ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΘΕΜΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

" ΑΥΤΟΜΑΤΟΣ ΠΙΛΟΤΟΣ ΠΛΩΤΟΥ ΣΚΑΦΟΥΣ"

ΟΝΟΜΑΤΑ ΦΟΙΤΗΤΩΝ:

ΑΓΡΙΜΗΣ ΕΥΑΓΓΕΛΟΣ

ΠΑΤΣΟΥΡΗΣ ΓΕΩΡΓΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΠΑΠΟΥΤΣΙΔΑΚΗΣ ΜΙΧΑΛΗΣ

ΑΙΓΑΛΕΩ, ΜΑΪΟΣ 2017

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο / Η κάτωθι υπογεγραμμένος / η ΑΓΡΙΜΗΣ ΕΥΑΓΓΕΛΟΣ,

του ΠΑΡΑΣΚΕΥΑ, με αριθμό μητρώου 39644 φοιτητής / τρια του Τμήματος **Μηχανικών Αυτοματισμού Τ.Ε.** του Α.Ε.Ι. Πειραιά Τ.Τ. 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 απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ο Δηλών

Ημερομηνία
21/05/17

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο / Η κάτωθι υπογεγραμμένος / η ΠΑΤΣΟΥΡΗΣ ΓΕΩΡΓΙΟΣ,

του ΑΝΑΣΤΑΣΙΟΥ, με αριθμό μητρώου 39394 φοιτητής / τρια του Τμήματος **Μηχανικών Αυτοματισμού Τ.Ε.** του Α.Ε.Ι. Πειραιά Τ.Τ. 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 απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ο Δηλών

Ημερομηνία

21/05/17

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της πτυχιακής μας εργασίας θα θέλαμε να εκφράσουμε τις θερμές μας ευχαριστίες στον επιβλέποντα Επίκουρο Καθηγητή Παπουτσιδάκη Μιχάλη όχι μόνο για τη δυνατότητα που μας έδωσε να πραγματοποιήσουμε την πτυχιακή μας εργασία υπό τις οδηγίες του αλλά και για όλες τις συμβουλές και τα εφόδια που μας προσέφερε καθ' όλη τη διάρκεια των σπουδών μας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΥΜΒΟΛΩΝ & ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

GPS	Global Positioning System
HDS	Hitachi Data Systems
NMEA	National Marine Electronics Association
I^2C	Inter - Integrated Circuit
LCD	Liquid Crystal Display
TTL	Transistor - Transistor Logic
UART	Universal Asynchronous Receiver Transmitter
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
DBT	Depth Below Transducer
SDA	Serial Data
SCL	Serial Clock
ACK	Acknowledged
NACK	Not – Acknowledged
DCM	Direction Cosine Matrix
IMU	Intertial Measurement Unitd

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία έχουμε επιχειρήσει να υλοποιήσουμε ένα ολοκληρωμένο αυτόματο σύστημα πλοήγησης πλωτού σκάφους. Με τα μέσα που είχαμε στη διάθεσή μας κατορθώσαμε να συνθέσουμε και να προγραμματίσουμε έναν πραγματικό αυτόματο πιλότο. Στις παρακάτω σελίδες θα αναφερθούμε αναλυτικά τόσο στα μέσα και υλικά χρησιμοποιήσαμε προκειμένου να ολοκληρώσουμε την κατασκευή, όσο και στον κώδικά που συγγράψαμε.

Αναλογιζόμενοι τις ανάγκες της σύγχρονης ναυσιπλοΐας, αποφασίσαμε να ασχοληθούμε με το συγκεκριμένο θέμα για δύο λόγους: Πρώτον, διότι αποτελεί μια έμπρακτη εφαρμογή των γνώσεων που αποκτήσαμε κατά τη διάρκεια της φοίτησής μας και δεύτερον διότι καταδεικνύει την αναγκαιότητα της ειδικότητας και του γνωστικού μας αντικειμένου ως μηχανικοί αυτοματισμού στην σύγχρονη αγορά εργασίας.

Η εργασία, όπως φαίνεται και στα περιεχόμενα χωρίζεται σε τρία κυρίως μέρη. Στο πρώτο μέρος παρουσιάζουμε την ιστορική αναδρομή του αντικειμένου της εργασίας όπως και αναλυτική περιγραφή για κάθε υλικό που χρησιμοποιήθηκε. Ονοματίζουμε και περιγράφουμε τα τεχνικά χαρακτηριστικά του κάθε υλικού ξεχωριστά.

Στο δεύτερο μέρος ασχολούμαστε με το πρακτικό μέρος της διάταξης μας. Προσπαθούμε να δείξουμε, με χρήση εικόνων και διαγραμμάτων, την κατασκευή μας όπως αυτή εξελίχθηκε μέχρι την ολοκλήρωσή της.

Στο τρίτο μέρος παραθέτουμε τον κώδικα και τον αναλύουμε όσο το δυνατόν περισσότερο με επεξηγήσεις και σχόλια, ώστε η ανάγνωσή του να είναι ευκολότερη και συντομότερη.

ABSTRACT

In the present diploma thesis we have attempted to materialize a complete automatic pilot (autopilot) for floating vessels. With the means we had available we managed to compose and program a real autopilot. In the following pages we are going to analyze the means and materials we used in order to complete the autopilot as well as the code behind it.

Thinking of the needs of the modern navigation we decided to deal with this topic for two main reasons: Firstly, because it is an application of the knowledge we obtained during our studies and secondly because it points out that our area of expertise as automation engineers is greatly needed in today's employment market.

The paper is divided in three parts. In the first part, we present the history behind navigation systems as well as pictures and analytical descriptions for every material we used. We name and describe the technical characteristics of each material separately.

In the second part we emphasize on the practical part of the formation. We try to demonstrate, using pictures and diagrams, our structure as it was evolving until it was finally completed.

In the third part we state our code and analyze the it as much as possible with comments, so its reading is easier and faster.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Ιστορική Αναδρομή

Από τη στιγμή που ο άνθρωπος μετακινήθηκε για πρώτη φορά στις θάλασσες δημιουργήθηκε και η ανάγκη επινόησης μεθόδων που θα του επέτρεπαν την ασφαλή του πλοήγηση. Ασφαλώς σε πρώιμο στάδιο, ο άνθρωπος δεν επεδίωξε μεγάλες διαδρομές. Περιορίστηκε σε μετακινήσεις από ακτή σε ακτή έτσι ώστε να χρησιμοποιεί για οδηγό του «σημεία», που ο ίδιος όριζε πάνω στις ακτές. Μετέπειτα όμως, όταν επειχείρησε να μετακινηθεί πιο ανοικτά από τις ακτές αντιλήφθηκε την ανάγκη επινόησης νέων μεθόδων προσδιορισμού κατευθύνσεων, μέτρησης αποστάσεων και εύρεσης της θέσης του προκειμένου να προγραμματίζει τις επόμενες κινήσεις του για να φτάσει στον επιθυμητό προορισμό. Παρατήρησε πως χρησιμοποιώντας τα ουράνια σώματα μπορούσε πλέον να καθορίσει την πορεία του χωρίς να έχει ορατή επαφή με τη στεριά.[1] Με τη χρήση κλεψύδρας κατόρθωσε να υπολογίζει την απόσταση που διένυε βάσει του χρόνου και την ταχύτητα του πλοίου ανάλογα με τους κτύπους των κουπιών. Εν ολίγοις ο άνθρωπος κατόρθωσε να βρει τρόπους να υπολογίζει τρεις βασικούς παράγοντες που επηρεάζουν την πλοήγηση, δηλαδή τη θέση, την απόσταση και την ταχύτητα. [2]

Μέσα στους επόμενους αιώνες πραγματοποιήθηκαν άλματα στην εξέλιξη των μέσων πλοήγησης. Πλεόν, χρησιμοποιείται η πυξίδα, ο αστρολάβος, που όπως υποδηλώνει το όνομά του χρησιμοποιούνταν για την παρατήρηση του ήλιου και των άστρων. Γινόταν χρήση του εξάντα και αργότερα του χρονομέτρου.

Η πραγματική επανάσταση όμως στον τομέα της πλοήγησης πραγματοποιήθηκε μέσα στον 20^0 αιώνα. Με την παράλληλη εξέλιξη και των υπολοίπων επιστημών έχουμε πλέον στα χέρια μας το γυροσκόπιο, το ραντάρ και κυρίως το ευρέως γνωστό GPS (Global Positioning System). Παρακάτω αναφερόμαστε αναλυτικότερα στα μέσα που χρησιμοποιήσαμε για την υλοποίηση της κατασκευής.[3][4]

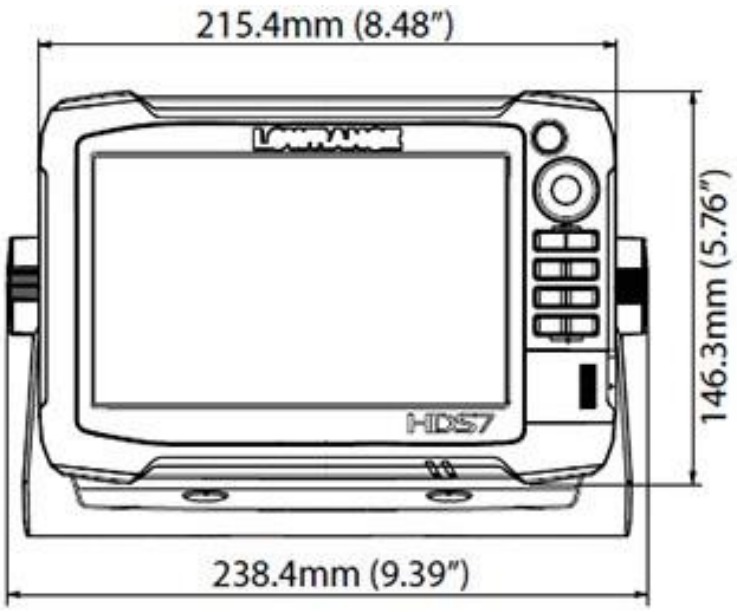
1.2 Τεχνικός Εξοπλισμός

1.2.1 GPS (Global Positioning System)

Το GPS λειτουργεί εκμεταλλευόμενο τους δορυφόρους που κινούνται σε συγκεκριμένη τροχιά γύρω από τη Γη. Οι δορυφόροι αυτοί εκπέμπουν ηλεκτρομαγνητικά σήματα προς τη Γη, η απαραίτητη πληροφορία αυτών των σημάτων είναι η θέση στην οποία βρίσκονται κάθε φορά.

Η τελευταία πληροφορία είναι σημαντική καθώς ο δέκτης διαθέτει ενσωματωμένο ρολόι και έτσι ξέρει ακριβώς τι ώρα έλαβε το σήμα. Χρησιμοποιώντας αυτά τα χρονισμένα σήματα και δεδομένου ότι το σήμα έχει μια συγκεκριμένη ταχύτητα υπολογίζεται η απόσταση του δέκτη από τον κάθε δορυφόρο που εντοπίζει.

Έτσι λοιπόν γνωρίζοντας την ακριβή θέση των δορυφόρων και την απόσταση που μεσολαβεί, ο δέκτης υπολογίζει τη θέση στην οποία βρισκόμαστε κάθε φορά.[9]



Εικόνα 1.1 Lowrance HDS-7 GEN3

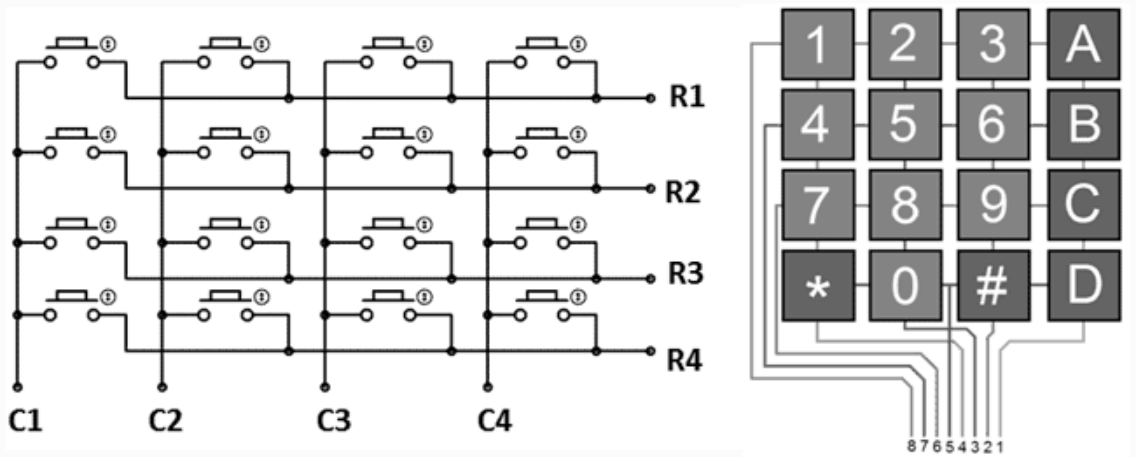
Στην εργασία μας χρησιμοποιήσαμε το LOWRANCE HDS-7 GEN3.Το συγκεκριμένο όργανο χρησιμοποιείται ως Fishfinder (βυθόμετρο) και ως GPS Plotter για την καθοδήγηση στη θάλασσα.

Στον παρακάτω πίνακα φαίνονται τα τεχνικά του χαρακτηριστικά :

Πίνακας 1.1 Τεχνικά Χαρακτηριστικά GPS

Display	7 in./178 mm (16:9) Widescreen
Resolution	WVGA color TFT LCD 800 x 480
Backlighting	LED With Adjustable Display and Keypad
NMEA output	NMEA 0183 and NMEA 2000®
Supply Voltage	12 vDC (10-17 vDC min-max)
GPS Antenna Type	Internal 10Hz ultra-high-sensitivity WAAS + EGNOS + MSAS

1.2.2 Matrix Keypad



Εικόνα 1.2 Διάταξη Matrix Keypad

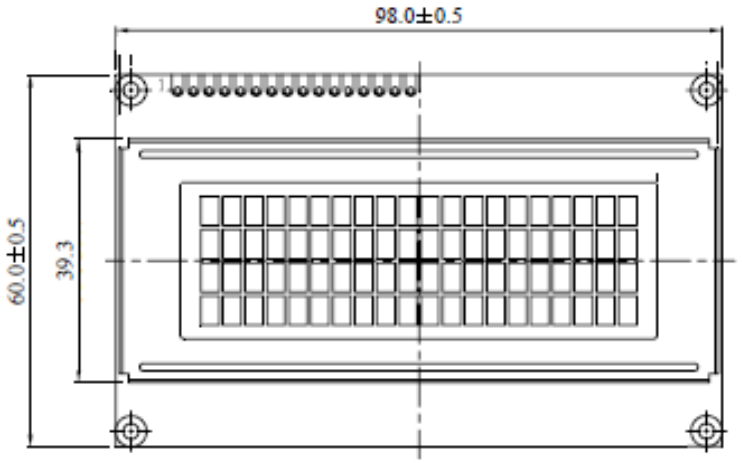
Χρησιμοποιήσαμε ένα πληκτρολόγιο 4x4 (4 γραμμές-4 στήλες) για την καταχώριση της επιθυμητής κατεύθυνσης και την ακύρωση ειδοποιήσεων σφάλματος από τον χρήστη.

Το πληκτρολόγιο έχει τα εξής χαρακτηριστικά :

Πίνακας 1.2 Τεχνικά Χαρακτηριστικά Keypad

Maximum Rating	24 VDC, 30 mA
Interface	8-pin access to 4x4 matrix
Operating temperature	32 to 122 °F (0 to 50°C)
Dimensions: Keypad	2.7 x 3.0 in (6.9 x 7.6 cm)
Cable	0.78 x 3.5 in (2.0 x 8.8 cm)

1.2.3 Οθόνη Χαρακτήρων



PIN NO.	SYMBOL
1	Vss
2	Vdd
3	Vo
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	Vss
16	Vdd

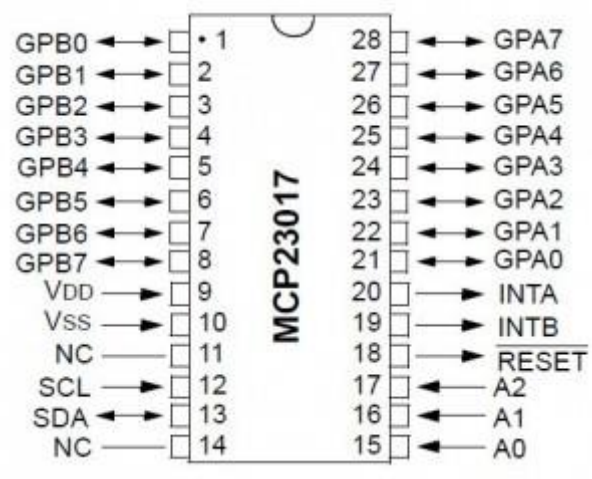
Εικόνα 1.3 Διάταξη Οθόνης Χαρακτήρων

Χρησιμοποιήσαμε μια οθόνη υγρών κρυστάλλων (LCD) 4x20 για την προβολή των απαραιτήτων πληροφοριών προς τον χρήστη όπως η κατάσταση λειτουργίας (Mode) ,την τρέχουσα διεύθυνση (Heading) ή πορεία (Course) (ανάλογα αν ο τρόπος λειτουργίας που έχει επιλέξει ο χρήστης είναι η πυξίδα ή το plotter), την επιθυμητή που έχει καταχωρήσει ο χρήστης αλλά και μηνύματα διευκόλυνσης του χειρισμού του αυτόματου πιλότου.

Πίνακας 1.3 Τεχνικά Χαρακτηριστικά Οθόνης

Dots per Character	5 x 8
Character Width	3 mm
Number of Rows	4
Characters per Row	20
Backlight Current Consumption	Max 240 mA

1.2.4 MCP23017 I/O expander



Εικόνα 1.4 Σχηματικό διάγραμμα των ακροδεκτών του MCP23017

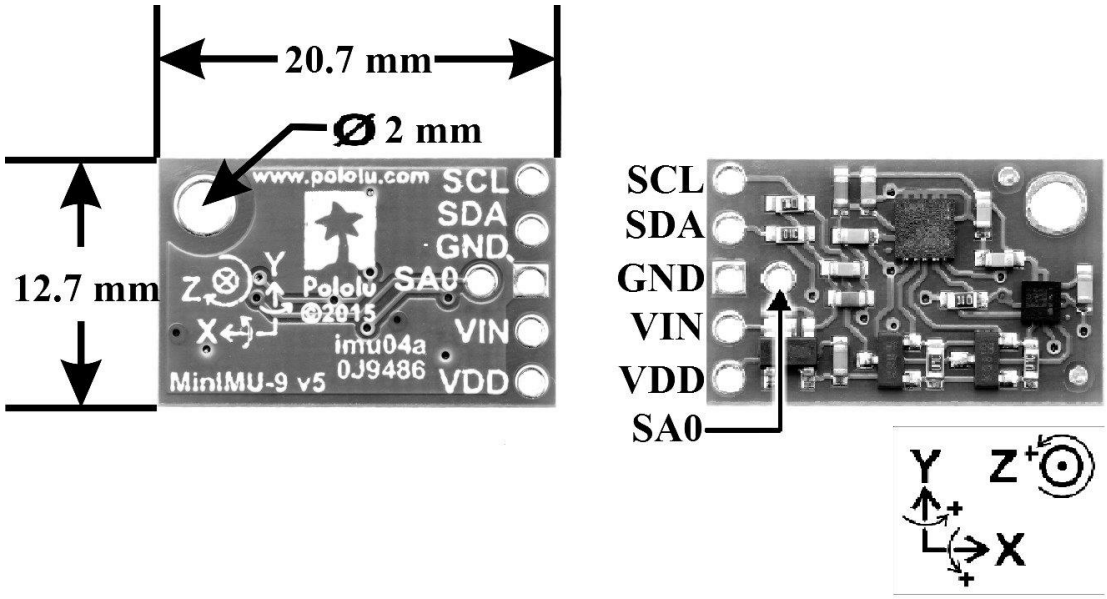
Το ολοκληρωμένο MCP23017 παρέχει επέκταση των ψηφιακών εισόδων-εξοδων I/O του μικροελεγκτή με τον οποίο είναι συνδεδεμένο και αποτελείται από δύο πόρτες των 8 bit (GPA και GPB), επικοινωνεί με τον μικροελεγκτή μέσω του πρωτοκόλλου I^2C.

Παρακάτω τα βασικά τεχνικά χαρακτηριστικά :

Πίνακας 1.4 Τεχνικά Χαρακτηριστικά Chip MCP23017 I/O expander

Operating Voltage Range (V)	1.8 to 5.5
Operating Temp Range (°C)	-40 to 125
Interface	I^2C
Max. Bus Frequency (kBits/s)	1700

1.2.5 Pololu MinIMU-9 v5 Gyroscope, Accelerometer and Magnetometer (Compass) Board



Εικόνα 1.5 Pololu MinIMU-9 v5 Board

Το MinIMU-9 περιλαμβάνει δύο ολοκληρωμένα, το LSM6DS33 που αποτελείται από δυο αισθητήρια ένα γυροσκόπιο και ένα επιταχυνσιόμετρο τριών αξόνων και το LIS3MDL που αποτελείται από ένα μαγνητόμετρο τριών αξόνων.Το LSM6DS33 έχει ως ρόλο τον καθορισμό της κλίσης για τους άξονες x,y,z. Το LIS3MDL μετρά το μαγνητικό πεδίο της γης για τους τρεις άξονες.

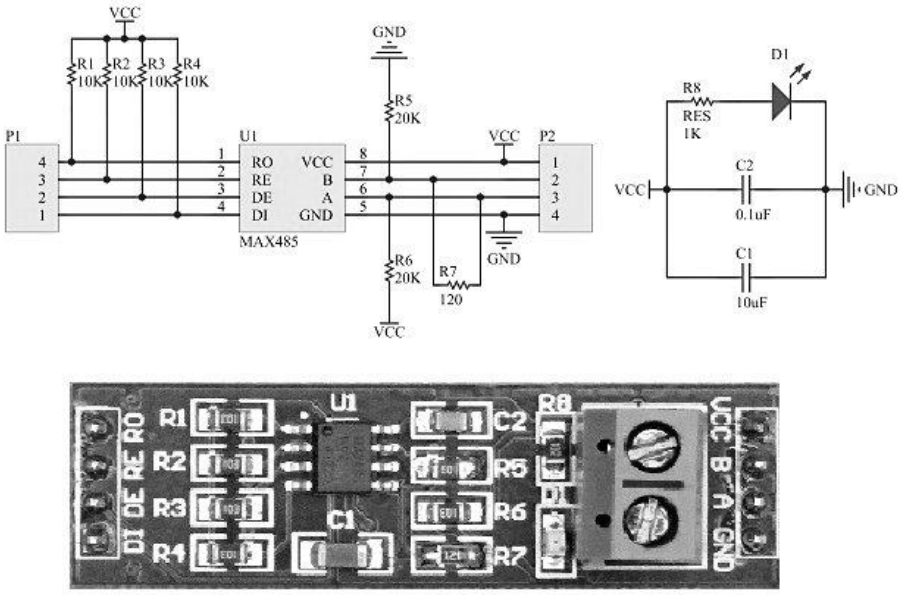
Παρακάτω τα βασικά τεχνικά χαρακτηριστικά :

Πίνακας 1.5 Τεχνικά Χαρακτηριστικά MinIMU-9

Operating voltage	2.5 V to 5.5 V
Supply current	5mA
Interface	I²C

Gyro sensitivity range	±125, ±245, ±500, ±1000, or ±2000°/s
Accelerometer sensitivity range	±2, ±4, ±8, or ±16 g
Magnetometer sensitivity range	±4, ±8, ±12, or ±16 gauss

1.2.6 Transceiver MAX485-RS485 (receiver-transmeter)



Εικόνα 1.6 Transceiver MAX485-RS485

Το RS485 είναι ένα πρότυπο σειριακής επικοινωνίας που έχει ως σκοπό τον καθορισμό των ηλεκτρικών σημάτων του φυσικού επιπέδου με δύο αγωγούς που επιτυγχάνουν μονόδρομη, πολλαπλών σημείων σειριακή διασύνδεση (ένας πομπός πολλοί δέκτες).

Προσφέρει ταχύτητα μετάδοσης δεδομένων 35 Mbit/s για αποστάσεις έως και 10 μέτρα και 100 kbit/s για αποστάσεις έως και 1200 m. Χρησιμοποιώντας διαφορική ισορροπημένη γραμμή καλωδίων συστρεφόμενου ζεύγους (twisted pair) μπορεί να καλύψει αποστάσεις έως και 1200 μέτρα.

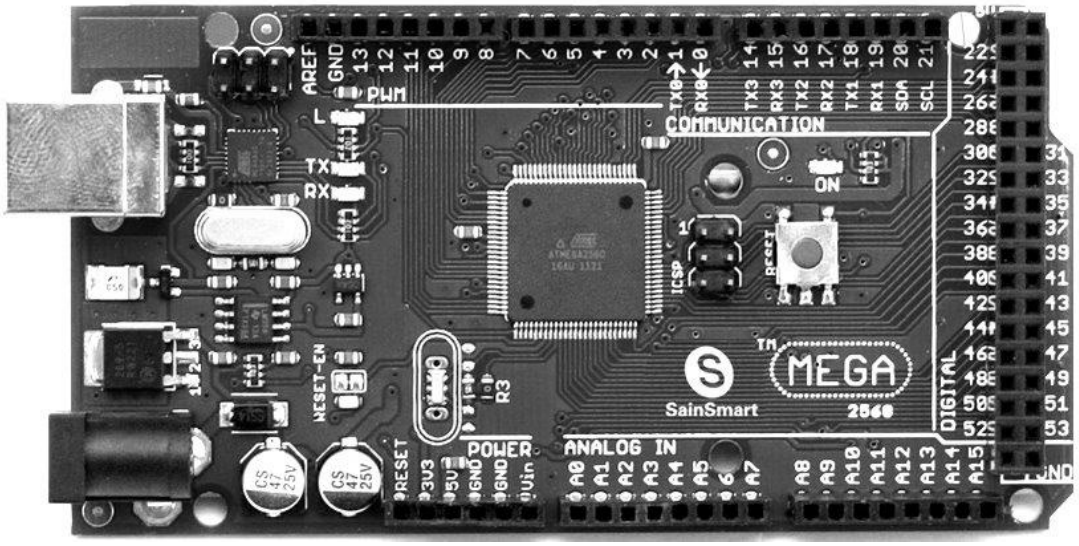
Ο λόγος που χρησιμοποιήθηκε στην εφαρμογή μας είναι πως η επικοινωνία με το πλότερ δεν μπορεί να επιτευχτεί με απευθείας σύνδεση τον μικροελεγκτή καθώς ο μικροελεγκτής μπορεί να δεχτεί μόνο TTL σήματα (έως 5 V), ενώ η σειριακή έξοδος του πλότερ είναι στα 12V.

Ακολουθούν τα τεχνικά χαρακτηριστικά :

Πίνακας 1.6 Τεχνικά Χαρακτηριστικά Transceiver RS485 (receiver-transmeter)

Power module	+5V
Module size	44 (mm) x14 (mm)
Data rate	2500 kbps
Rx/Tx on bus	32
Duplex	Half

1.2.7 Arduino Mega



Εικόνα 1.7 Arduino Mega Board

Το Arduino Mega είναι ένας μικροελεγκτής με 54 ψηφιακές εισόδους/εξόδους, 16 αναλογικές εξόδους, 4 UART (σειριακές θύρες hardware), έναν ταλαντωτή 16 MHz και σύνδεση USB. Παρακάτω τα τεχνικά χαρακτηριστικά του :

Πίνακας 1.7 Τεχνικά Χαρακτηριστικά Arduino Mega

Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
DC Current per I/O Pin	20 m
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB (of which 8 KB used by bootloader)
SRAM	8 KB
EEPROM	4 KB

Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Όσον αφορά τον υπόλοιπο τεχνικό εξοπλισμό χρησιμοποιήσαμε επίσης αντιστάσεις και ποτενσιόμετρα, κλαίμες, έναν διακόπτη (ON/OFF), ένα σερβοκινητήρα, καλώδια χαλκού 2 και 6 αγωγών , ένα buzzer για την ένδειξη σφάλματος και βέβαια τροφοδοτικό 12V για την προσομοίωση της συνηθισμένης τάσης τροφοδοσίας σε ένα σκάφος. Όλα τα παραπάνω απεικονίζονται αναλυτικά στα σχέδια των πλακετών που έχουμε κατασκευάσει.

1.3 Πρωτόκολλο NMEA 0183

Το πρωτόκολλο NMEA 0183 ή αλλιώς NMEA 0183 standard χρησιμοποιείται για την επικοινωνία μεταξύ ναυτικών ηλεκτρονικών συσκευών όπως τα σόναρ, ανεμόμετρα, γυροσκόπια, αυτόματοι πιλότοι, GPS και πολλών άλλων οργάνων.Τα δεδομένα που στέλνονται μέσω του NMEA μεταδίδονται από συσκευές όπως ένα GPS ή ένα γυροσκόπιο. Οι συσκευές αυτές, οι οποίες αποστέλλουν τα δεδομένα αποκαλούνται «talkers». Από την άλλη, οι συσκευές που παραλαμβάνουν τα δεδομένα πχ τα ραντάρ, ένας υπολογιστής ή η απεικόνιση του NMEA (NMEA display), αποκαλλούνται «listeners».Ένα bus δεδομένων NMEA 0183 έχει πάντα μόνο μια συσκευή talker αλλά μπορεί να έχει παραπάνω από μια συσκευές listeners.

Οι έξοδοι (outputs) των talkers δεν μπορούν να απενεργοποιηθούν ούτε και να μετατραπούν σε δέκτες, πράγμα που σημαίνει ότι οι έξοδοι λειτουργούν συνεχώς. Αυτός είναι και ο λόγος για τον οποίο μπορούμε να έχουμε μόνο μια συσκευή talker κάθε φορά σε ένα δίκτυο NMEA.

Όλα τα δεδομένα NMEA στέλνονται σε μορφή κειμένου και συγκεκριμένα σαν μια πρόταση.Κάθε μια πρόταση ξεκινάει είτε με ‘$’ είτε με ‘!’ και κάθε μέρος της πρότασης

χωρίζεται με κόμμα.Οι κώδικες NMEA είναι γραμμένοι σε ASCII και έχουν την ακόλουθη μορφή :

$yyXXX,…… *xx <0D><0A>

Η παραπάνω πρόταση όπως προείπαμε ξεκινάει με '$'.Στη συνέχεια ακολουθεί ένας κώδικας δύο ψηφίων 'yy', ο οποίος μας υποδεικνύει τον τύπο της συσκευής.Για παράδειγμα εάν πρόκειται για GPS θα πρέπει να αναγράφεται το 'GP'. Στη συνέχεια ακολουθεί ένας κώδικας 3 ψηφίων 'XXX' ο οποίος φανερώνει τον τύπο δεδομένων της πρότασης. Για παράδειγμα εάν αναγράφεται το 'GGA' τότε πρόκειται για 'Global Positioning System Fix Data', δηλαδή για δεδομένα GPS ενώ αν αναγράφεται το 'DBT' πρόκειται για 'Depth Below Transducer', δηλαδή για δεδομένα βυθομέτρου.Στη συνέχεια ακολουθεί κόμμα και έπειτα το περιεχόμενο της πρότασης, το οποίο αλλάζει ανάλογα με τον τύπο δεδομένων και τις τιμές που έχει στην παρούσα φάση αυτό που παρακολουθούμε. Το τελευταίο κομμάτι είναι προαιρετικό και αφορά ένα διψήφιο checksum (έλεγχος αθροίσματος).Πριν το checksum προηγείται ο χαρακτήρας '*' και υπολογίζεται την 8-bit αποκλειστική OR όλων των χαρακτήρων της πρότασης συμπεριλαμβανομένων και των διαχωριστικών ',' και αποκλείοντας τους χαρακτήρες '$' / '!' και το διαχωριστικό '*'.Η πρόταση τελειώνει πάντα με την επιστροφή του κρατουμένου και τον συνδυασμό γραμμής τροφοδοσίας (Hex 0D 0A, ASCII '\r\n').[10]

1.4 Πρωτόκολλο I^2C

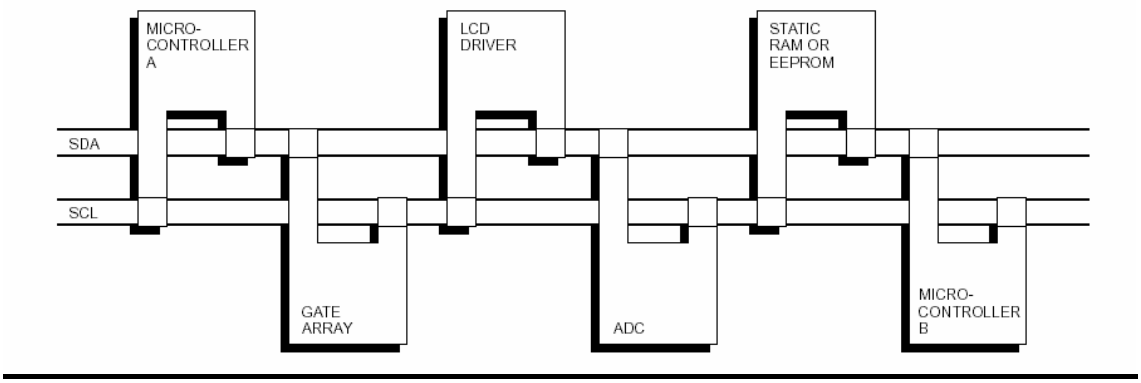
Το πρωτόκολλο I^2C (inter-integrated circuit) αναπτύχθηκε από την Philips με σκοπό τη σύνδεση περιφερειακών κυκλωμάτων στο Motherboard. Το I^2C χρησιμοποιείται και για την επικοινωνία συσκευών που συνδέονται με καλώδια. Οι περισσότερες εφαρμογές πρωτοκόλλου I^2C συναντώνται σε συσκευές modem,σε κινητά και συμβατικά τηλέφωνα, σε ATM's αλλά και στην επικοινωνία διάφορων αισθητήρων όπως πχ θερμοκρασίας,πίεσης κλπ με οθόνες στις οποίες στέλνουν τις μετρήσεις.

Πιο πρακτικά, στις περισσότερες ηλεκτρονικές συσκευές συναντάει κανείς έναν συγκεκριμένο τρόπο σχεδίασης. Αυτό σημαίνει ότι λογικά θα δούμε κάποιον

μικροεπεξεργαστή, μνήμες RAM, ρολόγια, μετατροπείς ψηφιακού σε αναλογικό (D/A) ή και αντίστροφα (A/D), EEPROMs (electrically erasable programmable read-only memory) κλπ.
Συνεπώς, αντιμετοπίζουμε το εξής πρόβλημα : πως θα κατορθώσουμε να επιτύχουμε την επικοινωνία όλων αυτών μεταξύ τους ; Η απάντηση σε αυτό το πρόβλημα είναι το I^2C.

1.4.1 Υλοποίηση ενός πρωτοκόλλου I^2C

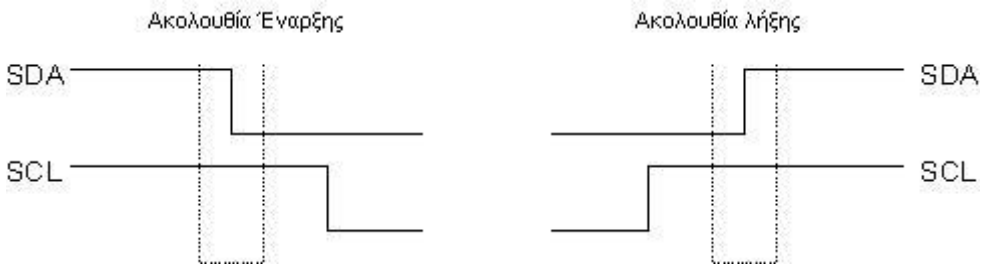
Προκειμένου να υλοποιήσουμε το I^2C χρησιμοποιούμε δύο καλώδια, το ένα είναι το SDA (Serial Data) για τη μεταφορά δεδομένων και το άλλο το SCL (Serial Clock) για το ρολόι.Βέβαια υπάρχει και ένα τρίτο καλώδιο, το οποίο είναι η γείωση. Κάθε μια συσκευή ή εξάρτημα που συνδέεται στον δίαυλο έχει μια δική της μοναδική κατεύθυνση. Οι σχέσεις μεταξύ των εξαρτημάτων χαρακτηρίζονται ως master-slave. Συνεπώς κάθε εξάρτημα συνδεδεμένο στον δίαυλο λειτουργεί είτε ως Master, άρα λαμβάνει τις αποφάσεις για την οποιαδήποτε λειτουργία που συντελείται πάνω στον δίαυλο, είτε ως Slave όπου σε αυτήν την περίπτωση «υπακούει» στις απαιτήσεις του Master.



Εικόνα 1.8 Παράδειγμα σύνδεσης συσκευών σε δίαυλο I^2C

Σε αυτό το σημείο πρέπει να αναφέρουμε πως το πρωτόκολλο I^2C υποστηρίζει δύο λειτουργίες : την αμφίδρομη και τη μονόδρομη μεταφορά δεδομένων. Παρόλα αυτά θα επικεντρωθούμε μόνο στην αμφίδρομη καθώς είναι αυτή που συναντάται συχνότερα στις καταναλωτικές συσκευές.
Σε μια αμφίδρομη λοιπόν μεταφορά δεδομένων μια συσκευή εκτός από το να λειτουργεί είτε ως Master είτε ως Slave μπορεί επίσης να λειτουργεί και σαν πομπός (transmitter) ή δέκτης

(receiver). Αυτό βέβαια έχει να κάνει με το αν αν στέλνει ή λαμβάνει δεδομένα. Όταν Master και Slave θέλουν να επικοινωνήσουν ξεκινά από την Master μια ακολουθία έναρξης ή αλλιώς start sequence. Αντίστοιχα υπάρχει και η ακολουθία λήξης (stop sequence). Η κανονική κατάσταση των δύο γραμμών είναι το λογικό 1 δηλαδή HIGH, αυτό σημαίνει πως στην προκειμένη χρονική στιγμή είναι ελεύθερες. Όταν εκπέμπεται παλμός από κάποια συσκευή συνδεδεμένη σε SCL ή σε SDA τότε το λογικό 1 γίνεται 0 και η κατάσταση από HIGH σε LOW.[11]



Εικόνα 1.9 Ακολουθία Έναρξης και Ακολουθία Λήξης

1.4.2 Μετάδοση Δεδομένων

Η μετάδοση των δεδομένων γίνεται σε ακολουθίες των 8 bits , τα οποία τοποθετούνται πάνω στη γραμμή SDA. Τη στιγμή εκείνη αλλάζει η κατάσταση της γραμμής SCL από HIGH σε LOW. Ανά 8 bits δεδομένων που μεταφέρονται έρχεται η επιβεβαίωση λήψης από τη συσκευή στη μορφή ενός bit ACK. Άρα αυτό σημαίνει πως η συσκευή receiver είναι έτοιμη να λάβει την επόμενη ακολουθία. Υπάρχει βέβαια και η περίπτωση λήψης ενός NACK (Not Acknowledged) bit, το οποίο σημαίνει πως η receiver συσκευή αδυνατεί να λάβει παραπάνω δεδομένα και έτσι η master συσκευή πρέπει να σταματήσει την αποστολή. Τότε λοιπόν αποστέλλεται η ακολουθία λήξης όπως προαναφέραμε. Ένα NACK bit μπορεί να σημαίνει πως η συσκευή που λαμβάνει τα δεδομένα δεν τα αναγνωρίζει είτε πως η διεύθυνση του slave-receiver είναι εσφαλμένη είτε απλά πως η master συσκευή επιθυμεί να τερματίσει την επικοινωνία.

ΚΕΦΑΛΑΙΟ 2

Εκτελεστικό Μέρος

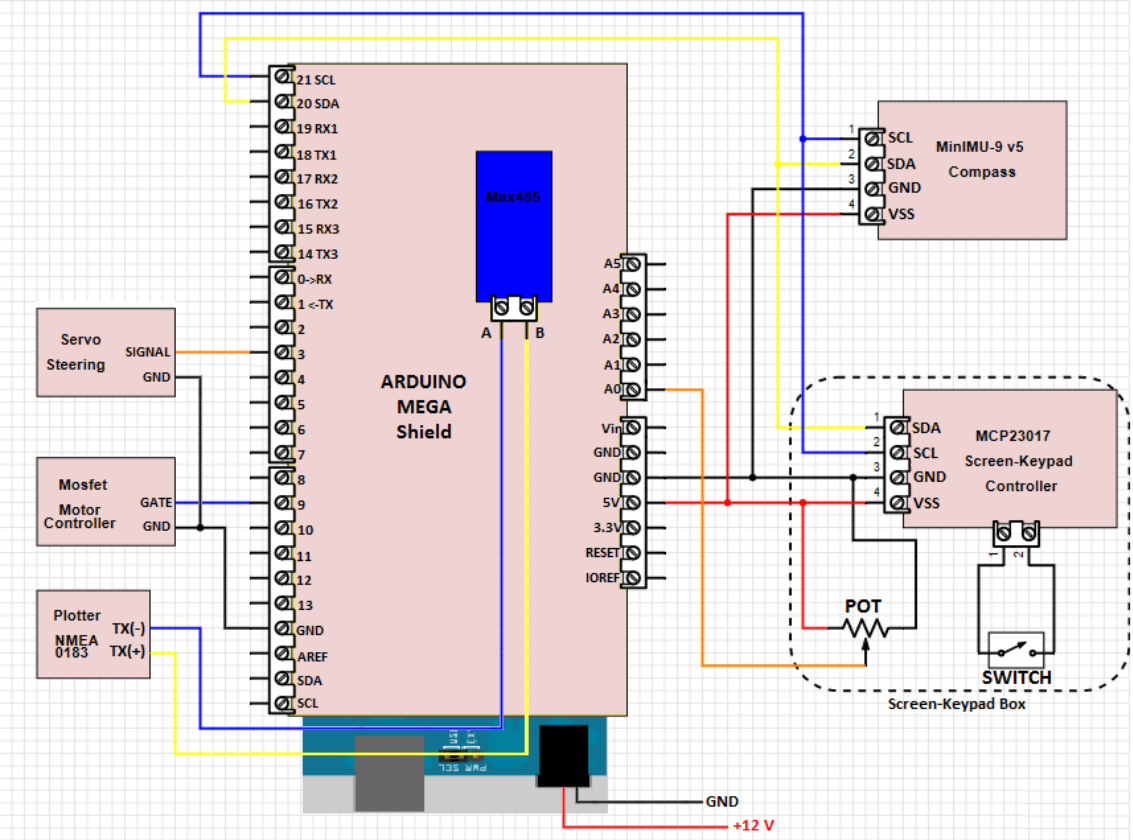
2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα ασχοληθούμε με το πρακτικό κομμάτι της εργασίας. Θα ξεκινήσουμε παραθέτοντας τα διαγράμματα συνδεσμολογίας εξηγώντας περιληπτικά την διάταξή μας. Εν συνεχεία θα παραθέσουμε όλον τον κώδικά που αποτελεί και την ουσία της εργασίας αυτής. Ο κώδικας, ο οποίος είναι γραμμένος σε C, αποτελείται από το κυρίως σώμα (main program) και τις επιμέρους συναρτήσεις. Τοποθετώντας σχόλια σε κάθε σχεδόν γραμμή επιδιώξαμε να κάνουμε την ανάγνωση όπως και την κατανόηση όσο το δυνατόν ευκολότερη. Θεωρούμε επίσης αναγκαίο πρωτού αναλύσουμε τον κώδικα να αναφέρουμε πως ακριβώς λειτουργεί στο σύνολό της η διάταξη.

Όταν ο χρήστης έχει τον διακόπτη στην θέση Manual μπορεί να χειρίζεται το σερβοκινητήρα άρα και το πηδάλιο του σκάφους από το ποτενσιόμετρο. Όταν ο διακόπτης μετατεθεί από την θέση Manual στην θέση Auto καταχωρείται η τρέχουσα πορεία του σκάφους από την ηλεκτρονική πυξίδα ως επιθυμητή τιμή από τον χρήστη. Επίσης από το πληκτρολόγιο του αυτόματου πιλότου έχουμε την δυνατότητα να πληκτρολογήσουμε την πορεία που θέλουμε να ακολουθήσουμε σε μοίρες. Όταν οριστεί στο plotter κάποια πορεία ή διαδρομή και ο διακόπτης βρίσκεται στην θέση Auto τότε λαμβάνουμε από το plotter μέσω του πρωτοκόλλου NMEA 0183 πληροφορίες καθώς και την τιμή της τρέχουσας πορείας υπολογισμένης από το GPS του plotter. Ο ελεγκτής διατηρεί την επιθυμητή πορεία ελέγχοντας τον σερβοκινητήρα και με αυτόν τον τρόπο διορθώνεται η όποια παρέκκλιση υπάρχει .Η κύρια διάφορα με την χρήση του plotter εκτός του ότι μπορεί να ακολουθήσει μια διαδρομή που έχει σχεδιαστεί από τον χρήστη είναι πως η τρέχουσα πορεία δεν είναι απλός η κατεύθυνση της πλώρης μας (όπως συμβαίνει με την ηλεκτρονική πυξίδα) αλλά η πραγματική πορεία προς την οποία κινείται το σκάφος λόγο πλευρικών ανέμων, ρευμάτων και άλλων παραγόντων. Τέλος στην οθόνη χαρακτήρων εμφανίζονται όλες οι απαραίτητες πληροφορίες για τις οποίες πρέπει να είναι ενήμερος ο χρήστης .

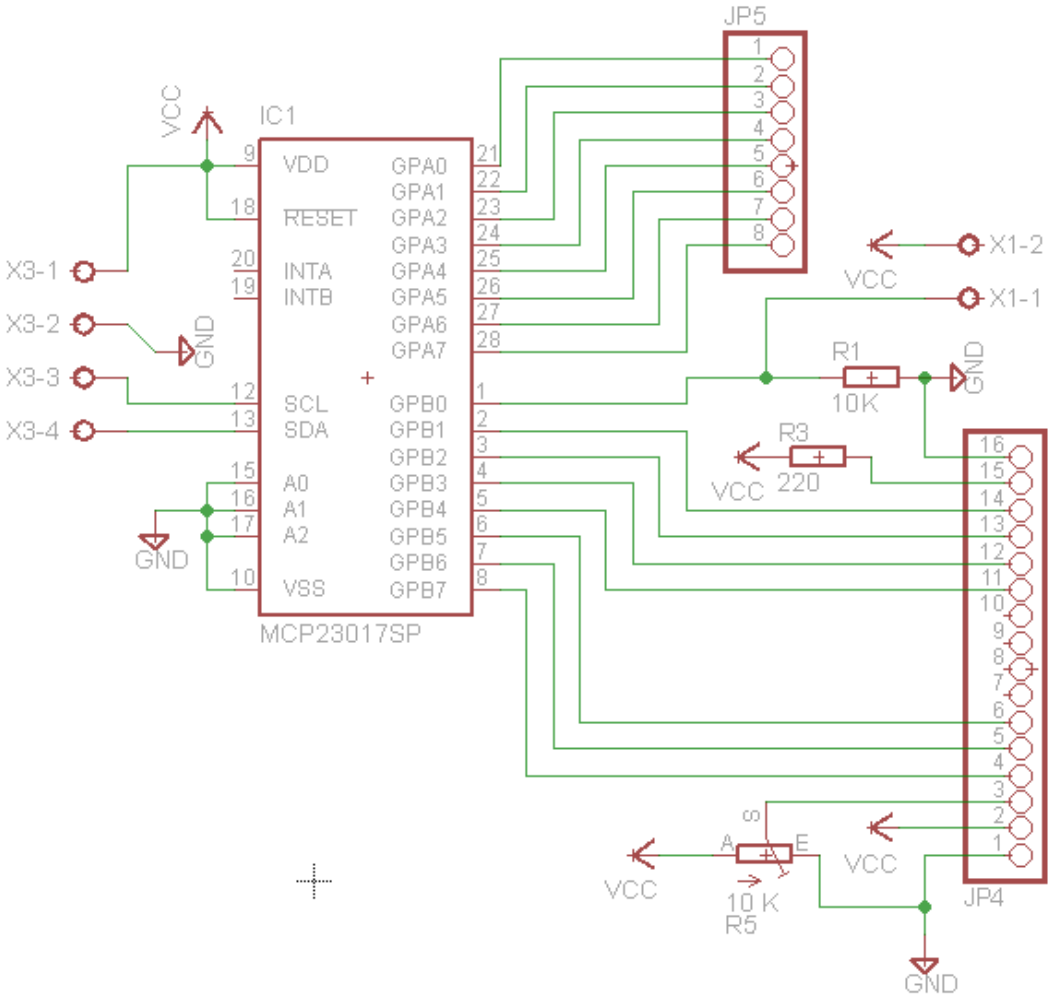
2.1.1 Διαγράμματα Συνδεσμολογίας

Το πρώτο σχήμα αποτελεί μια επισκόπηση της διάταξης. Στο Arduino έχουμε συνδέσει το πηδάλιο (steering), τον ελεγκτή του κινητήρα, το plotter, την πυξίδα, και το mcp23017 που πρακτικά είναι ο ελεγκτής οθόνης και πληκτρολογίου.



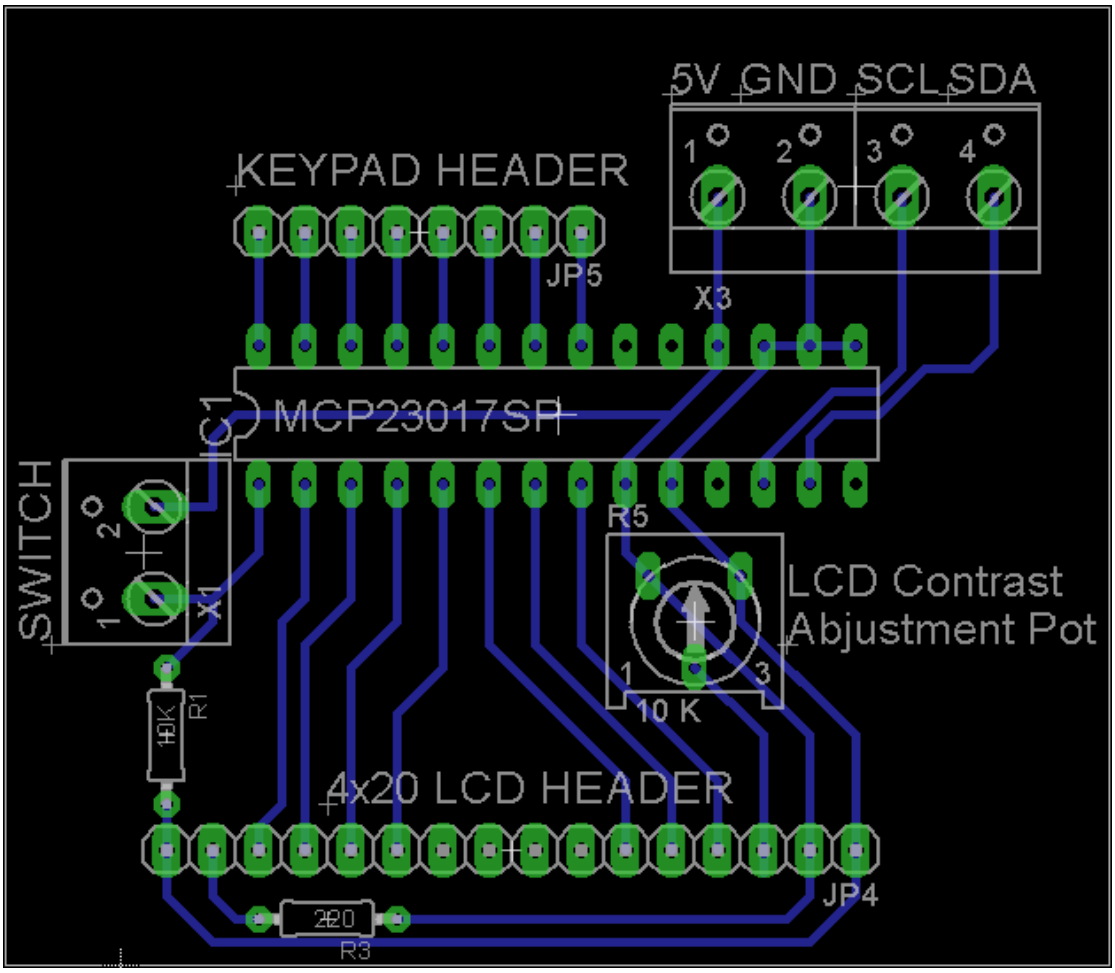
Εικόνα 2.1 Επισκόπηση Διάταξης - Διάγραμμα Συνδεσμολογίας

Παρακάτω παρατίθεται το σχηματικό διάγραμμα του κυκλώματος που δημιουργήσαμε με σκοπό τον έλεγχο της οθόνης χαρακτήρων, του πληκτρολογίου και του διακόπτη λειτουργίας με την χρήση του ολοκληρωμένου MCP23017. Η επιλογή αυτού του ολοκληρωμένου έγινε με γνώμονα την χρήση όσο το δυνατόν μικρότερου αριθμού αγωγών κατά την εγκατάσταση σε ένα σκάφος. Επιτύχαμε με την χρήση 2 καλωδίων δεδομένων (SDA, SCL) του πρωτόκολλου I^2C τον έλεγχο 16 ψηφιακών καναλιών εισόδου-εξόδου.



Εικόνα 2.2 Σχηματικό διάγραμμα κυκλώματος για τον έλεγχο οθόνης-πληκτρολογίου και διακόπτη λειτουργίας

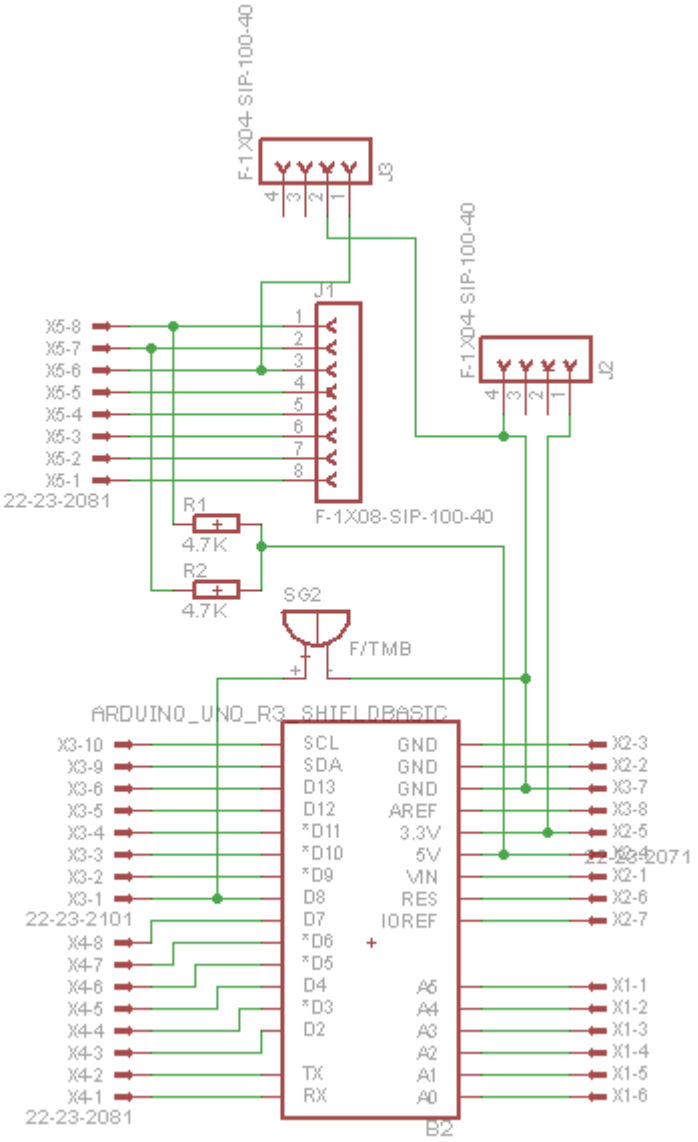
Εν συνέχεια καθορίζουμε την διάταξη των ηλεκτρονικών εξαρτημάτων στην πλακέτα μιας όψης ως εξής:



Εικόνα 2.3 Διάταξη ηλεκτρονικών εξαρτημάτων

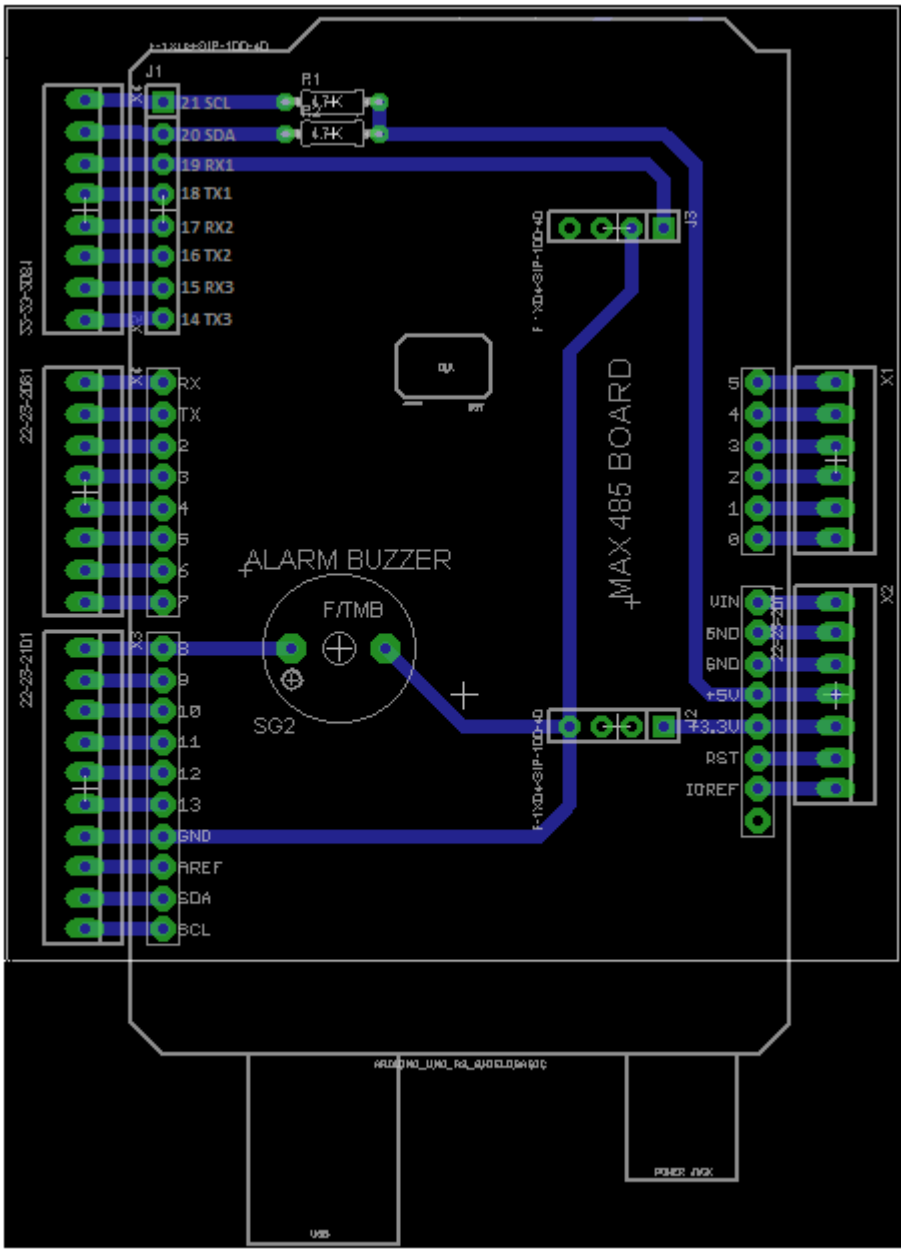
Σχεδιάσαμε ένα Shield για το Arduino Mega που με την χρήση κλεμών εξασφαλίζει την ασφαλή συγκράτηση των αγωγών, επίσης έχει ενσωματωθεί μια πλακέτα RS-485 που επιτρέπει την επικοινωνία με το plotter μέσω του πρωτοκόλλου NMEA 0183 ,τέλος έχει τοποθετηθεί ένα Buzzer για την ειδοποίηση του χρήστη σε περίπτωση δυσλειτουργίας.

Ως shield ορίζουμε τις πλακέτες που μπορούν να εφαρμοσθούν πάνω σε ένα Arduino επεκτείνωντας τις δυνατότητές του.



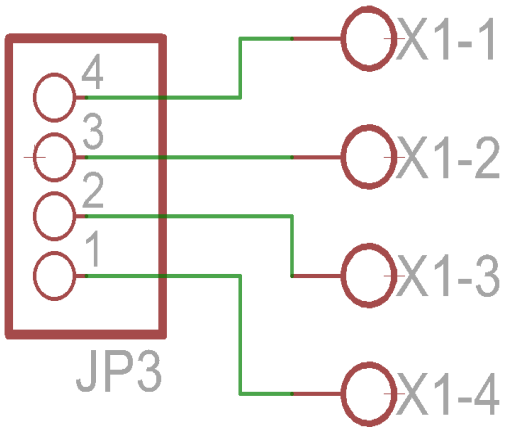
Εικόνα 2.4 Συνδέσεις Arduino Mega Shield

Στο επόμενο σχήμα φαίνεται η διάταξη του Shield.

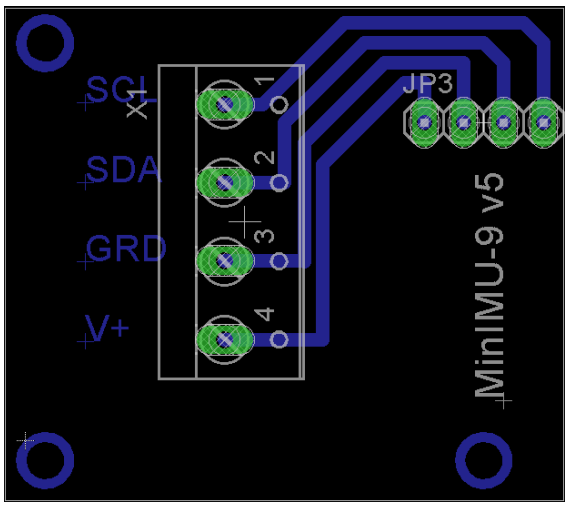


Εικόνα 2.5 Διάταξη του Shield - Arduino Mega Shield

Για την ασφαλή εγκατάσταση της πυξίδας κατασκευάσαμε μια πλακέτα με οπές στερέωσης και κλέμες καλωδίων στην όποια τοποθετείται το MinIMU-9 v5.



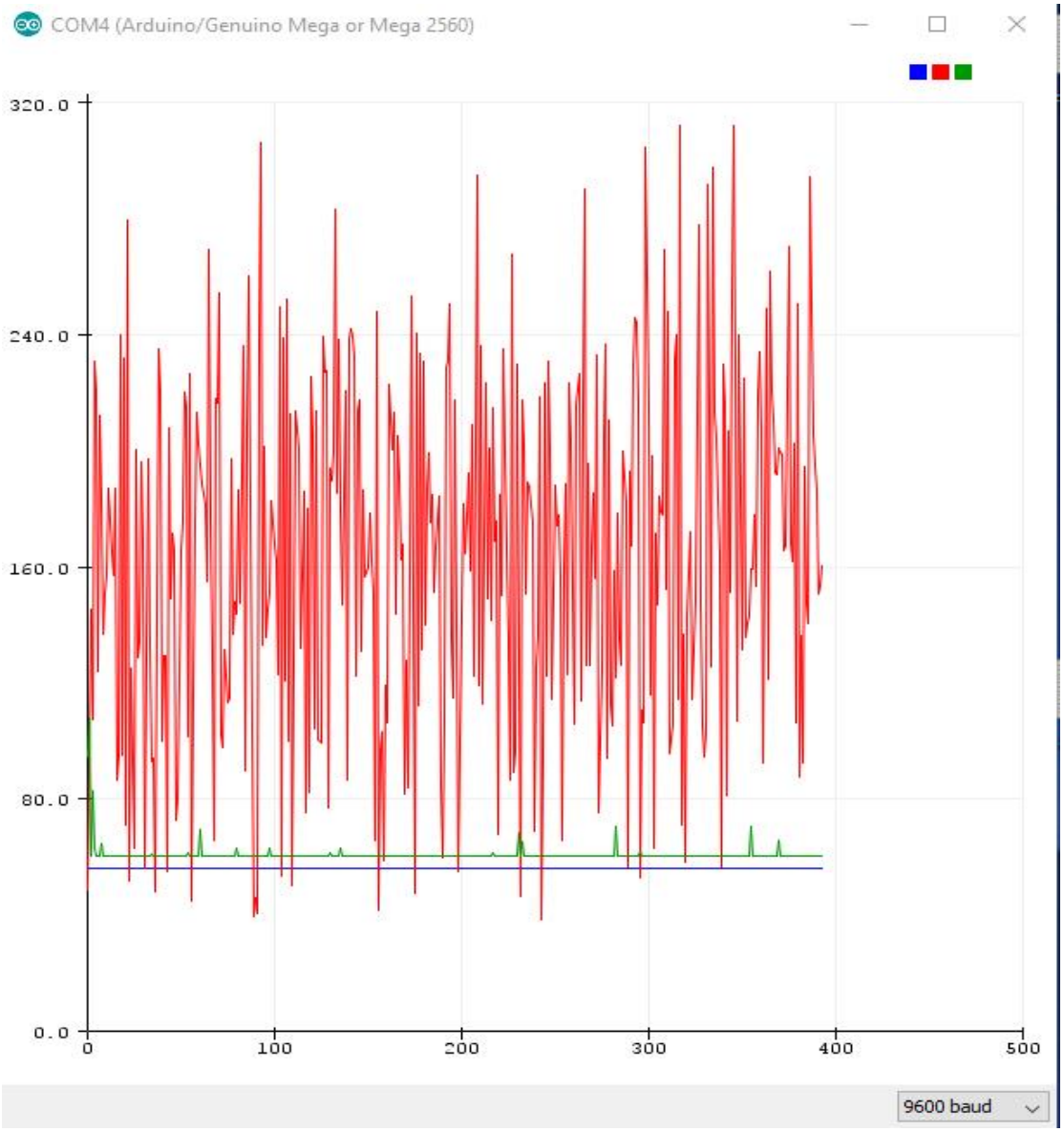
Εικόνα 2.6 Διάταξη καλωδίων ασφαλούς εγκατάστασης πυξίδας



Εικόνα 2.7 Πλακέτα ασφαλούς εγκατάστασης πυξίδας

2.1.2 Εξομάλυνση των τιμών που λαμβάνουμε από την ηλεκτρονική πυξίδα

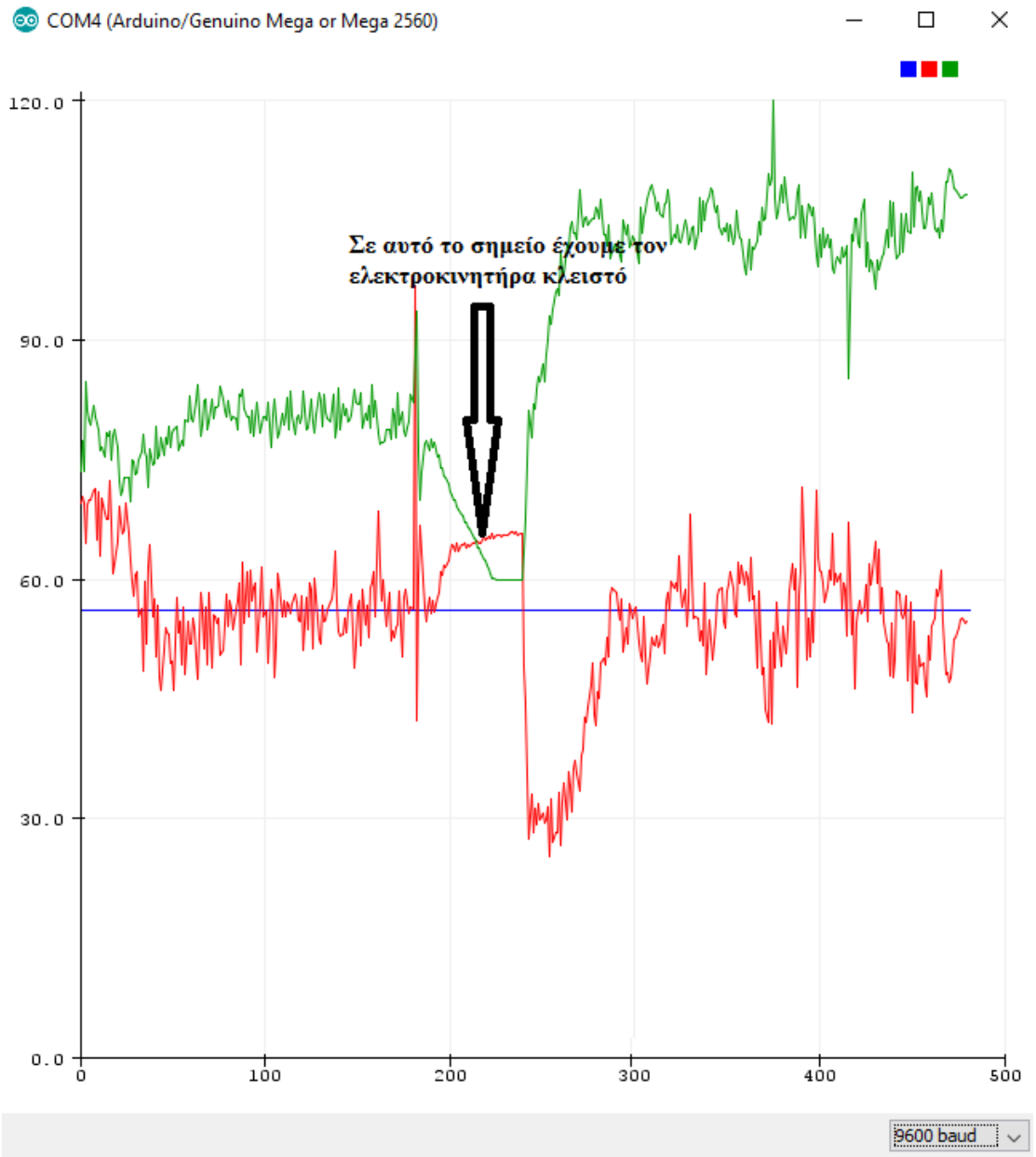
Αντιμετωπίσαμε μεγάλο πρόβλημα ηλεκτρομαγνητικών αλλά και άλλων παρεμβολών όπως κραδασμών και δονήσεων με αποτέλεσμα χωρίς την χρήση φίλτρων η τρέχουσα διεύθυνση της πυξίδας να έχει την παρακάτω μορφή (κόκκινο χρώμα),το μπλε χρώμα αποτυπώνει την επιθυμητή τιμή και το πράσινο την έξοδο του ελεγκτή που ρυθμίζει την θέση του σερβοκινητήρα :



Εικόνα 2.8 Απεικόνηση διαταραχών

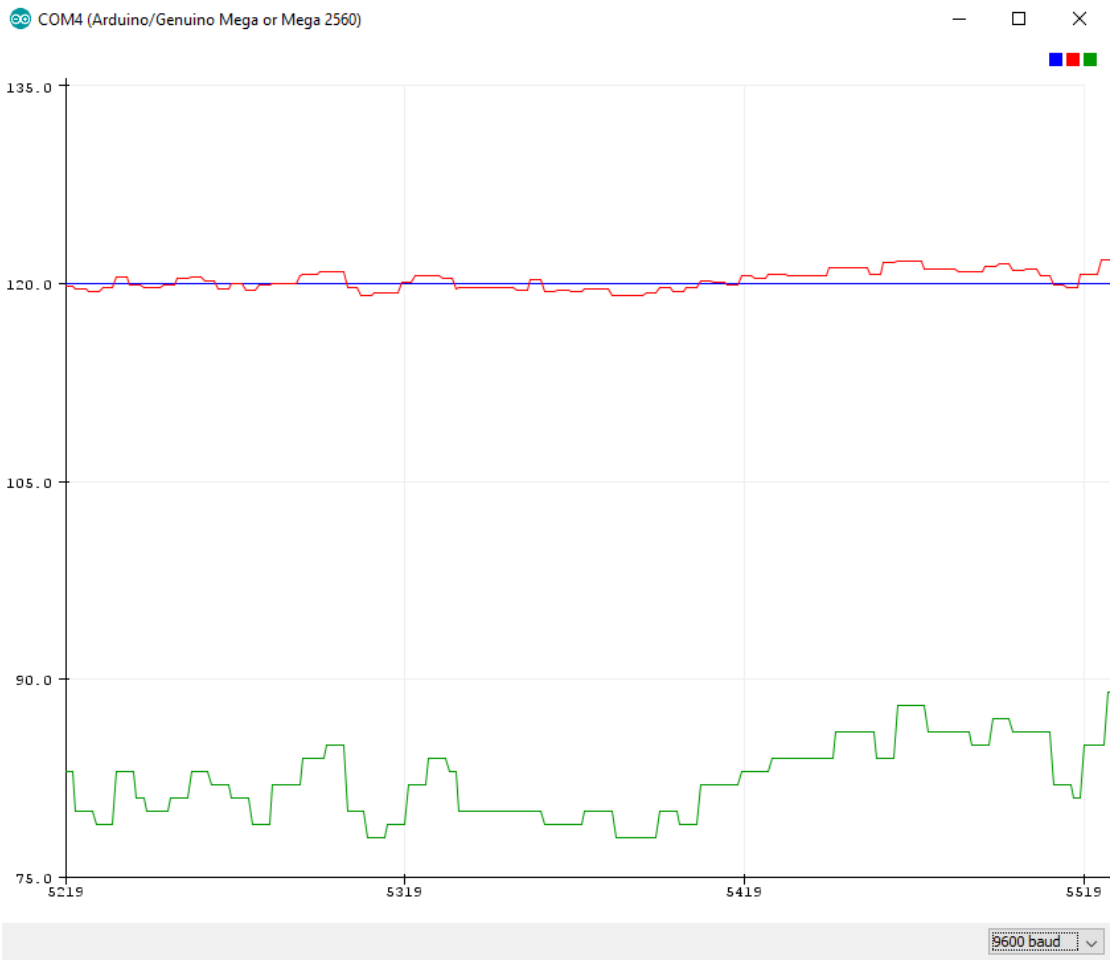
Έγινε προφανές πως με τόσο μεγάλες αποκλίσεις από την πραγματική διεύθυνση η χρήση φίλτρων ήταν επιτακτική.

Αρχικά χρησιμοποιήσαμε ένα complementary (συμπληρωματικό) φίλτρο για καθένα από τους τρεις άξονες του επιταχυνσιόμετρου και του γυροσκοπίου με αποτέλεσμα να βελτιωθεί το πρόβλημα του θορύβου χωρίς όμως να εξαλειφθεί τελείως.



Εικόνα 2.9 Απεικόνηση με χρήση συμπληρωματικού φίλτρου

Στην συνέχεια προσπαθήσαμε να χρησιμοποιήσουμε έναν αλγόριθμο που έχει σχεδιαστεί και χρησιμοποιείται ευρέως σε τέτοιου είδους αισθητήρα, τον DCM (Direction Cosine Matrix) και καταφέραμε να εξαλείψουμε τελείως τον θόρυβο. Και το αποτέλεσμα είναι το εξής:



Εικόνα 2.9.1 Απεικόνηση με αλγορίθμου DCM

Ο αλγόριθμος DCM (Direction Cosine Matrix) είναι σχεδιασμένος ώστε να συγχέει μετρήσεις γυροσκοπίου και επιταχυνσιόμετου χαμηλότερου κόστους. Ένα εκτενές φίλτρο Kalman χρησιμοποιείται για να υπολογίζει τη θέση σε μια σύνθεση DCM. Μια μέθοδος συνδιακύμανσης χρησιμοποιείται για τις μετρήσεις της επιτάχυνσης, ώστε να διασφαλίσει την σταθερότητα έναντι των προσωρινών αντιβαρυντικών επιταχύνσεων, οι οποίες προκαλούν σφάλματα σε έναν αλγόριθμο IMU.

2.1.3 IMU

Τα IMU (intertial measurement units) είναι ηλεκτρονικές συσκευές που μετρούν την επιτάχυνση και την γωνιακή ταχύτητα που ασκούνται σε ένα σώμα καθώς και το μαγνητικό πεδίο που το περιβάλει. Αυτές οι μετρήσεις πραγματοποιούνται από το επιταχυνσιόμετρο, το γυροσκόπιο και το μαγνητόμετρο αντίστοιχα. Ο σκοπός των αισθητήρων επιτάχυνσης και γυροσκοπίου στην εφαρμογή μας είναι να προσδιορίσουν την γωνιακή θέση του αντικειμένου. Το γυροσκοπιο υπολογίζει την γωνιακή ταχύτητα αυτού, ενώ το επιταχυνσιόμετρο προσδιορίζει την θέση του βαρυτικού διανύσματος.

Το μεγαλύτερο πρόβλημα με το επιταχυνσιόμετρο είναι πως είναι επιρρεπές σε στιγμιαίες επιταχύνσεις όπως είναι οι δονήσεις. Γενικά κάθε δύναμη που ασκείται στο σώμα επηρεάζει την τιμές του επιταχυνσιομέτρου.

Το γυροσκόπιο αντιθέτως δεν επηρεάζεται από τις εξωτερικές δυνάμεις που ασκούνται στο αντικείμενο, έχει όμως την τάση να ολισθαίνει κατά την πάροδο του χρόνου, με αποτέλεσμα οι μετρήσεις του γυροσκοπίου να είναι αξιόπιστες μόνο για ένα βραχυπρόθεσμο χρονικό διάστημα.

Το Complimentary (συμπληρωματικό) φίλτρο αντιμετωπίζει τα παραπάνω προβλήματα αφού χρησιμοποιεί την μέτρηση του γυροσκοπίου για ένα βραχυπρόθεσμο χρόνο, ενώ η τιμή του επιτυχυσιομετρου χρησιμοποιείται ως μακροπρόθεσμος όρος μέσω ενός βαθυπερατού φίλτρου.

Το φίλτρο Kalman χρησιμοποιεί μια σειρά μετρήσεων που έχουν ληφθεί κατά την περίοδο του χρόνου, οι οποίες περιέχουν θόρυβο και ανακρίβειες και παράγει εκτιμήσεις που τείνουν να είναι περισσότερο ακριβείς σε σχέση με αυτές που υπολογίζονται με την τελευταία μονάχα μέτρηση.

ΚΕΦΑΛΑΙΟ 3

3.1 Κώδικας

Λόγω του μεγέθους του κώδικα (1.451 γραμμές), τον παραθέτουμε ξεκινώντας από το κυρίως μέρος (main program) και συνεχίζοντας με τις συναρτήσεις, ώστε η ανάγνωση να είναι όσο το δυνατόν ευκολότερη. Ο κώδικας σχεδόν σε κάθε του γραμμή συνοδεύεται από σχόλια που επεξηγούν την κάθε εντολή.

MAIN PROGRAM

```
//Η βιβλιοθήκη LSM6 χρησιμοποιείται για την ενεργοποιήση και λήψη τιμών από το επιταχυσιόμετρο και το γυροσκόπιο της πυξίδας

#include <LSM6.h>

//Η βιβλιοθήκη LIS3MDL χρησιμοποιείται για την ενεργοποιήση και λήψη τιμών από το μαγνητόμετρο της πυξίδας

#include <LIS3MDL.h>

//Η βιβλιοθήκη PID υπολογίζει την θέση του σερβοκινητήρα άρα και του πηδάλιου

#include <PID_v1.h>

//Η βιβλιοθήκη Wire έχει ως σκοπό την I2C επικοινωνία με το LSM303 και MCP23017

#include <Wire.h>

//Η βιβλιοθήκη Keypad_MC17 βοηθάει στην χρήση του πληκτρολογίου μέσω του MCP23017

#include <Keypad_MC17.h>

//Η βιβλιοθήκη Keypad χρησιμοποιείται για το matrix keypad

#include <Keypad.h>

//Η βιβλιοθήκη LiquidTWI2 χρησιμοποιείται για προβολή πληροφοριών στην οθόνη χαρακτήρων μέσω του MCP23017
```

```
#include <LiquidTWI2.h>

//Η βιβλιοθήκη Adafruit_MCP23017 μας επιτρέπει την χρηση διακοπτών στο MCP23017

#include "Adafruit_MCP23017.h"

//Η βιβλιοθήκη Servo χρησιμοποιείται για την πλοήγηση του σκάφους από τον σερβοκινητήρα

#include <Servo.h>

//Η βιβλιοθήκη TinyGPS βοηθάει στην διαχείριση των σειριακών δεδομένων προερχόμενων από το plotter

#include <TinyGPS++.h>



//Δημιουργία ενος Adafruit_MCP23017 αντικειμένου  με το όνομα mcptoggle

Adafruit_MCP23017 mcptoggle;


//Δημιουργία ενος TinyGPS++ αντικειμένου  με το όνομα nmea

TinyGPSPlus nmea;


//Δημιουργία ενος LSM6 αντικειμένου  με το όνομα gyro_acc

LSM6 gyro_acc;

//Δημιουργία ενος LIS3MDL αντικειμένου  με το όνομα mag

LIS3MDL mag;



//Δημιουργία ενος Servo αντικειμένου  με το όνομα myservo

Servo myservo;
```

```
//Δημιουργία ενος LiquidTWI2 αντικειμένου  με το όνομα lcd
LiquidTWI2 lcd(0);

//Δημιουργία δείκτη με όνομα arrivalAlarmnmea του έκτου στοιχείου από την πρόταση GPAPB
TinyGPSCustom arrivalCirclenmea(nmea, "GPAPB", 7);
//Δημιουργία δείκτη με όνομα arrivalAlarmnmea του έκτου στοιχείου από την πρόταση GPAPB
TinyGPSCustom arrivalAlarmnmea(nmea, "GPAPB", 7);
//Δημιουργία δείκτη με όνομα autopheading του ενδέκατου στοιχείου από την πρόταση GPAPB
TinyGPSCustom autopheading(nmea, "GPAPB", 11);
//Δημιουργία δείκτη με όνομα autopmode του δέκατου πέμπτου στοιχείου από την πρόταση GPAPB
TinyGPSCustom autopmode(nmea, "GPAPB", 15);

//Καθορισμός της I2C διεύθυνσης του MCP23017 για το πληκτρολόγιο
#define I2CADDR 0x20
//Καθορισμός της διεύθυνσης για την ενεργοποίηση των εσωτερικών Pull Up αντιστάσεων
#define GPPUA    0x0C



//Κατασκευή της μήτρας του πληκτρολόγιου
const byte ROWS = 4; //Τέσσερις σειρές
```

```
const byte COLS = 4; //Τέσσερις στήλες

//Καταχώριση των χαρακτήρων του πληκτρολογίου

char keys[ROWS][COLS] = {

  {'1', '2', '3', 'A'},

  {'4', '5', '6', 'B'},

  {'7', '8', '9', 'C'},

  {'*', '0', '#', 'D'}

};


//Ορίζουμε την διάταξη των ακροδεκτών των σειρών

byte rowPins[ROWS] = {7 , 6 , 5, 4};

//Ορίζουμε την διάταξη των ακροδεκτών των στηλών

byte colPins[COLS] = {3 , 2, 1, 0};


//Δημιουργία ενος Keypad_MC17 αντικειμένου  με το όνομα keypad1

Keypad_MC17 keypad1( makeKeymap(keys), rowPins, colPins, ROWS, COLS, I2CADDR
);



//Δήλωση μεταβλητών που χρησιμοποιούνται από το κυρίως πρόγραμμα αλλά και από τις
επιμέρους συναρτήσεις


//Δήλωση της μεταβλητής apheading που χρησιμοποιείται για την καταχώρηση της
επιθυμητης πορείας που διαβαζουμε από το Plotter

float apheading;
```

```
//Δήλωση της μεταβλητής gpscourse που χρησιμοποιείται  για την καταχώρηση της τρέχουσας πορείας του σκάφους που διαβαζουμε από το Plotter

float gpscourse;

//Δήλωση της λογικής μεταβλητής nmeapiloting ως ψευδής που χρησιμεύει ως σημαία εάν μια πορεία ή διαδρομή από το plotter είναι διαθέσιμη

bool nmeapiloting = false;

//Δήλωση της λογικής μεταβλητής nmeaWarning ως ψευδής που χρησιμεύει ως σημαία εάν χαθεί η επικοινωνία με το plotter

bool nmeaWarning = false;

//Δήλωση της λογικής μεταβλητής buttonheading ως ψευδής που χρησιμεύει ως σημαία εάν ο διακόπτης λειτουργίας μετατεθεί από την θέση Manual σε Auto

bool buttonheading = false;

//Δήλωση της λογικής μεταβλητής kpdheading ως ψευδής που χρησιμεύει ως σημαία εάν πληκτρολογηθεί μια επιθυμητή από το χρήστη κατεύθυνση

bool kpdheading = false;

//Η μεταβλητή keypadSetpoint χρησιμοποιείται για την καταχώριση της τιμής που πληκτρολογήθηκε από τον χρήστη στην μεταβλητή Setpoint

int keypadSetpoint;

//Timer για την λειτουργία του Servo χωρίς την χρήση delay

unsigned long servo_delay = 0;




//Δήλωση μεταβλητών που χρησιμοποιούνται μόνο από το κυρίως πρόγραμμα
```

```
//Δήλωση της λογικής μεταβλητης wasManual ως αληθής που χρησιμεύει στην αποθήκευση της προηγούμενης κατάστασης του διακόπτη λειτουργίας

bool wasManual = true;

//Η μεταβλητή Inputval χρησιμοποιείται για την καταχώρηση της τρέχουσας τιμής της πυξίδας ενώ όταν έχουμε ορίσει μια πορεία ή διαδρομή από το plotter καταχωρείται η πορεία του σκάφους από το GPS

double Inputval;

//Μεταβλητή που έχει ως σκοπό την καταχώριση της τιμής του διακόπτη λειτουργίας

int modeState = 0;

//Μεταβλητή τιμής ελεγχού του Servo

double OutputTemp;




//Δήλωση μεταβλητών που αφορούν τον PID ελεγκτή

double Setpoint, Input, Output = 90;

//Καθορισμός των παραμέτρων ρύθμισης του PID

double Kp = 3.79, Ki = 0.6, Kd = 0;


//Δημιουργία ενός PID αντικειμένου  με το όνομα myPID

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);




//Function για την ενεργοποίηση των Pull Up αντιστάσεων στο MCP23017
```

```
void expanderWriteBoth (const byte reg, const byte data )
{
  //Εναρξη μετάδοσης στη διεύθυνση 0x20
  Wire.beginTransmission (0x20);
  //Μεταδίδουμε το περιεχόμενο της μεταβλητής reg
  Wire.write (reg);
  //Μεταδίδουμε το περιεχόμενο της μεταβλητής data για τις θύρες A του MCP23017
  Wire.write (data);
  //Μεταδίδουμε το περιεχόμενο της μεταβλητής data για τις θύρες B του MCP23017
  Wire.write (data);
  //Λήξη μετάδοσης
  Wire.endTransmission ();
}
```

```
void setup(void)
{

  //Εναρξη της I2C επικοινωνίας
  Wire.begin ();

  //Εναρξη της επικοινωνίας για την χρήση διακοπτών
```

```
  mcptoggle.begin();
  //Χρήση του PIN 8 του MCP23017 ως INPUT
  mcptoggle.pinMode(8, INPUT);
  //Ενεργοποίηση της Pull Up αντίστασης στο PIN 8
  mcptoggle.pullUp(8, HIGH);



  //Ορισμός του τύπου I/O expander που χρησιμοποιούμε για την λειτουργια της LCD οθόνης
  lcd.setMCPType(LTI_TYPE_MCP23017);
  //Έναρξη της επικοινωνίας της οθόνης και ορισμός του μεγέθους της (20 χαρακτήρων - 4 γραμμών)
  lcd.begin(20, 4);
  //Ενεργοποίηση των Pull Up αντιστάσεων του MCP23017 για όλα τα PINS
  expanderWriteBoth (GPPUA, 0xFF);
  //Θέτουμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
  lcd.setCursor(0, 1);
  //Προβάλουμε κείμενο στην οθόνη χαρακτήρων
  lcd.print("AUTOPILOT VER 1.0");

  //Καλούμε μια καθυστέρηση 1500 millisecond
  delay (1500);
  //Καθαρίζουμε την οθόνη από το κείμενο
  lcd.clear();
  //Εκκίνηση της σειριακής επικοινωνίας (κυρίως για λόγους ανάπτυξης του προγράμματος και εντοπισμού σφαλμάτων μέσω του USB)
```

```
  Serial.begin(9600);
  //Εκκίνηση της σειριακής θύρας 1 για την επικοινωνία με το plotter μέσω του πρωτοκόλλου NMEA 0183
  Serial1.begin(4800);



  //Αρχικοποίηση του γυροσκοπίου και του αξελερόμετρου
  gyro_acc.init();
  //Χρήση των προεπιλεγμένων ρυθμίσεων
  gyro_acc.enableDefault();
  //Αλλαγή της ταχύτητας δειγματοληψίας σε 52 Hz για το γυροσκόπιο
  gyro_acc.writeReg(LSM6::CTRL1_XL, 0x3C);

  //Αρχικοποίηση της πυξίδας
  mag.init();
  //Χρήση των προεπίλεγμενων ρυθμίσεων
  mag.enableDefault();

  //Αλλαγή ταχύτητας δειγματοληψίας σε 104 Hz για το αξελερόμετρο
  gyro_acc.writeReg(LSM6::CTRL2_G, 0x4C);

  //Καλούμε μια καθυστέρηση 20 millisecond
  delay(20);

  //Καλούμε την συνάρτηση εύρεσης των τιμών αντιστάθμισης
```

```
  offset_calculation ();



  //Αποδίδουμε το PIN 3 στο αντικείμενο servo
  myservo.attach(3);



  //Ενεργοποίηση του αντικειμένου myPID
  myPID.SetMode(AUTOMATIC);



  //Καταχώριση στη μεταβλητή Setpoint την τρέχουσα τιμή της πυξίδας
  Setpoint = ((int)  compass());

  pinMode(9, OUTPUT);



}

void loop(void)
{
```

```
  //Καλούμε την συνάρτηση lcdKeypad που έχει ως σκοπό την έλεγχο της οθόνης και του πληκτρολογίου
  lcdKeypad();



  //Καλούμε την συνάρτηση nmeafc που διαχειρίζεται τα εισαρχόμενα σειριακά δεδομένα απο το plotter
  nmeafc();



  //Ελέγχουμε εάν δόθηκε  κάποια επιθυμητή τιμή κατεύθυνσης (μέσω της σημαίας kpdheading) από το πληκτρολόγιο
  if (kpdheading == true) {
    //Καταχωρούμε την τιμή που πληκτρολογήθηκε στη μεταβλητή Setpoint
    Setpoint = keypadSetpoint ;
    //Αλλάζουμε την λογική μεταβλητή kpdheading σε ψευδή
    kpdheading = false;
  }



  //Καταχωρούμε την τιμή του διακόπτη λειτουργίας στην μεταβλητή modeState
  modeState = (mcptoggle.digitalRead(8));
```

```
 //Εάν  ο διακόπτης είναι στη θέση Auto
 if (modeState == HIGH) {


   //Εάν η σημαία nmeapiloting είναι αληθής (έχουμε ορίσει μια πορεία ή διαδρομή από το plotter)
   if (nmeapiloting == true)
   { //Καταχωρούμε στη μεταβλητή Setpoint την μεταβλητή apheading που λαμβάνουμε από το plotter
     Setpoint = apheading;
     //Καταχωρούμε στη μεταβλητή Inputval την μεταβλητή gpscourse που λαμβάνουμε από το plotter
     Inputval = gpscourse;
   }


   //Αλλιώς (αν δηλαδή η σημαία nmeapiloting είναι ψευδής)
   else {
     //Εάν η σημαία wasManual είναι αληθής (στην προηγούμενη κατάσταση ο διακόπτης ήταν στη θέση manual)
     if (wasManual == true) {
       //Καταχωρούμε στη μεταβλητή Setpoint το ακέραιο μέρος της τρέχουσας τιμής της πυξίδας
       Setpoint = ((int)  compass());
       //Αλλάζουμε τη σημαία wasManual σε ψευδής
       wasManual = false;
```

```
      //Αλλάζουμε τη σημαία buttonheading σε αληθής για να εμφανιστεί στην οθόνη το νεο Setpoint
       buttonheading = true;
     }

     //Καταχωρούμε στη μεταβλητή Inputval την τρέχουσα τιμής της πυξίδας
     Inputval = compass();
   }

   //Καταχωρούμε στη μεταβλητή heading_error την διαφορά μεταξύ της τρέχουσα τιμής της πυξίδας (Inputval) και της επιλεγμένης από τον χρήστη πορείας (Setpoint)
   float heading_error = Inputval - Setpoint;
   //Σε αυτό το σημείο υπολογίζουμε την συντομότερη στροφή που θα χρειαστεί να πράξουμε αν η απόλυτη τιμή της μεταβλητή heading_error είναι μεγαλύτερη από 180 μοίρες
   if (abs(heading_error) > 180)
   {

     //Αν επιλεγμένη από τον χρήστη πορεία (Setpoint)είναι μεγαλύτερη από την τρέχουσα τιμής της πυξίδας (Inputval)
     if (Inputval < Setpoint) {
       //Μετατρέπουμε την τιμής της πυξίδας σε μια τιμή μεγαλύτερη από 360 μοίρες
       Inputval = (360 - Setpoint) + Inputval + 360;
```

```
    }

    //Αλλιώς (αν η τρέχουσα τιμής της πυξίδας (Inputval) είναι μεγαλύτερη από την επιλεγμένη από τον χρήστη πορεία (Setpoint))
    else {
      //Μετατρέπουμε την τιμής της πυξίδας σε μια τιμή μικρότερη από 0 μοίρες
      Inputval = -(360 - Inputval) - Setpoint;
    }
  }

  //Καταχωρούμε στη μεταβλητή Input που αφορά την τιμή εισόδου του ελεγκτή την μεταβλητή Inputval
  Input = Inputval;

  //Θέτουμε τα όρια εξόδου του ελεγκτή από 60 έως 120
  myPID.SetOutputLimits(60, 120);

  //Υπολογίζουμε την έξοδο του ελεγκτή
  myPID.Compute();
```

```
  //Μετατρέπουμε την τιμή της εξόδου του ελεγκτή PID από 60 σε 120 ως κατώτερο όριο και 120 σε 60 ως ανώτερο όριο
  OutputTemp = map(Output, 60, 120, 120, 60);



  //Αν η τιμή των τρεχόντων millisecond είναι μεγαλύτερη ή ίση με την μεταβλητή servo_delay
  if (millis() >= servo_delay) {
    //Μεταβάλουμε την θέση του σερβοκινητήρα μέσω της μεταβλητής OutputTemp
    myservo.write(OutputTemp);
    //Καταχωρούμε στην μεταβλητή servo_delay την τιμή των τρεχόντων millisecond συν 15 millisecond για τον έλεγχο του σερβοκινητήρα
    servo_delay = millis() + 15;
  }
 }

 //Αλλιώς (αν  ο διακόπτης είναι στη θέση Manual)
 else {
  //Καλούμε την συνάρτηση potSteering που χρησιμοποιείται στον έλεγχο του σερβοκινητήρα από το ποτενσιόμετρο
  potSteering();
  //Αλλάζουμε τη σημαία wasManual σε αληθής
  wasManual = true;
 }

 //Καλούμε την συνάρτηση εμφάνισης  πληροφοριών σε γραφική παράσταση
```

```
  plot();


}
```

ΚΑΡΤΕΛΑ ELECTRONIC COMPASS

```
//Ρυθμίζουμε την διεύθυνση των x,y,z για το γυροσκόπιο, αξελερόμετρο και μαγνητόμετρο αντίστοιχα

int SENSOR_SIGN[9] = {1, -1, -1, -1, 1, 1, 1, -1, -1};



//Ορίζουμε στη σταθερά GRAVITY την τιμή 256 καθώς αυτή η τιμή ισοδυναμεί με 1G στα ακατέργαστα δεδομένα που προέρχονται από το επιταχυνσιόμετρο

#define GRAVITY 256

//Η μεταβλητή ToRad που επιστρέφει την τιμή σε RAD (*π/180)

#define ToRad(x) ((x)*0.01745329252)



//Ορίζουμε στη σταθερά Gyro_Gain_X το κέρδος του γυροσκοπίου για τον άξονα X

#define Gyro_Gain_X 0.07

//Ορίζουμε στη σταθερά Gyro_Gain_Y το κέρδος του γυροσκοπίου για τον άξονα Y

#define Gyro_Gain_Y 0.07

//Ορίζουμε στη σταθερά Gyro_Gain_Z το κέρδος του γυροσκοπίου για τον άξονα Z

#define Gyro_Gain_Z 0.07
```

```
//Η μεταβλητή Gyro_Scaled_X επιστρέφει την αντισταθμισμένη τιμή που λαμβάνουμε από γυροσκόπιο για τον X άξονα σε RAD

#define Gyro_Scaled_X(x) ((x)*ToRad(Gyro_Gain_X))

//Η μεταβλητή Gyro_Scaled_Y επιστρέφει την αντισταθμισμένη τιμή που λαμβάνουμε από γυροσκόπιο για τον Y άξονα σε RAD

#define Gyro_Scaled_Y(x) ((x)*ToRad(Gyro_Gain_Y))

//Η μεταβλητή Gyro_Scaled_Z επιστρέφει την αντισταθμισμένη τιμή που λαμβάνουμε από γυροσκόπιο για τον Z άξονα σε RAD

#define Gyro_Scaled_Z(x) ((x)*ToRad(Gyro_Gain_Z))


//min: {  -315,  -5453,   +921}   max: { +6729,  +1452,  +7917}

//min: {  -906,  -6434,   +836}   max: { +6497,  +1008,  +8146}

//Βαθμονόμηση των ελαχίστων και μέγιστων τιμών του μαγνητόμετρου για τον κάθε άξονα

#define M_X_MIN -51

#define M_Y_MIN -5805

#define M_Z_MIN +1564

#define M_X_MAX  +5953

#define M_Y_MAX  +47

#define M_Z_MAX +7503


/*

//Βαθμονόμηση των ελαχίστων και μέγιστων τιμών του μαγνητόμετρου για τον κάθε άξονα

#define M_X_MIN -51

#define M_Y_MIN -5805

#define M_Z_MIN +1564
```

```
#define M_X_MAX  +5953

#define M_Y_MAX  +47

#define M_Z_MAX +7503

*/

//Ορίζουμε στη σταθερά Kp_ROLLPITCH το αναλογικό κέρδος για το Roll και Pitch

#define Kp_ROLLPITCH 0.02

//Ορίζουμε στη σταθερά Ki_ROLLPITCH το ολοκληρωτικό κέρδος για το Roll και Pitch

#define Ki_ROLLPITCH 0.00002

//Ορίζουμε στη σταθερά Kp_YAW το αναλογικό κέρδος για το Yaw

#define Kp_YAW 1.2

//Ορίζουμε στη σταθερά  Ki_YAW το ολοκληρωτικό κέρδος για το Yaw

#define Ki_YAW 0.00002



//Δήλωση της μεταβλητής G_Dt που έχει ως σκοπό την καταμέτρηση του χρόνου ολοκλήρωσης ενός κύκλου προγράμματος (χρησιμοποιείται από τον αλγόριθμο DCM)

float G_Dt = 0.02;

//Δήλωση της μεταβλητής timer λειτουργεί ως χρονικό

long timer = 0;

//Δήλωση της μεταβλητής timer_old που καταχωρείται ο χρόνος του προηγούμενου κύκλου προγράμματος

long timer_old;
```

```
//Δήλωση του πίνακα AN που καταχωρούνται οι τιμές του γυροσκοπίου και του αξελερόμετρου

int AN[6];

//Δήλωση του AN_OFFSET πίνακα που καταχωρούνται οι τιμές της αντιστάθμισης των αισθητηρίων του γυροσκοπίου και αξελερόμετρου

int AN_OFFSET[6] = {0, 0, 0, 0, 0, 0};


//Δήλωση των μεταβλητών που χρησιμοποιούνται για την καταχώρηση των τιμών του γυροσκοπίου για κάθε άξονα

int gyro_x, gyro_y, gyro_z;

//Δήλωση των μεταβλητών που χρησιμοποιούνται για την καταχώρηση των τιμών του αξελερόμετρου για κάθε άξονα

int accel_x, accel_y, accel_z;

//Δήλωση των μεταβλητών που χρησιμοποιούνται για την καταχώρηση των τιμών του μαγνητόμετρου για κάθε άξονα

int magnetom_x, magnetom_y, magnetom_z;


//Δήλωση των μεταβλητών που χρησιμοποιούνται για την καταχώρηση των αντισταθμισμένων τιμών του μαγνητόμετρου για κάθε άξονα

float c_magnetom_x, c_magnetom_y, c_magnetom_z;

//Δήλωση της μεταβλητής MAG_Heading που καταχωρούμε την αντισταθμισμένη από κλίση τιμή της πυξίδα

float MAG_Heading;

//Δήλωση της μεταβλητής MAG_Heading_deg που καταχωρούμε την αντισταθμισμένη από κλίση τιμή της πυξίδα σε μοίρες

float MAG_Heading_deg;

//Δήλωση πίνακα διανυσμάτων που αφορούν το αξελερόμετρο
```

```
float Accel_Vector[3] = {0, 0, 0};

//Δήλωση πίνακα διανυσμάτων που αφορούν το γυροσκόπιο

float Gyro_Vector[3] = {0, 0, 0};

//Δήλωση πίνακα διορθωμένων διανυσμάτων που αφορούν το γυροσκόπιο

float Omega_Vector[3] = {0, 0, 0};

//Δήλωση πίνακα αναλογικής διόρθωσης των τιμών του γυροσκοπίου

float Omega_P[3] = {0, 0, 0};

//Δήλωση πίνακα ολοκληρωτικής διόρθωσης των τιμών του γυροσκοπίου

float Omega_I[3] = {0, 0, 0};

//Δήλωση πίνακα που καταχωρούνται οι διορθωμένες τιμές του γυροσκοπίου από ολίσθηση

float Omega[3] = {0, 0, 0};


//Δήλωση της μεταβλητής roll για την καταχώρηση του διατοιχισμού του σκάφους άρα και της πυξίδας

float roll;

//Δήλωση της μεταβλητής pitch για την καταχώρηση της πρόνευσης του σκάφους άρα και της πυξίδας

float pitch;

//Δήλωση της μεταβλητής yaw για την καταχώρηση της εκτροπής του σκάφους άρα και της πυξίδας

float yaw;


//Δήλωση πίνακα που καταχωρούνται οι διορθωμένες τιμές του διατοιχισμού και της πρόνευσης

float errorRollPitch[3] = {0, 0, 0};

//Δήλωση πίνακα που καταχωρούνται διορθωμένες τιμές της εκτροπής
```

```
float errorYaw[3] = {0, 0, 0};


//Δήλωση της μεταβλητής counter που χρησιμεύει ως μετρητής
unsigned int counter = 0;



//Δήλωση ενός πίνακα 3x3 με το όνομα DCM_Matrix για την καταχώριση των τιμών του αλγορίθμου DCM
float DCM_Matrix[3][3] = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};



//Δήλωση ενός πίνακα 3x3 με το όνομα Update_Matrix που χρησιμοποιείται για την καταχώριση των τιμών του γυροσκοπίου
float Update_Matrix[3][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};


//Δήλωση ενός πίνακα 3x3 με το όνομα Temporary_Matrix για την καταχώριση προσωρινών των τιμών για τον αλγόριθμο DCM
float Temporary_Matrix[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
//Δήλωση μεταβλητής newheading που χρησιμοποιείται για την καταχώρηση της φιλτραρισμένης τιμής της πυξίδας
float newheading;
//Δήλωση μεταβλητής filtered_once που χρησιμοποιείται ως δείκτης για το χαμηλοπερατό φίλτρο
bool filtered_once = false;
```

```
//Συνάρτηση υπολογισμού των τιμών της αντιστάθμισης  του γυροσκοπίου και αξελερόμετρου
void offset_calculation () {
  //Η δομή επανάληψης έχει ως σκοπό την λήψη μετρήσεων με σκοπό τον καθορισμό των τιμών αντιστάθμισης του γυροσκοπίου και του αξελερόμετρου
  for (int i = 0; i < 32; i++)
  {
    //Καλούμε την συνάρτηση Read_Gyro
    Read_Gyro();
    //Καλούμε την συνάρτηση Read_Accel
    Read_Accel();

    //Η δομή επανάληψης έχει ως σκοπό το άθροισμα των τιμών για κάθε άξονα του γυροσκοπίου και του αξελερόμετρου αντίστοιχα
    for (int y = 0; y < 6; y++)
      //Προσθέτουμε την τρέχουσα τιμή του κάθε άξονα
      AN_OFFSET[y] += AN[y];
    //Καλούμε μια καθυστέρηση 20 millisecond
    delay(20);
  }

  //Για κάθε άξονα (x,y,z) για το γυροσκόπιο και το αξελερόμετρο αντίστοιχα
  for (int y = 0; y < 6; y++) {
    //Διαιρούμε το κάθε στοιχείο του πινάκα με τον αριθμό των δειγμάτων
    AN_OFFSET[y] = AN_OFFSET[y] / 32;
```

```
  }
  //Αφαιρούμε από το στοιχείο 5 του πινάκα (που αντιστοιχεί  στον Y άξονα του αξελερόμετρου) την επίδραση της βαρύτητας
  AN_OFFSET[5] -= GRAVITY * SENSOR_SIGN[5];


  //Καταχωρούμε στην μεταβλητή timer την τιμή των τρεχόντων millisecond
  timer = millis();
  delay(20);

  //Καταχωρούμε στον counter την τιμή μηδέν
  counter = 0;

}

//Συνάρτηση καταχώρησης των τρεχόντων τιμών του γυροσκοπίου
void Read_Gyro() {
  //Διαβάζουμε το μητρώο από το γυροσκόπιο για τους τρεις άξονες
  gyro_acc.readGyro();

  //Καταχωρούμε στον πίνακα AN στη θέση 0 την τιμή του γυροσκοπίου για τον x άξονα
  AN[0] = gyro_acc.g.x;
  //Καταχωρούμε στον πίνακα AN στη θέση 1 την τιμή του γυροσκοπίου για τον y άξονα
  AN[1] = gyro_acc.g.y;
  //Καταχωρούμε στον πίνακα AN στη θέση 2 την τιμή του γυροσκοπίου για τον z άξονα
```

```
  AN[2] = gyro_acc.g.z;


  //Καταχωρούμε στη μεταβλητή gyro_x την αντισταθμισμένη τιμή του γυροσκοπίου για τον x άξονα με την σωστή κατεύθυνση
  gyro_x = SENSOR_SIGN[0] * (AN[0] - AN_OFFSET[0]);
  //Καταχωρούμε στη μεταβλητή gyro_y την αντισταθμισμένη τιμή του γυροσκοπίου για τον y άξονα με την σωστή κατεύθυνση
  gyro_y = SENSOR_SIGN[1] * (AN[1] - AN_OFFSET[1]);
  //Καταχωρούμε στη μεταβλητή gyro_z την αντισταθμισμένη τιμή του γυροσκοπίου για τον z άξονα με την σωστή κατεύθυνση
  gyro_z = SENSOR_SIGN[2] * (AN[2] - AN_OFFSET[2]);
}



//Συνάρτηση καταχώρησης των τρεχόντων τιμών του αξελερόμετρου
void Read_Accel()
{
  //Διαβάζουμε το μητρώο από το αξελερόμετρο για τους τρεις άξονες
  gyro_acc.readAcc();


  //Καταχωρούμε στον πίνακα AN στη θέση 3 την τιμή του αξελερόμετρου για τον x άξονα (αφού έχουμε μετατοπίσει κατά τέσσερα ψηφία αριστερά την τιμή)
  AN[3] = gyro_acc.a.x >> 4;
  //Καταχωρούμε στον πίνακα AN στη θέση 4 την τιμή του αξελερόμετρου για τον y άξονα (αφού έχουμε μετατοπίσει κατά τέσσερα ψηφία αριστερά την τιμή)
  AN[4] = gyro_acc.a.y >> 4;
```

```
  //Καταχωρούμε στον πίνακα AN στη θέση 5 την τιμή του αξελερόμετρου για τον z άξονα (αφού έχουμε μετατοπίσει κατά τέσσερα ψηφία αριστερά την τιμή)
  AN[5] = gyro_acc.a.z >> 4;

  //Καταχωρούμε στη μεταβλητή accel_x την αντισταθμισμένη τιμή του αξελερόμετρου για τον x άξονα με την σωστή κατεύθυνση
  accel_x = SENSOR_SIGN[3] * (AN[3] - AN_OFFSET[3]);
  //Καταχωρούμε στη μεταβλητή accel_y την αντισταθμισμένη τιμή του αξελερόμετρου για τον y άξονα με την σωστή κατεύθυνση
  accel_y = SENSOR_SIGN[4] * (AN[4] - AN_OFFSET[4]);
  //Καταχωρούμε στη μεταβλητή accel_z την αντισταθμισμένη τιμή του αξελερόμετρου για τον z άξονα με την σωστή κατεύθυνση
  accel_z = SENSOR_SIGN[5] * (AN[5] - AN_OFFSET[5]);
}

//Συνάρτηση καταχώρησης των τρεχόντων τιμών του μαγνητόμετρου
void Read_Compass()
{
  //Διαβάζουμε το μητρώο από το μαγνητόμετρο για τους τρεις άξονες
  mag.read();

  //Καταχωρούμε στη μεταβλητή magnetom_x την τιμή του μαγνητόμετρου για τον x άξονα με την σωστή κατεύθυνση
  magnetom_x = SENSOR_SIGN[6] * mag.m.x;
```

```
  //Καταχωρούμε στη μεταβλητή magnetom_y την τιμή του μαγνητόμετρου για τον y άξονα με την σωστή κατεύθυνση
  magnetom_y = SENSOR_SIGN[7] * mag.m.y;
  //Καταχωρούμε στη μεταβλητή magnetom_z την τιμή του μαγνητόμετρου για τον z άξονα με την σωστή κατεύθυνση
  magnetom_z = SENSOR_SIGN[8] * mag.m.z;

}

//Η συνάρτηση compass επιστρέφει την τρέχουσα τιμή της πυξίδας
float compass () {
  // Αν η διαφορά των τρεχόντων millisecond και της μεταβλητής timer είναι μεγαλύτερη ή ίση με 20
  if ((millis() - timer) >= 20)
  {
    //Αυξάνουμε την μεταβλητή counter κατά 1
    counter++;
    //Καταχωρούμε στην μεταβλητή timer_old την τιμή της μεταβλητής timer
    timer_old = timer;
    //Στη μεταβλητή timer καταχωρούμε την τιμή των τρεχόντων millisecond
    timer = millis();
    //Αν η τιμή της μεταβλητής timer είναι μεγαλύτερη από την μεταβλητή timer_old
    if (timer > timer_old)
```

```
  { //Στη μεταβλητή G_Dt καταχωρείται ο πραγματικός χρόνος που χρειάζεται ένας βρόχος προγράμματος (χρησιμοποιείται από τον αλγόριθμο DCM)
    G_Dt = (timer - timer_old) / 1000.0;
    //Αν η μεταβλητή G_Dt είναι μεγαλύτερη από 0.2 δηλαδή (200 millisecond)
    if (G_Dt > 0.2)
      //Αγνοούμε χρόνος ολοκλήρωσης ενός βρόχου μεγαλύτερους από 200 millisecond καταχωρώντας στη μεταβλητή G_Dt την τιμή μηδέν
      G_Dt = 0;
  }
  //Αλλιώς
  else {
    //Καταχωρούμε στη μεταβλητή G_Dt την τιμή μηδέν
    G_Dt = 0;
  }

  //Καλούμε την συνάρτηση Read_Gyro που έχει ως σκοπό την λήψη και καταχώριση των τιμών του γυροσκοπίου για τους άξονες x,y,z
  Read_Gyro();
  //Καλούμε την συνάρτηση Read_Accel που έχει ως σκοπό την λήψη και καταχώριση των τιμών του αξελερόμετρου για τους άξονες x,y,z
  Read_Accel();
```

```
  //Αν η μεταβλητή counter είναι μεγαλύτερη από ένα (διαβάζουμε τα δεδομένα της πυξίδας κάθε δύο βρόχους)
  if (counter > 1)
  {
   //Καταχωρούμε στη μεταβλητή counter την τιμή μηδέν
   counter = 0;
   //Καλούμε την συνάρτηση Read_Compass που έχει ως σκοπό την λήψη και καταχώριση των τιμών του μαγνητόμετρου για τους άξονες x,y,z
   Read_Compass();
   //Καλούμε την συνάρτηση Compass_Heading που υπολογίζει την μαγνητική μας κατεύθυνση
   Compass_Heading();
  }

  //Καλούμε την συνάρτηση Matrix_update που έχει ως σκοπό την αναναίωση των τιμών του πίνακα που υπολογίζονται από τον αλγόριθμο DCM
  Matrix_update();
  //Καλούμε την συνάρτηση Normalize που έχει ως σκοπό την ομαλοποίηση των τιμών
  Normalize();
  //Καλούμε την συνάρτηση Drift_correction που έχει ως σκοπό την διόρθωση της ολίσθησης του γυροσκοπίου
  Drift_correction();
  //Καλούμε την συνάρτηση Euler_angles που υπολογίζει τις τιμές διατοιχισμού (Pitch) ,πρόνευσης (Roll) και εκτροπής (Yaw) από τις τιμές που έχουν υπολογιστεί από τον αλγόριθμο DCM
  Euler_angles();
```

```
    //Καταχωρούμε στην μεταβλητή newheading την υπάρχουσα τιμή πολλαπλασιασμένη με 0.8 και 0.2 της νέας τιμής MAG_Heading_deg (χαμηλοπερατό φίλτρο)
    newheading = newheading * 0.8 + MAG_Heading_deg * 0.20;



  }


  //Αν είναι η πρώτη φορά που χρησιμοποιούμε το φίλτρο
  if (filtered_once == false) {
    //Επιστρέφουμε την αντισταθμισμένη από κλίση τιμή της πυξίδας δίχως να έχει φιλτραριστεί
    return MAG_Heading_deg;
    filtered_once = true;
  }
  //Αλλιώς
  else {
    //Επιστρέφουμε την φιλτραρισμένη και αντισταθμισμένη από κλίση τιμή της πυξίδας
    return newheading;
  }
}
```

```
//Συνάρτηση καθορισμού της κατεύθυνσης του σκάφους αντισταθμισμένη από την κλίση του
void Compass_Heading()
{
  //Δήλωση μεταβλητών για την καταχώρηση των αντισταθμισμένων από κλήση τιμών του μαγνητόμετρου για τον άξονα x και y αντίστοιχα
  float MAG_X, MAG_Y;
  //Δήλωση μεταβλητών για την καταχώρηση αποτελεσμάτων τριγωνομετρικών συναρτήσεων
  float cos_roll, sin_roll;
  float cos_pitch, sin_pitch;

  //Καταχωρούμε στη μεταβλητή cos_roll το συνημίτονο του διατοιχισμού
  cos_roll = cos(roll);
  //Καταχωρούμε στη μεταβλητή sin_roll το ημίτονο του διατοιχισμού
  sin_roll = sin(roll);
  //Καταχωρούμε στη μεταβλητή cos_pitch το συνημίτονο της πρόνευσης
  cos_pitch = cos(pitch);
  //Καταχωρούμε στη μεταβλητή sin_pitch το ημίτονο της πρόνευσης
  sin_pitch = sin(pitch);

  //Προσαρμόζουμε την τιμή του μαγνητόμετρου για τον κάθε άξονα αλλάζοντας την κατεύθυνση ,αντισταθμίζοντας την απόκλιση του και μετατρέπουμε την τιμή του ώστε να κυμαίνεται από -0.5 έως +0.5
  c_magnetom_x = (float)(magnetom_x - SENSOR_SIGN[6] * M_X_MIN) / (M_X_MAX - M_X_MIN) - SENSOR_SIGN[6] * 0.5;
  c_magnetom_y = (float)(magnetom_y - SENSOR_SIGN[7] * M_Y_MIN) / (M_Y_MAX - M_Y_MIN) - SENSOR_SIGN[7] * 0.5;
```

```
  c_magnetom_z = (float)(magnetom_z - SENSOR_SIGN[8] * M_Z_MIN) / (M_Z_MAX - M_Z_MIN) - SENSOR_SIGN[8] * 0.5;



  //Αντιστάθμιση κλίσης για τον άξονα x του μαγνητόμετρου
  MAG_X = c_magnetom_x * cos_pitch + c_magnetom_y * sin_roll * sin_pitch + c_magnetom_z * cos_roll * sin_pitch;
  //Αντιστάθμιση κλίσης για τον άξονα y του μαγνητόμετρου
  MAG_Y = c_magnetom_y * cos_roll - c_magnetom_z * sin_roll;
  //Υπολογίζουμε την κατεύθυνση μας , αντισταθμισμένη από κλήση σε μοίρες
  MAG_Heading = atan2(-MAG_Y, MAG_X);
  MAG_Heading_deg = 180 * atan2(-MAG_Y, MAG_X) / PI;
  //Αν η κατεύθυνση μας έχει αρνητική τιμή προσθέτουμε 360 μοίρες
  if (MAG_Heading_deg < 0) {
    MAG_Heading_deg += 360;
  }

}



//Συνάρτηση διόρθωσης της παρέκκλισης του γυροσκοπίου
void Drift_correction(void)
{
  //Δήλωση μεταβλητών για την καταχώρηση της τιμής του μαγνητόμετρου για τους άξονες x και y
```

```
  float mag_heading_x, mag_heading_y;
  //Δήλωση μεταβλητής για την καταχώρηση του σφάλματος της κατεύθυνσης
  float errorCourse;

  //Compensation the Roll, Pitch and Yaw drift.
  //Δήλωση πίνακα Scaled_Omega_P για την καταχώριση των αναγομένων τιμών του πίνακα Omega_P
  static float Scaled_Omega_P[3];
  //Δήλωση πίνακα Scaled_Omega_I για την καταχώριση των αναγομένων τιμών του πίνακα Omega_I
  static float Scaled_Omega_I[3];
  //Δήλωση μεταβλητής Accel_magnitude για την καταχώρηση του μέτρου της επιτάχυνσης
  float Accel_magnitude;
  //Δήλωση μεταβλητής Accel_weight για την καταχώρηση του της τιμής του φίλτρου αξιοπιστίας
  float Accel_weight;

  //Υπολογισμός του διατοιχισμού (Roll) και της πρόνευσης (Pitch)

  //Υπολογίζουμε το μετρό του διανύσματος για το επιταχυνσιόμετρο
  Accel_magnitude = sqrt(Accel_Vector[0] * Accel_Vector[0] + Accel_Vector[1] * Accel_Vector[1] + Accel_Vector[2] * Accel_Vector[2]);
  //Κλιμακώνουμε την τιμή της μεταβλητής Accel_magnitude σε σχέση με την βαρύτητα
  Accel_magnitude = Accel_magnitude / GRAVITY;
```

```
  //Καταχωρούμε στην μεταβλητή Accel_weight την δυναμικά σταθμισμένη τιμή της μεταβλητής Accel_magnitude (reliability filter)

  Accel_weight = constrain(1 - 2 * abs(1 - Accel_magnitude), 0, 1); //

  //Καλούμε την συνάρτηση Vector_Cross_Product που επιστρέφει στην μεταβλητή errorRollPitch[0] το διανυσματικό γινόμενο μεταξύ των τιμών Accel_Vector[0] και DCM_Matrix[2][0]

  Vector_Cross_Product(&errorRollPitch[0], &Accel_Vector[0], &DCM_Matrix[2][0]); //adjust the ground of reference

  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή errorRollPitch[0] με το αναλογικό κέδος Kp_ROLLPITCH και την καταχωρήσουμε στην μεταβλητή Omega_P[0]

  Vector_Scale(&Omega_P[0], &errorRollPitch[0], Kp_ROLLPITCH * Accel_weight);

  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή errorRollPitch[0] με το ολοκληρωτικό κέδος Ki_ROLLPITCH και την καταχωρήσουμε στην μεταβλητή Scaled_Omega_I

  Vector_Scale(&Scaled_Omega_I[0], &errorRollPitch[0], Ki_ROLLPITCH * Accel_weight);

  //Καταχωρούμε στον πίνακα Omega_I το άθροισμα των διανυσμάτων των πινάκων Omega_I Scaled_Omega_I

  Vector_Add(Omega_I, Omega_I, Scaled_Omega_I);



  //Κάνουμε διόρθωση της παρέκκλισης του γυροσκοπίου για τον άξονα εκτροπής (Yaw) με την βοήθεια της μαγνητικής κατεύθυνσης

  //Καταχωρούμε στην μεταβλητή mag_heading_x το συνημίτονο της τιμής MAG_Heading (της μαγνητικής κατεύθυνσης)

  mag_heading_x = cos(MAG_Heading);
```

```
  //Καταχωρούμε στην μεταβλητή mag_heading_y το ημίτονο της τιμής MAG_Heading (της μαγνητικής κατεύθυνσης)

  mag_heading_y = sin(MAG_Heading);

  //Υπολογίζουμε το σφάλμα εκτροπής

  errorCourse = (DCM_Matrix[0][0] * mag_heading_y) - (DCM_Matrix[1][0] * mag_heading_x);

  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή DCM_Matrix[2][0] στο σφάλμα εκτροπής errorCourse και την καταχωρήσουμε στην μεταβλητή errorYaw

  Vector_Scale(errorYaw, &DCM_Matrix[2][0], errorCourse);

  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή errorYaw[0] με το αναλογίκο κέδος Kp_YAW και την καταχωρήσουμε στην μεταβλητή Scaled_Omega_P[0]

  Vector_Scale(&Scaled_Omega_P[0], &errorYaw[0], Kp_YAW);

  //Καταχωρούμε στον πίνακα Omega_P το άθροισμα των διανυσμάτων των πινάκων Omega_P Scaled_Omega_P

  Vector_Add(Omega_P, Omega_P, Scaled_Omega_P);

  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή errorYaw[0] με το ολοκληρωτικό κέδος Ki_YAW και την καταχωρήσουμε στην μεταβλητή Scaled_Omega_I[0]

  Vector_Scale(&Scaled_Omega_I[0], &errorYaw[0], Ki_YAW);

  //Καταχωρούμε στον πίνακα Omega_I το άθροισμα των διανυσμάτων των πινάκων Omega_I Scaled_Omega_I

  Vector_Add(Omega_I, Omega_I, Scaled_Omega_I);

}
```

```
//Συνάρτηση ομαλοποίησης των τιμών του πίνακα DCM_Matrix
```

```
void Normalize(void)
{
  //Δήλωση της μεταβλητής error που χρησιμοποιείται για την καταχώριση του σφάλματος
  float error = 0;
  //Δήλωση ενός πίνακα 3x3 με το όνομα temporary για την καταχώριση προσωρινών τιμών
  float temporary[3][3];
  //Δήλωση της μεταβλητής renorm που χρησιμοποιείται για την περαιτέρω ομαλοποίηση
  float renorm = 0;

  //Καταχωρούμε στην μεταβλητή error το εσωτερικό γινόμενο των διανυσμάτων DCM_Matrix[0][0] και DCM_Matrix[1][0] πολλαπλασιασμένο κατα -0.5
  error = -Vector_Dot_Product(&DCM_Matrix[0][0], &DCM_Matrix[1][0]) * .5;
  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή DCM_Matrix[1][0] με την τιμή του σφάλματος error και την καταχωρήσουμε στην μεταβλητή temporary[0][0]
  Vector_Scale(&temporary[0][0], &DCM_Matrix[1][0], error);
  //Καλούμε την συνάρτηση Vector_Scale για να ανάγουμε την τιμή DCM_Matrix[0][0] με την τιμή του σφάλματος error και την καταχωρήσουμε στην μεταβλητή temporary[1][0]
  Vector_Scale(&temporary[1][0], &DCM_Matrix[0][0], error);
  //Καταχωρούμε στη μεταβλητή temporary[0][0] το άθροισμα των διανυσμάτων temporary[0][0] και DCM_Matrix[0][0]
  Vector_Add(&temporary[0][0], &temporary[0][0], &DCM_Matrix[0][0]);
  //Καταχωρούμε στη μεταβλητή temporary[1][0] το άθροισμα των διανυσμάτων temporary[1][0] και DCM_Matrix[1][0]
  Vector_Add(&temporary[1][0], &temporary[1][0], &DCM_Matrix[1][0]);//eq.19
  //Καταχώρουμε στην μεταβλητή temporary[2][0] το εσωτέρικο γινόμενο των διανυσμάτων temporary[0][0] και temporary[1][0]
```

```
  Vector_Cross_Product(&temporary[2][0], &temporary[0][0], &temporary[1][0]);
  //Καταχωρούμε στην μεταβλητή renorm το αποτέλεσμα της παρακάτω πράξης
  renorm = .5 * (3 - Vector_Dot_Product(&temporary[0][0], &temporary[0][0]));
  //Καλούμε την συνάρτηση Vector_Scale για να ομαλοποιήσουμε την τιμή temporary[0][0] με τιμή renorm και την καταχωρήσουμε στην μεταβλητή DCM_Matrix[0][0]
  Vector_Scale(&DCM_Matrix[0][0], &temporary[0][0], renorm);
  //Καταχωρούμε στην μεταβλητή renorm το αποτέλεσμα της παρακάτω πράξης
  renorm = .5 * (3 - Vector_Dot_Product(&temporary[1][0], &temporary[1][0]));
  //Καλούμε την συνάρτηση Vector_Scale για να ομαλοποιήσουμε την τιμή temporary[1][0] με τιμή renorm και την καταχωρήσουμε στην μεταβλητή DCM_Matrix[1][0]
  Vector_Scale(&DCM_Matrix[1][0], &temporary[1][0], renorm);
  //Καταχωρούμε στην μεταβλητή renorm το αποτέλεσμα της παρακάτω πράξης
  renorm = .5 * (3 - Vector_Dot_Product(&temporary[2][0], &temporary[2][0]));
  //Καλούμε την συνάρτηση Vector_Scale για να ομαλοποιήσουμε την τιμή temporary[2][0] με τιμή renorm και την καταχωρήσουμε στην μεταβλητή DCM_Matrix[2][0]
  Vector_Scale(&DCM_Matrix[2][0], &temporary[2][0], renorm);
}
```

```
//Συνάρτηση υπολογισμού και καταχώρησης σε πίνακα των τιμών του γυροσκοπίου και του αξελερόμετρου
void Matrix_update(void)
{
```

//Καταχωρούμε στη θέση 0 του πίνακα Gyro_Vector την τιμή του γυροσκοπίου για τον άξονα x αφού την ανάγουμε με την συνάρτηση Gyro_Scaled_X (ουσιαστικά πολλαπλασιάζεται με 0.07 και μετατρέπεται σε Rad)

Gyro_Vector[0] = Gyro_Scaled_X(gyro_x);

//Καταχωρούμε στη θέση 1 του πίνακα Gyro_Vector την τιμή του γυροσκοπίου για τον άξονα y αφού την ανάγουμε με την συνάρτηση Gyro_Scaled_Y (ουσιαστικά πολλαπλασιάζεται με 0.07 και μετατρέπεται σε Rad)

Gyro_Vector[1] = Gyro_Scaled_Y(gyro_y);

//Καταχωρούμε στη θέση 2 του πίνακα Gyro_Vector την τιμή του γυροσκοπίου για τον άξονα z αφού την ανάγουμε με την συνάρτηση Gyro_Scaled_Z (ουσιαστικά πολλαπλασιάζεται με 0.07 και μετατρέπεται σε Rad)

Gyro_Vector[2] = Gyro_Scaled_Z(gyro_z);

//Καταχωρούμε στη θέση 0 του πίνακα Accel_Vector την τιμή του αξελερόμετρου για τον άξονα x

Accel_Vector[0] = accel_x;

//Καταχωρούμε στη θέση 1 του πίνακα Accel_Vector την τιμή του αξελερόμετρου για τον άξονα y

Accel_Vector[1] = accel_y;

//Καταχωρούμε στη θέση 2 του πίνακα Accel_Vector την τιμή του αξελερόμετρου για τον άξονα z

Accel_Vector[2] = accel_z;

//Καταχωρούμε στον πίνακα Omega στην θέση 0 το άθροισμα των διανυσμάτων Gyro_Vector[0] και Omega_I[0] (προσθέτουμε τον αναλογικό όρο)

Vector_Add(&Omega[0], &Gyro_Vector[0], &Omega_I[0]);

```
  //Καταχωρούμε στον πίνακα Omega στην θέση 0 το άθροισμα των διανυσμάτων Omega[0] και Omega_P[0] (προσθέτουμε τον ολοκληρώτικο όρο)
  Vector_Add(&Omega_Vector[0], &Omega[0], &Omega_P[0]);

  //Καταχωρούμε στον πίνακα Update_Matrix τις διορθωμένες τιμές του γυροσκοπίου από τον πίνακα Omega_Vector πολλαπλασιασμένες με χρόνο ολοκλήρωσης ενός κύκλου προγράμματος G_Dt

  Update_Matrix[0][0] = 0;
  Update_Matrix[0][1] = -G_Dt * Omega_Vector[2]; // (αφορά τον αξονα -z) (αφορά τον αξονα y)
  Update_Matrix[0][2] = G_Dt * Omega_Vector[1]; //(αφορά τον αξονα y)
  Update_Matrix[1][0] = G_Dt * Omega_Vector[2]; //(αφορά τον αξονα z)
  Update_Matrix[1][1] = 0;
  Update_Matrix[1][2] = -G_Dt * Omega_Vector[0]; //(αφορά τον αξονα -x)
  Update_Matrix[2][0] = -G_Dt * Omega_Vector[1]; //(αφορά τον αξονα -y)
  Update_Matrix[2][1] = G_Dt * Omega_Vector[0]; //(αφορά τον αξονα x)
  Update_Matrix[2][2] = 0;

  //Καλούμε την συνάρτηση Matrix_Multiply υπολογίζει το γινόμενο των πινάκων Update_Matrix και Temporary_Matrix
  Matrix_Multiply(DCM_Matrix, Update_Matrix, Temporary_Matrix);

  //Δομή επανάληψης που χρησιμοποιείται για τις γραμμές των πινάκων DCM_Matrix και Temporary_Matrix
  for (int x = 0; x < 3; x++) //Matrix Addition (update)
```

```
  { //Δομή επανάληψης που χρησιμοποιείται για τις στήλες των πινάκων DCM_Matrix και Temporary_Matrix
    for (int y = 0; y < 3; y++)
    {
      //Προσθέτουμε σε κάθε στοιχείο του πίνακα DCM_Matrix το αντίστοιχο στοιχείο του Temporary_Matrix
      DCM_Matrix[x][y] += Temporary_Matrix[x][y];
    }
  }
}



//Η συνάρτηση Euler_angles υπολογίζει τον  διατοιχισμό (pitch),την πρόνευση (roll) και την εκτροπή (yaw) του σκάφους και της πυξίδας αντίστοιχα από τις διορθωμένες τιμές του DCM αλγορίθμου
void Euler_angles(void)
{
  //Υπολογίζουμε τον διατοιχισμό μέσω της συνάρτησης του αντίστροφου ημιτόνου σε rad
  pitch = -asin(DCM_Matrix[2][0]);
  //Υπολογίζουμε την πρόνευση από το τόξο της εφαπτομένης των δύο παρακάτω στοιχείων του πίνακα DCM_Matrix
  roll = atan2(DCM_Matrix[2][1], DCM_Matrix[2][2]);
  //Υπολογίζουμε την εκτροπή από το τόξο της εφαπτομένης των δύο παρακάτω στοιχείων του πίνακα DCM_Matrix
  yaw = atan2(DCM_Matrix[1][0], DCM_Matrix[0][0]);
}
```

```
//Η συνάρτηση Vector_Dot_Product υπολογίζει το εσωτερικό γινόμενο των δύο διανυσμάτων
float Vector_Dot_Product(float vector1[3], float vector2[3])
{ //Δηλώνουμε την μεταβλητή op που έχει ως σκοπό την καταχώρηση την υπολογισμένης τιμής
  float op = 0;
  //Δομή επανάληψης που χρησιμοποιείται για τον υπολογισμό του γινομένου και για τα τρία στοιχεία των πινάκων
  for (int c = 0; c < 3; c++)
  {
    //Προσθέτουμε στην μεταβλητή op το αποτέλεσμα του γινομένου
    op += vector1[c] * vector2[c];
  }
  //Επιστρέφουμε το αποτέλεσμα του γινομένου
  return op;
}

//Η συνάρτηση Vector_Cross_Product έχει ως σκοπό τον υπολογισμό του εξωτερικού γινομένου δύο διανυσμάτων
void Vector_Cross_Product(float vectorOut[3], float v1[3], float v2[3])
{
  //Καταχωρούμε για κάθε στοιχείο του πίνακα vectorOut το αποτέλεσμα της πράξης
```

```
  vectorOut[0] = (v1[1] * v2[2]) - (v1[2] * v2[1]);
  vectorOut[1] = (v1[2] * v2[0]) - (v1[0] * v2[2]);
  vectorOut[2] = (v1[0] * v2[1]) - (v1[1] * v2[0]);
}
```

```
//Η συνάρτηση Vector_Scale υπολογίζει το γινόμενο ενός διανύσματος με έναν πραγματικό αριθμό
void Vector_Scale(float vectorOut[3], float vectorIn[3], float scale2)
{ //Δομή επανάληψης που χρησιμοποιείται για τον υπολογισμό του γινομένου και για τα τρία στοιχεία των πινάκων
  for (int c = 0; c < 3; c++)
  {
    //Καταχωρούμε στον πίνακα vectorOut το αποτέλεσμα του γινομένου
    vectorOut[c] = vectorIn[c] * scale2;
  }
}
```

```
//Η συνάρτηση Vector_Add υπολογίζει το άθροισμα δυο διανυσμάτων
void Vector_Add(float vectorOut[3], float vectorIn1[3], float vectorIn2[3])
{
  //Δομή επανάληψης που χρησιμοποιείται για τον υπολογισμό του αθροίσματος και για τα τρία στοιχεία των πινάκων
  for (int c = 0; c < 3; c++)
```

```
  {
    //Καταχωρούμε στον πίνακα vectorOut το αποτέλεσμα του αθροίσματος
    vectorOut[c] = vectorIn1[c] + vectorIn2[c];
  }
}
```

```
//Η συνάρτηση Matrix_Multiply υπολογίζει το γινόμενο δυο πινάκων τριών γραμμών και τριών στηλών
void Matrix_Multiply(float a[3][3], float b[3][3], float mat[3][3])
{ //Δομή επανάληψης που χρησιμοποιείται ως δείκτης των γραμμών των πινάκων mat και a
  for (int x = 0; x < 3; x++)
  {
    //Δομή επανάληψης που χρησιμοποιείται ως δείκτης των στηλών των πινάκων mat και b
    for (int y = 0; y < 3; y++)
    {
      //Αρχικοποιούμε τον πίνακα mat καταχωρώντας σε όλες τις θέσεις του την τιμή μηδέν
      mat[x][y] = 0;

      //Δομή επανάληψης που χρησιμοποιείται ως δείκτης της γραμμής του πίνακα b και στήλης του πίνακα a
      for (int w = 0; w < 3; w++)
      {
        mat[x][y] += a[x][w] * b[w][y];
```

```
    }
   }
  }
}
```

ΚΑΡΤΕΛΑ POTSEERING

```
//Η μεταβλητή potpin χρησιμοποιείται για την καταχώριση του Pin που συνδέεται το ποτενσιόμετρο
int potpin = 0;
//Στη μεταβλητη val καταχωρειται η τιμη του ποτενσιομέτρου
int val;

//Η συνάρτηση potSteering καλείται όταν πλοηγούμε το σκάφος από το ποτενσιόμετρο
void potSteering()
{
 //Διαβάζουμε την τιμή του ποτενσιόμετρου (παίρνει τιμή από 0 έως 1023)
 val = analogRead(potpin);

 //Μετατρέπουμε την τιμή του ποτενσιόμετρου για να μπορεί να χρησιμοποιηθεί από το servo
 val = map(val, 0, 1023, 60, 120);
```

```
  //Αν η τιμή των τρεχόντων millisecond είναι μεγαλύτερη ή ίση με την μεταβλητή servo_delay
  if (millis() >= servo_delay) {
    //Μεταβάλουμε την θέση του σερβοκινητήρα μέσω της μεταβλητής OutputTemp
    myservo.write(val);
    //Καταχωρούμε στην μεταβλητή servo_delay την τιμή των τρεχόντων millisecond συν 15 millisecond για τον έλεγχο του σερβοκινητήρα
    servo_delay = millis() + 15;

  }

}
```

ΚΑΡΤΕΛΑ LCD_KEYPAD

```
//Δήλωση πίνακα για την εισαγωγή επιθυμητής πορείας από το πληκτρολόγιο
char inData[] = {0, 0, 0, 0};
//Δείκτης που χρησιμεύει στον προσδιορισμό της θέσης του πίνακα
int i = 0;
//Μεταβλητή για την καταχώριση της τελευταίας φόρας που ενημερώθηκε η γραμμή 0 της οθόνης (έχει ως σκοπό την προβολή πληροφοριών στην γραμμή 0 της οθόνης χωρίς την χρήση delay)
unsigned long previousMillisl0 = 0;
```

//Μεταβλητή για την καταχώριση της επόμενης φόρας που πρέπει να ενημερωθεί η γραμμή 1 της οθόνης (έχει ως σκοπό την προβολή πληροφοριών στην γραμμή 1 της οθόνης χωρίς την χρήση delay)

unsigned long nextMillisl1 = 0;

//Μεταβλητή για την καταχώριση της επόμενης φόρας που πρέπει να ενημερωθεί η γραμμή 2 της οθόνης (έχει ως σκοπό την προβολή πληροφοριών στην γραμμή 2 της οθόνης χωρίς την χρήση delay)

unsigned long nextMillisl2 = 0;

//Μεταβλητή που χρησιμοποιείται για τον καθορισμό της χρονικής καθυστέρησης που αφορά την γραμμή 0 της οθόνης χαρακτήρων

int interval0 = 1000;

//Μεταβλητή που χρησιμοποιείται για τον καθορισμό της χρονικής καθυστέρησης που αφορά την γραμμή 1 της οθόνης χαρακτήρων

int interval1 = 1000;

//Μεταβλητή που χρησιμοποιείται για τον καθορισμό της χρονικής καθυστέρησης που αφορά την γραμμή 2 της οθόνης χαρακτήρων

int interval2 = 2000;

//Καταχωρούμε στην μεταβλητή currentMillis την τιμή των τρεχόντων millisecond

unsigned long currentMillis = millis();

//Η μεταβλητή pBuzzerMillis χρησιμοποιείται για την καταχώρηση της χρονικής στιγμής που ενεργοποιήθηκε το buzzer

unsigned long pBuzzerMillis = 0;

//Μεταβλητή που χρησιμοποιείται για τον καθορισμό της χρονικής καθυστέρησης για την χρήση του Buzzer δίχως την χρήση delay

int BuzzerInterval = 800;

//Δήλωση της λογικής μεταβλητής once ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 0 της οθόνης

bool once = false;

//Δήλωση της λογικής μεταβλητής oncel1 ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 1 της οθόνης

bool oncel1 = true;

//Δήλωση της λογικής μεταβλητής oncel1A ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 1 της οθόνης κατα της επιλογή Auto

bool oncel1A = false;

//Δήλωση της λογικής μεταβλητής oncel1M ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 1 της οθόνης κατα της επιλογή Manual

bool oncel1M = false;

//Δήλωση της λογικής μεταβλητής oncel1Nmea ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 1 της οθόνης όταν πλοηγούμαστε απο το plotter

bool oncel1Nmea = false;

//Δήλωση της λογικής μεταβλητής oncel1Warn ως ψευδής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 1 της οθόνης κατά τη αναγγελία ειδοποίησης ότι χάθηκε η επικοινωνία με το plotter

bool oncel1Warn = false;

//Δήλωση της λογικής μεταβλητής oncel2a και oncel2b ως αληθής που χρησιμεύει ως σημαία για την αποφυγή τρεμοπαίγματος στην γραμμή 2 της οθόνης

bool oncel2a = true;

bool oncel2b = true;

//Η Function lcdKeypad εχει ως αντικείμενο τον έλεγχο της οθόνης και του πληκτρολόγιου

void lcdKeypad(void)

```
{


  //Αν ο δείκτης i είναι μηδέν (δεν έχει πληκτρολογηθεί από τον χρήστη) και η σημαία once είναι ψευδής (είναι πρώτη φορά εμφάνισης της πρώτης γραμμής της οθόνης)
  if (i == 0 && (once == false)) {
    //Αν η σημαία nmeapiloting είναι ψευδής (δεν έχουμε ορίσει μια πορεία ή διαδρομή από το plotter) και oncel1Warn είναι ψευδής (δεν έχει προβληθεί μήνυμα σφάλματος στην οθόνη)
    if (nmeapiloting == false && oncel1Warn == false) {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
      lcd.setCursor(0, 1);
      //Εμφανιζουμε το κείμενο στην οθόνη χαρακτήρων
      lcd.print("Type disere heading");
    }

  }

  //Αν η σημαία nmeapiloting είναι ψευδής (δεν έχουμε ορίσει μια πορεία ή διαδρομή από το plotter) και η σημαία nmeaWarning ψευδής (δεν υπάρχει σφάλμα στην σύνδεση με το plotter)
  if (nmeapiloting == false && nmeaWarning == false) {
    //Καταχωρούμε στην μεταβλητή key ότι τιμή λάβουμε από το πληκτρολόγιο
    char key = keypad1.getKey();



    switch (key)
```

```
{
 //Σε περίπτωση που δεν πατήθηκε κάποιο πλήκτρο από τον χρήστη
 case NO_KEY:
  //Διακόπτουμε την δομή σύγκρισης
  break;

 //Σε περίπτωση που πληκτρολογηθεί κάποιος αριθμητικός χαρακτήρας από το 0 εως 9
 case '0': case '1': case '2': case '3': case '4':
 case '5': case '6': case '7': case '8': case '9': {
    //Καταχωρούμε την τιμή στον πίνακα inData
    inData[i] = key;
    //Αυξάνουμε την τιμή του δείκτη i κατα 1
    i++;

    //Αν τιμή του δείκτη i είναι 1
    if (i == 1) {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
      lcd.setCursor(0, 1);
      //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
      lcd.print("You typed:         ");
    }
    //Στη συνέχεια Θέτουμε τον κέρσορα στη στήλη 9 + i  και στην 1 γραμμή
    lcd.setCursor((9 + i), 1);
    // Και εμφανίζουμε τον αριθμητικό χαρακτήρα που πληκτρολογήσαμε
```

```
        lcd.print(key);
      }
      //Διακόπτουμε την δομή σύγκρισης
      break;

    //Σε περίπτωση που πληκτρολογηθεί ο χαρακτήρας # (Enter)
    case '#': {
        //Αν ο δείκτης i είναι διάφορος του μηδενός (έχει πληκτρολογηθεί τουλάχιστον ένα ψηφίο)
        if (i!=0){
      //Καταχωρούμε στον δείκτη i την τιμή 3
      i = 3;
       }
    }
    //Διακόπτουμε την δομή σύγκρισης
    break;

    case 'C': {
       analogWrite(9, 55);
     }
     break;

     case 'D': {
       analogWrite(9, 0);
     }
```

```
    break;


  //Σε περίπτωση που πληκτρολογηθεί ο χαρακτήρας * (Delete)
  case '*': {
    //Εαν ο δείκτης i είναι ίσος με 0
    if (i == 0) {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
      lcd.setCursor(0, 2);
      //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
      lcd.print("Nothing to delete!  ");
      //Καταχωρούμε στην μεταβλητή nextMillisl2 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval2 για την εμφάνιση του παραπάνω μηνήματος για προκαθορισμένο χρόνο
      nextMillisl2 = (millis()) + interval2;
      //Καταχωρούμε στη λογική μεταβλητή oncel1 την τιμή ψευδής
      oncel1 = false;
      //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή ψευδής
      oncel2a = false;

    }


    //Αλλιώς
    else {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
      lcd.setCursor(0, 2);
      //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
      lcd.print("Heading deleted!");
```

```
        //Καταχωρούμε στην μεταβλητή nextMillisl2 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval2 για την εμφάνιση του παραπάνω μηνήματος για προκαθορισμένο χρόνο
        nextMillisl2 = (millis()) + interval2;
        //Καταχωρούμε στον δείκτη i την τιμή 0
        i = 0;
        //Καταχωρούμε στη λογική μεταβλητή once την τιμή ψευδής
        once = false;
        //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή ψευδής
        oncel2a = false;
        //Αρχικοποιούμε τον πίνακα καταχωρώντας σε όλες τις θέσεις του την τιμή 0
        for (int j = 0; j < 3; j++) {
          inData[j] = 0;
        }
       }
      }

   }
  }


  //Εαν ο δείκτης i είναι ίσος με 3
  if (i == 3)
  {
   //Μετατρέπουμε τον πίνακα inData σε ακέραιο αριθμό και τον καταχωρούμε στη μεταβλητή keypadSetpoint
   keypadSetpoint = atoi(inData);
   //Καταχωρούμε στον δείκτη i την τιμή 0
   i = 0;
   //Καταχωρούμε στη λογική μεταβλητή once την τιμή αληθής
   once = true;
   //Καταχωρούμε στη λογική μεταβλητή oncel1 την τιμή ψευδής
   oncel1 = false;
```

```
  //Καταχωρούμε στην μεταβλητή nextMillisl1 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval1 για την εμφάνιση του μηνήματος στην πρώτη γραμμή για προκαθορισμένο χρόνο
  nextMillisl1 = (millis()) + interval1;
  //Αρχικοποιούμε τον πίνακα καταχωρώντας σε όλες τις θέσεις του την τιμή 0
  for (int j = 0; j < 3; j++) {
   inData[j] = 0;
  }


  //Αν η τιμή που πληκτρολογήθηκε είναι μεγαλύτερη από 360 μοίρες (εμφανίζουμε μήνυμα λάθους)
  if (keypadSetpoint > 360) {
   //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
   lcd.setCursor(0, 2);
   //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
   lcd.print("Number from 0 to 360");
   //Καταχωρούμε στην μεταβλητή nextMillisl2 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval2 για την εμφάνιση του παραπάνω μηνήματος για προκαθορισμένο χρόνο
   nextMillisl2 = (millis()) + interval2;
   //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή ψευδής
   oncel2a = false;
  }

  //Αλλιως αν η τιμή που πληκτρολογήθηκε είναι ίση 360 μοίρες
  else if (keypadSetpoint == 360) {
   //Καταχωρούμε στη μεταβλητή keypadSetpoint την τιμή 0 (καθώς οι 360 μοίρες ισούνται με 0 μοίρες)
   keypadSetpoint = 0;
   //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
   lcd.setCursor(0, 2);
   //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
   lcd.print("Heading stored      ");
```

```
      //Καταχωρούμε στην μεταβλητή nextMillisl2 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval2 για την εμφάνιση του παραπάνω μηνήματος για προκαθορισμένο χρόνο
      nextMillisl2 = (millis()) + interval2;
      //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή ψευδής
      oncel2a = false;
      //Καταχωρούμε στη λογική μεταβλητή kpdheading την τιμή αληθής
      kpdheading = true;

    }

    //Αλλιώς
    else {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
      lcd.setCursor(0, 2);
      //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
      lcd.print("Heading stored      ");
      //Καταχωρούμε στην μεταβλητή nextMillisl2 την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής interval2 για την εμφάνιση του παραπάνω μηνήματος για προκαθορισμένο χρόνο
      nextMillisl2 = (millis()) + interval2;
      //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή ψευδής
      oncel2a = false;
      //Καταχωρούμε στη λογική μεταβλητή kpdheading την τιμή αληθής
      kpdheading = true;
    }
  }


  //Καταχωρούμε στην μεταβλητή currentMillis την τιμή των τρεχόντων millisecond
  currentMillis = millis();
```

```
  //Αν η διαφορά μεταξύ των τρεχόντων millisecond και της τιμής της μεταβλητής previousMillisl0 είναι μεγαλύτερη ή ίση με την μεταβλητή interval0 (χρονική καθυστέρηση της γραμμής 0)
  if (currentMillis - previousMillisl0 >= interval0) {
    //Καταχωρούμε στην μεταβλητή previousMillisl0 την τιμή των τρεχόντων millisecond
    previousMillisl0 = currentMillis;



    //Εάν η σημαία nmeapiloting είναι αληθής (καθοδηγούμαστε από το plotter)
    if (nmeapiloting == true) {
        //Θέτουμε τον κέρσορα στη στήλη 0 και στην 3 γραμμή
    lcd.setCursor(0, 3);
    //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
    lcd.print("Course: ");
     //Εμφανίζουμε το περιεχόμενο της μεταβλητής gpscourse στην οθόνη χαρακτήρων
     lcd.print(gpscourse);
     //Και στην συνέχεια εμφανίζουμε το παρακάτω κείμενο
     lcd.print(" GPS  ");
    }

    //Αλλιώς
    else {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 3 γραμμή
    lcd.setCursor(0, 3);
    //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
    lcd.print("Heading:");
     //Εμφανίζουμε το περιεχόμενο της συνάρτησης compass στην οθόνη χαρακτήρων
     lcd.print(compass());
     //Και στην συνέχεια εμφανίζουμε το παρακάτω κείμενο που αφαιρεί χαρακτήρες που έχουν παραμείνει από προηγούμενο μήνυμα
      lcd.print("      ");

    }
```

```
  }


  //Εάν η λογική μεταλητή nmeapiloting είναι αληθής (καθοδηγούμαστε από το plotter)
  if (nmeapiloting == true) {
    //Θέτουμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
    lcd.setCursor(0, 1);
    //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων (διαγράφουμε τους χαρακτήρες που έχουν παραμείνει από προηγούμενο μήνυμα)
    lcd.print("                ");
    //Καταχωρούμε στη λογική μεταβλητή once την τιμή ψευδής
    once = false;
  }

  //Αλλιώς
  else {
    //Αν η μεταβλητή currentMillis είναι μεγαλύτερη ή ίση με την μεταβλητή nextMillisl1 και η σημαία oncel1 είναι ψευδής και η σημαία oncel1Warn είναι ψευδής
    if (currentMillis  >= nextMillisl1 && oncel1 == false && oncel1Warn == false) {
      //Θέτουμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
      lcd.setCursor(0, 1);
      //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
      lcd.print("Type disere heading");
      //Καταχωρούμε στη λογική μεταβλητή oncel1 την τιμή αληθής
      oncel1 = true;

    }
  }

  //Αν η μεταβλητή currentMillis είναι μεγαλύτερη ή ίση με την μεταβλητή nextMillisl2 και η λογική μεταβλητή oncel2a είναι ψευδής
  if (currentMillis  >= nextMillisl2 && oncel2a == false) {
    //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
    lcd.setCursor(0, 2);
```

```
  //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων (ουσιαστίκα διαγράφουμε τους χαρακτήρες που έχουν παραμείνει από προηγούμενο μήνυμα)
  lcd.print("                ");
  //Καταχωρούμε στη λογική μεταβλητή oncel2a την τιμή αληθής
  oncel2a = true;
  //Καταχωρούμε στη λογική μεταβλητή oncel2b την τιμή ψευδής
  oncel2b = false;

 }

 //Εάν η λογική μεταλητή nmeapiloting είναι αληθής (καθοδηγούμαστε από το plotter)
 if (nmeapiloting == true) {
  //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
  lcd.setCursor(0, 2);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
  lcd.print("Setpoint:");
  //Εμφανίζουμε το περιεχόμενο της συνάρτησης compass στην οθόνη χαρακτήρων
  lcd.print(apheading);
  //Ακολούθως εμφανίζουμε το παρακάτω κείμενο στην οθόνη
  lcd.print (" GPS  ");

 }
 //Αλλιώς
 else {

  //Εάν η λογική μεταλητή oncel2b είναι ψευδής
  if (oncel2b == false) {
   //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
   lcd.setCursor(0, 2);
   //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
   lcd.print("Setpoint:");
   //Εμφανίζουμε το περιεχόμενο της συνάρτησης Setpoint στην οθόνη χαρακτήρων
   lcd.print(Setpoint);
   //Καταχωρούμε στη λογική μεταβλητή oncel2b την τιμή αληθής
```

```
      oncel2b = true;

    }
  }

  //Εάν η λογική μεταλητή oncel2a είναι αληθής και η μεταβλητή buttonheading είναι επίσης αληθής
  if (oncel2a == true && buttonheading == true) {
    //Θέτουμε τον κέρσορα στη στήλη 0 και στην 2 γραμμή
    lcd.setCursor(0, 2);
    //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
    lcd.print("Setpoint:");
    //Εμφανίζουμε το περιεχόμενο της συνάρτησης Setpoint στην οθόνη χαρακτήρων
    lcd.print(Setpoint);
    //Καταχωρούμε στη λογική μεταβλητή buttonheading την τιμή ψευδής
    buttonheading = false;
  }


  //Αν η μεταβλητή nmeaWarning είναι αληθής (έχει χαθεί η σύνδεση με το plotter)
  if (nmeaWarning == true) {
    //Καταχωρούμε στην μεταβλητή key ότι τιμή λάβουμε από το πληκτρολόγιο
    char key = keypad1.getKey();
    //Αν η διαφορά μεταξύ των τρεχόντων millisecond και της τιμής της μεταβλητής pBuzzerMillis είναι μεγαλύτερη ή ίση με την μεταβλητή BuzzerInterval
    if (currentMillis - pBuzzerMillis >= BuzzerInterval) {
      //Καταχωρούμε στην μεταβλητή pBuzzerMillis την τιμή των τρεχόντων millisecond
      pBuzzerMillis = millis();
      //Ενεργοποιούμε το buzzer που συνδέεται στο Pin 8 με συχνότητα 720 hertz και διάρκεια 500 milliseconds
      tone(8, 720, 500);

    }
```

```
  //Αν έχει πληκτρολογηθεί ο χαρακτήρας A ακυρώνουμε την αναγγελία σφάλματος στην επικοινωνία με το plotter
  if (key == 'A') {
    //Καταχωρούμε στη λογική μεταβλητή nmeaWarning την τιμή ψευδής
    nmeaWarning = false;
    //Καταχωρούμε στη λογική μεταβλητή oncel1Warn την τιμή ψευδής
    oncel1Warn = false;
    //Απενεργοποιούμε το buzzer
    noTone(8);
  }

 }



 //Αν η σημαία nmeaWarning είναι αληθής και ο διακόπτης είναι στη θέση Auto και η σημαία oncel1Warn είναι ψευδής
 if (nmeaWarning == true && modeState == HIGH && oncel1Warn == false) {
  //Θέτουμε τον κέρσορα στη στήλη 0 και στην 0 γραμμή
  lcd.setCursor(0, 0);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων (καθαρίζουμε την 0 γραμμή)
  lcd.print("                ");
  //Τοποθετούμε ξανά τον κέρσορα στη στήλη 0 και στην 0 γραμμή
  lcd.setCursor(0, 0);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
  lcd.print("WARNING! LOST NMEA");
  //Τοποθετούμε τον κέρσορα στη στήλη 0 και στην 1 γραμμή
  lcd.setCursor(0, 1);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
  lcd.print("PRESS 'A' TO EXIT");
  //Δήλωση της λογικής μεταβλητής oncel1A ως ψευδής
  oncel1A = false;
  //Καταχωρούμε τη λογική μεταβλητής oncel1M ως ψευδής
  oncel1M = false;
```

```
  //Καταχωρούμε τη λογική μεταβλητής oncel1Nmea ως ψευδής
  oncel1Nmea = false;
  //Καταχωρούμε τη λογική μεταβλητή oncel1Nmea ως αληθής
  oncel1Warn = true;



}



 //Αν η σημαία nmeapiloting είναι αληθής (καθοδιγούμαστε από το plotter) και η λογική μεταβλητή oncel1Nmea είναι ψευδής (είναι η πρώτη φορά εμφάνισης του μηνύματος στην γραμμή 1) και ο διακόπτης είναι στη θέση Auto
 if (nmeapiloting == true && oncel1Nmea == false && modeState == HIGH) {
  //Θέτουμε τον κέρσορα στη στήλη 0 και στην 0 γραμμή
  lcd.setCursor(0, 0);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων (καθαρίζουμε την 0 γραμμή)
  lcd.print("                ");
  //Τοποθετούμε ξανά τον κέρσορα στη στήλη 0 και στην 0 γραμμή
  lcd.setCursor(0, 0);
  //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
  lcd.print("Mode:");
  //Ακολούθως εμφανίζουμε το κείμενο
  lcd.print("PLOTER (NMEA)");
  //Καταχωρούμε τη λογική μεταβλητή oncel1A ως ψευδής
  oncel1A = false;
  //Καταχωρούμε τη λογική μεταβλητή oncel1M ως ψευδής
  oncel1M = false;
  //Καταχωρούμε τη λογική μεταβλητή oncel1Nmea ως αληθής
  oncel1Nmea = true;

  //Αρχικοποιούμε τον πίνακα καταχωρώντας σε όλες τις θέσεις του την τιμή 0
  for (int j = 0; j < 3; j++) {
```

```
    inData[j] = 0;
   }
   //Καταχωρούμε στον δείκτη i την τιμή 0
   i = 0;

 }
```

//Αν ο διακόπτης είναι στη θέση Auto και η σημαία oncel1A είναι ψευδής (είναι η πρώτη φορά εμφάνισης του μηνύματος στην γραμμή 1) και η λογική μεταβλητή nmeapiloting είναι ψευδής (δεν καθοδιγούμαστε από το plotter) και η σημαία nmeaWarning είναι ψευδής (δεν ύπαρχει σφάλμα με την NMEA σύνδεση)

```
 if (modeState == HIGH && oncel1A == false && nmeapiloting == false && nmeaWarning == false ) {
   //Θέτουμε τον κέρσορα στη στήλη 0 και στην 0 γραμμή
   lcd.setCursor(0, 0);
   //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων (καθαρίζουμε την 0 γραμμή)
   lcd.print("                ");
   //Τοποθετούμε ξανά τον κέρσορα στη στήλη 0 και στην 0 γραμμή
   lcd.setCursor(0, 0);
   //Εμφανίζουμε το παρακάτω κείμενο στην οθόνη χαρακτήρων
   lcd.print("Mode:");
   //Ακολούθως εμφανίζουμε το κείμενο
   lcd.print("COMPASS HEADING");
   //Καταχωρούμε τη λογική μεταβλητή oncel1A ως αληθής
   oncel1A = true;
   //Καταχωρούμε τη λογική μεταβλητή oncel1M ως ψευδής
   oncel1M = false;
   //Καταχωρούμε τη λογική μεταβλητή oncel1Nmea ως ψευδής
   oncel1Nmea = false;
 }
```

```
  //Αν ο διακόπτης είναι στη θέση Manual και η λογική μεταβλητή oncel1M είναι ψευδής (είναι η πρώτη φορά εμφάνισης του μηνύματος στην γραμμή 1)
  else if (modeState == LOW && oncel1M == false) {
    //Θέτουμε τον κέρσορα στη στήλη 0 και στην 0 γραμμή
    lcd.setCursor(0, 0);
    //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων (καθαρίζουμε την 0 γραμμή)
    lcd.print("                ");
    //Τοποθετούμε ξανά τον κέρσορα στη στήλη 0 και στην 0 γραμμή
    lcd.setCursor(0, 0);
    //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
    lcd.print("Mode:");
    //Εμφανίζουμε το κείμενο στην οθόνη χαρακτήρων
    lcd.print("MANUAL");
    //Καταχωρούμε τη λογική μεταβλητή oncel1M ως αληθής
    oncel1M = true;
    //Καταχωρούμε τη λογική μεταβλητή oncel1A ως ψευδής
    oncel1A = false;
    //Καταχωρούμε τη λογική μεταβλητή oncel1Nmea ως ψευδής
    oncel1Nmea = false;
    //Καταχωρούμε τη λογική μεταβλητή nmeaWarning ως ψευδής
    nmeaWarning = false;
    //Καταχωρούμε τη λογική μεταβλητή oncel1Warn ως ψευδής
    oncel1Warn = false;
  }


}
```

ΚΑΡΤΕΛΑ NMEA

```
//Μεταβλητή nmeaInterval χρησιμοποιείται για τον καθορισμό του μέγιστου επιτρεπτού χρονικού διαστήματος κατα το οποίο έχει "χαθεί" η σειριακή επικοινωνία με το plotter
int nmeaInterval = 3000;
//Δήλωση της μεταβλητής nmeaNextMillis που χρησιμοποιείτα ως χρονικό
```

```
unsigned long nmeaNextMillis = 0;
//Δήλωση του μονοδιάστατου πίνακα c
char c[] = "F";
//Δήλωση του μονοδιάστατου πίνακα d
char d[] = "F";
//Δήλωση μεταβλητής arrivalCircle
float arrivalCircle;

void nmeafc () {


  //Ενώ ύπαρχουν χαρακτήρες διαθέσιμοι για να διαβαστούν στην σειριακή πόρτα 1
  while (Serial1.available() > 0) {

    //Αποκωδικοποιούμε τους εισερχόμενους χαρακτήρες
    nmea.encode(Serial1.read());
    //Καταχωρούμε στην μεταβλητή nmeaNextMillis την τιμή των τρεχόντων millisecond συν την τιμή της μεταβλητής nmeaInterval
    nmeaNextMillis = millis() + nmeaInterval;
    //Καταχωρούμε στη λογική μεταβλητή nmeaWarning την τιμή ψευδής
    nmeaWarning = false;
    //Καταχωρούμε στη λογική μεταβλητή oncel1Warn την τιμή ψευδής
    oncel1Warn = false;

     arrivalCircle= strtod((arrivalCirclenmea.value()),NULL);
    //Η συνάρτηση autopheading.value() επιστρέφει μια φράση που την μετατρέπουμε μέσω της εντολής strtod σε πραγματικό αριθμό και τον καταχωρούμε στην μεταβλητή apheading
    apheading = strtod((autopheading.value()), NULL);
    //Καταχωρούμε στην μεταβλητή gpscourse ότι επιστρέφει η συνάρτηση nmea.course.deg() , την πορεία δηλαδή του σκάφους υπολογισμένη από το plotter
    gpscourse = nmea.course.deg();
```

```
  //Η συνάρτηση autopmode.value() επιστρέφει μια φράση που την μετατρέπουμε μέσω της εντολής strcpy σε έναν χαρακτήρα που τον καταχωρούμε στον πίνακα c (ο χαρακτήρας αυτός υποδεικνύει εάν μια πορεία ή διαδρομή από το plotter είναι διαθέσιμη
  strcpy(c, autopmode.value());
 Serial.println(arrivalCirclenmea.value());
  //Αν η μεταβλητή c είναι ίση με A (άρα μια πορεία ή διαδρομή από το plotter είναι διαθέσιμη)
  if (*c == 'A') {
    //Καταχωρούμε στη λογική μεταβλητή nmeapiloting την τιμή αληθής
    nmeapiloting = true;
  }

  //Αλλιώς αν η μεταβλητή c είναι ίση με N (οπότε δεν υπάρχει μια πορεία ή διαδρομή από το plotter)
  else if (*c == 'N') {
    //Καταχωρούμε στη λογική μεταβλητή nmeapiloting την τιμή ψευδής
    nmeapiloting = false;
  }

  //Η συνάρτηση arrivalAlarmnmea.value() επιστρέφει μια φράση που την μετατρέπουμε μέσω της εντολής strcpy σε έναν χαρακτήρα που τον καταχωρουμε στον πινακα d (ο χαρακτήρας αυτός υποδεικνύει εάν έχουμε φτάσει στον προορισμό μας)
  strcpy(d, arrivalAlarmnmea.value());


  //Αν η μεταβλητή d είναι ίση με A (οπότε δεν υπάρχει μια πορεία ή διαδρομή από το plotter)
  if (*d == 'A') {
    //Καταχωρούμε στη λογική μεταβλητή nmeapiloting την τιμή ψευδής
    nmeapiloting = false;
  }

 }
```

```
  //Αν η μεταβλητή nmeapiloting είναι αλήθης και η τιμή των τρέχοντων millisecond είναι μεγαλύτερη απο την μεταβλητή nmeaNextMillis
  if ((nmeapiloting == true ) && millis() > nmeaNextMillis) {
    //Καταχωρούμε στη λογική μεταβλητή nmeapiloting την τιμή ψευδής
    nmeapiloting = false;
    //Καταχωρούμε στη λογική μεταβλητή nmeaWarning την τιμή αληθής
    nmeaWarning = true;

  }


}
```

ΚΑΡΤΕΛΑ PLOT

```
//Συνάρτηση εμφάνισης στην γραφική παράσταση σημαντικών πληροφοριών
void plot() {
 //Εμφανίζουμε στην γραφική παράσταση την επιθυμητή διεύθυνση που έχουμε επιλέξει (μπλέ χρώμα)
  Serial.print (Setpoint);
  Serial.print(" ");
  //Εμφανίζουμε την τρέχουσα τιμή της πορείας του σκάφους (κόκκινο χρώμα)
  Serial.print (Input);
  Serial.print(" ");
  //Εμφανίζουμε την τιμή εξόδου του ελεγκτή που ελέγχει την θέση του σερβοκινητήρα (πράσινο χρώμα)
  Serial.println (OutputTemp);
}
```

ΚΕΦΑΛΑΙΟ 4

Συμπεράσματα

Η παρούσα εργασία μέσα από τον συνδυασμό θεωρίας και πράξης μας έδωσε τη δυνατότητα να εξοικειωθούμε με την σχεδίαση και κατασκευή ενός ολοκληρωμένου συστήματος πλοήγησης. Αναλυτικότερα εμβαθύναμε τις γνώσεις μας όσον αφορά τα πρωτόκολλα επικοινωνίας όπως το NMEA (National Marine Electronics Association) που χρησιμοποιείται από όλες την ηλεκτρονικές συσκευές στην ναυτιλία για την μεταξύ τους διασύνδεση. Άλλο ένα πρωτόκολλο που χρησιμοποιήσαμε είναι το I^2C που είναι ευρέως διαδεδομένο για την διασύνδεση περιφεριακών ηλεκτρονικών συσκευών. Ασχοληθήκαμε με τα συστήματα διεπαφής ,τα αποκαλούμενα UI (user interface), που έχουν ως σκοπό τον όσο το δυνατόν φιλικότερο προς τον χρήστη αλλά και αποτελεσματικότερο έλεγχο μιας μηχανής, (στην περίπτωση μας του Αυτόματου Πιλότου), που επιτυγχάνουν συγχρόνως την άμεση προειδοποίηση του χρήστη σε περίπτωση δυσλειτουργίας. Στην υπάρχουσα περίπτωση είναι αποτελούμενο από μια LCD οθόνη, ένα πληκτρολόγιο και έναν βομβητή. Επίσης αποκτήσαμε γνώσεις όσο αναφορά τον τρόπο σχεδίασης και κατασκευής ηλεκτρονικών πλακετών. Συμπερασματικά, θεωρούμε πως έχοντας στα χέρια μας μια κατασκευή που μπορεί να λειτουργήσει υπό κανονικές συνθήκες έχουμε πετύχει τον στόχο μας. Σαφώς μπορεί να αποτελέσει και αντικείμενο περαιτέρω έρευνας και εξέλιξης στο μέλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Toghill, Jeff E. Celestial Navigation, New York, W.W. Norton & Co. 1988

2. https://www.weems-plath.com

3. Billur Barshan “Gyroscopes” in Wiley Encyclopedia of Electrical and Electronics Engineering 2007, John Wiley & Sons Inc.

4. https://www.imo.org (International Maritime Organization)

5. https://www.waterencyclopedia.com

6. https://www.hellaspath.gr

7. https://www.disigma.gr

8. https://boatingmag.com

9. https://www.gpsworld.com

10. https://www.marineelectronicsjournal.com

11. Mastering the I2C bus: LabWorx 1, Vincent Himpe 2011, Elektor Electronics Publishing

12. https://www.mathworks.com

13. https://www.circuitbasics.com

14. https://www.allaboutcircuits.com

15. https://www.build-electronic-circuits.com