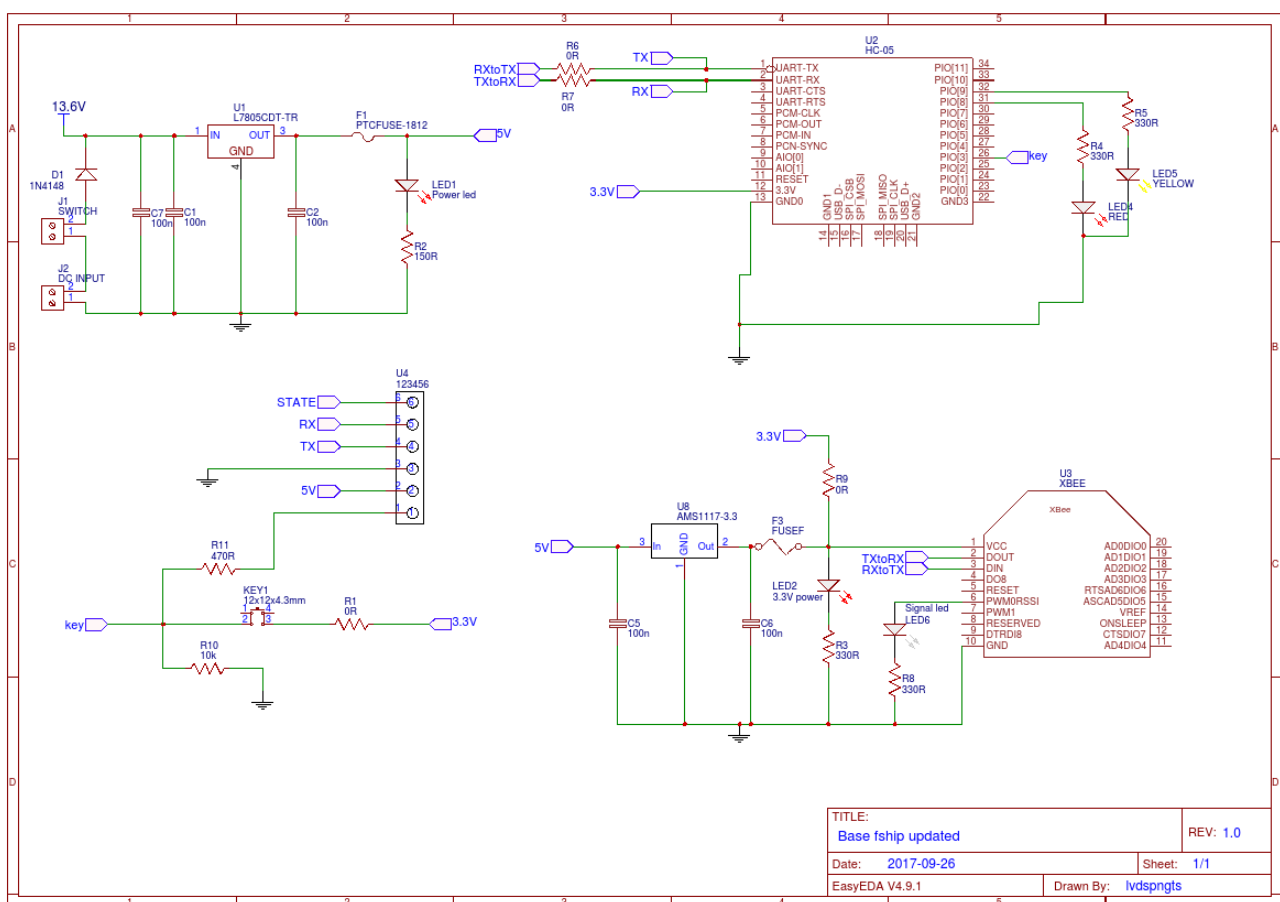
}

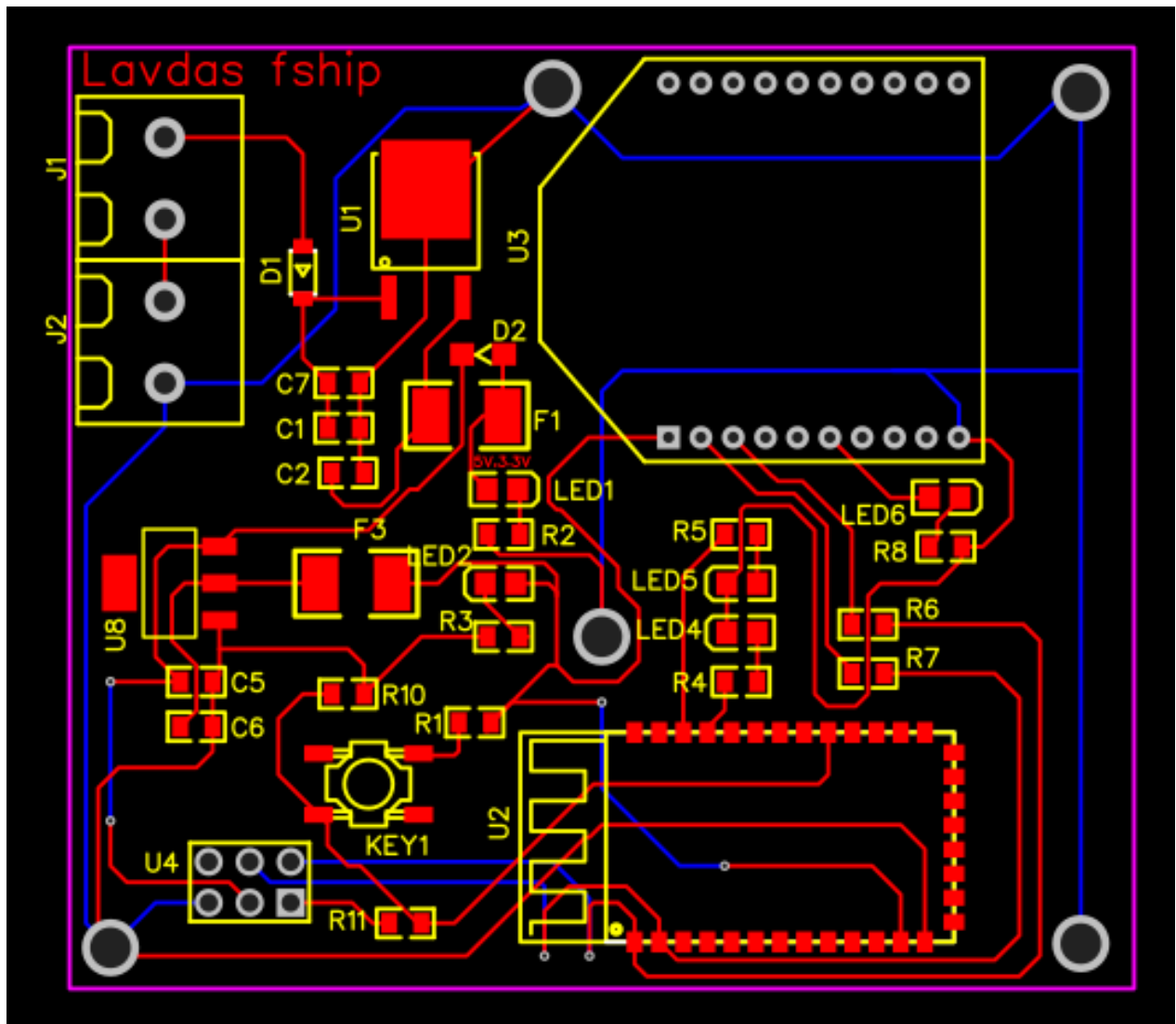
enable = digitalRead(on); delay(40);

}

6. Κύκλωμα στη βάση και τρόπος λειτουργίας.

Η βάση του σκάφους, εκτός από βάση για το σκάφος έχει και ένα μικρό κύκλωμα το οποίο μετατρέπει και μεταφέρει τα δεδομένα μέσω Bluetooth από το κινητό και από τη βάση μέσω xbee στο σκάφος. Το baud rate επιλέχθηκε 115200.





7. Εφαρμογή android

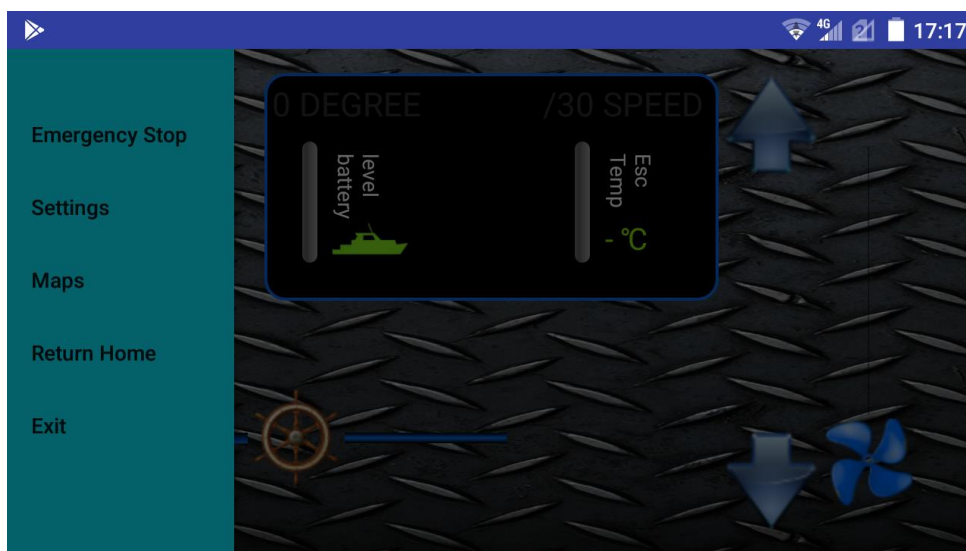
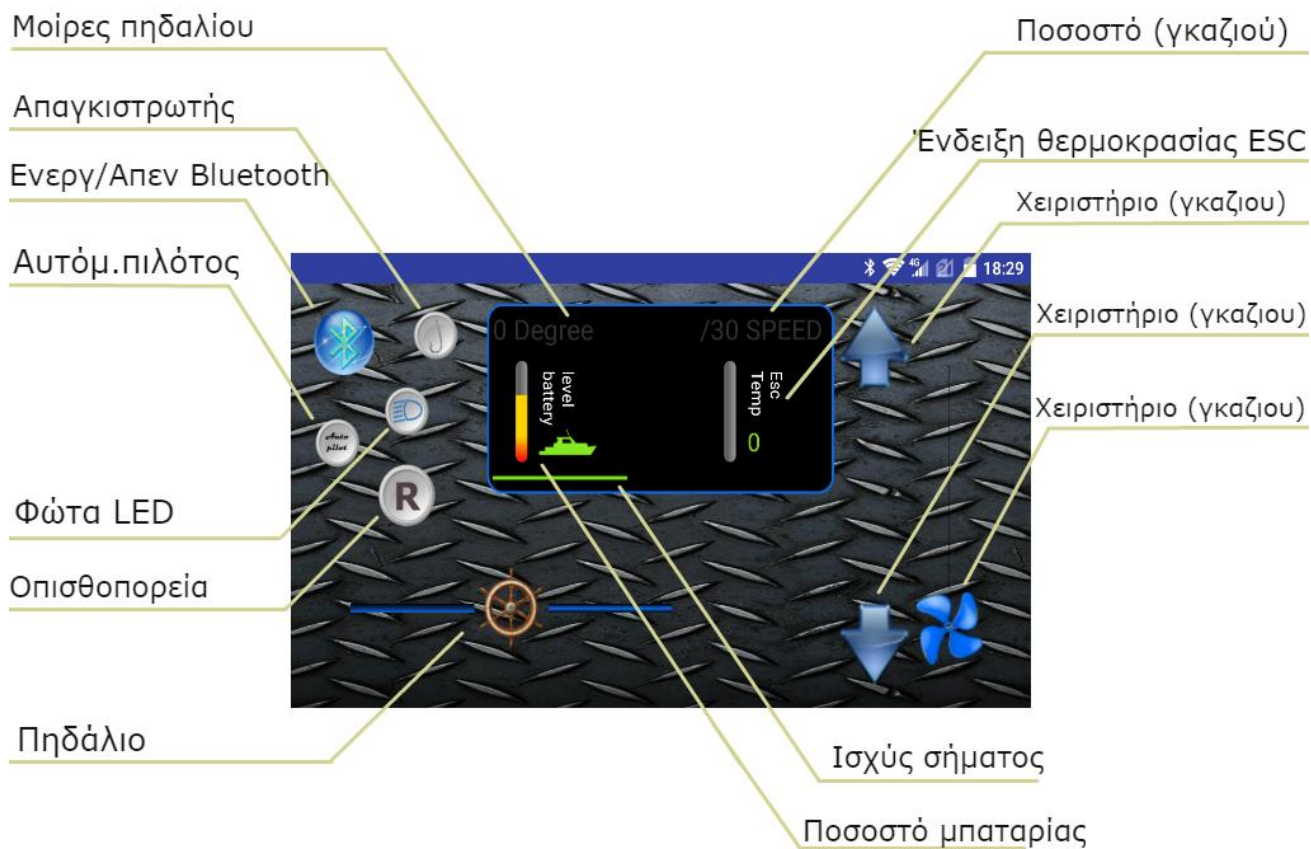
7.1 Γενικά

Το android είναι λειτουργικό σύστημα αρχικά για συσκευές κινητής τηλεφωνίας αλλά με το πέρασμα του χρόνου χρησιμοποιείται σε όλο και περισσότερες συσκευές όπως tablet, υπολογιστές, φωτογραφικές μηχανές, android-box κ.α. Το android είναι βασισμένο, δηλαδή, τρέχει τον πυρίνα του λειτουργικού linux. Με το λειτουργικό android, οι κατασκευαστές λογισμικού μπορούν να γράφουν κώδικα με τη χρήση της γλώσσας προγραμματισμού κατά κυρίαρχο λόγο, την java, ελέγχοντας τη συσκευή μέσω βιβλιοθηκών αναπτυγμένων από την Google.

Η ανάπτυξη της εφαρμογής έγινε με το android studio. Το android studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον για την ανάπτυξη εφαρμογών σε συσκευές που τρέχουν android λογισμικό. Οι περισσότερες εφαρμογές στο android γράφονται με την γλώσσα προγραμματισμού java. Βέβαια μπορούν να τρέξουν και εφαρμογές από άλλες γλώσσες προγραμματισμού όπως τις εφαρμογές web χρησιμοποιώντας HTML και JavaScript και μπορούν να χρησιμοποιήσουν το (NDK) Native Development Kit, ώστε να περιλαμβάνει κώδικα C/C++ για λόγους φορητότητας και αύξησης της απόδοσης.

7.2 Το layout (.xml) της εφαρμογής

Η εφαρμογή μας δίνει τη δυνατότητα να χειριστούμε το πηδάλιο και την ταχύτητα του σκάφους. Υπάρχουν επίσης κάποιες ενδείξεις στην οθόνη, όπως του επιπέδου της μπαταρίας, της θερμοκρασίας, της ισχύος του σήματος, των μοιρών του πηδαλίου και του ποσοστού ταχύτητας του γκαζιού. Υπάρχουν και κάποια Button από τα οποία μπορούμε να ενεργοποιήσουμε/απενεργοποιήσουμε τα φώτα, τον αυτόματο πιλότο, τον απαγκιστρωτή, το Bluetooth, την άμεση διακοπή του κινητήρα και να δούμε που βρίσκεται το σκάφος εκείνη τη στιγμή μέσω του G.P.S.



7.3 ο κώδικας της εφαρμογής, xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:keepScreenOn="true"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:background="@drawable/br"
        tools:context="com.example.lavdas.fship.MainActivity">

        <com.example.lavdas.fship.VerticalSeekBar
            android:id="@+id/seekBar1"
            android:thumb="@drawable/prop"
            android:max="30"
            android:layout_width="58dp"
            android:layout_height="320dp"
            android:indeterminate="false"
            android:paddingStart="@dimen/abc_action_bar_default_height_material"
            android:paddingEnd="@dimen/abc_action_bar_default_height_material"
            android:layout_alignParentBottom="true"
            android:layout_toEndOf="@+id/upbutton" />

        <SeekBar
            android:layout_width="310dp"
            android:layout_height="65dp"
            android:max="70"
            android:progress="35"
            android:id="@+id/seekBar2"
            android:progressDrawable="@drawable/seek_bar"
            android:thumb="@drawable/shipwheel"
            android:indeterminate="false"
            android:layout_marginBottom="30dp"
            android:layout_marginStart="20dp"
            android:minHeight="6dip"
            android:maxHeight="6dip"
            android:layout_alignParentBottom="true"
            android:layout_alignParentStart="true" />

        <ImageButton
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:background="@drawable/btcon"
            android:text="connect"
```

```

        android:id="@+id/connectbutton"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />

<FrameLayout
    android:layout_width="300dp"
    android:layout_height="150dp"
    android:background="@drawable/background_frame"
    android:id="@+id/frameLayout"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="58dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#50999999"
        android:text="/30 SPEED"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="5dp"
        android:layout_gravity="right|top" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="56dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#50999999"
        android:text="0 DEGREE"
        android:id="@+id/textView"
        android:layout_alignTop="@+id/textView2"
        android:layout_alignStart="@+id/seekBar2"
        android:layout_alignBottom="@+id/textView2"
        android:layout_gravity="left|top" />

    <ImageView
        android:id="@+id/hookview"

        android:layout_width="50dp"
        android:layout_height="30dp"
        android:layout_gravity="center_horizontal|center"
        android:alpha="0"/>

    <ImageView
        android:id="@+id/imagelig"
        android:layout_width="50dp

        android:layout_height="30dp"
        android:layout_gravity="center_horizontal|top"
        android:alpha="0"/>

    <TextView
        android:id="@+id/pilottext1"
        android:layout_width="60dp"
        android:layout_height="30dp"
        android:text="Autopilot"
        android:textColor="#FFFFFF"
        android:layout_gravity="center_horizontal|bottom"
        android:alpha="0"/>

```

```

<ImageView
    android:id="@+id/imageView"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginLeft="40dp"
    android:layout_marginTop="80dp"
    android:background="@drawable/boat" />

<ProgressBar
    android:id="@+id/progressBarBattery"
    style="@style/Widget.ProgressBar.Vertical"
    android:layout_width="10dp"
    android:layout_height="80dip"
    android:layout_alignParentTop="true"
    android:layout_marginTop="40dp"
    android:layout_marginLeft="20dp"
    android:progress="0"
    android:layout_toStartOf="@+id/backwardtxt" />

<TextView
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:textColor="#FFFFFF"
    android:rotation="90"
    android:layout_marginTop="53dp"
    android:layout_marginLeft="30dp"
    android:text="level battery"/>

<ProgressBar
    android:id="@+id/esctemp"
    style="@style/Widget.ProgressBar.Vertical2"
    android:layout_width="10dp"
    android:layout_height="80dip"
    android:layout_alignParentTop="true"
    android:layout_marginTop="40dp"
    android:layout_marginLeft="200dp"
    android:progress="0"
    android:layout_toStartOf="@+id/backwardtxt" />

<TextView
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:textColor="#FFFFFF"
    android:rotation="90"
    android:layout_marginTop="53dp"
    android:layout_marginLeft="210dp"
    android:text="Esc Temp"/>

<TextView
    android:id="@+id/esctemp1"
    android:layout_width="50dp"
    android:layout_height="40dp"
    android:textColor="#89e106"
    android:textSize="20dp"
    android:layout_marginTop="90dp"
    android:layout_marginLeft="220dp"
    android:text="- \u2103"
/>

<ProgressBar
    android:id="@+id/signal"
    android:layout_width="fill_parent"

```

```

        android:layout_height="wrap_content"
        android:progressTint="#76e40f"
        android:progress="0"
        android:layout_marginTop="125dp"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"/>

</FrameLayout>
<ImageButton
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/lightbtn"
    android:id="@+id/light"
    android:layout_below="@+id/connectbutton"
    android:layout_toEndOf="@+id/connectbutton" />

<ImageButton
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/hook"
    android:id="@+id/hook"
    android:layout_alignStart="@+id/light"
    android:layout_marginStart="25dp" />

<ImageButton
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/pilotbtn"
    android:id="@+id/pilotbtn"
    android:layout_alignBottom="@+id/frameLayout"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="15dp" />

<ImageButton
    android:id="@+id/upbutton"
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:background="@drawable/upbutton"
    android:layout_alignParentTop="true"
    android:layout_toEndOf="@+id/frameLayout" />

<ImageButton
    android:id="@+id/backward"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/backward"
    android:layout_centerVertical="true"
    android:layout_alignStart="@+id/light" />

<TextView
    android:id="@+id/backwardtxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#FFFFFF"
    android:textSize="25dp"
    android:layout_alignBottom="@+id/backward"
    android:layout_centerHorizontal="true" />

<ImageButton
    android:id="@+id/downbutton"

```

```
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:background="@drawable/downbutton"
    android:layout_alignParentBottom="true"
    android:layout_toEndOf="@+id/frameLayout" />
```

```
</RelativeLayout>
```

```
<android.support.design.widget.NavigationView
    android:id="@+id/navigation"
    android:background="#046069"
    android:layout_width="150dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    app:menu="@menu/menu_navigation" />
```

```
</android.support.v4.widget.DrawerLayout>
```

7.4 ο κώδικας της εφαρμογής, Java

```
package com.example.lavdas.fship;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.design.widget.NavigationView;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {
    private NavigationView navigationView;
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle = "test1";
    private CharSequence mTitle = "test 2";

    ImageButton conbutton;
    SeekBar seekBar;
    SeekBar verticalsb;
    TextView degree;
    TextView speed;
    TextView autopilottext;
    ImageButton light;
    ImageView imagelight;
    ImageButton autopilot;
    ImageButton hookbutton;
    ImageView hookview;
    ProgressBar batteryBar;
    ImageButton upspeed;
    ImageButton downspeed;
    ImageButton Backward;
    TextView Backwardtxt;
    TextView Esctemp1;
    ProgressBar Esctemp;
```

```

ProgressBar Signal;
public static double latf, longf;

boolean trueconnection = false;
boolean checkBackward = false;

int x;
int data;
private static final String TAG = "Jon";
private OutputStream outputStream = null;
InputStream mmInputStream = null;
byte[] readBuffer;
int readBufferPosition;
Thread workerThread;
private BluetoothAdapter bluetoothAdapter = null;
String address = "20:16:08:22:13:24";
private ProgressDialog progress;
BluetoothSocket btSocket = null;
volatile boolean stopWorker;
private boolean isBtConnected = false;
static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

int datasend;

@Override
protected void onCreate(final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mTitle = mDrawerTitle = getTitle();
    navigationView = (NavigationView) findViewById(R.id.navigation);
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
    mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
R.string.drawer_open, R.string.drawer_close) {

        /** Called when a drawer has settled in a completely closed state. */
        public void onDrawerClosed(View view) {
            super.onDrawerClosed(view);
            invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
        }

        /** Called when a drawer has settled in a completely open state. */
        public void onDrawerOpened(View drawerView) {
            super.onDrawerOpened(drawerView);
            invalidateOptionsMenu(); // creates call to
onPrepareOptionsMenu()
        }
    };

    // Set the drawer toggle as the DrawerListener
    mDrawerLayout.setDrawerListener(mDrawerToggle);

navigationView.setNavigationItemSelectedListener(onNavigationItemSelectedListener
);

    seekBar = (SeekBar) findViewById(R.id.seekBar2);
    verticalsb = (SeekBar) findViewById(R.id.seekBar1);
    batteryBar = (ProgressBar) findViewById(R.id.progressBarBattery);
    degree = (TextView) findViewById(R.id.textView);

```

```

speed = (TextView) findViewById(R.id.textView2);
conbutton = (ImageButton) findViewById(R.id.connectbutton);
imagelight = (ImageView) findViewById(R.id.imagelig);
light = (ImageButton) findViewById(R.id.light);
autopilot = (ImageButton) findViewById(R.id.pilotbtn);
hookbutton = (ImageButton) findViewById(R.id.hook);
hookview = (ImageView) findViewById(R.id.hookview);
imagelight.setImageResource(R.drawable.lig);
hookview.setImageResource(R.drawable.hookview);
autopilottext = (TextView) findViewById(R.id.pilottext1);
imagelight.setTag(true);
hookview.setTag(true);
autopilottext.setTag(true);
upspeed = (ImageButton) findViewById(R.id.upbutton);
downspeed = (ImageButton) findViewById(R.id.downbutton);
Backward = (ImageButton) findViewById(R.id.backward);
Backwardtxt = (TextView) findViewById(R.id.backwardtxt);
Backwardtxt.setTag(true);
Esctempl = (TextView) findViewById(R.id.esctempl);
Esctemp = (ProgressBar) findViewById(R.id.esctemp);
Signal = (ProgressBar) findViewById(R.id.signal);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        x = progress - 35;
        degree.setText(x + " Degree");
        datasend = 135 + x;
        try {
            Log.v("rudder", "degree="+ datasend);
            btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(datasend).getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
        }
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        Log.v("APP", "wrgwrehe stop");
        seekBar.setProgress(35);
        try {
            btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(135).getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
        }
    }
}

```

```

    });

    Backward.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Animation anim = new AlphaAnimation(0.0f, 1.0f);
            if ((Boolean) Backwardtxt.getTag()) {
                Backwardtxt.setText("Backward");
                checkBacward = true;
                anim.setDuration(50); //You can manage the blinking time with
this parameter

                anim.setStartOffset(20);
                anim.setRepeatMode(Animation.REVERSE);
                anim.setRepeatCount(Animation.INFINITE);
                Backwardtxt.startAnimation(anim);
                Backwardtxt.setTag(false);
                if (verticalsb.getProgress() >= 1) {
                    verticalsb.setProgress(0);
                    speed.setText(String.valueOf(verticalsb.getProgress()) +
"/30 SPEED");
                }

            } else {

                if (verticalsb.getProgress() >= 1) {
                    verticalsb.setProgress(0);
                    speed.setText(String.valueOf(verticalsb.getProgress()) +
"/30 SPEED");
                }
                Backwardtxt.setText("");
                Backwardtxt.setTag(true);
                checkBacward = false;
            }
        }
    });

    conbutton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

            bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
            if (bluetoothAdapter == null) {
                //Show a mensag. that thedevice has no bluetooth adapter
                Toast.makeText(getApplicationContext(), "Bluetooth Device Not
Available", Toast.LENGTH_LONG).show();
                //finish apk
                finish();
            } else {
                if (bluetoothAdapter.isEnabled()) {

                    ConnectBT connectBT = new ConnectBT();
                    connectBT.execute();
                } else {
                    //Ask to the user turn the bluetooth on

```

```

        Intent turnBTon = new
Intent (BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon, 1);
    }
}
});

light.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if ((Boolean) imagelight.getTag()) {

imagelight.animate().alphaBy(0f).alpha(1f).setDuration(400).start();
            imagelight.setTag(false);
            try {
                btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(400).getBytes());
            } catch (IOException e) {
                e.printStackTrace();

            } catch (NullPointerException e) {
                Toast.makeText(getApplicationContext(), "check bluetooth
conection", Toast.LENGTH_SHORT).show();
            }
        } else {

imagelight.animate().alphaBy(1f).alpha(0f).setDuration(401).start();
            imagelight.setTag(true);
            try {
                btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(401).getBytes());
            } catch (IOException e) {
                e.printStackTrace();
            } catch (NullPointerException e) {
            }
        }
        //turnOnLed();
    }
});

hookbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if ((Boolean) hookview.getTag()) {

hookview.animate().alphaBy(0f).alpha(1f).setDuration(400).start();
            hookview.setTag(false);
            try {
                btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(300).getBytes());
            } catch (IOException e) {
                e.printStackTrace();
            } catch (NullPointerException e) {
                Toast.makeText(getApplicationContext(), "check bluetooth

```

```

connection", Toast.LENGTH_SHORT).show();

        }
    } else {

hookview.animate().alphaBy(1f).alpha(0f).setDuration(400).start();
        hookview.setTag(true);
        try {
            btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(301).getBytes());
            } catch (IOException e) {
                e.printStackTrace();
            } catch (NullPointerException e) {
            }
        }
    }
});

    autopilot.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            if ((Boolean) autopilottext.getTag()) {

autopilottext.animate().alphaBy(0f).alpha(1f).setDuration(400).start();
                autopilottext.setTag(false);
                try {
                    btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(500).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (NullPointerException e) {
                    Toast.makeText(getApplicationContext(), "check bluetooth
connection", Toast.LENGTH_SHORT).show();
                }
            } else {

autopilottext.animate().alphaBy(1f).alpha(0f).setDuration(400).start();
                autopilottext.setTag(true);
                try {
                    btSocket.getOutputStream().write(String.valueOf("
").getBytes());

btSocket.getOutputStream().write(String.valueOf(501).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (NullPointerException e) {
                }
            }
        }
    });

    upspeed.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

```

```

        if (verticalsb.getProgress() <= 20) {
            verticalsb.setProgress(verticalsb.getProgress() + 1);
            speed.setText(String.valueOf(verticalsb.getProgress()) + "/30
SPEED");
        }
    });

    downspeed.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (verticalsb.getProgress() >= 1) {
                verticalsb.setProgress(verticalsb.getProgress() - 1);
                speed.setText(String.valueOf(verticalsb.getProgress()) + "/30
SPEED");
            }
        }
    });

    verticalsb.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
            speed.setText(String.valueOf(progress) + "/30 SPEED");

            if (checkBackward == false) {
                data = progress + 1000;
                try {
                    btSocket.getOutputStream().write(String.valueOf("
").getBytes());

                    btSocket.getOutputStream().write(String.valueOf(data).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (NullPointerException e) {
                }
            } else if (checkBackward == true) {
                data = progress + 1200;
                try {
                    btSocket.getOutputStream().write(String.valueOf("
").getBytes());

                    btSocket.getOutputStream().write(String.valueOf(data).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (NullPointerException e) {
                }
            }
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {

        }
    });

```

```

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    NavigationView.OnNavigationItemSelectedListener
    onNavigationItemSelectedListener = new
    NavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelectedListener(MenuItem item) {
            switch (item.getItemId()) {
                // Respond to the action bar's Up/Home button
                case R.id.emergencystop:
                    verticalsb.setProgress(0);
                    try {
                        btSocket.getOutputStream().write(String.valueOf("
").getBytes());
                    btSocket.getOutputStream().write(String.valueOf(700).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                }
                } catch (NullPointerException e) {
                    Toast.makeText(getApplicationContext(), "check bluetooth
conection", Toast.LENGTH_SHORT).show();
                }
                Log.v("NavigationView", "Emergencystop");
                mDrawerToggle.onDrawerClosed(navigationView);
                return true;
            case R.id.Maps:
                try {
                    btSocket.getOutputStream().write(String.valueOf("
").getBytes());
                    btSocket.getOutputStream().write(String.valueOf(800).getBytes());
                } catch (IOException e) {
                    e.printStackTrace();
                }
                } catch (NullPointerException e) {
                    Toast.makeText(getApplicationContext(), "check bluetooth
conection", Toast.LENGTH_SHORT).show();
                }
                Intent intent = new Intent(getApplicationContext(),
MapsActivity.class);
                startActivityForResult(intent, 1);
                mDrawerToggle.onDrawerClosed(navigationView);
            }
        }
    }

```

```

        Log.v("NavigationView", "Openmaps");
        return true;

        case R.id.exit:

            finish();
            System.exit(0);
            mDrawerToggle.onDrawerClosed(navigationView);
            Log.v("NavigationView", "Exit");
            return true;

        default:
            return true;
    }
}

@Override
public void onStart() {
    super.onStart();
}

@Override
public void onStop() {
    super.onStop();
}

private class ConnectBT extends AsyncTask<Void, Void, Void> // UI thread
{
    private boolean ConnectSuccess = true; //if it's here, it's almost
connected

    @Override
    protected void onPreExecute() {
        progress = ProgressDialog.show(MainActivity.this, "Connecting...",
        "Please wait!!!"); //show a progress dialog
    }

    @Override
    protected Void doInBackground(Void... devices) //while the progress
dialog is shown, the connection is done in background
    {
        try {
            if (btSocket == null || !isBtConnected) {
                BluetoothDevice dispositivo =
                bluetoothAdapter.getRemoteDevice(address); //connects to the device's address and
                checks if it's available
                bluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                btSocket =
                dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID); //create a RFCOMM
                (SPP) connection
                btSocket.connect(); //start connection
            }
        } catch (IOException e) {

```

```

        ConnectSuccess = false; //if the try failed, you can check the
exception here
    }
    return null;
}

@Override
protected void onPostExecute(Void result) //after the doInBackground, it
checks if everything went fine
{
    super.onPostExecute(result);

    if (!ConnectSuccess) {
        msg("Connection Failed. Is it a SPP Bluetooth? Try again.");
        finish();
    } else {
        msg("Connected.");

        isBtConnected = true;
        trueconnection = true;

        Log.v("APP", "dujdtjtykrtkuytykb Connected");

        try {

            btSocket.getOutputStream().write(String.valueOf("
).getBytes());

btSocket.getOutputStream().write(String.valueOf(601).getBytes());
            mmInputStream = btSocket.getInputStream();

            beginListenForData();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    progress.dismiss();
}

void beginListenForData() {

    final Handler handler = new Handler();
    final byte delimiter = 10; //This is the ASCII code for a newline
character

    stopWorker = false;
    readBufferPosition = 0;
    readBuffer = new byte[1024];
    workerThread = new Thread(new Runnable() {
        public void run() {
            while (!Thread.currentThread().isInterrupted() && !stopWorker) {
                try {

                    int bytesAvailable = mmInputStream.available();
                    if (bytesAvailable > 0) {
                        byte[] packetBytes = new byte[bytesAvailable];
                        mmInputStream.read(packetBytes);
                        for (int i = 0; i < bytesAvailable; i++) {

```

```

        final byte b = packetBytes[i];
        if (b == delimiter) {
            byte[] encodedBytes = new
byte[readBufferPosition];
            System.arraycopy(readBuffer, 0, encodedBytes,
0, encodedBytes.length);
            final String data = new String(encodedBytes,
"US-ASCII");
            readBufferPosition = 0;
            handler.post(new Runnable() {
                public void run() {
                    Log.v("APP", "Receiver data: " +
data);
                    try {
                        if (data.contains("#")) {
                            int myNum =
Integer.parseInt(data.replaceAll("#", "").replaceAll("\\D+", ""));
                            batteryBar.setProgress(myNum);
                        } else if (data.contains("$")) {
                            int myNum =
Integer.parseInt(data.replaceAll("$", "").replaceAll("\\D+", ""));
                            Esctemp.setProgress(myNum);
                        } else if (data.contains("&")) {
                            int myNum =
Integer.parseInt(data.replaceAll("&", "").replaceAll("\\D+", ""));
                            Signal.setProgress(myNum);
                        } else if (data.contains("%")) {
                            //lat
                            latf=Double.parseDouble(data.replaceAll("%", "").replaceAll("\\\\+", ""));
                        } else if (data.contains("@")) {
                            //long
                            longf=Double.parseDouble(data.replaceAll("@", "").replaceAll("\\\\+", ""));
                        }
                    }
                    LocationEvent.getInstance().post(latf, longf);
                    Log.v("APP", "1Receiver data: " +
latf + " " + longf);
                } catch (NumberFormatException nfe) {
                    //Toast.makeText(MainActivity.this,data,Toast.LENGTH_LONG).show();
                }
            });
        } else {
            readBuffer[readBufferPosition++] = b;
        }
    }
} catch (IOException ex) {

```


8. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. «Ο ΜΙΚΡΟΕΛΕΓΚΤΗΣ 8051» ΒΙΒΛΙΟ ΧΡΗΣΤΟΥ ΚΑΡΑΪΣΚΟΥ
2. «ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ» ΒΙΒΛΙΟ ΔΡ.ΔΗΜΗΤΡΙΟΥ Ι.ΤΣΕΛΕ
3. http://www.eln.teilam.gr/sites/default/files/lesson06_0.pdf
4. <http://www.eln.teilam.gr/sites/default/files/Protocola%20Diasyndeshs.pdf>
5. <https://el.wikipedia.org/wiki/Θερμίστορ>
6. <http://www.mechatronics.gr/el/rd/12-pid-controller.html>
7. <https://www.digitaltrends.com/mobile/how-does-bluetooth-work/>
8. [https://el.wikipedia.org/wiki/Android Studio](https://el.wikipedia.org/wiki/Android_Studio)
9. <https://el.wikipedia.org/wiki/Android>