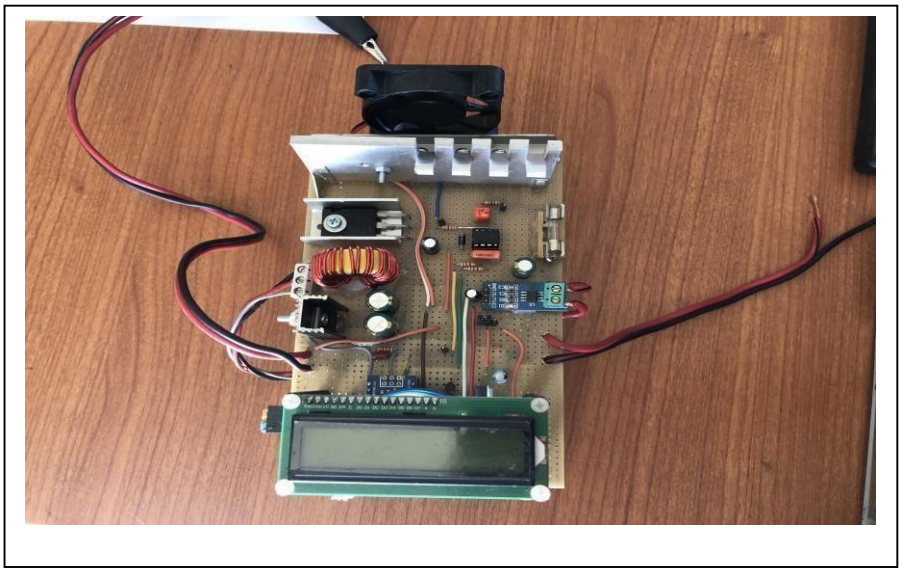
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

(ΑΥΤΟΝΟΜΗ ΜΙΚΡΗ ΦΩΤΟΒΟΛΤΑΪΚΗ ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΑΠΛΟΥ ΦΟΡΤΙΣΤΗ ΜΠΑΤΑΡΙΑΣ ΜΕ ΑΛΓΟΡΙΘΜΟ MPPT)



Φοιτητής: (Κοσμαδάκης Χαράλαμπος)
ΑΜ: (48346733)

Επιβλέπων Καθηγητής

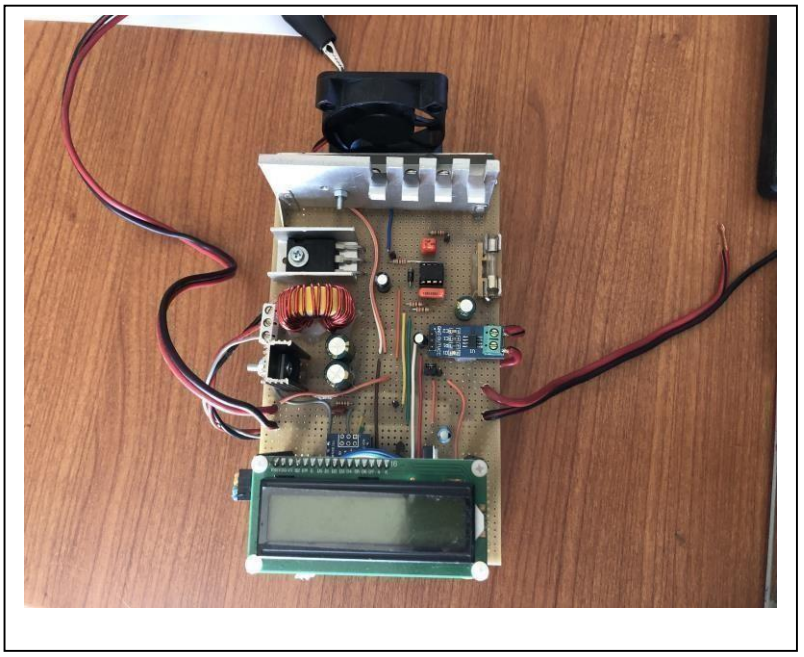
Βόκας Γεώργιος
Καθηγητής ΠΑΔΑ

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, (ΙΟΥΛΙΟΣ) (2022)

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

(CONSTRUCTION OF A SMALL AUTONOMOUS PHOTOVOLTAIC SYSTEM CHARGING A BATTERY WITH MPPTALGORITHM)



Student: (Kosmadakis Charalampos)
Registration Number: (48346733)

Supervisor

(Georgios Vokas)
(PROFESSOR)

ATHENS-EGALEO, (JULY) (2022)

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

ΒΟΚΑΣ ΓΕΩΡΓΙΟΣ Καθηγητής	ΠΑΠΑΓΕΩΡΓΑΣ ΠΑΝΑΓΙΩΤΗΣ Καθηγητής	ΠΥΡΟΜΑΛΗΣ ΔΗΜΗΤΡΙΟΣ Επ. Καθηγητής
Georgios Vokas Digitally signed by Georgios Vokas Date: 2022.07.26 22:53:32 +03'00' (Υπογραφή)	Digitally signed by Panagiotis Papageorgas Date: 2022.07.26 13:23:38 +03'00' (Υπογραφή)	Dimitrios Pyromalis Digitally signed by Dimitrios Pyromalis Date: 2022.07.26 14:11:53 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η ΚΟΣΜΑΔΑΚΗΣ ΧΑΡΑΛΑΜΠΟΣ
του ΜΑΡΙΟΥ, με αριθμό μητρώου 48346733 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.
Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Ο Δηλών
ΚΟΣΜΑΔΑΚΗΣ ΧΑΡΑΛΑΜΠΟΣ

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω την οικογένεια μου , η οποία με στήριξε σε όλη τη διάρκεια των σπουδών μου.

Επίσης θα ήθελα να ευχαριστήσω από καρδιάς, τον καθηγητή κύριο Γεώργιο Βόκα που με βοήθησε με τις γνώσεις του αλλά και με την υπομονή σε ό,τι πρόβλημα συνάντησα σε όλη την πορεία της διπλωματικής μου εργασίας, ώστε να καταφέρω να υλοποιήσω το πείραμα το οποίο ανέλαβα.

Ακόμη θα ήθελα να βοηθήσω τον κύριο Κορακιανίτη Νικόλαο , ο οποίος μου παρείχε ό,τι χρειάστηκα από το εργαστήριο της σχολής , αλλά και με βοήθησε με τις γνώσεις του.

Επιπροσθέτως θα ήθελα να ευχαριστήσω την τριμελής επιτροπή για τον χρόνο που αφιέρωσε για την εργασία μου.

Τέλος θα ήθελα να ευχαριστήσω την σχολή και το εργατικό της προσωπικό, αλλά κυρίως όλους τους καθηγητές που κατάφεραν να με κάνουν να αγαπήσω το αντικείμενο το οποίο σπούδασα.

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την υλοποίηση αλγορίθμων mppt (κυρίως P&O), για χρήση σε φωτοβολταϊκά συστήματα και ανεμογεννήτριες. Με τους αλγόριθμους αυτούς επιτυγχάνεται μέγιστη μεταφορά ισχύος από την πηγή στο φορτίο. Σε αυτό το πεδίο έχουν αναπτυχθεί αρκετοί αλγόριθμοι, κάθε ένας από τους οποίους έχει ορισμένα πλεονεκτήματα και μειονεκτήματα. Η προτεινόμενη υλοποίηση λαμβάνει υπ' όψη της το χαμηλό κόστος, την ευκολία ανεύρεσης υλικών και την ευκολία υλοποίησης. Ακολουθούν συγκρίσεις κλασσικού αλγορίθμου και P&O.

Λέξεις – κλειδιά

Mppt, φορτιστής μπαταρίας, κατασκευή, μικροεπεξεργαστής, αλγόριθμος P&O, ανανεώσιμες πηγές ενέργειας.

Abstract

The present thesis concerns the implementation of mppt algorithms (mainly P&O), in photovoltaic systems and wind generators. By use of these algorithms, maximum power transference from source to load is possible. In this field a lot of algorithms have been developed, each of which presents both advantages and disadvantages. The present thesis takes into consideration low cost as well ease of implementation. Comparisons between classic (pwm) algorithm and mppt algorithms follow.

Keywords

Mppt, battery charger, contracture, Arduino, Algorithm P&O, renewable energy sources, battery.

Περιεχόμενα

Κατάλογος Εικόνων

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια περιληπτική αναφορά στην κατασκευή του πειράματος και στους αλγόριθμους που εξετάζονται.

Αντικείμενο της διπλωματικής εργασίας

Το κύριο θέμα της διπλωματικής εργασίας είναι η κατασκευή ενός φορτιστή μπαταρίας που θα τροφοδοτείται από ένα φωτοβολταϊκό, ο οποίος θα λειτουργεί με έναν αλγόριθμο (P&O) και θα φορτίζει το φορτίο παρέχοντάς του την μέγιστη ισχύ. Τα φωτοβολταϊκά είναι μια ανανεώσιμη πηγή ενέργειας και δεδομένου της τεράστιας ενεργειακής κρίσης που διανύουμε ,έχει μεγάλο ενδιαφέρον η κατασκευή ενός φορτιστή που θα αποδίδει την μέγιστη ενέργεια έχοντας τις μικρότερες δυνατές απώλειες.

Σκοπός και στόχοι

Σκοπός της διπλωματικής εργασίας είναι η ορθή υλοποίηση του φορτιστή καθώς και η διερεύνηση του αλγόριθμου (P&O) ώστε να παρέχεται στο φορτίο η μέγιστη ισχύς ανάλογα με τις καιρικές συνθήκες σκοπός και οι στόχοι της εργασίας, αναλυόμενοι σε επιμέρους ερωτήματα προς διερεύνηση.

Μεθοδολογία

Για την υλοποίηση του πειράματος ακολουθήθηκαν τα εξής πειραματικά στάδια:

1. Μελέτη για τον σχεδιασμό του πειράματος.
2. Εύρεση/αγορά των κατάλληλων υλικών που χρησιμοποιήθηκαν.
3. Κατασκευή των πλακετών.
4. Προγραμματισμούς του μικροεπεξεργαστή.
5. Έλεγχος ορθής λειτουργίας.

Καινοτομία

Η καινοτομία του πειράματος που υλοποιήθηκε είναι το ότι είναι low budget δηλαδή μπορεί να κατασκευαστεί χωρίς να δαπανηθούν μεγάλα χρηματικά ποσά.

Δομή

Δομή της διπλωματικής εργασίας (οργάνωση σε κεφάλαια και υποκεφάλαια).

ΚΕΦΑΛΑΙΟ 1ο : (ΕΙΣΑΓΩΓΗ)

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της πτυχιακής εργασίας και γίνεται μια περιληπτική αναφορά στην κατασκευή του φορτιστή και στους αλγόριθμους που εξετάζονται.

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η υλοποίηση ενός φορτιστή συσσωρευτών με φωτοβολταϊκά ή ανεμογεννήτριες για έλεγχο εγκατάστασης και σύγκριση απόδοσης αλγορίθμων που τρέχουν στον φορτιστή για την επίτευξη μέγιστης ενέργειας στο φορτίο.

.

1.2 Περιληπτική αναφορά

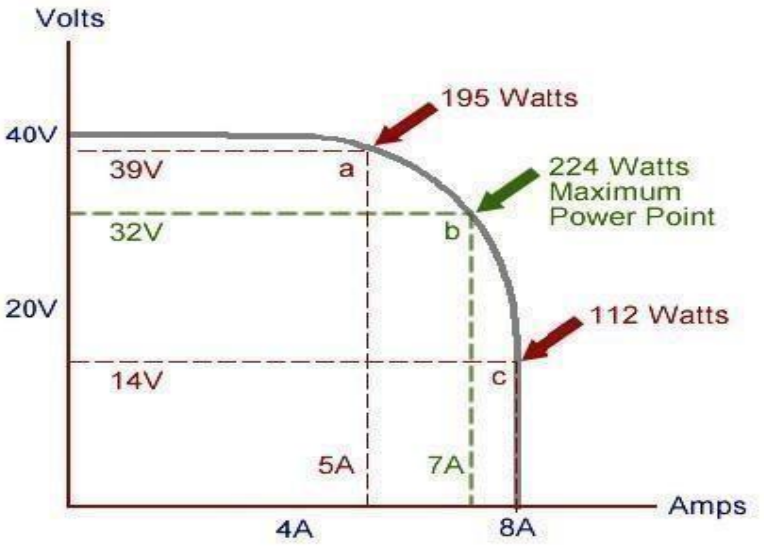
Ως γνωστό, οι ανανεώσιμες πηγές ενέργειας είναι σημαντικές για την οικολογία του πλανήτη. Τις τελευταίες δεκαετίες έχουν γνωρίσει τεράστια ανάπτυξη η οποία και συνεχίζεται. Ένα από τα βασικά προβλήματα που υπάρχει για την εκμετάλλευση της ενέργειας που παράγεται από τις ανανεώσιμες πηγές είναι ο τρόπος μεταφοράς αλλά και η αποθήκευση αυτής. Χρησιμοποιώντας τον όρο μεταφορά ενέργειας εννοούμε τον τρόπο που θα μεταφερθεί η ενέργεια από το φωτοβολταϊκό στις μπαταρίες ή ακόμα και στο δίκτυο. Πρέπει η μεταφορά αυτή να γίνεται με τέτοιο τρόπο ώστε η απόδοση να είναι μέγιστη. Παλαιότερα ήταν σε χρήση απλοί υποβιβαστές τάσης , αλλά η ενέργεια που κατανάλωναν έριχνε αρκετά την απόδοση του συνολικού συστήματος με αποτέλεσμα να μη ήταν τόσο κερδοφόρο. Σήμερα πλέον χρησιμοποιείται τεχνολογία dc-dc μετατροπέων. Ουσιαστικά μιλάμε για μετασχηματιστές ισχύος οι οποίοι ανεβάζουν αρκετά την απόδοση. Ένα μέρος της συγκεκριμένης εργασίας είναι η κατασκευή ενός τέτοιου μετατροπέα χρησιμοποιώντας μικροεπεξεργαστή προσπαθώντας να επιτύχω αρκετά μεγάλη απόδοση, το οποίο επιτυγχάνεται αφού η τελική απόδοση του πειράματος ξεπερνάει το 90%. Στον μικροεπεξεργαστή τεράστιο ενδιαφέρον υπάρχει στην διερεύνηση του αλγόριθμου που θα τρέξει. Χρειάζεται ανάλογα με τις δεδομένες καιρικές συνθήκες να παίρνει από την πηγή την μέγιστη δυνατή ισχύ, ώστε να εκμεταλλεύεται όλη την ενέργεια που δίνεται.

2 ΚΕΦΑΛΑΙΟ 2ο : (Σημείο προσαρμογής μέγιστης ισχύος) (Maximum power point tracking (MPPT))

Σε αυτό το κεφάλαιο περιγράφονται ο αλγόριθμος υλοποίησης αλλά και η αρχή λειτουργίας του MPPT.

2.1 Τι είναι το mppt

Το mppt είναι μία τεχνική που χρησιμοποιούν οι μετατροπείς συνεχούς σε συνεχή τάση (DC to DC converters) και οι ρυθμιστές φόρτισης, ώστε να βελτιστοποιούν την απόδοση ισχύος από την πηγή , δηλαδή το φωτοβολταϊκό ή την ανεμογεννήτρια.

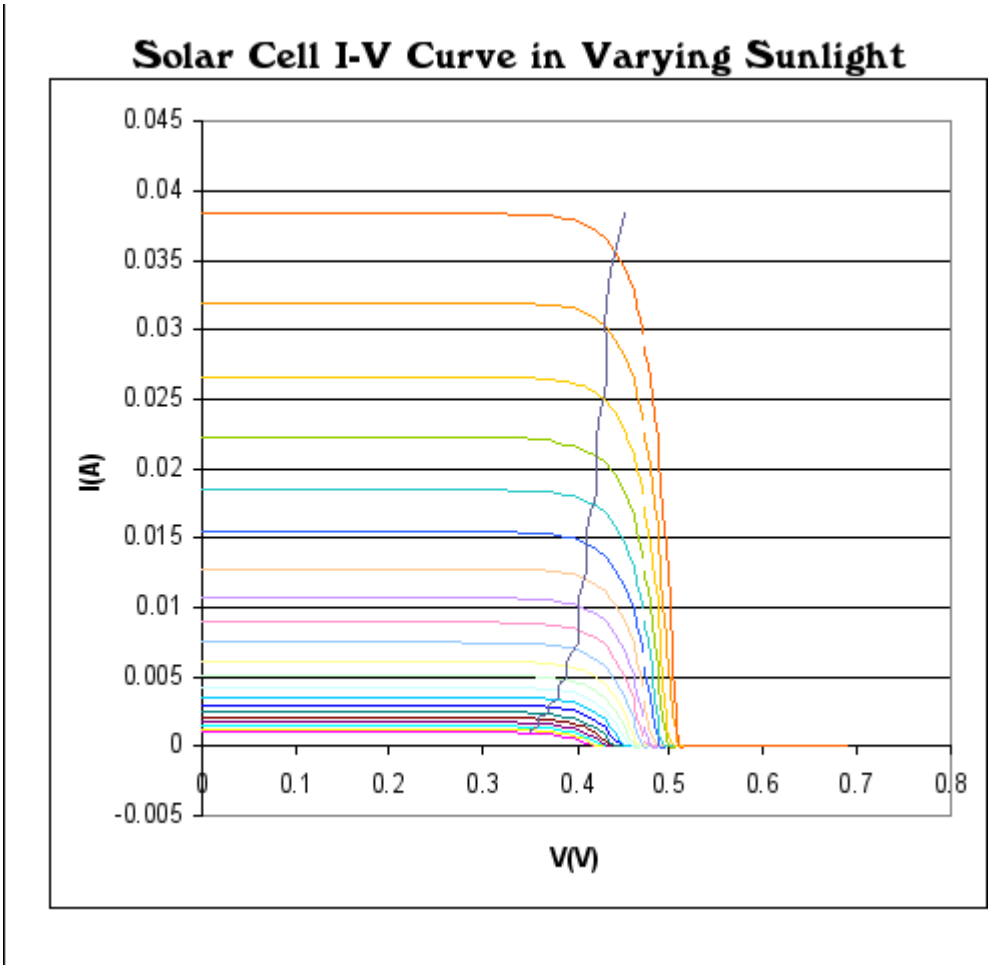


E-2.1 -ΣΗΜΕΙΟ ΛΕΙΤΟΥΡΓΙΑΣ MPPT

Η αρχή λειτουργίας του mppt είναι ουσιαστικά η προσαρμογή της αντίστασης του φορτίου με την πηγή. Πετυχαίνουμε την μέγιστη απόδοση όταν η αντίσταση του φορτίου είναι ίση με την σύνθετη αντίσταση της πηγής.

Για παράδειγμα:
Έστω ότι έχω ένα φωτοβολταϊκό πάνελ των 50W, το οποίο αποδίδει σύμφωνα με τα χαρακτηριστικά του 2,083A στα 24Volts (24X 2.083 =50). Θεωρώντας το φωτοβολταϊκό μια πηγή ρεύματος, αν του βάλω μια μπαταρία ονομαστικής τάσης 12V, αυτό θα αποδώσει 24.996 W (12X2.083=24.996). Δηλαδή οι απώλειες είναι ίσες 25 watts.

Σύμφωνα με την εικόνα 2.1 στο σημείο b έχουμε μέγιστη ισχύς(απόδοση) , η οποία είναι 224W(32V X 7A). Το σημείο b βρίσκεται στο γόνατο της καμπύλης. Σε οποιοιδήποτε άλλο σημείο το φωτοβολταϊκό μας πάνελ έχει μικρότερη απόδοση.

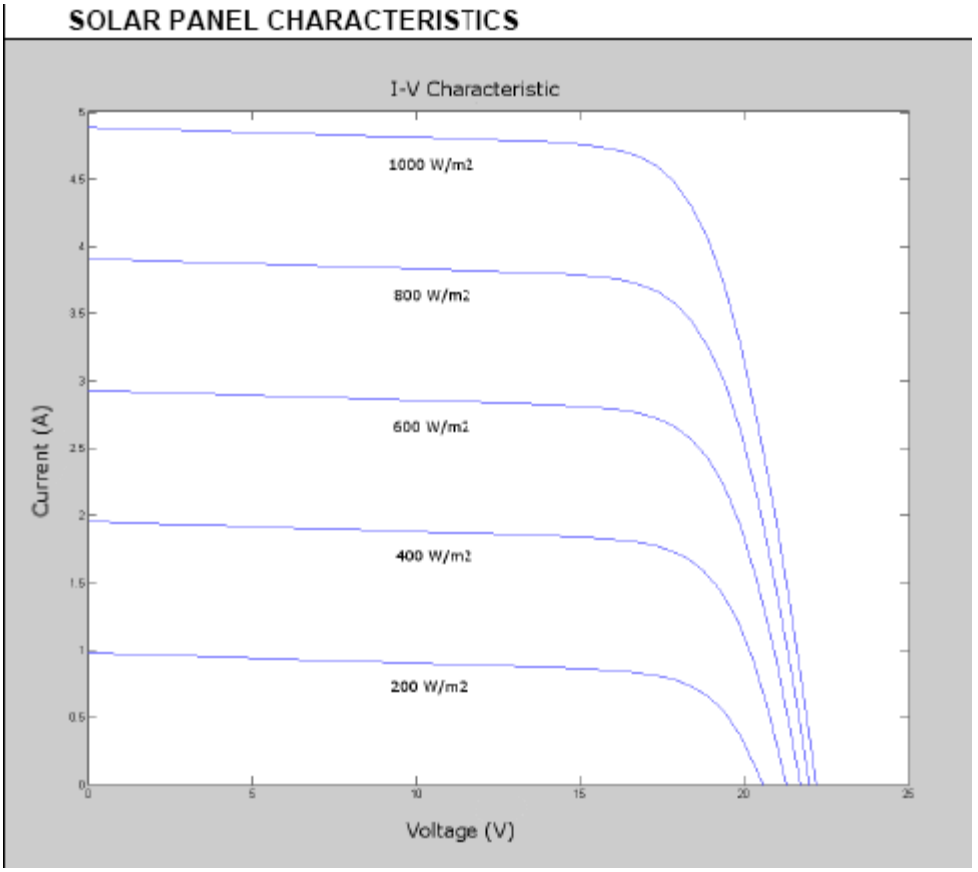


E-2.2 -*Μεταβολή του mppt ανάλογα με την ένταση της προσπίπτουσας ηλιακής ακτινοβολίας*

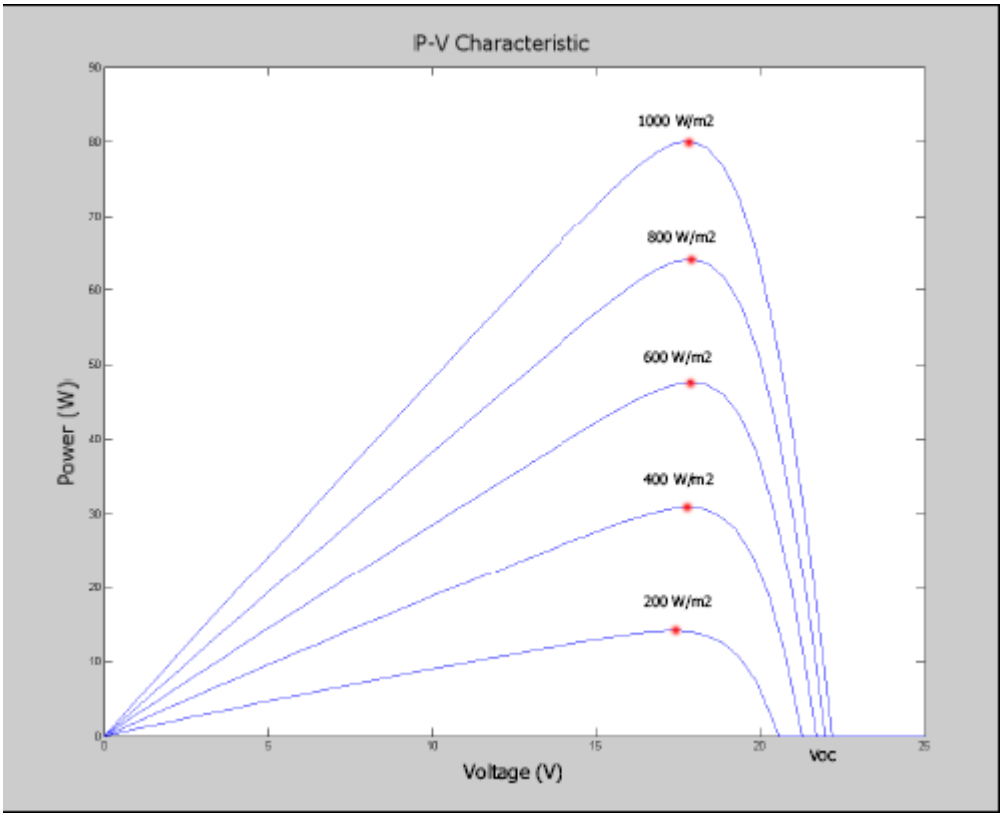
Χρησιμοποιώντας εδώ τους αλγόριθμους mppt προσπαθώ να αυξήσω την απόδοση ενέργειας , κρατώντας το σημείο λειτουργίας όσο πιο κοντά γίνεται στο σημείο b της εικόνας **E-2.1.** Ο μικροεπεξεργαστής παράγει κατάλληλους παλμούς (PWM), καθορίζοντας τον λόγο τάσης εξόδου προς τάση εισόδου. Γενικά ισχύει:

$$\frac{Vout}{Vin} = \frac{1}{1-D} \quad \text{(T2.1.1)}$$

Vout = τάση εξόδου
Vin = τάση εισόδου
D = duty-cycle του PWM σήματος.
Με τον όρο duty cycle εννοώ ουσιαστικά την συχνότητα χρήσης



E2.3 *– Καμπύλη I-V φωτοβολταϊκού για διάφορες εντάσεις προσπίπτουσας ηλιακής Ακτινοβολίας [2]*



E2.4 *– Σημεία λειτουργίας mppt για διάφορες εντάσεις της προσπίπτουσας ηλιακής*

ακτινοβολίας και καμπύλη P-V φωτοβολταϊκού [2]

2.2 Αλγόριθμος απλός και αλγόριθμος mppt

Ο απλός ή αλλιώς κλασικός αλγόριθμος αυξάνει τους παλμούς που δίνει ο μικροεπεξεργαστής (pwm) μέχρι η μπαταρία να φορτίσει ολοκληρωτικά. Έπειτα το Pwm σταδιακά χαμηλώνει, με σκοπό να αποφύγουμε την υπερφόρτιση της μπαταρίας. Ο αλγόριθμος ελέγχει συνεχώς την τάση της μπαταρίας με αποτέλεσμα αν μειωθεί , θα αυξήσει άμεσα το pwm ακλουθώντας την διαδικασία από την αρχή. Στο σημείο που θα αυξομειώνεται συνεχώς το pwm με μικρό βήμα ώστε η τάση της μπαταρίας να παραμένει στην επιθυμητή τάση ονομάζεται float mode . Ο κλασσικός αλγόριθμος προσπαθεί να φτάσει στην επιθυμητή τάση αλλά δεν ενδιαφέρεται για το αν αυτή είναι η μέγιστη ισχύς που μπορεί να αποδώσει το φωτοβολταϊκό, επομένως χρησιμοποιώντας αυτόν τον τρόπο έχω σίγουρα απώλειες ενέργειας. Ουσιαστικά πρέπει να βελτιώσουμε την απόδοση (n) , πλησιάζοντας την όσο πιο κοντά γίνεται στην μονάδα. Αυτό που μας ενδιαφέρει να βελτιώσουμε είναι το mpp:

$$n = \frac{Pout}{P\max}$$

Pout = ισχύς εξόδου
Pmax = μέγιστη ισχύς που αποδίδει το φωτοβολταϊκό.

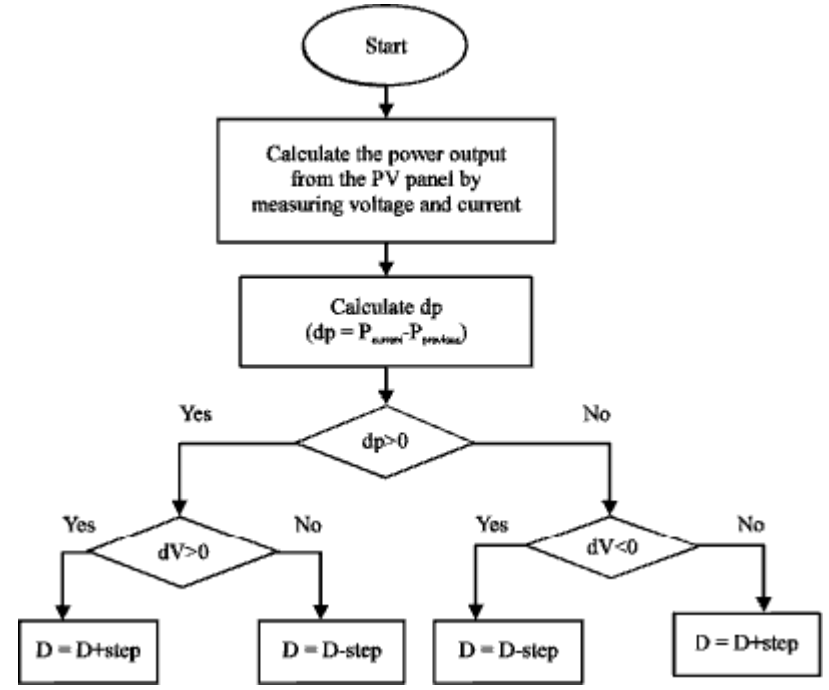
Για να επιτευχθεί η εύρεση του mpp υπάρχουν οι αλγόριθμοι Hill-climbing. Ονομάζονται έτσι γιατί προσπαθεί με βήματα επί της καμπύλης I=f(V) του φωτοβολταϊκού, να βρει το mpp. Υπάρχουν αρκετοί τέτοιοι αλγόριθμοι όπως:

- Perturb and Observe (P&O) – Διαταραχή και παρατήρηση
- $\frac{dP}{dV}$ Feedback Control – Έλεγχος ανάδρασης
- Incremental Conductance – Αυξητική αγωγιμότητα
- Στην διπλωματική αυτή εργασία θα ασχοληθώ κυρίως με τον 1ο αλγοριθμο Perturb and Observe (P&O) – Διαταραχή και παρατήρηση

2.2.1 Perturb and Observe (P&O) – Διαταραχή και παρατήρηση

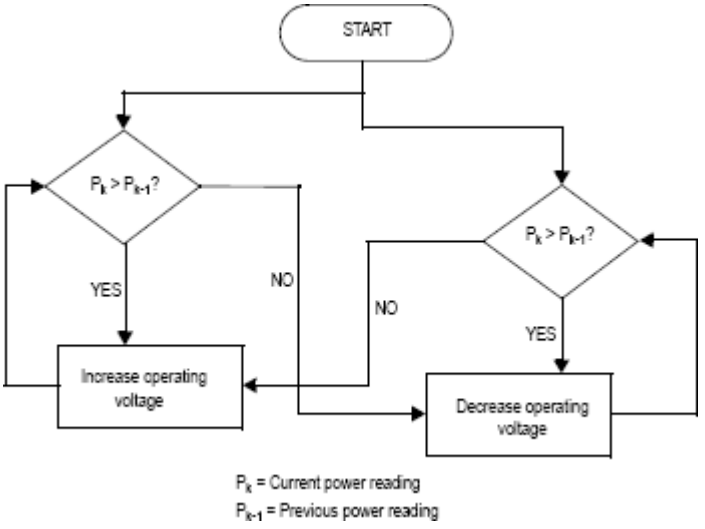
Συμφωνα με τους Zegaoui, A., Aillerie, M., Petit, P., Sawicki, J. P., Jaafar, A., Salame, C., & Charles, J. P. στη δημοσίευση τους Comparison of Two Common Maximum Power Point Trackers by Simulating of PV Generators. Energy Procedia [15], ο P&O αλγόριθμος αλλάζει το duty-cycle του PWM που ελέγχει τον dc-dc converter, έτσι ώστε κάνοντας βήματα πάνω στην καμπύλη χαρακτηριστικών του φωτοβολταϊκού να επιτυγχάνεται το mpp. Με αυτήν την αλλαγή βρίσκουμε ένα νέο σημείο λειτουργίας με διαφορετική ισχύ εξόδου. Σε περίπτωση που η τρέχουσα ισχύς εξόδου είναι μεγαλύτερη από την αμέσως προηγούμενη, το νέο σημείο λειτουργίας καθίσταται τρέχον και επαναλαμβάνεται διαταραχή. Σε περίπτωση που η τρέχουσα ισχύς εξόδου είναι

μικρότερη από την αμέσως προηγούμενη, το τρέχον σημείο λειτουργίαςρυθμίζεται σε χαμηλότερο ή υψηλότερο σημείο το οποίο εξαρτάται από την κατεύθυνση του προηγούμενου βήματος. Ουσιαστικά μετράμε την ισχύ εξόδου του φωτοβολταϊκού. Συγκρίνουμε την τρέχουσα ισχύ με την προηγούμενη και στη συνέχεια την τρέχουσα τάση με την προηγούμενη. Ανάλογα με τα αποτελέσματα μεταβάλλουμε το duty-cycle του PWM και επαναλαμβάνουμε τον βρόγχο συνέχεια. Τα παραπάνω απεικονίζονται ξεκάθαρα στοπαρακάτω διάγραμμα ροής:



E-2.5 *– Διάγραμμα ροής αλγόριθμου P&O – αύξηση του D κατά κάποιο βήμα στην πραγματικότητα είναι αύξηση του PWM κατά κάποιο συγκεκριμένο βήμα*

Στον ιστότοπο της εταιρίας microchip υπάρχει ένα απλούστερος τρόπος, ο οποίος φαίνεται στην εικόνα **E-2.6**.. Ο τρόπος λειτουργίας γίνεται εύκολα κατανοητός αν συνδυαστούν οι εικόνες **E-2.4** και **E-2.5**. Από την **E-2.4** βλέπουμε ότι μείωση της τάσης στην δεξιά πλευρά του mppt, αυξάνει την ισχύ. Επίσης βλέπουμε ότι με αύξηση της τάσης αριστερά του σημείου mmpt έχουμε αύξηση ισχύος. Αυξάνοντας την τάση αυξάνουμε και το duty-cycle του PWM. Ας υποθέσουμε ότι αφού πραγματοποιήσουμε μια αύξηση της τάσης, ο αλγόριθμος συγκρίνει την τρέχουσα ισχύ με την προηγούμενη. Αν η ισχύς έχει αυξηθεί επαναλαμβάνει το ίδιο (στην ίδια κατεύθυνση δηλαδή πάλι αύξηση και μετά σύγκριση). Αλλιώς αλλάζει κατεύθυνση (μειώνοντας την τάση) και συνεχίζει ακολουθώντας αυτή την κατεύθυνση.[15] , [16]



E.2.6 *– Διάγραμμα ροής αλγόριθμου P&O – απλούστευση της microchip*

όπου
Pk = τρέχουσα ισχύς
Pk-1 = προηγούμενη ισχύ

Αφού προσεγγίσει το σημείο MPP ο αλγόριθμος(σχεδόν ποτέ δεν το φτάνει ακριβώς) προκαλεί αλλαγές στο pwm και στην ουσία κάνει αυξομειώσεις pwm ώστε να είναι όσο πιο κοντά γίνεται στο MPP. Όσο πιο μεγάλο είναι το βήμα , τόσο πιο μεγάλες αυξομειώσεις κάνει ο αλγόριθμος και πλησιάζει πολύ γρήγορα το MPP αλλά δεν καταφέρνει ποτέ να πλησιάσει αρκετά κοντά , ενώ με μικρό βήμα αργεί να βρει το MPP , αλλά όταν το βρει , το προσεγγίζει με πολύ μεγαλύτερη ακρίβεια. Συνοπτικά αν έχουμε συχνές μεταβολές στην ηλιοφάνεια συνιστάται μεγάλο βήμα αλλιώς μικρό. Βάζοντας μια ενδιάμεση τιμή καλύπτω και τις δύο περιπτώσεις.

2.2.2 $\frac{dP}{dV}$ Feedback Control – Έλεγχος ανάδρασης

$\frac{dP}{dV}$
Ο βασίζεται στην παράγωγο της ισχύος (P) ως προς την τάση (V). Αρχικά υπολογίζεται το γινόμενο της τασης με το ρεύμα του φωτοβολταϊκού, δηλαδή η ισχύς (P=V*I) και στη συνεχεια υπολογίζεται η παράγωγος της ισχύος ως προς την τάση V. Στο σημείο μεγιστης ισχυος (mpp) η παράγωγος μηδενίζεται. Στον μηδενμισμό αυτό βασίζεται ο αλγόριθμος feedback control, όπως μπορούμε να δουμε και στις παρακάτω εξισώσεις.

1. $\frac{dP}{dV} = 0$ στο mpp
2. $\frac{dP}{dV} > 0$ αριστερά του σημείου mpp
3. $\frac{dP}{dV} < 0$ δεξιά του σημείου mpp

Ρυθμιζοντας καταλληλα το duty cycle του pwm συμφωνα με τις παραπάνω σχέσεις βρίσκεται το σημείο mpp. Μια μεταβολή στην ηλιοφάνεια αλλάζει το σημείο μέγιστης ισχύος (mpp), επομένως η παράγωγος δεν είναι μηδενική και ξεκινούν διαταραχές μέχρι να ξαναφθάσουμε στο νέο σημείο μέγιστης ισχύος(mpp), χωρίς να υπάρχουν ταλαντώσεις. Και εδώ το βήμα είναι καθοριστικό και ισχύει ότι και στον αλγόριθμο P&O με την διαφορά ότι αν το βήμα είναι μικρό ουσιαστικά δεν έχουμε ταλαντώσεις. Ο αλγόριθμος $\frac{dP}{dV}$ είναι πιο δύσκολος στην υλοποίησή του από τον P&O.

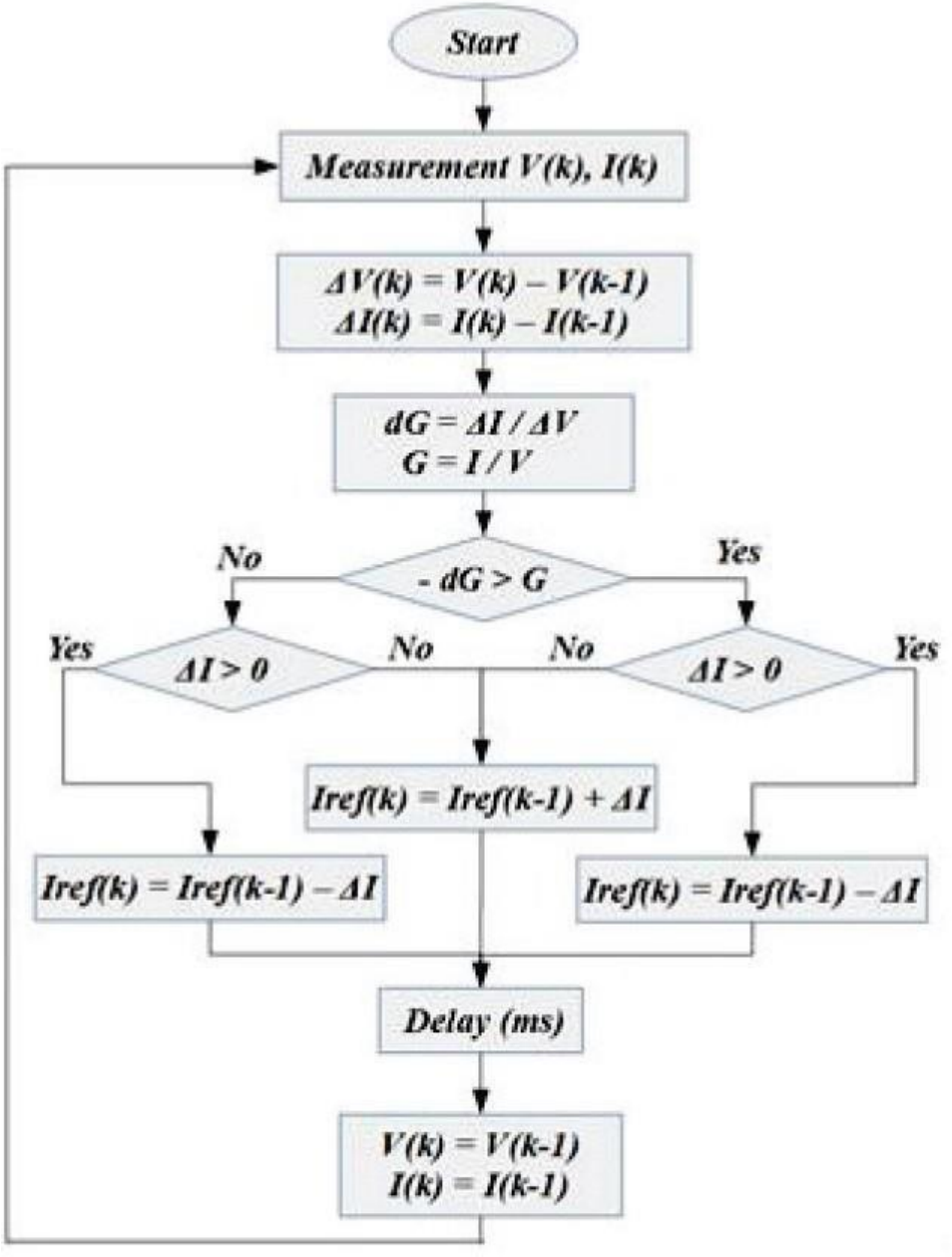
2.2.3 Incremental Conductance (InCond) – Αυξητική αγωγιμότητα

Ο αλγόριθμος InCond είναι μια εξελιγμένη μορφή του $\frac{dP}{dV}$. Αναλυτικά έχουμε:

$$P = VI \Rightarrow \frac{dP}{dV} = V\frac{dI}{dV} + I\frac{dV}{dV} = V\frac{dI}{dV} + I$$

Η παραπάνω εξίσωση σε συνδυασμό με τις εξισώσεις της ενότητας 2.2.2 δίνουν:

1. $\frac{dI}{dV} = -\frac{I}{V}$ στο mpp (σχέση a)
2. $\frac{dI}{dV} > -\frac{I}{V}$ αριστερά του σημείου mpp (σχέση b)
3. $\frac{dI}{dV} < -\frac{I}{V}$ δεξιά του σημείου mpp (σχέση c)



E-2.6.1 *– Διάγραμμα ροής αλγόριθμου InCond*

Στο παραπάνω διάγραμμα ροής ο δείκτης Κ δείχνει την τωρινή μέτρηση ενώ το Κ-1 την προυγούμενη.

Για να υλοποιηθεί αποτελεσματικά ο αλγόριθμος InCond μπορεί να χρησιμοποιηθεί ενας ελεγκτής PI. Κατασκευάζουμε ένα σήμα λάθους σαν είσοδο στον ελεγκτή, σύμφωνα με τον παρακάτω τυπο:

$$e = \frac{dI}{dV} + \frac{I}{V}$$

Ο PI ελεγκτής κάνει το e -> 0 που σημαίνει ότι το σημείο λειτουργίας είναι το σημείο μέγιστης ισχύος. Η απόδοση του αλγορίθμου αυτού μπορεί να ξεπεράσει το 98%.

Χρησιμοποιούμε τον αλγόριθμο αυξητικής αγωγιμότητας και υπολογίζεται το σημείο μέγιστης ισχύος . Από τη σχέση a, σχέση b και σχέση c παρατηρείται ότι γίνεται να υπολογιστεί ακριβώς την θέση λειτουργίας του φωτοβολταϊκού σε αντίθεση με τον αλγόριθμο p & o που απλώς την προσεγγίζει . Επομένως αφού το υπολογίζεται ακριβώς όταν η ηλιοφάνεια μεταβάλλεται γρήγορα, υπολογίζεται ξανά το σημείο αυτό χωρίς να γίνονται ταλαντώσεις γύρω από το σημείο αυτό. Ο αλγόριθμος αυτός είναι λίγο πιο δύσκολος στην υλοποίηση του προγράμματος.

Όλοι οι 3 αυτοί αλγόριθμοι λέγονται Hill-climbing .

2.2.4 Άλλοι αλγόριθμοι

Ονομαστική αναφορά άλλων αλγορίθμων :

1. **Current Sweep Method (Μέθοδος Εναλλαγής Ρεύματος).** Ουσιαστικά βασίζεται στην λύση μιας απλής διαφορικής εξίσωσης [22]
2. **Direct Method (Άμεση Μέθοδος).** Από την στιγμή που γνωρίζουμε τα όρια της πρώτης παραγώγου μπορούμε να βρούμε τα ολικά μέγιστα (όχι δηλαδή μόνο τα τοπικά). Η μέθοδος αυτή υπολογίζει πάντα το μέγιστο του φωτοβολταϊκού χωρίς να υπολογίσει ποτέ το τοπικό μέγιστο αντί για το ολικό όπως ο αλγόριθμος P & O . Τέλος είναι ταχύτερη στην εύρεση του mpp μετά από αλλαγή της ηλιοφάνειας . [21] , [23]
3. **Parasitic Capacitance (PC) (Παρασιτική Χωρητικότητα).** Στον υπολογισμό του σημείο μέγιστης ισχύος , λαμβάνεται υπόψη η παρασιτική χωρητικότητα που έχουν τα φωτοβολταϊκά , δηλαδή τα συσσωρευμένα φορτία στις επαφές των ηλιακών στοιχείων. Η μέθοδος αυτή βασίζεται στην InCond μέθοδο που αναλύθηκε στην προηγούμενη παράγραφο (**Παράγραφος 2.2.3**) [20]
4. **Fuzzy Logic Control.** Ο αλγόριθμος αυτός χρησιμοποιεί έναν πίνακα για την λήψη αποφάσεων ενώ βασίζεται σε έλεγχο σφάλματος. Αν και αρκετά γρήγορος αλγόριθμος έχει σαν μειονέκτημα ότι ο πίνακας κατασκευάζεται για συγκεκριμένο κάθε φορά φωτοβολταϊκό. [13]

Σύμφωνα με τους Atik, L., Petit, P., Sawicki, J. P., Ternifi, Z. T., Bachir, G., & Aillerie στη δημοσίευσή τους Comparison of four MPPT techniques for PV systems [14] , ο αλγόριθμος Perturb and Observe (P&O) σε σχέση με τους αλγορίθμους, Feedback Control, InCond, Current Sweep Method (Μέθοδος Εναλλαγής Ρεύματος), Direct Method, Parasitic Capacitance (PC) (Παρασιτική Χωρητικότητα) , Fuzzy Logic Control που έχει χρησιμοποιηθεί στην διπλωματική υστερεί σε ακρίβεια από τους υπόλοιπους. Εν αντιθέτως είναι πολύ πιο εύκολος στην κατανόηση του και δεν χρειάζεται γρήγορο μικροεπεξεργαστή για να λειτουργήσει. Πιο συγκεκριμένα με βάση τη δημοσίευση [14] ο αλγόριθμος InCond , ο οποίος υπολογίζει το σημείο μέγιστης ισχύος μέσω της παραγώγου της ισχύος ως προς την τάση βρίσκει ακριβώς το σημείο μέγιστης ισχύος (Mpp) και δεν το προσεγγίζει κάνοντας ταλαντώσεις γύρω από αυτό όπως ο αλγόριθμος P&O. Ακόμη με βάση την δημοσίευση των Zegaoui, A., Aillerie, M., Petit, P., Sawicki, J. P., Jaafar, A., Salame, C., & Charles, J. P. (2011), *Comparison of Two Common Maximum Power Point Trackers by Simulating of PV Generators. Energy Procedia* [15] ο αλγόριθμος αυξητικής αγωγιμότητας – InCond εκτός από την εύρεση του Mpp έχει και σαν πλεονέκτημα την μεγάλη ταχύτητα σε εναλλαγές ηλιοφάνειας , δηλαδή μπορεί πολύ γρήγορα να ξανά υπολογίσει το καινούργιο Mpp χωρίς να προβληματίζει τον προγραμματιστή για την ρύθμιση του βήματος του pwm.

Ο αλγόριθμος Fuzzy Logic Control σύμφωνα με τους Atik, L., Petit, P., Sawicki, J. P., Ternifi, Z. T., Bachir, G., Della, M., & Aillerie στη δημοσίευση τους Maximum power point tracking algorithm based on sliding mode and fuzzy logic for photovoltaic sources under variable environmental conditions [13], είναι αρκετά πιο γρήγορος από τον αλγόριθμο P&O. Ο αλγοριθμος Fuzzy Logic Control (FLC), για να λειτουργήσει χρησιμοποιεί έναν πίνακα για την λήψη αποφάσεων και βασίζεται στον έλεγχο σφάλματος. Αυτό τον καθιστά πολύ πιο γρήγορο αλλά έχει το μεγάλο μειονέκτημα , ότι ο πίνακας κατασκευαζεται για ένα συγκεκριμένο φωτοβολταϊκό επομένως πρέπει ο προγραμματιστής να ρυθμίζει το πρόγραμμα αν για κάποιο λόγο γίνει αλλαγή του φωτοβολταϊκού.

Table 3. MPPT efficiency.

MPPT methods	Sinusoidal profile	Ramp profile
P&O	95.36%	97.34%
Proposed algorithm combined with P&O	99.07%	99.19%
Fuzzy Logic Controller	95.51%	96.58%
Proposed algorithm with FLC	99%	99.15%

Στην εικόνα 2.2.4.1 με βάση την δημοσίευση των Karami, N., Moubayed, N., & Outbib, R. με τίτλο General review and classification of different MPPT Techniques. Renewable and Sustainable Energy Reviews [22] , φαίνεται η διαφορά στην απόδοση των δύο αλγορίθμων αφού ο αλγόριθμος P&O αποδίδει 97,34% ενώ ο Fuzzy Logic Control 96,58% .

Ο αλγόριθμος Current Sweep Method (Μέθοδος Εναλλαγής Ρεύματος) σύμφωνα με τους Karami, N., Moubayed, N., & Outbib, R. , σε δημοσιεύσεις που έχουν γίνει [22] , η μέθοδος αυτή βασίζεται στον προσδιορισμό της του παραγώγου της ισχύος έξοδου ως προς το ρεύμα.

Ο αλγόριθος Parasitic Capacitance (PC) (Παρασιτική Χωρητικότητα) συμφωνα με τους Hohm, D. P., & Ropp, M. E. και της δημοσίευσης Comparative study of maximum power point tracking algorithms. Progress in Photovoltaics: Research and Applications [20] , είναι παρόμοιος με τον αλγόριθμο InCond. Περιλαμβάνεται η επίδραση της χωρητικότητας παρασιτικής διασταύρωσης των ηλιακών κυψελών CP, η οποία μοντελοποιεί την αποθήκευση φορτίου στις συνδέσεις p-n των ηλιακών κυψελών.

Ο αλγόριθμος Direct Method (Άμεση Μέθοδος) με βάση τους Olusegun, A. T., Adebukola, A. Z., Denwigwe, I. H., Oluseyi, P. O., & Olubayo, B. M. (2019) στην δημοσίευση τους Comparative Analysis of Two Direct MPPT Methods Used for Tracking Maximum Power Points in a Photovoltaic System. World Scientific News και στους] Bendib, B., Belmili, H., & Krim, F. (2015) και την δημοσίευση τους A survey of the most used MPPT methods: Conventional and advanced algorithms applied for photovoltaic systems. Renewable and Sustainable Energy Reviews, βρίσκει τα ολικά μέγιστα(όχι μόνο τα τοπικά) από την στιγμή που είναι γνωστά τα όρια της πρώτης παραγώγου . Αυτό έχει ως αποτέλεσμα να βρίσκει το σημείο μεγίστης ισχύος πολύ γρήγορα ο αλγόριθμος και να μην υπάρχει κάποιο πρόβλημα στις γρήγορες αλλαγές της ηλιοφάνειας.

Για τους αλγόριθμους, Feedback Control – Έλεγχος ανάδρασης, Incremental Conductance (InCond) – Αυξητική αγωγιμότητα, Current Sweep Method (Μέθοδος Εναλλαγής Ρεύματος), Direct Method (Άμεση Μέθοδος), Parasitic Capacitance (PC) (Παρασιτική Χωρητικότητα) είναι απαραίτητη η χρήση ενός δεύτερου μετρητή ρεύματος , ώστε να μπορούμε να μετράμε και την ισχύ στην έξοδο Αυτό αυτόματα αυξάνει το κόστος της κατασκευής. Επομένως επέλεξα να χρησιμοποιήσω τον αλγόριθμο Perturb and Observe (P&O) – Διαταραχή και παρατήρηση για δυο βασικούς λόγους:

- Low budget κατασκευή
- Πιο εύκολος στην κατανόηση αλλά και στον προγραμματισμό του μικροεπεξεργαστή

Για την υπολοίηση του mppt υπάρχουν αρκετοί αλγόριθμοι , στην παρούσα διπλωματική αναφέρθηκαν κάποιοι από αυτούς.

3 ΚΕΦΑΛΑΙΟ 3ο : (ΜΕΤΑΤΡΟΠΕΑΣ ΥΠΟΒΙΒΑΣΜΟΥ)

Μετατροπέας υποβιβασμού – ανύψωσης τάσης (Buck converter – Boost converter)

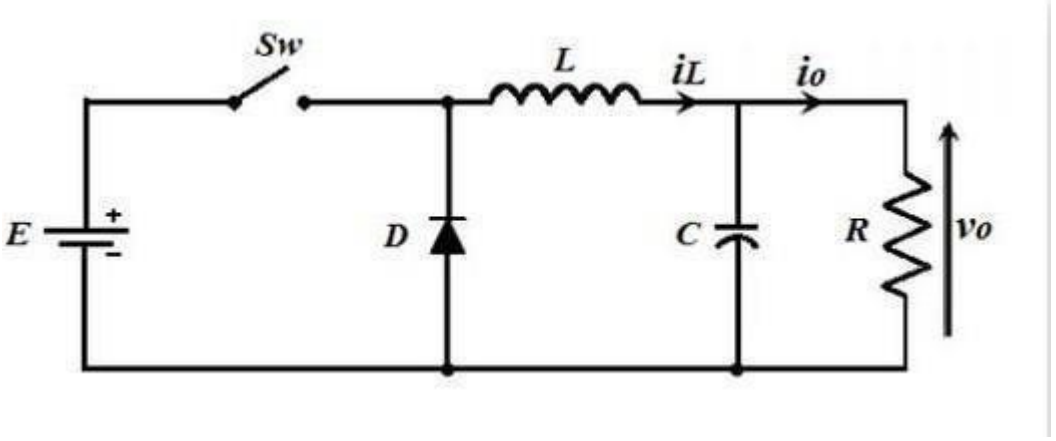
Σε αυτό το κεφάλαιο περιγράφεται η αρχή λειτουργίας του buck converter και υπολογισμοί για τον μετατροπέα της εργασίας.

3.1 Τι είναι ο buck converter

Ένας μετατροπέας buck είναι ένας μετατροπέας ισχύος DC σε DC που μειώνει την τάση από την είσοδο στην έξοδο του. Είναι μια κατηγορία τροφοδοτικού μεταγωγής που συνήθως περιέχει τουλάχιστον δύο ημιαγωγούς και τουλάχιστον ένα στοιχείο αποθήκευσης ενέργειας, έναν πυκνωτή, έναν επαγωγέα ή και τα δύο σε συνδυασμό.

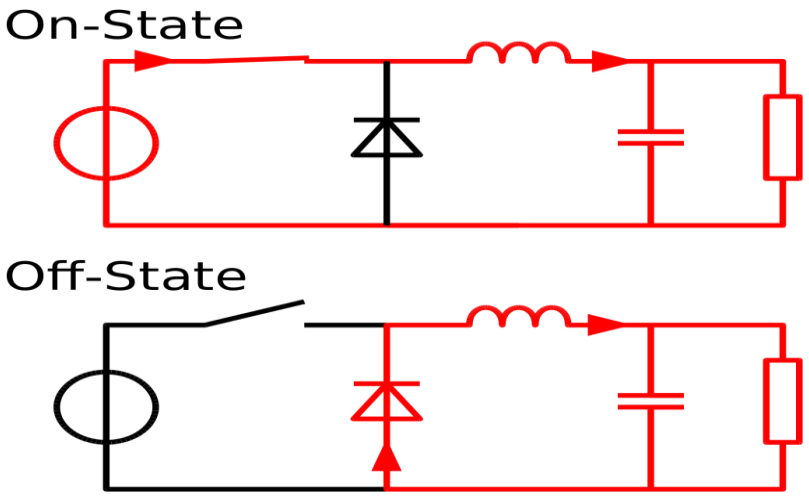
3.2 Θεωρία λειτουργίας του buck converter

Η λειτουργία του buck converter είναι να ελέγχει το ρεύμα ενός πηνίου μέσω δύο διακοπτών. Οι δύο αυτοί διακόπτες είναι συνήθως transistor ή fet και ο δεύτερος συνήθως δίοδος ή transistor ή fet.Αν η δίοδος αντικατασταθεί με transistor ή fet τότε ο converter λέγεται σύγχρονος (**synchronous)**



E-3.2.1 *- Βασικό κύκλωμα buck converter*

Όπως φαίνεται και στην εικόνα ***E-3.2.1*** *αφού* ο διακόπτης είναι ανοιχτός το κύκλωμα δεν διαρρέεται από ρεύμα . Με το κλείσιμο του διακόπτη ξεκινάει το ρεύμα να περνάει στο τοκύκλωμα. Στην συνέχεια όπως θα δούμε και στην εικόνα **E-3.2.2(κατάσταση on state)** παρακάτω το ρεύμα αυτό σταθεροποιείται αποθηκεύοντας την ενέργεια αυτή στο πηνίο υπό την μορφήμαγνητικού πεδίου . Με το άνοιγμα του διακόπτη η ενέργεια που έχει αποθηκευτεί στο πηνίο μεταδίδεται στην δίοδο , όπως θα δούμε στην εικόνα E-3.2.2(κατάσταση off state). Με κόκκινο είναι σχεδιασμένη η πορεία του ηλεκτρικού ρεύματος σε κάθε μια περίπτωση.



E-3.2.2 *– Καταστάσεις λειτουργίας buck converter*

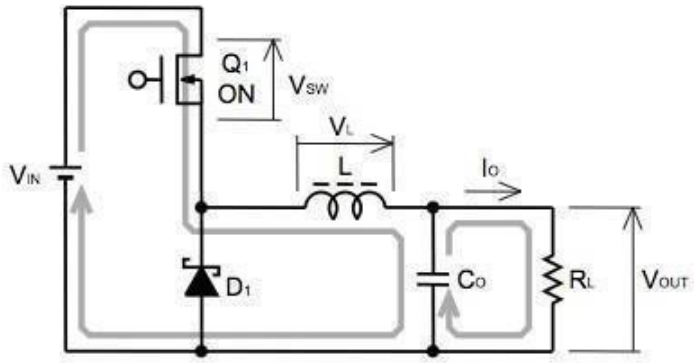


Figure 1 Basic Buck Converter Circuit Switching Element ON

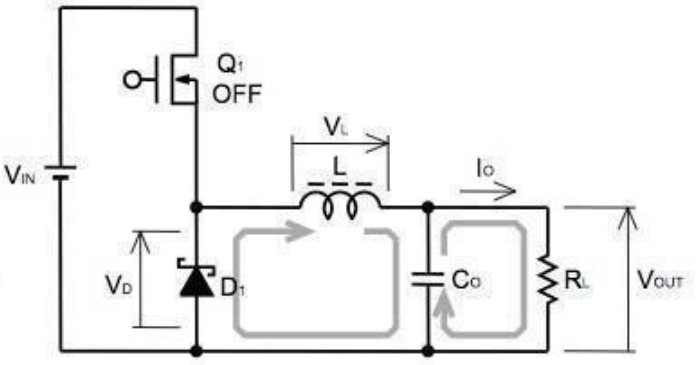


Figure 2 Basic Buck Converter Circuit Switching Element OFF

E-3.2.3 *– Καταστάσεις λειτουργίας buck converter*

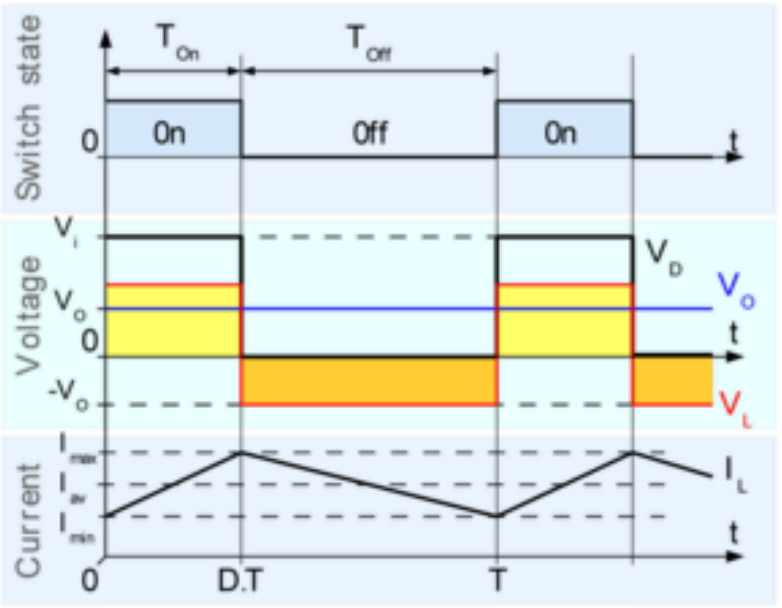
Ο buck converter μπορεί να λειτουργήσει με δυο διαφορετικούς τρόπους . Τον συνεχή τρόπο λειτουργίας και τον ασυνεχή. Στην εργασία μου θα χρησιμοποιήσω τον συνεχή τρόπο λειτουργίας του buck converter. [4] , [5]

<u>Συνεχής τρόπος λειτουργίας :</u>

Στον τρόπο λειτουργίας αυτόν το ρεύμα στο πηνίο δεν μηδενίζεται ποτέ όπως φαίνεται και στο

κάτω γράφημα της εικόνας ***E-3.2.4***. Με το κλείσιμο του διακόπτη ο πυκνωτής φορτίζεται και ταυτόχρονα αυξάνει το ρεύμα στο πηνίο. Όταν ο διακόπτης είναι ανοιχτός το ρεύμα ελαττώνεται όμως πριν μηδενιστεί ο διακόπτης (ή το Q1) κλείνει ξανά το κύκλωμα και ξανά φορτίζονται το πηνίο και ο πυκνωτής.

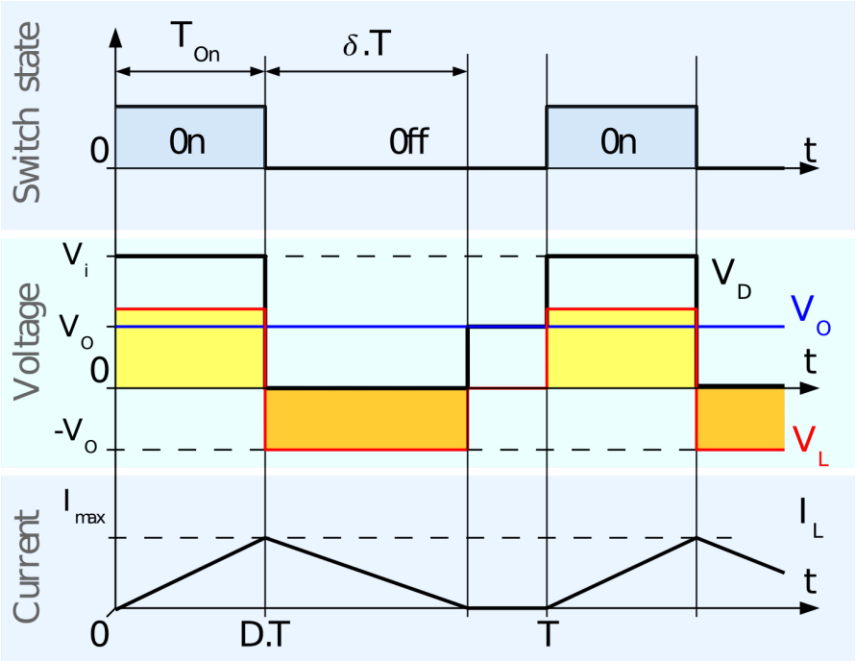
.



E-3.2.4 *– Γραφική παράσταση της τάσης και του ρεύματος σε ιδανικό buck converter στο συνεχή τρόπο λειτουργίας*

<u>Ασυνεχής τρόπος λειτουργίας :</u>

Στον τρόπο λειτουργίας αυτόν όπως παρατηρούμε και στην εικόνα ***E-3.2.5*** το ρεύμα στο πηνίο μηδενίζεται αλλά για ένα πολύ μικρό χρονικό διάστημα, έχοντας ως αποτέλεσμα το πηνίο να εκφορτίζεται πλήρως. Ο τρόπος αυτός λειτουργίας χρησιμοποιείται για πολύ μικρές απαιτήσεις ενέργειας(ρεύματος) στο φορτίο.



E-3.2.5 *– Γραφική παράσταση της τάσης και του ρεύματος σε ιδανικό buck converter στον ασυνεχή τρόπο λειτουργίας*

3.3 Σχεδίαση του buck converter

Ο buck converter έχει τα εξής βασικά χαρακτηριστικά.

1. Ισχύς
2. Τάση εισόδου
3. Τάση εξόδου

Για τον buck converter που κατασκεύασα, η ισχύς είναι 50 -150 w, η τάση εισόδου μικρότερη των 24V και η τάση εξόδου περίπου 12V DC. Στον υπολογισμό του κυκλώματος ενός buck converter καταλυτικό ρόλο έχουν και η αυτεπαγωγή του πηνίου, η χωρητικότητα του πυκνωτή αλλά και η συχνότητά λειτουργίας . Αυτά καθορίζουν την χαμηλή επιθυμητή κυμάτωση (ripple) (ρεύματος και τάσης). Γενικά όσο μεγαλύτερη είναι η συχνότητα λειτουργίας τόσο μικρότερα είναι τα μεγέθη του πηνίου και του πυκνωτή , δηλαδή αυτό σημαίνει άμεσα μικρότερο κόστος κατασκευής. Αν αυξηθεί η συχνότητα του παλμού που δίνει ο μικροεπεξεργαστής μικραίνει ο όγκος του πυκνωτή και του πηνίου και όπως προανέφερα μειώνεται το κόστος κατασκευής του πειράματος. Μαζί με το κόστος πειράματος μειώνεται ταυτόχρονα και η απόδοση όμως των fet αφού σε χαμηλές συχνότητες δεν δουλεύουν σωστά γιατί έχουν μεγάλες απώλειες (switching losses). Για τον λόγο αυτόν προσπάθησα να βρω μια μέση λύση. Στην εργασία χρησιμοποιήθηκαν δοκιμαστικά δύο συχνότητες(50KHz και 20KHz), σχετικά χαμηλές γιατί δεν υπήρχαν περιορισμοί ως προς το κόστος και τον όγκο των υλικών. Συνήθως οι buck converter του εμπορίου εργάζονται σε συχνότητες από 80KHz -500KHz .

3.4 Υπολογισμός και υλοποίηση του πηνίου του buck converter

Σύμφωνα με την εικόνα Ε-3.2.4 για τον buck converter (για συνεχές ρεύμα πηνίου) ισχύουν οι παρακάτω σχέσεις.

$$\overline{V_o} = \text{output average voltage} = \frac{1}{T_s}\int_0^{T_s} V_o(t)dt = \frac{1}{T_s}\left[\int_0^{T_{on}} V_{in}dt + \int_0^{T_s} 0dt\right] = V_{in}\frac{T_{on}}{T_s} = V_{in}D \qquad (3.4.3)$$

$$\overline{I_o} = \text{μέση τιμή του ρεύματος εξόδου} = \overline{I_L} = \frac{\overline{V_o}}{R} \qquad (3.4.4)$$

Όπου :

D=βαθμός χρησιμοποίησης του διακόπτη (duty cycle t_{on}/T_s)

T_s= Διακοπτική περίοδος λειτουργίας του διακόπτη

$\overline{V_o}$ = Μέση τιμή της τάσης εξόδου

V_{in}=Τάση εισόδου

Όπως διαπιστώνεται από τη σχέση (3.4.3), κρατώντας τη διακοπτική συχνότητα Ts σταθερή, μπορούμε να ελέγξουμε την τάση εξόδου, αυξομειώνοντας το χρόνο αγωγής (t_{on}) του διακόπτη. Όταν ο χρόνος ton = 0 τότε η τάση εξόδου παίρνει την ελάχιστη τιμή ίση με το μηδέν. Όταν ο χρόνος ton γίνει ίσος με τη διακοπτική συχνότητα Ts τότε η τάση εξόδου παίρνει τη μέγιστη τιμή ίση με την τάση τροφοδοσίας.

Επίσης από την εικόνα Ε-3.2.4 προκύπτουν οι παρακάτω σχέσεις.

$V_L(t) = V_{in} - V_o$, όταν ο διακόπτης (fet) είναι κλειστός (3.4.5)

$V_L(t) = -V_o$, όταν ο διακόπτης (fet) είναι ανοιχτός (3.4.6)

Επίσης ισχύει ότι :

$$\frac{dv_o}{dt} = \frac{i_c}{C} \qquad (3.4.7)$$

$$V_L(t) = L\frac{di_L(t)}{dt} \qquad (3.4.8)$$

Όπως μπορεί να διαπιστωθεί από την εικόνα Ε-3.2.4, η οποία παρουσιάζει το ρεύμα και την τάση του πυκνωτή εξόδου, το ρεύμα του πυκνωτή για τη μισή διακοπτική περίοδο είναι θετικό με αποτέλεσμα να εξαναγκάζει την τάση του πυκνωτή να αυξηθεί μεταξύ του κατώτατου και ανώτατου ορίου του. Σε όλη αυτή τη χρονική διάρκεια, το ολικό φορτίο q, το οποίο αποθηκεύεται στον πυκνωτή, δίνεται από την ακόλουθη σχέση:

Αλλαγή φορτίου πυκνωτή = C x Αλλαγή τάσης πυκνωτή

$\Delta Q = C \times \Delta V_o$ (3.4.9)

Το ολικό φορτίο είναι το εμβαδόν του τριγώνου που φαίνεται στο σχήμα 7.6 και δίνεται από την ακόλουθη σχέση:

$$\Delta Q = \frac{1}{2}\left(\frac{T_S}{2}\right)\left(\frac{\Delta I_L}{2}\right) = \frac{T_S \Delta I_L}{8} \qquad (3.4.10)$$

Από τις σχέσεις 3.4.9 και 3.4.10 προκύπτει ότι :

$$\Delta V_O = \frac{\Delta I_L T_S}{8C} \qquad (3.4.11)$$

Όταν το fet είναι ανοικτό (άγει), τότε ο ρυθμός αύξησης του ρεύματος του πηνίου δίνεται από την ακόλουθη σχέση :

$$\frac{dI_L}{dt} = \frac{V_{IN} - \overline{V_O}}{L} \quad \text{ή} \quad \frac{\Delta I_L}{t_{ON}} = \frac{V_{IN} - \overline{V_O}}{L}$$

$$\text{ή} \quad \Delta I_L = \frac{V_o(1-D)}{Lf_S} \tag{3.4.12}$$

Επομένως ,

$$\Delta V_O = \frac{(1-D)V_O T_S^2}{8LC} = \frac{V_O(1-D)}{8CLf_s^2} \tag{3.4.13}$$

Η μέγιστη και η ελάχιστη τιμή του ρεύματος του πηνίου υπολογίζεται από τους παρακάτω τύπους :

$$I_{L,MAX} = \underline{I}_O + \frac{\Delta I_L}{2} = \frac{\overline{V_O}}{R} + \frac{\Delta I_L}{2} = \frac{V_{IN}D}{R} + \frac{\Delta I_L}{2} \tag{3.4.14}$$

ΚΑΙ

$$I_{L,MIN} = \overline{I_O} - \frac{\Delta I_L}{2} = \frac{V_O}{R} - \frac{\Delta I_L}{2} = \frac{V_{IN}D}{R} - \frac{\Delta I_L}{2} \tag{3.4.15}$$

Αντικαθιστώντας την (3.4.12) στις (3.4.14) και (3.4.15) προκύπτει :

$$I_{L,MAX} = \frac{\overline{V_O}}{R} + \frac{\overline{V_O}(1-D)T_S}{2L} = \frac{V_{IN}D}{R} + \frac{V_{IN}D(1-D)T_S}{2L} \tag{3.4.16}$$

ΚΑΙ

$$I_{L,MAX} = \frac{\overline{V_O}}{R} - \frac{\overline{V_O}(1-D)T_S}{2L} = \frac{V_{IN}D}{R} - \frac{V_{IN}D(1-D)T_S}{2L} \tag{3.4.17}$$

Από τις 2 παραπάνω σχέσεις (3.4.16) και (3.4.17) προκύπτει ότι για να είναι το ρευμα του πηνίου συνεχές χρειάζεται να ισχύει η παρακάτω ανισότητα:

$$\frac{V_{IN}D}{R} > \frac{V_{IN}D(1-D)T_S}{2L} \quad \Leftrightarrow \quad f_s > \frac{(1-D)R}{2L} \tag{3.4.18}$$

ΕΠΟΜΕΝΩΣ το ρεύμα του πηνίου είναι :

$$L_{\min} = \frac{(1-D)R}{2f_S} \quad (H)$$

Συγκεκριμένα στο πείραμά μου έχω τα εξής χαρακτηριστικά :

V_{IN}=18V

V_{OUT}=13V

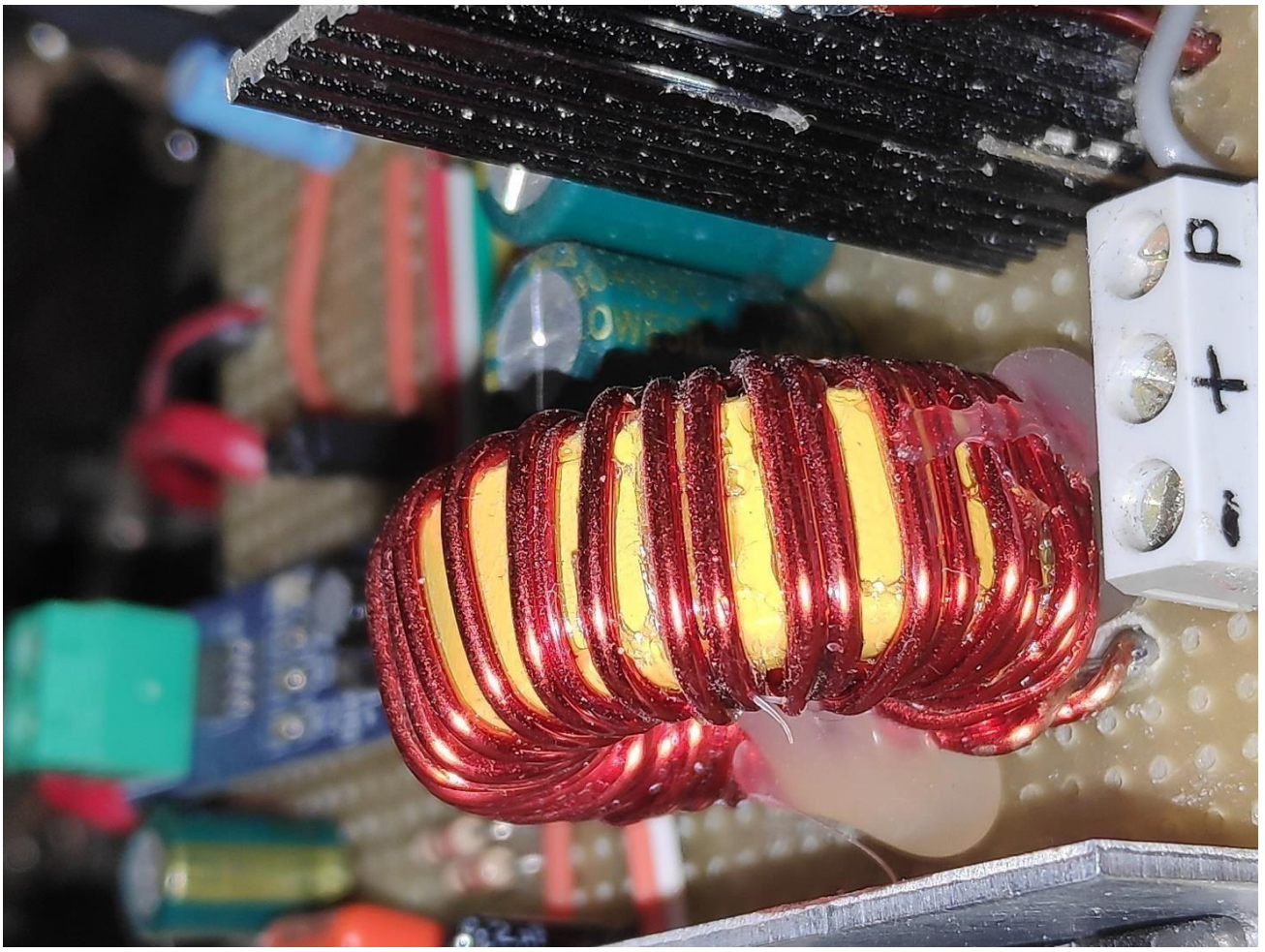
I_L=1,3A

$$D = \frac{V_{OUT}}{V_{IN}} = \frac{13V}{18V} = 0.72$$

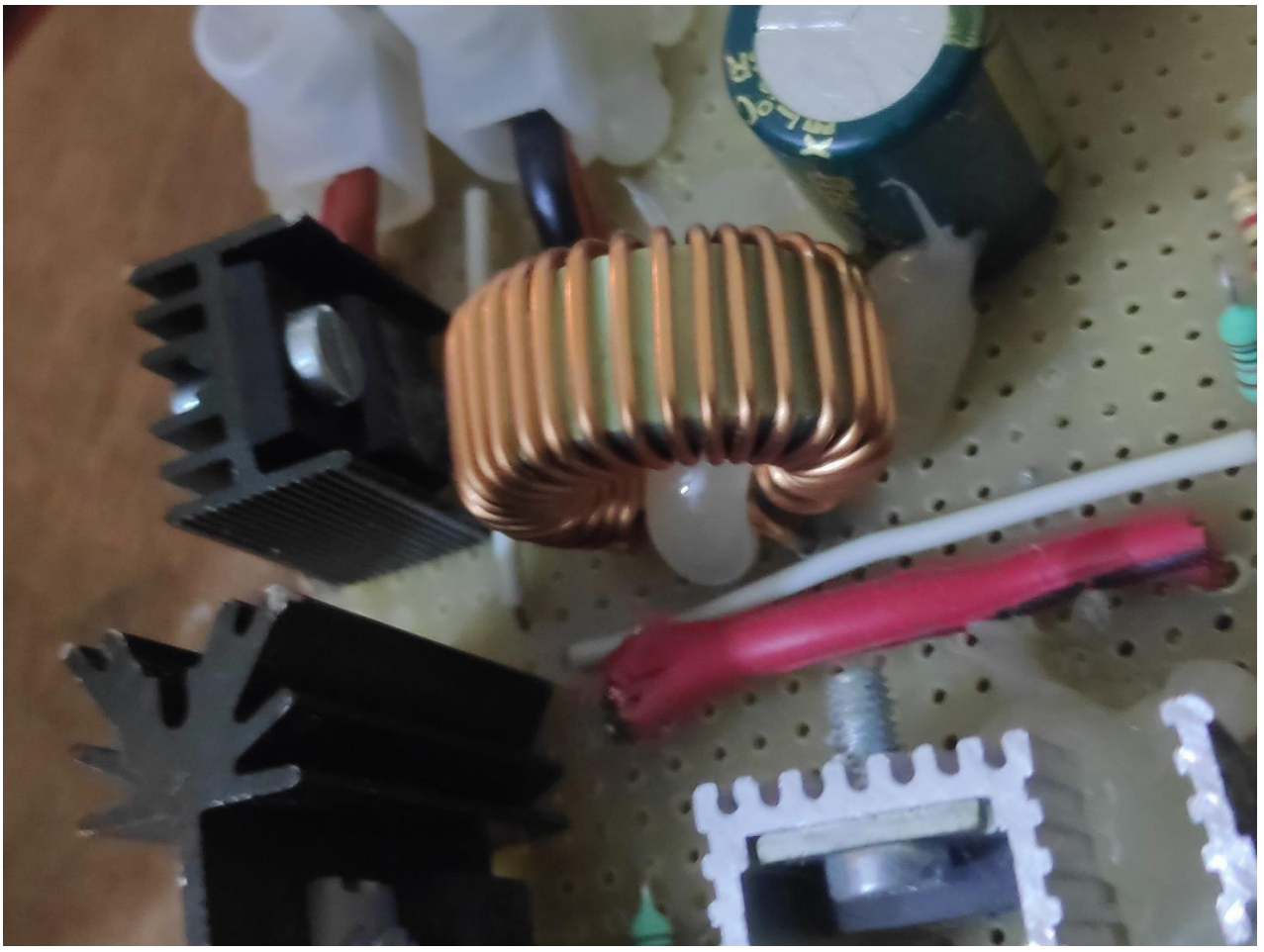
Επιθυμητή κυμάτωση 50% , επομένως : ΔI_L=1.3Ax0.5=0.65A

Άρα $L = \frac{V_O(1-D)}{\Delta I_L f_s} = \frac{13\times(1-0.72)}{0.65\times 50000} = 112\mu H$.

Το πηνίο που χρησιμοποίησα είναι 100μH.



Ε-3.4.1 Πηνίο που χρησιμοποιήθηκε σε ένα από τα πειράματα.



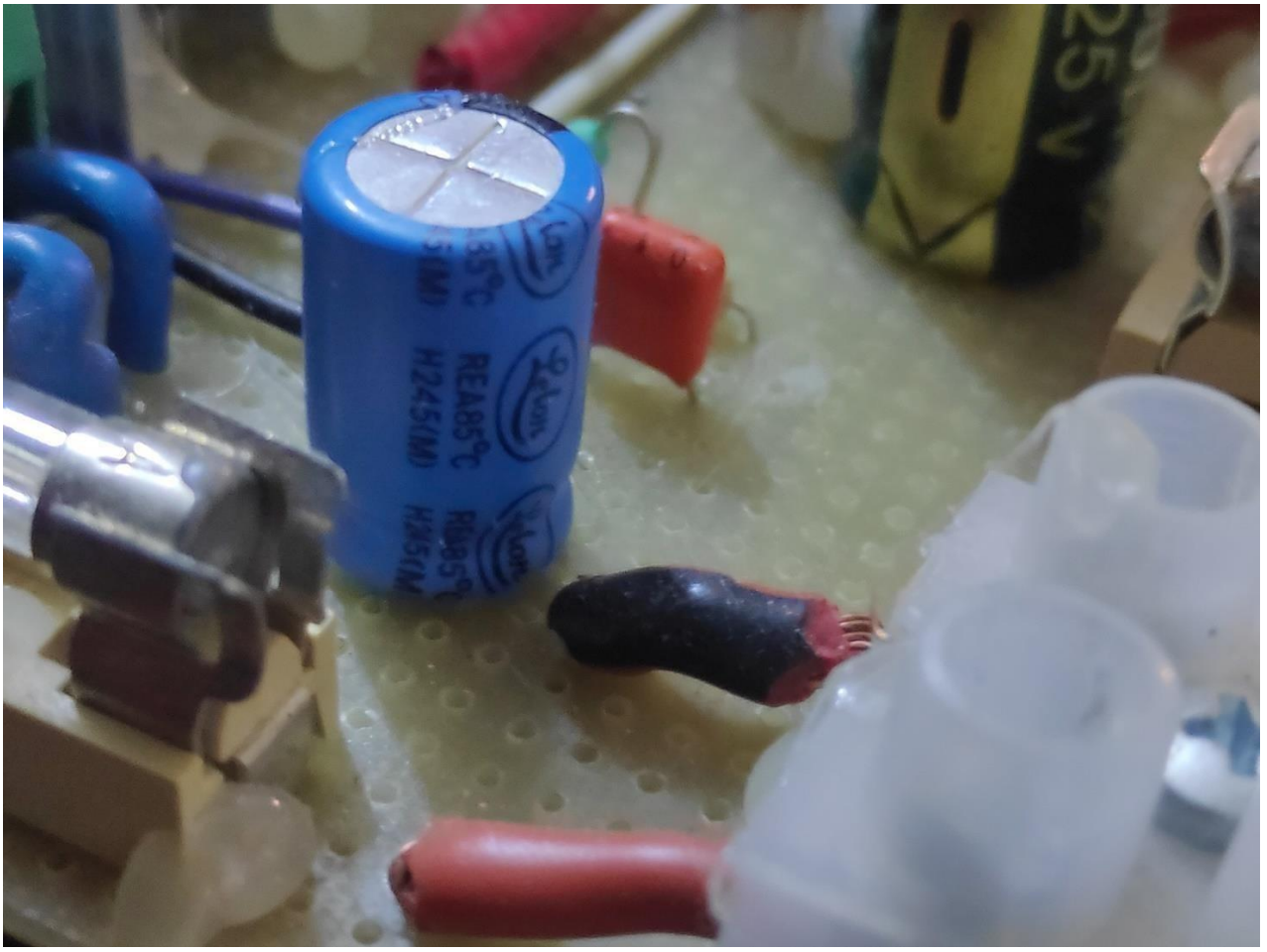
E-3.4.2 Πηνίο που χρησιμοποιήθηκε σε ένα από τα πειράματα.

3.5 Υπολογισμός του πυκνωτή του buck converter

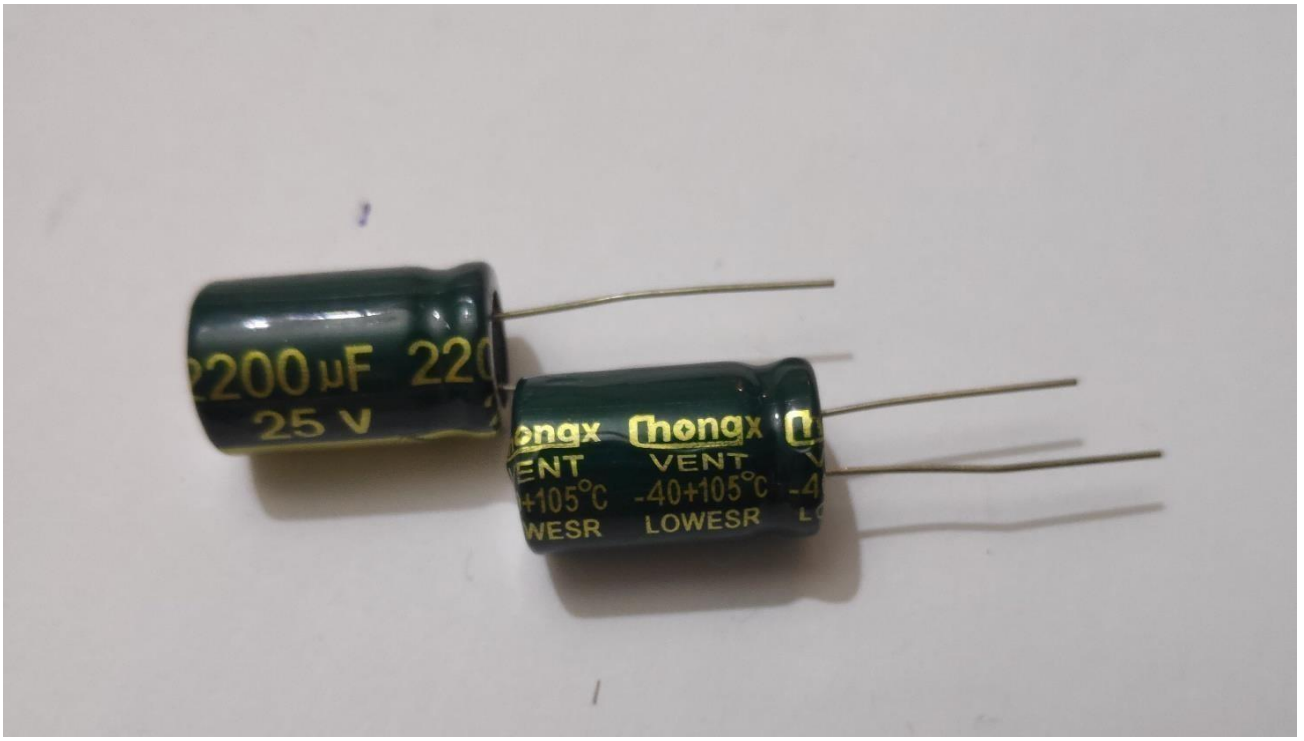
Ο πυκνωτής χρησιμοποιείται για την εξομάλυνση της κυμάτωσης (ripple) αλλά και για την ελαχιστοποίηση της υπέρτασης (overshoot). Για την επίτευξη των παραπάνω απαιτείται η χρήση πυκνωτή με υπολογισμένη ελάχιστη χωρητικότητα. Για να έχω όσο το δυνατόν λιγότερες απώλειες χρησιμοποιώ πυκνωτή με χαμηλή ισοδύναμη εσωτερική αντίσταση ESR(equivalent seriesresistance) Η ESR μετριέται με ac τροφοδοσία σε ορισμένες συχνότητες. Για παράδειγμα η ESR για κεραμικούς πυκνωτές είναι μεταξύ 0.01 και 0.1 ohms. Σε ηλεκτρολυτικούς πυκνωτές είναι μεγαλύτερη και οι ηλεκτρολυτικοί που αντέχουν σε μεγάλη θερμοκρασία, έχουν όχι μόνο μεγαλύτερη διάρκεια ζωής αλλά και χαμηλότερη ESR. Η ESR αυξάνεται με την πάροδο του χρόνου. [Ο πυκνωτής εξόδου υπολογίζεται από τον τύπο: [10]

$$C_{out} = \frac{1-D}{8L\left(\frac{\Delta V_C}{V_O}\right)^2 f_s} = \frac{1-0.72}{8\times 100\times 10^{-6}\times 0.001\times 50000^2} = 140\mu F \qquad (3.5.1)$$

Τότε σύμφωνα με τον τύπο (3.5.1) με απλή αντικατάσταση βρίσκουμε C=140μF για F_{sw} = 50KHz Με αυτόν τον τρόπο καταφέρνω να κρατήσω την κυμάτωση σε σχετικά χαμηλές τιμές(0.1% της μέσης τιμής της τάσης εξόδου). Η χωρητικότητα που βρήκα από την σχέση 3.5.1 δεν υπάρχει στο εμπόριο , επομένως θα χρησιμοποιήσω μεγαλύτερες τιμές 200μF) αυξάνοντας λίγο το κόστος κατασκευής.



E-3.5.1 Πυκνωτής που χρησιμοποιήθηκε σε ένα από τα πειράματα.



E-3.5.2 Πυκνωτής που χρησιμοποιήθηκε σε ένα από τα πειράματα.

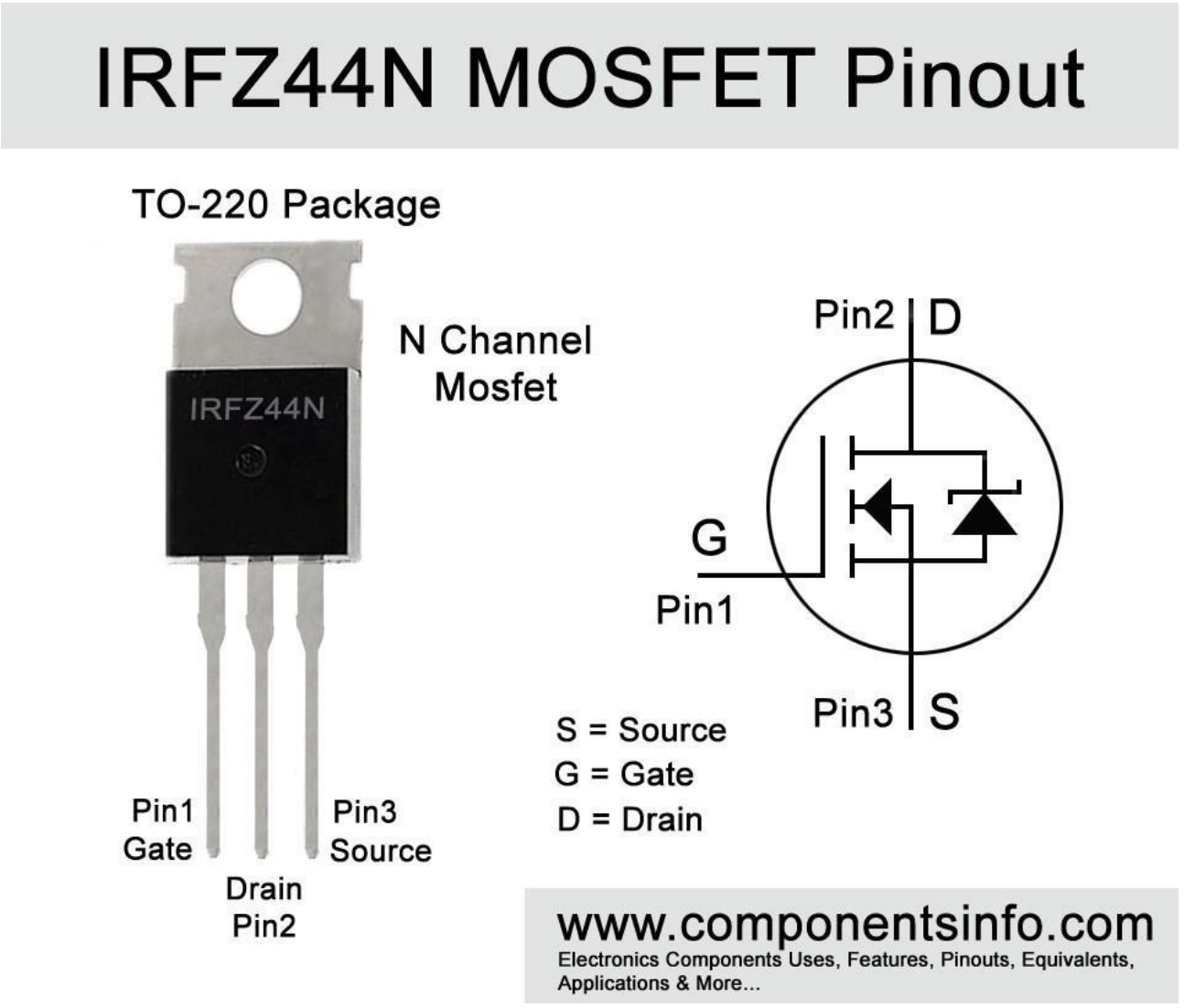
3.6 Υπολογισμός των FET και των οδηγών αυτών

Βασικοί παράγοντες για την επιλογή των MOSFET ήταν οι εξής :

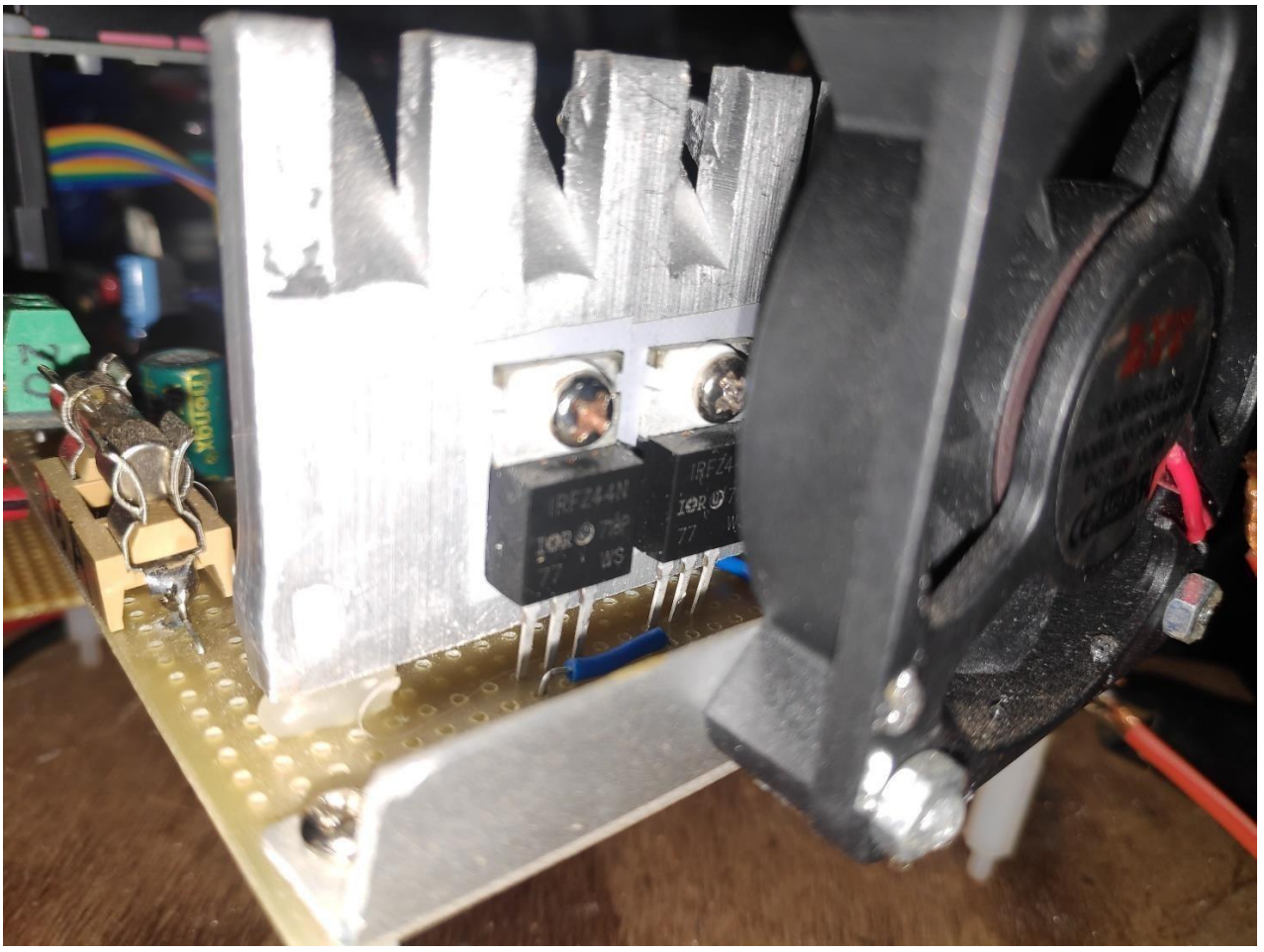
1. Ρεύμα λειτουργίας (Ids).
2. ON αντίσταση (Rds ON).(Πρέπει να είναι όσο το δυνατόν χαμηλή).
3. Τάση λειτουργίας (Vds).

4.Απώλειες λειτουργίας: Έχουν άμεση σχέση με την Rds – on και το dutycycle. Οι απώλειες πρέπει να είναι μικρές.

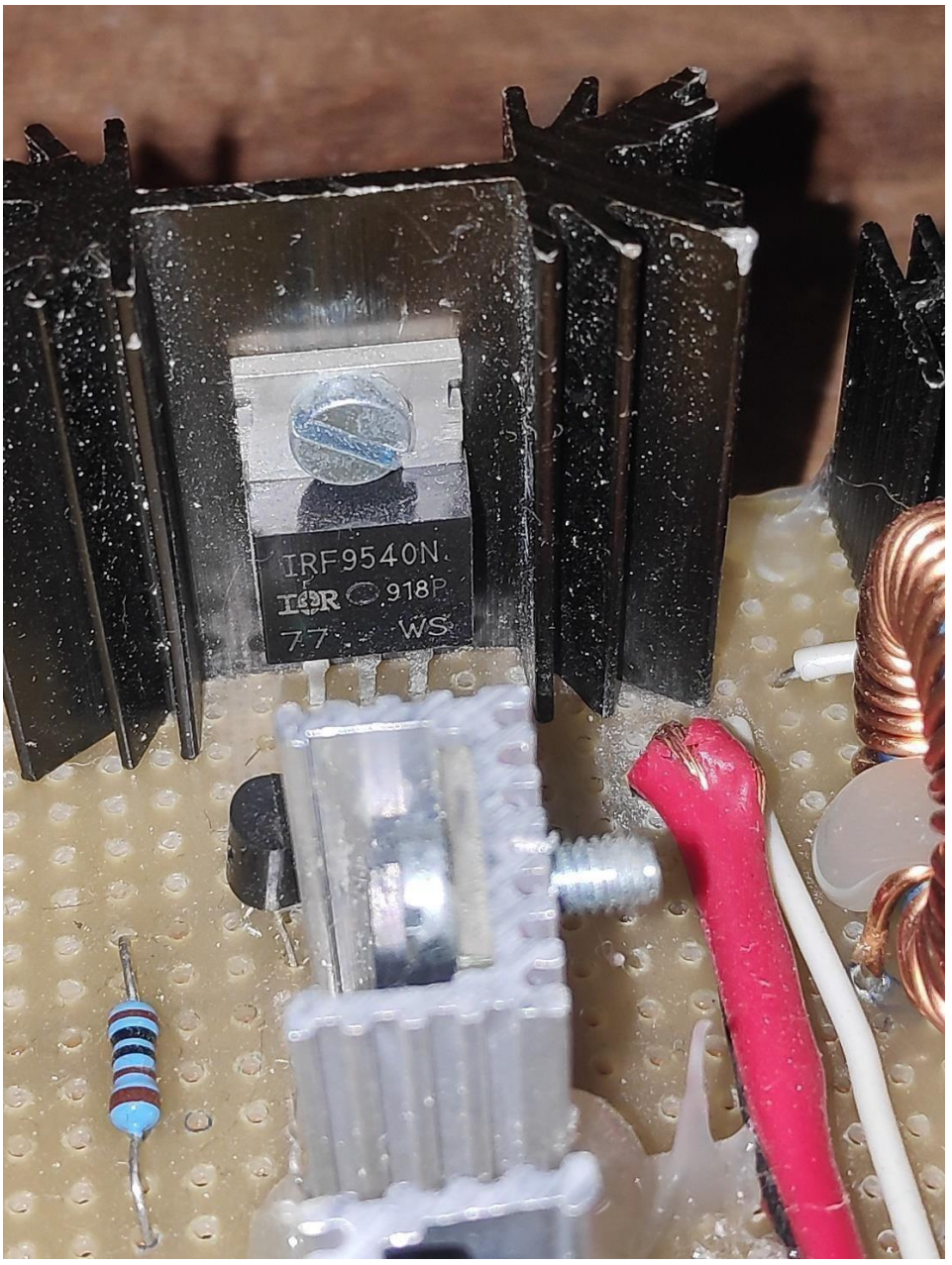
5. Απώλειες μεταγωγής (switching loss): Πρέπει να είναι μικρές. Εξαρτώνται κυρίως από την συχνότητα (αυξάνονται με αύξηση συχνότητας). Βάσει των δεδομένων αυτών επιλέχθηκαν τα IRFZ44N(55V- 49A) ενώ στην 1η πλακέτα χρησιμοποίησα τα IRF-9540



E-3.6.1 Mosfet IRFZ44N

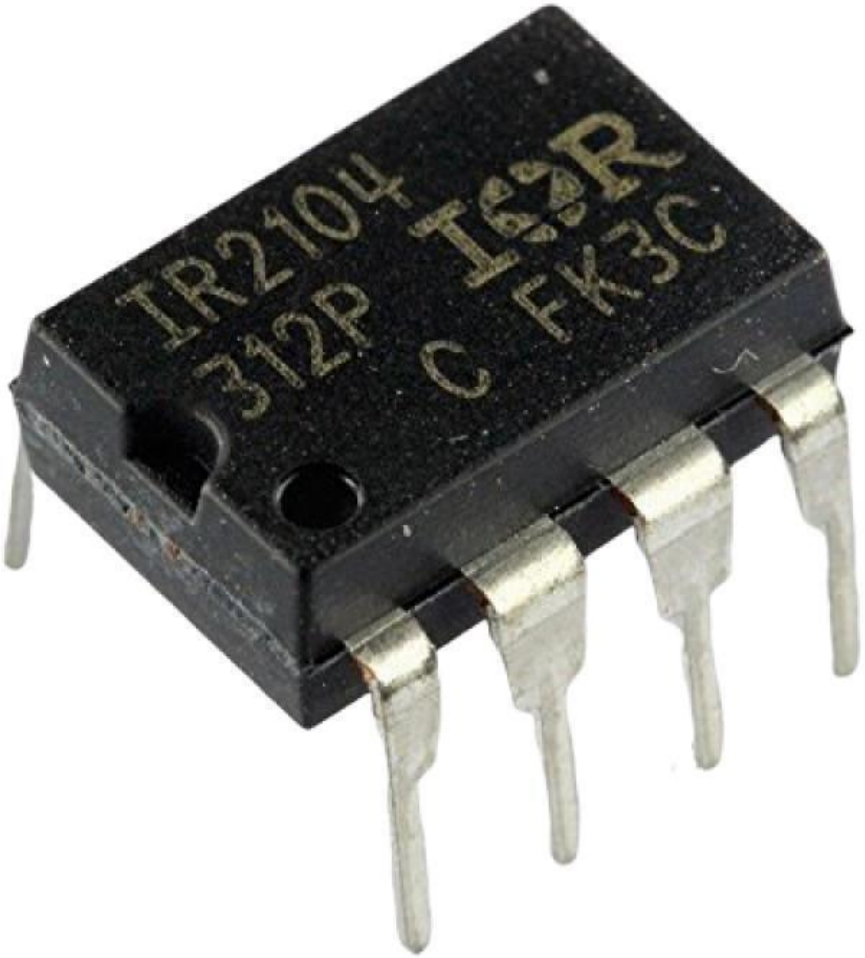


E-3.6.2 Mosfet IRFZ44N που χρησιμοποιήθηκαν σε ένα από τα πειράματα.



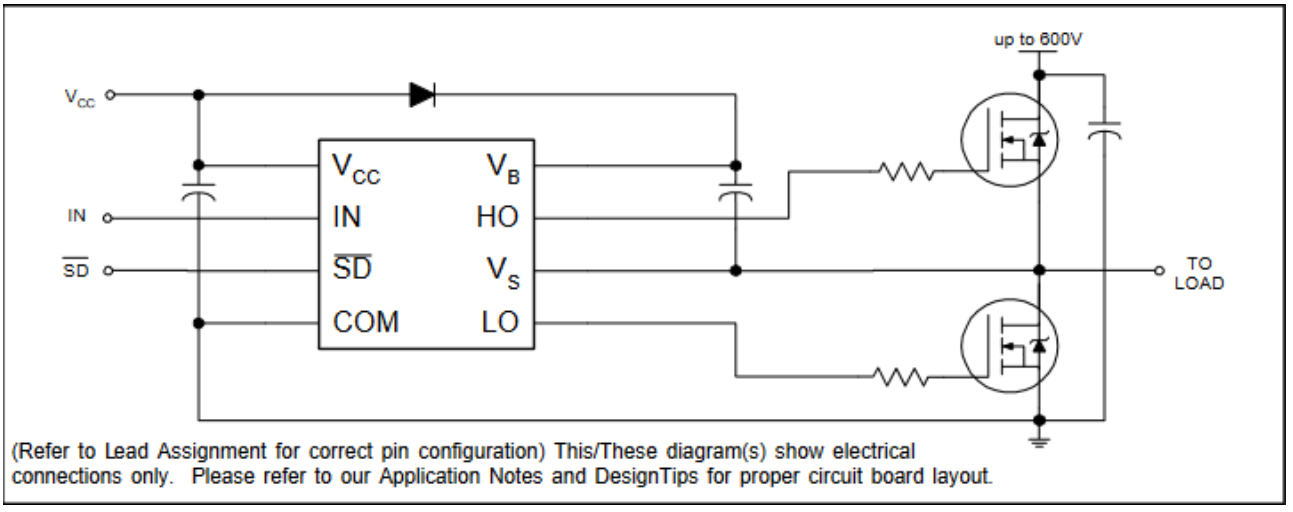
E-3.6.3 Mosfet IRF9540 που χρησιμοποιήθηκαν σε ένα από τα πειράματα.

. Θα μπορούσε να επιλεγεί και οποιοσδήποτε άλλος τύπος με παρόμοια η καλύτερα χαρακτηριστικά. Τα MOSFET χρειάζονται οδήγηση είτε από τρανζίστορ είτε από ειδικό ολοκληρωμένο κύκλωμα. Το ολοκληρωμένο επιτρέπει στο χαμηλό ρεύμα από τον μικροελεγκτή να οδηγήσει την πύλη των MOSFET που χρειάζονται περίπου 10V στην πύλη τους (gate) για να οδηγηθούν σωστά. Τα MOSFET έχουν μια χωρητικότητα πύλης που φορτιζόμενη αναγκάζει τα fet να άγουν και εκφορτιζόμενη να μην άγουν. Όσο περισσότερο ρεύμα δίνουμε στην πύλη τόσο πιο γρήγορα γίνεται η εναλλαγή on-off. Το driver που επιλέχθηκε για την οδήγηση των mosfet είναι το IR2104, για την απλότητά του και το χαμηλό κόστος.
[9]



E-3.6.4 IR2104

Προτιμήθηκε (και είναι φθηνότερος) ένας απλός dc-dc converter που τροφοδοτείται από το φωτοβολταϊκό. Στην τελική κατασκευή τροφοδοτείται κατευθείαν από την μπαταρία. Το σήμα PWM από το pin D9 του arduino ενώνεται με το IN του ολοκληρωμένου (εικόνα 3.6.4)

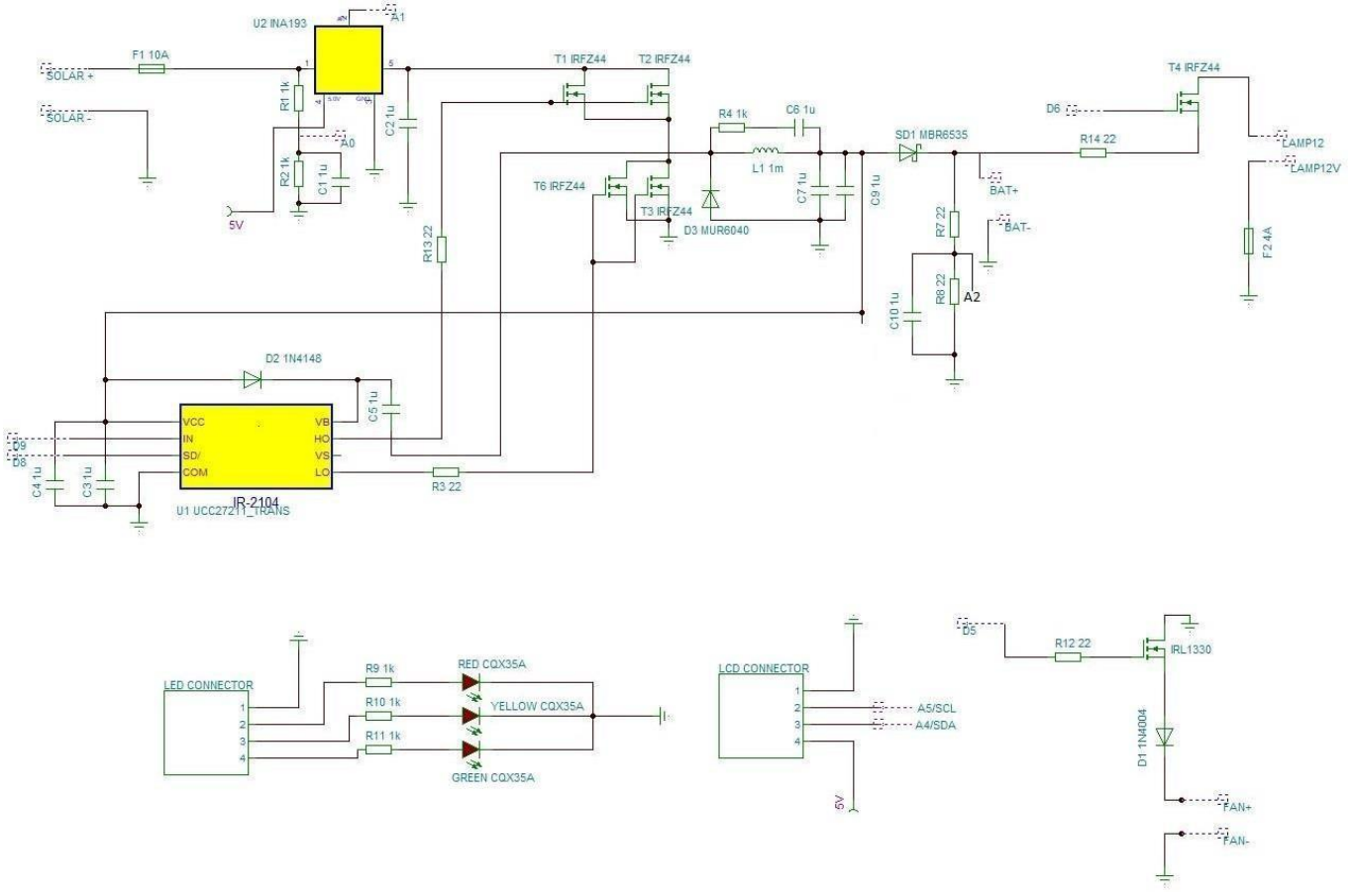


E-3.6.5 – Από το datasheet IR2104της I.R.

. Από εκεί το IR2104 δημιουργεί τα απαραίτητα σήματα στα pin HO και LO. Τα σήματα αυτά είναι τέτοια ώστε να εξασφαλίζεται ο συνεχής τρόπος λειτουργίας (continuous mode) του buck converter, οδηγώντας κατάλληλα τα 2 mosfet. Ο πυκνωτής μεταξύ του V_B και V_S μαζί με την δίοδο σχηματίζουν την αντλία φορτίου. Διπλασιάζεται έτσι η τάση εισόδου για την σωστή οδήγηση των mosfet. Το pin D8 του arduino ενώνεται στο pin SD του IR2104 και ελέγχει τη λειτουργία (on-off) του ολοκληρωμένου. [7] , [8]

3.7 Λειτουργία του buck converter

Πάνω αριστερά είναι η πηγή ισχύος (φωτοβολταϊκά). Η δίοδος που ακολουθεί (D2) είναι δίοδος για προστασία υπέρτασης (TVS – P6KE36CA ή 33CA). Μετά υπάρχει ένας διαιρέτης τάσης αποτελούμενος από τις αντιστάσεις R3 και R4. Το σημείο A0 του διαιρέτη πηγαίνει στο αντίστοιχο pin του μικροελεγκτή για να διαβάζει την τάση εισόδου. Στον Arduino είναι το pin A0. Τοκύκλωμα που ακολουθεί μετράει το ρεύμα που δίνει το φωτοβολταϊκό. Στο συγκεκριμένο σχέδιο είναι το ACS712, αλλά έχουν χρησιμοποιηθεί αρκετά διαφορετικά κυκλώματα (βλέπε παράρτημαΑ και Γ). Η έξοδος του αισθητήρα ρεύματος οδηγεί το pin A1 του Arduino . Αντίστοιχα κυκλώματα με διαιρέτη τάσης (R6, R7) και αισθητήρα ρεύματος έχουν χρησιμοποιηθεί και στην έξοδο του κυκλώματος που οδηγούν αντίστοιχα τα pin A2 και A3 του Arduino. Εν σειρά με το πρώτο αισθητήρα ρεύματος έχει τοποθετηθεί μια δίοδος *Schottky (MBR2545CT) (ή άλλη μικρότερη των 6A στο nano)* ώστε να μπλοκάρεται η ροή ρεύματος από την μπαταρία προς το φωτοβολταϊκό την νύχτα (στο σχήμα Ε3.7.1 δεν φαίνεται αλλά φαίνεται στο σχήμα Ε4.1.1)



E-3.7.1 Βασικό σχέδιο του buck converter στην πλακέτα pcb3

.Το κυρίως κύκλωμα του **buck converter** ξεκινάει από τον πυκνωτή C8 και τελειώνει στην έξοδο του πηνίου L1. Η καρδιά του **buck converter** είναι το ολοκληρωμένο IR2104. Αυτό έχει διπλό

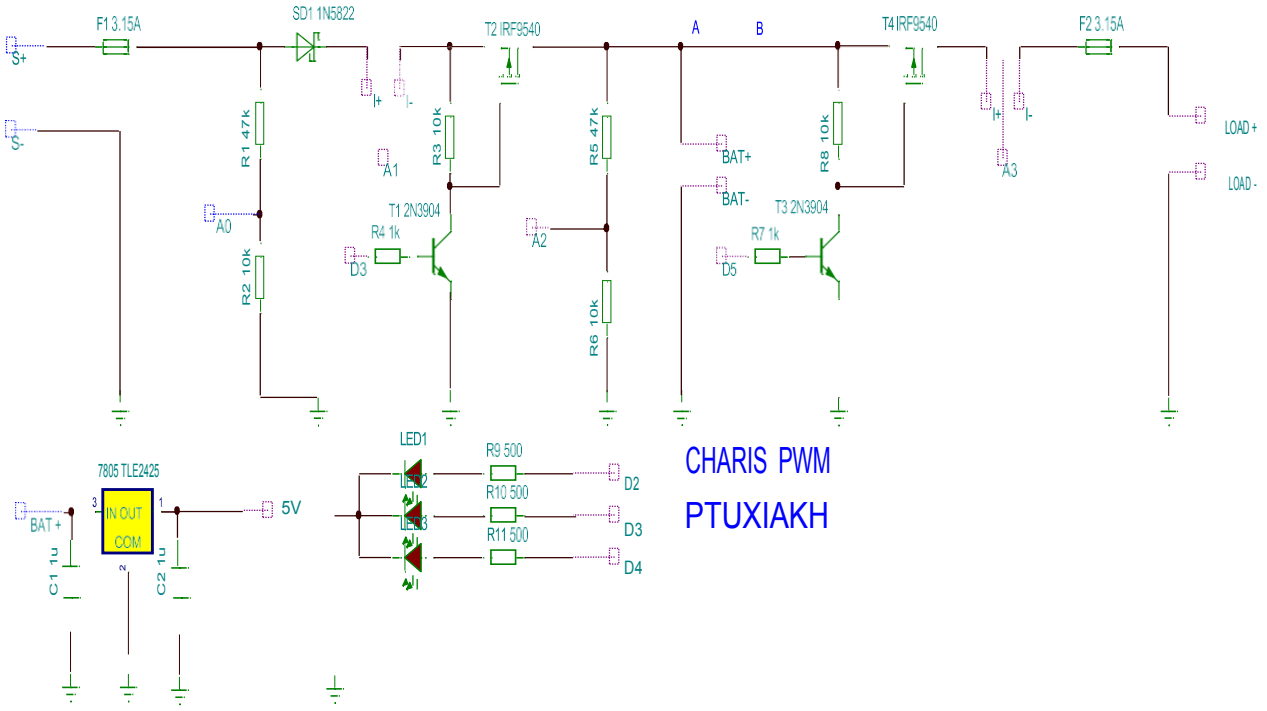
σκοπό. Οδηγεί τα MOSFET (low and high side) δίνοντας την κατάλληλη τάση και τον κατάλληλο χρονισμό. Το IR2104 (pin 2) οδηγείται με PWM από το pin D9 του Arduino. Το pin 3 του IR2104 οδηγείται από το pin D8 του Arduino και από εδώ μπορούμε να απενεργοποιήσουμε το ολοκληρωμένο. Η δίοδος D1 και ο πυκνωτής C1 απαιτούνται σύμφωνα με τις οδηγίες του κατασκευαστή (bootstrap circuit). Το πρόγραμμα στον μικροελεγκτή ελέγχει το PWM duty-cycle και δεν είναι ποτέ 100%. Το Q4 (IRLZ44N) ελέγχει το φορτίο. Χρησιμοποιήθηκε αυτό το MOSFET γιατί θέλει οδήγηση TTL και δεν χρειάζεται driver ολοκληρωμένο ή τρανζίστορ.

Το βασικό σχέδιο για την λειτουργία του **buck converter** φαίνεται στην εικόνα Ε3.7.1.Η πηγη ισχύος που φαίνεται πάνω αριστερά στο σχήμα είναι το φωτοβολταϊκό στην περίπτωση του πειράματος μου. Η δίοδος που ακολουθεί (D2) είναι δίοδος για προστασία υπέρτασης. Μετά υπάρχει ένας διαιρέτης τάσης αποτελούμενος από τις αντιστάσεις R3 και R4. Το σημείο A0 του διαιρέτη πηγαίνει στο αντίστοιχο pin του μικροελεγκτή για να διαβάζει την τάση εισόδου. Στον Arduino είναι το pin A0. Το κύκλωμα που ακολουθεί μετράει το ρεύμα που δίνει το φωτοβολταϊκό. Στο συγκεκριμένο σχέδιο είναι το ACS712, αλλά έχουν χρησιμοποιηθεί αρκετά διαφορετικά κυκλώματα (βλέπε παράρτημα Α και Γ). Η έξοδος του αισθητήρα ρεύματος οδηγεί το pin A1 του Arduino. Αντίστοιχα κυκλώματα με διαιρέτη τάσης (R6, R7) και αισθητήρα ρεύματος έχουν χρησιμοποιηθεί και στην έξοδο του κυκλώματος που οδηγούν αντίστοιχα τα pin A2 και A3 του Arduino. Εν σειρά με το πρώτο αισθητήρα ρεύματος έχει τοποθετηθεί μια δίοδος *Schottky (MBR2545CT) (ή άλλη μικρότερη των 6A στο nano)* ώστε να μπλοκάρεται η ροή ρεύματος από την μπαταρία προς το φωτοβολταϊκό την νύχτα (στο σχήμα Ε3.7.1 δεν φαίνεται αλλά φαίνεται στο σχήμα Ε4.1.1). Στην θέση της διόδου μπορεί να χρησιμοποιηθεί και ένα MOSFET με καλλίτερα αποτελέσματα. Το κυρίως κύκλωμα του **buck converter** ξεκινάει από τον πυκνωτή C8 και τελειώνει στην έξοδο του πηνίου L1. Η καρδιά του **buck converter** είναι το ολοκληρωμένο IR2104. Αυτό έχει διπλό σκοπό. Οδηγεί τα MOSFET (low and high side) δίνοντας την κατάλληλη τάση και τον κατάλληλο χρονισμό. Το IR2104 (pin 2) οδηγείται με PWM από το pin D9 του Arduino. Το pin 3 του IR2104 οδηγείται από το pin D8 του Arduino και από εδώ μπορούμε να απενεργοποιήσουμε το ολοκληρωμένο. Η δίοδος D1 και ο πυκνωτής C1 απαιτούνται σύμφωνα με τις οδηγίες του κατασκευαστή (bootstrap circuit). Το πρόγραμμα στον μικροελεγκτή ελέγχει το PWM duty-cycle και δεν είναι ποτέ 100%. Το Q4 (IRLZ44N) ελέγχει το φορτίο. Χρησιμοποιήθηκε αυτό το MOSFET γιατί θέλει οδήγηση TTL και δεν χρειάζεται driver ολοκληρωμένο ή τρανζίστορ.

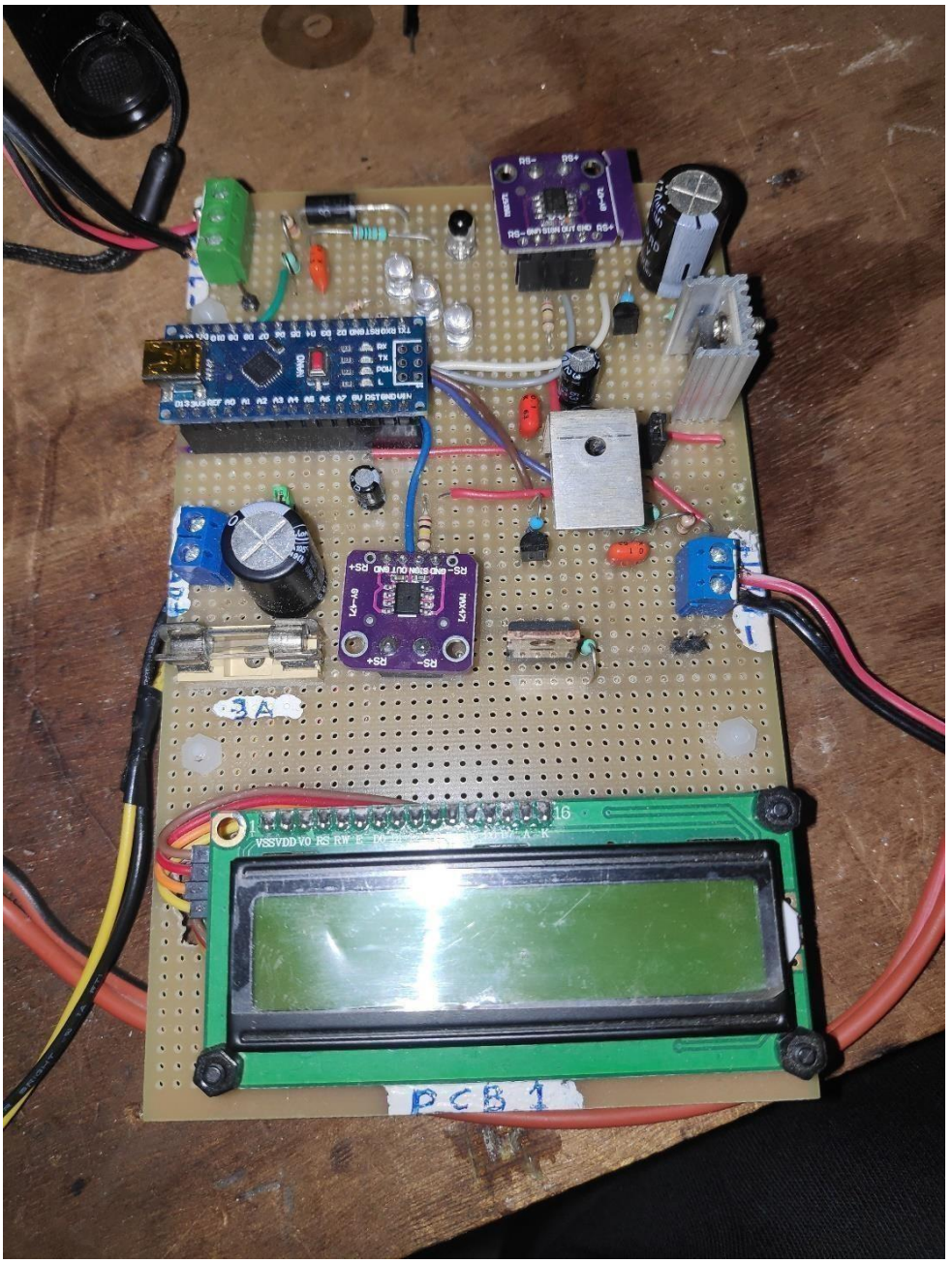
4 Υλοποίηση τελικών κυκλωμάτων με Arduino nano

4.1 Κατασκευή 1ης και 2ης πλακέτας

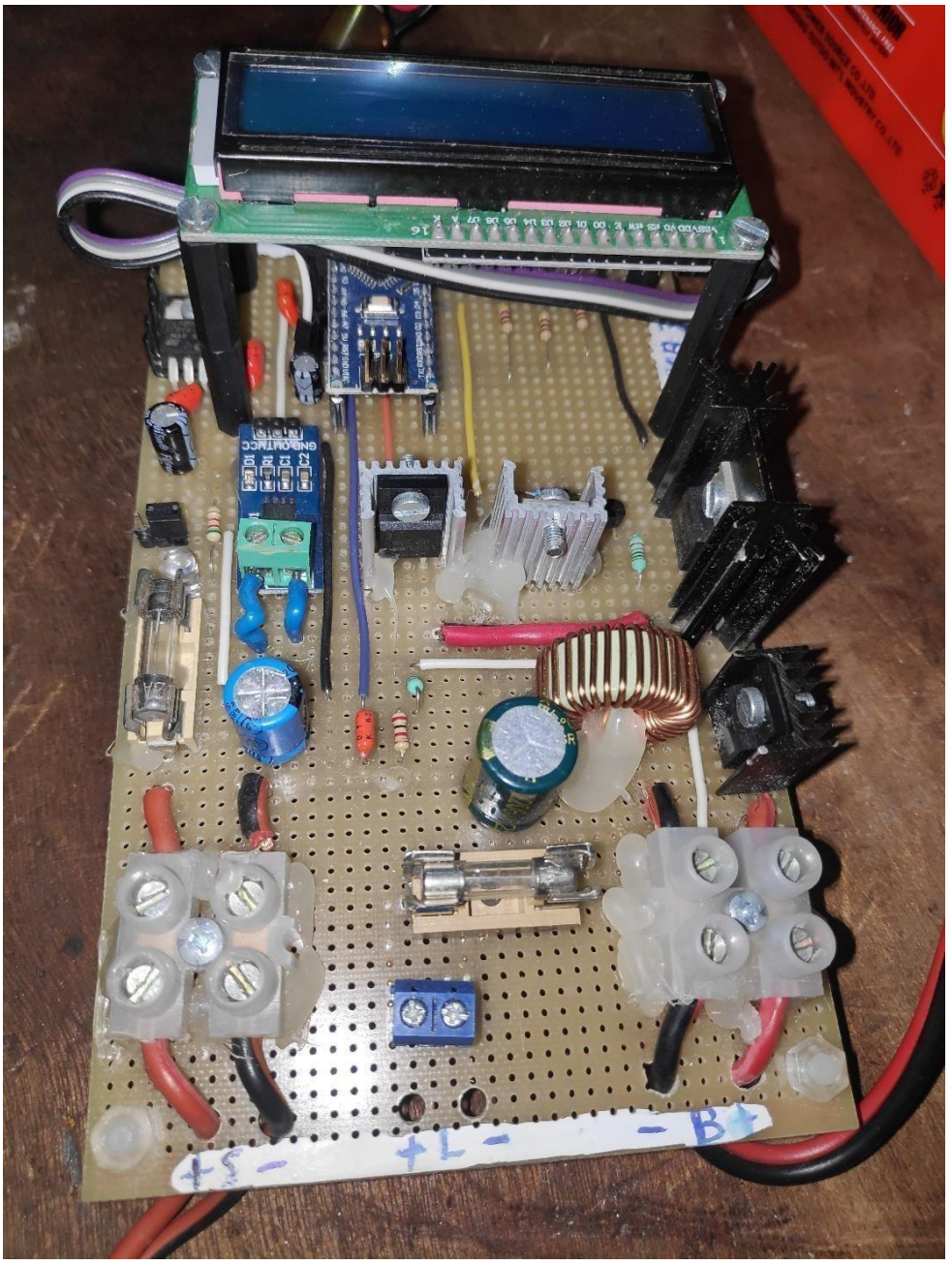
Συνολικά κατασκεύασα 3 πλακέτες . Ξεκινώντας από την πλακέτα 1 (PCB 1) εικόνα 4.1



Ε-4.1.1 Σχέδιο 1ης πλακέτας (PCB1)



E-4.1.2 Υλοποίηση 1ης πλακέτας PCB1



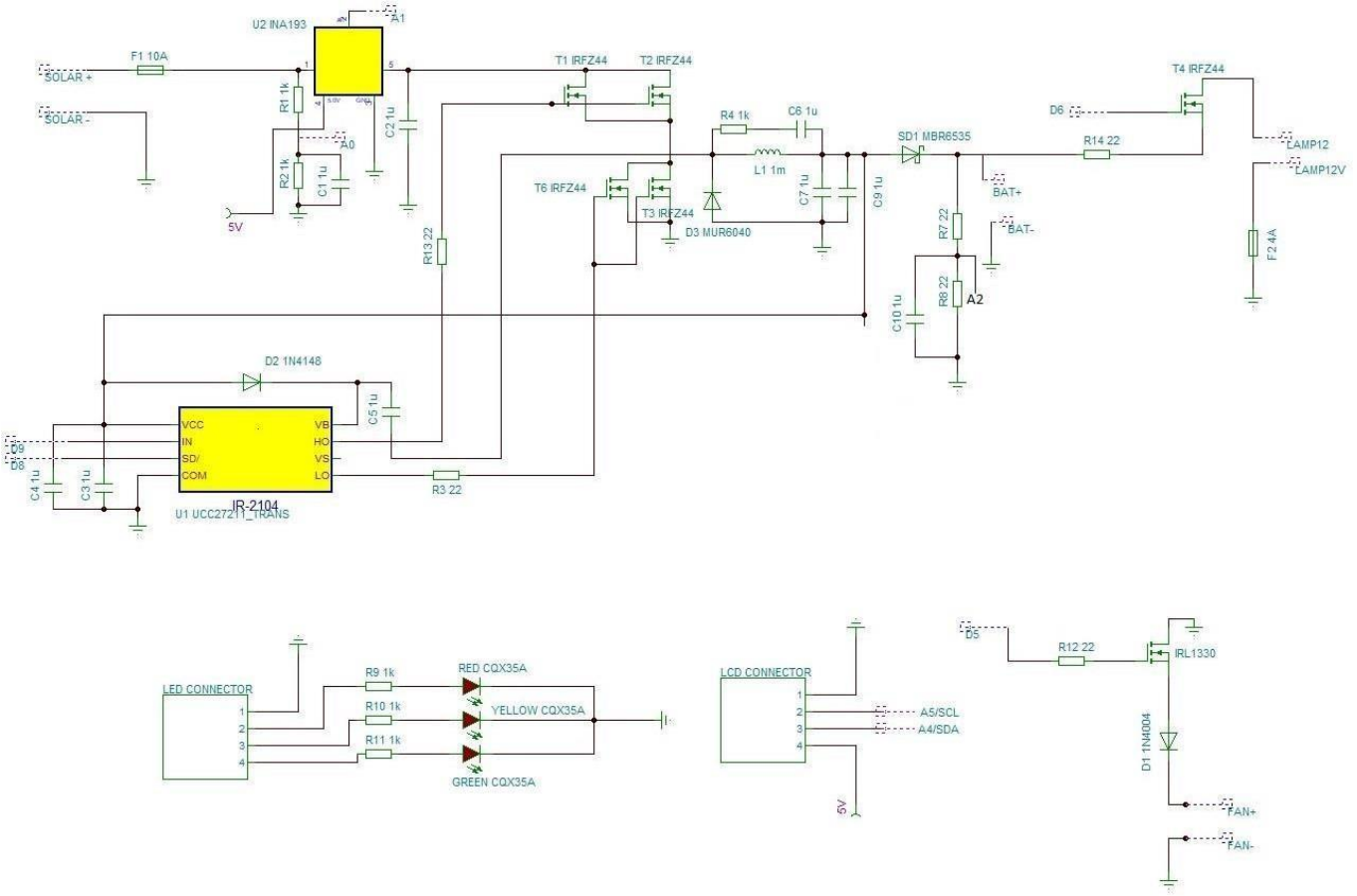
E-4.1.3 Υλοποίηση 2ης πλακέτας PCB2

. Στην 1η πλακέτα είναι ουσιαστικά μια απλή υλοποίηση ενός φορτιστή μπαταρίας , χωρίς dc/dc converter ο οποίος δουλεύει σε χαμηλές συχνότητες και βοηθάει αρκετά στην κατανόηση της εννοίας των παλμών (PWM) που δίνει ο μικροεπεξεργαστής . Το σχέδιο του κυκλώματος υπάρχει στην εικόνα 4.2 .Στον ακροδέκτη A0 του μικροεπεξεργαστή έχω τον a/d converter .Επειδή ο Arduino δέχεται μέχρι 5v έχω φτιάξει και έναν διαιρέτη τάσης χρησιμοποιώντας 2 αντιστάσεις R1=47 KΩ και R2=10KΩ. Στην συνέχεια έχω βάλει μια δίοδο (D1) η οποία εμποδίζει το ρεύμα το βραδύ να περάσει πίσω στο φωτοβολταΐκό. Από τον ακροδέκτη D3 του Arduino δίνω παλμούς στο IRF9540 το οποίο οδηγώ χρησιμοποιώντας ένα τρανζίστορ (2N3904) . Για την επιλογή του τρανζίστορ χρησιμοποίησα ένα από το ποιο γνωστά στην αγορά , το οποίο έκανε την ενίσχυση που χρειαζόμουν για να λειτουργήσει το mosfet και ταυτόχρονα ήτανε και από τα πιο οικονομικά στην αγορά τους .Έπειτα χρησιμοποιώντας R5=47 KΩ και R6=10KΩ σαν διαιρέτη τάσης μετράω την

τάση στην έξοδο του mosfet (ουσιαστικά την τάση που δίνω στην μπαταρία) Στην συνέχεια συνδέω την μπαταρία , την οποία φορτίζω. Έχοντας πάλι χρησιμοποιήσει ένα τρανζίστορ και ένα Mosfet με ακριβώς τα ίδια χαρακτηριστικά με τα προηγούμενα ,τροφοδοτώ και το οποιαδήποτε φορτίο . (Στην έξοδο έχω βάλει και έναν μετρητή ρεύματος MAX471 όπως και στην αρχή για να μπορώ να βλέπω την απόδοση του κυκλώματος μου) . Για την τροφοδοσία του Arduino , επειδή θέλω σταθερά 5V για να λειτουργεί σωστά ο μικροεπεξεργαστής έχω χρησιμοποιήσει. Στους ακροδέκτες 1 και 3 του σταθεροποιητή τάσης (7805) έχω βάλει 2 πυκνωτές οι οποίοι μειώνουν την κυμάτωση της τάσης . Τέλος στους ακροδέκτες D2,D3,D4 του Arduino έχω βάλει 3 led (πράσινο , κόκκινο , κίτρινο) για να βλέπω κάθε φορά την τάση της μπαταρίας .

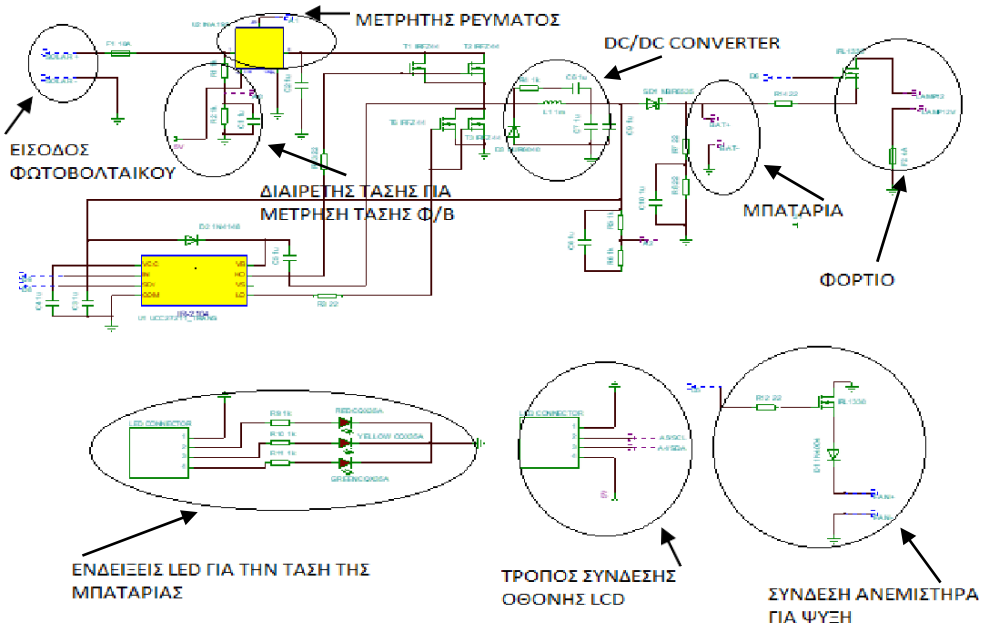
Η πλακέτα 2 (PCB2)(εικόνα 4.3) είναι παρόμοια με την πλακέτα 1 . Έχω χρησιμοποιήσει διαφορετικούς μετρητές ρεύματος (ACS 712) , αφού ο MAX471 μετράει μέχρι 3A και επειδή έχω προσθέσει και έναν dc/dc μετατροπέα (εικόνα 4.4) το κύκλωμα μπορεί να λειτουργήσει και για αρκετά μεγάλες συχνότητες (30KHz και άνω , ενώ το κύκλωμα 1 λειτουργούσε για 50 Hz). Και στα δυο αυτά κυκλώματα για επεξεργαστή έχω χρησιμοποιήσει τον Arduino nano

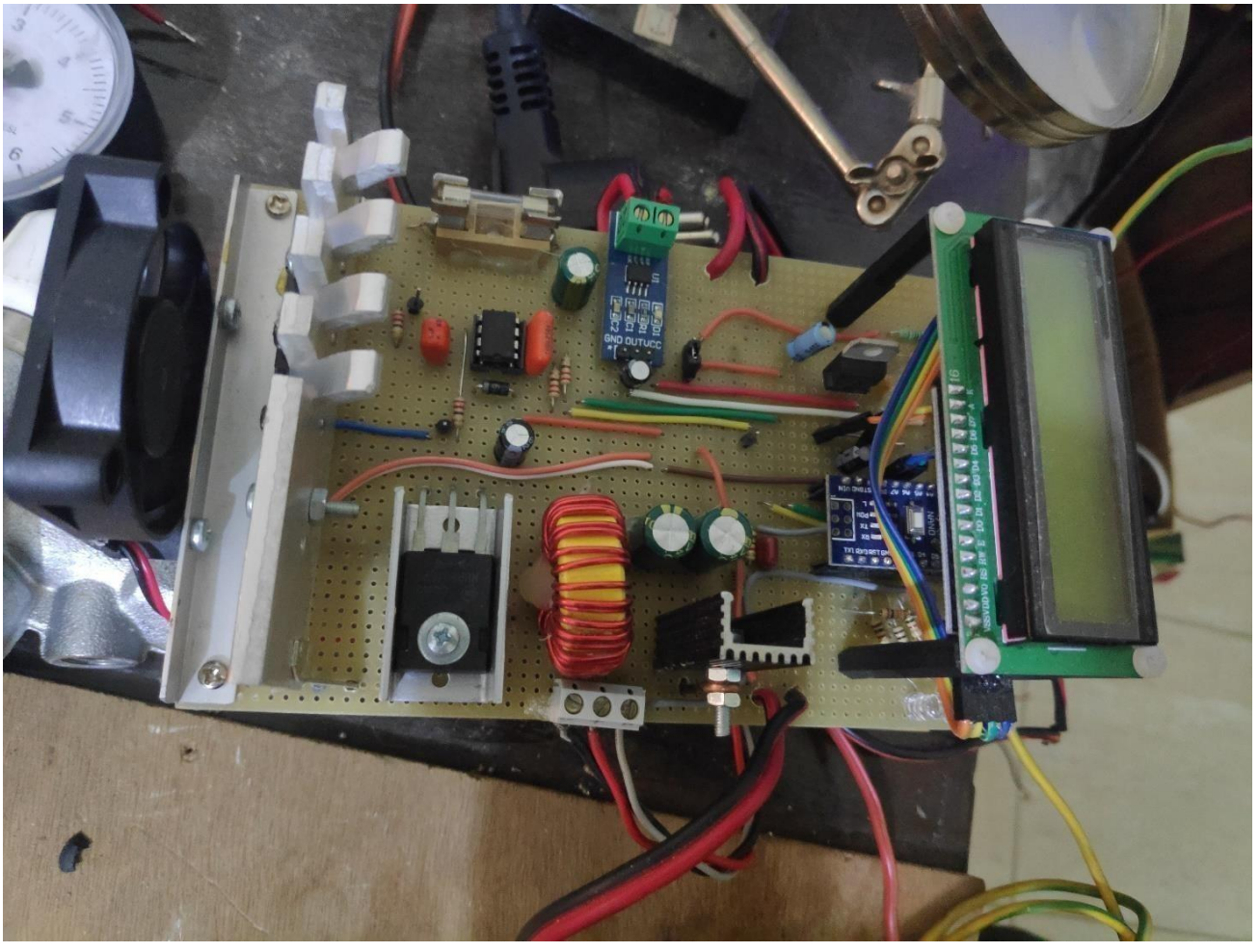
4.2 Κατασκευή 3ης πλακέτας



E-4.2.1 Σχέδιο 3ης πλακέτας(PCB3)

Πιο αναλυτικά :





Ε-4.2.1 Υλοποίηση 3ης πλακέτας(PCB3)

5 Προγραμματισμός μικροεπεξεργαστή (Arduino)-Αλγόριθμοι λειτουργίας

5.1 Αλγόριθμος λειτουργίας 1ης πλακέτας χωρίς MPPT

Αρχικά ο αλγόριθμος διαβάζει την τάση εισόδου μέσω ενός διαιρέτη τάσης που καταλήγει στο A/D converter (PIN A0) και της τάση της μπαταρίας με τον ίδιο τρόπο(PIN A2). Αν η τάσηεισόδου είναι μεγαλύτερη της τάσης εισόδου(δεν είναι νύχτα ή δεν έχει πολλά σύννεφα) , τότε ελέγχω την τάση της μπαταρίας ανάλογα με 3 επίπεδα. Το 1ο επίπεδο είναι το charged το οποίοείναι ορισμένο στα 13,8V . το 2ο επίπεδο είναι το fully charged το οποίο είναι ορισμένο στα 14.2V και το 3ο επίπεδο είναι το fully empty το οποίο είναι ορισμένο στα 11.9V. Επομένως εάν το φωτοβολταϊκό μπορεί να φορτίσει την μπαταριά έχω τα εξής στάδια φόρτισης. Αν η τάση της μπαταρίας είναι μικρότερη του charged (από 11.9V-13.8V) τότε αυξάνω το pwm με αρκετά μεγάλορυθμό ώστε να φτάσω γρήγορα την επιθυμητή τάση (13.8V). Αν η τάση της μπαταρίας είναι ανάμεσα στα 14.2V και στα 13.8V τότε αυξάνω με πολύ αργό βήμα το pwm . Όταν τελικά φτάσει ημπαταρία στο επίπεδο fully charged τότε μειώνω το Pwm με σκοπό να κρατήσω την τάση σταθερά στο επίπεδο fully charged .Όταν δεν έχω αρκετή ηλιοφάνεια μηδενίζω την τιμή του Pwm και δεν φορτίζει η μπαταρία. Για να αποφευχθεί οποιαδήποτε λάθος μέτρηση τάσης παίρνω 200 μετρήσεις και βγάζω τον μέσο όρο αυτών.

5.2 Αλγόριθμος λειτουργίας 1ης πλακέτας με MPPT

Ουσιαστικά ο αλγόριθμος αυτός λειτουργεί με τον ίδιο τρόπο απλά αυτή την φορά έχω εισάγει μια ακόμα παράμετρο, την προηγούμενη ισχύς. Όλες οι αλλαγές γίνονται στο στάδιο που αυξάνω με μεγάλο βήμα το pwm (δηλαδή από 11.9V μέχρι 13.8V).Συγκρίνω την προηγούμενη με τηντρέχουσα , εάν η τρέχουσα είναι μεγαλύτερη από την προηγούμενη τιμή τότε συνεχίζω να αυξάνω το Pwm (Δηλαδή συνεχίζω να αυξάνω την τάση κινούμενος πάνω στην καμπύλη της εικόνας **E- 2.1**). Όταν η προηγούμενη ισχύς είναι μεγαλύτερη σημαίνει ότι πέρασα το σημείο MPP οπότε αντί να συνεχίσω να προσθέτω το σταθερό βήμα στο Pwm το αφαιρώ(αλλάζω την φορά) και ξανά ελέγχω την ισχύς. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να φτάσω την επιθυμητή τάση . Στην συνέχεια ο αλγόριθμος λειτουργεί ακριβώς όπως και ο προηγούμενος. Αξίζει να σημειωθεί ότι αν η τάση της μπαταρίας για κάποιο λόγο πέσει στα 11.5V και κάτω απενεργοποιώ το φορτίο μου ,μέσω του (PIN A5) και του IRF9540 που ελέγχει το φορτίο.

5.3 Αλγόριθμος λειτουργίας 2 ης πλακέτας χωρίς MPPT

Η πλακέτα αυτή λειτουργεί με τον ίδιο ακριβώς τρόπο με την 1η. Αρχικά ο αλγόριθμος διαβάζει την τάση εισόδου μέσω ενός διαιρέτη τάσης που καταλήγει στο A/D converter (PIN A0) και της τάση της μπαταρίας με τον ίδιο τρόπο(PIN A2). Αν η τάση εισόδου είναι μεγαλύτερη της τάσης εισόδου(δεν είναι νύχτα ή δεν έχει πολλά σύννεφα) , τότε ελέγχω την τάση της μπαταρίας ανάλογα με 3 επίπεδα. Το 1ο επίπεδο είναι το charged το οποίο είναι ορισμένο στα 13,8V . το 2ο επίπεδο είναι το fully charged το οποίο είναι ορισμένο στα 14.2V και το 3ο επίπεδο είναι το fully empty το οποίο είναι ορισμένο στα 11.9V. Επομένως εάν το φωτοβολταϊκό μπορεί να φορτίσει την μπαταριά έχω τα εξής στάδια φόρτισης. Αν η τάση της μπαταρίας είναι μικρότερη του charged (από 11.9V-13.8V) τότε αυξάνω το pwm με αρκετά μεγάλο ρυθμό ώστε να φτάσω γρήγορα την επιθυμητή τάση (13.8V). Αν η τάση της μπαταρίας είναι ανάμεσα στα 14.2V και στα 13.8V τότε αυξάνω με πολύ αργό βήμα το pwm . Όταν τελικά φτάσει η μπαταρία στο επίπεδο fully charged τότε μειώνω

το Pwm με σκοπό να κρατήσω την τάση σταθερά στο επίπεδο fully charged .Όταν δεν έχω αρκετή ηλιοφάνεια μηδενίζω την τιμή του Pwm και δεν φορτίζει η μπαταρία. Για να αποφευχθεί οποιαδήποτε λάθος μέτρηση τάσης παίρνω 200 μετρήσεις και βγάζω τον μέσο όρο αυτών.

5.4 Αλγόριθμος λειτουργίας 2 ης πλακέτας με MPPT

Ουσιαστικά ο αλγόριθμος αυτός λειτουργεί με τον ίδιο τρόπο απλά αυτή την φορά έχω εισάγει μια ακόμα παράμετρο, την προηγούμενη ισχύς. Όλες οι αλλαγές γίνονται στο στάδιο που αυξάνω με μεγάλο βήμα το pwm (δηλαδή από 11.9V μέχρι 13.8V).Συγκρίνω την προηγούμενη με τηντρέχουσα , εάν η τρέχουσα είναι μεγαλύτερη από την προηγούμενη τιμή τότε συνεχίζω να αυξάνω το Pwm (Δηλαδή συνεχίζω να αυξάνω την τάση κινούμενος πάνω στην καμπύλη της εικόνας **E- 2.1**). Όταν η προηγούμενη ισχύς είναι μεγαλύτερη σημαίνει ότι πέρασα το σημείο MPP οπότε αντί να συνεχίσω να προσθέτω το σταθερό βήμα στο Pwm το αφαιρώ(αλλάζω την φορά) και ξανά ελέγχω την ισχύς. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να φτάσω την επιθυμητή τάση . Στην συνέχεια ο αλγόριθμος λειτουργεί ακριβώς όπως και ο προηγούμενος. Αξίζει να σημειωθεί ότι αν η τάση της μπαταρίας για κάποιο λόγο πέσει στα 11.5V και κάτω απενεργοποιώ το φορτίο μου ,μέσω του (PIN A5) και του IRF9540 που ελέγχει το φορτίο.

5.5 Αλγόριθμος λειτουργίας 3 ης πλακέτας χωρίς MPPT

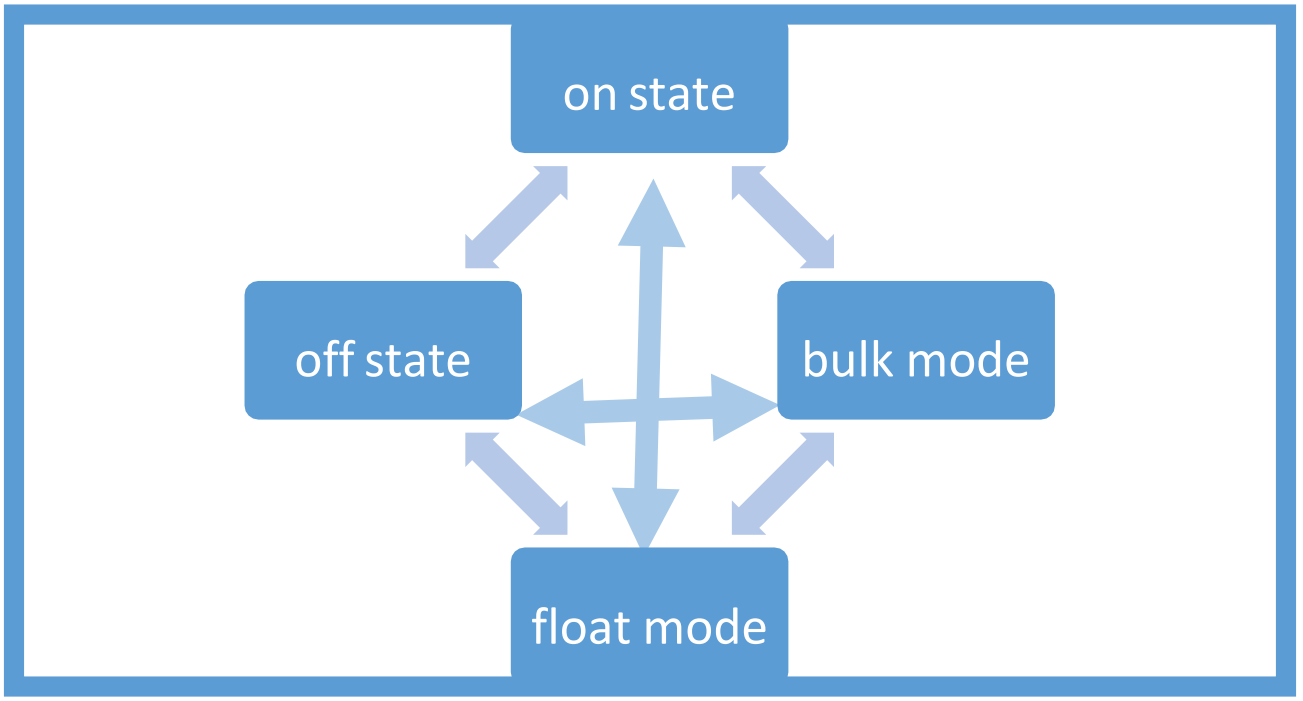
Η πλακέτα αυτή λειτουργεί με τον ίδιο ακριβώς τρόπο με την 1η. Αρχικά ο αλγόριθμος διαβάζει την τάση εισόδου μέσω ενός διαιρέτη τάσης που καταλήγει στο A/D converter (PIN A0) και της τάση της μπαταρίας με τον ίδιο τρόπο(PIN A2). Αν η τάση εισόδου είναι μεγαλύτερη της τάσης εισόδου(δεν είναι νύχτα ή δεν έχει πολλά σύννεφα) , τότε ελέγχω την τάση της μπαταρίας ανάλογα με 3 επίπεδα. Το 1ο επίπεδο είναι το charged το οποίο είναι ορισμένο στα 13,8V . το 2ο επίπεδο είναι το fully charged το οποίο είναι ορισμένο στα 14.2V και το 3ο επίπεδο είναι το fully empty το οποίο είναι ορισμένο στα 11.9V. Επομένως εάν το φωτοβολταϊκό μπορεί να φορτίσει την μπαταριά έχω τα εξής στάδια φόρτισης. Αν η τάση της μπαταρίας είναι μικρότερη του charged (από 11.9V-13.8V) τότε αυξάνω το pwm με αρκετά μεγάλο ρυθμό ώστε να φτάσω γρήγορα την επιθυμητή τάση (13.8V). Αν η τάση της μπαταρίας είναι ανάμεσα στα 14.2V και στα 13.8V τότε αυξάνω με πολύ αργό βήμα το pwm . Όταν τελικά φτάσει η μπαταρία στο επίπεδο fully charged τότε μειώνω το Pwm με σκοπό να κρατήσω την τάση σταθερά στο επίπεδο fully charged .Όταν δεν έχω αρκετή ηλιοφάνεια μηδενίζω την τιμή του Pwm και δεν φορτίζει η μπαταρία. Για να αποφευχθεί οποιαδήποτε λάθος μέτρηση τάσης παίρνω 200 μετρήσεις και βγάζω τον μέσο όρο αυτών.

5.6 Αλγόριθμος λειτουργίας 3ης πλακέτας χωρίς MPPT

Η πλακέτα αυτή λειτουργεί με τον ίδιο ακριβώς τρόπο με την 1η. Αρχικά ο αλγόριθμος διαβάζει την τάση εισόδου μέσω ενός διαιρέτη τάσης που καταλήγει στο A/D converter (PIN A0) και της τάση της μπαταρίας με τον ίδιο τρόπο(PIN A2). Αν η τάση εισόδου είναι μεγαλύτερη της τάσης εισόδου(δεν είναι νύχτα ή δεν έχει πολλά σύννεφα) , τότε ελέγχω την τάση της μπαταρίας ανάλογα με 3 επίπεδα. Το 1ο επίπεδο είναι το charged το οποίο είναι ορισμένο στα 13,8V . το 2ο επίπεδο είναι το fully charged το οποίο είναι ορισμένο στα 14.2V και το 3ο επίπεδο είναι το fully empty το οποίο είναι ορισμένο στα 11.9V. Επομένως εάν το φωτοβολταϊκό μπορεί να φορτίσει την μπαταριά έχω τα εξής στάδια φόρτισης. Αν η τάση της μπαταρίας είναι μικρότερη του charged (από 11.9V-

13.8V) τότε αυξάνω το pwm με αρκετά μεγάλο ρυθμό ώστε να φτάσω γρήγορα την επιθυμητή τάση (13.8V). Αν η τάση της μπαταρίας είναι ανάμεσα στα 14.2V και στα 13.8V τότε αυξάνω με πολύ αργό βήμα το pwm . Όταν τελικά φτάσει η μπαταρία στο επίπεδο fully charged τότε μειώνω το Pwm με σκοπό να κρατήσω την τάση σταθερά στο επίπεδο fully charged .Όταν δεν έχω αρκετή ηλιοφάνεια μηδενίζω την τιμή του Pwm και δεν φορτίζει η μπαταρία. Για να αποφευχθεί οποιαδήποτε λάθος μέτρηση τάσης παίρνω 200 μετρήσεις και βγάζω τον μέσο όρο αυτών.

5.7 Αλγόριθμος λειτουργίας 3 ης πλακέτας με MPPT



E-5.6.1 Επίπεδα φόρτισης

Όπως και στις προηγούμενες πλακέτες έτσι και σε αυτή διαβάζω την τάση εισόδου μέσω ενός διαιρέτη τάσης που καταλήγει στο A/D converter (PIN A0) και της τάση της μπαταρίας με τον ίδιο τρόπο(PIN A2).Σε αυτόν τον αλγόριθμο όμως έχω χρησιμοποιήσει διαφορετική σκέψη , χωρίς να είναι καλύτερο ή χειρότερο από τα προηγούμενα. Επίσης έχω εισάγει και κάποιες παραπάνω μεταβλητές όπως min_sol_watts(0.1W ελάχιστη ισχύς του φωτοβολταϊκού),

low_solaw_Watts(0.3W χαμηλή ισχύς φωτοβολταϊκού). Εκτός από τα 3 στάδια φόρτισης (που υπάρχουν και εδώ) έχω ορίσει 4 καταστάσεις οι οποίες συνδέονται όλες μεταξύ τους. Κάθε φορά από το main loop τρέχει και μία κατάσταση.

Κατάσταση ON STATE :

Στην κατάσταση **on state** μπαίνει όταν ισχύς που παράγει το φωτοβολταϊκό είναι μεταξύ των τιμών MIN_SOL_WATTS και LOW_SOL_WATTS. Δηλαδή ο φορτιστής μεταβαίνει στην κατάσταση αυτή όταν η παραγόμενη ισχύς είναι χαμηλή για να μετάβει στην bulk αλλά όχι και πολύ χαμηλή για απενεργοποίηση του φορτιστή (off state).Σε αυτή την κατάσταση βάζουμε topwm στο μέγιστο για να πάρουμε την λιγοστή ισχύ που παράγεται.

Κατάσταση BULK STATE : (FAST)

Ο φορτιστής μεταβαίνει στην κατάσταση αυτή **bulk state** όταν η παραγόμενη ισχύς είναι μεγαλύτερη από MIN_SOL_WATTS εδώ έχουμε ισχυρή φόρτιση μπαταρίας και βασίζεται στον αλγόριθμο MPPT. Η μπαταρία φορτίζεται με το μέγιστο δυνατό ρεύμα που μπορεί να δώσει εκείνη τη στιγμή το φωτοβολταϊκό.

Κατάσταση FLOAT STATE :

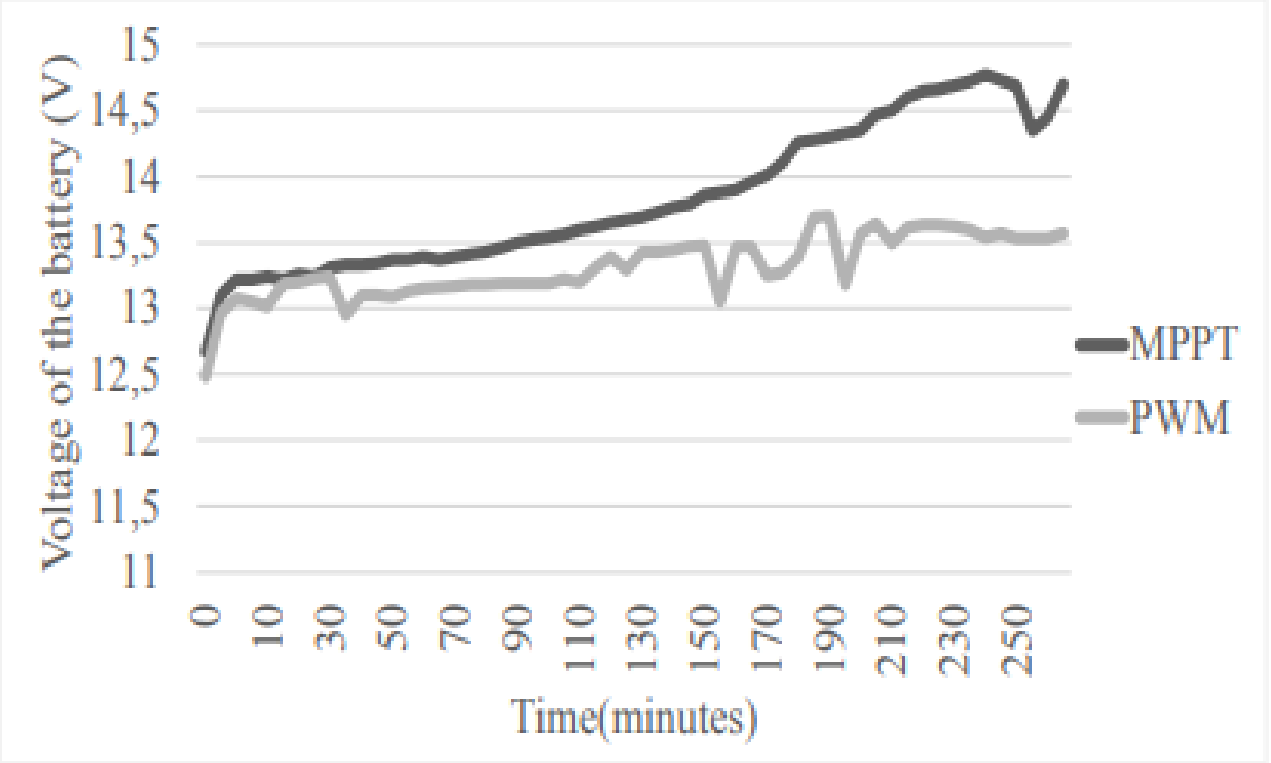
Στην κατάσταση αυτή **float state** καθώς φορτίζεται η μπαταρία η τάση της αυξάνει. Αν υπερβεί την καθορισμένη τιμή MAX_BAT_VOLTS αρχίζει η υπερφόρτιση και για προστασία μεταβαίνουμε στην κατάσταση float. Σε αυτή την κατάσταση κρατείται η τάση περίπου σταθερή στο MAX_BAT_VOLTS αυξομειώνοντας την τιμή pwm. Στο πρόγραμμα έχει τοποθετηθεί χωρίς μεταβλητή η τιμή μεταβολής 10 (υψηλότερη τιμή χάνουμε σε ακρίβεια αλλά κερδίζουμε σε χρόνο). Αν το pwm φθάσει το 1023 σημαίνει ότι δεν μπορεί η μπαταρία να κρατηθεί στο MAX_BAT_VOLTS (πιθανό λόγω φορτίου) και επιστρέφουμε στην κατάσταση bulk.

Κατάσταση OFF STATE :

Ο φορτιστής μπαίνει σε αυτή την κατάσταση όταν η παραγόμενη ισχύς από το φωτοβολταϊκό είναι μικρότερη από την προκαθορισμένη τιμή MIN_SOL_WATTS. Ουσιαστικά τα φωτοβολταϊκά δεν παράγουν ισχύ (πχ νύχτα) και τα FET μέσω του driver κλείνουν για να μην επιστρέφει ισχύς από την μπαταρία στα φωτοβολταϊκά.

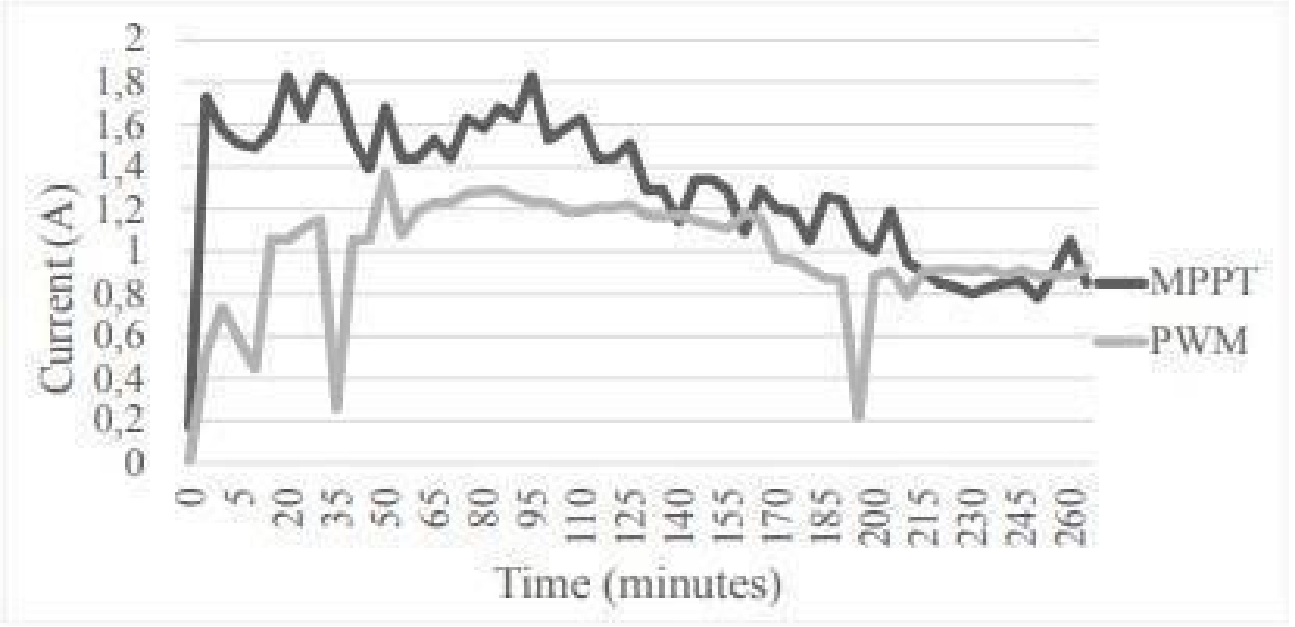
6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Σύγκριση-Πλεονεκτήματα σε σχέση ρυθμιστή φόρτισης τύπου PWM και τύπου MPPT



Ε-6.1 Διάγραμμα συνάρτησης τάσης-χρόνου στη διαδικασία φόρτισης της μπαταρίας με και χωρίς mppt

Όπως φαίνεται και στην εικόνα **Ε-6.1** κατά την λειτουργία φόρτισης με τον αλγόριθμο MPPT , τα πρώτα 200 περίπου λεπτά αντιστοιχούν στο bulk state (fast) , όπου η τάση της μπαταρίας αυξάνεται γραμμικά μέχρι να φτάσει περίπου τα 14,5V. Έπειτα ο ελεγκτής διατηρεί σταθερή την τιμή αυτήτης τάσης (float state) Στην περίπτωση του PWM κατά την διάρκεια των πρώτων 215 λεπτών ο ελεγκτής βρίσκεται στο στάδιο μαζικής αύξησης τάσης της μπαταρίας .Μόλις φτάσει στο στάδιο απορρόφησης , ο ελεγκτής διατηρεί την τάση στα 13,7V περίπου. Ο ελεγκτής MPPT παρέχει περισσότερο ρεύμα από το φωτοβολταϊκό στην μπαταρία και για αυτό η τάση αυξάνεται περισσότερο στο bulk state ενώ ταυτόχρονα έχω λιγότερες απώλειες ενέργειας.
[11]



Ε-6.2 Διάγραμμα του ρεύματος που παρέχεται στην μπαταρία σε σχέση με το χρόνο στη διαδικασία φόρτισης

Στην εικόνα Ε-6.2 το ρεύμα που παρέχεται από τον ελεγκτή MPPT μεγαλώνει πολύ γρήγορα (μέσα στα πρώτα 5 λεπτά) ,φτάνει τα 1.8Α ενώ μέχρι τα πρώτα 210 περίπου λεπτά το ρεύμα είναι πάνω από 1Α .Στην περίπτωση του ελεγκτή PWM το ρεύμα αυξάνεται σε τιμές 1-1.2Α κατά τα πρώτα 210 λεπτά, όταν δηλαδή λειτουργεί στο στάδιο μαζικής φόρτισης (bulk state) και μόλις φτάσει στο float state το ρεύμα μειώνεται ελαφρώς .Στο 35 όπως και στο 195 λεπτό το ρεύμα που παρέχεται στην μπαταρία μειώνεται δραστικά , επειδή δεν θα υπήρξε επαρκής ηλιοφάνεια την στιγμή των μετρήσεων και ο ελεγκτής PWM δεν μπορεί να μεγιστοποιήσει την ισχύ οπότε είχαμε αυτή την πτώση που βλέπουμε στην παραπάνω εικόνα (Ε-6.2) .
[12]

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] http://ww1.microchip.com/downloads/en/AppNotes/01211B.pdf

[2] http://ww1.microchip.com/downloads/en/AppNotes/00001521A.pdf

[3].https://www.homemade-circuits.com/mppt-circuit-using-pic16f88-with-3/

[4] https://en.wikipedia.org/wiki/Buck_converter

[5] https://latexdraw.com/dc-dc-buck-converter-latex-circuitikz/

[6] http://mixanikos365.blogspot.com/2019/08/hd-384.html

[7] https://datasheet.datasheetarchive.com/originals/dk/DKDS-31/600068.pdf

[8] https://www.electronicscomp.com/ir2104-half-bridge-driver-ic

[9].https://www.componentsinfo.com/irfz44n-pinout-equivalent/

[10] https://www.eeweb.com/inductor-calculation-for-buck-converter-ic/

[11] https://www.researchgate.net/figure/Comparison-of-voltage-on-the-battery-versus-time-in-the-charging-process_fig3_332879967

[12] https://www.redalyc.org/journal/849/84959429001/html/

[13] Atik, L., Petit, P., Sawicki, J. P., Ternifi, Z. T., Bachir, G., Della, M., & Aillerie, M. (2017). *Maximum power point tracking algorithm based on sliding mode and fuzzy logic for photovoltaic sources under variable environmental conditions.*

[14] Atik, L., Petit, P., Sawicki, J. P., Ternifi, Z. T., Bachir, G., & Aillerie, M. (2016). *Comparison of four MPPT techniques for PV systems.*

[15] Zegaoui, A., Aillerie, M., Petit, P., Sawicki, J. P., Jaafar, A., Salame, C., & Charles, J. P. (2011). *Comparison of Two Common Maximum Power Point Trackers by Simulating of PV Generators. Energy Procedia, 6, 678–687.*

[16] Atik, L., Petit, P., Sawicki, J. P., Ternifi, Z. T., Bachir, G., & Aillerie, M. (2016). *Comparison of four MPPT techniques for PV systems.*

[17 Zegaoui, A., Aillerie, M., Petit, P., Sawicki, J. P., Charles, J. P., & Belarbi, A. W. (2011). *Dynamic behaviour of PV generator trackers under irradiation and temperature changes. Solar Energy, 85(11), 2953–2964.*

[18] Podder, A. K., Roy, N. K., & Pota, H. (2019). *MPPT Methods for Solar PV Systems: A Critical Review Based on Tracking Nature. IET Renewable Power Generation.*

[19] Petit, P., Zgaoui, A., Sawicki, J.-P., Aillerie, M., & Charles, J.-P.

(2011). *New architecture for high efficiency DC-DC converter dedicated to photovoltaic conversion. Energy Procedia, 6, 688–694.*

[20] Hohm, D. P., & Ropp, M. E. (2002). *Comparative study of maximum power point tracking algorithms. Progress in Photovoltaics: Research and Applications, 11(1), 47–62.*

[21] Olusegun, A. T., Adebukola, A. Z., Denwigwe, I. H., Oluseyi, P. O., & Olubayo, B. M. (2019). *Comparative Analysis of Two Direct MPPT Methods Used for Tracking Maximum Power Points in a Photovoltaic System. World Scientific News*, (131), 123–146.

[22] Karami, N., Moubayed, N., & Outbib, R. (2017). *General review and classification of different MPPT Techniques. Renewable and Sustainable Energy Reviews, 68, 1–18.*

[23] Bendib, B., Belmili, H., & Krim, F. (2015). *A survey of the most used MPPT methods: Conventional and advanced algorithms applied for photovoltaic systems. Renewable and Sustainable Energy Reviews, 45, 637–648.*

Παράρτημα Α

Α1)Αλγόριθμος λειτουργίας 1ης πλακέτας χωρίς MPPT

```
/*
  V2_R1
volt input sensor pin A0
current input sensor MAX471 pin A1
volt output sensor pin A2
current output sensor MAX471 pin A3

f=64Hz
 */

#include "TimerOne.h"
#include <LiquidCrystal_I2C.h>

#define sol_VOLT A0     // Solar panel side voltage divider is connected to pin A0
#define sol_CURRENT A1 // Solar panel MAX471 current sensor is connected to pin A1
#define bat_VOLT A2     // Load side voltage divider is connected to pin A2
#define bat_CURRENT A3  // Bat MAX471 current sensor is connected to pin A3

#define AVG_NUM 200     // number of iterations of the adc routine to average the adc readings

#define LED_RED 3
#define LED_GREEN 2
#define LED_YELLOW 4
#define LOAD_ENABLE 5     // pin-5 is used to enable LOAD
#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET

//-------------------------------------------------------------------------------
-----
//////////////////////DECLARATION OF ALL GLOBAL
VARIABLES////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------
-----
```

```
float solar_volt=0;
float   bat_volt=0;
float solar_current=0;
float bat_current=0;
float charge_status=0;
float  watts_in=0;
float watts_in_old=0;
float watts_out=0;

//battery
float fully_charged = 14.2;
float charged =       13.8;
float fully_empty =  11.9;
//int  bat_status=1;

int pwm_value = 0;
int counter1 = 0;
int pwm_step_inc = 21;

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** SETUP PROGRAM START
***********************************************
void setup()
{
 pinMode(LOAD_ENABLE,OUTPUT);
 pinMode(PWM_PIN,OUTPUT);
 pinMode(LED_RED,OUTPUT);
 pinMode(LED_GREEN,OUTPUT);
 pinMode(LED_YELLOW,OUTPUT);

 Timer1.initialize(15000); // set a timer of period length 15000uS f=64Hz
 Timer1.pwm(PWM_PIN, 0); // stop fet
 digitalWrite(LOAD_ENABLE, LOW); //STOP LOAD
```

```
  digitalWrite(LED_GREEN, LOW); //turn off leds
  digitalWrite(LED_YELLOW, LOW);
  digitalWrite(LED_RED, LOW);

  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines

  Serial.begin(9600); //serial speed to pc
  delay(100);
}

//********************** MAIN LOOP
***********************************************
void loop()
{
  delay(2000);
  read_data();
  power_calc();
  print_data();
  lcd_display();
  led_control();
  load_control();
  charger();
}

//********************** read volt currents
***********************************************
int read_adc(int adc_parameter)
{
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                              // loop through reading raw adc values AVG_NUM number of times
    sample = analogRead(adc_parameter);    // read the input pin
    sum += sample;                    // store sum for averaging
    delayMicroseconds(500);
```

```
  }
  return(sum / AVG_NUM);              // divide sum by AVG_NUM to get average and return it
}

//********************** read adc ***********************************************
void read_data(void)
{
  // 20kohms + 5,6kohms  works OK for solar
  solar_volt = read_adc(sol_VOLT) * (5.0 / 1023.0) * 5.75;//
  delayMicroseconds(500);

   // 3A max471 solar
  solar_current = (read_adc(sol_CURRENT)*(5.0/1023.0));
  delayMicroseconds(500);

  // 20kohms + 5,6kohms  works OK for battery
  bat_volt = read_adc(bat_VOLT) * (5.0 / 1023.0) * 5.75;
  delayMicroseconds(500);

  // 3A max471 battery
  bat_current = (read_adc(bat_CURRENT)*(5.0/1023.0));
  delayMicroseconds(500);
}
//********************** POWER AND ENERGY CALCULATION
***********************************************
void power_calc(void){
  watts_in = solar_current * solar_volt; //Watts
  watts_out = bat_current * bat_volt; //Watts
}
//********************** show data in pc serial monitor
***********************************************
void charger(void)
{
    if(solar_volt > bat_volt){
      if (bat_volt < charged){
```

```
      pwm_value+=pwm_step_inc;
      Serial.println("FAST");
     }
     else {
      if ((bat_volt > charged)&&(bat_volt < fully_charged)){
        pwm_value+=1;
      }
      else{
         pwm_value-=9;
         if (pwm_value<0){pwm_value=0;}
         if (bat_volt > fully_charged + 0.4){pwm_value=0;}
      }
      Serial.println("SLOW");
     }
    }

   // shut down when battery is fully charged or when sunlight is not enough
    else{
     pwm_value=0;
     Serial.println("NOT charging");
    }
    Serial.print("next pwm_value : ");
    Serial.println(pwm_value);
    Timer1.pwm(PWM_PIN, pwm_value); //
    if (pwm_value>999){pwm_value=800;}
    if (pwm_value>900 && watts_in<0.5){pwm_value=0;}
}
//********************** show data in pc serial monitor
***********************************************
void print_data(void)
{
  Serial.println("*****************************");
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
  Serial.println("V");
```

```
  Serial.print("Battery Voltage      : ");
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes : ");
  Serial.print(solar_current,2);
  Serial.println("A");
  Serial.print("Battery Amperes      : ");
  Serial.print(bat_current,2);
  Serial.println("A");
  Serial.print("Watt in     : ");
  Serial.print(watts_in,2);
  Serial.println("W");
  Serial.print("Watt in old : ");
  Serial.print(watts_in_old,2);
  Serial.println("W");
  Serial.print("Watt out    : ");
  Serial.print(watts_out,2);
  Serial.println("W");
  Serial.print("pwm value      : ");
  Serial.println(pwm_value);
}

//-------------------------------------------------------------------------------
////////////////////////////////////SYSTEM VOLTAGE AUTO DETECT
//////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------
/*
void system_voltage(void)
 {
 if ((bat_volt >BAT_MIN) && (bat_volt < BAT_MAX))
 {
 system_volt = 12;
 }

 else if  ((bat_volt > BAT_MIN*2 ) && (bat_volt < BAT_MAX*2))
```

```
 {
 system_volt=24;
 }
 else if ((bat_volt > BAT_MIN/2 ) && (bat_volt < BAT_MAX/2))
 {
 system_volt=6;
 }

 }
 */
//********************** LOAD CONTROL
***********************************************
void load_control()
{
  if ((bat_volt > fully_empty)&&(charge_status==1)){ // check if battery is healthy
    digitalWrite(LOAD_ENABLE, HIGH); // load is ON
    charge_status=1;
  }
  else if ((bat_volt > fully_empty + 0.5)&&(charge_status==0)){
    charge_status=1;
  }
  else{
    digitalWrite(LOAD_ENABLE, LOW); //load is OFF
    charge_status=0;
  }
}
//********************** LED CONTROL
***********************************************
void led_control()
{
  if (bat_volt > charged){ // check if battery is healthy
    digitalWrite(LED_YELLOW, HIGH);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
  }
```

```
  else if(bat_volt > fully_empty){
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, HIGH);
  }
  else{
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, HIGH);
    digitalWrite(LED_GREEN, LOW);
  }
}
//********************** LCD DISPLAY
************************************************
void lcd_display()
{
 delay(1);
 //lcd.noBacklight();// backlight OFF (flash)
 lcd.clear(); //clears LCD
 delay(1);
 lcd.backlight();//  backlight on

 lcd.setCursor(0, 0);
 lcd.print("SOL=");
 lcd.setCursor(4, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 //lcd.setCursor(0, 2);
 //lcd.print(solar_current);
 //lcd.print("A");
 lcd.setCursor(10, 0);
 lcd.print("P=");
 lcd.setCursor(12, 0);
 lcd.print((int)watts_in);
 //lcd.print(watts_in,1);
 lcd.print("W ");
```

```
 lcd.setCursor(0, 1);
 lcd.print("BAT=");
 lcd.setCursor(4, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(9,1);
 lcd.print("pwm=");
 lcd.setCursor(13,1);
 lcd.print(pwm_value);
}
```

Α2)Αλγόριθμος λειτουργίας 1ης πλακέτας με MPPT

```
/*
volt input sensor pin A0
current input sensor MAX471 pin A1
volt output sensor pin A2
current output sensor MAX471 pin A3
 */

//#include "Wire.h"
#include "TimerOne.h"
#include <LiquidCrystal_I2C.h>

#define sol_VOLT A0      // Solar panel side voltage divider is connected to pin A0
#define sol_CURRENT A1 // Solar panel MAX471 current sensor is connected to pin A1
#define bat_VOLT A2     // Load side voltage divider is connected to pin A2
#define bat_CURRENT A3  // Bat MAX471 current sensor is connected to pin A3

#define AVG_NUM 200     // number of iterations of the adc routine to average the adc readings

#define LED_RED 3
#define LED_GREEN 2
#define LED_YELLOW 4
#define LOAD_ENABLE 5      // pin-5 is used to enable LOAD
```

```
#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET

//------------------------------------------------------------------------
/////////////////////DECLARATION OF ALL GLOBAL
VARIABLES//////////////////////////////////////////////////////////////////
//------------------------------------------------------------------------
float solar_volt=0;
float   bat_volt=0;
float solar_current=0;
float bat_current=0;
float charge_status=0;
float  watts_in=0;
float watts_in_old=0;
float watts_out=0;

//battery
float fully_charged = 14.2;
float charged =       13.8;
float fully_empty =  11.5;

int pwm_value = 0;
int counter1 = 0;
int mppt_step = 21;

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** SETUP PROGRAM START
**********************************************
void setup()
{
 pinMode(LOAD_ENABLE,OUTPUT);
 pinMode(PWM_PIN,OUTPUT);
 pinMode(LED_RED,OUTPUT);
```

```
  pinMode(LED_GREEN,OUTPUT);
  pinMode(LED_YELLOW,OUTPUT);

  Timer1.initialize(15000); // set a timer of period length 15000uS f=64Hz
  Timer1.pwm(PWM_PIN, 0); // stop fet
  digitalWrite(LOAD_ENABLE, LOW); //STOP LOAD

  digitalWrite(LED_GREEN, LOW); //turn off leds
  digitalWrite(LED_YELLOW, LOW);
  digitalWrite(LED_RED, LOW);

  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines

  Serial.begin(9600); //serial speed to pc
  delay(100);
}

//********************** MAIN LOOP
***********************************************
void loop()
{
  delay(3000);
  read_data();
  power_calc();
  print_data();
  lcd_display();
  led_control();
  load_control();
  charger();
}

//********************** read volt currents
***********************************************
int read_adc(int adc_parameter)
{
```

```
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                                  // loop through reading raw adc values AVG_NUM number of times
   sample = analogRead(adc_parameter);    // read the input pin
   sum += sample;                    // store sum for averaging
   delayMicroseconds(500);
  }
  return(sum / AVG_NUM);              // divide sum by AVG_NUM to get average and return it
}

//********************** read adc ***********************************************
void read_data(void)
{
  // 20kohms + 5,6kohms  works OK for solar
  solar_volt = read_adc(sol_VOLT) * (5.0 / 1023.0) * 5.75;//
  delayMicroseconds(500);

   // 3A max471 solar
  solar_current = (read_adc(sol_CURRENT)*(5.0/1023.0) * 0.7);
  delayMicroseconds(500);

  // 20kohms + 5,6kohms  works OK for battery
  bat_volt = read_adc(bat_VOLT) * (5.0 / 1023.0) * 5.7;
  delayMicroseconds(500);

  // 3A max471 battery
  bat_current = (read_adc(bat_CURRENT)*(5.0/1023.0) * 0.7);
  delayMicroseconds(500);
}
//********************** POWER AND ENERGY CALCULATION
***********************************************
void power_calc(void){
  watts_in = solar_current * solar_volt; //Watts
  watts_out = bat_current * bat_volt; //Watts
```

```
}
//********************** show data in pc serial monitor
**********************************************
void charger(void)
{
 // float charge when battery is charged and when sunlight is enough
   if(solar_volt > bat_volt){
    if (bat_volt < charged){  //bulk charge
     if (watts_in < watts_in_old){mppt_step=-mppt_step;}

     pwm_value+=mppt_step;
     watts_in_old=watts_in;
     Serial.println(mppt_step);
    }
    else  {
     watts_in_old = 0;
     if (mppt_step<0){mppt_step=-mppt_step;} // change if still < 0
     if (bat_volt < fully_charged){  //float charge
      pwm_value+=1;
     }
     else{
      if (bat_volt > fully_charged + 0.6){
       pwm_value=0;
      }
      else{
       pwm_value-=7;
       if (pwm_value<0){pwm_value=0;}
      }
     }
    }
 }

 else{ // shut down when battery is fully charged or when sunlight is not enough
  pwm_value=0;
  Serial.println("NOT charging");
```

```
}
  Serial.print("next pwm_value : ");
  Serial.println(pwm_value);
  Timer1.pwm(PWM_PIN, pwm_value); //
  if (pwm_value>1000){pwm_value=800;}
  if (pwm_value<0){pwm_value=0;}
  if (pwm_value>900 && watts_in<0.5){pwm_value=0;}
}
//********************** show data in pc serial monitor
***********************************************
void print_data(void)
{
  Serial.println("*******************************");
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
  Serial.println("V");
  Serial.print("Battery Voltage      : ");
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes : ");
  Serial.print(solar_current,2);
  Serial.println("A");
  Serial.print("Battery Amperes      : ");
  Serial.print(bat_current,2);
  Serial.println("A");
  Serial.print("Watt in     : ");
  Serial.print(watts_in,3);
  Serial.println("W");
  Serial.print("Watt in old : ");
  Serial.print(watts_in_old,3);
  Serial.println("W");
  Serial.print("Watt out    : ");
  Serial.print(watts_out,2);
  Serial.println("W");
  Serial.print("pwm value      :");
```

```
  Serial.println(pwm_value);
}

//-------------------------------------------------------------------------
////////////////////////////////SYSTEM VOLTAGE AUTO DETECT
//////////////////////////////////////////////////////////
//-------------------------------------------------------------------------
/*
void system_voltage(void)
 {
 if ((bat_volt >BAT_MIN) && (bat_volt < BAT_MAX))
 {
 system_volt = 12;
 }

 else if  ((bat_volt > BAT_MIN*2 ) && (bat_volt < BAT_MAX*2))
 {
 system_volt=24;
 }
 else if ((bat_volt > BAT_MIN/2 ) && (bat_volt < BAT_MAX/2))
 {
 system_volt=6;
 }

 }
 */

//********************** LOAD CONTROL
***********************************************
void load_control()
{
  if ((bat_volt > fully_empty)&&(charge_status==1)){ // check if battery is healthy
    digitalWrite(LOAD_ENABLE, HIGH); // load is ON
    charge_status=1;
  }
```

```
  else if ((bat_volt > fully_empty + 0.5)&&(charge_status==0)){
    charge_status=1;
  }
  else{
    digitalWrite(LOAD_ENABLE, LOW); //load is OFF
    charge_status=0;
  }
}
//********************** LED CONTROL
***********************************************
void led_control()
{
  if (bat_volt > charged){ // check if battery is healthy
    digitalWrite(LED_YELLOW, HIGH);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
  }
  else if(bat_volt > fully_empty){
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, HIGH);
  }
  else{
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, HIGH);
    digitalWrite(LED_GREEN, LOW);
  }
}
//********************** LCD DISPLAY
***********************************************
void lcd_display()
{
 delay(1);
 //lcd.noBacklight();// backlight OFF (flash)
 lcd.clear(); //clears LCD
```

```
 delay(1);
 lcd.backlight();//  backlight on


 lcd.setCursor(0, 0);
 lcd.print("SOL=");
 lcd.setCursor(4, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 //lcd.setCursor(0, 2);
 //lcd.print(solar_current);
 //lcd.print("A");
 lcd.setCursor(10, 0);
 lcd.print("P=");
 lcd.setCursor(12, 0);
 lcd.print((int)watts_in);
 //lcd.print(watts_in,1);
 lcd.print("W ");
 lcd.setCursor(0, 1);
 lcd.print("BAT=");
 lcd.setCursor(4, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(9,1);
 lcd.print("pwm=");
 lcd.setCursor(13,1);
 lcd.print(pwm_value);
}
//********************** LCD DISPLAY
************************************************
void lcd_display1()
{
 delay(1);
 //lcd.noBacklight();// backlight OFF (flash)
 lcd.clear(); //clears LCD
 delay(1);
```

```
 lcd.backlight();//  backlight on


 lcd.setCursor(0, 0);
 lcd.print("SOL=");
 lcd.setCursor(4, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 //lcd.setCursor(0, 2);
 //lcd.print(solar_current);
 //lcd.print("A");
 lcd.setCursor(10, 0);
 //if (charge_status==0){lcd.println("off  ");};
 //if (charge_status==1){lcd.println("bulk  ");};
 //if (charge_status==2){lcd.println("float ");};
 lcd.setCursor(0, 1);
 lcd.print("BAT=");
 lcd.setCursor(4, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(9,1);
 lcd.print("pwm=");
 lcd.setCursor(13,1);
 lcd.print(pwm_value);
}
```

Παράρτημα Β

Β1) Αλγόριθμος λειτουργίας 2ης πλακέτας(ΧΩΡΙΣ MPPT)

```
/*
 ΠΑΔΑ
 no MPPT
 arduino NANO + LCD 20X2 I2C + BATTERY πλακέτα
 σειρά συνδεσμολογίας 1. μπαταρία 2. usb to pc (προαιρετικο) 3. solar panel

 το jumper στην πλακέτα να ειναι τοποθετημένο προς την πλευρά της μπαταρίας
```

για να δίνει τάση στο κύκλωμα η μπαταρία - στην αλλη πλευρά ολες οι τάσεις είναι
από το φωτοβολταϊκο

Ο αισθητήρας ρεύματος του φωτοβολταικου βασίζεται (20A acs712)
στο πιν A1 του arduino

volt input sensor pin A0
current input sensor pin A1
volt output sensor pin A2
timer 2000uS (συχνοτητα λειτουργιας 500Hz)
lcd = i2c 16X2 χρησιμοποιει τα A4 και A5
A4 (SDA), A5 (SCL) addreess= 0x27
*/

```
#include "Wire.h"
#include <math.h>
// χρήση της βιβλιοθήκης LCD I2C απο https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
#include <LiquidCrystal_I2C.h>
#include "TimerOne.h"

#define SOL_VOLT A0     // Πιν για μέτρηση της τασης του φωτοβολταϊκου μέσω του διαιρέτη τάσης (adc)
#define SOL_CURRENT A1 // INA 282 current sensor φωτοβολταϊκού στο pin A1
#define BAT_VOLT A2    // μέτρηση τάσης φορτίου στο pin A2
//#define THERMISTOR A3 // ΘΕΡΜΟΚΡΑΣΙΑ στο pin A3

#define AVG_NUM 100 // αριθμος επαναλήψεων για υπολογισμό μέσου όρου μετρώμενων τιμων απο την adc routine

#define MIN_BAT_VOLTS 11.80       //οριζόμενη ελαχιστη τιμη τάσης μπαταρίας
#define HIGH_BAT_VOLTS 13.80      //οριζόμενη επιθυμητή τιμη τάσης μπαταρίας
#define MAX_BAT_VOLTS 14.00       //οριζόμενη μέγιστη τιμη τάσης μπαταρίας

#define TURN_ON_FETS digitalWrite(PWM_ENABLE, HIGH)       // enable 2104 and FETS
```

```
#define TURN_OFF_FETS digitalWrite(PWM_ENABLE, LOW)       // disable 2104 and FETS

#define LED_RED 2
#define LED_GREEN 3
#define LED_YELLOW 4
//#define FAN_PIN 5
//#define LOAD_ENABLE 6         // pin-6 is used to enable LOAD
#define PWM_ENABLE 8         // pin-8 is used to enable driver
#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET //the default frequency
is about 490Hz
#define PWM_MAX 1020      // the value for maximum pwm duty cyle
#define PWM_MIN 5      // the value for minimum pwm duty cyle
#define PWM_START 20 // the value for initial pwm duty cyle
#define PWM_INC 11       //τιμη αυξησης ή μείωσης pwm για υλοποιηση αλγοριθμου mppt

//------------------------------------------------------------------------------
//ΟΡΙΣΜΟΣ GLOBAL VARIABLES
//------------------------------------------------------------------------------
float solar_volt=0;
float  bat_volt=0;
float solar_current=0;
int pwm = 0;                    //pwm duty cycle 0-1023

//float load_status=0;
float watts_in=0;

//int temp=0;           //θερμοκρασια

// enumerated variable που περιέχει τις καταστάσεις του state machine
//enum charger_mode {off, on, bulk, bat_float} charger_state;

// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
```

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** MAIN PROGRAM START
**********************************************
void setup()
{
  //pinMode(LOAD_ENABLE,OUTPUT);
  //pinMode(FAN_PIN,OUTPUT);
  pinMode(PWM_PIN,OUTPUT);
  pinMode(PWM_ENABLE,OUTPUT);
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  pinMode(LED_YELLOW,OUTPUT);

  TURN_OFF_FETS;  // disable driver 2104 and FETS

  Timer1.initialize(2000); // set a timer of length 2000uS (500Hz)
  Timer1.pwm(PWM_PIN, 0); // pwm on pin 9, 0% duty cycle, 768 = 75% duty cycle

  pwm = PWM_START;                  //αρχικη τιμη εκκίνησης pwm

  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines
}
//.....
//MAIN LOOP runs continiusly
//.....
void loop()
{
  delay(3000); //just to be able to see LCD and serial monitor - put 1 for normal operation
  read_data();         // read data for voltage and current
  power();             // υπολογισμος ισχυος
  TURN_ON_FETS;
  charger();           // no mppt just for test
```

```
  print_data();         //prints data
  //load_control();       // controls output
  led_output();         // led indication
  lcd_display();        // lcd display
}

//-------------------------------------------------------------------------------
// ΥΠΟΛΟΓΙΣΜΟς ΜΕΣΟΥ ΟΡΟΥ ADC ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΤΙΣ ΑΝΑΛΟΓΙΚΕΣ ΕΙΣΟΔΟΥΣ
//-------------------------------------------------------------------------------
int read_adc(int adc_parameter)
{
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                                 // διαβάζει adc τιμές AVG_NUM φορες
    sample = analogRead(adc_parameter);    // adc
    sum += sample;                      // αθροισμα για μεσο ορο
    delayMicroseconds(250);    // delay 250 microseconds
  }
  return(sum / AVG_NUM);              // επιστρέφει τον μέσο όρο των μετρήσεων
}
//-------------------------------------------------------------------------------
//READ THE DATA from inputs (adc)
//-------------------------------------------------------------------------------
void read_data(void)
{
  // 10kohms + 2kohms  works OK
  solar_volt = read_adc(SOL_VOLT) * (5.0 / 1023.0) * 6.08;//6.08=~12/2

  delayMicroseconds(500);
  //load_volt = read_adc(LOAD_ADC)*(4.38/1023.0)*5.70;
  //delayMicroseconds(100);

  //temp = read_adc(CURRENT_ADC_SOL)*(5.00/1023.00)*5.718;
```

```
// 20A ACS712 green
solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.498)/0.185;
delay(1);
//Serial.println(solar_current);
//Serial.println("a");
if (solar_current >0.18){
  solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.449)/0.185;
  //Serial.println("new");
}
if (solar_current >0.6){
  solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.419)/0.185;
  //Serial.println("new");
}
if (solar_current >1.1){
  solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.379)/0.185;
  //Serial.println("new");
}
if (solar_current >1.6){
  solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.319)/0.185;
  //Serial.println("new");
}

//solar_current = read_adc(SOL_CURRENT);
//solar_current = map(solar_current, 521, 560, 90, 1480) / 1000.00;

delayMicroseconds(500);

// 10kohms + 2kohms  works OK
bat_volt = read_adc(BAT_VOLT) * (5.0 / 1023.0) * 6.09;
//bat_volt = map(load_volt, 120, 324, 140, 4500) / 1000.00;

delayMicroseconds(500);

if (solar_current < 0) solar_current = 0;
```

```
}
//-----------------------------------------------------------------------------
// Χρήση του Timer1.pwm function για ρύθμιση του pwm duty cycle.
//-----------------------------------------------------------------------------
void set_pwm_duty(void) {

  if (pwm > PWM_MAX) {                  // pwm = PWM_MAX
    pwm = PWM_MAX;
  }
  else if (pwm < PWM_MIN) {             // αν pwm < PWM_MIN τοτε pwm = PWM_MIN
    pwm = PWM_MIN;
  }
  if (pwm < PWM_MAX) {
    Timer1.pwm(PWM_PIN, pwm);
  }
  else if (pwm == PWM_MAX) {            // an pwm = 1023 (maximum) μειωσε κατα 1
    Timer1.pwm(PWM_PIN,(pwm - 1));
  }
}
//********************* no mppt ******************************
void charger(void)
{
 // FAST charging most of the charging done here
 if((solar_volt > bat_volt )&& ( bat_volt <= HIGH_BAT_VOLTS )){
    TURN_ON_FETS;
    pwm+=PWM_INC;
    //pwm=50;
    Timer1.pwm(PWM_PIN, pwm);
    Serial.println("FAST");
    if (pwm > PWM_MAX){pwm = PWM_START;}
  }
  // SLOW charging
  else if((solar_volt > bat_volt)&&(bat_volt > HIGH_BAT_VOLTS)&& (bat_volt <
MAX_BAT_VOLTS + 1 )){
    TURN_ON_FETS;
```

```
    if (bat_volt >= MAX_BAT_VOLTS ){
      pwm-=4;
    }
    else {
      pwm+=1;
    }
    Timer1.pwm(PWM_PIN, pwm);
    Serial.println("SLOW");
    if (pwm > PWM_MAX){pwm = PWM_START;}
    if (pwm < 0){pwm = PWM_MIN;}
  }
  // shut down when battery is fully charged or when sunlight is not enough
  else if ((bat_volt > MAX_BAT_VOLTS+1) or (solar_volt < bat_volt)){
  pwm = PWM_MIN;
    Timer1.pwm(PWM_PIN, pwm);
    TURN_OFF_FETS;
    Serial.println("NOT charging");
  }
}
//-------------------------------------------------------------------------------------
//POWER CALCULATIONS
//-------------------------------------------------------------------------------------
void power(void)
{
  watts_in = solar_current * solar_volt; //Watts now from solar
}

//-------------------------------------------------------------------------------------
//PRINT DATA (SERIAL) PC MONITOR
//-------------------------------------------------------------------------------------
void print_data(void)
{
  delay(100);
  Serial.println("******************************");
  //prints data
```

```
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
  Serial.println("V");
  Serial.print("Battery Voltage : ");
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes: ");
  Serial.print(solar_current,2);
  Serial.println("A"); //εκτυπωση του γραμματος A
  Serial.print("Watt in : ");
  Serial.print(watts_in);
  Serial.println("W");
  Serial.print("pwm value : ");
  Serial.println(pwm);
}

//-------------------------------------------------------------------------------
//Led INDICATION
//-------------------------------------------------------------------------------

void led_output(void)
{
  if(bat_volt >= MAX_BAT_VOLTS )
  {
     leds_off_all();
     digitalWrite(LED_YELLOW, HIGH);
  }
  if(bat_volt > MIN_BAT_VOLTS && bat_volt < MAX_BAT_VOLTS)
  {
     leds_off_all();
     digitalWrite(LED_GREEN, HIGH);
  }
  if(bat_volt < MIN_BAT_VOLTS)
  {
     leds_off_all;
```

```
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_RED, HIGH);
  }
}
//-------------------------------------------------------------------------------------
// all leds off
//-------------------------------------------------------------------------------------
void leds_off_all(void)
{
 digitalWrite(LED_GREEN, LOW);
 digitalWrite(LED_RED, LOW);
 digitalWrite(LED_YELLOW, LOW);
}

//-------------------------------------------------------------------------------------
//LCD DISPLAY
//-------------------------------------------------------------------------------------
void lcd_display()
{
 delay(10);
 lcd.clear(); //clears LCD
 lcd.setCursor(0, 0);
 lcd.print("S ");
 lcd.setCursor(2, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 0);
 lcd.print(solar_current,1);
 lcd.print("A");
 lcd.setCursor(13, 0);
 lcd.print((int)watts_in);
 lcd.print("W ");

 lcd.setCursor(0, 1);
 lcd.print("B ");
```

```
 lcd.setCursor(2, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(12,1);
 lcd.print(pwm);
 //.....
 //κατασταση μπαταριας στο LCD
 //.....
 /*lcd.setCursor(8,3);
 if (bat_volt >= 14.00)
   {lcd.print( "100%");}
 else if (bat_volt >= 13.80 && bat_volt < 14.00)
   {lcd.print( " 90%");}
 else if (bat_volt >= 13.50 && bat_volt < 13.80)
   {lcd.print( " 80%");}
 else if (bat_volt >= 12.80 && bat_volt < 13.50)
   {lcd.print( " 70%");}
 else if (bat_volt >= 12.40 && bat_volt < 12.80)
   {lcd.print( " 60%");}
 else if (bat_volt >= 12.20 && bat_volt < 12.40)
   {lcd.print( " 50%");}
 else if (bat_volt >= 11.90 && bat_volt < 12.20)
   {lcd.print( " 40%");}
 else if (bat_volt >= 11.80 && bat_volt < 11.90)
   {lcd.print( " 30%");}
 else if (bat_volt >= 11.70 && bat_volt < 11.80)
   {lcd.print( " 20%");}
 else if (bat_volt >= 11.60 && bat_volt < 11.70)
   {lcd.print( " 10%");}
 else
  {lcd.print( "  0%");}*/
//.....
//Duty Cycle pwm στο LCD
//.....
 //lcd.setCursor(15,0);
```

```
 //lcd.print("PWM");
 //lcd.setCursor(15,1);
 //lcd.print(pwm);
 //-------------------------------------------------------------------------------
 //Load Status κατασταση φορτιου (LCD)
 //-------------------------------------------------------------------------------
 /*lcd.setCursor(15,2);
 lcd.print("Load");
 lcd.setCursor(15,3);
 if (load_status == 1)
 {
   lcd.print("On ");
 }
 else
 {
  lcd.print("Off");
 } */
}
```

Β.2)Αλγόριθμος λειτουργίας 2ης πλακέτας(ΜΕ MPPT)

```
/*
ΠΑΔΑ
MPPT
arduino NANO + LCD 20X2 I2C + BATTERY  πλακέτα
σειρά συνδεσμολογίας 1. μπαταρία 2. usb to pc (προαιρετικά) 3. solar panel

το jumper στην πλακέτα να είναι τοποθετημένο προς την πλευρά της μπαταρίας
για να δίνει τάση στο κύκλωμα η μπαταρία - στην αλλη πλευρά ολες οι τάσεις είναι
από το φωτοβολταϊκο

Ο αισθητήρας ρεύματος του φωτοβολταικου βασίζεται (20A acs712)
στο πιν A1 του arduino

volt input sensor pin A0
 current input sensor pin A1
```

```
 volt output sensor pin A2
 timer 2000uS (συχνοτητα λειτουργιας 500Hz)
 lcd = i2c 16X2 χρησιμοποιει τα A4 και A5
        A4 (SDA), A5 (SCL) addreess= 0x27
 */

#include "Wire.h"
#include <math.h>
// χρήση της βιβλιοθήκης LCD I2C απο https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
#include <LiquidCrystal_I2C.h>
#include "TimerOne.h"

#define SOL_VOLT A0     // Πιν για μέτρηση της τασης του φωτοβολταϊκου μέσω του διαιρέτη τάσης (adc)
#define SOL_CURRENT A1 // INA 282 current sensor φωτοβολταϊκού στο pin A1
#define BAT_VOLT A2     // μέτρηση τάσης φορτίου στο pin A2

#define AVG_NUM 100 // αριθμος επαναλήψεων για υπολογισμό μέσου όρου μετρώμενων τιμων απο την adc routine

#define MIN_BAT_VOLTS 11.80       //οριζόμενη ελαχιστη τιμη τάσης μπαταρίας
#define HIGH_BAT_VOLTS 13.80      //οριζόμενη επιθυμητή τιμη τάσης μπαταρίας
#define MAX_BAT_VOLTS 14.00       //οριζόμενη μέγιστη τιμη τάσης μπαταρίας

#define TURN_ON_FETS digitalWrite(PWM_ENABLE, HIGH)       // enable 2104 and FETS
#define TURN_OFF_FETS digitalWrite(PWM_ENABLE, LOW)       // disable 2104 and FETS

#define LED_RED 2
#define LED_GREEN 3
#define LED_YELLOW 4

#define PWM_ENABLE 8         // pin-8 is used to enable driver
#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET //the default frequency is about 490Hz
```

```
#define PWM_MAX 1020     // the value for maximum pwm duty cyle
#define PWM_MIN 5      // the value for minimum pwm duty cyle
#define PWM_START 20 // the value for initial pwm duty cyle

//-----------------------------------------------------------------------
//ΟΡΙΣΜΟΣ GLOBAL VARIABLES
//-----------------------------------------------------------------------
float solar_volt=0;
float  bat_volt=0;
float solar_current=0;
float  watts_in=0;
float watts_in_old=0;
int pwm = 0;                 //pwm duty cycle 0-1023
int PWM_INC = 11;            //τιμη αυξησης ή μείωσης pwm για υλοποιηση αλγοριθμου mppt
//float load_status=0;

// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** MAIN PROGRAM START
**********************************************
void setup()
{
  pinMode(PWM_PIN,OUTPUT);
  pinMode(PWM_ENABLE,OUTPUT);
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  pinMode(LED_YELLOW,OUTPUT);

  TURN_OFF_FETS;  // disable driver 2104 and FETS

  Timer1.initialize(2000); // set a timer of length 2000uS  (500Hz)
```

```
  Timer1.pwm(PWM_PIN, 0); // pwm on pin 9, 0% duty cycle, 768 = 75% duty cycle

  pwm = PWM_START;                  //αρχικη τιμη εκκίνησης pwm

  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines
}
//.....
//MAIN LOOP runs continiusly
//.....
void loop()
{
  delay(5000); //just to be able to see LCD and serial monitor - put 1 for normal operation
  read_data();          // read data for voltage and current
  power();              // υπολογισμος ισχυος
  TURN_ON_FETS;
  charger();            // no mppt just for test
  print_data();         //prints data
  led_output();         // led indication
  lcd_display();        // lcd display
}

//.....
// ΥΠΟΛΟΓΙΣΜΟς ΜΕΣΟΥ ΟΡΟΥ ADC ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΤΙΣ ΑΝΑΛΟΓΙΚΕΣ ΕΙΣΟΔΟΥΣ
//.....
int read_adc(int adc_parameter)
{
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                                 // διαβάζει adc τιμές AVG_NUM φορες
    sample = analogRead(adc_parameter);    // adc
    sum += sample;                      // αθροισμα για μεσο ορο
    delayMicroseconds(250);    // delay 250 microseconds
```

```
  }
  return(sum / AVG_NUM);              // επιστρέφει τον μέσο όρο των μετρήσεων
}
//-----------------------------------------------------------------------------
//READ THE DATA from inputs (adc)
//-----------------------------------------------------------------------------
void read_data(void)
{
  // 10kohms + 2kohms  works OK
  solar_volt = read_adc(SOL_VOLT) * (5.0 / 1023.0) * 6.08;//6.08=~12/2

  delayMicroseconds(500);
  //load_volt = read_adc(LOAD_ADC)*(4.38/1023.0)*5.70;
  //delayMicroseconds(100);

  //temp = read_adc(CURRENT_ADC_SOL)*(5.00/1023.00)*5.718;

  // 20A ACS712 green
  solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.489)/0.185;
  delay(1);
  if (solar_current >0.18){
    solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.449)/0.185;
  }
  if (solar_current >0.3){
    solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.429)/0.185;
  }
  if (solar_current >0.6){
    solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.419)/0.185;
  }
  if (solar_current >1.1){
    solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.379)/0.185;
  }
  if (solar_current >1.6){
    solar_current = ((read_adc(SOL_CURRENT)*(5.0/1023.0)) - 2.319)/0.185;
  }
```

```
  delayMicroseconds(500);

  // 10kohms + 2kohms  works OK
  bat_volt = read_adc(BAT_VOLT) * (5.0 / 1023.0) * 6.09;
  //bat_volt = map(load_volt, 120, 324, 140, 4500) / 1000.00;

  delayMicroseconds(500);

}
//-------------------------------------------------------------------------------
// Χρήση του Timer1.pwm function για ρύθμιση του pwm duty cycle.
//-------------------------------------------------------------------------------
void set_pwm_duty(void) {

  if (pwm > PWM_MAX) {                  // pwm = PWM_MAX
    pwm = PWM_MAX;
  }
  else if (pwm < PWM_MIN) {             // αν pwm < PWM_MIN τοτε pwm = PWM_MIN
    pwm = PWM_MIN;
  }
  if (pwm < PWM_MAX) {
    Timer1.pwm(PWM_PIN, pwm);
  }
  else if (pwm == PWM_MAX) {            // an pwm = 1023 (maximum) μειωσε κατα 1
    Timer1.pwm(PWM_PIN,(pwm - 1));
  }
}
//********************* no mppt ******************************
void charger(void)
{
 // FAST charging most of the charging done here
  // float charge when battery is charged and when sunlight is enough
    if(solar_volt > bat_volt){
      if (bat_volt < HIGH_BAT_VOLTS){  //bulk charge
```

```
      if (watts_in < watts_in_old){PWM_INC=-PWM_INC;}

      pwm+=PWM_INC;
      watts_in_old=watts_in;
      Serial.println(PWM_INC);
    }
    else  {
      watts_in_old = 0;
      if (PWM_INC<0){PWM_INC=-PWM_INC;} // change if still < 0
      if (bat_volt < MAX_BAT_VOLTS){  //float charge
        pwm+=1;
      }
      else{
        if (bat_volt > MAX_BAT_VOLTS + 0.6){
          pwm=0;
        }
        else{
          pwm-=7;
          if (pwm<0){pwm=0;}
        }
      }
    }
  }

  else{ // shut down when battery is fully charged or when sunlight is not enough
    pwm=0;
    Serial.println("NOT charging");
  }
  Serial.print("next pwm : ");
  Serial.println(pwm);
  Timer1.pwm(PWM_PIN, pwm);  //
  //if (pwm>1000){pwm=800;}
  if (pwm<0){pwm=0;}
  if (pwm>900 && watts_in<0.5){pwm=0;}
}
```

```
//-------------------------------------------------------------------------------
//POWER CALCULATIONS
//-------------------------------------------------------------------------------
void power(void)
{
  watts_in = solar_current * solar_volt; //Watts now from solar
}

//-------------------------------------------------------------------------------
//PRINT DATA (SERIAL) PC MONITOR
//-------------------------------------------------------------------------------
void print_data(void)
{
  delay(100);
  Serial.println("******************************");
  //prints data
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
  Serial.println("V");
  Serial.print("Battery Voltage : ");
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes: ");
  Serial.print(solar_current,2);
  Serial.println("A"); //εκτυπωση του γραμματος Α
  Serial.print("Watt in : ");
  Serial.print(watts_in);
  Serial.println("W");
  Serial.print("Watt in old: ");
  Serial.print(watts_in_old);
  Serial.println("W");
  Serial.print("pwm value : ");
  Serial.println(pwm);
}
```

```
//-------------------------------------------------------------------------------
//Led INDICATION
//-------------------------------------------------------------------------------

void led_output(void)
{
 if(bat_volt >= MAX_BAT_VOLTS )
 {
    leds_off_all();
    digitalWrite(LED_YELLOW, HIGH);
 }
 if(bat_volt > MIN_BAT_VOLTS && bat_volt < MAX_BAT_VOLTS)
 {
    leds_off_all();
    digitalWrite(LED_GREEN, HIGH);
 }
 if(bat_volt < MIN_BAT_VOLTS)
 {
    leds_off_all;
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_RED, HIGH);
 }
}
//-------------------------------------------------------------------------------
// all leds off
//-------------------------------------------------------------------------------
void leds_off_all(void)
{
 digitalWrite(LED_GREEN, LOW);
 digitalWrite(LED_RED, LOW);
 digitalWrite(LED_YELLOW, LOW);
}

//-------------------------------------------------------------------------------
//LCD DISPLAY
```

```
//-----------------------------------------------------------------------------------------------------
void lcd_display()
{
 delay(10);
 lcd.clear(); //clears LCD
 lcd.setCursor(0, 0);
 lcd.print("S ");
 lcd.setCursor(2, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 0);
 lcd.print(solar_current,1);
 lcd.print("A");
 lcd.setCursor(13, 0);
 lcd.print((int)watts_in);
 lcd.print("W ");

 lcd.setCursor(0, 1);
 lcd.print("B ");
 lcd.setCursor(2, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(9,1);
 lcd.print("pwm=");
 lcd.print(pwm);
 //-----------------------------------------------------------------------------------------
 //κατασταση μπαταριας στο LCD
 //-----------------------------------------------------------------------------------------
 /*lcd.setCursor(8,3);
 if (bat_volt >= 14.00)
   {lcd.print( "100%");}
 else if (bat_volt >= 13.80 && bat_volt < 14.00)
   {lcd.print( " 90%");}
 else if (bat_volt >= 13.50 && bat_volt < 13.80)
   {lcd.print( " 80%");}
```

```
 else if (bat_volt >= 12.80 && bat_volt < 13.50)
   {lcd.print( " 70%");}
 else if (bat_volt >= 12.40 && bat_volt < 12.80)
   {lcd.print( " 60%");}
 else if (bat_volt >= 12.20 && bat_volt < 12.40)
   {lcd.print( " 50%");}
 else if (bat_volt >= 11.90 && bat_volt < 12.20)
   {lcd.print( " 40%");}
 else if (bat_volt >= 11.80 && bat_volt < 11.90)
   {lcd.print( " 30%");}
 else if (bat_volt >= 11.70 && bat_volt < 11.80)
   {lcd.print( " 20%");}
 else if (bat_volt >= 11.60 && bat_volt < 11.70)
   {lcd.print( " 10%");}
 else
  {lcd.print( "  0%");}*/
//-------------------------------------------------------------------------------
//Duty Cycle pwm στο LCD
//-------------------------------------------------------------------------------
 //lcd.setCursor(15,0);
 //lcd.print("PWM");
 //lcd.setCursor(15,1);
 //lcd.print(pwm);
 //-------------------------------------------------------------------------------
 //Load Status κατασταση φορτιου (LCD)
 //-------------------------------------------------------------------------------
 /*lcd.setCursor(15,2);
 lcd.print("Load");
 lcd.setCursor(15,3);
 if (load_status == 1)
 {
   lcd.print("On ");
 }
 else
 {
```

```
  lcd.print("Off");
 } */
}
```

Παράρτημα Γ

1. Αλγόριθμος λειτουργίας 3ης πλακέτας χωρίς MPPT

```
/*
 ΠΑΔΑ ΚΟΣΜΑΔΑΚΗΣ ΧΑΡΑΛΑΜΠΟΣ ΤΕΛΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ PCB3 ΜΕ MPPT
 arduino MINI + LCD 20X2 I2C + BATTERY  πλακέτα
 σειρά συνδεσμολογίας 1. μπαταρία 2. usb to pc (προαιρετικά) 3. solar panel

 το jumper στην πλακέτα να ειναι τοποθετημένο προς την πλευρά της μπαταρίας
 για να δίνει τάση στο κύκλωμα η μπαταρία - στην αλλη πλευρά ολες οι τάσεις είναι
 από το φωτοβολταϊκο

 Ο αισθητήρας ρεύματος του φωτοβολταϊκού βασίζεται 20A acs712
 στο πιν A1 του arduino

 volt input sensor pin A0
 current input sensor pin A1
 volt output sensor pin A2
 THERMISTOR sensor pin A3

 lcd = i2c 16X2 χρησιμοποιει τα A4 και A5
        A4 (SDA), A5 (SCL) addreess= 0x27
 */

#include "Wire.h"
#include <math.h>
// χρήση της βιβλιοθήκης LCD I2C απο https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
```

```
#include <LiquidCrystal_I2C.h>

#include "TimerOne.h"


#define SOL_VOLT A0     // Πιν για μέτρηση της τασης του φωτοβολταϊκου μέσω του διαιρέτη τάσης (adc)

#define SOL_CURRENT A1 // INA 282 current sensor φωτοβολταϊκού στο pin A1

#define BAT_VOLT A2    // μέτρηση τάσης φορτίου στο pin A2

#define THERMISTOR A3 // ΘΕΡΜΟΚΡΑΣΙΑ στο pin A3


#define AVG_NUM 50 // αριθμος επαναλήψεων για υπολογισμό μέσου όρου μετρώμενων τιμων απο την adc routine


#define MIN_BAT_VOLTS 11.80       //οριζόμενη ελαχιστη τιμη τάσης μπαταρίας

#define HIGH_BAT_VOLTS 13.80      //οριζόμενη επιθυμητή τιμη τάσης μπαταρίας

#define MAX_BAT_VOLTS 14.00       //οριζόμενη μέγιστη τιμη τάσης μπαταρίας


#define TURN_ON_FETS digitalWrite(PWM_ENABLE, HIGH)       // enable 2104 and FETS

#define TURN_OFF_FETS digitalWrite(PWM_ENABLE, LOW)       // disable 2104 and FETS


#define LED_RED 2

#define LED_GREEN 3

#define LED_YELLOW 4

#define FAN_PIN 5

#define LOAD_ENABLE 6         // pin-6 is used to enable LOAD

#define PWM_ENABLE 8         // pin-8 is used to enable driver

#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET //the default frequency is about 490Hz

#define PWM_MAX 1020     // the value for maximum pwm duty cyle

#define PWM_MIN 600     // the value for minimum pwm duty cyle

#define PWM_START 700  // the value for initial pwm duty cyle

#define PWM_INC 15      //τιμη αυξησης ή μείωσης pwm για υλοποιηση αλγοριθμου mppt
```

```
//-------------------------------------------------------------------------
//ΟΡΙΣΜΟΣ GLOBAL VARIABLES
//-------------------------------------------------------------------------
float solar_volt=0;
float  bat_volt=0;
float solar_current=0;
int pwm = 0;                    //pwm duty cycle 0-1023

float watts_in=0;

int temp=0;          //θερμοκρασια

// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** MAIN PROGRAM START
**********************************************
void setup()
{
  pinMode(LOAD_ENABLE,OUTPUT);
  pinMode(FAN_PIN,OUTPUT);
  pinMode(PWM_PIN,OUTPUT);
  pinMode(PWM_ENABLE,OUTPUT);
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  pinMode(LED_YELLOW,OUTPUT);

  TURN_OFF_FETS;  // disable driver 2104 and FETS
```

```
  digitalWrite(LOAD_ENABLE, LOW); //STOP LOAD


  Timer1.initialize(40); // set a timer of length 40uS (25KHz)
  Timer1.pwm(PWM_PIN, 0); // pwm on pin 9, 0% duty cycle, 768 = 75% duty cycle


  pwm = PWM_START;                    //αρχικη τιμη εκκίνησης pwm


  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines
}
//-------------------------------------------------------------------------
//MAIN LOOP runs continiusly
//-------------------------------------------------------------------------
void loop()
{
  delay(3000); //just to be able to see LCD and serial monitor - put 1 for normal operation
  read_data();          // read data for voltage and current
  power();              // υπολογισμος ισχυος
  TURN_ON_FETS;
  charger();            // no mppt just for test
  print_data();         //prints data
  load_control();       // controls output
  led_output();         // led indication
  lcd_display();        // lcd display
}


//-------------------------------------------------------------------------------------------
// ΥΠΟΛΟΓΙΣΜΟς ΜΕΣΟΥ ΟΡΟΥ ADC ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΤΙΣ ΑΝΑΛΟΓΙΚΕΣ ΕΙΣΟΔΟΥΣ
```

```
//-----------------------------------------------------------------------------
int read_adc(int adc_parameter)
{
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                              // διαβάζει adc τιμές AVG_NUM φορες
    sample = analogRead(adc_parameter);    // adc
    sum += sample;                    // αθροισμα για μεσο ορο
    delayMicroseconds(250);    // delay 250 microseconds
  }
  return(sum / AVG_NUM);              // επιστρέφει τον μέσο όρο των μετρήσεων
}
// -----------------------------------------------------------------------------
//READ THE DATA from inputs (adc)
//  -----------------------------------------------------------------------------
void read_data(void)
{
  // 10kohms + 2kohms  works OK
  solar_volt = read_adc(SOL_VOLT) * (5.0 / 1023.0) * 6.00;//6.00=~12/2

  delayMicroseconds(500);

  solar_current = read_adc(SOL_CURRENT);
  solar_current = map(solar_current, 513, 556, 90, 2100) / 1000.00;

  delayMicroseconds(500);

  // 10kohms + 2kohms  works OK
```

```
  bat_volt = read_adc(BAT_VOLT) * (5.0 / 1023.0) * 6.01;


  delayMicroseconds(500);


  if (solar_current < 0) solar_current = 0;


  temp=Thermistor(analogRead(THERMISTOR));
  //Serial.println(temp);
  if (temp > 25){
    digitalWrite(FAN_PIN, HIGH);
  }
  else {
    digitalWrite(FAN_PIN, LOW);
  }
}
//-------------------------------------------------------------------------------
// Χρήση του Timer1.pwm function για ρύθμιση του pwm duty cycle.
//-------------------------------------------------------------------------------
void set_pwm_duty(void) {


  if (pwm > PWM_MAX) {                    // pwm = PWM_MAX
    pwm = PWM_MAX;
  }
  else if (pwm < PWM_MIN) {       // αν pwm < PWM_MIN τοτε pwm = PWM_MIN
    pwm = PWM_MIN;
  }
  if (pwm < PWM_MAX) {
    Timer1.pwm(PWM_PIN, pwm);
  }
```

```
  else if (pwm == PWM_MAX) {                // an pwm = 1023 (maximum) μειωσε κατα 1
    Timer1.pwm(PWM_PIN,(pwm - 1));
  }
}
//******************** no mppt ******************************
void charger(void)
{
 // FAST charging most of the charging done here
 if((solar_volt > bat_volt )&& ( bat_volt <= HIGH_BAT_VOLTS )){
    TURN_ON_FETS;
    pwm+=PWM_INC;
    //pwm=800;
    Timer1.pwm(PWM_PIN, pwm);
    Serial.println("FAST");
    if (pwm > PWM_MAX){pwm = PWM_START;}
  }
  // SLOW charging
  else if((solar_volt > bat_volt)&&(bat_volt > HIGH_BAT_VOLTS)&& (bat_volt < MAX_BAT_VOLTS + 0.1 )){
    TURN_ON_FETS;
    if (bat_volt >= MAX_BAT_VOLTS ){
      pwm-=9;
    }
    else {
      pwm+=1;
    }
    Timer1.pwm(PWM_PIN, pwm);
    Serial.println("SLOW");
    if (pwm > PWM_MAX){pwm = PWM_START;}
    if (pwm < 0){pwm = PWM_MIN;}
```

```
  }
  // shut down when battery is fully charged or when sunlight is not enough
  else if ((bat_volt > MAX_BAT_VOLTS) or (solar_volt < bat_volt)){
  pwm = PWM_MIN;
    Timer1.pwm(PWM_PIN, pwm);
    TURN_OFF_FETS;
    Serial.println("NOT charging");
  }
}
//-------------------------------------------------------------------------------
//POWER CALCULATIONS
//-------------------------------------------------------------------------------
void power(void)
{
  watts_in = solar_current * solar_volt; //Watts now from solar
}


//-------------------------------------------------------------------------------
//PRINT DATA (SERIAL) PC MONITOR
// -------------------------------------------------------------------------------
void print_data(void)
{
  delay(100);
  Serial.println("******************************");
  //prints data
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
  Serial.println("V");
  Serial.print("Battery  Voltage : ");
```

```
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes: ");
  Serial.print(solar_current,2);
  Serial.println("A"); //εκτυπωση του γραμματος A
  Serial.print("Watt in : ");
  Serial.print(watts_in);
  Serial.println("W");
  Serial.print("pwm value : ");
  Serial.println(pwm);
}

//-------------------------------------------------------------------------------
//LOAD CONTROL
//-------------------------------------------------------------------------------

void load_control()
{
  if(bat_volt > MIN_BAT_VOLTS+0.5)  // checks battery
  {
    digitalWrite(LOAD_ENABLE, HIGH); //ENABLE LOAD
  }
  if(bat_volt < MIN_BAT_VOLTS-0.5)
  {
    digitalWrite(LOAD_ENABLE, LOW); //DISABLE LOAD
  }
}
//-------------------------------------------------------------------------------
//Led INDICATION
```

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


void led_output(void)
{
  if(bat_volt >= MAX_BAT_VOLTS )
  {
     leds_off_all();
     digitalWrite(LED_YELLOW, HIGH);
  }
  if(bat_volt > MIN_BAT_VOLTS && bat_volt < MAX_BAT_VOLTS)
  {
     leds_off_all();
     digitalWrite(LED_GREEN, HIGH);
  }
  if(bat_volt < MIN_BAT_VOLTS)
  {
     leds_off_all;
     digitalWrite(LED_GREEN, LOW);
     digitalWrite(LED_RED, HIGH);
  }
}
//-----------------------------------------------------------------------------
// all leds off
// -----------------------------------------------------------------------------
void leds_off_all(void)
{
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_YELLOW, LOW);
```

```
}


//-------------------------------------------------------------------------------------
//LCD DISPLAY
//-------------------------------------------------------------------------------------
void lcd_display()
{
 delay(10);
 lcd.clear(); //clears LCD
 lcd.setCursor(0, 0);
 lcd.print("S ");
 lcd.setCursor(2, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 0);
 lcd.print(solar_current,1);
 lcd.print("A");
 lcd.setCursor(13, 0);
 lcd.print((int)watts_in);
 lcd.print("W ");
 lcd.setCursor(0, 1);
 lcd.print("B ");
 lcd.setCursor(2, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 1);
 lcd.print(temp,1);
 lcd.print("C");
 lcd.setCursor(12,1);
```

```
 lcd.print(pwm);



//-------------------------------------------------------------------------
double Thermistor(int RawADC) {
 double Temp;
 Temp = log(10000.0*((1024.0/RawADC-1)));
//        =log(10000.0/(1024.0/RawADC-1)) // for pull-up configuration
 Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
 Temp = Temp - 273.15;            // Convert Kelvin to Celcius
 return Temp;
}
```

2. Αλγόριθμος λειτουργίας 3ης πλακέτας με MPPT

```
/*
 ΠΑΔΑ ΚΟΣΜΑΔΑΚΗΣ ΧΑΡΑΛΑΜΠΟΣ ΤΕΛΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ PCB3 ΜΕ MPPT
 Arduino NANO + LCD 20X2 I2C + BATTERY πλακέτα
 σειρά συνδεσμολογίας 1. μπαταρία 2. usb to pc (προαιρετικά) 3. solar panel

 το jumper στην πλακέτα να είναι τοποθετημένο προς την πλευρά της μπαταρίας
 για να δίνει τάση στο κύκλωμα η μπαταρία - στην αλλη πλευρά ολες οι τάσεις είναι
 από το φωτοβολταϊκο

 Ο αισθητήρας ρεύματος του φωτοβολταϊκού βασίζεται 20A acs712
 στο πιν A1 του arduino

 volt input sensor pin A0
 current input sensor pin A1
 volt output sensor pin A2
 THERMISTOR sensor pin A3

 lcd = i2c 16X2 χρησιμοποιεί τα A4 και A5
       A4 (SDA), A5 (SCL) addreess= 0x27
```

```
 */

#include "Wire.h"
#include <math.h>
// χρήση της βιβλιοθήκης LCD I2C απο https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
#include <LiquidCrystal_I2C.h>
#include "TimerOne.h"

#define SOL_VOLT A0      // Πιν για μέτρηση της τασης του φωτοβολταϊκου μέσω του διαιρέτη τάσης (adc)
#define SOL_CURRENT A1 // INA 282 current sensor φωτοβολταϊκού στο pin A1
#define BAT_VOLT A2     // μέτρηση τάσης φορτίου στο pin A2
#define THERMISTOR A3 // ΘΕΡΜΟΚΡΑΣΙΑ στο pin A3

#define AVG_NUM 50 // αριθμος επαναλήψεων για υπολογισμό μέσου όρου μετρώμενων τιμων απο την adc routine

#define MIN_BAT_VOLTS 11.80       //οριζόμενη ελαχιστη τιμη τάσης μπαταρίας
#define HIGH_BAT_VOLTS 13.80      //οριζόμενη επιθυμητή τιμη τάσης μπαταρίας
#define MAX_BAT_VOLTS 14.00       //οριζόμενη μέγιστη τιμη τάσης μπαταρίας

#define TURN_ON_FETS digitalWrite(PWM_ENABLE, HIGH)       // enable 2104 and FETS
#define TURN_OFF_FETS digitalWrite(PWM_ENABLE, LOW)       // disable 2104 and FETS

#define LED_RED 2
#define LED_GREEN 3
#define LED_YELLOW 4
#define FAN_PIN 5
#define LOAD_ENABLE 6         // pin-6 is used to enable LOAD
#define PWM_ENABLE 8         // pin-8 is used to enable driver
#define PWM_PIN 9         // pin-9 is used to control the charging MOSFET //the default frequency is about 490Hz
#define PWM_MAX 1020     // the value for maximum pwm duty cyle
#define PWM_MIN 600     // the value for minimum pwm duty cyle
```

```
#define PWM_START 700 // the value for initial pwm duty cyle
#define PWM_INC 15      //τιμη αυξησης ή μείωσης pwm για υλοποιηση αλγοριθμου mppt

//------------------------------------------------------------------------
//ΟΡΙΣΜΟΣ GLOBAL VARIABLES
//------------------------------------------------------------------------
float solar_volt=0;
float  bat_volt=0;
float solar_current=0;
int pwm = 0;                    //pwm duty cycle 0-1023

float watts_in=0;

int temp=0;          //θερμοκρασια
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address

//********************** MAIN PROGRAM START
***********************************************
void setup()
{
  pinMode(LOAD_ENABLE,OUTPUT);
  pinMode(FAN_PIN,OUTPUT);
  pinMode(PWM_PIN,OUTPUT);
  pinMode(PWM_ENABLE,OUTPUT);
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  pinMode(LED_YELLOW,OUTPUT);

  TURN_OFF_FETS; // disable driver 2104 and FETS
  digitalWrite(LOAD_ENABLE, LOW); //STOP LOAD

  Timer1.initialize(40); // set a timer of length 20uS (50KHz)
  Timer1.pwm(PWM_PIN, 0); // pwm on pin 9, 0% duty cycle, 768 = 75% duty cycle
```

```
  pwm = PWM_START;                    //αρχικη τιμη εκκίνησης pwm

  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16,2);  // initialize the lcd for 16 chars 2 lines
}
//-----------------------------------------------------------------------
//MAIN LOOP runs continiusly
//-----------------------------------------------------------------------
void loop()
{
  delay(3000); //just to be able to see LCD and serial monitor - put 1 for normal operation
  read_data();          // read data for voltage and current
  power();              // υπολογισμος ισχυος
  TURN_ON_FETS;
  charger();            // no mppt just for test
  print_data();         //prints data
  load_control();       // controls output
  led_output();         // led indication
  lcd_display();        // lcd display
}

//-----------------------------------------------------------------------------------
// ΥΠΟΛΟΓΙΣΜΟς ΜΕΣΟΥ ΟΡΟΥ ADC ΜΕΤΡΗΣΕΩΝ ΑΠΟ ΤΙΣ ΑΝΑΛΟΓΙΚΕΣ ΕΙΣΟΔΟΥΣ
//-----------------------------------------------------------------------------------
int read_adc(int adc_parameter)
{
  long sum = 0;
  long sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                                   // διαβάζει adc τιμές AVG_NUM φορες
    sample = analogRead(adc_parameter);    // adc
    sum += sample;                        // αθροισμα για μεσο ορο
    delayMicroseconds(250);    // delay 250 microseconds
```

```
  }
  return(sum / AVG_NUM);              // επιστρέφει τον μέσο όρο των μετρήσεων
}
//-----------------------------------------------------------------------------
//READ THE DATA from inputs (adc)
//-----------------------------------------------------------------------------
void read_data(void)
{
  // 10kohms + 2kohms  works OK
  solar_volt = read_adc(SOL_VOLT) * (5.0 / 1023.0) * 6.00;//6.00=~12/2

  delayMicroseconds(500);

  solar_current = read_adc(SOL_CURRENT);
  solar_current = map(solar_current, 513, 556, 90, 2100) / 1000.00;

  delayMicroseconds(500);

  // 10kohms + 2kohms  works OK
  bat_volt = read_adc(BAT_VOLT) * (5.0 / 1023.0) * 6.01;

  delayMicroseconds(500);

  if (solar_current < 0) solar_current = 0;

  temp=Thermistor(analogRead(THERMISTOR));
  //Serial.println(temp);
  if (temp > 25){
    digitalWrite(FAN_PIN, HIGH);
  }
  else {
    digitalWrite(FAN_PIN, LOW);
  }
}
//-----------------------------------------------------------------------------
```

```
// Χρήση του Timer1.pwm function για ρύθμιση του pwm duty cycle.
//---------------------------------------------------------------------------------------------
void set_pwm_duty(void) {

 if (pwm > PWM_MAX) {                  // pwm = PWM_MAX
   pwm = PWM_MAX;
 }
 else if (pwm < PWM_MIN) {             // αν pwm < PWM_MIN τοτε pwm = PWM_MIN
   pwm = PWM_MIN;
 }
 if (pwm < PWM_MAX) {
   Timer1.pwm(PWM_PIN, pwm);
 }
 else if (pwm == PWM_MAX) {            // an pwm = 1023 (maximum) μειωσε κατα 1
   Timer1.pwm(PWM_PIN,(pwm - 1));
 }
}
//********************* no mppt ******************************
void charger(void)
{
 // FAST charging most of the charging done here
 if((solar_volt > bat_volt )&& ( bat_volt <= HIGH_BAT_VOLTS )){
   TURN_ON_FETS;
   pwm+=PWM_INC;
   //pwm=800;
   Timer1.pwm(PWM_PIN, pwm);
   Serial.println("FAST");
   if (pwm > PWM_MAX){pwm = PWM_START;}
 }
 // SLOW charging
 else if((solar_volt > bat_volt)&&(bat_volt > HIGH_BAT_VOLTS)&& (bat_volt <
MAX_BAT_VOLTS + 0.1 )){
   TURN_ON_FETS;
   if (bat_volt >= MAX_BAT_VOLTS ){
     pwm-=9;
```

```
    }
    else {
      pwm+=1;
    }
    Timer1.pwm(PWM_PIN, pwm);
    Serial.println("SLOW");
    if (pwm > PWM_MAX){pwm = PWM_START;}
    if (pwm < 0){pwm = PWM_MIN;}
  }
  // shut down when battery is fully charged or when sunlight is not enough
  else if ((bat_volt > MAX_BAT_VOLTS) or (solar_volt < bat_volt)){
  pwm = PWM_MIN;
    Timer1.pwm(PWM_PIN, pwm);
    TURN_OFF_FETS;
    Serial.println("NOT charging");
  }
}
//-------------------------------------------------------------------------------
//POWER CALCULATIONS
//-------------------------------------------------------------------------------
void power(void)
{
  watts_in = solar_current * solar_volt; //Watts now from solar
}

//-------------------------------------------------------------------------------
//PRINT DATA (SERIAL) PC MONITOR
//-------------------------------------------------------------------------------
void print_data(void)
{
  delay(100);
  Serial.println("******************************");
  //prints data
  Serial.print("Solar Panel Voltage : ");
  Serial.print(solar_volt,2);
```

```
  Serial.println("V");
  Serial.print("Battery Voltage : ");
  Serial.print(bat_volt,2);
  Serial.println("V");
  Serial.print("Solar Panel Amperes: ");
  Serial.print(solar_current,2);
  Serial.println("A"); //εκτυπωση του γραμματος Α
  Serial.print("Watt in : ");
  Serial.print(watts_in);
  Serial.println("W");
  Serial.print("pwm value : ");
  Serial.println(pwm);
}

//-------------------------------------------------------------------------------
//LOAD CONTROL
//-------------------------------------------------------------------------------

void load_control()
{
  if(bat_volt > MIN_BAT_VOLTS+0.5)  // checks battery
  {
    digitalWrite(LOAD_ENABLE, HIGH); //ENABLE LOAD
  }
  if(bat_volt < MIN_BAT_VOLTS-0.5)
  {
    digitalWrite(LOAD_ENABLE, LOW); //DISABLE LOAD
  }
}
//-------------------------------------------------------------------------------
//Led INDICATION
//-------------------------------------------------------------------------------

void led_output(void)
{
```

```
 if(bat_volt >= MAX_BAT_VOLTS )
 {
    leds_off_all();
    digitalWrite(LED_YELLOW, HIGH);
 }
 if(bat_volt > MIN_BAT_VOLTS && bat_volt < MAX_BAT_VOLTS)
 {
    leds_off_all();
    digitalWrite(LED_GREEN, HIGH);
 }
 if(bat_volt < MIN_BAT_VOLTS)
 {
    leds_off_all;
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_RED, HIGH);
 }
}
//-------------------------------------------------------------------------------------
// all leds off
//-------------------------------------------------------------------------------------
void leds_off_all(void)
{
 digitalWrite(LED_GREEN, LOW);
 digitalWrite(LED_RED, LOW);
 digitalWrite(LED_YELLOW, LOW);
}

//-------------------------------------------------------------------------------------
//LCD DISPLAY
//-------------------------------------------------------------------------------------
void lcd_display()
{
 delay(10);
 lcd.clear(); //clears LCD
 lcd.setCursor(0, 0);
```

```
 lcd.print("S ");
 lcd.setCursor(2, 0);
 lcd.print(solar_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 0);
 lcd.print(solar_current,1);
 lcd.print("A");
 lcd.setCursor(13, 0);
 lcd.print((int)watts_in);
 lcd.print("W ");
 lcd.setCursor(0, 1);
 lcd.print("B ");
 lcd.setCursor(2, 1);
 lcd.print(bat_volt,1);
 lcd.print("V");
 lcd.setCursor(8, 1);
 lcd.print(temp,1);
 lcd.print("C");
 lcd.setCursor(12,1);
 lcd.print(pwm);


//.....
double Thermistor(int RawADC) {
 double Temp;
 Temp = log(10000.0*((1024.0/RawADC-1)));
//         =log(10000.0/(1024.0/RawADC-1)) // for pull-up configuration
 Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
 Temp = Temp - 273.15;            // Convert Kelvin to Celcius
 return Temp;
}
```