ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ

«Αυτόματη μηχανή ψαρέματος»

Automatic fishing mashine

Του φοιτητή

Σπυρίδωνα Γρηγοριάδη

Αρ.Μητρώο: 514327

Επιβλέπων

Άγγελος Γιακουμής

Επίκουρος Καθηγητής

Σεπτέμβριος 2023

«Automatic fishing mashine».

Κωδικός Π.Ε. 22252

Φοιτητής: Σπυρίδωνας Γρηγοριάδης

Εισηγητής: Άγγελος Γιακουμής

Ημερομηνία ανάληψης Π.Ε. 14-10-2022

Ημερομηνία περάτωσης Π.Ε. 23/01/24

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

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

Πρόλογος

Για την εκπόνηση της πτυχιακής μου εργασίας ανέλαβα την κατασκευή μιας προγραμματιζόμενης αυτόματης μηχανής ψαρέματος, για την χρήση αλιείας απο σκάφος. Η ιδέα για την αλιεία, επαγγελματική η ερασιτεχνική χόμπι και γενικώς η αγάπη για το ψάρεμα, μου οδήγησε στην αναζήτηση πληροφοριών εως και την ολοκλήρωση της δικής μου αυτόματης μηχανής αλιείας.Μέσα απο την ιστορική διαδρομή δίνονται τρόποι αλιεύματος απο την αρχαιότητα εως και σήμερα. Στο εμπόριο μετά απο έρευνα υπάρχουν κάποιες εταιρίες οι οποίες κατασκευάζουν τέτοιου είδους προγραμματιζόμενες αυτόματες μηχανές για επαγγελματική και ερασιτεχνική χρήση. Στην αλιεία ως επαγγελματική και ερασιτεχνική χρήση βασίστηκε και η δική μου εργασία.

Περίλιψη

Περιγραφή: Η εργασία αφορά την κατασκευή αυτόματης μηχανής ψαρέματος «Automatic fishing mashine».

Βασιζόμενο στην αναπτιξιακή πλακέτα STM32 NUCLEO-144 με STM32H743ZIT6U MCU. Στην κατασκευή χρησιμοποιείται ένα DC μοτέρ το οποίο είναι οδηγούμενο απο μία γέφυρα για την ρύθμιση στροφών και κατεύθυνσης, καθώς υπάρχει μια πλακέτα απομόνωσης μεταξύ του επεξεργαστή και της οδήγησεις μοτέρ. Η μηχανή αλιείας διαθέτη επίσης ένα ηλεκτρομαγνητικό φρένο το οποίο οδηγήται απο ενα mosfet δια μέσου ενος μετατροπέα step up, μια lcd οθόνη η οποία εμφανίζει σε πραγματικό χρόνο τον αισθητήρα, καθώς επίσης και ένα πληκτρολόγιο για τις ανάλογες ρυθμίσεις. Ολα τα υλικά (πλακέτες) ειναι εσωτερικά τοποθετημένα σε πλαστικό στεγανό κιβώτιο ενω μεταλική πλάκα 3mm συγκρατεί εξωτερικά τον ηλεκτρομαγνήτη και εσωτερικά το μοτέρ, καθώς η πλαστική ρόδα τοποθετείτε εξωτερικά επάνω στον άξονα του μοτέρ.

Λειτουργία: Ο χρήστης πρέπει να φορτώσει το νήμα, να ρυθμίσει τον αισθητήρα ανίχνευσης, το φρένο ,και τα μέτρα στα οποία θέλει να επιλέξει για να αλιεύση.

Εξοπλισμένο με ένα κύριο διπλό μεγάλο μπουτόν, για standby και engine start και έναν γενικό διακόπτη. Καθώς γίνει εφικτός ο ορισμός των ρυθμίσεων, ο χρήστης διευθετεί τον πλάνο στη θάλασσα και ενεργοποιεί την μηχανή με το εμφανή μπουτόν . Ο αισθητήρας θα ενεργοποιηθεί οταν ο πλάνος φθάσει στο επίπεδο ψαρέματος τον οποίο έχει επιλέξει ο χρήστης. Εαν ο αισθητήρας ανιχνεύσει σήμα μεγαλύτερο απο το σήμα που του έχει οριστεί, τότε ενεργοποιείται το φρένο, και στην συνέχεια το μοτέρ για την επιστροφή του νήματος.

«Automatic fishing mashine»

«Spyridon Grigoriadis»

Abstract

Description: The work concerns the construction of an automatic fishing machine.

Based on STM32 NUCLEO-144 development board with STM32H743ZIT6U MCU. In the construction, a DC motor is used which is driven by a bridge to adjust the speed and direction, as there is an isolation board between the processor and the motor drive. The fishing machine also has an electromagnetic brake which is driven by a mosfet through a step up converter, an LCD screen which displays the sensor in real time, as well as a keyboard for the corresponding settings. All the materials (plates) are placed internally in a plastic sealed box, while a 3mm metal plate holds the electromagnet externally and the motor internally, as the plastic wheel is placed externally on the motor shaft.

Operation: The user needs to load the line, set the detection sensor, the brake, and the gauges he wants to choose for fishing.

Equipped with a main double large button for standby and engine start and a general switch. As it becomes possible to define the settings, the user arranges the plan in the sea and activates the machine with the visible button. The sensor will activate when the shot reaches the fishing level selected by the user. If the sensor detects a signal greater than the signal set for it, then the brake is activated, and then the motor to return the thread.

Ευχαριστίες

Θέλω να ευχαριστήσω την οικογένεια μου για την συμπαράσταση τους, καθώς και τον επιβλέπων επίκουρο καθηγητή κ.Γιακουμή Άγγελο για την επιστημονική και παιδαγωγική του καθοδήγηση.

Conents

Κατάλογος Σχημάτων-Εικόνων

Κεφάλαιο 1ο:

1.1 Εισαγωγή – Σκοπός της εργασίας

Ο κύριος στόχος αυτής της εργασίας είναι η κατασκευή ενός αυτόματου συστήματος αλιείας με αισθητήρες που υλοποιείται με STM32H743ZIT6U MCU έχοντας επικοινωνία πραγματικού χρόνου.

Συγκεκριμένα, οι επιμέρους στόχοι της εργασίας είναι:

Ο χρήστης να μην χρειάζεται να έχει την συνεχή επαφή της αίσθησης του νήματος προς το αλίευμα.

Ο χρήστης να παρακολουθεί την αλιεία προγραμματίζοντας την μηχανή , καθώς η χρήση ενός τέτοιου συστήματος καθιστάτε επαγγελματική.

Ποιο συγκεκριμένα η κατασκευή ενός τέτοιου μηχανήματος έχει σαν στόχο την ημιεπαγγελματική , και επαγγελματική χρήση.

Έτσι, στόχος του κάθε ψαρά με βάρκα είναι να μπορεί να αλιεύσει σε μεγαλύτερο βάθος, καθώς εκεί βρίσκονται και τα μεγαλύτερα ψάρια.

Η χρήση χειροκίνητου μηχανισμού όμως πρακτικά θεωρείται πολύ κουραστική, όχι μόνο για τους απλούς ερασιτέχνες αλλά και για τους ημιεπαγγελματίες και επαγγελματίες ψαράδες, διότι απαιτεί αρκετή δύναμη μεν, για να ελέγξεις την δύναμη ενός μεγάλου ψαριού, ιδίως από μεγάλο βάθος, αφετέρου, ο χρόνος ο οποίος θα χρειαστεί για να ανεβάσει ο ψαράς το ψάρι από τον βυθό, θα είναι αρκετά μεγάλος, έτσι λοιπόν όπως έχω αναφέρει και ποιο πάνω η εργασία μου στοχεύει στην επαγγελματική χρήση αυτού του μηχανήματος .

1.2 Βιβλιογραφική Ανασκόπηση

Προτού αναλυθεί ο μηχανισμός ψαρέματος , θα αναφερθούμε στην ιστορική αναδρομή της αλιείας, και σε παρόμοια συστήματα ερασιτεχνικά και επαγγελματικά που προτείνονται στην βιβλιογραφία και έχουν υλοποιηθεί.

1.2.1 Η αλιεία στα παλιά χρόνια.

Η αλιεία, όπως και η γεωργία είναι μια μορφή πρωτογενούς παραγωγής.

Ιστορικά, τα θαλάσσια μαλάκια ήταν μεταξύ των πρώτων ειδών διατροφής του ανθρώπου και ιδιαίτερα οι αχηβάδες, τα μύδια και τα στρείδια. Αυτό αποδείχθηκε από τον εντοπισμό σε πολλές περιοχές σε ολόκληρο τον κόσμο.

Ο προϊστορικός άνθρωπος, όταν ψάρευε, δεν ήξερε ακόμη να χρησιμοποιεί, ούτε το δίχτυ ούτε και το αγκίστρι, άλλωστε και τα δυο αυτά δεν είχαν ακόμη εφευρεθεί. Τα ψάρια τα έπιανε, είτε με το χέρι είτε χρησιμοποιούσε ένα μυτερό ξύλο για να τα καρφώνει. Ο τρόπος αυτός χρησιμοποιείται ακόμη από πολλούς πρωτόγονους λαούς και σήμερα, ενώ τον χρησιμοποιούν και οι δικοί μας ψαράδες για τα χταπόδια κλπ. (το καμάκι).

Αργότερα, άρχισε να χρησιμοποιεί και το βέλος, για να χτυπάει τα ψάρια που βρίσκονται στα ανάβαθα νερά. Τον τρόπο αυτό του χρησιμοποιούν ακόμη πρωτόγονες φυλές.

Στη νεολιθική περίοδο κάνουν την εμφάνισή τους τα πρώτα αγκίστρια από κόκαλο ή ξύλο με τη μορφή περίπου που τα ξέρουμε και σήμερα. Στην εποχή του σίδερου χρησιμοποιούνται πλέον αγκίστρια από σίδερο, που δε διέφεραν σχεδόν καθόλου από τα σημερινά. Το ψάρεμα με το καλάμι πρέπει να εμφανίστηκε πολύ αργότερα και σε περιοχές που είχαν βράχους στις ακτές.[1]

1.2.2 Είδη Αλιείας

Η Αλιεία βασικά διακρίνεται σε θαλάσσια και εσωτερικών υδάτων (λιμνών, ποταμών).

Η Θαλάσσια αλιεία τυγχάνει πολλών διακρίσεων όπως:

-Ατομική αλιεία (μικρής έκτασης χειρωνακτική) και Μεγάλη αλιεία ή βιομηχανική αλιεία, (με χρήση μηχανοκινήτων σκαφών για αλιεία μεγάλων ποσοτήτων ψαριών).
-Παράκτια αλιεία (inshore fishery)και Αλιεία βαθέων υδάτων ή Αλιεία ανοικτής θάλασσας ή Ωκεάνια αλιεία ή Υπερπόντια αλιεία (offshore fishery).
-Αλιεία επιφανείας (αφρόψαρων όπως τόνοι, παλαμίδες, ρέγγες, κολιοί, σαρδέλες κ.ά.) και Αλιεία βυθού (π.χ. για γλώσσες, μπακαλιάρους, μπαρμπούνια κ.ά.).
-Αλιεία χειρωνακτική και σε Αλιεία μηχανοκίνητη. [1]

Εικόνα 1.1 Ψαρεύοντας[1]

1.2.3 Παλιοί και νεότεροι μέθοδοι αλιείας

α) *Η αλιεία με ηλεκτρισμό.* Στήνονται, σ' ορισμένα σημεία, δίχτυα ή φράχτες. Από την αντίθετη μεριά, με αδιάκοπο ή εναλλασσόμενο ρεύμα, τα ψάρια αναγκάζονται να πάνε προς τα δίχτυα ή τα φράγματα και εκει πιάνονται από τους ψαράδες.

β) *Ψάρεμα με τράτα***.** Μικρό αλιευτικό πλοίο, συνήθως πλατύ και μεγάλης αντοχής, εφοδιασμένο με μεγάλο σάκο. Το ψάρεμα με τράτα γίνεται με μια μεγάλη συρόμενη σαγήνη (γρίπο) μακριά από την ακτή. Σαρώνει το βυθό και μαζεύει στο σάκο ότι βρίσκει.

γ) *Με δίχτυα.* Απλούστερος και συνηθέστερος τρόπος είναι η αλιεία με τα *δίχτυα*. Ανάλογα με το ψάρι που επιδιώκει να πιάσει ο ψαράς, είναι κατασκευασμένο και το δίχτυ. Αν πρόκειται, για μικρά ψάρια (σαρδέλες, μπαρμπούνια, γαρίδες κ.ά.), το δίχτυ

είναι κατασκευασμένο από ψιλό νήμα κι έχει μικρές τρύπες. Αν πρόκειται για μεγάλα ψάρια, το δίχτυ είναι χοντρό και γερό και τ' ανοίγματα μεγάλα. Πάνω στο δίχτυ είναι δεμένοι φελλοί και βαρίδια. Τα δίχτυα διακρίνονται σε δίχτυα βυθού και σε δίχτυα επιφάνειας.

δ) *Ψάρεμα με γρι_γρι.* Όνομα αλιευτικού συγκροτήματος, το οποίο αποτελείται από ένα καΐκι που σέρνει από πίσω του 5-6 βάρκες· αυτές φέρνουν στο πίσω μέρος από μια μεγάλη λάμπα η οποία ανάβει με υγραέριο ή ασετυλίνη και δίνει πολύ δυνατό φως με το οποίο προσελκύονται τα ψάρια. Το ψάρεμα γίνεται με δίχτυα και κυρίως τη νύχτα.

ε)*Η αλιεία με αγκίστρι και πετονιά.* Είναι η ατομική αλιεία, συνήθως ερασιτεχνών ψαράδων, που δε χρησιμοποιούν βάρκες ή άλλα μέσα για πιο συστηματικό ψάρεμα.

ζ) *Υποβρύχια αλιεία.* Στην ανάπτυξη της υποβρύχιας αλιείας οδήγησε το επιστημονικό ενδιαφέρον για την εξερεύνηση του βυθού των θαλασσών. Το ψαροτούφεκο, που χρησιμοποιείται στην υποβρύχια αλιεία, κατασκευάζεται από ένα είδος αλουμινίου για να είναι ελαφρύ. Μ' αυτό, και χρησιμοποιώντας βέλη, σκοτώνονται τα ψάρια ή τα άλλα θαλάσσια ζώα. Τα εξαρτήματα του δύτη είναι: στολή από ελαστικό που δεν τη διαπερνάει το νερό, προσωπίδα, ζώνη, μαχαίρι, βατραχοπέδιλα, μετρητής βάθους, μετρητής πίεσης, ρολόι και αναπνευστική συσκευή με πιεσμένο αέρα. Τέτοιοι αλιείς είναι οι σφουγγαράδες και οι αλιείς μαργαριταριών[1].

η) *Η αλιεία απο βάρκα*. Εξοπλισμένοι οι αλιείς με διαφόρων τύπων μηχανισμών, καλάμια, καθώς και ηλεκτρικά συστήματα για ημιεπαγγελματική η επαγγελματική χρήση.

Εικόνα 1.2 Ηλεκτρονικός μηχανισμός [2].

Εικόνα 1.3 Ηλεκτρικός μηχανισμός καθετής[3].

Εικόνα 1.4 Ηλεκτρονικός μηχανισμός καθετής[4].

Εικόνα 1.5 Σπαστό καλάμι ψαρέματος[5].

1.3 Δομή της εργασίας

Στο κεφάλαιο πρώτο παρουσιάζεται μια εισαγωγή της εργασίας καθώς και οι στόχοι της, επίσης αναφέρεται βιβλιογραφική ανασκόπιση στην αλιεία και σε παρόμοια προγραμματιζόμενα ηλεκτρονικά συστήματα ψαρέματος.

Στο κεφάλαιο δεύτερο περιγράφεται η θεωρητική προσέγγιση των εξαρτημάτων καθώς και η ηλεκτρονική διάταξη του συστήματος.

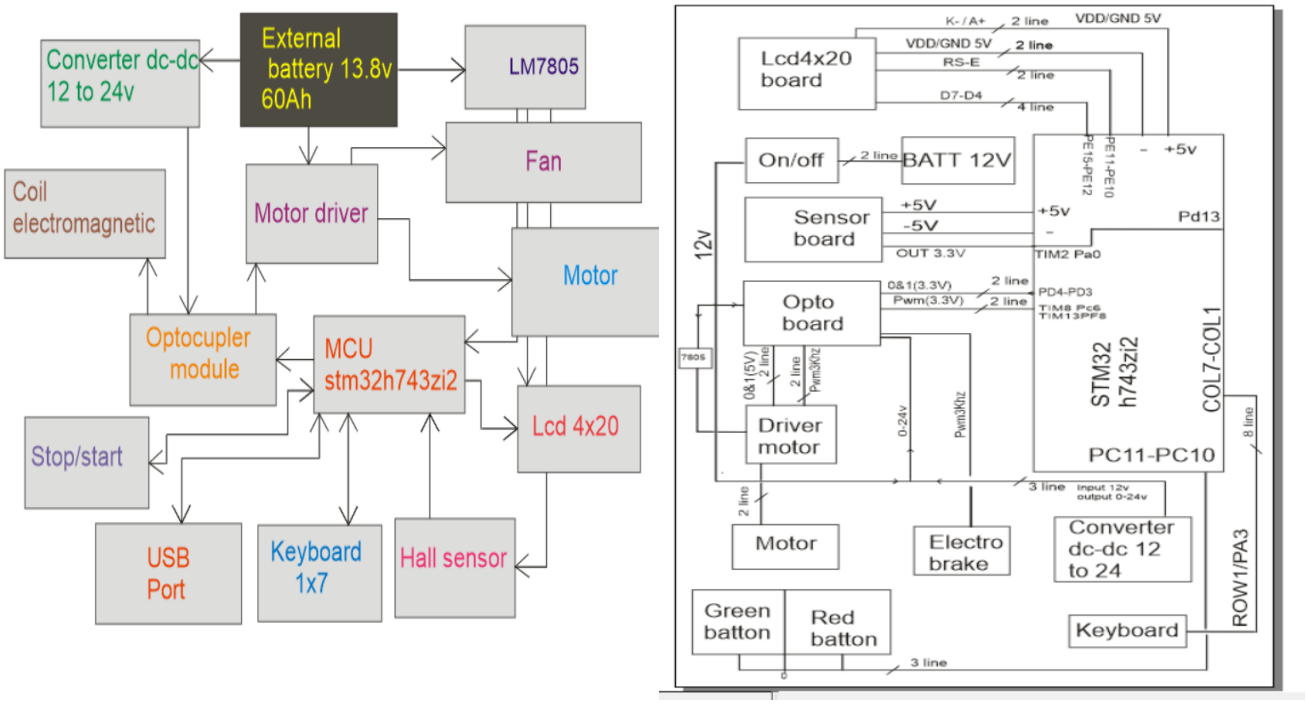
Κεφάλαιο 2ο: Θεωρητική προσέγγιση εξαρτημάτων του συστήματος

2.1 Εισαγωγή

Το σύστημα αυτόματου μηχανησμού που υλοποιήθηκε, αποτελείται από τα εξείς υλικά:

1) Ηλεκτρικός κινητήρας 12v/ DC 250W.
2) Μειωτήρας στροφών.
3) Πληκτρολόγιο 1x7.
4) Πλακέτα μετατροπέα DC-DC 12v σε 24v.
5) Πλακέτα αισθητήρων.
6) Πλακέτα οθόνης 4x20.
7) Σταθεροποιητής τάσης.
8) Πλακέτα μονάδας optocoupler.
9) Βοηθητική εξωτερική είσοδος USB.
10) Ένα διπλό μπουτόν.
11) Ενας κύριος διακόπτης.
12) Πλαστικό στεγανό κιβώτιο.
13) Πλαστική καρούλα.
14) Μεταλλική πλάκα 5mm.
15) Ηλεκτρομαγνητικό φρένο.
16) Μπαταρία 12v / 60Ah.
17) Πλακέτα οδήγησεις κινητήρα.
18) Πλακέτα μικροελγκτή.

2.2 Σχηματική διάταξη συστήματος



Εικόνα 2.1 Σχηματική διάταξη του ηλεκτρονικού συστήματος & διασύνδεση των επιμέρους μονάδων

2.3 Ηλεκτρικός κινητήρας DC

Εικόνα 2.2 Ηλεκτρικός DC κινητήρας 12v/250w/3000RPM. [6]

Οι κινητήρες αυτής της κατηγορίας τροφοδοτούνται από κάποια πηγή συνεχούς τάσης. Από κατασκευαστικής απόψεως, δεν παρουσιάζουν καμία διαφορά σε σχέση με τις γεννήτριες ΣΡ. Βασικό πλεονέκτημα τους αποτελεί η ευκολία ελέγχου της ροπής και της ταχύτητας τους σε ένα μεγάλο εύρος τιμών.

Ένα μέτρο σύγκρισης μεταξύ διαφορετικών κινητήρων ΣΡ είναι η **διακύμανση ταχύτητας**, η οποία ορίζεται από τη σχέση

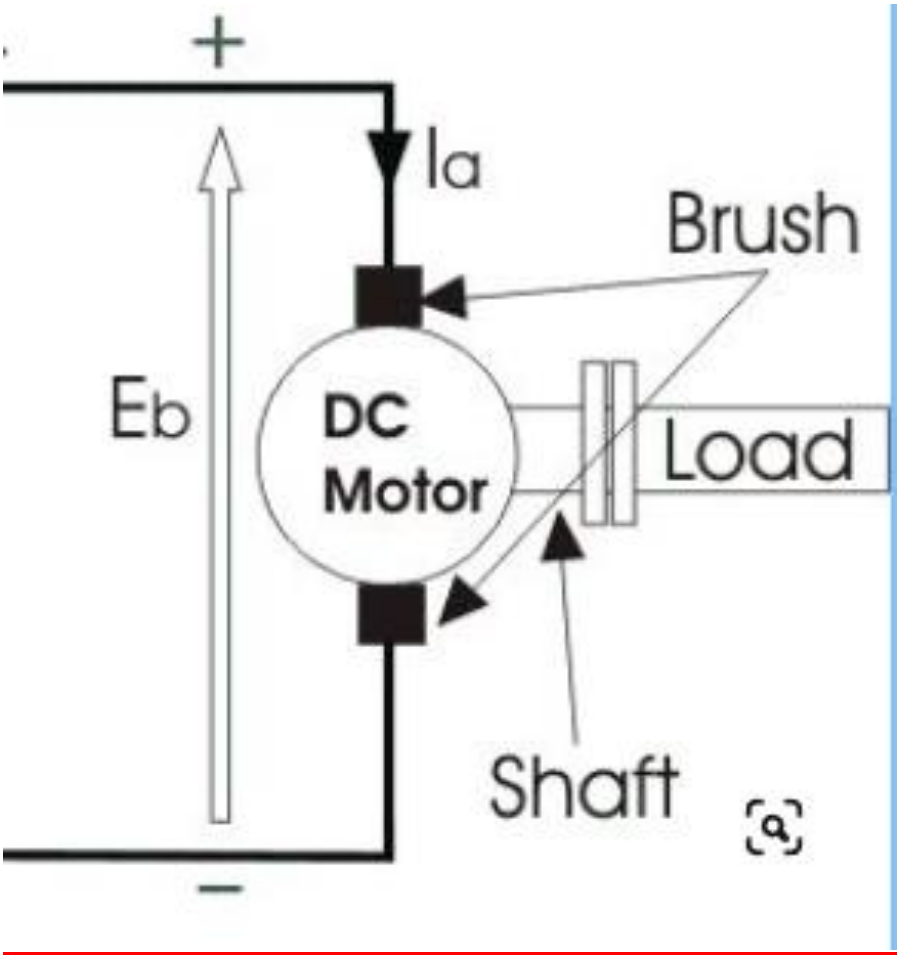
$$SR = \frac{\omega nl - \omega fl}{\omega fl} 100\%$$

με τους δείκτες *nl* και *fl* να υποδεικνύουν, αντίστοιχα λειτουργία υπό μηδενικό και υπο πλήρες φορτίο. Θετική τιμή της *SR* υποδεικνύει μείωση της ταχύτητας του κινητήρα κατά την αύξηση του φορτίου, ενώ αρνητική τιμή της *SR* υποδεικνύει αύξηση της ταχύτητας κατά την αύξηση του φορτίου.[7]

Το μέγεθος αυτό εκφράζει προφανώς την ικανότητα ενός κινητήρα να διατηρεί την ταχύτητά του σταθερή, όταν μεταβάλεται το εφαρμοζόμενο φορτίο (μικρότερη απόλυτη τιμή της *SR* υποδουλώνει μεγαλύτερη σταθερότητα).

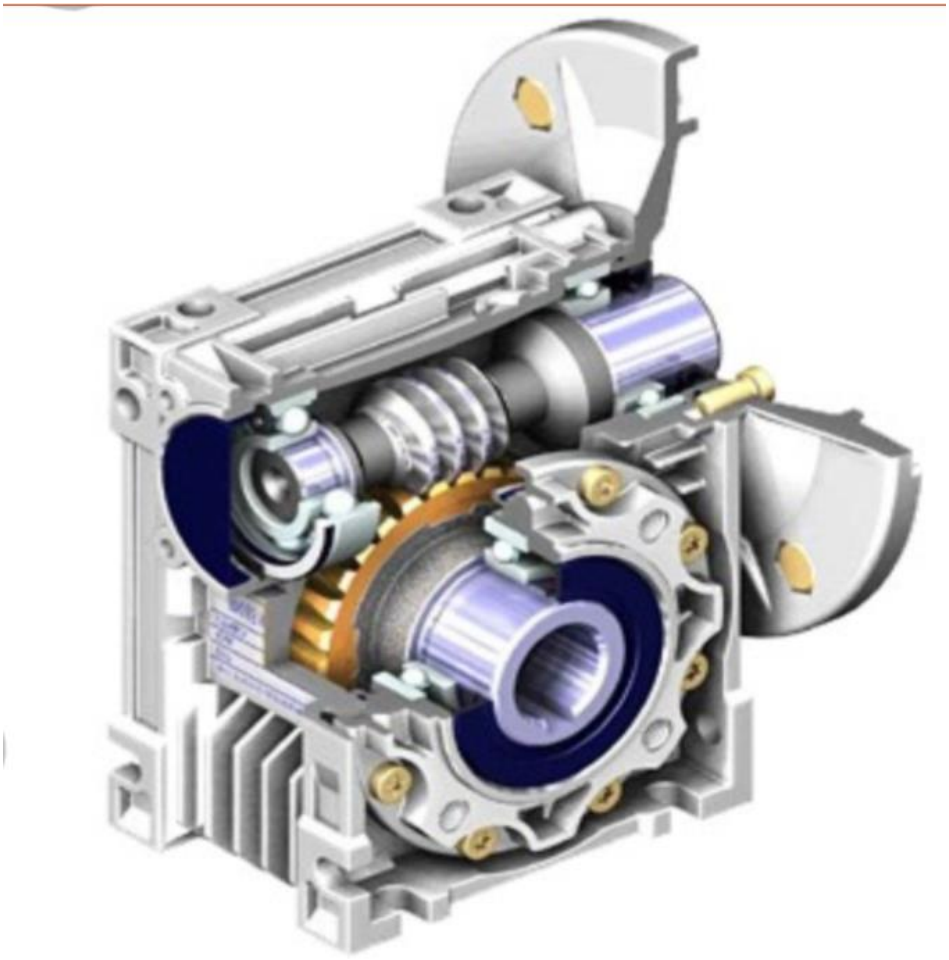
Όπως είναι γνωστό, ένας ηλεκτρικός κινητήρας παρέχει μηχανική ισχύ σε μια μηχανή η οποία αποτελεί το “φορτίο” του.

Όταν ο κινητήρας περιστρέφεται χωρίς να υπάρχει κάποιο φορτίο συνδεδεμένο, τότε η λειτουργεία του χαρακτηρίζεται ως εν “κενό”[7]



Εικόνα 2.3 Ηλεκτρικό σχήμα DC κινητήρα [8].

2.3.1 Ηλεκτρομειωτήρας στροφών



Εικόνα 2.4 Ηλεκτρομειωτήρας στροφών NMRV030 1:20 [9].

Η Μηχανολογία αποτελείται από διάφορους τομείς μελέτης, με πιο αξιοσημείωτο εκείνον της μετάδοσης κίνησης. Η ιδιαίτερη σημασία της μπορεί να εντοπιστεί σε πολλά **συστήματα μετάδοσης κίνησης**, από την πιο απλή μορφή, όπως το σύστημα αλυσοκίνησης στα ποδήλατα, μέχρι την πιο περίπλοκη, όπως αυτή στα κιβώτια ταχυτήτων για τα αυτοκίνητα.

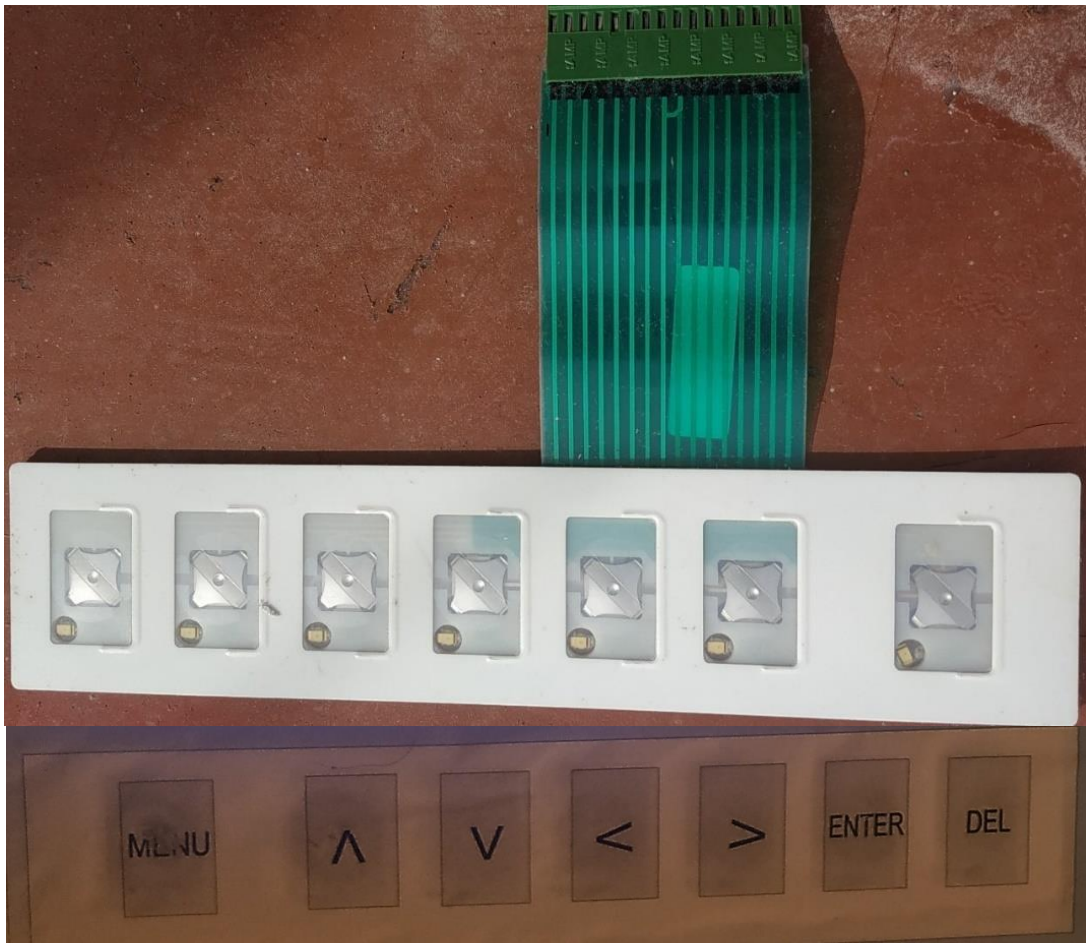
Με τη χρήση τους, μπορούν να συνδέσουν την πηγή κινητικής ενέργειας με το σύστημα παραγωγής ωφέλιμου έργου, να μεταφέρουν τη στρεπτική ροπή ενός κινητήρα με τρόπο ώστε να καλύπτονται οι λειτουργικές ανάγκες της μηχανοκατασκευής, ενώ μπορούν να διατηρήσουν ή και να μετατρέψουν την ομαλή περιστροφική κίνηση σε άλλες μορφές κίνησης, όπως πχ. σε ευθύγραμμη, ευθύγραμμη παλινδρομική και περιστροφική. Για την επίτευξη όμως μιας σταθερής και ελεγχόμενης κίνησης σε μια κατασκευή, γίνεται χρήση του **ηλεκτρομειωτήρα**.[10]

2.3.2 Τι είναι και πως λειτουργεί ο ηλεκτρομειωτήρας;

Ο **ηλεκτρομειωτήρας** αποτελεί ένα μηχάνημα, που συνδυάζει τον ηλεκτροκινητήρα με τον μειωτήρα στροφών. Χάρη στα χαρακτηριστικά που διαθέτει, μπορεί να μειώσει τον αριθμό των εκάστοτε στροφών που παρέχει ο ηλεκτροκινητήρας ώστε να ελέγχει την ταχύτητα και τη μετάδοση της κίνησης. Χρησιμοποιείται στο βιομηχανικό αυτοματισμό, ιδιαίτερα σε εγκαταστάσεις και μηχανήματα, όπου είναι απαραίτητη η διαρκής μεταφορά της απαραίτητης ροπής ώστε να επιτυγχάνεται η βέλτιστη λειτουργία τους.

Ο **μειωτήρας στροφών** είναι ένας μηχανισμός μετάδοσης κίνησης μέσω ζευγών οδοντωτών τροχών (βαθμίδες μείωσης). Αυτές οι βαθμίδες μπορούν να αποτελούνται από συνδυασμό μετωπικό οδοντωτών τροχών, με ελικοειδείς οδοντωτούς τροχούς και κωνικούς οδοντωτούς τροχούς. Με τη λειτουργία τους εντός ειδικού διαμορφωμένου κιβωτίου μέσα σε λουτρό λιπαντικού, επιτυγχάνεται η ομαλή μεταφορά της ισχύος του ηλεκτροκινητήρα καθώς και η μεταφορά της επιτυγχόμενης ροπής ανάλογα με τη σχέση μείωσης του μειωτήρα [10].

2.4 Πληκτρολόγιο 1x7



Εικόνα 2.5 Πληκτρολόγιο 1x7

Menu: Επιλογή ρυθμίσεων

^ : Επιλογή up

v : Επιλογή down

< : Επιλογή left

> : Επιλογή right

Enter : Επιλογή εισαγωγής

Del : Επιλογή τέλους ψαρέματος

Το **πληκτρολόγιο** είναι μία συσκευή εισόδου ενός ηλεκτρονικού υπολογιστή. Η βασική λειτουργία για την οποία χρησιμοποιείται το πληκτρολόγιο είναι η εισαγωγή χαρακτήρων ή κειμένου στον υπολογιστή ή τον σταθμό εργασίας από ένα χρήστη πατώντας τα πλήκτρα του με τα δάχτυλά του ή κάποιο άλλο μέρος του σώματός του. Τα πιο συνηθισμένα πληκτρολόγια προορίζονται για χρήση με τα δάκτυλα. Πέρα από τα πλήκτρα που αντιστοιχούν στα γράμματα του αλφαβήτου, τους τόνους ή πνεύματα, τα σημεία στίξης και άλλους χαρακτήρες, περιλαμβάνει αρκετά πλήκτρα που διευκολύνουν τη χρήση του λειτουργικού συστήματος και των διαφόρων προγραμμάτων του υπολογιστή καθώς και την «πλοήγηση» ανάμεσά τους ή ανάμεσα στις διαφορετικές τους λειτουργίες/χρήσεις [11].

2.5 Πλακέτα μετατροπέα DC-DC 12v σε 24v.

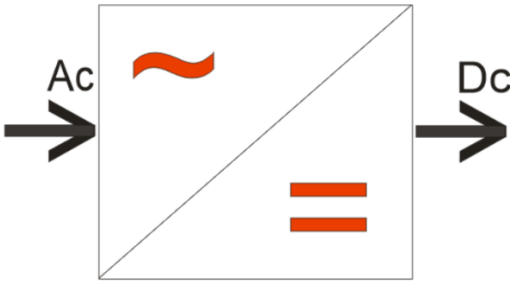
Εικόνα 2.6 Μετατροπέας 150w dc-dc 12 σε 24v [12]

2.5.1 Ειδη μετατροπέων ηλεκτρικής ενέργειας

Οι μετατροπείς ηλεκτρικής ενέργειας (ισχύος) διακρίνονται σε τέσσερις κύριες κατηγορίες, ανάλογα με τη μορφή της ισχύος εισόδου και εξόδου.

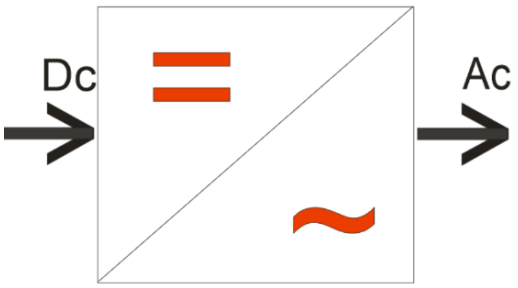
Αυτές είναι:

α) Μετατροπείς *ac-dc* η ανορθωτές (*Rectifiers*). Ανορθωτές ονομάζονται οι διατάξεις ισχύος, οι οποίες μετατρέπουν το εναλλασόμενο ρεύμα σε συνεχές. Ανάλογα με τη μορφή της εναλλασόμενης εισόδου, οι ανορθωτές διακρίνονται σε μονοφασικούς και πολυφασικούς(διφασικούς, τριφασικούς, εξαφασικούς). Ακόμη, διακρίνονται σε ελεγχόμενους και σε μη ελεγχόμενους, ανάλογα με το αν η τάση εξόδου είναι μεταβαλόμενη είτε σταθερή.



Εικόνα 2.7 Γενικό σύμβολο του ανορθωτή AC to DC

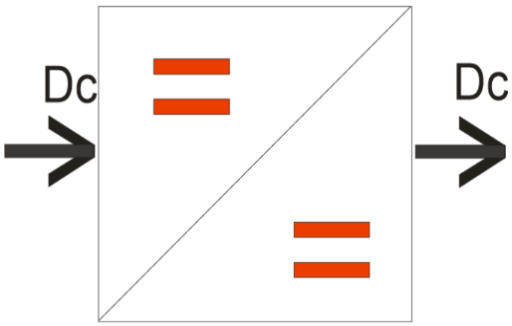
β) Μετατροπείς *ac- Dc* η Αντιστροφείς *(Inverters).* Οι αντιστροφείς μετατρέπουν την ηλεκτρική ενέργεια συνεχούς μορφής σε εναλλασόμενη.Η λειτουργία τους είναι δηλαδή αντίθετη από εκείνη των ανορθωτών. Η έξοδος των αντιστροφέων είναι μονοφασική είτε πολυφασική(συνήθως τριφασική). Επίσης, η συχνότητα και το πλάτος της τάσης η του ρεύματος είναι ελεγχόμενα.



Εικόνα 2.8 Γενικό σύμβολο του αντιστροφέα DC to AC

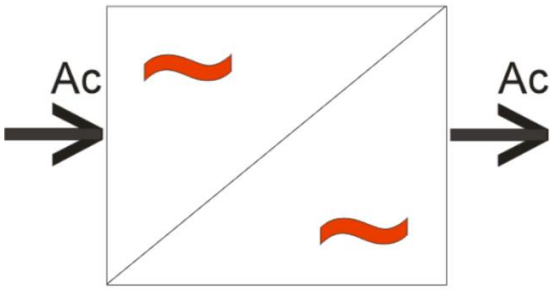
γ) Μετατροπέας συνεχούς ρεύματος (***dc-dc converters, choppers***). Οι μετατροπείς συνεχούς ρεύματος μετατρέπουν την συνεχεί τάση με ορισμένο πλάτος

και πολικότητα.Διακρίνονται σε μετατροπείς υποβιβασμού (*step down)* και ανύψωσεις (*step-up*) της τάσης, ανάλογα με το αν ητάση εξόδου είναι μικρότερη η μεγαλύτερη της τάσης εισόδου. Ακόμη, διακρίνονται σε μετατροπείς με απομόνωση και χωρίς απομόνωση της εξόδου από την είσοδο τους.



Εικόνα 2.9 Γενικό σύμβολο μετατροπέα συνεχούς ρεύματος DC to DC

δ) Μετατροπείς εναλλασομένου ρεύματος η κυκλωμετατροπείς (*Cycloconverters*). Οι κυκλωμετατροπείς μετατρέπουν απευθείας, την εναλλασόμενη τάση με ρυθμιζόμενο πλάτος και συχνότητα. Ο κυκλομετατροπέας ονομάζεται υποβιβασμού συχνότητας (step-down) όταν η συχνότητα εξόδου είναι μικρότερη της συχνότητας εισόδου. Διαφορετικά χαρακτηρίζεται ως ανυψωτής (step-up). Μια ειδική κατηγορία των μετατροπέων εναλλασομένου ρεύματος είναι οι ρυθμιστές εναλλασόμενης τάσης (ac voltage controllers). Οι ρυθμιστές εναλλασόμενης τάσης παρέχουν στην έξοδό τους μια τάση μεταβαλόμενου πλάτους, η συχνότητα της οποίας είναι σταθερή και ίση με τη συχνότητα της ac πηγής εισόδου.[Βιβλίο θεωρίας Ηλεκτρονικά ισχύος Ιορδάνης Κιοσκερίδης σελ.28-30] [13]



Εικόνα 2.10 Γενικό σύμβολο του κυκλομετατροπέα AC to AC

2.5.2 Λειτουργεία ανύψωσεις της τάσης

Για την κατασκευή χρησιμοποιήθηκε μετατροπέας ανύψωσης της τάσης *(step-up).*

Ο μετατροπέας ανύψωσης της τάσης παρέχει στην έξοδο μια τάση ***Vo*** η οποία είναι μεγαλύτερη της τάσης εισόδου ***Vdc***. Η κυκλωματική δομή του μετατροπέα εικονίζεται στο Σχ.13.10. Όταν ο διακόπτης ***Sw*** είναι σε αγωγή, η πηγή εισόδου παρέχει ενέργει στην επαγωγή η οποία αποθηκεύεται με τη μορφή μαγνητικού πεδίου. Η δίοδος είναι ανάστροφα πολωμένη και δεν άγει. Έτσι η έξοδος είναι απομονομένη από την είσοδο. Όταν ο διακόπης οδηγηθεί στη αποκοπή, το ρεύμα της επαγωγής τείνει να μειωθεί, ενώ η τάση ***uL*** αλλάζει πολικότητα. Μόλις η τάση επαγωγής γίνει ίση με ***Vdc-Vo***, η δίοδος πολώνεται ορθά και η ενέργεια μεταφέρεται από την πηγή και την επαγωγή στο φορτίο.

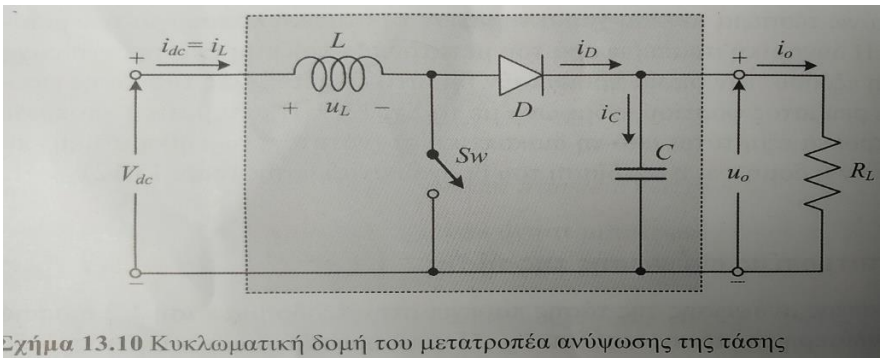
Ο μετατροπέας ανύψωσης της τάσης λειτουργεί με **συνεχή αγωγή** του ρεύματος, όταν το ρεύμα στην επαγωγή ρέει συνεχώς. Όταν η ροή του ρεύματος διακόπτεται σε κάποιο τμήμα της περιόδου, όπου ο διακόπτης είναι σε αποκοπή, ο μετατροπέας λειτουργεί με ασυνεχή αγωγή του ρεύματος.[14]

Ο λόγος της τάσης εξόδου προς την τάση εισόδου, από μηδενική μέση τιμή της τάσης στα άκρα της.

$$\mathrm{VdcDT} + (\mathrm{Vdc} - \mathrm{Vo})(1 - \mathrm{D})\mathrm{Ts} = 0$$

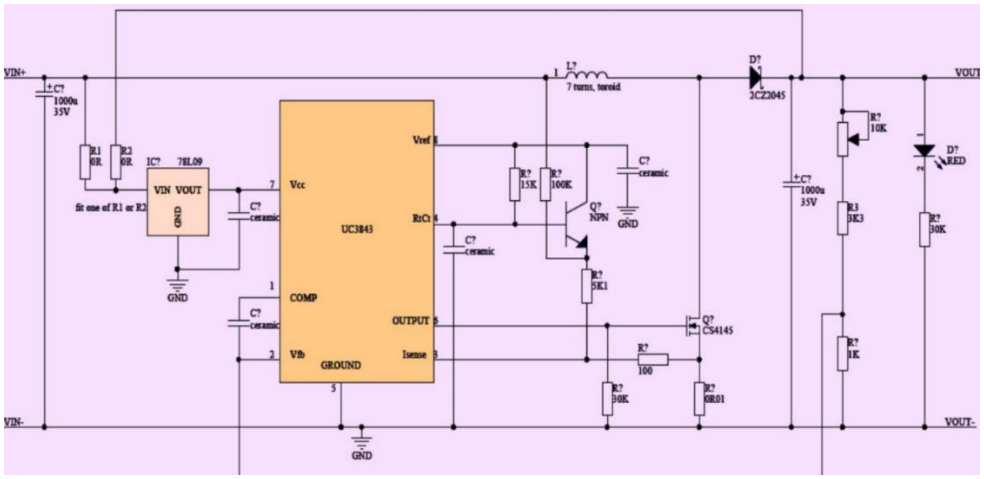
από την οποία

$$\frac{Vo}{Vdc} = \frac{1}{1-D}$$



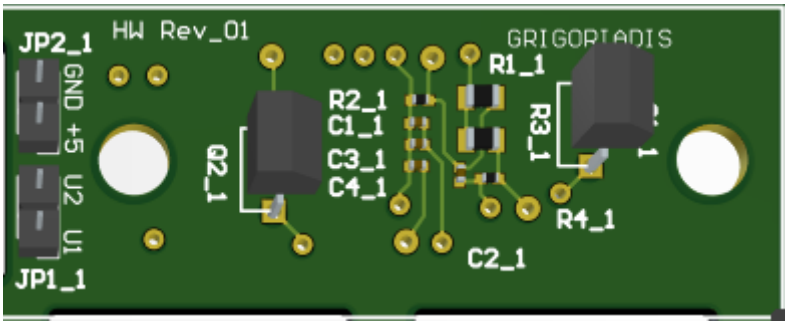
Σχήμα 13.10 Κυκλωματική δομή του μετατροπέα ανύψωσης της τάσης

Εικόνα 2.11 Κυκλωματική δομή του μετατροπέα ανύψωσης της τάσης [Βιβλίο θεωρίας Ηλεκτρονικά ισχύος Ιορδάνης Κιοσκερίδης σελ.456] [14]

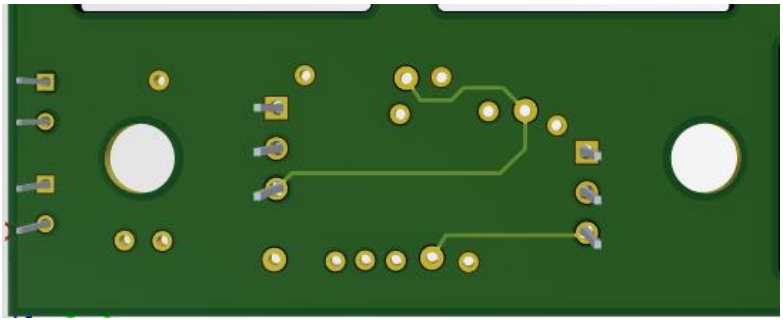


Εικόνα 2.12 Ηλεκτρονικό διάγραμμα 150w dc-dc boost converter [15]

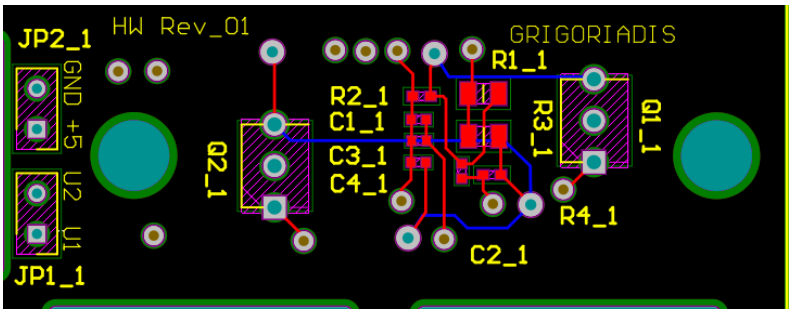
2.6 Πλακέτα αισθητήρων.



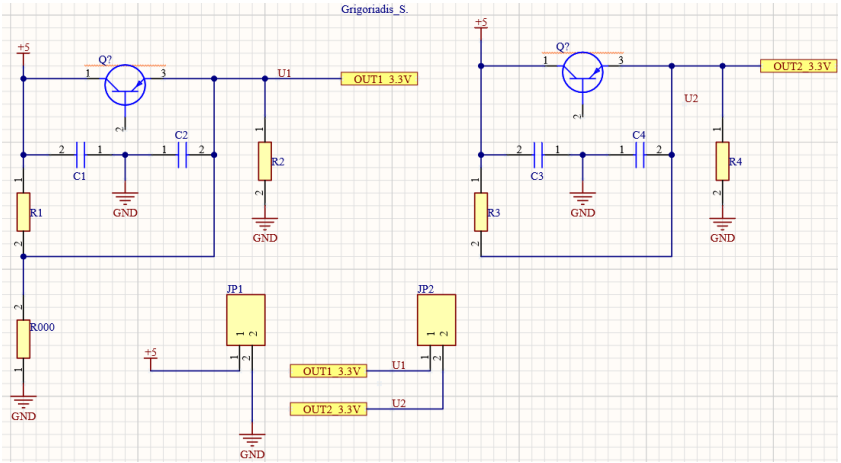
Εικόνα 2.13 Πλακέτα αισθητήρων Hall sensor εμπρόσθια όψη (3D)



Εικόνα 2.14 Πλακέτα αισθητήρων Hall sensor πίσω όψη (3D)



Εικόνα 2.15 Διάγραμμα αγωγών Hall sensor (4 layer)



Εικόνα 2.16 Ηλεκτρονικό διάγραμμα αισθητήρων

2.6.1 Αισθητήρας Hall sensor

Εικόνα 2.17 Αισθητήρας TLE4905L [16]

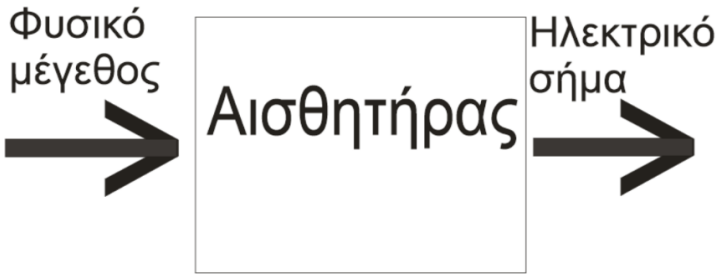
2.6.1.1 Τι είναι οι αισθητήρες

Οι αισθητήρες είναι μια διάταξη που χρησιμοποιείται για την μέτρηση ενός φυσικού μεγέθους.

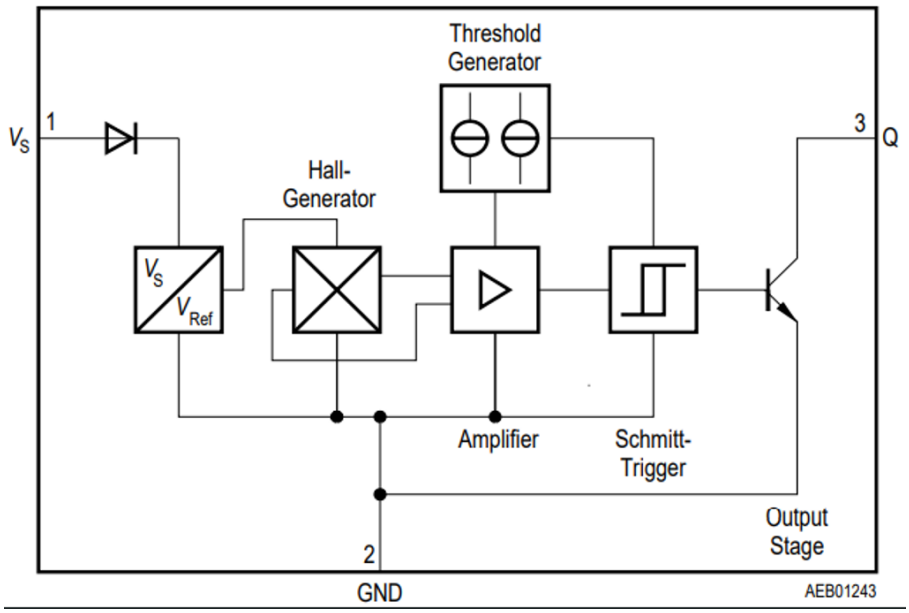
Μετατρέπει το φυσικό μέγεθος σε ηλεκτρικό σήμα.

Φυσικά μεγέθη: Θέση, ταχύτητα,επιτάχυνση,δύναμη, πίεση,θερμοκρασία κλπ. Ειδικοί αισθητήρες μπορούν να ανιχνεύσουν χημικές ουσίες, ήχο,ακτινοβολία κλπ.

Το ηλεκτρικό σήμα εξόδου ενός αισθητήρα είναι είτε τάση είτε ρεύμα.[17]



Εικόνα 2.18 Σχέδιο φυσικού μεγέθους σε ηλεκτρικό



Εικόνα 2.19 Διάγραμμα TLE4905L [18]

2.6.1.2 Περιγραφή κυκλώματος tle4905l

Το κύκλωμα περιλαμβάνει γεννήτρια Hall, ενισχυτή και Schmitt-Trigger σε ένα τσιπ. Το εσωτερικό παρέχει την τάση τροφοδοσίας για τα εξαρτήματα. Ένα μαγνητικό πεδίο κάθετο στην επιφάνεια του τσιπ προκαλεί τάση στον ανιχνευτή του ***Hall*** . Αυτή η τάση ενισχύεται και ενεργοποιεί ένα Schmit ttrigger με έξοδο ανοιχτού συλλέκτη. Η τάση λειτουργίας του ανέρχεται από 3,8 εως 32v [18]

Ο αισθητήρας στην πλακέτα τροφοδοτείται με 5v, ενώ η έξοδος παρέχει 3,3v δια μέσου ενός διαιρέτη τασης για την οδήγηση του στον επεξεργαστή.

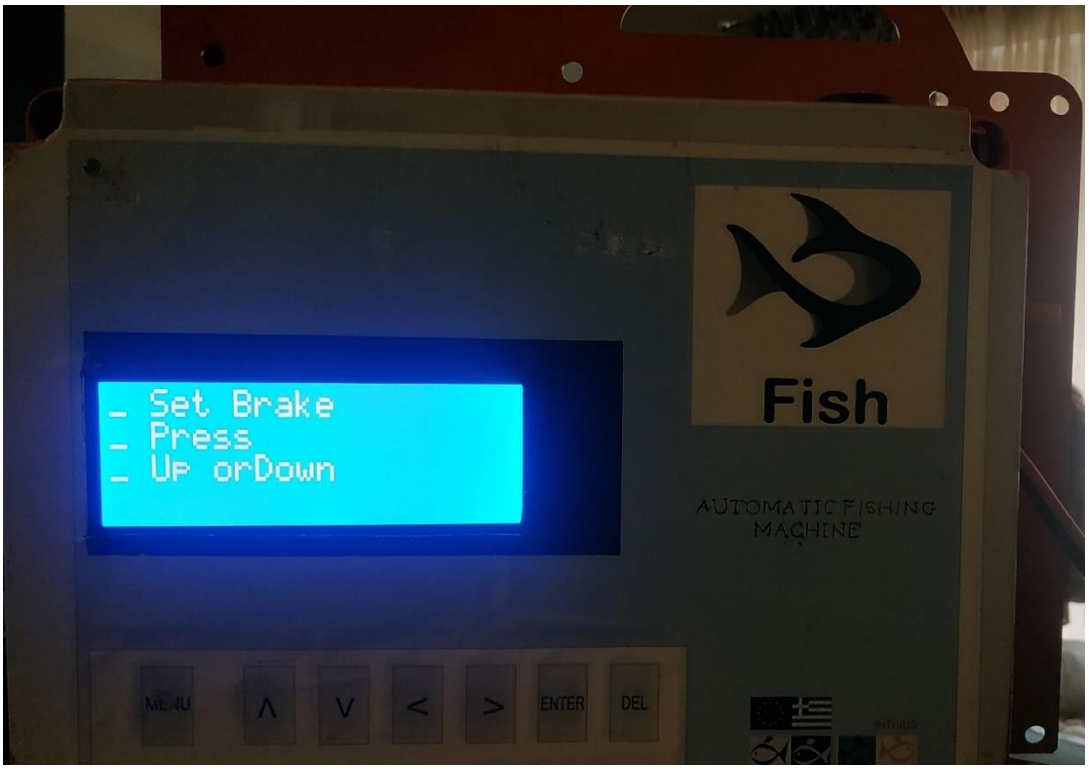
2.7 Πλακέτα οθόνης 4x20.

Εικόνα 2.20 Οθόνη lcd 4x20 NHD-0420E2Z-NSW-BBW [19]

Εικόνα 2.21 Αρχικό πρώτο μύνημα

Εικόνα 2.22 Αρχικό δεύτερο μήνυμα

Εικόνα 2.23 Βασικές ενδείξεις οθόνης, μετά από την ρύθμιση του menu



Εικόνα 2.24 Ένδειξη ρύθμισης φρένου

Εικόνα 2.25 Ένδειξη ρύθμισης αισθητήρα

Εικόνα 2.26 Ένδειξη ρύθμισης load or unload thread

Εικόνα 2.27 Ένδειξη ρύθμισης του βάθους ψαρέματος

Εικόνα 2.28 Ένδειξη παρακαλω περιμένεται, μετά από enter, φόρτωση η εκφόρτωση νήματος

Εικόνα 2.29 Ένδειξη επιστροφής νήματος μετά από (delete)

Εικόνα 2.30 Ένδειξη φόρτωσεις νήματος

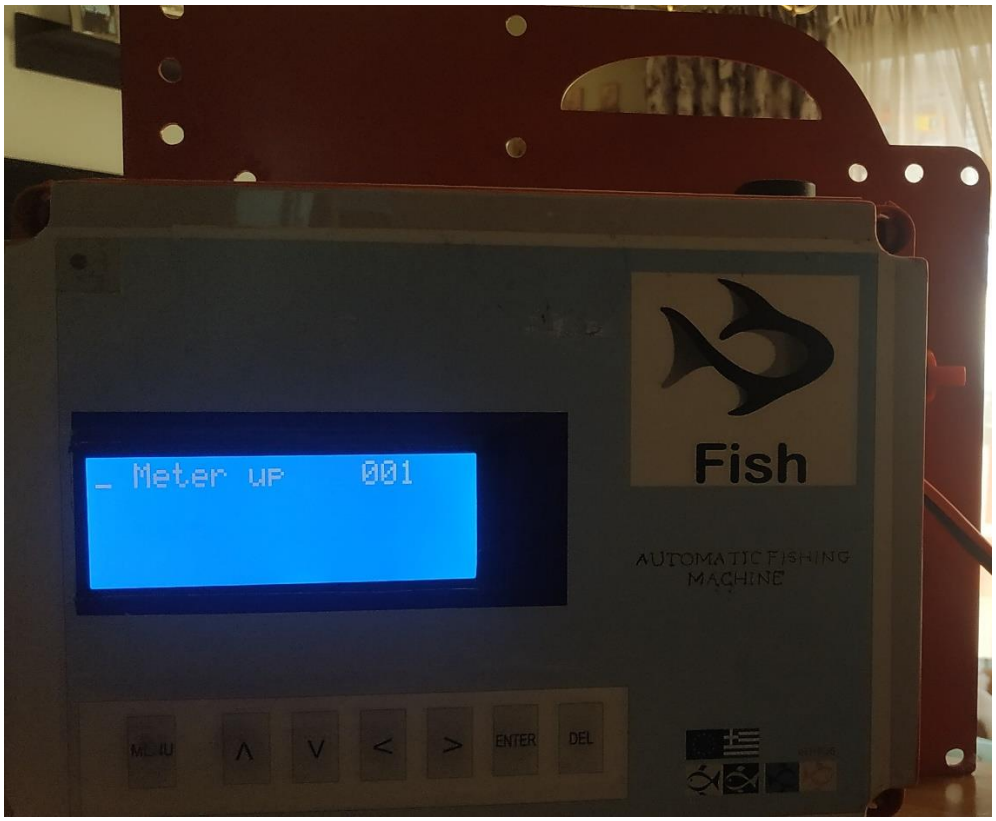
Εικόνα 2.31 Ένδειξη εκφόρτωσεις νήματος

Εικόνα 2.32 Ένδειξη ρύθμισεις της τιμής φρένου

Εικόνα 2.33 Ένδειξη ρύθμισεις της τιμής φρένου

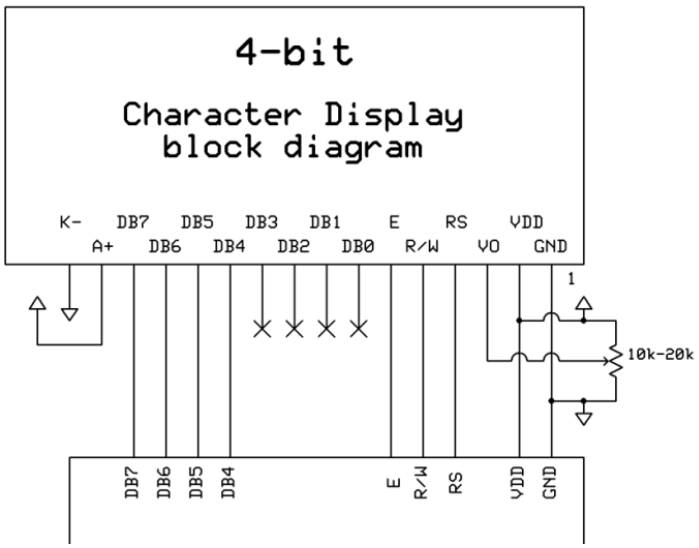
Εικόνα 2.34 Ένδειξη ρύθμισεις της τιμής αισθητήρα

Εικόνα 2.35 Ένδειξη ρύθμισεις της τιμής αισθητήρα



Εικόνα 2.36 Ένδειξη ρύθμισεις της τιμής των μέτρων ψαρέματος

Εικόνα 2.37 Ένδειξη ρύθμισεις της τιμής των μέτρων ψαρέματος



Εικόνα 2.38 Ηλεκτρονικό διαγραμμα σύνδεσης οθόνης 4bit [20]

2.7.1 Λειτουργία οθόνης

Οι οθόνες LCD 2x16 ή 4x20 είναι πολύ εύκολες στη σύνδεση και στη χρήση τους σε μικροελεγκτές. Χρειάζονται ελάχιστες γραμμές κώδικα για την προετοιμασία τους (initialize) και για τις εντολές λειτουργίας τους. Διαθέτουν 8 bit data bus αλλά είναι δυνατό να προγραμματιστούν για λειτουργία με **4 bit,** ώστε μαζί με τα 3 σήματα ελέγχου να μπορούν να συνδεθούν σε ένα port. Όταν είναι προγραμματισμένες για λειτουργία 4 bit στέλνεται πρώτα το ανώτερο μέρος του byte (upper) και στη συνέχεια το κατώτερο (lower). Έχουν έναν καταχωρητή (DD RAM) για αποθήκευση των δεδομένων που εμφανίζονται, ο οποίος είναι εγγραφής – ανάγνωσης. Στις οθόνες με 2 σειρές, τα 40 πρώτα byte αφορούν την πρώτη σειρά με διευθύνσεις στην DD RAM **0x00** έως **0x27** και τα επόμενα 40 τη δεύτερη με διευθύνσεις **0x40** έως **0x67**. Στις οθόνες με 4 σειρές, τα 20 πρώτα αφορούν την πρώτη σειρά με διευθύνσεις στην DD RAM **0x00** έως **0x13**, τα επόμενα 20 (21 – 40) την τρίτη σειρά με διευθύνσεις **0x14** έως **0x27**, τα επόμενα (41 – 60) τη δεύτερη σειρά με διευθύνσεις **0x40** έως **0x53** και τα τελευταία 20 (61 – 80) την τέταρτη με διευθύνσεις **0x54** έως **0x67**. Αν κάθε σειρά έχει 16 θέσεις για εμφάνιση χαρακτήρων, εμφανίζονται τα δεδομένα που είναι αποθηκευμένα στις 16 πρώτες θέσεις του καταχωρητή, ενώ τα υπόλοιπα μπορούν να εμφανιστούν με ολίσθηση χρησιμοποιώντας την αντίστοιχη εντολή της οθόνης. Υπάρχει δυνατότητα να εμφανίζεται ή όχι ο κέρσορας ή να αναβοσβήνει, επίσης υπάρχει δυνατότητα να ολισθαίνει αυτόματα ο κέρσορας ή η οθόνη καθώς στέλνονται χαρακτήρες. Οι οθόνες διαθέτουν, ακόμα, οπίσθιο φωτισμό (back light) στα pin BL+, BL- και ρύθμιση αντίθεσης στο pin Vo.

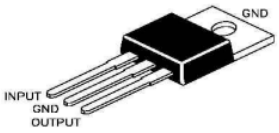
Τα δεδομένα στέλνονται στην οθόνη σε κωδικοποίηση χαρακτήρων ASCII και εμφανίζονται όλοι οι λατινικοί χαρακτήρες και τα σύμβολα που υπάρχουν στις 128 πρώτες θέσεις. Για τις επόμενες 128 θέσεις του ASCII η ROM της οθόνης περιέχει πληροφορίες για εμφάνιση ειδικών χαρακτήρων και ανάλογα με την έκδοση της ROM μπορούν να εμφανιστούν Ελληνικοί, Κυριλλικοί, Κινέζικοι ή Γιαπωνέζικοι χαρακτήρες και κάποια μαθηματικά σύμβολα. Επίσης υπάρχει η δυνατότητα να δημιουργηθούν pixel – pixel 8 χαρακτήρες, οι οποίοι αποθηκεύονται σε έναν καταχωρητή (CG RAM) και μπορούν να εμφανιστούν, όμως διαγράφονται με απώλεια τροφοδοσίας.

Η οθόνη δέχεται εντολές με **0** στο pin RS. Οι εντολές είναι για τη θέση του κέρσορα, τις ιδιότητες του κέρσορα, εντολές ολίσθησης, καθαρισμού της οθόνης, δημιουργίας χαρακτήρων και ανάγνωσης και έχει διαφορετικό χρόνο εκτέλεσης η κάθε μια. Η ανάγνωση χαρακτήρων γίνεται με **1** στο pin R/W. Οι πληροφορίες λαμβάνονται από την οθόνη με **1** στο pin E είτε αυτές είναι δεδομένα, είτε εντολές[20].

2.8 Σταθεροποιητής τάσης LM7805

3-TERMINAL POSITIVE VOLTAGE REGULATOR — **LM7805**

TO-220
Plastic Package



The Voltages Available allow these Regulators to be used in Logic Systems, Instrumentation, Hi-Fi Audio Circuits and other Solid State Electronic Equipment

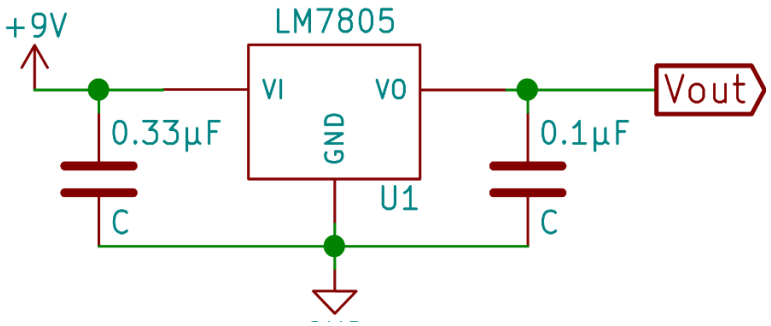
ABSOLUTE MAXIMUM RATINGS

DESCRIPTION	SYMBOL	VALUE	UNIT
Input Voltage	V_{IN}	35	V
Continuous Total Dissipation at T_a=25°C free air Temperature	P_D	2.0	W
Continuous Total Dissipation at T_c=25°C case Temperature	P_D	15	W
Operating free-air, case, or Virtual Junction Temperature Range	T_{OPR}	0 to 150	°C
Storage Temperature Range	T_{stg}	- 65 to +150	°C
Lead Temperature 1.6mm (1/16 inch) from Case for 10 seconds	T_L	260	°C

ELECTRICAL CHARACTERISTICS (T_j=25°C unless specified otherwise)

V_I=10V, I_O=500mA

DESCRIPTION	SYMBOL	TEST CONDITION		MIN	TYP	MAX	UNIT
Output Voltage	V_O		T_j=25°C	4.80		5.20	V
		I_O=5mA ~ 1A, V_I=7V ~ 20V, P≤ 15W	T_j=0 ~ 125°C	4.75		5.25	V
Line Regulation	R_{EGV}	V_I=7.0 ~ 25V	T_j=25°C			100	mV
		V_I=8.0 ~ 12V				50	mV
Ripple Rejection	R_R	V_I=8.0 ~ 18V, f=120Hz	T_j=0 ~ 125°C	62			dB
Load Regulation	R_{EGL}	I_O=5mA ~ 1.5A	T_j=25°C			100	mV
		I_O=250mA ~ 750mA				50	mV
Output Resistance	R_O	f=1KHz	T_j=0 ~ 125°C		0.017		Ω
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O=5mA	T_j=0 ~ 125°C		- 1.1		mV/°C
Output Noise Voltage	V_{NO}	f=10Hz ~ 100KHz	T_j=25°C		40		μV
Dropout Voltage	V_d	I_O=1A	T_j=25°C		2.0		V
Quiescent Current	I_Q		T_j=25°C			8.0	mA
Quiescent Current Change	ΔI_Q	V_I=7.0 ~ 25V	T_j=0 ~ 125°C			1.3	mA
		I_O=5mA ~ 1A				0.5	mA
Short Circuit Output Current	I_{SC}		T_j=25°C		750		mA
Peak Output Current	I_{PK}		T_j=25°C		2.2		A

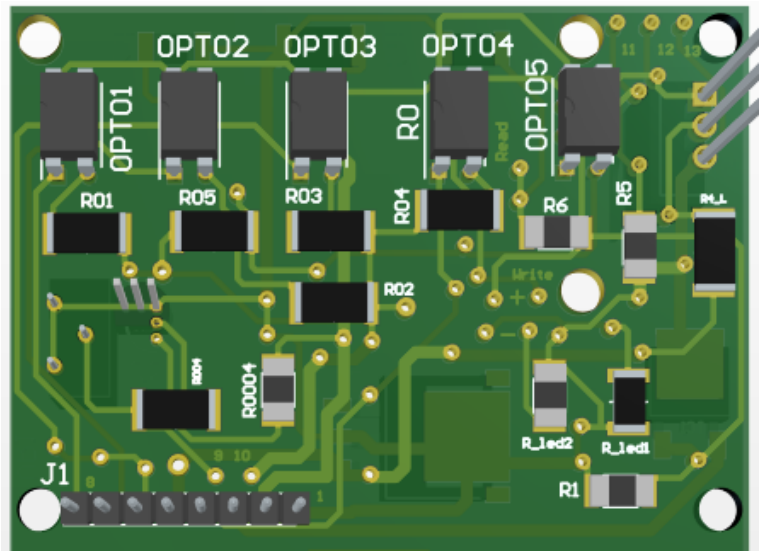


Εικόνα 2.39 Voltage regulator [21] [22].

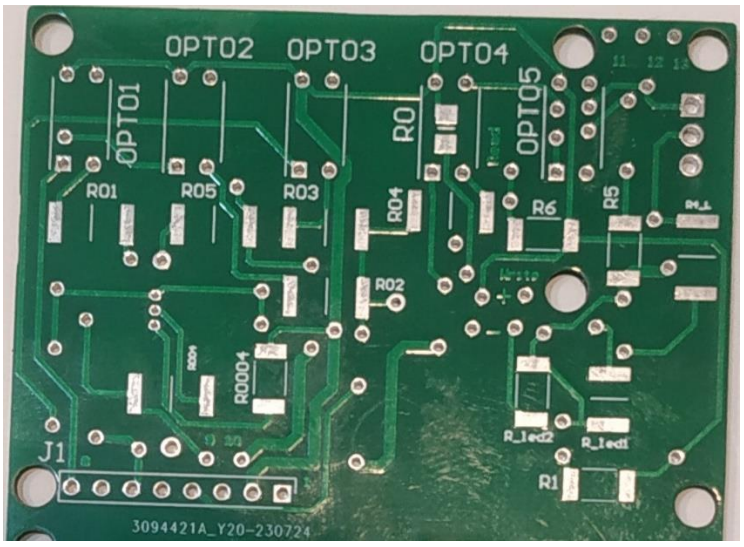
2.7.1 Ο σταθεροποιητής τάσης η ***Voltage regulator*** είναι ένα κύκλωμα ρυθμιστής τάσης που δημιουργεί και διατηρεί μια σταθερή τάση εξόδου, ανεξάρτητα από τις αλλαγές στην τάση εισόδου ή τις συνθήκες φορτίου.

Το ***Voltage regulator*** ICs LM7805 έχει τάση εισόδου 7-35v με σταθεροποιημένη την έξοδο στα 5v [23].

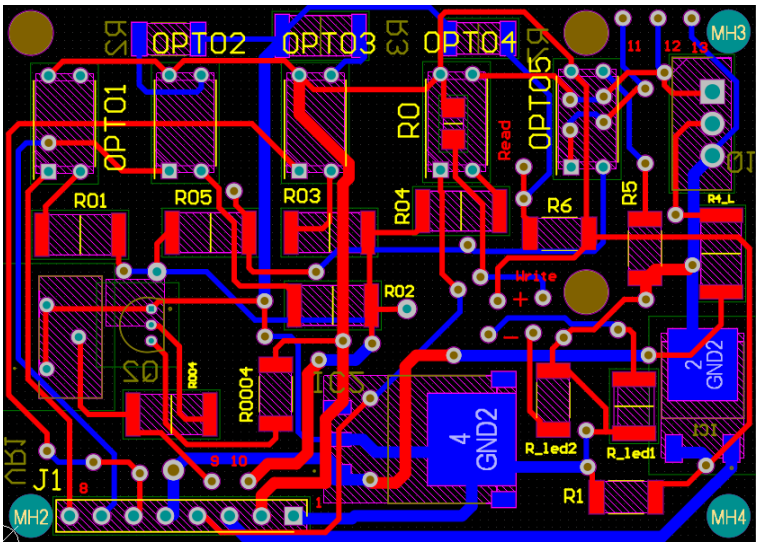
2.9 Πλακέτα μονάδας optocoupler



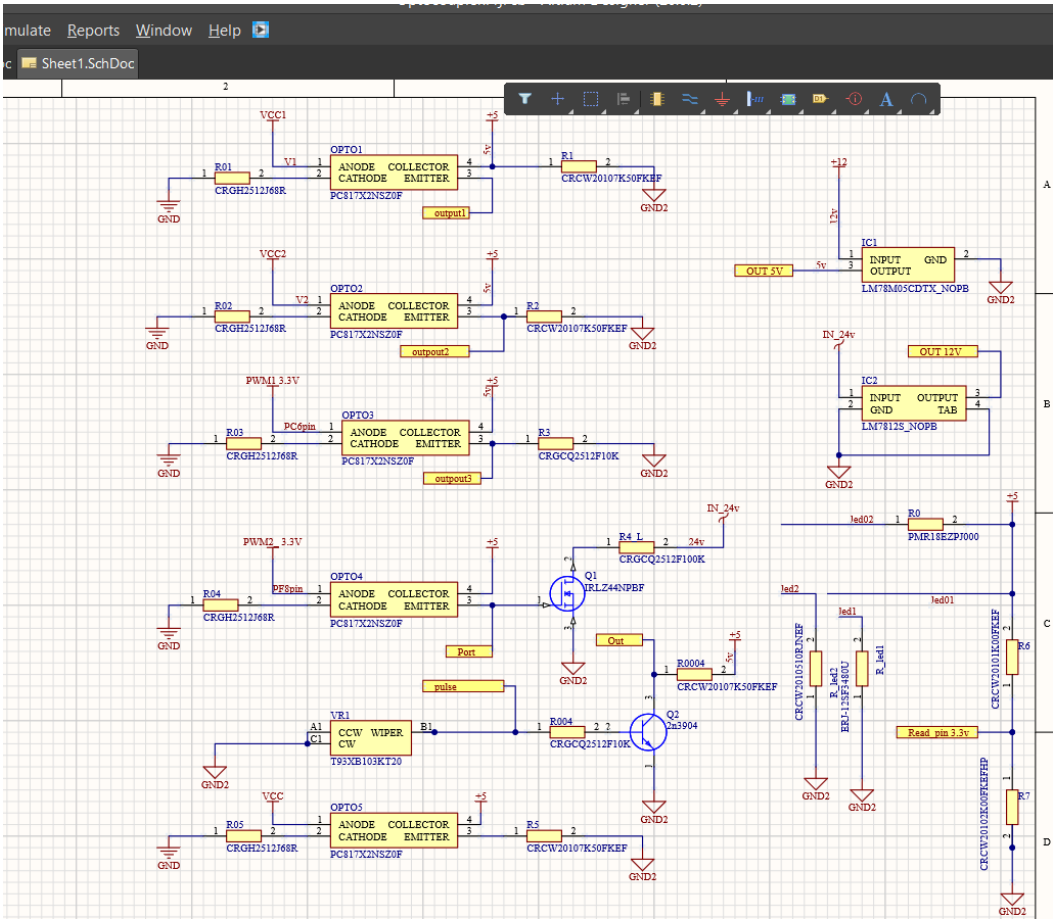
Εικόνα 2.40 optocoupler board 3D



Εικόνα 2.41 optocoupler board κατασκευή από την ***JLCPCB***



Εικόνα 2.42 Διάγραμμα αγωγών του optocoupler board (2 layer)



Εικόνα 2.43 Ηλεκτρονικό διάγραμμα πλακέτας optocoupler (Altium Designer)

2.9.1 Optocoupler

2.9.1.1 Opto-isolator

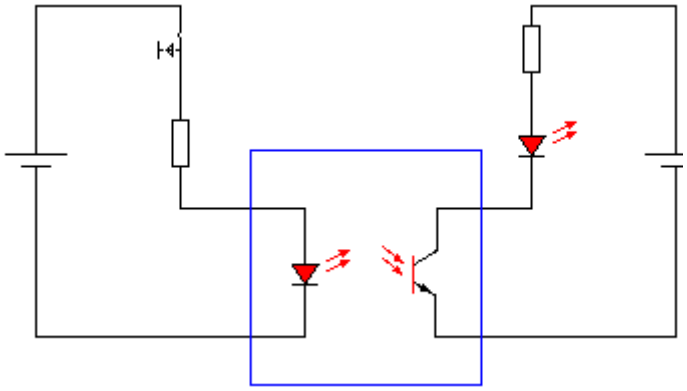
Σε όλα τα κυκλώματα στα οποία χρησιμοποιείται η γαλβανική απομόνωση, χρίζουν οι οπτοζεύκτες.

Οι οπτικοί απομονωτές μεταδίδουν πληροφορίες διαμορφώνοντας το φως . Ο αποστολέας (πηγή φωτός) και ο δέκτης (φωτοευαίσθητη συσκευή) ***δεν είναι ηλεκτρικά συνδεδεμένοι***. Συνήθως συγκρατούνται στη θέση τους μέσα σε μια μήτρα από διαφανές, μονωτικό πλαστικό ή μέσα σε ένα ολοκληρωμένο κύκλωμα . Η οπτική απομόνωση είναι γενικά πολύ περιορισμένη σε ισχύ, αλλά μπορεί να μεταφέρει σήματα δεδομένων πολύ υψηλής ταχύτητας.[24]

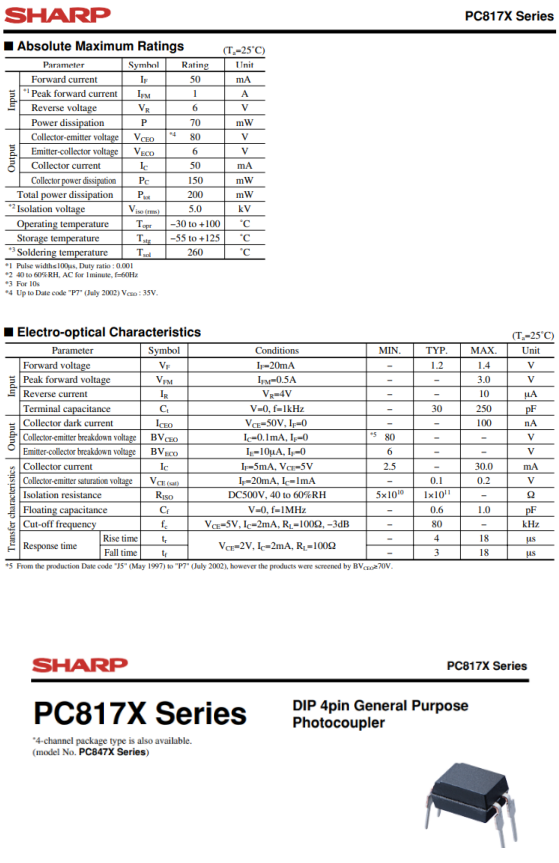
Όπως φαίνετε και από την παραπάνω εικόνα (2.43.) η είσοδος των optocoupler τροφοδοτείται με pwm +3.3V τα οποία είναι προερχόμενα από τον επεξεργαστή και η έξοδος αυτών με +5V, ετσι ώστε με την αναλλαγή της τάσης εισόδου από 0 σε +3.3V να έχουμε pwm +5V στην έξοδο.

Επίσης η έξοδος του optocoupler 4 οδηγεί ένα mosfet (IRLZ44)το οποίο ρυθμίζει το ρεύμα του ηλεκτρομαγνήτη.

Οι είσοδοι των Vcc1 και Vcc2 τροφοδοτούνται με +3.3V και είναι τα σήματα εισόδου για την ορθή και την ανάστροφη πολικότητα της οδήγησεις του κινητήρα.



Εικόνα 2.44 Γαλβανική απομόνωση [24]



SHARP PC817X Series

■ Absolute Maximum Ratings (T_a=25°C)

	Parameter	Symbol	Rating	Unit
Input	Forward current	I_F	50	mA
	*1 Peak forward current	I_{FM}	1	A
	Reverse voltage	V_R	6	V
	Power dissipation	P	70	mW
Output	Collector-emitter voltage	V_{CEO}	*4 80	V
	Emitter-collector voltage	V_{ECO}	6	V
	Collector current	I_C	50	mA
	Collector power dissipation	P_C	150	mW
Total power dissipation		P_{tot}	200	mW
*2 Isolation voltage		$V_{iso\ (rms)}$	5.0	kV
Operating temperature		T_{opr}	−30 to +100	°C
Storage temperature		T_{stg}	−55 to +125	°C
*3 Soldering temperature		T_{sol}	260	°C

*1 Pulse width≤100μs, Duty ratio : 0.001
*2 40 to 60%RH, AC for 1minute, f=60Hz
*3 For 10s
*4 Up to Date code "P7" (July 2002) V_{CEO} : 35V.

■ Electro-optical Characteristics (T_a=25°C)

	Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input	Forward voltage		V_F	I_F=20mA	–	1.2	1.4	V
	Peak forward voltage		V_{FM}	I_{FM}=0.5A	–	–	3.0	V
	Reverse current		I_R	V_R=4V	–	–	10	μA
	Terminal capacitance		C_t	V=0, f=1kHz	–	30	250	pF
Output	Collector dark current		I_{CEO}	V_{CE}=50V, I_F=0	–	–	100	nA
	Collector-emitter breakdown voltage		BV_{CEO}	I_C=0.1mA, I_F=0	*5 80	–	–	V
	Emitter-collector breakdown voltage		BV_{ECO}	I_E=10μA, I_F=0	6	–	–	V
Transfer characteristics	Collector current		I_C	I_F=5mA, V_{CE}=5V	2.5	–	30.0	mA
	Collector-emitter saturation voltage		$V_{CE\ (sat)}$	I_F=20mA, I_C=1mA	–	0.1	0.2	V
	Isolation resistance		R_{ISO}	DC500V, 40 to 60%RH	5×10^{10}	1×10^{11}	–	Ω
	Floating capacitance		C_f	V=0, f=1MHz	–	0.6	1.0	pF
	Cut-off frequency		f_c	V_{CE}=5V, I_C=2mA, R_L=100Ω, −3dB	–	80	–	kHz
	Response time	Rise time	t_r	V_{CE}=2V, I_C=2mA, R_L=100Ω	–	4	18	μs
		Fall time	t_f		–	3	18	μs

*5 From the production Date code "J5" (May 1997) to "P7" (July 2002), however the products were screened by BV_{CEO}≥70V.

SHARP PC817X Series

PC817X Series DIP 4pin General Purpose Photocoupler

*4-channel package type is also available.
(model No. **PC847X Series**)

Εικόνα 2.45 Optocoupler PC817 [25]

2.9.1.2 Υπολογισμός των τιμών της αντίστασης των Optocoupler

VCC1 = 3.3V

3.3V – 1.2V = 2.1V για την R01

Επομένως R01 = 2.1V÷30mA = 68Ω

Η ελάχιστη τιμή της R1 πρέπει να είναι 5V÷30mA = 167Ω

Ο λόγος μεταφοράς ρεύματος ή **CTR**, ο οποίος είναι απλώς ο λόγος του ρεύματος εξόδου προς το ρεύμα εισόδου (I_C / I_F) που συνήθως εκφράζεται ως ποσοστό.

Κατά ελάχιστη τιμή CTR=100%

2.9.1.3 Mosfet IRLZ44N

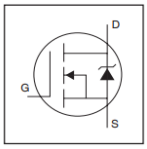
International
IOR Rectifier

PD - 94831

IRLZ44NPbF

HEXFET® Power MOSFET

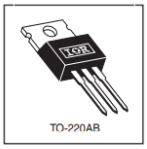
- Logic-Level Gate Drive
- Advanced Process Technology
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated
- Lead-Free



V_{DSS} = 55V

$R_{DS(on)}$ = 0.022Ω

I_D = 47A



Description

Fifth Generation HEXFETs from International Rectifier utilize advanced processing techniques to achieve the lowest possible on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET Power MOSFETs are well known for, provides the designer with an extremely efficient device for use in a wide variety of applications.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.

Absolute Maximum Ratings

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	47	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	33	
I_{DM}	Pulsed Drain Current ①	160	
P_D @T_C = 25°C	Power Dissipation	110	W
	Linear Derating Factor	0.71	W/°C
V_{GS}	Gate-to-Source Voltage	±16	V
E_{AS}	Single Pulse Avalanche Energy ②	210	mJ
I_{AR}	Avalanche Current①	25	A
E_{AR}	Repetitive Avalanche Energy①	11	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.0	V/ns
T_J	Operating Junction and	-55 to + 175	°C
T_{STG}	Storage Temperature Range		
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 screw.	10 lbf•in (1.1N•m)	

Thermal Resistance

	Parameter	Min.	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	——	——	1.4	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	——	0.50	——	
$R_{\theta JA}$	Junction-to-Ambient	——	——	62	

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	55	——	——	V	V_{GS} = 0V, I_D = 250µA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	——	0.070	——	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	——	——	0.022	Ω	V_{GS} = 10V, I_D = 25A ④
		——	——	0.025		V_{GS} = 5.0V, I_D = 25A ④
		——	——	0.035		V_{GS} = 4.0V, I_D = 21A ④
$V_{GS(th)}$	Gate Threshold Voltage	1.0	——	2.0	V	V_{DS} = V_{GS}, I_D = 250µA
g_{fs}	Forward Transconductance	21	——	——	S	V_{DS} = 25V, I_D = 25A
I_{DSS}	Drain-to-Source Leakage Current	——	——	25	µA	V_{DS} = 55V, V_{GS} = 0V
		——	——	250		V_{DS} = 44V, V_{GS} = 0V, T_J = 150°C
I_{GSS}	Gate-to-Source Forward Leakage	——	——	100	nA	V_{GS} = 16V
	Gate-to-Source Reverse Leakage	——	——	-100		V_{GS} = -16V
Q_g	Total Gate Charge	——	——	48	nC	I_D = 25A
Q_{gs}	Gate-to-Source Charge	——	——	8.6		V_{DS} = 44V
Q_{gd}	Gate-to-Drain ("Miller") Charge	——	——	25		V_{GS} = 5.0V, See Fig. 6 and 13 ④
$t_{d(on)}$	Turn-On Delay Time	——	11	——	ns	V_{DD} = 28V
t_r	Rise Time	——	84	——		I_D = 25A
$t_{d(off)}$	Turn-Off Delay Time	——	26	——		R_G = 3.4Ω, V_{GS} = 5.0V
t_f	Fall Time		15			R_D = 1.1Ω, See Fig. 10 ④
L_D	Internal Drain Inductance	——	4.5	——	nH	Between lead, 6mm (0.25in.) from package and center of die contact
L_S	Internal Source Inductance	——	7.5	——		
C_{iss}	Input Capacitance	——	1700	——	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	——	400	——		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	——	150	——		f = 1.0MHz, See Fig. 5

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	——	——	47	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode) ①	——	——	160		
V_{SD}	Diode Forward Voltage	——	——	1.3	V	T_J = 25°C, I_S = 25A, V_{GS} = 0V ④
t_{rr}	Reverse Recovery Time	——	80	120	ns	T_J = 25°C, I_F = 25A
Q_{rr}	Reverse RecoveryCharge	——	210	320	nC	di/dt = 100A/µs ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				

Notes:

① Repetitive rating; pulse width limited by max. junction temperature. (See fig. 11)

② V_{DD} = 25V, starting T_J = 25°C, L = 470µH R_G = 25Ω, I_{AS} = 25A. (See Figure 12)

③ I_{SD} ≤ 25A, di/dt ≤ 270A/µs, V_{DD} ≤ $V_{(BR)DSS}$, T_J ≤ 175°C

④ Pulse width ≤ 300µs; duty cycle ≤ 2%.

Εικόνα 2.46 Χαρακτηριστηκά Mosfet IRLZ44N [26]

Η πλακέτα των Optocoupler ενσωματώνει και ενα mosfet το IRLZ44N το οποίο χρησιμοποιείτε για τον έλεγχο ρύθμισης του ηλεκτρομαγνήτη με συχνότητα 3Khz PWM (pulse width modulation).

2.10 Εξωτερική είσοδος USB.

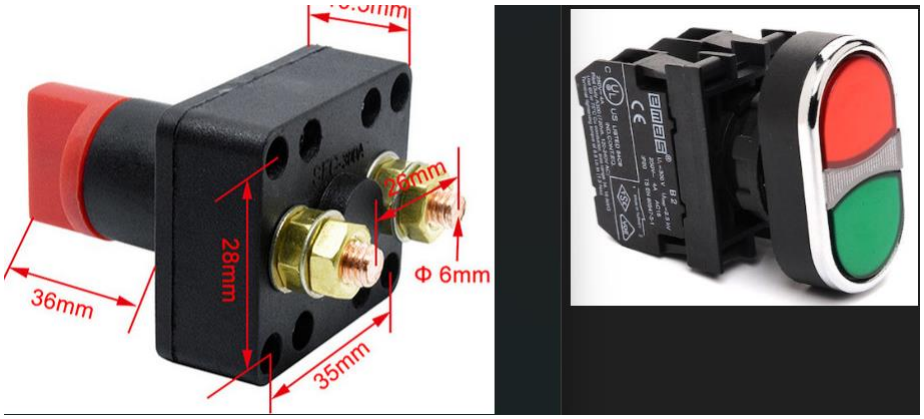
Εικόνα 2.47 Εξωτερική είσοδος USB

Ο **Ενιαίος Σειριακός Δίαυλος**, γνωστός και ως **Universal Serial Bus** ή απλά **USB**, είναι ένα σύστημα διαύλου, το οποίο χρησιμοποιείται για την επικοινωνία ενός υπολογιστή και όχι μόνο με περιφερειακά συστήματα.

Για την σύνδεση της αναπτυξιακής πλακέτας με τον Η/Υ χρειάζεται μόνο ένα καλώδιο USB τύπου A σε micro B. Στην κατασκευή χρησιμοποιήθηκε μια μικρή προέκταση usb 10cm, με είσοδο USB η οποία είναι τοποθετημένη εξωτερικά του πλαστικού κιβωτίου.Η χρήση τώρα πλέον γίνεται με καλώδιο USB τύπου A σε A.

Η ίδια θύρα χρησιμοποιείται και για τον προγραμματισμό της εσωτερικής μνήμης του μικροελεγκτή τύπου flash με την χρήση εξωτερικού λογισμικού STM32CubeIDE της εταιρείας ST και την αποσφαλμάτωση της εφαρμογής.[27]

2.11 Μπουτόν ελέγχου - Γενικός διακόπτης



Εικόνα 2.48 Γενικώς διακόπτης – Διπλό μπουτόν [28] [29]

2.11.1 Αρχή λειτουργίας

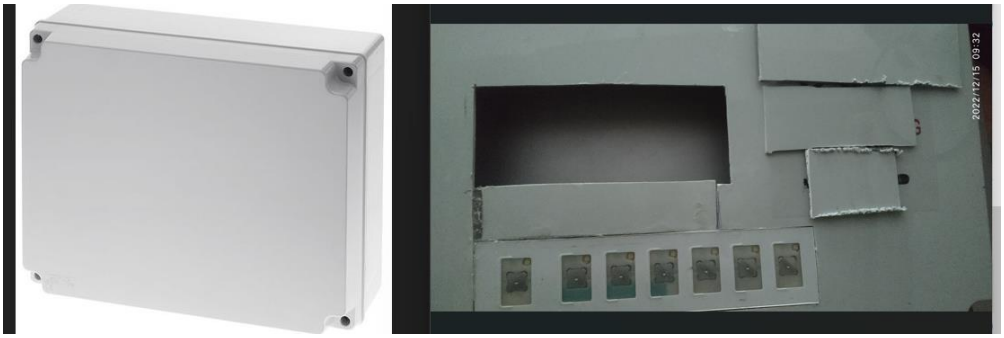
Οι διακόπτες έχουν σημεία με τα οποία συνδέονται με το κύκλωμα τα οποία ονομάζονται *ακροδέκτες*. Κάθε διακόπτης έχει δύο καταστάσεις, την κατάσταση που είναι *κλειστός* και την κατάσταση που είναι *ανοικτός*. Όταν ένας διακόπτης είναι ανοικτός δεν επιτρέπει τη διέλευση ηλεκτρικού ρεύματος μεταξύ των ακροδεκτών του, ενώ όταν είναι κλειστός επιτρέπει τη διέλευση ηλεκτρικού ρεύματος μεταξύ των ακροδεκτών του. Ο διακόπτης διατηρεί την κατάσταση στην οποία βρίσκεται, ενώ αυτή μεταβάλλεται μόνο από εξωτερικούς του στοιχείου παράγοντες, όπως είναι το πάτημα ενός κουμπιού ή αλλαγή στο ηλεκτρικό πεδίο. Κάθε κλειστός διακόπτης μπορεί να ανοίξει, ενώ κάθε ανοικτός διακόπτης μπορεί να κλείσει.

Η αλλαγή της κατάστασης ενός διακόπτη γίνεται είτε μεταβάλλοντας την αγωγιμότητα ενός τμήματός του που παρεμβάλλεται μεταξύ των ακροδεκτών του είτε αλλάζοντας την απόσταση μεταξύ δύο αγώγιμων μερών του, που ονομάζονται **επαφές**. Συνήθως ο πρώτος τρόπος χρησιμοποιείται σε αυτόματους διακόπτες, ενώ ο δεύτερος σε χειροκίνητους. Σε αυτήν την περίπτωση μία επαφή είναι σταθερή στη θέση της, ενώ η άλλη μετακινείται μηχανικά.

Για να διέλθει ηλεκτρικό ρεύμα μέσω ενός διακόπτη, πρέπει να είναι κλειστός και να εφαρμοστεί στους ακροδέκτες του διαφορά δυναμικού. Για να μη διέλθει ηλεκτρικό ρεύμα αρκεί να είναι ανοικτός, αν και είναι πιθανό όταν είναι κλειστός να μη διαρρέεται από ηλεκτρικό ρεύμα, γιατί δεν υπάρχει τάση.

Επιπλέον, οι διακόπτες μεταφέρουν τις στοιχειώδεις πληροφορίες ***0 ή ψευδής*** όταν είναι ανοιχτοί και ***1 ή αληθής*** όταν είναι κλειστοί, όπως συμβαίνει στους υπολογιστές.[30]

2.12 Πλαστικό κιβώτιο



Εικόνα 2.49 Πλαστικό κιβώτιο[31]

Το πλαστικό κιβώτιο το οποίο χρησιμοποιήθηκε για την κάλυψη όλων των ηλεκτρονικών εξαρτημάτων, καλωδίων κλπ.

Οι διαστάσεις του οποίου είναι 310x240x120, στην εικόνα φαίνεται η αρχική κατάσταση καθώς και η μετέπειτα διαμόρφωση.

2.13 Πλαστική καρούλα

Εικόνα 2.50 Διαμόρφωση καρούλας [32]

2.13.1 Διαμόρφωση καρούλας

Εχεί διαμορφωθεί μεγαλύτερη οπή για την θέση του ρουλεμάν, επίσης έχει κατασκευαστεί όπως φαίνεται και στην εικόνα μεταλικός δίσκος ενός χιλιοστού ώστε να έχει περισότερη αντοχή, και καποια προδιαγραφή για το φρένο. Να επισημάνω ότι όλα τα μεταλλικά εξαρτήματα έχουν σχεδιαστεί, με το σχεδιαστικό πρόγραμμα corel DRAW 12 , επίσης η κοπεί των μετάλλων έγινε με laser από την εταιρία DELIGIANNIS DIMITRIOS A.S.M.P.C.

2.14 Μεταλική βάση

Εικόνα 2.51 Μεταλλική βάση

2.14.1 Μεταλλική πλάκα 400x350x3 σχεδιασμένη με το πρόγραμμα Corel draw 12 και κομμένη με laser για την ακριβή τοποθέτηση των υλικών.

2.15 Ηλεκτρομαγνητικό φρένο

Εικόνα 2.52 Ηλεκτρομαγνητικό φρένο [33]

2.15.1 Electromagnetic clutch

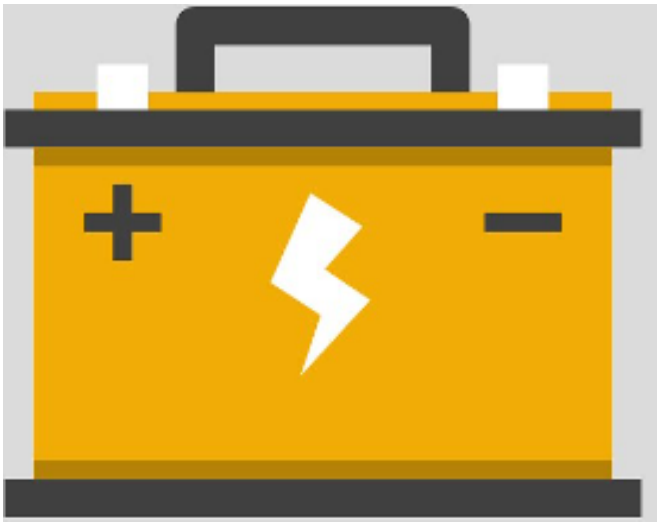
Οι ηλεκτρομαγνητικοί συμπλέκτες λειτουργούν μέσω ηλεκτρικής ενεργοποίησης, αλλά μεταδίδουν τη ροπή μηχανικά . Όταν εφαρμόζεται τάση/ρεύμα στο πηνίο του συμπλέκτη, το πηνίο γίνεται ηλεκτρομαγνήτης και παράγει μαγνητικές γραμμές ροής. Αυτή η ροή στη συνέχεια μεταφέρεται μέσω του μικρού κενού αέρα μεταξύ του πεδίου και του ρότορα.[33]

Η σύμπλεξη πραγματοποιείται μέσω ηλεκτρομαγνητικών φορτίων που δημιουργούνται μεταξύ των δύο μεταλλικών επιφανειών επαφής, από τις οποίες η μία είναι συνδεδεμένη με τον άξονα του κινητήρα (κινητήριο τμήμα).

Στην κατασκευή, ο μηχανισμός που είναι συνδεδεμένος με το κινητήριο τμήμα (ρότορας) εμπεριέχει ένα μαγνήτη ο οποίος όταν τροφοδοτηθεί από ρεύμα, δημιουργεί ένα ηλεκτρομαγνητικό πεδίο το οποίο προσελκύει τον οπλισμό (καρούλα) στη μεταξύ τους επαφή. Με αυτή την επαφή επιτυγχάνεται σύμπλεξη, οπότε πραγματοποιείται η μετάδοση της κίνησης. Όταν επιθυμούμε αποσύμπλεξη, αυτό γίνεται με την διακοπή παροχής ρεύματος στο μαγνήτη ώστε να εξαλειφθεί το μαγνητικό πεδίο.

Στο ηλεκτρομαγνητικό φρένο , ανάμεσα στις επιφάνειες επαφής υπάρχει λάδι εμπλουτισμένο με ρινίσματα σιδήρου ή απλά ρινίσματα σιδήρου. Η εφαρμογή ενός ηλεκτρομαγνητικού πεδίου μεταξύ των επιφανειών έχει ως αποτέλεσμα τον προσανατολισμό των ρινισμάτων και την έλξη τους. Με την κίνηση αυτή, το κινητήριο τμήμα παρασύρει το κινούμενο και μεταφέρει την ροπή από τον κινητήρα στην(καρούλα). Ο έλεγχος της σύμπλεξης γίνεται με τη μεταβολή του ηλεκτρικού ρεύματος που παρέχεται και δημιουργεί το ηλεκτρομαγνητικό πεδίο.[33]

2.16 Τροφοδοσία συστήματος



Εικόνα 2.53 Σχήμα μπαταρία αυτοκινήτου [34]

2.16.1 Μπαταρία 12v / 60Ah

Περιγραφή

Ο συσσωρευτής στην ηλεκτρολογία είναι χημική πηγή ρεύματος, ικανή να αποθηκεύσει ηλεκτρική ενέργεια (αφού τη μετατρέψει σε χημική) και όταν χρειαστεί, να την αποδώσει σε εξωτερικό κύκλωμα. Αποτελείται από δοχείο κατασκευασμένο από μονωτικό υλικό (εβονίτη, πλαστικό, γυαλί) με ηλεκτρολύτη (οξύ ή αλκάλιο), στο οποίο βυθίζονται τα ηλεκτρόδια. Η σύνδεσή τους σε εξωτερικό κύκλωμα προκαλεί σε αυτό διέλευση ρεύματος (εκφόρτιση του ηλεκτρικού συσσωρευτή)..

Ο εκφορτισμένος ηλεκτρικός συσσωρευτής φορτίζεται όταν περάσει από αυτόν συνεχές ρεύμα από άλλη πηγή, ενώ ταυτόχρονα στον ηλεκτρικό συσσωρευτή γίνονται αντίστροφες χημικές διεργασίες, με τις οποίες η ηλεκτρική ενέργεια μετατρέπεται σε χημική. Ο ηλεκτρικός συσσωρευτής χαρακτηρίζεται από τη χωρητικότητα, δηλ. την ποσότητα του ηλεκτρισμού σε αμπερώρια, που μπορεί ο συσσωρευτής να δώσει στο κύκλωμα που τροφοδοτεί, από τη μέση τάση σε volt κατά το χρόνο της φόρτισης και εκφόρτισης, από την ειδική ενέργεια κατά βάρος και όγκο, δηλ. την ενέργεια σε βατώρια που παρέχεται κατά την εκφόρτιση από 1 kg βάρους ή 1 δεκατόμετρο του όγκου του ηλεκτρικού συσσωρευτή, από την απόδοση κατά χωρητικότητα, δηλ. τον λόγο της ποσότητας των αμπερωρίων που αποδίδεται κατά την εκφόρτιση προς την ποσότητα των αμπερωρίων που απορροφάται κατά τη φόρτιση, από την απόδοση κατά ενέργεια (ή βαθμό απόδοσης), δηλ. το λόγο της

ενέργειας που αποδίδεται κατά την εκφόρτιση προς την ενέργεια που απορροφάται κατά τη φόρτιση.

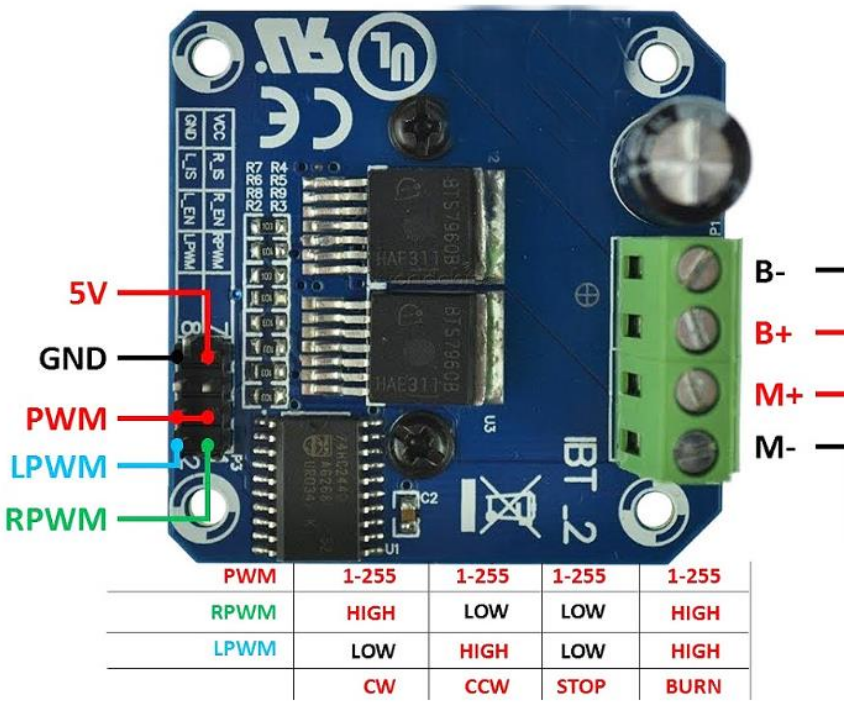
Οι συσσωρευτές (μπαταρίες) αναγράφουν τη χωρητικότητά τους σε Ah (αμπέρ ανά ώρα). Έτσι, ένας συσσωρευτής των 12 volt και 100 Ah παρέχει 12 X 100 = 1.200 watt συνεχούς ρεύματος (DC) για 1 ώρα ή 120 watt για 10 ώρες ή 12 watt για 100 ώρες. Ένας ακόμη σημαντικός δείκτης είναι αυτός που μας παρέχει την πληροφορία σχετικά με τον ρυθμό εκφόρτισης με βάση τον οποίο ο συσσωρευτής μπορεί να δώσει τις αναγραφόμενες Ah. Έτσι, μια μπαταρία που γράφει ότι έχει χωρητικότητα 100 Ah σε C20 σημαίνει ότι οι 100 Ah επιτυγχάνονται όταν η σταδιακή εκφόρτιση διαρκεί 20 ώρες. Για λιγότερες ώρες (π.χ. C10, 10 ώρες) παίρνουμε λιγότερες Ah, ενώ σε σταδιακή εκφόρτιση περισσότερων ωρών (π.χ. C100, 100 ώρες) παίρνουμε σημαντικά περισσότερες Ah.

Ο δείκτης C10, C20 και C100 δεν είναι αυθαίρετος. Περιγράφεται από συγκεκριμένες προδιαγραφές και είναι τυποποιημένος. ΟΛΕΣ οι μπαταρίες (όλες οι μπαταρίες μολύβδου δηλαδή) έχουν να δώσουν διαφορετική χωρητικότητα σε ρυθμό εκφόρτισης 10 ωρών (C10), μεγαλύτερη σε C20 και ακόμη μεγαλύτερη σε C100. Είναι προτιμότερο κατά τη λειτουργία τους να παρέχουν λίγα watt για περισσότερες ώρες παρά πολλά watt για λίγες, επειδή στη δεύτερη περίπτωση μειώνεται δραστικά ο χρόνος ζωής τους. Ποτέ δεν εκφορτίζουμε τελείως τους συσσωρευτές γιατί αυτό μπορεί να τους καταστρέψει.

Υπάρχουν συσσωρευτές διαφόρων τύπων με διαφορετικό βαθμό επιτρεπόμενης εκφόρτισης. Ο γενικός κανόνας είναι κατά τη συνηθισμένη χρήση να μην επιτρέπουμε εκφόρτιση πάνω από 50% περίπου και μόνο σε εξαιρετικές περιπτώσεις ανάγκης να φθάνουμε το 80%.

Για την τροφοδοσία του συστήματος αλιείας, θα υπολογίσουμε, ότι όταν λειτουργεί η μηχανή στην μέγιστη ισχύ δηλαδή στα 250w για 1h, που σημαίνει 12x20,83Ah=250w. Εάν χρησιμοποιούμε μπαταρία 12v και 60Ah θα μας καλύψη συνεχούς λειτουργείας, 2.88 ώρες. Αλλα επειδή συνήθως η λειτουργεία δεν είναι συνεχώμενη, μεγαλώνει κατά πολύ ο χρόνος αποφώρτησεις, και έτσι ο χρόνος στο ψάρεμα είναι μεγαλύτερος. Ανάλογα λοιπόν με τις ανάγκες που έχουμε θα αγοράσουμε και την κατάλληλη μπαταρία σε Ah.[35]

2.17 Πλακέτα οδήγησεις dc κινητήρα



Εικόνα 2.54 Πλακέτα οδήγησεις κινητήρα BTS7960 [36]

2.17.1 Χαρακτηριστικά προγράμματος οδήγησης κινητήρα BTS7960

Το BTS7960 είναι μια μονάδα οδήγησης κινητήρα υψηλού ρεύματος πλήρους γέφυρας (H bridge driver).

Τα βασικά χαρακτηριστικά είναι:

- Τάση εισόδου: 6V έως 27V
- Μέγιστο επιτρεπόμενο ρεύμα: 6-11 A, typically 8.5 A.
- Δυνατότητα PWM: έως 25 kHz
- Δύο ακίδες εξόδου PWM για έλεγχο ταχύτητας σε άμεσες και αντίστροφες κατευθύνσεις
- Δύο ακίδες εξόδου EN για τον έλεγχο των κινητήρων
- Δύο ακίδες εισόδου IS για προστασία από υψηλό ρεύμα και θερμότητα

Αυτές οι μονάδες ελέγχουν τους κινητήρες DC χρησιμοποιώντας την τεχνική PWM (Pulse Width Modulation). Αυτές οι μονάδες μετατρέπουν μια σταθερή τάση εισόδου σε μια μεταβλητή τάση για τον κινητήρα. Η ταχύτητα μπορεί να ελεγχθεί αλλάζοντας την τάση του κινητήρα DC. Τα PWM έχουν συνήθως μια σταθερή συχνότητα και μπορούν να ελεγχθούν ελέγχοντας το χρόνο που ο παλμός είναι ΥΨΗΛΟΣ (Κύκλος λειτουργίας).

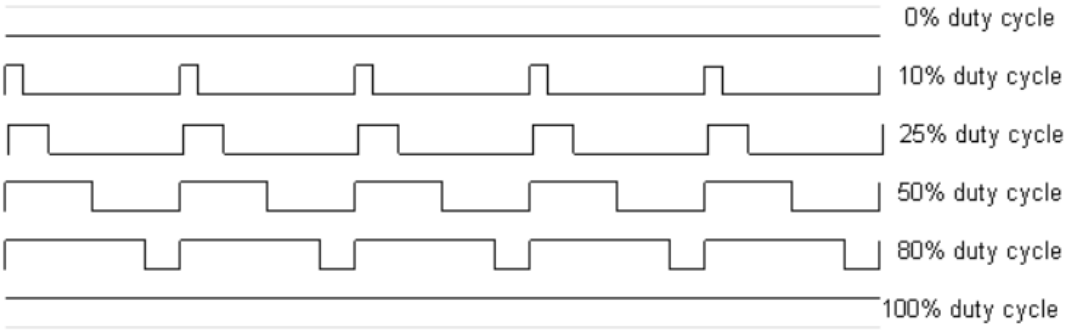
Ο κύκλος λειτουργείας ορίζεται ως ο λόγος του εύρου του παλμού (tw) προς την περίοδο (T) του και εκφράζεται ως ποσοστό.

$$DC = \frac{\text{tw}}{\text{T}} * 100\%$$

Ο τύπος από την τιμή του TIM8 για την οδήγηση του κινητήρα είναι:

$$\frac{APB2}{\text{x}*\text{Counter Period}} = Presc \quad x=3016\ Hz$$

$$\frac{100*10\text{^}6}{\text{x}*255} = 131 - 1$$



Εικόνα 2.55 Διαμόρφωση εύρους παλμού (PWM)[37]

Pin οδήγησης κινητήρα

- **VCC** : Τροφοδοσία μονάδας +5V
- **GND** : Γείωση
- **Channel 1**
- **IS-R:** Σήμα εισόδου για ανίχνευση υψηλού ρεύματος – δεξιά περιστροφή
- **IS-L:** Σήμα εισόδου για ανίχνευση υψηλού ρεύματος – αριστερή περιστροφή
- **Channel 2**

- **EN-R:** Σήμα εξόδου για έλεγχο κατεύθυνσης κινητήρα – δεξιά περιστροφή
- **EL-L:** Σήμα εξόδου για έλεγχο κατεύθυνσης κινητήρα – αριστερή περιστροφή
- **WM-R:** Σήμα PWM για τον έλεγχο της ταχύτητας του κινητήρα – δεξιά περιστροφή
- **PWM-L:** Σήμα PWM για τον έλεγχο της ταχύτητας του κινητήρα – αριστερή περιστροφή

Pin κινητήρα (Υψηλό ρεύμα):

- **M+:** Θετικός πόλος κινητήρας
- **M-:** Αρνητικό πόλος κινητήρα
- **B+:** Θετικός πόλος μπαταρίας
- **B-:** Αρνητικός πόλος μπαταρίας

Επισήμανση ότι στην κατασκευή του συστήματος δεν χρησιμοποιήται το μέγιστο PWM.

Εως (165 dec) τιμή στην οποία αντιστοιχή ποσοστό 64.7%

Τα σήματα ενεργοποιησείς εισόδου για τον ελέγχο δεξιά η αριστερη κίνηση, τροφοδοτούνται με 5V καθώς επίσης και είσοδος PWM [36].

2.18 Μικροελεγκτής

2.18.1 Εισαγωγή

Σε συστήματα τα οποία έχουν κρισιμότητα ως προς τον χρόνο (Real time) και όχι μόνο, οι μικροελεγκτές προσφέρουν την κατάλληλη απάντηση.

Το υλικό που βασίζεται σε μικροελεγκτή μπορεί να χρησιμοποιηθεί χωρίς μετατροπές (με παραμετροποίηση του προγράμματος) σε ποκιλία εφαρμογών, ενώ ένα ψηφιακό σύστημα που κατασκευάζεται με πύλες χρησιμοποιείται μόνο σε μια εφαρμογή. Έτσι, με τους μικροεπεξεργαστές το πρόβλημα της σχεδίασης ψηφιακών συστημάτων μετατοπίστηκε από το υλικό στο λογισμικό και η σχεδίαση απαλλάχθηκε από την ακαμψία του υλικού[38].-> ΔΟΜΗ & ΛΕΙΤΟΥΡΓΙΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ 2ου ΚΥΚΛΟΥ σελ.71

Ένας μικροελεγκτής είναι ένα μικρό υπολογιστικό κύκλωμα, σχεδιασμένο σε ένα και μόνο ολοκληρωμένο κύκλωμα υψηλής κλίμακας ολοκλήρωσης. Όπως κάθε υπολογιστικό κύκλωμα, περιέχει κεντρική μονάδα επεξεργασίας, έναν αριθμό καταχωρητών, κυκλώματα μνήμης και κυκλώματα ελέγχου περιφερειακών συσκευών. Κάθε μικροελεγκτής είναι λοιπόν ικανός να ανταλλάξει σήματα με το εξωτερικό περιβάλλον, να εκτελέσει πράξεις ανάμεσα σε μεταβλητές και να καταχωρήσει κάποιες τιμές στη μνήμη RAM που διαθέτει.

Κάθε μικροελεγκτής περιέχει μέσα σε ένα και μοναδικό ολοκληρωμένο κύκλωμα τα παρακάτω στοιχεία:

Έναν αριθμό από καταχωρητές ειδικού σκοπού (συσσωρευτή, καταχωρητή κατάστασης, μετρητή προγράμματος, καταχωρητή εντολών, καταχωρητή δείκτη), εσωτερικούς χρονιστές - απαριθμητές αριθμητική και λογική μονάδα (ALU), μονάδα αποκωδικοποίησης εντολών.

Βασικά στοιχεία ενός μικροελεγκτή αποτελούν:

Η μνήμη προγράμματος (ROM ή EPROM) και η μνήμη καταχωρητών / μεταβλητών (RAM).

Εικόνα 2.56 Βασική δομή ενός μικροελεγκτή[39]

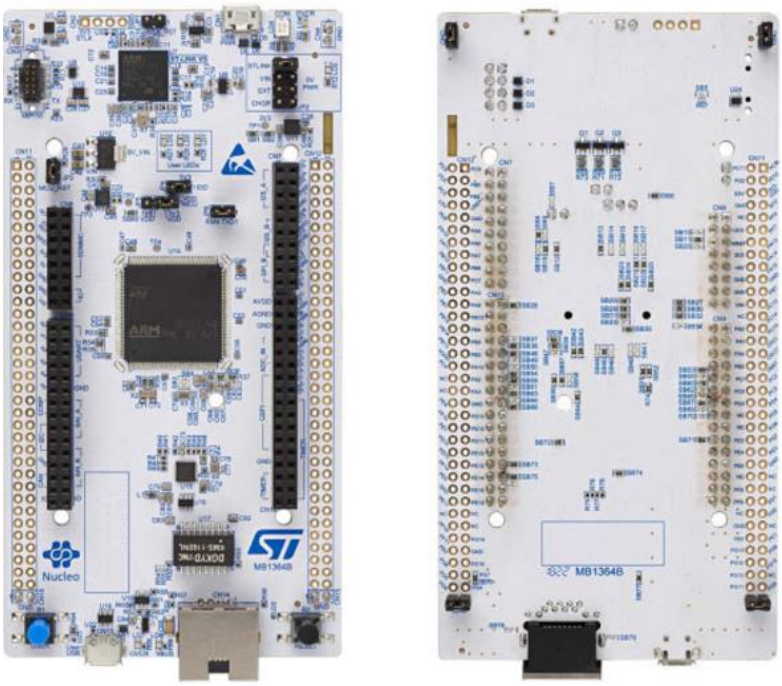
Στους μικροελεγκτές διακρίνουμε επίσης τα κυκλώματα χρονισμού και ελέγχου Τέλος, βασικά μέρη ενός μικροελεγκτή είναι παράλληλες θύρες εισόδου/εξόδου άλλα περιφερειακά κυκλώματα (UART, A/D μετατροπείς κλπ.).

Μέσα από τις θύρες I/O ένας μικροελεγκτής μπορεί να δέχεται σήματα εισόδου με τη μορφή λογικών ψηφιακών καταστάσεων, χαρακτήρες ή bytes δεδομένων με την τεχνική της ασύγχρονης ή της σύγχρονης σειριακής επικοινωνίας, σήματα διακοπών, ή σε ορισμένες περιπτώσεις και αναλογικά σήματα, τα οποία στη συνέχεια μετατρέπονται σε ψηφιακά. Επίσης μπορεί να αποστέλλει σήματα σε άλλες συσκευές μέσα από θύρες εξόδου, να οδηγεί ηλεκτρονόμους, διόδους LED και άλλα κατάλληλα κυκλώματα, που συνήθως περιλαμβάνονται σε κάθε μορφής αυτοματισμό.[39]

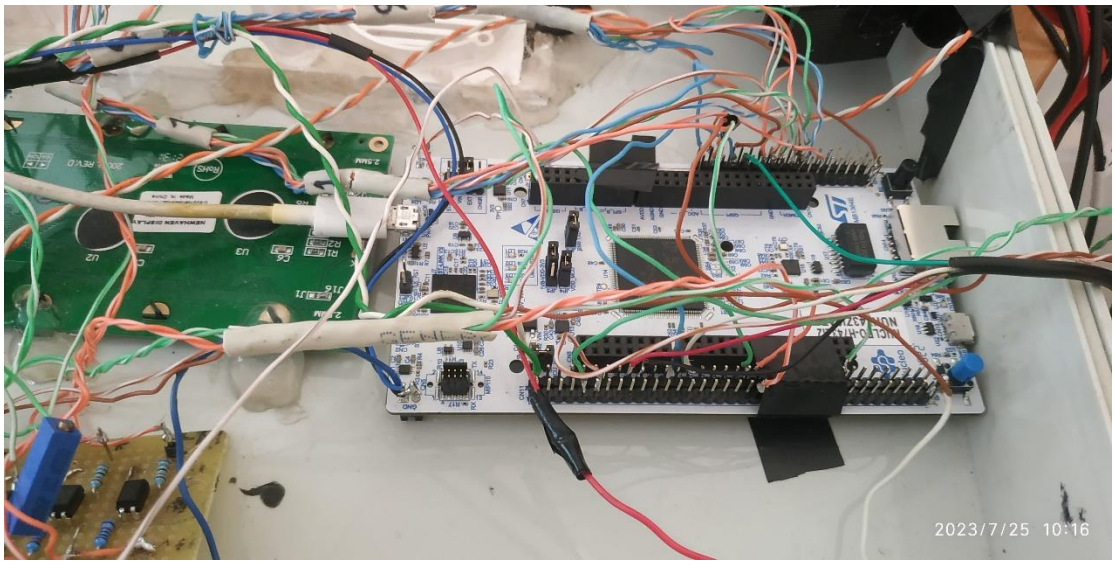
3.1 STM32h743zit6u

Η υλοποίηση της παρούσας πτυχιακής επιτεύχθηκε με τον μικροελεκτή STM32H743ZIT6U της εταιρίας ST microelectronics.

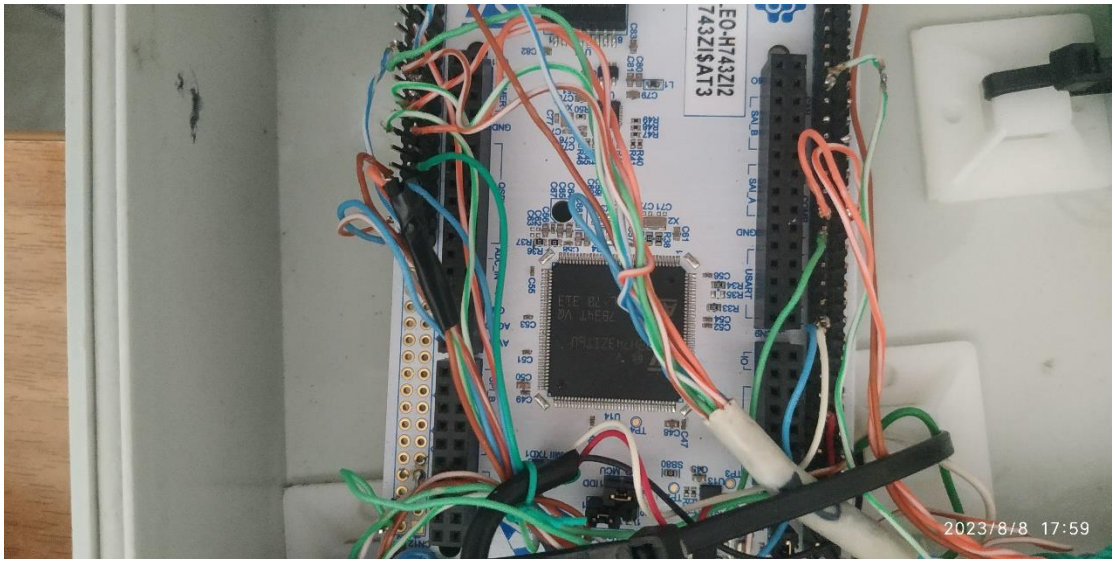
Ο πυρήνας επεξεργαστή **Arm® Cortex®-M7 32-bit** προσφέρει την καλύτερη απόδοση μεταξύ της σειράς Cortex-M. Διαθέτει αποκλειστικά **μπλοκ IP Επεξεργασίας Ψηφιακού Σήματος** Digital signal processing **(DSP)** , συμπεριλαμβανομένης μιας προαιρετικής **μονάδας κινητής υποδιαστολής διπλής ακρίβειας (FPU)** . Τα χαρακτηριστικά υψηλής απόδοσης του πυρήνα Arm Cortex-M7 αντιμετωπίζουν τέλεια τις απαιτητικές εφαρμογές ελέγχου ψηφιακού σήματος, οι οποίες απαιτούν αποτελεσματικό, εύχρηστο έλεγχο, χωρίς την ανάγκη περίπλοκων λειτουργικών συστημάτων. Τυπικά παραδείγματα εφαρμογών περιλαμβάνουν το IoT, τον έλεγχο κινητήρα, τη διαχείριση ισχύος, τον ενσωματωμένο ήχο, συμπεριλαμβανομένης της αναγνώρισης φωνής, τον βιομηχανικό και οικιακό αυτοματισμό, την υγειονομική περίθαλψη και τις εφαρμογές ευεξίας.[40]



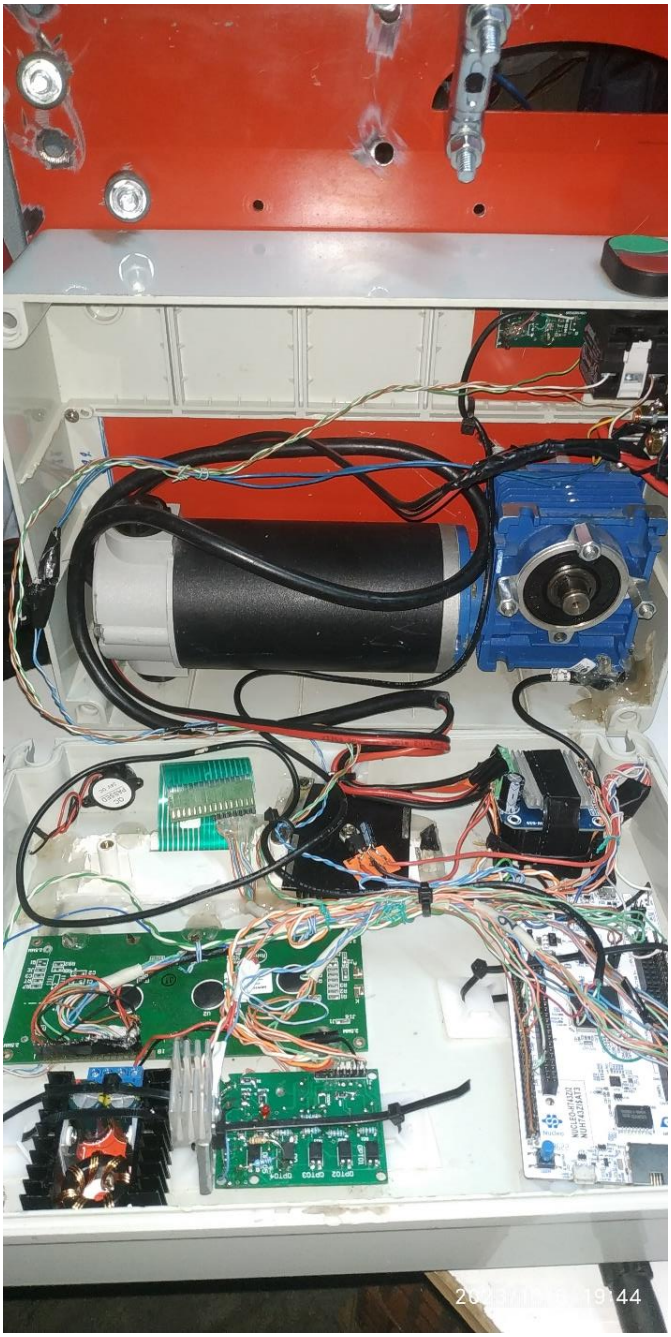
Εικόνα 3.1 Πλακέτα Nucleo- STM32H743zi2 [41]



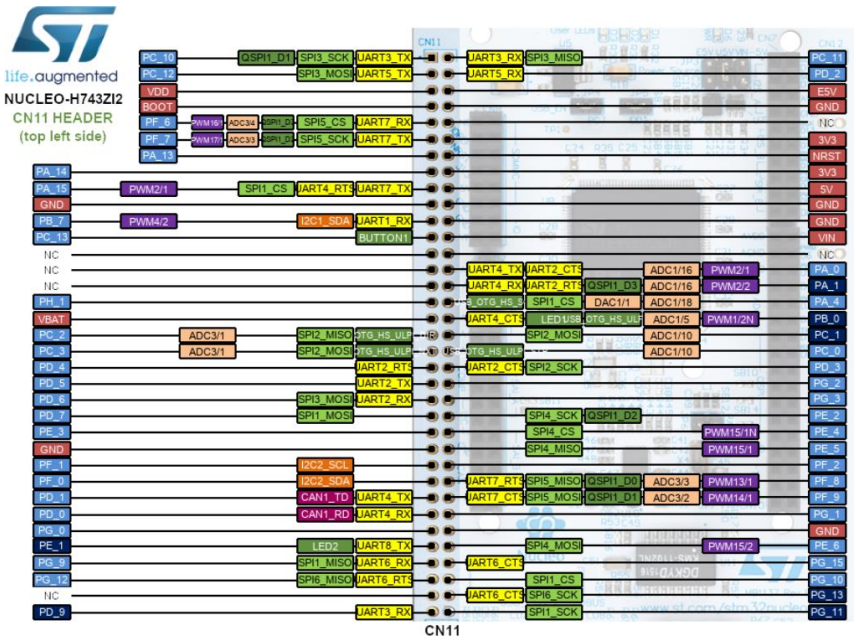
Εικόνα 3.2 Συνδέσεις καλωδίων με επεξεργαστή



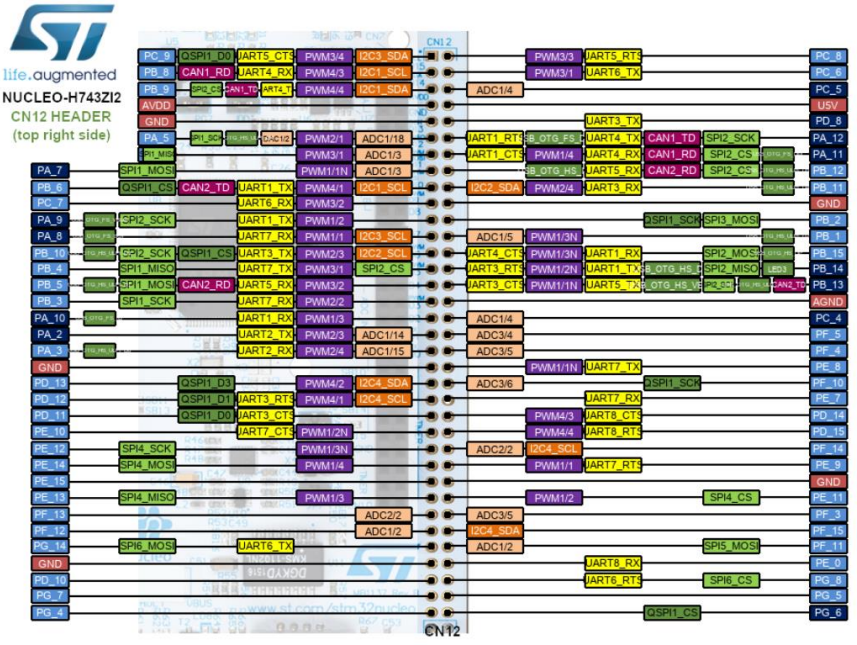
Εικόνα 3.3 Συνδέσεις καλωδίων με επεξεργαστή



Εικόνα 3.4 Ολική φωτογραφία κατασκευής



Εικόνα 3.5 Κονέκτορ C11 Nucleo- STM32H743zi2 [42]

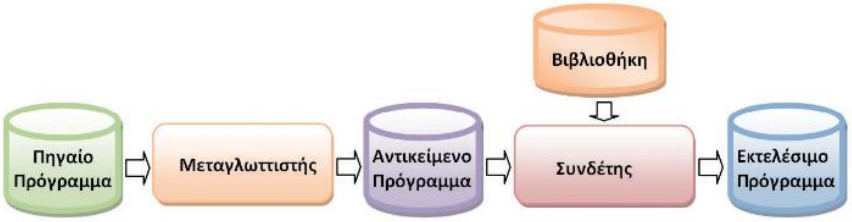


Εικόνα 3.6 Κονέκτορ C12 Nucleo- STM32H743zi2 [42]

Η αναπτυξιακή πλακέτα STM32H743zi2 Nucleo-144 διαθέτει εργαλείο προγραμματισμού και εντοπισμού σφαλμάτων STLINK-V3E. Οι debuggers είναι εργαλεία λογισμικού που επιτρέπουν στον προγραμματιστή να παρακολουθεί την εκτέλεση του προγράμματος, να τη σταματά, να την επανεκκινεί, να ορίζει σημεία διακοπής (breakpoints) και να αλλάζει τιμές στη μνήμη [43]. Υπάρχουν δύο διαφορετικοί τρόποι προγραμματισμού ή εντοπισμού σφαλμάτων του ενσωματωμένου STM32H743zi2 MCU:• Χρήση του ενσωματωμένου STLINK-V3E• Χρήση εξωτερικού εργαλείου εντοπισμού σφαλμάτων συνδεδεμένο στην υποδοχή MIPI-10 (CN5).Το ενσωματωμένο STLINK-V3E υποστηρίζει μόνο SWD(serial wire debug) και VCP (virtual com port) για συσκευές STM32H7 [44].

3.2 Ποια η διαδικασία ανάπτυξης προγράμματος

Η διαδικασία προγραμματισμού χρίζει ένα σύνολο εργαλείων, από τον πηγαίο κώδικα εως και την ολοκλήρωση του εκτελέσιμου προγράμματος.



Εικόνα 3.7 Διαδικασία ανάπτυξης προγράμματος [45]

Τα κύρια εργαλεία που χρησιμοποιεί ένας προγραμματιστής για να αναπτύξει μία εφαρμογή σε μία συγκεκριμένη γλώσσα προγραμματισμού υψηλού επιπέδου είναι:

Ενας **συντάκτης κειμένων (editor)** με τον οποίο και γράφει το αρχικό πρόγραμμα, που ονομάζεται **πηγαίο πρόγραμμα ή κώδικας (source code)**. Το αρχείο που περιέχει τον πηγαίο κώδικα στη C συνήθως έχει επέκταση *.c. Στη C++ δίνεται συνήθως η επέκταση *.cpp. **ένα μεταφραστικό πρόγραμμα** (μεταγλωττιστή ή διερμηνευτή), το οποίο μεταφράζει το πηγαίο πρόγραμμα σε **αντικείμενο πρόγραμμα ή κώδικα (object code)**.

Το μεταφραστικό πρόγραμμα ελέγχει το πηγαίο πρόγραμμα για **συντακτικά λάθη**, εμφανίζει κατάλληλα διαγνωστικά μηνύματα, εάν βρεθούν λάθη, και μόνο αν δεν υπάρχουν λάθη παράγεται το αντικείμενο πρόγραμμα. Το αντικείμενο πρόγραμμα είναι σε γλώσσα μηχανής, αλλά δεν είναι ακόμη εκτελέσιμο από τον υπολογιστή και πρέπει να περάσει από κάποιες άλλες διαδικασίες ένα ειδικό πρόγραμμα που ονομάζεται **συνδέτης (linker)**, το οποίο πολλές φορές συνδέει το αντικείμενο πρόγραμμα ή ένα σύνολο από αντικείμενα προγράμματα με έτοιμα υποπρογράμματα της βιβλιοθήκης της γλώσσας προγραμματισμού ή του προγραμματιστή. Το τελικό πρόγραμμα που παράγεται είναι το **εκτελέσιμο πρόγραμμα ή κώδικας (executable**

code), είναι διατυπωμένο σε γλώσσα μηχανής και μπορεί να εκτελεστεί άμεσα από τον επεξεργαστή του υπολογιστή.

Εργαλεία εντοπισμού λαθών (debuggers) με τα οποία ο προγραμματιστής παρακολουθεί τι ακριβώς συμβαίνει στο παρασκήνιο κατά την εκτέλεση ενός προγράμματος.Ένα περιβάλλον (λογισμικό) που περιλαμβάνει τα παραπάνω εργαλεία και χρησιμοποιείται για την ανάπτυξη εφαρμογών ονομάζεται προγραμματιστικό περιβάλλον ή περιβάλλον ανάπτυξης εφαρμογών [45].

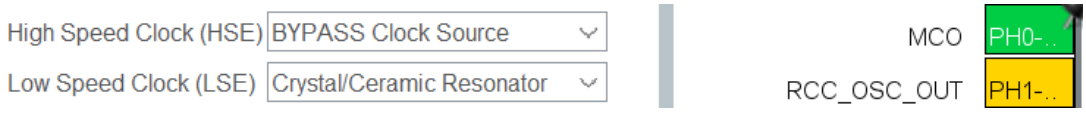
3.3 Λειτουργία συστήματος αλιείας

Ο μικροελεκτής χρησιμοποιεί πυρήνα ARM Cortex-M7 αρχιτεκτονικής 32 bit και μέγιστη λειτουργία ταχύτητας 480Mhz. Στήν παρούσα πτυχιακή χρησιμοποιήται λειτουργία ταχύτητας με 200Mhz.

Ο χρονισμός του συστήματος είναι η βασική αρχή στον μικροελεγκτή. Η πλακέτα βασίζεται σε εξωτερικό κρύσταλλο HSE, οπότε ενεργοποιήται το περιφερειακό (Reset and Clock Control) RCC. Καθώς ενεργοποιηθεί ο εξωτερικός ταλαντωτής, μπορούμε να καθορίσουμε την συχνότητα (μέσα στο μπλε πλαίσιο με την ένδειξη «Input Frequency») και για να διαμορφώσετε το κύριο PLL για την επιθυμητή ταχύτητα SYSCLK .

Διαφορετικά, η συχνότητα εισόδου εξωτερικού ταλαντωτή μπορεί να χρησιμοποιείται απευθείας ως ρολόι πηγής.

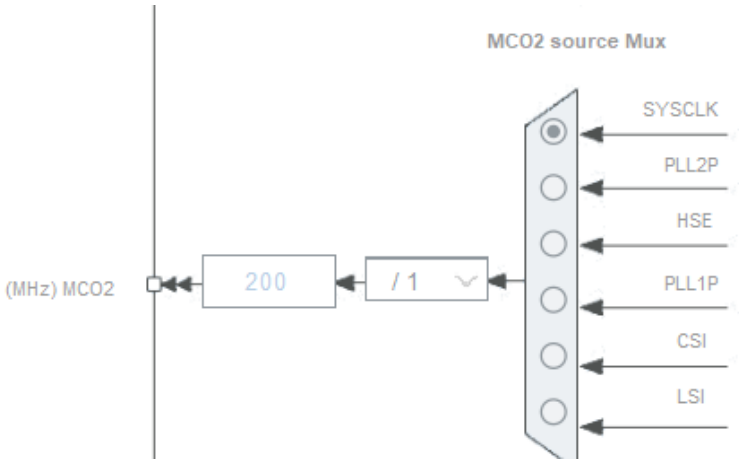
Η ρύθμιση BYPASS Clock Source χρησιμοποιείται ως μια εξωτερική πηγή ρολογιού. Η πηγή ρολογιού δημιουργείται από άλλη ενεργή συσκευή. Αυτό σημαίνει ότι το RCC_OSC_OUT μένει αχρησιμοποίητο και είναι δυνατό για να χρησιμοποιήθεί ως κανονικό GPIO.



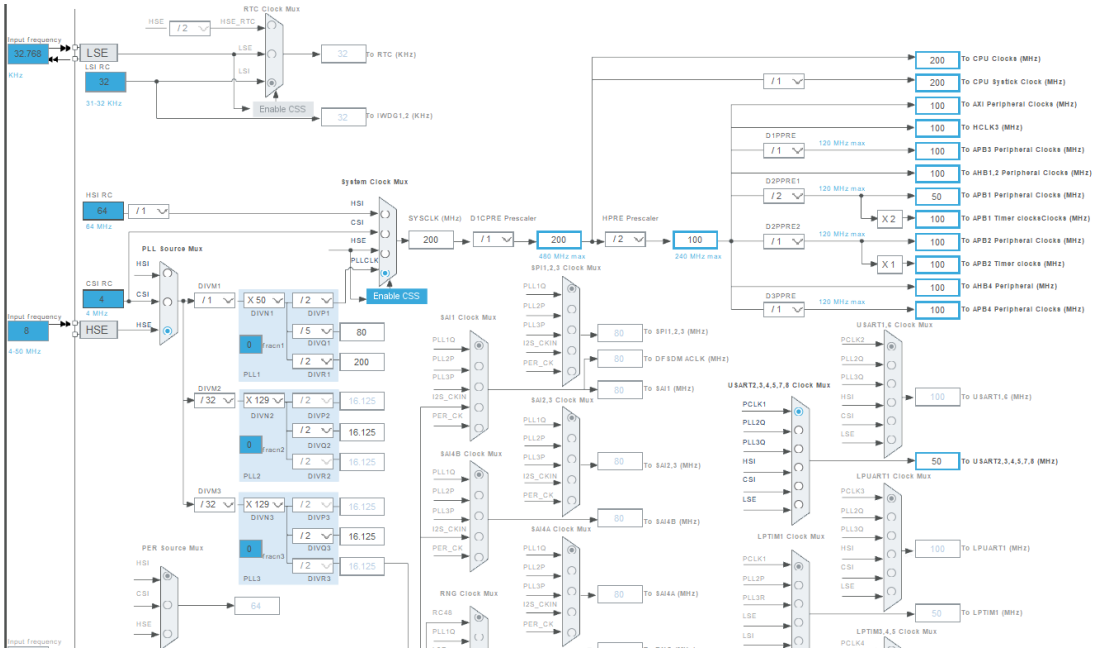
Εικόνα 3.8 Ενεργοποίηση HSE.

Το περιφερειακό RCC επιτρέπει επίσης την ενεργοποίηση της εξόδου Master Clock Output (MCO), η οποία είναι μια ακίδα που μπορεί να συνδεθεί σαν πηγή ρολογιού.

Ο ακροδέκτης εξόδου Master Clock Output (MCO) της διεπαφής ST-LINK χρησιμοποιείται ως εξωτερική πηγή ρολογιού για το STM32 MCU. Αυτή η συχνότητα δεν μπορεί να αλλάξει, είναι σταθερή στα 8 MHz. **Η ενεργοποίηση αυτής της επιλογής επιτρέπει τη χρήση του ST-LINK MCO ως HSE**. [Mastering STM32 Release 0.18 σελ.301-303] [46].



Εικόνα 3.9 Επιλογή ακίδας MCO για την πηγή ρολογιού



Εικόνα 3.10 Χρονισμός συστήματος

Η τροφοδοσία του συστήματος αλιείας παρέχεται εξωτερικά από την μπαταρία των +12V, η πλακέτα του αναπτυξιακού τροφοδοτείται από εξωτερική πηγή των +5V, η οποία πηγή τροφοδοτείται από την πηγή της μπαταρίας. Η λειτουργεία του μικροελεγκτή απαιτεί την τροφοδοσία των +3.3V η οποία παρέχεται από τον ενσωματωμένο σταθεροποιητή της αναπτυξιακής πλακέτας. Η πλακέτα διαθέτει επίσης ακροδέκτες, όπως δείχνουν οι παραπάνω εικόνες, οι οποίες συνδέονται στις πόρτες του μικροελεγκτή με σκοπό την σύνδεση εξωτερικών κυκλωμάτων με τα εσωτερικά περιφερειακά.

3.3.1 Λήψη και καταγραφή εισόδου

Με την ενεργοποίηση του παλμού, και αυτό γίνεται όταν η καρούλα περιστραφεί για τουλάχιστον κατά 90 μοίρες, οι μαγνήτες νεοδημίου που είναι τοποθετημένοι επάνω στην καρούλα, ενεργοποιούν με την σειρά τους, τους αισθητήρες, και έτσι έχουμε την λήψη του παλμού στην είσοδο του pin PA0 της αναπτυξιακής πλακέτας. Ο TIM2 εχει οριστεί σε Input Capture .

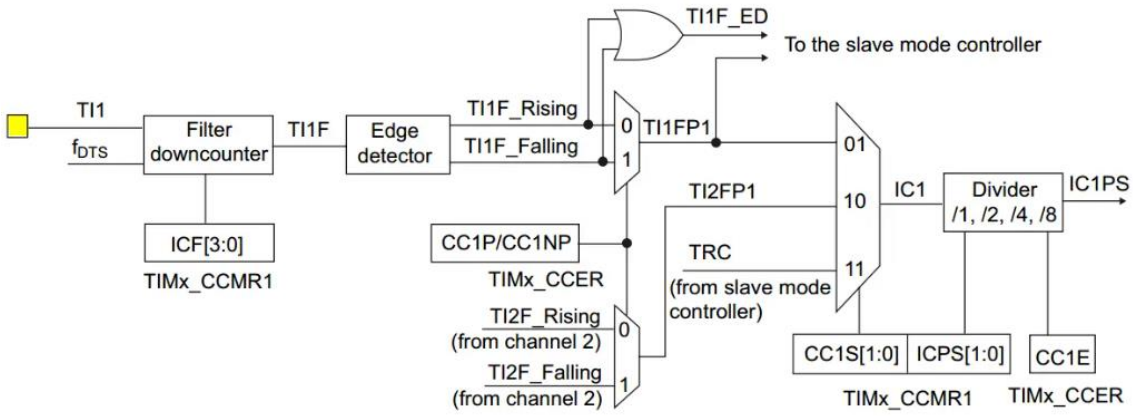
Ο TIM2 είναι συνδεδεμένος στο δίαυλο APB1(Advanced Peripheral Bus), το οποίο λειτουργεί στα 100 MHz, η τιμή του Prescaler είναι στο (100-1), και το ARR (AutoReload Register) στα 4294967295 .

Η ελάχιστη τιμή συχνότητας που θα μπορεί να διαβάσει ο TIM2 θα είναι:

TIM2= 1Mhz/4294967295 =2,328*E-4 Hz.

Ο TIM2 χρονίζεται από μια εσωτερική πηγή και η τρέχουσα τιμή του καταγράφεται και αποθηκεύεται στον καταχωρητή καταγραφής εισόδου κάθε φορά που συμβαίνει ένα συμβάν στον ακροδέκτη του καναλιού λήψης εισόδου.

Το στάδιο εισόδου για λήψη είναι με (ψηφιακό φίλτρο, πολυπλεξία και Prescaler) και το στάδιο εξόδου με(συγκριση και έλεγχο εξόδου). Το στάδιο εισόδου λαμβάνει δείγματα της αντίστοιχης εισόδου TIx για να δημιουργήσει ένα φιλτραρισμένο σήμα TIxF. Στη συνέχεια, ένας ανιχνευτής ακμών με επιλογή πολικότητας παράγει ένα σήμα (TIxFPx) το οποίο μπορεί να χρησιμοποιηθεί ως είσοδος ενεργοποίησης από τον ελεγκτή (Το the slave mode controller) ή ως εντολή λήψης. Καθώς έχει προκλιμακωθεί από τον (Prescaler) πριν από τον καταχωρητή καταγραφής (ICxPS)[47] .



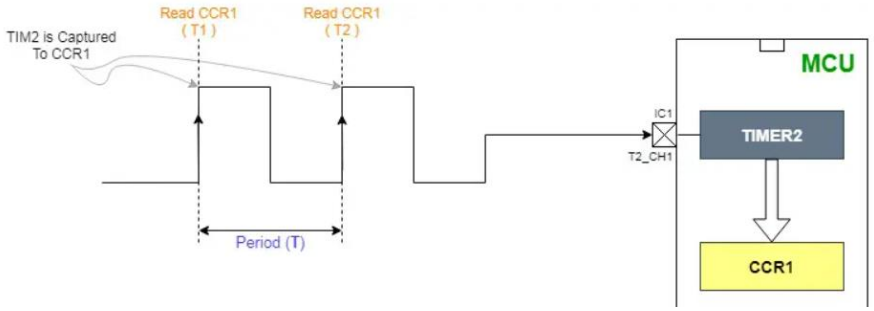
Εικόνα 3.11 Διάγραμμα λήψη/σύγκριση καναλιού [47]

Στη λειτουργία καταγραφής εισόδου, οι καταχωρητές Capture/Compare (TIMx_CCRx) χρησιμοποιούνται για να κλειδώσουν την τιμή του μετρητή μετά από μια μετάβαση που

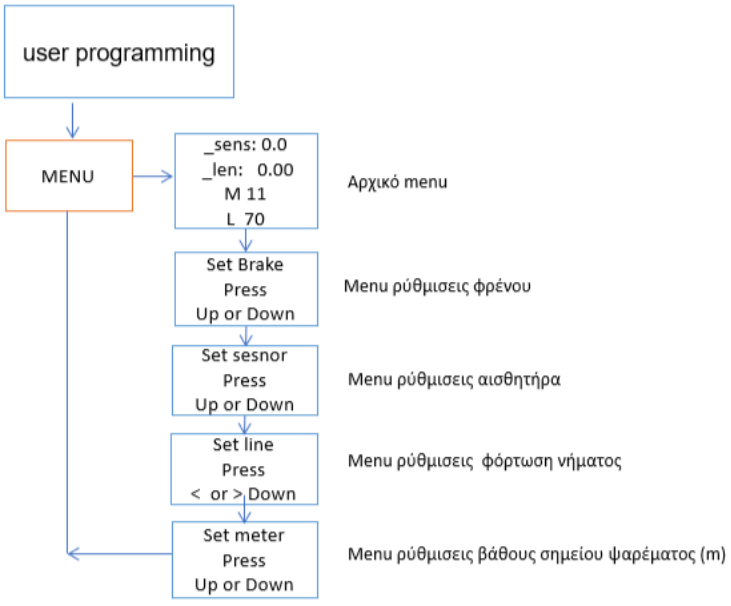
ανιχνεύεται από το αντίστοιχο σήμα ICx. Όταν συμβαίνει μια σύλληψη, ορίζεται η αντίστοιχη σημαία CCXIF (καταχωρητής TIMx_SR) και μπορεί να σταλεί μια διακοπή ή ένα αίτημα DMA εάν είναι ενεργοποιημένα. Εάν γίνει λήψη ενώ η σημαία CCxIF ήταν ήδη ψηλά, τότε έχει οριστεί η σημαία υπερβολικής λήψης CCxOF (καταχωρητής TIMx_SR). Το CCxIF μπορεί να διαγραφεί από το λογισμικό γράφοντάς το στο 0 ή διαβάζοντας τα δεδομένα που έχουν καταγραφεί που είναι αποθηκευμένα στον καταχωρητή TIMx_CCRx. Το CCxOF διαγράφεται όταν γράφεται στο 0.

Ο TIM2 χρησιμοποιεί το εσωτερικό ρολόι και διαμορφώνει το CH1 ως λήψη εισόδου σε κάθε ανερχόμενη άκρη.

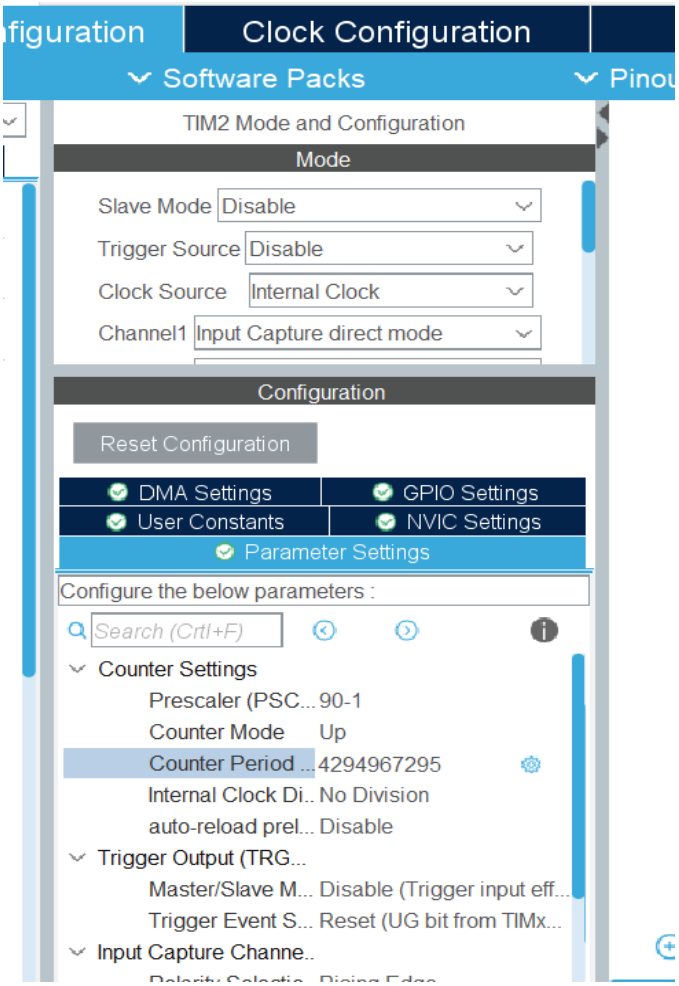
Διαβάζει τον καταχωρητή CCR1 και αποθηκεύει στη μεταβλητή T1 την πρώτη άκρη, στη δεύτερη άκρη διαβάζει πάλι το CCR1 και αποθηκεύει την τιμή στην μεταβλητή στο T2. Η περίοδος του σήματος εισόδου είναι T2 – T1.[47]



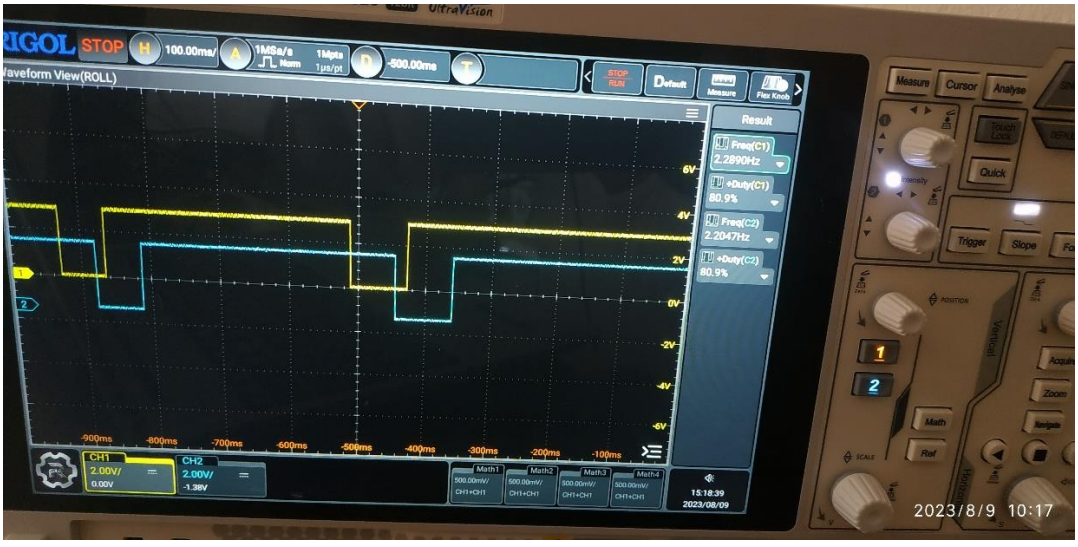
Εικόνα 3.12 Λήψη ακμών σε ανερχόμενο μέτωπο [47]



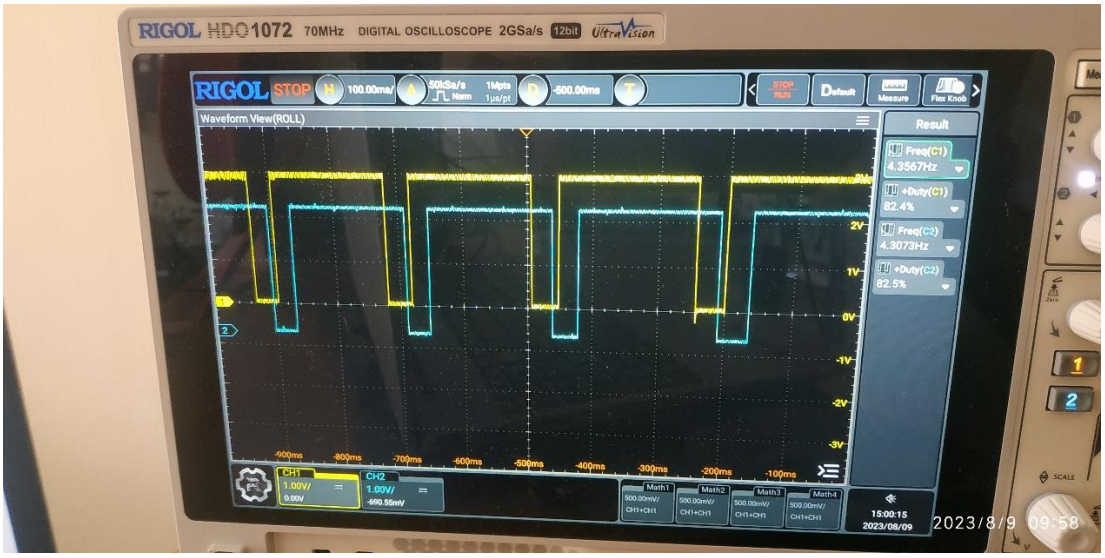
Εικόνα 3.13 Προγραμματισμός χρήστη



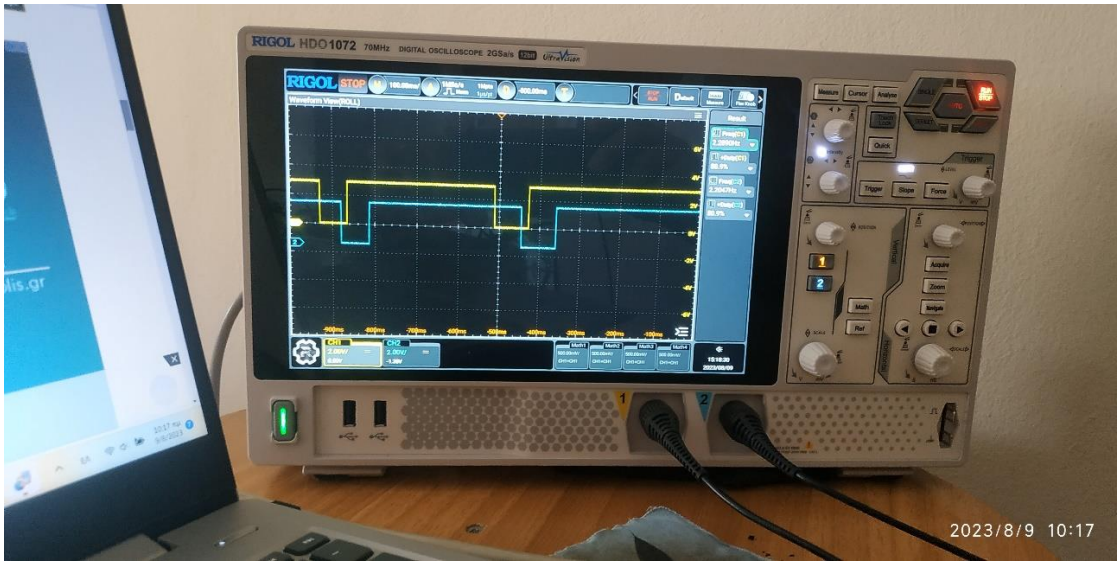
Εικόνα 3.14 Ρύθμιση TIM2 στο STM32CubeIDE



Εικόνα 3.15 Εμφάνιση λήψης παλμού των αισθητήρων από τον παλμογράφο



Εικόνα 3.16 Εμφάνιση λήψης παλμού των αισθητήρων από τον παλμογράφο



Εικόνα 3.17 Εμφάνιση λήψης παλμού των αισθητήρων από τον παλμογράφο

```
//-------------------------------------------------------------
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
{
    if(State == IDLE)
 {
     T1 = TIM2->CCR1;              // TIM2
     TIM2_OVC = 0;
     State = DONE;            //   EA@? ГI@?EI
 }
    else if(State == DONE)
{
     T2 = TIM2->CCR1;

      Ticks = (T2 + (TIM2_OVC * 4294967295)) - T1;
   Freq = (F_CLK/Ticks);    //  frequence



    if(Freq != 0)
{

     sec=(Ticks*0.133);    //  all time from overflow  (NA G
    // sec=(T_ms/1000);  //    time in sec


}
}
}

//-------------------------------------------------------------
```

Εικόνα 3.18 Λειτουργία του κώδικα λήψης παλμού

Παρουσιάστηκε και αναλύθηκε ένας μηχανισμός προγραμματιζόμενου σύστηματος αλιείας και υλοποιήθηκε με τον STM32h743 με συνδεδεμένα αισθητήρια για τον βασικό έλεγχο της λήψης του αλιεύματος. Ο κύριος στόχος της εργασίας έχει επιτευχθεί ώστε να υπάρχει σταθερή επικοινωνία πραγματικου χρόνου των αισθητήρων με την αναπτυξιακή πλακέτα. Έτσι ο χρήστης να μην χρειάζεται να έχει την συνεχή επαφή της αίσθησης του νήματος προς το αλίευμα.

Παρουσιάσθηκε εκτενέστερα ο κώδικας και ο παλμός για τη λήψη από τους αισθητήρες.

Ο χρήστης αναλαμβάνει αρχικώς τις βασικές ρυθμίσεις, δηλαδή να φορτώση την ποσότητα του νήματος στην καρούλα, να ρυθμίση τον αισθητήρα και το φρένο, και να ορίσει σε πόσα μέτρα θέλει να αλιεύσει.

Η κατανάλωση του συστήματος χωρίς βασικά φορτία, του ηλεκτρομαγνήτη και του κινητήρα είναι 390 mA.

Επίσης το κόστος ολόκληρης της κατασκευής ανέρχεται σε 430 ευρώ.

Η πιο σημαντική πρόταση βελτίωσης αφορά την ασφάλεια του συστήματος.

Δεν έχει τοποθετηθεί ασφάλεια, σε περίπτώσεις που εμφανιστεί κάποιο πρόβλημα την ώρα που το νήμα βρίσκεται στον βυθό.

Επίσης, υπάρχει συντονισμός στον ηλεκτρομαγνήτη που δημιουργείται από την συχνότητα των 3Khz την οποία λειτουργεί, και ακούγεται ένα χαμηλό σφήργημα.

Ακόμα επίσης η ταχύτητα του κινητήρα 250w δεν είναι στη μέγιστη τιμή του, η πλακέτα οδήγησεις είναι χαμηλής ισχύος (κινέζικη με κόστος περίπου 8Ε) και βεβαίως χαμηλής πιστότητας.

Η ακρίβεια των αισθητήρων ως προς την μέτρηση του μήκους του νήματος δεν ενέχει ανοχές, δημιουργούνται όμως ανοχές διότι υπάρχούν πολλά και διαφορετικά είδη ψαρέματος τα οποία θέτουν αλλαγή στο πάχος του νήματος, με αποτέλεσμα να αυξάνει η να μειώνεται το μήκος του νήματος.

Για την αποφηγεί τέτοιων καταστάσεων θα μπορούσε να τοποθετηθούν οι αισθητήρες σε διαφορετικό σημείο της μηχανής και να μετράνε το μήκος ενός μικρού κύκλου και όχι την ποσότητα του νήματος που βρίσκεται επάνω στην καρούλα.

Η ανάλυση ως προς την μικρότερη αύξηση που μπορεί να ανιχνεύσει ο αισθητήρας το μήκος του κύκλου είναι 14,175cm , καθώς ολόκληρος ο κύκλος είναι 56,7cm.

Ακόμη η όλη κατασκευή χρίζει νέο κιβώτιο και νέα κατασκευή πλακέτας με ενσωματωμένες όλες τις μονάδες.

Τέλος μπορούν να γραφούν διαφορετικά προγράμματα για διάφορετικά είδη αλιείας, το παρόν πρόγραμμα είναι γραμμένο για απλή βασική χρήση (καθετής), καθώς με κάποια παρελκόμενα μπορεί να χειριστεί ψάρεμα συρτής και τσαπαρί.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Web site

[1] https://agrogi.eu/index.php?option=com_content&view=article&id=47:%CE%B1%CE%BB%CE%B9%CE%B5%CE%AF%CE%B1&catid=20&Itemid=138

[2] https://fish.shimano.com/en-GB/product/reels/electric/a075f00002vmhavqaq.html

[3]https://www.hobbyfishing.gr/product/free-snap-hood-2/

[4] https://w97151.shop.textalk.se/en/ex/jigging-machine-bj5000ex.html

[5] https://ifishing.gr/shop/eidi-alieias/kalami-shimano-technium-xf78l/

[6] https://www.roycecrossgroup.com/EC250120/Transtecno-12v-DC-Motor-250W%2C-3000RPM%2C-D63-B14A-Flange%2C-11mm-Shaft%2C-IP44/pd.php

[7] http://eclass.opencourses.teicm.gr/eclass/modules/document/file.php/TMA112/mix_hm_05_DC_MOTORS.pdf

[8] https://in.pinterest.com/pin/dc-motor-or-direct-current-motor-what-is-it-diagram-included--762304674435817622/

[9] https://docplayer.gr/56248408-I-ergasia-ayti-afieronetai-ston-horigo-moy-zagora-foteino-gia-tin-ypostirixi-kai-tin-ypomoni-toy-kata-ti-diarkeia-ton-spoydon-moy.html

[10] https://www.boznos.gr/etaireia/nea/item/42-meiotiras-ti-einai

[11] https://el.wikipedia.org/wiki/%CE%A0%CE%BB%CE%B7%CE%BA%CF%84%CF%81%CE%BF%CE%BB%CF%8C%CE%B3%CE%B9%CE%BF

[12] https://solderingmind.com/150w-dc-dc-boost-converter-schematic/

[13] https://www.tziola.gr/book/ilektronika-ischyos-2i-ekdosi/ .[Βιβλίο θεωρίας Ηλεκτρονικά ισχύος Ιορδάνης Κιοσκερίδης σελ.28-30]

[14] https://www.tziola.gr/book/ilektronika-ischyos-2i-ekdosi/ [Βιβλίο θεωρίας Ηλεκτρονικά ισχύος Ιορδάνης Κιοσκερίδης σελ.456]

[15] https://solderingmind.com/wp-content/uploads/2023/06/150w-dc-dc-boost-converter-schematics.jpg

[16] https://www.infineon.com/cms/en/product/sensor/magnetic-sensors/magnetic-position-sensors/magnetic-switches/tle4905l/

[17] https://eclass.emt.ihu.gr/modules/document/file.php/ED153/2.%20%CE%94%CE%99%CE%91%CE%9B%CE%95%CE%9E%CE%95%CE%99%CE%A3%20%CE%9C%CE%91%CE%98%CE%97%CE%9C%CE%91%CE%A4%CE%9F%CE%A3/%CE%98%CE%95%CE%A9%CE%A1%CE%97%CE%A4%CE%99%CE%9A%CE%9F%20%CE%9C%CE%95%CE%A1%CE%9F%CE%A3/1_%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE%20%CF%83%CF%84%CE%BF%

CF%85%CF%82%20%CE%91%CE%B9%CF%83%CE%B8%CE%B7%CF%84%CE%AE%CF%81%CE%B5%CF%82.pdf
[18] **https://www.mouser.com/datasheet/2/196/Data_Sheet_TLE49x5L_Family_V1_5-57396.pdf**
[19] **https://eu.mouser.com/ProductDetail/Newhaven-Display/NHD-0420E2Z-NSW-BBW?qs=3vk7fz9CmNwAEidaKnp9zQ%3D%3D**
[20] https://www.porlidas.gr/AVR/LCD2x16Gr.htm

[21] https://www.tme.eu/Document/8aa8ed83dc2f8e8a5f3ad4a71fd201bc/lm7805.pdf

[22] https://en.m.wikipedia.org/wiki/File:LM7805_with_Decoupling_Capacitor.svg

[23] https://www.electronicsforu.com/technology-trends/learn-electronics/7805-ic-voltage-regulator

[24] https://en.wikipedia.org/wiki/Galvanic_isolation

[25] https://global.sharp/products/device/lineup/data/pdf/datasheet/PC817XxNSZ1B_e.pdf
[26] https://www.infineon.com/dgdl/irlz44npbf.pdf?fileId=5546d462533600a40153567217c3272
[27] https://el.wikipedia.org/wiki/USB

[28] https://www.skroutz.gr/s/29652162/Mpouton-diplo-F22-START-STOP-NO-NC-4A-B102K20KY-EMAS.html

[29] https://badu.gr/%CE%BA%CF%8D%CF%81%CE%B9%CE%BF%CF%82-%CE%B4%CE%B9%CE%B1%CE%BA%CF%8C%CF%80%CF%84%CE%B7%CF%82-%CE%BC%CF%80%CE%B1%CF%84%CE%B1%CF%81%CE%AF%CE%B1%CF%82-%CE%B1%CF%85%CF%84%CE%BF%CE%BA%CE%B9%CE%BD%CE%AE%CF%84%CE%BF%CF%85-%CE%B1%CF%80%CE%BF%CF%83%CF%8D%CE%BD%CE%B4%CE%B5%CF%83%CE%B7-%CF%80%CE%B5%CF%81%CE%B9%CF%83%CF%84%CF%81%CE%BF%CF%86%CE%B9%CE%BA%CF%8C%CF%82-%CE%B4%CE%B9%CE%B1%CE%BA%CF%8C%CF%80%CF%84%CE%B7%CF%82-%CF%86%CE%BF%CF%81%CF%84%CE%B7%CE%B3%CF%8C-%CE%B8%CE%B1%CE%BB%CE%AC%CF%83%CF%83%CE%B9%CE%BF%CF%85-%CF%83%CE%BA%CE%AC%CF%86%CE%BF%CF%85%CF%82-cut-off-isolator-kill-switch-300a-p-674257.html

[30] https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%BA%CF%8C%CF%80%CF%84%CE%B7%CF%82
[31] https://www.ledison.gr/gr/aca-stegano-kouti-diakladosis-310x240x120-me-lia-epifania-ip65.html?gclid=EAIaIQobChMI57fekvabggMVj-F3Ch01uw43EAsYAiABEgIZN_D_BwE
[32] https://www.kmtools.gr/shop/ergaleia-exartemata/diaphora-3/rodes-gia-karotsia/1948.html?quantity=4

[33] http://www.eureka.teithe.gr/jspui/bitstream/123456789/13752/1/%CE%91%CE%93%CE%93%CE%95%CE%9B%CE%91%CE%9A%CE%9F%CE%A3%20%CE%93%CE%95%CE%A9%CE%A1%CE%93%CE%99%CE%9F%CE%A3.pdf

[34] https://depositphotos.com/gr/similar-vectors/336725254.html

35] https://technolysis-hts.gr/%CE%B2%CE%B1%CF%83%CE%B7-%CE%B3%CE%BD%CF%89%CF%83%CE%B5%CF%89%CE%BD/%CE%B1%CF%85%CF%84%CF%8C%CE%BD%CE%BF%CE%BC%CE%B1-%CF%86%CF%89%CF%84%CE%BF%CE%B2%CE%BF%CE%BB%CF%84%CE%B1%CF%8A%CE%BA%CE%AC/

[36] https://www.mybotic.com.my/bts7960-motor-driver-module-43a

[37] https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm

[38] http://ebooks.edu.gr/ebooks/handle/8547/3864 ΔΟΜΗ & ΛΕΙΤΟΥΡΓΙΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ 2ου ΚΥΚΛΟΥ σελ.71

[39] http://teachers.cm.ihu.gr/kalomiros/Mtptx/e-books/Embedded_PIC_new.pdf

[40] https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m7.html

[41] https://www.st.com/resource/en/user_manual/um2407-stm32h7-nucleo144-boards-mb1364-stmicroelectronics.pdf

[42] https://www.st.com/resource/en/user_manual/um2407-stm32h7-nucleo144-boards-mb1364-stmicroelectronics.pdf

[43] https://el.wikipedia.org/wiki/%CE%91%CF%80%CE%BF%CF%83%CF%86%CE%B1%CE%BB%CE%BC%CE%AC%CF%84%CF%89%CF%83%CE%B7

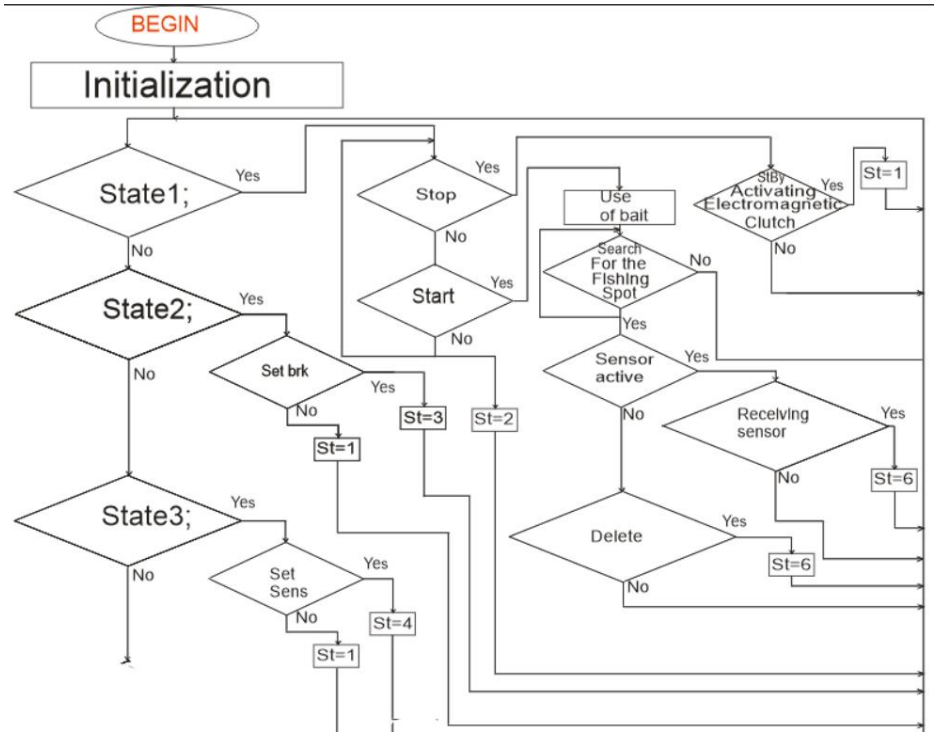
[44] https://www.st.com/resource/en/user_manual/um2407-stm32h7-nucleo144-boards-mb1364-stmicroelectronics.pdf

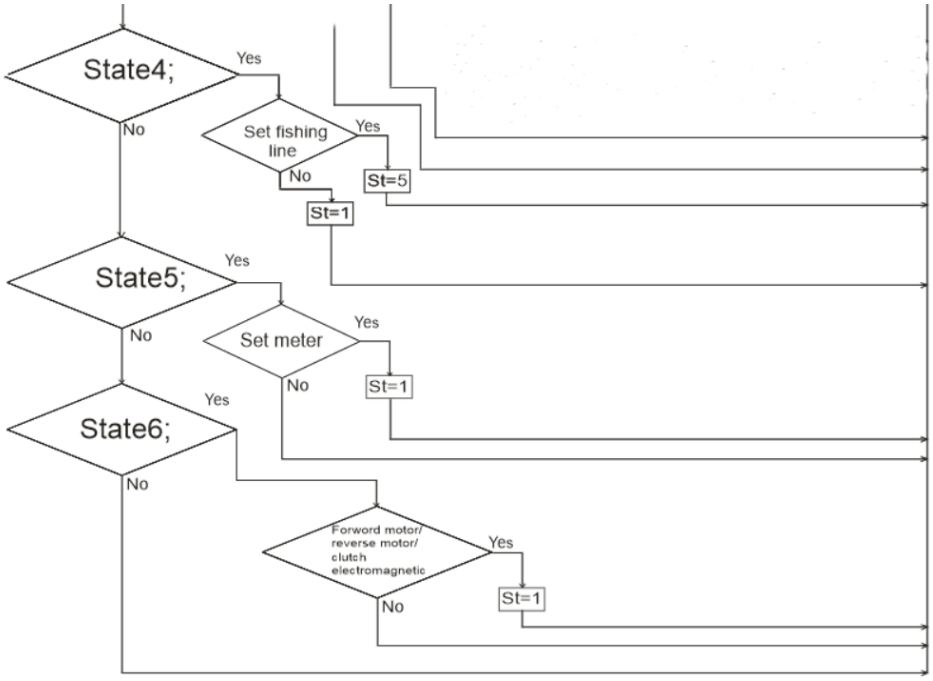
[45] http://ebooks.edu.gr/ebooks/v/html/8547/2714/Pliroforiki_A-Lykeiou_html-empl/index2_6.html

[46] https://leanpub.com/mastering-stm32 [Mastering STM32 Release 0.18 σελ.301-303]

[47] https://deepbluembedded.com/stm32-input-capture-frequency-measurement-example-timer-input-capture-mode/#google_vignette

Παρουσίαση σχηματικού διαγράμματος ροής του κώδικα





ΠΑΡΑΡΤΗΜΑ Β

Παρουσίαση διαγράμματος ροής του κώδικα

main.c

```
/* USER CODE END Header */
/* Includes ------------------------------------------
#include "main.h"
#include "string.h"

/* Private includes ----------------------------------
/* USER CODE BEGIN Includes */
//#include "i2c-lcd.h"
#include "tm_stm32f4_hd44780.h"
#include "string.h"
#include "stm32h7xx_hal.h"
#include "stdio.h"
#include "FLASH_SECTOR_H7.h"
#include "keypad.h"
#include <stdlib.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------
/* USER CODE BEGIN PTD */
#define KEYPAD_NO_PRESSED          0xFF
#define IDLE   0
#define DONE   1
#define F_CLK 100000000UL   // from 200.....UL

#define ROW_1_Pin GPIO_PIN_3
#define ROW_1_GPIO_Port GPIOA
#define COL_1_GPIO_Port GPIOF  //  define  port F
#define COL_1_Pin GPIO_PIN_5   //  define  number pin 5
#define COL_2_Pin GPIO_PIN_10
#define COL_2_GPIO_Port GPIOF
#define COL_3_Pin GPIO_PIN_12
#define COL_3_GPIO_Port GPIOF
#define COL_4_Pin GPIO_PIN_15
#define COL_4_GPIO_Port GPIOD
#define COL_5_Pin GPIO_PIN_3
#define COL_5_GPIO_Port GPIOF
#define COL_6_Pin GPIO_PIN_4
#define COL_6_GPIO_Port GPIOF
#define COL_7_Pin GPIO_PIN_0
#define COL_7_GPIO_Port GPIOC
```

main.c

```
64 #define IN1_D1_PIN              GPIO_PIN_1
65 #endif
66 #ifndef IN2_D2_PIN
67 #define IN2_D2_PORT             GPIOD
68 #define IN2_D2_PIN              GPIO_PIN_2
69 #endif
70 #ifndef IN3_D3_PIN
71 #define IN3_D3_PORT             GPIOD
72 #define IN3_D3_PIN              GPIO_PIN_3
73 #endif
74 #ifndef IN4_D4_PIN
75 #define IN4_D4_PORT             GPIOD
76 #define IN4_D4_PIN              GPIO_PIN_4
77 #endif
78
79 char c;
80 char buf18;
81 char buf  [20];
82 char buf0 [20];
83 char buf1 [30];
84 char buf2 [20];
85 char buf3 [20];
86 char buf4 [20];
87 char buf5 [20];
88 char buf6 [20];
89 char str2 [20];
90 char arr0  [20]={"*"};
91 char arr1  [20]={"Stb"};
92 char arr13 [20]={"                "};
93 char arr14 [20]={"    "};
94
95
96 char arr2 [15] [20]={
97     "_ Set line",
98     "_ press ",
99     "_ <  or  >"
100
101 };
102 char arr2_1 [15] [20]={
103     "Load",
104       "m"
```

main.c

```
112
113 char arr3 [15] [20]={
114     "_ Set sensor ",
115     "_ Press",
116     "_ Up orDown"
117 };
118 char arr3_1 [15] [20]={
119      "_ Sensor up"
120 };
121 char arr3_2 [15] [20]={
122
123      "_ Sensor down"
124
125 };
126 char arr4 [15] [20]={
127    "_ Set Brake",
128    "_ Press",
129    "_ Up orDown"
130 };
131 char arr4_1 [15] [20]={
132     "_ Brake up"
133
134 };
135 char arr4_2 [15] [20]={
136     "_ Brake down"
137
138
139 };
140 char arr5 [15] [20]={
141    "_ P1",
142    "Press START"
143 };
144
145 char arr6 [15] [20]={
146     "_ Batt"
147 };
148
149 char arr8 [15] [20]={
150     "_ Insert Thread",
151     "_ Press Menu"
152
153
154 };
```

main.c

```
154 };
155 char arr9 [15] [20]={
156     "_ Meter up "
157 };
158 char arr9_1 [15] [20]={
159     "_ Meter down "
160 };
161 char arr9_2 [15] [20]={
162     "_ Set meter",
163     "_ Press",
164     "_ Up orDown"
165 };
166
167 char arr11 [15] [20]={
168      "_ SetP2",
169      "_ Press",
170      "_ Up orDown"
171 };
172 char arr12 [15] [20]={
173      "_ delete"
174
175 };
176 char arr15 [15] [20]={
177      "_ PressMenu"
178
179 };
180
181 char arr16 [15] [20]={
182      "_ Please Wait"
183
184 };
185 char arr17 [16] [20]={
186      "_ Thread is over",
187      "_ Press delete"
188
189 };
190 ETH_DMADescTypeDef DMATxDscrTab[ETH_TX_DESC_CNT] __attribute__((section(".TxDecripSection")));
191
```

main.c

```
196
197 #define FLASH_USER_START_ADDR   ADDR_FLASH_SECTOR_7_BANK2      /* Start @ of user Flash area Bank2
198 #define FLASH_USER_END_ADDR     (ADDR_FLASH_SECTOR_7_BANK2 - 1)  /* End @ of user Flash area Bank2
199
200 /* USER CODE END PD */
201
202 /* Private macro -------------------------------------------------------------*/
203 /* USER CODE BEGIN PM */
204
205 /* USER CODE END PM */
206
207 /* Private variables ---------------------------------------------------------*/
208 #if defined ( __ICCARM__ ) /*!< IAR Compiler */
209
210 #pragma location=0x30040000
211 ETH_DMADescTypeDef  DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet Rx DMA Descriptors */
212 #pragma location=0x30040060
213 ETH_DMADescTypeDef  DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet Tx DMA Descriptors */
214 #pragma location=0x30040200
215 uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE]; /* Ethernet Receive Buffers */
216
217 #elif defined ( __CC_ARM )  /* MDK ARM Compiler */
218
219 __attribute__((at(0x30040000))) ETH_DMADescTypeDef  DMARxDscrTab[ETH_RX_DESC_CNT]; /* Ethernet Rx
220 __attribute__((at(0x30040060))) ETH_DMADescTypeDef  DMATxDscrTab[ETH_TX_DESC_CNT]; /* Ethernet Tx
221 __attribute__((at(0x30040200))) uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE]; /* Ethernet
222
223 #elif defined ( __GNUC__ ) /* GNU Compiler */
224
225 ETH_DMADescTypeDef DMARxDscrTab[ETH_RX_DESC_CNT] __attribute__((section(".RxDecripSection"))); /*
226 ETH_DMADescTypeDef DMATxDscrTab[ETH_TX_DESC_CNT] __attribute__((section(".TxDecripSection")));   /
227 uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_MAX_PACKET_SIZE] __attribute__((section(".RxArraySection")));
228
229 #endif
230
231 ETH_TxPacketConfig TxConfig;
232
233 ETH_HandleTypeDef heth;
234
235 TIM_HandleTypeDef htim2;
236 TIM_HandleTypeDef htim4;
237 TIM_HandleTypeDef htim8;
238 TIM_HandleTypeDef htim13;
```

main.c

```
238 TIM_HandleTypeDef htim13;
239 TIM_HandleTypeDef htim15;
240
241 UART_HandleTypeDef huart3;
242
243 /* USER CODE BEGIN PV */
244 //uint32_t flash_Address=0x081E0000;
245
246 uint32_t  max=0;
247 uint32_t  min=0;
248 uint8_t State = IDLE;
249 uint32_t T1 = 0;
250 uint32_t T2 = 0;
251 uint32_t Ticks = 0;
252 uint16_t TIM2_OVC=0;
253 uint16_t TIM4_OVC=0;
254 uint16_t TIM13_OVC=0;
255 uint32_t Freq = 0;
256 uint32_t Freq2=0;
257 uint32_t Down=0;
258
259 float  sensor=0;
260 float  RxVal;
261 float  RxVal1;
262 //float RxVal2=0;
263 float  flag19=0;
264 float  a1=0;
265 float  du=0;
266 float  dt=0;
267 float  u3=0;
268 float  u1=0;
269 float  u2=0;
270 float  u=0;
271 float  t=0;
272 float  m=0;
273 float  v=0;
274 float  vKgr=0;
275 float  P=0;
276 float  g=9.81;
277 float TIME;
278 float  sec=0;
279 float  cycle1=56.00;
280 float  cycle2=58.09;
```

main.c

```
280 float  cycle2=58.09;
281 float  cycle3=64.37;
282 float  point1=9.158333333;
283 float  point2=14.00;
284 float  threadLoading=0;
285 float  right=0;
286 float  left=0;
287 float  SensC=0;
288 float  zero;
289 float  countUp=0;
290 float  countDown=0;
291 float  Readflash;
292 float  value=0;
293 float  sum1;
294 float  sum2;
295 float  Time=0;
296 float  TurnOn=0;
297 volatile float  F=0;
298 volatile float  B=0;
299 volatile int loadLR;
300 volatile float leng1=0;
301 volatile float leng2=0;
302 volatile float leng3=0;
303 volatile float count=0;
304 volatile float count1=0;
305 volatile float count2=0;
306 volatile float count3=0;
307 volatile float Up=0;
308
309 int scan=0;
310 int flag00=0;
311 int  tsapari=0;
312 int Bazzer=0;
313 int Enable_Kpd_On=1;
314 int Unenable_Kpd_Off=0;
315 int Ctrl_Enter=1;
316 int pwm1;
317 int pwm2;
318 int PE0;
319 int PE1;
320 int PF1;
321 int IN1=0;
322 int IN2=0;
```

main.c

```
322 int IN2=0;
323 int ENA=0;
324 int ENB=0;
325 int enc_value;
326 int StartL=0;
327 int TestS=0;
328 int buzzer=0;
329 int sum3;
330 int sum4;
331 int Safe=0;
332 int SafeB2=0;
333 int SafeB3=0;
334 int SafeB4=0;
335 int SafeB5=0;
336 int SafeB6=0;
337 int SafeB7=0;
338 int SafeB8=0;
339 int SafeB9=0;
340 int SafeB10=0;
341 int SafeB11=0;
342 int SafeB12=0;
343 int SafeB13=0;
344 int SafeB14=0;
345 int SafeB15=0;
346 int SafeB16=0;
347 int flag50=0;
348 int flag51=0;
349 int menu=1;
350 int newload=0;
351 int prevload=0;
352 int menu1=0;
353 int menu2=0;
354 int menu3=0;
355 int menu4=0;
356 int menu5=0;
357 int menu6=0;
358 int menu7=0;
359 int menu8=0;
360 int sens=0;
361 int brk=0;
362 int stop=0;
363 int loadR=0;
364 int loadL=0;
```

main.c

```
int loadL=0;
int i=0;
int metritis=0;
int flag100=0;
int flag200=0;
int flag201=0;
int flag202=1;
int flag203=0;
int flag204=0;
int flag205=0;
int S1=0;
int S2=0;
int pwmD=0;
int pwmD1=0;
int pwmM0=0;
int pwmM=1;
int pwmM1=0;
int pwmM5=0;
int pwmM10=0;
int pwmM20=0;
int pwmM25=0;
int pwmM50=0;
int pwmM100=0;
int pwmD100=0;
int pwmD0=0;
int pwmD5=0;
int pwmD10=0;
int pwmD20=0;
int pwmD25=0;
int pwmD50=0;
int pwm1;
int PWM=0;
int PWM2=0;
int PWM3=0;

float test1=0;
float  x=0;
float  y=0;
float rate=0;
float spd=0;
float spd1=0;
float spd2=0;
float spd3=0;
```

main.c

```
406 float spd3=0;
407 float spd4=0;
408 float spd5=0;
409 float spd6=0;
410 float spd7=0;
411 float spd8=0;
412 float spd9=0;
413 float spd10=0;
414 float spd11=0;
415 float NetM=0;
416 float R1=100;
417 float R2=90;
418 float sum1=0;
419 float sum2=0;
420 float sum0=0;
421 int f1=0;
422 int f2=0;
423 int f3=0;
424 int f4=0;
425 int f5=0;
426
427 int flag01=0;
428 int lcd_flag=0;
429 int val=0;
430 int speedM=0;
431 int speedD=0;
432
433
434 int pwm2stp=0;
435 int flag=0;
436 int flag0;
437 int flag1=0;
438 int flag2=0;
439 int flag3=0;
440 int flag4=0;
441 int flag5=0;
442 int flag6=0;
443 int flag7=0;
444 int flag8=0;
445 int flag9=0;
446 int flag10=0;
447 int flag11=0;
448 int flag12=0;
```

main.c

```
448 int flag12=0;
449 int flag13=0;
450 int flag17=0;
451 int flag18=0;
452 int flag20=0;
453 int flag21=0;
454 int flag23=0;
455 int flag24=0;
456 int flag25=0;
457 int flag26=0;
458 int flag27=1;
459 int flag28=0;
460 int flag30=0;
461 int flag31=0;
462 int flag32=0;
463 int flag33=0;
464 int flag34=0;
465 int flag35=0;
466 int flag40=0;
467 int flag41=0;
468 int flag42=0;
469 int  clear=0;
470 int write1=0;
471 int M=0;
472
473 int flag001=0;
474 int CLR;
475 int memory1=0;
476 int memory2=0;
477
478 int MtrForward=0;
479 int MtrBuck=0;
480 int MtrStop=0;
481 int Magnetic=0;
482
483 int meter=0;
484 int LOAD=0;
485 int UNLOAD=0;
486 int P1=0;
487 int Start=0;
488 int StBy=0;
489 int empty=0;
490 int full=0;
```

main.c

```
490 int full=0;
491 int delete;
492 int b;
493 int mess1=0;
494 int flag1000=0;
495 int flag101=0;
496 int flag102=0;
497 int sos=0;
498 int reset=0;
499
500 double  T_ms=0;
501
502 /* USER CODE END PV */
503
504 /* Private function prototypes -----------------
505 void SystemClock_Config(void);
506 static void MX_GPIO_Init(void);
507 static void MX_ETH_Init(void);
508 static void MX_USART3_UART_Init(void);
509 static void MX_USB_OTG_FS_USB_Init(void);
510 static void MX_TIM4_Init(void);
511 static void MX_TIM2_Init(void);
512 static void MX_TIM8_Init(void);
513 static void MX_TIM13_Init(void);
514 static void MX_TIM15_Init(void);
515 /* USER CODE BEGIN PFP */
516
517 /* USER CODE END PFP */
518
519 /* Private user code ---------------------------
520 /* USER CODE BEGIN 0 */
521
522 static uint16_t newCount;
523 static uint16_t prevCount;
524
525
526  //char string[100];
527
528 // float RxVal;
529
530
531 //-----------------------------------------------
532 /*
```

main.c

```
/*
void delay(uint16_t delay)
{
__HAL_TIM_SET_COUNTER(&htim15,0);
while(__HAL_TIM_GET_COUNTER(&htim15) < delay);
}
*/
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* Enable I-Cache---------------------------------
  SCB_EnableICache();

  /* Enable D-Cache---------------------------------
  SCB_EnableDCache();

  /* MCU Configuration--------------------------------

  /* Reset of all peripherals, Initializes the Flash
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
```

*main.c

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ETH_Init();
MX_USART3_UART_Init();
MX_USB_OTG_FS_USB_Init();
MX_TIM4_Init();
MX_TIM2_Init();
MX_TIM8_Init();
MX_TIM13_Init();
MX_TIM15_Init();
/* USER CODE BEGIN 2 */


    HAL_TIM_Base_Start_IT(&htim2);
    HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
    HAL_TIM_Base_Start(&htim15);
    HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_1);
    HAL_TIM_Base_Start_IT(&htim8);
    HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
    HAL_TIM_Encoder_Start_IT(&htim4,TIM_CHANNEL_ALL);
    HAL_TIM_PWM_Start(&htim13, TIM_CHANNEL_1);
  // TM_HD44780_Init(0,0);


    TM_HD44780_CLEAR();
    TM_HD44780_Init(20,4);
    sprintf(buf, "by Grigoriadis S");
     TM_HD44780_Puts (3,2,buf);
     HAL_Delay(1500);
     TM_HD44780_CLEAR();
    sprintf(buf, "fishing machine %c");
    TM_HD44780_Puts (3,4,buf);
    sprintf(buf1, "GR ver.0.1 light %c");
    TM_HD44780_Puts (3,2,buf1);
    HAL_Delay(500);
//---------------------------------------

//-----------------------------------------------
    flag18=1;
  //  flag23=1;
    flag25=1;
    flag28=1;
    htim13.Instance->CCR1=150;
```

main.c

```
616       htim13.Instance->CCR1=150;
617       HAL_Delay(1500);
618       htim13.Instance->CCR1=50;
619     /* USER CODE END 2 */
620
621     /* Infinite loop */
622     /* USER CODE BEGIN WHILE */
623   //-------------------------------------------------------------
624
625   //-------------------------------------------------------------
626       a1= 1.0f;
627
628       while (1)
629       {
630
631                               if(flag19==0)
632                         {
633
634                               if(menu==1)
635                          {
636                   //-----------------------------------------------
637                               if(flag25==1)     //  VIEW CLEAR
638                           {
639
640                             TM_HD44780_CLEAR();
641                             flag25=0;
642                           }
643
644                   //-----------------------------------------------
645                            sprintf(buf2, "_sens:%3.1f",sensor);
646                            TM_HD44780_Puts (0,4,buf2);
647
648
649                            sprintf(buf4, "_batt:%3.2f ");
650                            TM_HD44780_Puts (0,3,buf4);
651                            sprintf(buf3, "_len :%3.2f", leng1 );
652                            TM_HD44780_Puts (0,1,buf3);
653                            sprintf(buf5,"L%2.1f",RxVal);
654                            TM_HD44780_Puts (14,3,buf5);
655                            sprintf(buf6,"M%1.2d",M);
656                            TM_HD44780_Puts (14,2,buf6);
657
658                         }
```

main.c

```
658                         }
659                       }
660                             if(mess1==1)   //   run leng1
661                            {
662
663                                TM_HD44780_Puts (3,1,(char*)(arr16 ));
664                                sprintf(buf18, "len :%3.2f", leng1 );
665                                TM_HD44780_Puts (2,3,buf18);
666
667                            }
668
669  //-------------------------------------------------
670
671                                if(RxVal>0)
672                             {
673                                sum1=RxVal-leng1;
674
675
676
677                                if(sum1<=2)
678                                {
679                                  htim13.Instance->CCR1=240;
680
681                                }
682                              }
683
684 //-------------------------------------------------------------------
685                                if(RxVal>loadLR)
686                                {
687                                 spd1=RxVal-loadLR;
688                                }
689                                else if(RxVal<loadLR)
690                                {
691                                    spd1=-RxVal+loadLR;
692                                }
693                                 x=spd1/R1;
694
695
696  //-------------------------------------------------------------------
697                                    spd5=RxVal-M;
698                                    NetM=RxVal-spd5;
699                                    spd6=NetM/R1;
700
```

main.c

```
700
701 //-------------------------------------------------------------------------------
702                                     if(PWM==1)
703                                   {
704                                       if(flag100==1)
705
706                                     {
707                                         pwmM100=1;
708
709                                     }
710                                    }
711 //-------------------------------------------------------------------------------
712                                     if(PWM2==1)
713                                   {
714                                     if(flag200==1)
715
716                                    {
717                                     pwmD100=1;
718                                     htim8.Instance->CCR1=160;
719                                    }
720                                   }
721
722
723
724
725 //-------------------------------------------------------------------------------
726
727//-------------------------------------------------------------------------------
728 //                                    LED FLASHER HERE
729
730                                  if(loadR==1)
731                                  {
732                                      if(LOAD==1)
733                                {
734
735                                  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_SET);
736                                }
737                                   }
738
739                                  if(loadL==1)
740                                  {
741                                     if(LOAD==1)
742                                 {
```

main.c

```
743                                 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);
744
745                                }
746                                 }
747 //-------------------------------------------------------------------------------------------
748
749                                     if(Start==1)
750                                    {
751
752
753                                     if(TestS==1)
754                                     {
755                                         sensor=0*sensor;
756                                         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
757                                         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
758                                         htim13.Instance->CCR1=0;
759                                         htim8.Instance->CCR1=0;
760     //-------------------------------------------------------------------------------
761
762                                    if (leng1>=M)
763                                   {
764                                      PWM=1;
765                                      flag202=0;
766                                      TestS=0;
767
768                                       }
769                                       }
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
```

main.c

```
784
785
786 //-------------------------------------------------------------------------------------------
787
788
789  //-------------------------------------------------------------------------------------------
790
791                                 if (leng1>=M)
792                                     {
793                                       scan=0;
794                                       PWM=1;
795                                       flag202=0;
796                                       TestS=0;
797                                      }
798
799
800
801
802                             }
803
804 //-------------------------------------------------------------------------------------------
805                                    if(sensor>sens)
806                                    {
807                                      PWM=0;
808                                      pwmM100=1;
809                                      flag100=0;
810                                      flag202=1;
811                                      htim13.Instance->CCR1=0;
812                                      PWM3=1;
813                                      flag=1;
814                                    }
815
816                                     if(sensor>1 && sensor<sens)
817
818                                       {
819                                         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);
820                                         HAL_Delay(3);
821                                         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);
822
823                                     //  buzzer=0;
824                                         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_SET);
825                                         HAL_Delay(300);
826                                         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
```

main.c

```
826                                     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
827                                   }
828
829          //-------------------------------------------------------------------------------
830                                     if (sensor==0.000)
831                                {
832                                       if(flag202==0)
833                                    {
834
835
836
837                                     if(brk==1)
838
839                                     {
840                                      htim13.Instance->CCR1=20;
841
842
843                                     }
844                                     if (brk==2)
845                                     {
846                                         htim13.Instance->CCR1=23;  // 68,40
847                                     }
848                                     if (brk==3)
849                                    {
850                                      htim13.Instance->CCR1=27;    // 70,45
851                                    }
852                                     if (brk==4)
853                                    {
854                                       htim13.Instance->CCR1=30;   // 75,49
855                                    }
856                                     if (brk==5)
857                                    {
858                                     htim13.Instance->CCR1=33;      // 80,52
859                                    }
860                  //-------------------------------------------------
861
862                   //-------------------------------------------------------------
863
864                                      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
865                                      if (leng1<=1.5)
866                                     {
867                                         htim13.Instance->CCR1=0;
868                                     }
```

main.c

```
868                                         }
869                                       }
870
871                                     }
872
873                                   }
874
875
876 //-------------------------------------------------------------------------------
877
878 //-------------------------------------------------------------------------------
879                  if(flag==1)
880
881                 {
882
883                      sensor=0*sensor;
884                      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_SET);
885                     // buzzer=1;
886                      HAL_GPIO_WritePin(GPIOC,                    IN_SET);
887                      HAL_Delay(1000);
888                      HAL_GPIO_WritePin(GPIOC,                    IN_RESET);
889                      Start=0;
890
891
892
893                      if(    PWM3==1)
894                      {
895                     for (pwm2 = 0; pwm2 <= 252; pwm2 +=1)
896                           {
897                                 __HAL_TIM_SET_COMPARE(&htim13, TIM_CHANNEL_1, pwm2);
898                                   HAL_Delay(7);
899
900                                   if(pwm2<=252)
901                         {
902
903                                htim13.Instance->CCR1=254;
904
905                                PWM3=0;
906                         }
907                          }
908
909                      }
910
```

Macro Expansion
((uint16_t)0x0400)
Press 'F2' for focus

main.c

```
910
911                     htim8.Instance->CCR1=160;
912                     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_SET);
913                     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
914
915 //-------------------------------------------------------------------------------
916                     TM_HD44780_CLEAR();
917                      sprintf(buf3, "Return len :%3.2f", leng1 );
918                      TM_HD44780_Puts (3,1,buf3);
919                     flag41=0;
920                     flag42=0;
921                     flag19=1;
922 //-------------------------------------------------------------------------------
923                        if(leng1<=1.50)
924
925                   {
926                     htim13.Instance->CCR1=180;
927                     htim8.Instance->CCR1=0;
928
929                    sensor=0*sensor;
930                    TM_HD44780_CLEAR();
931                    flag19=0;
932
933                   pwmM100=0;
934                   flag202=1;
935
936
937                   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_12, GPIO_PIN_RESET);
938                   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);
939                   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
940                   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
941
942                   flag=0;
943
944
945
946
947                 }
948               }
949 //-------------------------------------------------------------------------------
950
951                       if(LOAD==1)
952                {
```

main.c

```
952                 {
953
954                     if(flag101==1)
955                     {
956                     htim13.Instance->CCR1=200;
957                     for (pwm1 = 0; pwm1 <= 165; pwm1 +=3)
958                      {
959                         __HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1, pwm1);
960                         HAL_Delay(40);
961
962                         if(pwm1>=165)
963                        {
964
965
966                           htim8.Instance->CCR1=165;
967                           flag101=0;
968                         }
969                       }
970                   }
971
972                             if(leng1<=0)
973                            {
974                             LOAD=0;
975
976                             htim8.Instance->CCR1=0;
977                             htim13.Instance->CCR1=230;
978                             HAL_Delay(10);
979                             htim13.Instance->CCR1=0;
980
981                            }
982                         }
983
984
985 //-------------------------------------------------------------------------
986
987                        if(speedM==1)
988             {
989                     if(flag101==1)
990                   {
991                      htim13.Instance->CCR1=200;
992                      for (pwm1 = 0; pwm1 <= 162; pwm1 +=3)
993                   {
994                     __HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1, pwm1);
```

main.c

```
994                 __HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1, pwm1);
995                  HAL_Delay(60);
996
997                  if(pwm1>=162)
998               {
999
1000
1001                htim8.Instance->CCR1=162;
1002                flag101=0;
1003              }
1004            }
1005          }
1006              if(leng1<=0)
1007          {
1008              LOAD=0;
1009
1010              htim8.Instance->CCR1=0;
1011              htim13.Instance->CCR1=230;
1012              HAL_Delay(10);
1013              htim13.Instance->CCR1=0;
1014             }
1015
1016            }
1017 //-------------------------------------------------------------------
1018              if(flag24==1)
1019   {
1020
1021              {
1022                 flag0=0;
1023                 HAL_FLASH_Unlock();
1024                 Flash_Write_NUM(0x080E0000 ,meter);
1025                 HAL_FLASH_Lock();
1026                 menu=1;
1027
1028                 enc_value=0;
1029                 flag24=0;
1030          }
1031   }
1032
1033 //-------------------------------------------------------------------
1034
1035                   if(flag50==1)
1036
```

main.c

```
1036
1037                {
1038                        flag24=0;
1039                        HAL_FLASH_Unlock();
1040                        Flash_Write_NUM( 0x081E0000,loadLR);
1041                        HAL_FLASH_Lock();
1042                        enc_value=0;
1043                        LOAD=0;
1044                        flag0=0;
1045                        sum0=0;
1046                        mess1=0;
1047                        stop=0;
1048
1049                        TM_HD44780_Clear();
1050                        menu=1;
1051                        flag50=0;
1052
1053                }
1054 //-----------------------------------------------------
1055                        pwm();
1056
1057 //-----------------------------------------------------
1058
1059                           if(flag32==1)
1060                        {
1061                              if(leng1>=sum0)
1062
1063
1064                         {
1065
1066                           loadL=0;
1067
1068                           HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
1069                           HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
1070                           HAL_GPIO_WritePin(GPIOC, GPIO_PIN_10, GPIO_PIN_RESET);
1071
1072                           htim13.Instance->CCR1=230;
1073                           HAL_Delay(100);
1074                           htim13.Instance->CCR1=0;
1075                           htim8.Instance->CCR1=0;
1076                           leng1=0;
1077                           enc_value=0;
1078            volatile       loadLR;
```

```
1078            volatile    loadLR;
1079                        zero=0;
1080                        LOAD=0;
1081
1082                       flag19=0;
1083
1084                       if( flag0==1)
1085                       {
1086                        flag50=1;
1087                       }
1088                        flag32=0;
1089                      }
1090                  }
1091
1092 //-----------------------------------------------------------------
1093
1094                      if(flag34==1)
1095                {
1096                  if(leng1>=sum0)
1097                 {
1098                     loadR=0;
1099
1100                     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
1101                     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
1102                     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_RESET);
1103
1104                     htim13.Instance->CCR1=230;
1105                     HAL_Delay(100);
1106                     htim13.Instance->CCR1=0;
1107                     htim8.Instance->CCR1=0;
1108                     LOAD=0;
1109                     enc_value=0;
1110                     leng1=0;
1111
1112                    if(flag0==1)
1113                     {
1114                     flag50=1;
1115                     }
1116                     flag34=0;
1117                    }
1118
1119                    }
1120
```

main.c

```
1120
1121
1122 //-------------------------------------------------------------------------------
1123
1124                 HAL_FLASH_Unlock();
1125
1126                 RxVal = Flash_Read_NUM(0x081E0000);
1127                 HAL_FLASH_Lock();
1128
1129
1130 //-------------------------------------------------------------------------
1131
1132               HAL_FLASH_Unlock();
1133               M = Flash_Read_NUM(0x080E0000 );
1134               HAL_FLASH_Lock();
1135
1136
1137
1138 //-------------------------------------------------------------------------------
1139
1140                           if(LOAD==1)
1141                     {
1142
1143                       if(flag32==1)
1144                       {
1145
1146                         stop=0;
1147                         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_SET);
1148                         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
1149                       }
1150
1151                       if(flag34==1)
1152                       {
1153
1154                          stop=0;
1155                          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
1156                          HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);
1157                       }
1158                     }
1159
1160 //-------------------------------------------------------------------------------
1161
1162                    if (StBy==1)
```

main.c

```
1162                    if (StBy==1)
1163
1164
1165            {
1166                  sensor=0*sensor;   //  IN 23-5-23
1167                  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4,GPIO_PIN_SET );
1168                  HAL_Delay(500);
1169                  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4,GPIO_PIN_RESET );
1170
1171                   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_SET);
1172                   HAL_Delay(300);
1173                   HAL_GPIO_WritePin(GPIOC, GPIO_PIN_11, GPIO_PIN_RESET);
1174                   stop=1;
1175                   Start=0;
1176                  if(loadL==1)
1177              {
1178                  if(menu==1)
1179                  {
1180                     LOAD=0;
1181                     loadL=0;
1182                  }
1183               }
1184                  else if(loadR==1)
1185                {
1186                     if(menu==1)
1187                  {
1188                     LOAD=0;
1189                     loadR=0;
1190                  }
1191              }
1192
1193                  tsapari=0;
1194
1195  //-----------------------------------------------------------------
1196
1197                  if(leng1<=1.5)
1198                  {
1199                    htim13.Instance->CCR1 =200;
1200                  }
1201
1202                  TM_HD44780_Puts (16,4,(char*)(arr1 ));
1203                  delay(10);
1204 // -----------------------------------------------------------------
```

main.c

```
1204 // -----------------------------------------------------------
1205                     if(stop=1)
1206           {
1207             LOAD=0;
1208             htim8.Instance->CCR1 =0;
1209 
1210             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
1211             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
1212 
1213           }
1214 
1215         }
1216                     delay(2000);
1217 //-------------------------------------------------------------------
1218 
1219 
1220  //-------------------------------------------------------------------
1221        c = KeypadGetKey();
1222 
1223        if (c != KEYPAD_NO_PRESSED)
1224 
1225    {
1226 //-------------------------------------------------------------------
1227       switch (c)
1228 //-------------------------------------------------------------------
1229      {
1230         case '1':
1231 
1232        {
1233 
1234         if(menu==3)
1235        {
1236          sens+=5;
1237          HAL_Delay(150);
1238        }
1239          if(sens>100)
1240        {
1241 
1242          sens=100;
1243        }
1244 
1245          if (menu==3 )
1246      {
```

main.c

```
1246      {
1247         TM_HD44780_CLEAR();
1248
1249        TM_HD44780_Puts (2,1,(char*)(arr3_1 ));
1250
1251         sprintf(buf,"%3.3d",sens);
1252         TM_HD44780_Puts (14,4,buf);
1253       }
1254 //-------------------------------------------------
1255
1256         if(menu==2)
1257         {
1258
1259         brk+=1;
1260         HAL_Delay(100);
1261         pwmM=brk;
1262
1263         }
1264
1265          if(brk>5)
1266         {
1267           brk=0;
1268         }
1269
1270          if(menu==2)
1271        {
1272
1273              TM_HD44780_CLEAR();
1274         TM_HD44780_Puts (2,1,(char*)(arr4_1));
1275
1276
1277         sprintf(buf,"%3.3d",brk);
1278         TM_HD44780_Puts (14,4,buf);
1279        }
1280
1281  //-------------------------------------------------
1282
1283          if(menu==5)
1284
1285         {
1286
1287         meter+=1;
1288         HAL_Delay(100);
```

main.c

```
1288          HAL_Delay(100);
1289          zero=meter;
1290
1291          }
1292
1293            if(meter>990)
1294          {
1295          meter=990;
1296          }
1297
1298            if(menu==5)
1299          {
1300
1301
1302           TM_HD44780_CLEAR();
1303           TM_HD44780_Puts (2,1,(char*)(arr9 ));
1304           sprintf(buf, "%3.3d",meter);
1305           TM_HD44780_Puts (14,4,buf);
1306          }
1307            break;
1308        }
1309 //-----------------------------------------------
1310
1311        case '2':
1312
1313        {
1314
1315           if(menu==3)
1316        {
1317                sens-=5;
1318                HAL_Delay(150);
1319        }
1320
1321          if(sens<0)
1322          {
1323          sens=0;
1324          }
1325
1326          if(menu==3)
1327         {
1328           TM_HD44780_CLEAR();
1329          TM_HD44780_Puts (2,1,(char*)(arr3_2 ));
1330          sprintf(buf, "%3.3d",sens);
```

main.c

```
1330          sprintf(buf, "%3.3d",sens);
1331          TM_HD44780_Puts (14,4,buf);
1332         }
1333
1334 //-----------------------------------------
1335
1336       if(menu==2)
1337       {
1338
1339          brk-=1;
1340          HAL_Delay(100);
1341          pwmM=brk;
1342
1343         if(brk<=0)
1344         {
1345           brk=0;
1346         }
1347
1348     if(menu==2)
1349      {
1350
1351         TM_HD44780_CLEAR();
1352         TM_HD44780_Puts (2,1,(char*)(arr4_2 )
1353
1354
1355         sprintf(buf, "%3.3d",brk);
1356         TM_HD44780_Puts (14,4,buf);
1357       }
1358
1359     }
1360 //-----------------------------------------
1361       if(menu==5)
1362        {
1363
1364        meter-=1;
1365        HAL_Delay(100);
1366        zero=meter;
1367
1368       }
1369
1370        if(meter<0)
1371        {
1372         meter=0;
```

main.c

```
1372          meter=0;
1373
1374        }
1375
1376        if(menu==5)
1377
1378      {
1379
1380
1381        TM_HD44780_CLEAR();
1382        TM_HD44780_Puts (2,1,(char*)(arr9_1 ));
1383        sprintf(buf, "%3.3d",meter);
1384        TM_HD44780_Puts (14,4,buf);
1385      }
1386
1387          break;
1388      }
1389
1390 //-------------------------------------------------
1391                 case '3':
1392       {
1393         CLR=menu+=1;
1394
1395          TM_HD44780_CLEAR();
1396
1397               if(menu>5)
1398            {
1399               menu=1;
1400
1401              HAL_Delay(150);
1402            }
1403 //-------------------------------------------------
1404
1405        if (menu==2)
1406      {
1407
1408        TM_HD44780_CLEAR();
1409        TM_HD44780_Puts (0,4,(char*)(arr4 ));
1410        TM_HD44780_Puts (0,1,(char*)(arr4 +1));
1411        TM_HD44780_Puts (0,2,(char*)(arr4 +2));
1412        lcd_flag=0;
1413       }
1414
```

```
1414
1415
1416        if (menu==3)
1417      {
1418
1419
1420            TM_HD44780_CLEAR();
1421        TM_HD44780_Puts (0,4,(char*)(arr3 ));
1422        TM_HD44780_Puts (0,1,(char*)(arr3+1 ));
1423        TM_HD44780_Puts (0,2,(char*)(arr3+2 ));
1424      }
1425
1426
1427       if(menu==4)
1428      {
1429
1430
1431           TM_HD44780_CLEAR();
1432         TM_HD44780_Puts (0,4,(char*)(arr2 ));
1433         TM_HD44780_Puts (0,1,(char*)(arr2+1 ));
1434         TM_HD44780_Puts (0,3,(char*)(arr2+2 ));
1435
1436       }
1437
1438       if(menu==5)
1439       {
1440
1441
1442         TM_HD44780_CLEAR();
1443         TM_HD44780_Puts (0,4,(char*)(arr9_2 ));
1444         TM_HD44780_Puts (0,1,(char*)(arr9_2+1 ));
1445         TM_HD44780_Puts (0,2,(char*)(arr9_2+2 ));
1446
1447       }
1448
1449       if(menu==8){
1450       TM_HD44780_Puts (0,4,(char*)(arr17 ));
1451       TM_HD44780_Puts (0,1,(char*)(arr17+1 ));
1452       }
1453                    if(menu==7)
1454                    {
1455                                 sprintf(buf2, "sens:%3.5f",sensor);
1456                                 TM_HD44780_Puts (0,4,buf2);
```

main.c

```
1456                                TM_HD44780_Puts (0,4,buf2);
1457
1458
1459                                sprintf(buf4, "batt:%3.2f ");
1460                                TM_HD44780_Puts (0,3,buf4);
1461                                sprintf(buf3, "len :%3.2f", leng1 );
1462                                TM_HD44780_Puts (0,1,buf3);
1463                                sprintf(buf5,"L%3.2f",RxVal);
1464                                TM_HD44780_Puts (14,3,buf5);
1465                                sprintf(buf6,"M%1.2d",M);
1466                                TM_HD44780_Puts (14,2,buf6);
1467
1468                    }
1469
1470         break;
1471        }
1472
1473 //-------------------------------------------------------------------
1474        case '7':
1475
1476    {
1477
1478      loadLR+=20;
1479
1480
1481      if(menu==4)
1482   {
1483
1484          if(loadLR>RxVal)
1485          {
1486           flag34=0;
1487           flag32=1;
1488           loadL=1;
1489            loadR=0;
1490
1491            sum0=(-RxVal)+loadLR;
1492           }
1493
1494          else if(loadLR<RxVal)
1495
1496                          {
1497                            flag34=1;
1498                            flag32=0;
```

main.c

```
1498                            flag32=0;
1499                            loadL=0;
1500                            loadR=1;
1501                            sum0=RxVal-loadLR;
1502                           }
1503
1504       if(loadLR<0)
1505        {
1506             loadLR=0;
1507        }
1508
1509        if(loadLR>999)
1510       {
1511        loadLR=999;
1512       }
1513   }
1514       if(menu==4)
1515       {
1516            if(loadLR>RxVal)
1517            {
1518
1519
1520        TM_HD44780_CLEAR();
1521
1522
1523         TM_HD44780_CLEAR();
1524        TM_HD44780_Puts (2,1,(char*)(arr2_1 ));
1525
1526        TM_HD44780_Puts (11,1,(char*)(arr2_1 +1));
1527
1528        sprintf(buf,"%3.3d",loadLR);
1529        TM_HD44780_Puts (8,1,buf);
1530        sprintf(buf5,"L%2.2f",RxVal);
1531        TM_HD44780_Puts (8,3,buf5);
1532        }
1533
1534            else  if(loadLR<RxVal)
1535            {
1536              TM_HD44780_CLEAR();
1537              TM_HD44780_CLEAR();
1538              TM_HD44780_Puts (13,1,(char*)(arr2_2 ));
1539              TM_HD44780_Puts (11,1,(char*)(arr2_2 +1));
1540              sprintf(buf5,"%3.3d",loadLR);
```

```
1540            sprintf(buf5,"%3.3d",loadLR);
1541            TM_HD44780_Puts (8,1,buf5);
1542            sprintf(buf5,"L%2.2f",RxVal);
1543            TM_HD44780_Puts (8,3,buf5);
1544            }
1545         }
1546
1547        break;
1548      }
1549 //-------------------------------------------------------------
1550
1551               case '8':
1552             {
1553                 loadLR-=20;
1554
1555                 if(menu==4)
1556             {
1557                 if(loadLR<RxVal)
1558
1559               {
1560                 flag34=1;
1561                 flag32=0;
1562                 loadL=0;
1563                 loadR=1;
1564
1565                 sum0=RxVal-loadLR;
1566               }
1567
1568                 else if(loadLR>RxVal)
1569                        {
1570                         flag34=0;
1571                         flag32=1;
1572                         loadL=1;
1573                          loadR=0;
1574                          sum0=(-RxVal)+loadLR;
1575                         }
1576          if(loadLR<0)
1577     {
1578         loadLR=0;
1579     }
1580      }
1581
1582     if(menu==4)
```

```
1582      if(menu==4)
1583     {
1584     if(loadLR<RxVal)
1585          {
1586       TM_HD44780_CLEAR();
1587
1588
1589       TM_HD44780_CLEAR();
1590       TM_HD44780_Puts (13,1,(char*)(arr2_2 ));
1591
1592       TM_HD44780_Puts (11,1,(char*)(arr2_2 +1));
1593       TM_HD44780_Puts (3,1,(char*)(arr2_2 +2));
1594       sprintf(buf5,"%3.3d",loadLR);
1595       TM_HD44780_Puts (8,1,buf5);
1596       sprintf(buf5,"L%2.2f",RxVal);
1597       TM_HD44780_Puts (8,3,buf5);
1598       }
1599
1600          else if(loadLR>RxVal)
1601               {
1602
1603            TM_HD44780_CLEAR();
1604
1605
1606             TM_HD44780_CLEAR();
1607            TM_HD44780_Puts (2,1,(char*)(arr2_1 ));
1608
1609            TM_HD44780_Puts (11,1,(char*)(arr2_1 +1));
1610            TM_HD44780_Puts (13,1,(char*)(arr2_1 +2));
1611            sprintf(buf,"%3.3d",loadLR);
1612            TM_HD44780_Puts (8,1,buf);
1613            sprintf(buf5,"L%2.2f",RxVal);
1614            TM_HD44780_Puts (8,3,buf5);
1615            }
1616
1617           }
1618        break;
1619       }
1620
1621 //-----------------------------------------------------------------
1622
1623          case 'A':
1624       {
```

```
1624        {
1625            TM_HD44780_CLEAR();
1626            flag41=1;
1627            htim13.Instance->CCR1 =0;
1628            TM_HD44780_Puts (16,4,(char*)(arr14 ));
1629
1630            if(menu==1)
1631            {
1632               Start=1;
1633             }
1634
1635            TestS=1;
1636
1637              StBy=0;
1638
1639              if(menu==4)
1640              {
1641              LOAD=1;
1642
1643              }
1644
1645          break;
1646        }
1647
1648        delay(100);
1649    //-----------------------------------------------
1650
1651
1652             case 'B':
1653             {
1654              TM_HD44780_CLEAR();
1655               if(menu==5)
1656              {
1657              flag24=1;
1658
1659              }
1660
1661             if(menu==4)
1662             {
1663              flag0=1;
1664
1665              mess1=1;
1666              TM_HD44780_CLEAR();
```

```
1666                TM_HD44780_CLEAR();
1667
1668                 LOAD=1;
1669                 flag101=1;
1670               }
1671
1672              Start=0;
1673              flag18=1;
1674
1675
1676
1677   if(LOAD==1)
1678
1679 {
1680
1681       if (loadL==1)
1682  {
1683         loadR=0;
1684        sensor=sensor*0;
1685        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_SET);
1686        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
1687        flag32=1;
1688  }
1689
1690 //---------------------------------------
1691
1692     if(loadR==1)
1693 {
1694        loadL=0;
1695        sensor=sensor*0;
1696        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_3, GPIO_PIN_RESET);
1697        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);
1698        flag34=1;
1699 }
1700
1701 }
1702         if(menu==5)
1703        {
1704
1705         P1=0;
1706
1707        }
1708
```

```
1708
1709 //-----------------------------------------
1710
1711         if(menu==6)
1712
1713         {
1714
1715          menu=1;
1716         }
1717              delay(31000);
1718          break;
1719        }
1720
1721 //-----------------------------------------------------
1722                 case '0':
1723
1724           {
1725
1726               sensor=100;
1727               flag101=1;
1728
1729           break;
1730
1731          }
1732 //-----------------------------------------------------
1733
1734
1735          case '6':
1736
1737          {
1738
1739              Start=0;
1740
1741              flag41=0;
1742
1743              if(flag19==1)
1744                 {
1745                    StBy=0;
1746                 }
1747
1748
1749                 if(leng1<=1.50)
1750                 {
```

```
1750                {
1751                     StBy=1;
1752                }
1753
1754
1755                     flag=0;
1756                 break;
1757              }
1758          }
1759       }
1760      }
1761 }
1762
1763 //-----------------------------------------------------------------------------
1764
1765     /* USER CODE END WHILE */
1766
1767     /* USER CODE BEGIN 3 */
1768
1769   /* USER CODE END 3 */
1770   /*
1771   * @brief System Clock Configuration
1772   * @retval None
1773   */
1774 void SystemClock_Config(void)
1775 {
1776   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
1777   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
1778
1779   /** Supply configuration update enable
1780   */
1781   HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);
1782   /** Configure the main internal regulator output voltage
1783   */
1784   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);
1785
1786   while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}
1787   /** Initializes the RCC Oscillators according to the specified parameters
1788   * in the RCC_OscInitTypeDef structure.
1789   */
1790   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI48|RCC_OSCILLATORTYPE_HSE;
1791   RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
1792   RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
```

```
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : Mgnetic_Pin */
  GPIO_InitStruct.Pin = Mgnetic_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(Mgnetic_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
//-------------------------------------------------------------------------------
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
{
    if(State == IDLE)
 {
     T1 = TIM2->CCR1;
     TIM2_OVC = 0;
     State = DONE;
 }
    else if(State == DONE)
{
     T2 = TIM2->CCR1;

     Ticks = (T2 + (TIM2_OVC * 65536)) - T1;
     Freq = (F_CLK/Ticks);


    if(Freq != 0)
{

     sec=(Ticks*0.133);



}
}
}

//-------------------------------------------------------------------------------
```

```
2451
2452
2453        {
2454         if (enc_value>0 && enc_value<1230 )
2455            {
2456             leng2=((enc_value*point2)/100);
2457             leng1=leng2;
2458
2459            }
2460
2461          else   if  (enc_value>1230 &&  enc_value <15000)
2462            {
2463             leng3=(((enc_value-1230)*point1)/100);
2464             leng1=leng3;
2465             leng1=(leng2+leng3);
2466            }
2467         }
2468
2469 //-----------------------------------------------------------------
2470
2471
2472        {
2473
2474         if (enc_value<0 && enc_value<1230 )
2475            {
2476             leng2=((enc_value*(-point2))/100);
2477             leng1=leng2;
2478
2479            }
2480
2481                else   if  (enc_value>(-1230) &&  enc_value <15000)
2482            {
2483                leng3=(((enc_value-1230)*(-point1))/100);
2484
2485            }
2486         }
2487
2488
2489
2490
2491 }
2492
```

```
2493
2494
2495 //-------------------------------------------------------------------------------------
2496   void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
2497
2498  {
2499       TIM2_OVC++;
2500
2501       if(TIM2_OVC==700)
2502      {
2503       TIM2_OVC=0,Ticks=0, T_ms=0, Freq=0, T1=0, T2=0, sensor=0, u=0, m=0, F=0;
2504        sec=0;
2505      }
2506
2507   }
2508
2509 //-------------------------------------------------------------------------------------
2510
2511 /* USER CODE END 4 */
2512
2513 /**
2514   * @brief  This function is executed in case of error occurrence.
2515   * @retval None
2516   */
2517 void Error_Handler(void)
2518 {
2519   /* USER CODE BEGIN Error_Handler_Debug */
2520   /* User can add his own implementation to report the HAL error return state */
2521   __disable_irq();
2522   while (1)
2523   {
2524   }
2525   /* USER CODE END Error_Handler_Debug */
2526 }
2527
2528 #ifdef  USE_FULL_ASSERT
2529 /**
2530   * @brief  Reports the name of the source file and the source line number
2531   *         where the assert_param error has occurred.
2532   * @param  file: pointer to the source file name
2533   * @param  line: assert_param error line source number
2534   * @retval None
2535   */
```

Writable | Smart Insert | 1 : 1 : 0

```
2535   */
2536 void assert_failed(uint8_t *file, uint32_t line)
2537 {
2538   /* USER CODE BEGIN 6 */
2539   /* User can add his own implementation to repo
2540      ex: printf("Wrong parameters value: file %s
2541   /* USER CODE END 6 */
2542 }
2543 #endif /* USE_FULL_ASSERT */
2544
2545 /************************ (C) COPYRIGHT STMicroe
2546
```

main.c keypad.c

```c
/** Includes ------------------------------------------------
#include "keypad.h"
#include "tm_stm32f4_hd44780.h"

GPIO_InitTypeDef _GPIO_InitStructKeypad;
extern TIM_HandleTypeDef htim15;
/** Public functions ----------------------------------------
/**
  ************************************************************
  * @brief  Initialize GPIO pins for keypad.
  * @param  None
  * @retval None
  ************************************************************
  */
/**
  ************************************************************
  * @brief  Get which key is pressed by scanning the columns a
  * @param  None
  * @retval Pressed key char value.
  ************************************************************
  ************************************************************
  ************************************************************
  ************************************************************
 */

/*
  void delay(uint16_t delay);

 {
  __HAL_TIM_SET_COUNTER(&htim15,0);
  while(__HAL_TIM_GET_COUNTER(&htim15) < delay);
 }

*/
extern menu;
extern flag19;
#define delay;
extern LOAD;
extern loadLR;
extern float leng1;
extern S1;
extern S2;
extern delete;
```

main.c | keypad.c

```
43 extern delete;
44 extern StBy;
45 extern sensor;
46 extern sens;
47 uint8_t KeypadGetKey()
48
49
50
51
52 {
53     // Scan column 1 (column 1 pin is grounded, other colu
54
55     HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_RESET);
56     HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
57     HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
58       HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
59       HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
60     //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PI
61       HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
62       HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
63       HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
64       HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
65    //         -----------------------
66      HAL_Delay(350);              // this setting here (spe
67
68      // delay(50000);
69     // Read rows
70
71     if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
72
73         return '3';       //  knob 1  (menu)
74 /*
75        if(flag19==1)
76        {
77          return '9';
78        }
79 */
80
81     // Scan column 2 (column 2 pin is grounded, other colu
82     HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
83     HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_RESET);
84     HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
85       HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
```

main.c keypad.c

```
85        HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
86        HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
87      //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_SET);
88       HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
89       HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
90       HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
91       HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
92      HAL_Delay(50);
93        //delay(50000);
94      // Read rows
95
96      if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
97
98          return '1';
99
100
101
102     // Scan column 3 (column 3 pin is grounded, other column pins
103       HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
104     HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
105     HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_RESET);
106       HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
107       HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
108     //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_SET);
109      HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
110      HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
111      HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
112      HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
113       HAL_Delay(30);
114         //delay(50000);
115     // Read rows
116
117     if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
118
119         return '2';
120
121 //-------------------------------------------------------------------
122
123     // Scan column 4 (column 4 pin is grounded, other column pins
124               HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
125             HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
126             HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
127               HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_RESET);
```

main.c keypad.c

```
127                HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_RESET);
128                HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
129              //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_SET);
130               HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
131               HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
132               HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
133               HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
134               HAL_Delay(30);
135              // delay(50000);
136              // Read rows
137
138              if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
139
140
141                  return '7';
142
143
144
145      //---------------------------------------------------------------------
146
147       // Scan column 5 (column 5 pin is grounded, other column pins is open
148        HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
149      HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
150      HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
151        HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
152        HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_RESET);
153      //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_SET);
154        HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
155        HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
156        HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
157        HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
158        HAL_Delay(30);
159       // delay(50000);
160      // Read rows
161
162      if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
163
164
165          return '8';
166
167
168  //---------------------------------------------------------------------
169       // Scan column 6 (column 6 pin is grounded, other column pins is open
```

main.c keypad.c

```
169      // Scan column 6 (column 6 pin is grounded, other column pi
170       HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
171       HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
172     HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
173       HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
174       HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
175     //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_RESE
176       HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_RESET);
177       HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
178       HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
179       HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
180       HAL_Delay(30);
181      // delay(50000);
182     // Read rows
183
184     if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
185
186               return 'B'; //
187
188
189      // Scan column 7 (column 7 pin is grounded, other column pi
190           HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
191           HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
192         HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
193           HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
194           HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
195         //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_
196           HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
197           HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_RESET);
198           HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
199           HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
200           HAL_Delay(30);
201           // delay(50000);
202         // Read rows
203
204         if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
205
206              return '0';
207
208 /*
209          {
210
211          if(leng1>=1)       // condition activated button delete
```

main.c | *keypad.c

```
211         if(leng1>=1)       // condition activated button delete
212         {
213
214            return '0';
215         }
216
217         }
218 */
219
220
221         // Scan column 8 (column 8 pin is grounded, other column pins is
222                   HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
223                   HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
224                 HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
225                   HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
226                   HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
227                 //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_RE
228                   HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
229                   HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
230                   HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_RESET);
231                   HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_SET);
232                   HAL_Delay(30);
233                  // delay(50000);
234                 // Read rows
235
236                 if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
237
238         // Scan column 9 (column 9 pin is grounded, other column pins is o
239                   HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
240                   HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
241                 HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
242                   HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
243                   HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
244                 //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_RE
245                   HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
246                   HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
247                   HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
248                   HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_RESET);
249                 HAL_Delay(30);
250                 //  delay(50000);
251                 // Read rows
252
253                 if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
```

main.c keypad.c

```
249         // Scan column 9 (column 9 pin is grounded, other column pins is
250                   HAL_GPIO_WritePin(GPIOF, COL_1_Pin,GPIO_PIN_SET);
251                   HAL_GPIO_WritePin(GPIOF, COL_2_Pin,GPIO_PIN_SET);
252                 HAL_GPIO_WritePin(GPIOF, COL_3_Pin,GPIO_PIN_SET);
253                   HAL_GPIO_WritePin(GPIOD, COL_4_Pin,GPIO_PIN_SET);
254                   HAL_GPIO_WritePin(GPIOF, COL_5_Pin,GPIO_PIN_SET);
255                 //HAL_GPIO_WritePin(COL_6_GPIO_Port, COL_6_Pin,GPIO_PIN_
256                   HAL_GPIO_WritePin(GPIOF, COL_6_Pin,GPIO_PIN_SET);
257                   HAL_GPIO_WritePin(GPIOC, COL_7_Pin,GPIO_PIN_SET);
258                   HAL_GPIO_WritePin(GPIOF, COL_8_Pin,GPIO_PIN_SET);
259                   HAL_GPIO_WritePin(GPIOF, COL_9_Pin,GPIO_PIN_RESET);
260                 HAL_Delay(30);
261                 //  delay(50000);
262                 // Read rows
263
264                 if (!HAL_GPIO_ReadPin(ROW_1_GPIO_Port, ROW_1_Pin))
265         {
266                     if(sensor>sens)   // command for delete
267                     {
268                         __NOP();
269                     }
270
271
272                     if(menu==1)  //with menu=1 it works normally, while
273                  {
274                     return 'A';
275                  }
276                     else if(menu==4)
277                                     {
278                                         __NOP();
279                                     }
280
281
282
283             }
284
285
286
287
288     return KEYPAD_NO_PRESSED;
289 }
290 /****************************** END OF FILE ************************
291 /*************************************************************************
```

main.c PWM.c

```c
/**Includes ------------------------------------

#include "PWM.h"

GPIO_InitTypeDef _GPIO_PWM;

extern TIM_HandleTypeDef htim8;
extern TIM_HandleTypeDef htim13;

/** Public functions -----------------------------

/**
 ************************************************
 * @brief   Initialize GPIO pins for keypad.
 *
 * @param   None
 * @retval  None
 ************************************************
 */
/**
 ************************************************
 * @brief   Get which key is pressed by scanning th
 * @param   None
 * @retval  Pressed key char value.
 ************************************************
 */

extern int pwmM0;
extern int pwmM1;
extern int pwmM5;
extern int pwmM10;
extern int pwmM20;
extern int pwmM25;
extern int pwmM50;
extern int pwmM100; // Magnetic pwmM100 = 100%pwm

extern float leng1;
extern float spd4;
extern int flag00;
extern int flag101;
extern int flag102;

extern int pwmD0;  // we don't know values in exter
```

main.c PWM.c

```
43 extern int pwmD0;  // we don't know values in external varia
44 extern int pwmD1;
45 extern int pwmD5;
46 extern int pwmD10;
47 extern int pwmD20;
48 extern int pwmD25;
49 extern int pwmD50;
50 extern int pwmD100;  // Driver Motor pwmD100 =100%pwm from T
51
52 extern int flag35;
53 extern int flag100;
54 extern int flag200;
55 extern int flag202;
56 extern int flag203;
57 extern int speedD;
58 extern int LOAD;
59 extern int PWM;
60 extern int PWM2;
61 extern int PWM3;
62 extern int pwm1;          // for driver motor
63 extern int pwm2;          // for driver magnetic
64 extern int pwm2stp;
65
66 extern float spd9;
67 extern float spd8;
68 extern float spd7;
69
70 void pwm(void)
71
72  {
73        if (PWM == 1)       //  condition for pwm magnetic =
74     {
75
76
77           if(pwmM100==0)
78     {
79         for (pwm2 = 0; pwm2 <= 252; pwm2 +=3)    //  for pwm
80       {
81             __HAL_TIM_SET_COMPARE(&htim13, TIM_CHANNEL_1, pw
82               HAL_Delay(1);
83              // pwmM100 = 0;
84              // LOAD=0;
85            if(pwm2==252)
```

main.c PWM.c

```
85            if(pwm2==252)
86            {
87       for (pwm2 = 230; pwm2 <= 0; pwm2 -=3)    //  for pwm2 = 0-255
88                    {
89                           __HAL_TIM_SET_COMPARE(&htim13, TIM_CHANNEL_1, pwm2);
90                             HAL_Delay(20);
91                            // pwmM100 = 0;
92                            // LOAD=0;
93                    }
94             }
95               if(pwm2<=3)
96              //  if(pwm2>=3)
97                 {
98
99                   flag100=1;
100                }
101          }
102       }
103     }
104
105
106       if (PWM2 == 1)       //  condition for pwm MOTER = 100%
107        {
108
109
110             if(pwmD100==0)
111        {
112            //for (pwm2 = 0; pwm2 <= 252; pwm2 +=3)    //  for pwm2 = 0-255
113               for (pwm1 = 0; pwm1 <= 160; pwm1 +=3)
114
115          {
116                __HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1, pwm1);   // pwm
117                  HAL_Delay(20);
118                 // pwmM100 = 0;
119                 // LOAD=0;
120                  if(pwm1>=160)       //   <------pwm1 !
121                  {
122
123                    flag200=1;
124                  }
125             }
126          }
127        }
```

main.c | tm_stm32f4_hd44780.c

```c
#include "main.h"
#include "tm_stm32f4_hd44780.h"
typedef struct {
    uint8_t DisplayControl;
    uint8_t DisplayFunction;
    uint8_t DisplayMode;
    uint8_t Rows;
    uint8_t Cols;
    uint8_t currentX;
    uint8_t currentY;
} HD44780_Options_t;

/* Private functions */
static void TM_HD44780_InitPins(void);
static void TM_HD44780_Cmd(uint8_t cmd);
static void TM_HD44780_Cmd4bit(uint8_t cmd);
static void TM_HD44780_Data(uint8_t data);
static void TM_HD44780_CursorSet(uint8_t col, uint8_t row);

/* Private variable */
static HD44780_Options_t HD44780_Opts;

/* Pin definitions */
#define HD44780_RS_LOW              HAL_GPIO_WritePin(HD44780_RS_PORT, HD44780_RS_PIN,GPIO_PIN_RESET)
#define HD44780_RS_HIGH             HAL_GPIO_WritePin(HD44780_RS_PORT, HD44780_RS_PIN,GPIO_PIN_SET)
#define HD44780_E_LOW               HAL_GPIO_WritePin(HD44780_E_PORT, HD44780_E_PIN,GPIO_PIN_RESET)
#define HD44780_E_HIGH              HAL_GPIO_WritePin(HD44780_E_PORT, HD44780_E_PIN,GPIO_PIN_SET)

 #define HD44780_E_BLINK           HD44780_E_HIGH; delay(30000); HD44780_E_LOW; delay(30000)
//#define HD44780_E_BLINK             HD44780_E_HIGH; HD44780_Delay(1); HD44780_E_LOW; HD44780_Delay(1)

#define HD44780_Delay(x)            HAL_Delay(x)
#define delay(x)           delay(x)
//#define HD44780_delay(x)            HAL_delay(x)


/* Commands*/
#define HD44780_CLEARDISPLAY        0x01
#define HD44780_RETURNHOME          0x02
#define HD44780_ENTRYMODESET        0x04
#define HD44780_DISPLAYCONTROL      0x08
#define HD44780_CURSORSHIFT         0x10
#define HD44780_FUNCTIONSET         0x20
```

main.c | tm_stm32f4_hd44780.c

```
43 #define HD44780_FUNCTIONSET         0x20
44 #define HD44780_SETCGRAMADDR        0x40
45 #define HD44780_SETDDRAMADDR        0x80
46
47 /* Flags for display entry mode */
48 #define HD44780_ENTRYRIGHT          0x00
49 #define HD44780_ENTRYLEFT           0x02
50 #define HD44780_ENTRYSHIFTINCREMENT 0x01
51 #define HD44780_ENTRYSHIFTDECREMENT 0x00
52
53 /* Flags for display on/off control */
54 #define HD44780_DISPLAYON           0x04
55 #define HD44780_CURSORON            0x02
56 #define HD44780_BLINKON             0x01
57
58 /* Flags for display/cursor shift */
59 #define HD44780_DISPLAYMOVE         0x08
60 #define HD44780_CURSORMOVE          0x00
61 #define HD44780_MOVERIGHT           0x04
62 #define HD44780_MOVELEFT            0x00
63
64 /* Flags for function set */
65 #define HD44780_8BITMODE            0x10
66 #define HD44780_4BITMODE            0x00
67 #define HD44780_2LINE               0x08
68 #define HD44780_1LINE               0x00
69 #define HD44780_5x10DOTS            0x04
70 #define HD44780_5x8DOTS             0x00
71
72 extern TIM_HandleTypeDef htim15;
73
74
75
76
77 void TM_HD44780_Init(uint8_t cols, uint8_t rows) {
78     /* Initialize delay */
79     //TM_DELAY_Init();
80
81     /* Init pinout */
82     TM_HD44780_InitPins();
83
84     /* At least 40ms */
85     //HD44780_Delay(40);                    //  here 500
```

main.c | tm_stm32f4_hd44780.c

```
43 #define HD44780_FUNCTIONSET         0x20
44 #define HD44780_SETCGRAMADDR        0x40
45 #define HD44780_SETDDRAMADDR        0x80
46
47 /* Flags for display entry mode */
48 #define HD44780_ENTRYRIGHT          0x00
49 #define HD44780_ENTRYLEFT           0x02
50 #define HD44780_ENTRYSHIFTINCREMENT 0x01
51 #define HD44780_ENTRYSHIFTDECREMENT 0x00
52
53 /* Flags for display on/off control */
54 #define HD44780_DISPLAYON           0x04
55 #define HD44780_CURSORON            0x02
56 #define HD44780_BLINKON             0x01
57
58 /* Flags for display/cursor shift */
59 #define HD44780_DISPLAYMOVE         0x08
60 #define HD44780_CURSORMOVE          0x00
61 #define HD44780_MOVERIGHT           0x04
62 #define HD44780_MOVELEFT            0x00
63
64 /* Flags for function set */
65 #define HD44780_8BITMODE            0x10
66 #define HD44780_4BITMODE            0x00
67 #define HD44780_2LINE               0x08
68 #define HD44780_1LINE               0x00
69 #define HD44780_5x10DOTS            0x04
70 #define HD44780_5x8DOTS             0x00
71
72 extern TIM_HandleTypeDef htim15;
73
74
75
76
77 void TM_HD44780_Init(uint8_t cols, uint8_t rows) {
78     /* Initialize delay */
79     //TM_DELAY_Init();
80
81     /* Init pinout */
82     TM_HD44780_InitPins();
83
84     /* At least 40ms */
85     //HD44780_Delay(40);                    //  her
```

main.c | tm_stm32f4_hd44780.c

```
85      //HD44780_Delay(40);                    //  here 500
86      delay(30000);
87      /* Set LCD width and height */
88      HD44780_Opts.Rows = rows;
89      HD44780_Opts.Cols = cols;
90
91      /* Set cursor pointer to beginning for LCD */
92      HD44780_Opts.currentX = 0;
93      HD44780_Opts.currentY = 0;
94
95      HD44780_Opts.DisplayFunction = HD44780_4BITMODE | HD44780_5x8DOTS | HD44780_1LINE;
96      if (rows > 1) {
97          HD44780_Opts.DisplayFunction |= HD44780_2LINE;
98      }
99
100     /* Try to set 4bit mode */
101     TM_HD44780_Cmd4bit(0x03);
102     HD44780_Delay(45);         // 45
103     //delay(30000);
104     /* Second try */
105     TM_HD44780_Cmd4bit(0x03);
106     HD44780_Delay(45);           //45
107     //delay(30000);
108     /* Third goo! */
109     TM_HD44780_Cmd4bit(0x03);
110     HD44780_Delay(45);            //45
111     //delay(30000);
112     /* Set 4-bit interface */
113     TM_HD44780_Cmd4bit(0x02);
114     HD44780_Delay(30);  //30
115     //delay(30000);
116     /* Set # lines, font size, etc. */
117     TM_HD44780_Cmd(HD44780_FUNCTIONSET | HD44780_Opts.DisplayFunction);
118
119     /* Turn the display on with no cursor or blinking default */
120     HD44780_Opts.DisplayControl = HD44780_DISPLAYON;
121     TM_HD44780_DisplayOn();
122
123     /* Clear lcd */
124     TM_HD44780_Clear();
125
126     /*CLEAR LCD */
127     TM_HD44780_CLEAR();
```

```
127     TM_HD44780_CLEAR();
128
129     /* Default font directions */
130     HD44780_Opts.DisplayMode = HD44780_ENTRYLEFT | HD44780_ENTRYSHIFTDECREMENT;
131     TM_HD44780_Cmd(HD44780_ENTRYMODESET | HD44780_Opts.DisplayMode);
132
133     /* Delay */
134     HD44780_Delay(30);    //  here 30ms
135       //delay(30000);
136
137     //HD44780_delay(1000);     //  8-2-23
138
139 }
140
141 void TM_HD44780_CLEAR(void){                                    //  8-2-23      CLEAI
142      TM_HD44780_Cmd(HD44780_CLEARDISPLAY);
143      delay(1000);              //  here  delay(1000);
144
145 }
146
147
148
149 void TM_HD44780_Clear(void) {                                     // 8-2-23
150     TM_HD44780_Cmd(HD44780_CLEARDISPLAY);
151     HD44780_Delay(300);           //  -------------->>>>>>  here 300ms no 0.2
152     //delay(32000);
153 }
154
155 void TM_HD44780_Puts(uint8_t x, uint8_t y, char* str) {
156     TM_HD44780_CursorSet(x, y);
157     while (*str) {
158         if (HD44780_Opts.currentX >= HD44780_Opts.Cols) {
159             HD44780_Opts.currentX = 0;
160             HD44780_Opts.currentY++;
161             TM_HD44780_CursorSet(HD44780_Opts.currentX, HD44780_Opts.currentY);
162         }
163         if (*str == '\n') {
164             HD44780_Opts.currentY++;
165             TM_HD44780_CursorSet(HD44780_Opts.currentX, HD44780_Opts.currentY);
166         } else if (*str == '\r') {
167             TM_HD44780_CursorSet(0, HD44780_Opts.currentY);
168         } else {
169             TM_HD44780_Data(*str);
```

```
169                 TM_HD44780_Data(*str);
170                 HD44780_Opts.currentX++;
171             }
172             str++;
173         }
174 }
175
176 void delay(uint16_t delay)
177 {
178 __HAL_TIM_SET_COUNTER(&htim15,0);
179 while(__HAL_TIM_GET_COUNTER(&htim15) < delay);
180 }
181
182
183 void TM_HD44780_DisplayOn(void) {
184     HD44780_Opts.DisplayControl |= HD44780_DISPLAYON;
185     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
186 }
187
188 void TM_HD44780_DisplayOff(void) {
189     HD44780_Opts.DisplayControl &= ~HD44780_DISPLAYON;
190     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
191 }
192
193
194
195 void TM_HD44780_BlinkOn(void) {
196     HD44780_Opts.DisplayControl |= HD44780_BLINKON;
197     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
198 }
199
200
201
202 void TM_HD44780_BlinkOff(void) {
203     HD44780_Opts.DisplayControl &= ~HD44780_BLINKON;
204     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
205 }
206
207 void TM_HD44780_CursorOn(void) {
208     HD44780_Opts.DisplayControl |= HD44780_CURSORON;
209     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
210 }
211
```

```
211
212 void TM_HD44780_CursorOff(void) {
213     HD44780_Opts.DisplayControl &= ~HD44780_CURSORON;
214     TM_HD44780_Cmd(HD44780_DISPLAYCONTROL | HD44780_Opts.DisplayControl);
215 }
216
217 void TM_HD44780_ScrollLeft(void) {
218     TM_HD44780_Cmd(HD44780_CURSORSHIFT | HD44780_DISPLAYMOVE | HD44780_MOVELEFT);
219 }
220
221 void TM_HD44780_ScrollRight(void) {
222     TM_HD44780_Cmd(HD44780_CURSORSHIFT | HD44780_DISPLAYMOVE | HD44780_MOVERIGHT);
223 }
224
225 void TM_HD44780_CreateChar(uint8_t location, uint8_t *data) {
226     uint8_t i;
227     /* We have 8 locations available for custom characters */
228     location &= 0x07;
229     TM_HD44780_Cmd(HD44780_SETCGRAMADDR | (location << 3));
230
231     for (i = 0; i < 8; i++) {
232         TM_HD44780_Data(data[i]);
233     }
234 }
235
236 void TM_HD44780_PutCustom(uint8_t x, uint8_t y, uint8_t location) {
237     TM_HD44780_CursorSet(x, y);
238     TM_HD44780_Data(location);
239 }
240
241 /* Private functions */
242 static void TM_HD44780_Cmd(uint8_t cmd) {
243     /* Command mode */
244     HD44780_RS_LOW;
245
246     /* High nibble */
247     TM_HD44780_Cmd4bit(cmd >> 4);
248     /* Low nibble */
249     TM_HD44780_Cmd4bit(cmd & 0x0F);
250 }
251
252 static void TM_HD44780_Data(uint8_t data) {
253     /* Data mode */
```

main.c | tm_stm32f4_hd44780.c

```
253     /* Data mode */
254     HD44780_RS_HIGH;
255
256     /* High nibble */
257     TM_HD44780_Cmd4bit(data >> 4);
258     /* Low nibble */
259     TM_HD44780_Cmd4bit(data & 0x0F);
260 }
261
262 static void TM_HD44780_Cmd4bit(uint8_t cmd) {
263     /* Set output port */
264     HAL_GPIO_WritePin(HD44780_D7_PORT, HD44780_D7_PIN, (cmd & 0x08));
265     HAL_GPIO_WritePin(HD44780_D6_PORT, HD44780_D6_PIN, (cmd & 0x04));
266     HAL_GPIO_WritePin(HD44780_D5_PORT, HD44780_D5_PIN, (cmd & 0x02));
267     HAL_GPIO_WritePin(HD44780_D4_PORT, HD44780_D4_PIN, (cmd & 0x01));
268     HD44780_E_BLINK;
269 }
270
271 static void TM_HD44780_CursorSet(uint8_t col, uint8_t row) {
272     uint8_t row_offsets[] = {0x00, 0x40, 0x14, 0x54};
273
274     /* Go to beginning */
275     if (row >= HD44780_Opts.Rows) {
276         row = 0;
277     }
278
279     /* Set current column and row */
280     HD44780_Opts.currentX = col;
281     HD44780_Opts.currentY = row;
282
283     /* Set location address */
284     TM_HD44780_Cmd(HD44780_SETDDRAMADDR | (col + row_offsets[row]));
285 }
286
287 static void TM_HD44780_InitPins(void) {
288     /* Init all pins
289     TM_GPIO_Init(HD44780_RS_PORT, HD44780_RS_PIN, TM_GPIO_Mode_OUT, TM_
290     TM_GPIO_Init(HD44780_E_PORT, HD44780_E_PIN, TM_GPIO_Mode_OUT, TM_GP
291     TM_GPIO_Init(HD44780_D4_PORT, HD44780_D4_PIN, TM_GPIO_Mode_OUT, TM_
292     TM_GPIO_Init(HD44780_D5_PORT, HD44780_D5_PIN, TM_GPIO_Mode_OUT, TM_
293     TM_GPIO_Init(HD44780_D6_PORT, HD44780_D6_PIN, TM_GPIO_Mode_OUT, TM_
294     TM_GPIO_Init(HD44780_D7_PORT, HD44780_D7_PIN, TM_GPIO_Mode_OUT, TM_
295
```

main.c | tm_stm32f4_hd44780.c

```
262 static void TM_HD44780_Cmd4bit(uint8_t cmd) {
263     /* Set output port */
264     HAL_GPIO_WritePin(HD44780_D7_PORT, HD44780_D7_PIN, (cmd & 0x08));
265     HAL_GPIO_WritePin(HD44780_D6_PORT, HD44780_D6_PIN, (cmd & 0x04));
266     HAL_GPIO_WritePin(HD44780_D5_PORT, HD44780_D5_PIN, (cmd & 0x02));
267     HAL_GPIO_WritePin(HD44780_D4_PORT, HD44780_D4_PIN, (cmd & 0x01));
268     HD44780_E_BLINK;
269 }
270
271 static void TM_HD44780_CursorSet(uint8_t col, uint8_t row) {
272     uint8_t row_offsets[] = {0x00, 0x40, 0x14, 0x54};
273
274     /* Go to beginning */
275     if (row >= HD44780_Opts.Rows) {
276         row = 0;
277     }
278
279     /* Set current column and row */
280     HD44780_Opts.currentX = col;
281     HD44780_Opts.currentY = row;
282
283     /* Set locati
284     TM_HD44780_Cmd(HD44780_SETDDRAMADDR | (col + row_offsets[row]));
285 }
286
287 static void TM_HD44780_InitPins(void) {
288     /* Init all pins
289     TM_GPIO_Init(HD44780_RS_PORT, HD44780_RS_PIN, TM_GPIO_Mode_OUT, TM_G
290     TM_GPIO_Init(HD44780_E_PORT, HD44780_E_PIN, TM_GPIO_Mode_OUT, TM_GPI
291     TM_GPIO_Init(HD44780_D4_PORT, HD44780_D4_PIN, TM_GPIO_Mode_OUT, TM_G
292     TM_GPIO_Init(HD44780_D5_PORT, HD44780_D5_PIN, TM_GPIO_Mode_OUT, TM_G
293     TM_GPIO_Init(HD44780_D6_PORT, HD44780_D6_PIN, TM_GPIO_Mode_OUT, TM_G
294     TM_GPIO_Init(HD44780_D7_PORT, HD44780_D7_PIN, TM_GPIO_Mode_OUT, TM_G
295
296     /* Set pins low */
297     HAL_GPIO_WritePin(HD44780_RS_PORT, HD44780_RS_PIN,GPIO_PIN_RESET);
298     HAL_GPIO_WritePin(HD44780_E_PORT, HD44780_E_PIN,GPIO_PIN_RESET);
299     HAL_GPIO_WritePin(HD44780_D4_PORT, HD44780_D4_PIN,GPIO_PIN_RESET);
300     HAL_GPIO_WritePin(HD44780_D5_PORT, HD44780_D5_PIN,GPIO_PIN_RESET);
301     HAL_GPIO_WritePin(HD44780_D6_PORT, HD44780_D6_PIN,GPIO_PIN_RESET);
302     HAL_GPIO_WritePin(HD44780_D7_PORT, HD44780_D7_PIN,GPIO_PIN_RESET);
303 }
304
```

uint8_t currentY;
Press 'F2' for focus