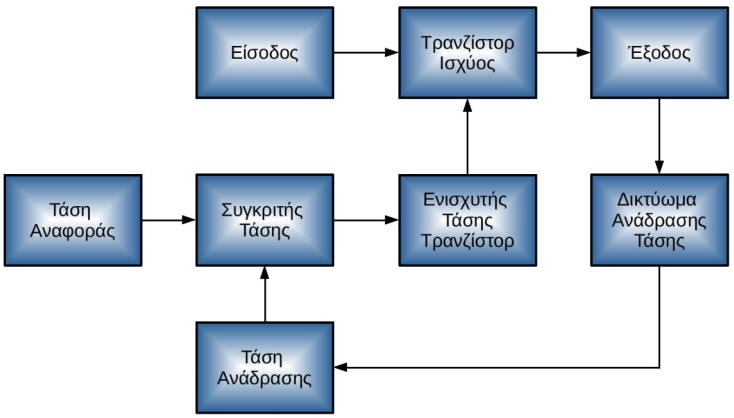
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μελέτη και κατασκευή
τροφοδοτικού συνεχούς τάσης
ελεγχόμενο από Η/Υ



Του φοιτητή
Κουφογεώργου Βασιλείου Ρωμανού
Αρ. Μητρώου: 514068

Επιβλέπων
Χατζόπουλος Αργύριος
Επίκουρος Καθηγητής

Σεπτέμβριος 2021

Τίτλος Δ.Ε. : Μελέτη και κατασκευή τροφοδοτικού συνεχούς τάσης ελεγχόμενο από Η/Υ
Κωδικός Δ.Ε. : 21176
Ονοματεπώνυμο φοιτητή : Κουφογεώργος Βασίλειος Ρωμανός
Ονοματεπώνυμο εισηγητή : Χατζόπουλος Αργύριος
Ημερομηνία ανάληψης Δ.Ε. 11-03-2021
Ημερομηνία περάτωσης Δ.Ε. 15-09-2021

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

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

Πρόλογος

Η παρούσα Διπλωματική Εργασία εκπονήθηκε κατά την περίοδο των ακαδημαϊκών ετών 2020-21 στο πλαίσιο του Προπτυχιακού Προγράμματος Σπουδών του τμήματος “Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων” του Διεθνούς Πανεπιστημίου της Ελλάδος υπό την επίβλεψη του κ. Αργυρίου Χατζόπουλου, Επίκουρου καθηγητή του τμήματος. Στην εργασία προσπάθησα να υλοποιήσω ένα διακοπτικό τροφοδοτικό πάγκου. Η επιλογή μου για το θέμα αυτό έγινε από το ενδιαφέρον μου στις εφαρμογές των ενσωματωμένων συστημάτων σε κυκλώματα ηλεκτρονικών ισχύος, αλλά και από τον βαθμό δυσκολίας που έχει αυτό το θέμα. Κατά την διάρκεια της εκπόνησης της διπλωματικής εργασίας εφάρμοσα τις θεωρητικές γνώσεις και τις εργαστηριακές δεξιότητες που απέκτησα κατά την διάρκεια των σπουδών μου. Επιπρόσθετα, απέκτησα αρκετή εργαστηριακή εμπειρία και γνώση στα διακοπτικά τροφοδοτικά και τους μικροελεγκτές σε πραγματικές συνθήκες λειτουργίας, καθώς απαιτήθηκε πολλές φορές να αντιμετωπίσω δυσκολίες που δεν αναφέρονταν στα βιβλία και στα φυλλάδια δεδομένων.

Περίληψη

Στην παρούσα διπλωματική εργασία μελετήθηκαν δυο είδη τροφοδοτικών: το διακοπτικό και το γραμμικό τροφοδοτικό πάγκου. Σχεδιάσαμε ένα γραμμικό τροφοδοτικό πάγκου, από το οποίο πήραμε αποτελέσματα. Ήταν σχετικά εύκολο στη σχεδίαση και τον έλεγχο του αλλά είχαμε το πρόβλημα της μείωσης του σφάλματος εξόδου. Καταφέραμε να το μειώσουμε με τη χρήση ενός εξωτερικού DAC, καθώς επίσης επιτεύχθηκε και ο έλεγχος - προγραμματισμός του τροφοδοτικού από κάποιο απομακρυσμένο σημείο.

Αρχικά, είχαμε σχεδιάσει ένα διακοπτικό τροφοδοτικό τύπου buck converter ελεγχόμενο από ένα ψηφιακό ελεγκτή PID που προγραμματίστηκε στον μικροελεγκτή PIC18F4550, όμως δεν καταφέραμε να πάρουμε αποτελέσματα. Το διακοπτικό ήταν δύσκολο και στη σχεδίαση και στην υλοποίηση του αλλά μάθαμε πολλά κατά την διάρκεια της εκτέλεσης του.

«Design and construction

of DC power supply

controlled by a computer »

«Vasilios Romanos Koufogeorgos»

Abstract

In this theses we studies two types of power supplies : the swithing and the linear bench power supply.We designed a linear bench power supply, from which we obtained results. It was relatively easy to design and test but we had the problem of reducing the output error. We managed to reduce it by using an external DAC, and also achieved control - programming the power supply from a remote point.Initially we designed a switching power supply of type buck converter ,controlled by a digital PID controller that was programmed in the PIC18F4550 microcontroller,but we were unable to get any results.The switching power supply was difficult to design and to construct ,but we learned a lot in the time it took us.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τη μητέρα μου και την αδερφή που με στήριξαν κατά την διάρκεια της εκπόνησης της διπλωματικής, καθώς επίσης τον καθηγητή κύριο Αργύριο Χατζόπουλο για την βοήθεια που μου πρόσφερε.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Η/Υ	Ηλεκτρονικός Υπολογιστής
SMPS	Switch Mode Power Supply
MOSFET	Metal Oxide Semiconductor field Effect Transistor
PWM	Pulse Width Modulation
CCM	Continuous Conduction Mode
DCM	Discontinuous Conduction Mode
PID	Proportional Integral Derivative

Εισαγωγή

Η διπλωματική αφορά τη μελέτη και την υλοποίηση τροφοδοτικού πάγκου , τον έλεγχο του τροφοδοτικού από έναν ηλεκτρονικό υπολογιστή. Ο προγραμματισμός ενός τροφοδοτικού μέσω ηλεκτρονικού υπολογιστή μπορεί να χρησιμοποιηθεί σε διάφορες περιπτώσεις όπως σε εργαστήρια ηλεκτρονικών πλακετών και τον έλεγχο πειραματικών διατάξεων σε ράστερ. Στην εργασία θα επεξηγηθούν τα κυκλώματα που αποτελείται το τροφοδοτικό καθώς και οι περιφερειακές μονάδες του μικροελεγκτή που χρησιμοποιήθηκαν για τον έλεγχο του.

- Στο κεφάλαιο 1 γίνεται αναφορά στα είδη τροφοδοτικών πάγκου και στη συνέχεια η επεξήγηση των περιφερειακών που χρησιμοποιήθηκαν στα τροφοδοτικά.

- Στο κεφάλαιο 2 γίνεται εμβάθυνση στη λειτουργία του διακοπτικού τροφοδοτικού, ηλεκτρονική ανάλυση καθώς και η εύρεση των τιμών των υλικών και οι προσομοιώσεις του κυκλώματος.

- Στο κεφάλαιο 3 περιγράφεται ο προγραμματισμός και ο κώδικας των περιφερειακών που χρησιμοποιήθηκαν στον μικροελεγκτή σαν διακοπτικό τροφοδοτικό.

- Στο κεφάλαιο 4 επεξηγούνται τα κυκλώματα που αποτελούν το διακοπτικό τροφοδοτικό και υλοποιούνται οι πειραματικές διατάξεις του τροφοδοτικού.

- Στο κεφάλαιο 5 γίνεται επεξήγηση της λειτουργίας του γραμμικού τροφοδοτικού, ηλεκτρονική ανάλυση και οι προσομοιώσεις του κυκλώματος.

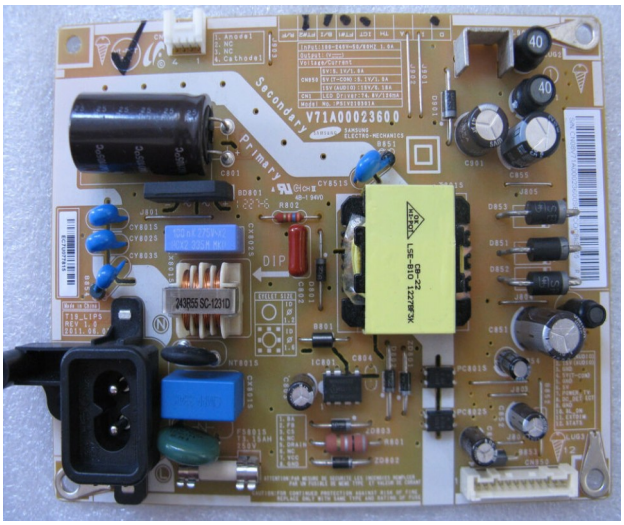
- Στο κεφάλαιο 6 περιγράφεται ο προγραμματισμός και ο κώδικας των περιφερειακών που χρησιμοποιήθηκαν στον μικροελεγκτή σαν γραμμικό τροφοδοτικό.

- Στο κεφάλαιο 7 επεξηγούνται τα κυκλώματα που αποτελούν το γραμμικό τροφοδοτικό και υλοποιούνται οι πειραματικές διατάξεις του τροφοδοτικού.

- Τέλος, στο κεφάλαιο 8 παρουσιάζονται τα αποτελέσματα και οι βελτιώσεις που μπορούν να γίνουν.

Κεφάλαιο 1 Εισαγωγή στα τροφοδοτικά

Τα τροφοδοτικά είναι ηλεκτρικές συσκευές που μετατρέπουν την ένταση του ρεύματος μιας πηγής όπως μπαταρίες, γεννήτριες ή το δίκτυο σε τάση και ένταση που θέλουμε. Κάποια από αυτά βρίσκονται μέσα στις συσκευές που χρειάζονται τροφοδοσία, σαν αυτά των Η/Υ, ενώ άλλα βρίσκονται έξω όπως οι φορτιστές κινητών συσκευών ή τροφοδοτικά πάγκου.



Εικόνα 1.1: Τροφοδοτικό Τηλεόρασης


Εικόνα 1.2 Φορτιστής

Ένας ακόμη διαχωρισμός είναι ο τύπος ρεύματος εξόδου του τροφοδοτικού που μπορεί να είναι είτε εναλλασσόμενο ρεύμα AC (Alternating Current) είτε συνεχές ρεύμα DC (Direct Current). Στη διπλωματική αυτή αναφέρονται μόνο τα τροφοδοτικά με DC έξοδο, τα οποία διαχωρίζονται και αυτά με τη σειρά τους σε γραμμικά, διακοπτικά και χωρητικά τροφοδοτικά. Παρακάτω θα αναφερθούν μόνο το γραμμικό και το διακοπτικό.

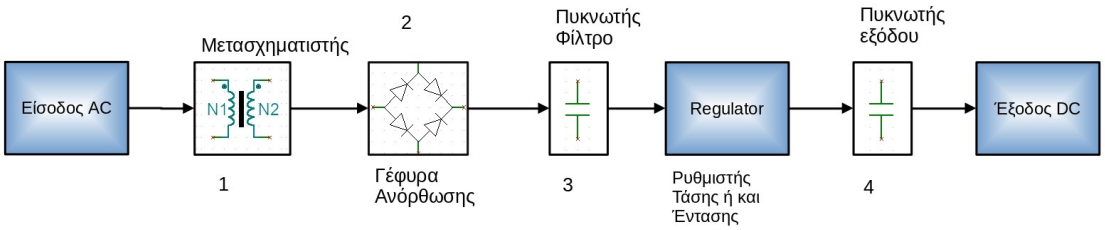
1.1 Τροφοδοτικά

1.1.1 Γραμμικό Τροφοδοτικό

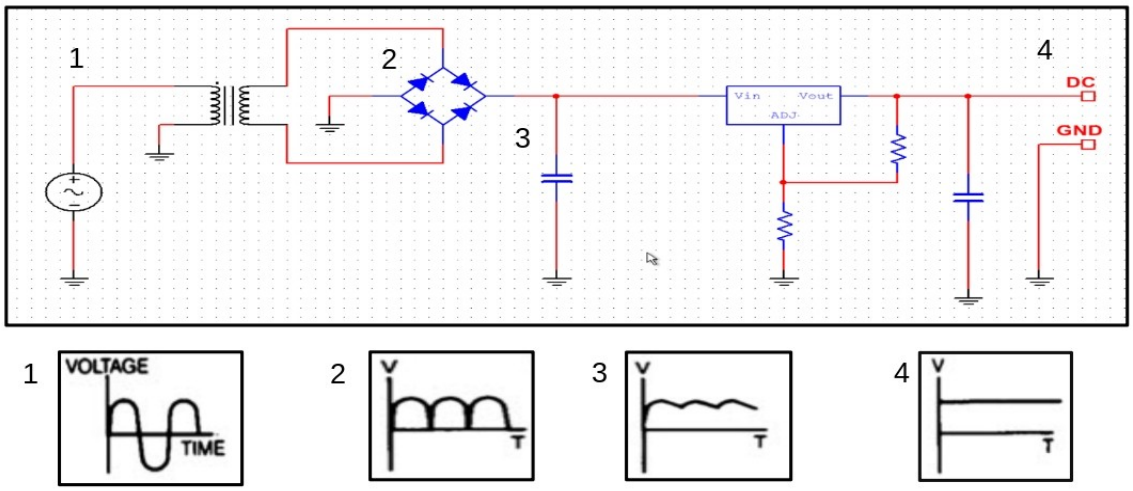
Το γραμμικό τροφοδοτικό είναι το πιο διαδεδομένο στα τροφοδοτικά πάγκου σε σχέση με τα παλμοτροφοδοτικά λόγω της αξιοπιστίας του και της τεχνολογίας που χρησιμοποιείται σ΄αυτό και είναι πιο εύκολη η κατασκευή του. Επίσης, η τεχνολογία του τροφοδοτικού είναι καθιερωμένη και διαθέσιμη περισσότερα χρόνια.

Ο τρόπος με τον οποίο λειτουργεί το τροφοδοτικό μοιάζει με αυτό μιας βαλβίδας νερού (βρύσης), την οποία ρυθμίζουμε ανάλογα με την πίεση (ένταση) ή ποσότητα (τάση) νερού που θέλουμε. Αυτή η λειτουργία γίνεται από τον ρυθμιστή (regulator), που μπορεί να ρυθμίζει την τάση ή και την ένταση του ρεύματος. Ο ρυθμιστής μπορεί να είναι ένα κύκλωμα από διακριτά υλικά ή και ένα ολοκληρωμένο κύκλωμα. Εξαιτίας της αυξομείωσης της αγωγιμότητας του ρυθμιστή καταναλώνεται πολύ ισχύς επάνω του σαν θερμότητα που καθιστά το γραμμικό τροφοδοτικό μη αποδοτικό στη ρύθμιση μεγάλων ρευμάτων. Ο ρυθμιστής λειτουργεί με συγκεκριμένη τάση του ρεύματος που πρέπει να είναι DC.

Για να επιτευχθεί, το ρεύμα από το δίκτυο περνάει από έναν μετασχηματιστή ο οποίος μετατρέπει το ρεύμα σε χαμηλότερη τάση (1), στη συνέχεια το εναλλασσόμενο ρεύμα που παίρνουμε ανορθώνεται (2), περνάει από έναν πυκνωτή φίλτρο και τέλος πηγαίνει στον ρυθμιστή και έχουμε έξοδο DC. Μια πιο αναλυτική περιγραφή για την λειτουργία του γραμμικού τροφοδοτικού θα εξηγηθεί σε επόμενο κεφάλαιο.



Εικόνα 1.3: Block Διάγραμμα Γραμμικού Τροφοδοτικού



Εικόνα 1.4: Κύκλωμα Γραμμικού Τροφοδοτικού με Ρυθμιστή Ολοκληρωμένο Κύκλωμα (Chip)

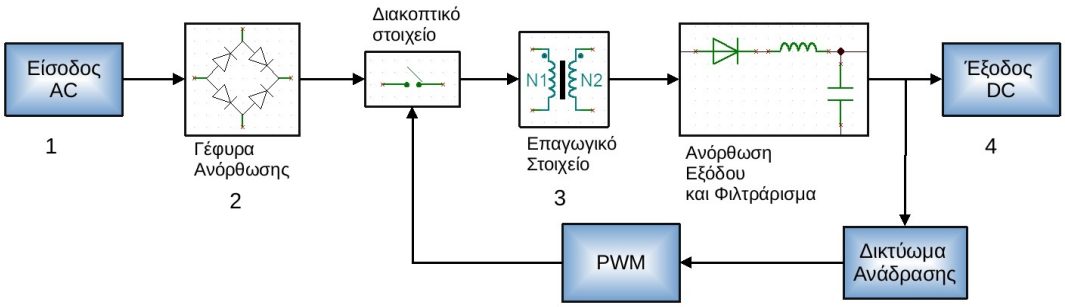
1.1.2 Διακοπτικό Τροφοδοτικό

Το διακοπτικό τροφοδοτικό ή παλμοτροφοδοτικό ή SMPS αν και είναι πιο περίπλοκο σε σχέση με το γραμμικό, χρησιμοποιείται ευρέως σε εφαρμογές που θέλουμε μεγάλη αποδοτικότητα και μικρό μέγεθος όπως φορτιστές κινητών και τροφοδοτικά Η/Υ. Επίσης, χρησιμοποιείται και ως τροφοδοτικό πάγκου όταν θέλουμε μεγάλα ρεύματα και τέλος, υπάρχουν εφαρμογές που το διακοπτικό τροφοδοτικό είναι πιο φθηνό από το γραμμικό για την υλοποίηση του.

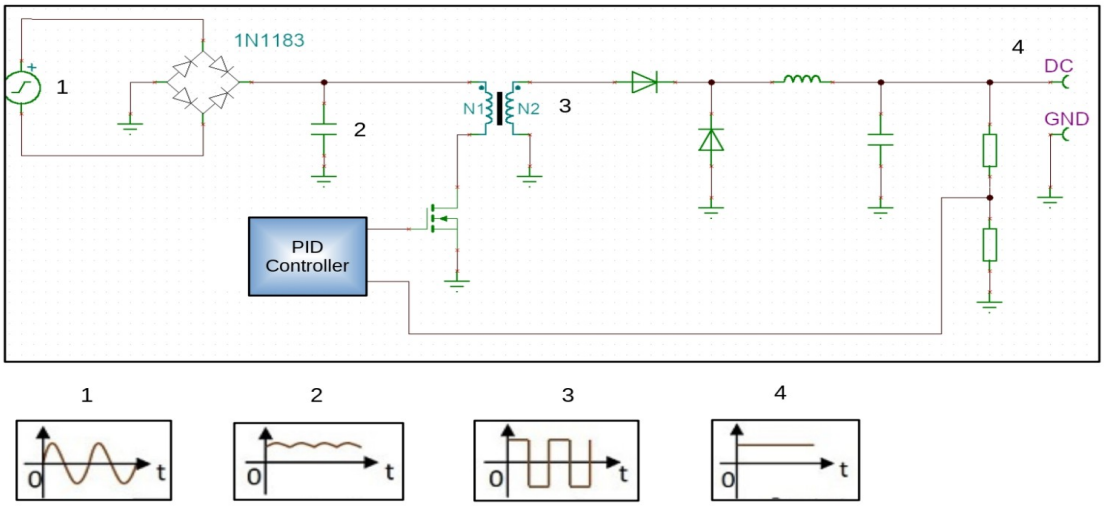
Ο τρόπος λειτουργίας των παλμοτροφοδοτικών, ανεξαρτήτως του τύπου τους και της συνδεσμολογίας τους, είναι παρόμοιος. Υπάρχει σε όλα ένα είδος transistor ή ένα ολοκληρωμένο κύκλωμα που λειτουργεί σαν διακόπτης. Βρίσκεται σε μια από τις δυο καταστάσεις, ανοιχτό ON ή κλειστό OFF και ανοιγοκλείνει πολλές φορές το δευτερόλεπτο.

Αποτελούνται πάντα από ένα ή περισσότερα επαγωγικά στοιχεία, αυτεπαγωγές (πηνίο) ή και μετασχηματιστή που αποθηκεύουν ενέργεια, ένα ή και περισσότερους πυκνωτές, φίλτρα και διόδους ελεύθερης προσέλευσης (flywheel diode) ή και διόδους ανόρθωσης. Να σημειωθεί ότι τα παλμοτροφοδοτικά χωρίζονται σε δυο κατηγορίες, αυτά που έχουν μετασχηματιστή άρα και γαλβανική απομόνωση και αυτά που δεν έχουν μετασχηματιστή.

Το διακοπτικό στοιχείο δέχεται μια DC τάση είτε ανορθωμένη AC (1 και 2) είτε απλή DC. Εξαιτίας των διακοπών ON-OFF καταναλώνεται λίγη ισχύς πάνω στον διακόπτη γιατί δεν άγει συνέχεια κατά την διάρκεια της λειτουργίας. Κατόπιν από τον διακόπτη βγαίνει ένας τετραγωνικός παλμός (3) με το εύρος των παλμών να είναι ανάλογο των διακοπών ON-OFF(PWM). Τέλος, ο παλμός περνάει μέσα από το πηνίο και τον πυκνωτή που λειτουργούν σαν χαμηλοπερατό φίλτρο, έτσι στην έξοδο παίρνουμε DC(4) τάση. Μια πιο αναλυτική περιγραφή για την λειτουργία του παλμοτροφοδοτικού θα εξηγηθεί σε επόμενο κεφάλαιο.



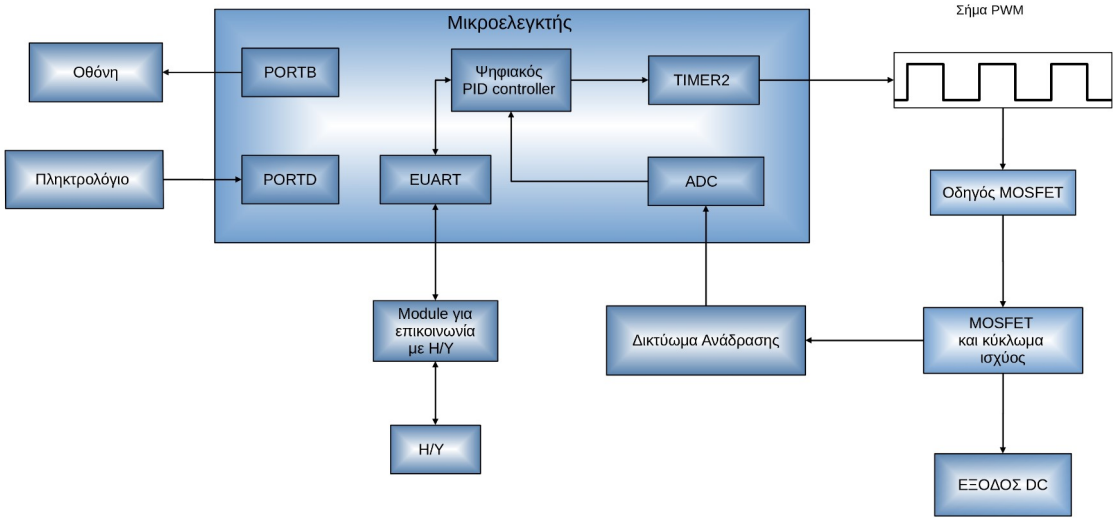
Εικόνα 1.5: Block Διάγραμμα Παλμοτροφοδοτικού



Εικόνα 1.6: Κύκλωμα Παλμοτροφοδοτικού

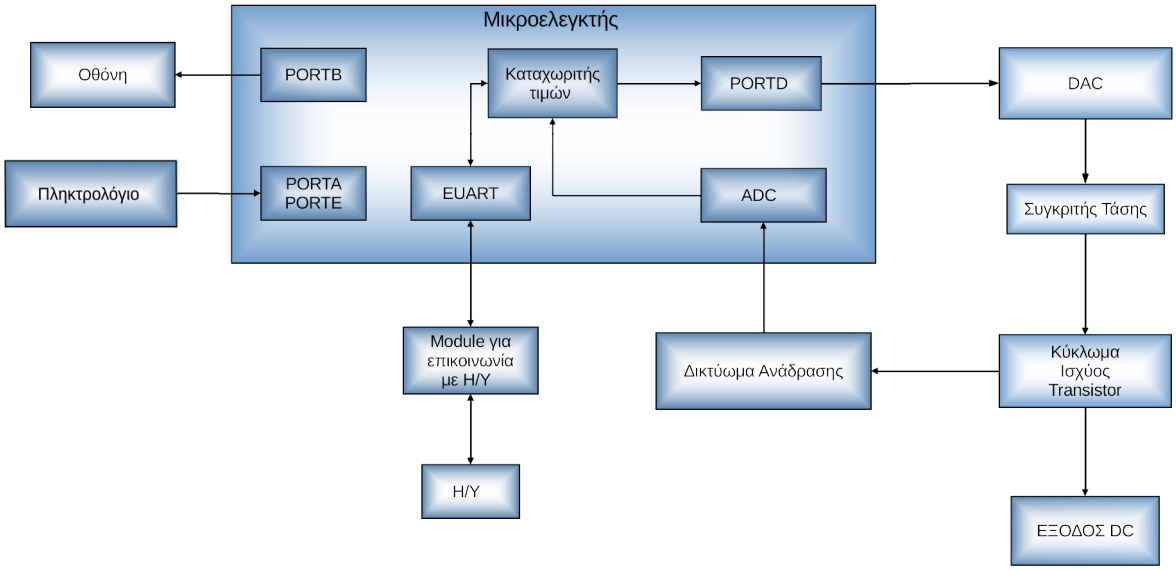
1.2 Τροφοδοτικά Διπλωματικής

Τα προγραμματιζόμενα τροφοδοτικά είναι μια πιο γενική υποκατηγορία τροφοδοτικών που είναι είτε γραμμικά είτε διακοπτικά, με τη διαφορά ότι ο χρήστης μπορεί να επιλέγει την τάση, την ένταση ή και τη συχνότητα από ένα απομακρυσμένο σημείο. Ο έλεγχος γίνεται από έναν Η/Υ συνδεμένο με το τροφοδοτικό. Ο προγραμματισμός του τροφοδοτικού γίνεται από κάποιο πρόγραμμα στον Η/Υ. Οι συνήθεις εφαρμογές τέτοιων τροφοδοτικών υπάρχουν για τον αυτοματισμό πειραματικών εφαρμογών και εργασιών και δεν μπορεί ο χρήστης να αλλάζει χειροκίνητα την έξοδο. Στη διπλωματική αυτή μελετηθήκαν δυο ειδών τροφοδοτικά, ένα παλμοτροφοδοτικό τύπου μείωσης της τάσης, step-down ή Buck converter και ένα γραμμικό μείωσης της τάσης, υλοποιήθηκε το γραμμικό λόγω δυσκολιών. Ο τρόπος λειτουργίας του Buck converter και του γραμμικού τροφοδοτικού θα εξηγηθεί σε επόμενο κεφάλαιο, σε αυτό θα εξηγηθεί ο τρόπος λειτουργίας των τροφοδοτικών ως σύνολο.



Εικόνα 1.7: Block Διάγραμμα Παλμοτροφοδοτικού Τροφοδοτικού Διπλωματικής

Και τα δυο τροφοδοτικά λειτουργούν με παρόμοιο τρόπο, ο χρήστης πληκτρολογεί την τιμή της τάσης εξόδου ή στέλνει εντολή μέσω Η/Υ που θέλει, στη συνέχεια ο μικροελεγκτής υπολογίζει και ελέγχει την τιμή εξόδου που θέλει ο χρήστης, στην περίπτωση του παλμοτροφοδοτικού τον έλεγχο τον κάνει ο ψηφιακός PID και στο γραμμικό γίνεται μέσω ενός συγκριτή τάσης .

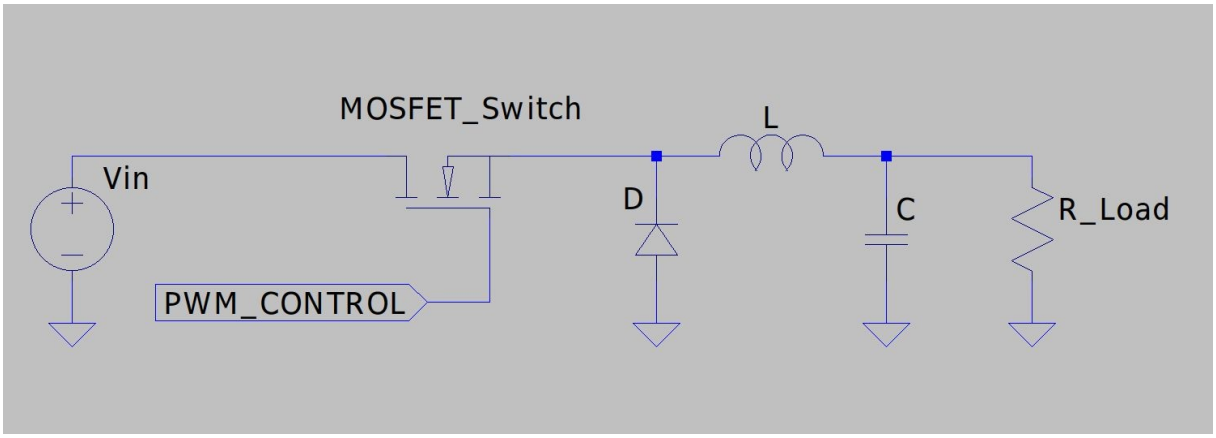


Εικόνα 1.8: Block Διάγραμμα Γραμμικού Τροφοδοτικού Διπλωματικής

Κεφάλαιο 2 Διακοπτικό Τροφοδοτικό

2.1 Αρχή λειτουργίας Διακοπτικού Τροφοδοτικού Buck Converter

Ο Buck Converter είναι η πιο απλή διάταξη στα SMPS μετατρέπει DC τάση σε DC, υποβιβάζοντας την τάση που δέχεται στην είσοδο του σε χαμηλότερη στην έξοδο του, έχοντας σταθερό ρεύμα.

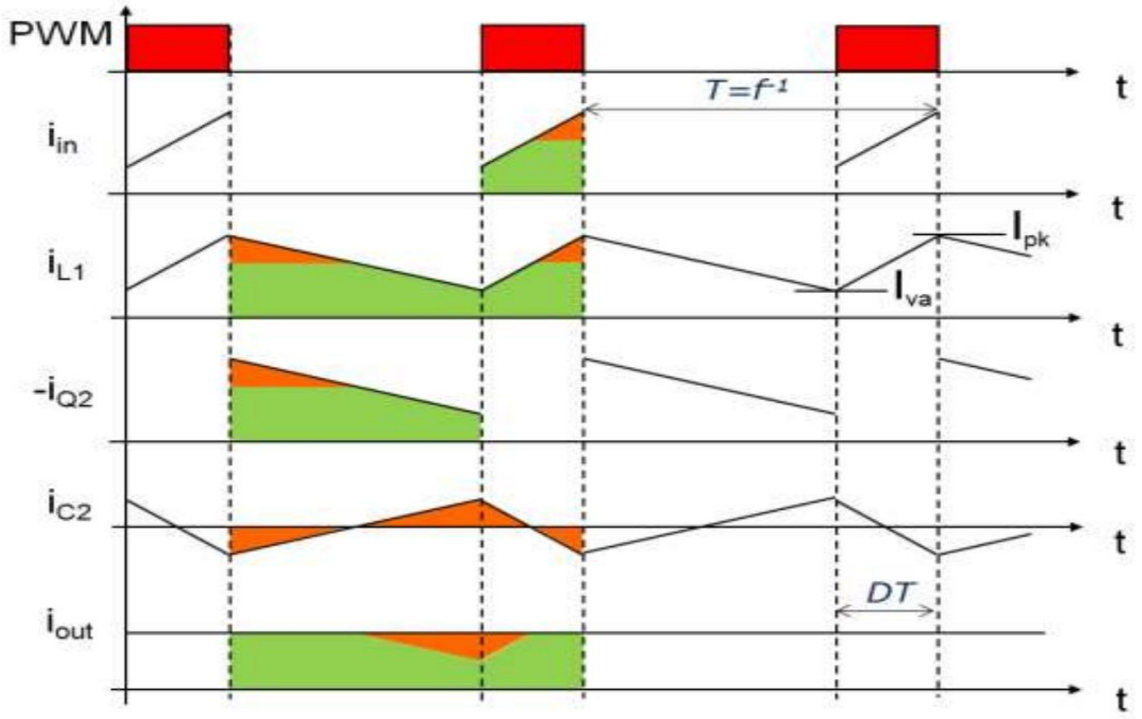


Εικόνα 2.1: Κύκλωμα Buck Convetrer

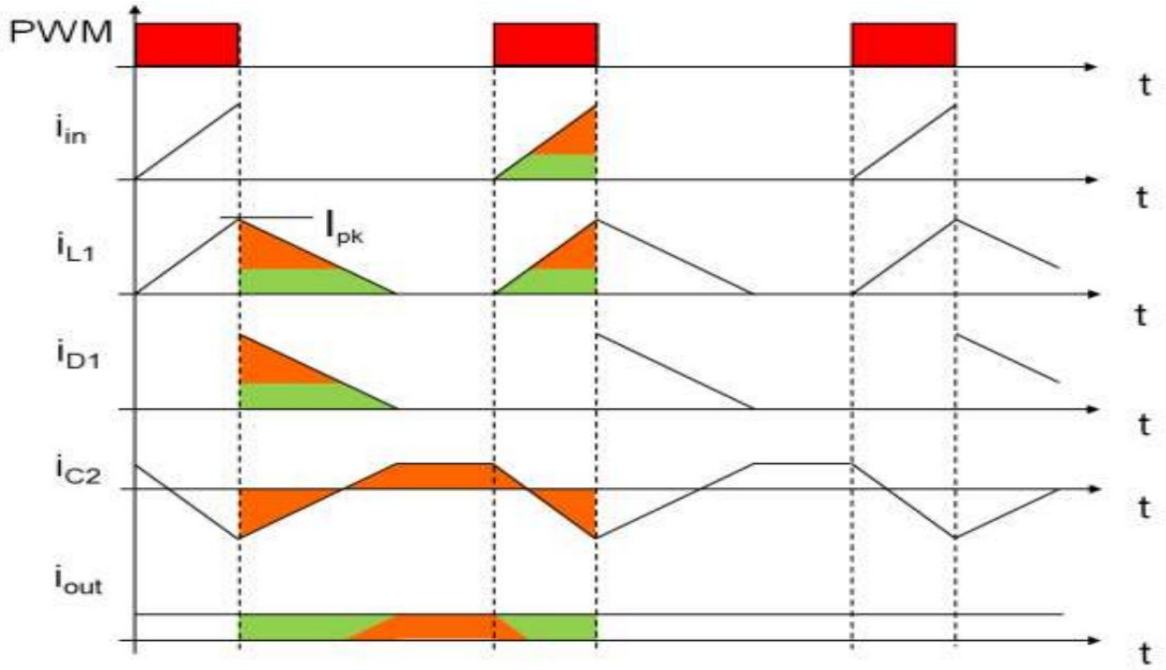
Το κύκλωμα έχει δυο καταστάσεις, όταν ο διακόπτης (MOSFET Switch) είναι ανοιχτός (ON) και όταν είναι κλειστός (OFF). Στη διάρκεια που ο διακόπτης είναι ON η τάση της Vin περνάει μέσω του διακόπτη που φορτίζει γραμμικά το πηνίο L και τον πυκνωτή C έχοντας σταθερή τάση στην έξοδο. Όταν ο διακόπτης είναι OFF το ρεύμα στο πηνίο L δεν αποφορτίζει απευθείας κρατώντας την τάση στο φορτίο R σε χαμηλότερη τιμή. Καθώς αυτό γίνεται, ο πυκνωτής αποφορτίζει μέσω του φορτίου βοηθώντας την τάση εξόδου να μην πέσει. Αφότου περάσει λίγος χρόνος η μια άκρη του πηνίου L γίνεται πιο αρνητική με αποτέλεσμα να άγει η δίοδος D και να αποφορτίσει το πηνίο μέσω του φορτίου.

Η τάση της εξόδου ρυθμίζεται ανάλογα με το εύρος των διακοπών από το PWM στον διακόπτη μαζί με τον συνδυασμό του πηνίου L και του πυκνωτή C που είναι ένα χαμηλοπερατό φίλτρο LC. Το φίλτρο κόβει τις υψηλές συχνότητες που ανοιγοκλείνει ο διακόπτης και αφήνει να περάσουν οι χαμηλές.

Τα περισσότερα SMPS έχουν δυο διαφορετικούς τρόπους λειτουργίας, τη συνεχή περιοχή CCM και την ασυνεχή περιοχή DCM. Τα τροφοδοτικά σχεδιάζονται να λειτουργούν και στις δυο.
Στη συνεχή περιοχή η ένταση του ρεύματος στο πηνίο L μένει θετική καθόλη τη διάρκεια της περιόδου, ενώ στην ασυνεχή η ένταση του ρεύματος στο πηνίο L γίνεται για κάποιο χρονικό διάστημα αρνητική.

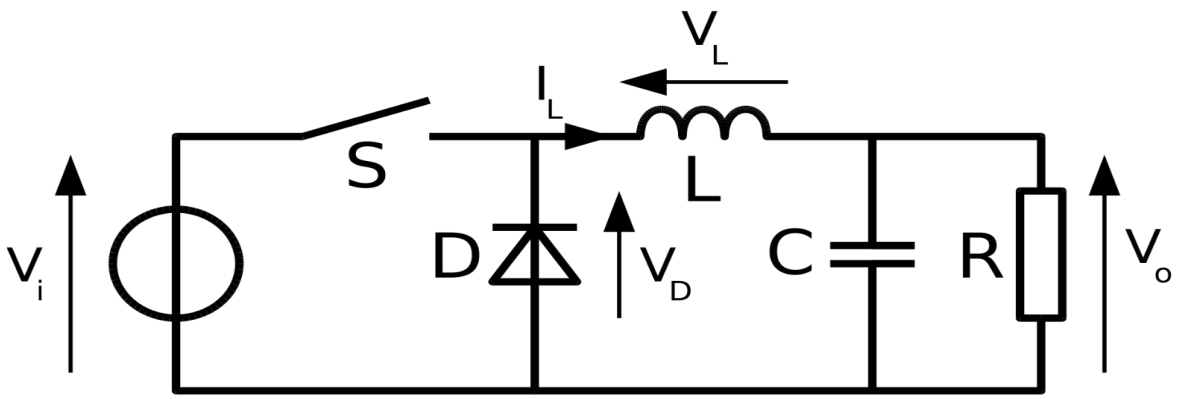


Εικόνα 2.2: Συνεχής Περιοχή Buck Converter με iin : το ρεύμα εισόδου Vin, iL1:το ρεύμα στο πηνίο, iQ2: το ρεύμα στον διακόπτη, iC2: το ρεύμα στον πυκνωτή, iout: το ρεύμα εξόδου



Εικόνα 2.3: Ασυνεχής Περιοχή Buck Converter με iin: το ρεύμα εισόδου Vin, iL1: το ρεύμα στο πηνίο, iQ2: το ρεύμα στον διακόπτη, iC2: το ρεύμα στον πυκνωτή, iout: το ρεύμα εξόδου

2.1.1 Ανάλυση Κυκλώματος Buck Converter



Εικόνα 2.4: Buck Convetrer

Όπως αναφέρθηκε παραπάνω το κύκλωμα έχει δυο καταστάσεις, όταν ο διακόπτης είναι κλειστός ON και όταν είναι ανοιχτός OFF. Ο λόγος του χρόνου σε κατάσταση ON σε σχέση με τη συνολική περίοδο του σήματος (ON+OFF) ονομάζεται Duty cycle (Κύκλος Λειτουργίας). Αν έχουμε για παράδειγμα $D=60\%$, σημαίνει ότι διακόπτης είναι ON για το 60% ή 0.6 της περιόδου T.

Για την ανάλυση του κυκλώματος δεν λαμβάνονται υπόψη ατέλειες υλικών και θεωρούμε ότι λειτουργεί στη συνεχή περιοχή.

Όταν ο διακόπτης είναι ON τότε περνάει ρεύμα μέσω του πηνίου που φορτίζει τον πυκνωτή και έχουμε σταθερή τάση στο φορτίο. Η πηγή όταν είναι ON πολώνει τη δίοδο ανάστροφα. Εφαρμόζοντας τον κανόνα του Kirchoff για την τάση έχουμε:

$$V_i - L\frac{di}{dt} - V_o = 0 \Rightarrow L\frac{di}{dt} = V_i - V_o \Rightarrow \frac{di}{dt} = \frac{V_i - V_o}{L} \qquad (2.1)$$

Όπως φαίνεται και από τις παραπάνω κυματομορφές, εικόνα 2.2 και 2.3 το ρεύμα αυξομειώνεται γραμμικά.

Όταν ο διακόπτης είναι OFF τότε η V_i αποσυνδέεται, σταματάει να διοχετεύει στο κύκλωμα ρεύμα με αποτέλεσμα ο ακροδέκτης του πηνίου που είναι συνδεμένος με το διακόπτη να αρχίσει να γίνεται πιο αρνητικός, η δίοδος πολώνεται ορθά, άρα άγει, ωθεί το ρεύμα του πηνίου να ξεφορτίσει μέσω της διόδου στο φορτίο. Καθώς γίνεται αυτό ο πυκνωτής ξεφορτίζει και αυτός μέσω του φορτίου την ενεργεία που είχε αποθηκεύσει σε μορφή τάσης και έχουμε σταθερή τάση στην έξοδο.

Εφαρμόζοντας το κανόνα του Kirchoff για τάσεις :

$$L\frac{di}{dt} - V_o = 0 \Rightarrow \frac{di}{dt} = \frac{V_o}{L} \qquad (2.2)$$

Από την (2.1) έχουμε την αλλαγή του ρεύματος στο πηνίο κατά την ON κατάσταση:

$$\Delta I_{ON} = \frac{-V_O + V_i}{L} * T_{ON} \qquad (2.3)$$

Από την (2.2) έχουμε την αλλαγή στην OFF κατάσταση:

$$\Delta I_{OFF} = \frac{V_O}{L} * T_{OFF} \tag{2.4}$$

Σε έναν ολοκληρωμένο κύκλο το ρεύμα στο πηνίο θα πρέπει να είναι μηδέν άρα έχουμε:

$$\Delta I_{ON} = \Delta I_{OFF} \tag{2.5}$$

Εξισώνοντας τις (2.3) και (2.4) με την (2.5) έχουμε:

$$\frac{-V_O + V_i}{L} * T_{ON} = \frac{V_O}{L} * T_{OFF} \Rightarrow \frac{V_O}{V_i} = \frac{T_{ON}}{T_{ON} + T_{OFF}} \tag{2.6}$$

Από παραπάνω αναφορά γνωρίζουμε ότι το Duty cycle κύκλος λειτουργίας, είναι το ποσοστό των ON σε σχέση με όλη τη περίοδο:

$$D = \frac{T_{ON}}{T_{ON} + T_{OFF}} \tag{2.7}$$

Από την (2.6) και τη (2.7) έχουμε:

$$D = \frac{V_O}{V_i} \quad \text{ή} \quad V_O = V_i * D \tag{2.8}$$

Από την (2.8) φαίνεται πως το Duty cycle ρυθμίζει την έξοδο του κυκλώματος.

Οι παραπάνω εξισώσεις είναι οι βασικές που περιγράφουν το κύκλωμα, συνδυάζοντας την (2.1) και (2.2) παίρνουμε μια πιο γενική εξίσωση που περιγράφει την αλλαγή του ρεύματος στο πηνίο:

$$\frac{di}{dt} = \frac{x * V_i - V_o}{L} \tag{2.9}$$

με x=1 όταν ο διακόπτης είναι ON και x=0 όταν ο διακόπτης είναι OFF.

Το ρεύμα του κυκλώματος μπορεί να βρεθεί αν ολοκληρωθεί το Duty cycle του ρεύματος:

$$i = \int \frac{di}{dt} * dt \tag{2.10}$$

Το ρεύμα στον πυκνωτή μπορεί να βρεθεί πολλαπλασιάζοντας τον πυκνωτή C το Duty cycle της τάσης:

$$i = C * \frac{dV_o}{dt} \tag{2.11}$$

Από την (2.10) και (2.11) έχουμε:

$$V_o = \frac{1}{C} * \int i_c * dt \tag{2.12}$$

Παίρνοντας υπόψη τον κανόνα του Kirchoff για τα ρεύματα:

$$i_c = i_L - i_o \quad \text{και} \quad i_o = \frac{V_o}{R} \tag{2.13) και (2.14}$$

με i_c το ρεύμα του πυκνωτή ,i_L το ρεύμα του πηνίου και i_o το ρεύμα της εξόδου ή του φορτίου.

Άρα από την (2.13) και (2.14) παίρνουμε:

$$i_c = i_L - \frac{V_o}{R} \tag{2.15}$$

Συνδυάζοντας την (2.1) και (2.9) έχουμε:

$$L\frac{di_L}{dt} = -V_o + V\,i * x \tag{2.16}$$

Συνδυάζοντας την (2.11) και (2.15):

$$\frac{C * dV_o}{dt} = i_L - \frac{V_o}{R} \tag{2.17}$$

[1]

2.1.2 PID Controller

Για να μπορεί το κύκλωμα να διατηρεί σταθερή έξοδο θα χρησιμοποιηθεί ένας ελεγκτής που παίρνει την τιμή της εξόδου και ελέγχει αν αυτή είναι σωστή (τάση ή και ένταση) και αν είναι λάθος να την διορθώσει.

Ο ελεγκτής για να διορθώσει την έξοδο του κυκλώματος παίρνει το σφάλμα, που είναι το αποτέλεσμα της αφαίρεσης της εξόδου και ενός αριθμού αναφοράς που ορίζει ο χρήστης που είναι η θεμιτή έξοδος ή αλλιώς έχουμε αρνητική ανάδραση, και διορθώνει την έξοδο με τις απαραίτητες αλλαγές.

Ο PID - Proportional Integral Derivative controller - είναι ο συνδυασμός τριών ελεγκτών: P αναλογικός, πολλαπλασιάζει το σφάλμα ανάλογα με το μέγεθος του, I ολοκληρωτής που ολοκληρώνει το σφάλμα ανάλογα με το μέγεθος του στη διάρκεια του χρόνου του και D διαφορικός που "μαντεύει" το σφάλμα που θα επέλθει.

Για να βρεθούν οι τιμές του PID θα πρέπει το κύκλωμα του buck converter να αναλυθεί ακολουθώντας την μέθοδο state space, δηλαδή να βρεθεί το μέσο μοντέλο συστήματος του (averaging), που είναι το μαθηματικό του μοντέλο για κάθε κατάσταση στη συγκεκριμένη ON και OFF, 0 και 1. Οι εξισώσεις (2.16), (2.17) και $y = V_o$ (2.18) περιγράφουν ένα δευτεροβάθμιο γραμμικό σύστημα με μορφή $\dot{x} = Ax + bu$ και $y = c^T x$,

με x είναι η κατάσταση του συστήματος $x(t) = \begin{bmatrix} x1(t) \\ x2(t) \\ \vdots \\ xn(t) \end{bmatrix}$,

u η διανυσματική είσοδος του συστήματος $u(t) = \begin{bmatrix} u1(t) \\ u2(t) \\ \vdots \\ un(t) \end{bmatrix}$

και y η διανυσματική έξοδος $y(t) = \begin{bmatrix} y1(t) \\ y2(t) \\ \vdots \\ yn(t) \end{bmatrix}$.

$$\text{Με} \quad A=\begin{bmatrix} 0 & \frac{-1}{L} \\ \frac{1}{C} & \frac{-1}{RC} \end{bmatrix} \quad , \quad b=\begin{bmatrix} \frac{E}{L} \\ 0 \end{bmatrix} \quad \text{και} \quad c^T=\begin{bmatrix} 0 & 1 \end{bmatrix} \tag{2.19}$$

Οι παραπάνω πίνακες βγήκαν με βάση τις (2.17), (2.18) και (2.19), ο πίνακας ελέγχου Kalman είναι

$$C=[b, Ab] \tag{2.20}$$

$$\text{και δίνεται από} \quad C=\begin{bmatrix} \frac{E}{L} & 0 \\ 0 & \frac{E}{LC} \end{bmatrix} \tag{2.21}$$

Το σύστημα είναι ελέγξιμο γιατί η ορίζουσα του πίνακα $\frac{E^2}{L^{2C}} \neq 0$ [6].

Η συνάρτηση μεταφοράς του PID είναι:

$$F_{PID}=K_p\left(1+\frac{1}{T_i}s+T_d s\right) \tag{2.22}$$

Με K_p η μεταβλητή του P, T_i η μεταβλητή του I και T_d η μεταβλητή του D.

Η συνάρτηση μεταφοράς μέσης κατάστασης του συστήματος είναι:

$$V_o\frac{(s)}{U_{av}(s)}=\frac{\frac{E}{LC}}{s^2+\frac{1}{RC}s+\frac{1}{LC}} \tag{2.23}$$

Που είναι αποτέλεσμα των (16), (17), (18) αν λυθεί το κύκλωμα σαν σύστημα πινάκων (που καταλήγει στη (2.19)).

Από την (21) και (22) έχουμε την συνάρτηση μεταφοράς κλειστού βρόγχου:

$$H(s)=\frac{K_p T_d T_i s^2+K_p T_i s+K_p\frac{E}{LC}}{s^3+\frac{1}{RC}+\frac{E K_p+T_d}{LC}s^2+\frac{1+EK_p}{LC}s+\frac{EK_p}{LCT_i}} \tag{2.24}$$

Η χαρακτηριστική εξίσωση του κλειστού συστήματος είναι ο παρανομαστής της (23):

$$s^3+\frac{1}{RC}+\frac{E K_p+T_d}{LC}s^2+\frac{1+EK_p}{LC}s+\frac{EK_p}{LCT_i}=0 \tag{2.25}$$

Οι μεταβλητές K_p, T_i και T_d διαλέγονται ώστε η (23) να γίνει ένα πολυώνυμο Hurwitz τρίτου βαθμού

$$p(s)=(s^2+2\zeta\omega_n s+\omega_n)(s+\alpha)=s^3+s^2(2\zeta\omega_n\alpha)+s(\omega_n^2+2\zeta\omega_n\alpha)+\omega_n^2\alpha \tag{2.26}$$

με $\zeta=\frac{\sqrt{LC}}{2R}$ είναι ο βαθμός απόσβεσης και $\omega_n=\frac{1}{2\pi\sqrt{LC}}$ είναι η φυσική συχνότητα σε Hz.

Εξισώνοντας τις (2.25) και (2.26) έχουμε:

$$K_p = \frac{2\zeta\omega_n \alpha LC + \omega_n^2 LC - 1}{E} \tag{2.27}$$

$$T_i = \frac{EK_p}{LC\alpha\omega_n^2} \tag{2.28}$$

$$T_d = \frac{LC}{EK_p}\left(\alpha + 2\omega_n - \frac{1}{RC}\right) \tag{2.29}$$

Το α βρέθηκε προσεγγιστικά να είναι $\alpha \geq \frac{1}{LC}$, εξισώνοντας τις (2.27), (2.28) και (2.29) ίσες με το μηδέν[2].

2.2 Προσομοιώσεις

2.2.1 Gnu Octave εύρεση τιμών υλικών, PID και ευστάθειας

Ο τρόπος εύρεσης των τιμών των υλικών του κυκλώματος και των μεταβλητών καθώς και η ευστάθεια θα βρεθούν με τη βοήθεια του προγράμματος Gnu Octave, όπου τρέχουν scipts (κώδικες) για αριθμητική ανάλυση.

Όπως είχε προαναφερθεί τα SMPS λειτουργούν σε δυο περιοχές, την συνεχή και την ασυνεχή. Στην πρώτη το ρεύμα δεν μηδενίζει ενώ στη δεύτερη μηδενίζει για κάποιο χρόνο, αυτό το όριο ονομάζεται I_{lim} το κατώτατο όριο δηλαδή, στο σχήμα 2 ονομάζεται I_{va}, και ισούται με:

$$I_{limO} = L_{limL} = \frac{1}{2} I_{L(max)} = \frac{V_o(1-D)T_s}{2L} \tag{2.30}$$

Από την (2) θα βρεθεί το μέγεθος του πηνίου:

$$L = \frac{u_L(1-D)T_s}{\Delta I} \tag{2.31}$$

Με u_L η τάση του πηνίου ίση με V_o και ΔI το ripple του ρεύματος στην έξοδο, που είναι το πόσο αυξομειώνεται το ρεύμα στην έξοδο, διαλέγει ο σχεδιαστής του κυκλώματος το πόσο μικρό θέλει να είναι αλλά όσο μικρότερο το ΔI τόσο μεγαλύτερο γίνεται το πηνίο.

Για να βρεθεί το μέγεθος του πυκνωτή θα δοθεί από τον τύπο:

$$V_{ripple(p-p)} = ESR_c * \Delta I \tag{2.32}$$

Και από τον τύπο:

$$C = \frac{65 * 10^{-6}}{ESR_c} \tag{2.33}$$

Με $V_{ripple(p\text{-}p)}$ το ripple της τάσης εξόδου που επιλέγεται από τον σχεδιαστή και ESR_c είναι η αντίσταση σε σειρά του πυκνωτή.

Τα αποτελέσματα του script εύρεσης υλικών είναι:

Πυκνωτής C	650uF
ESR_c	0.1Ω
Πηνίο L	1,1mH
Αντίσταση εξόδου R	4Ω
Duty Cicle ποσοστό επί τοις εκατό	42%
I_{ilm}	0.1A

Πίνακας 1: Μεγέθη Υλικών Κυκλώματος

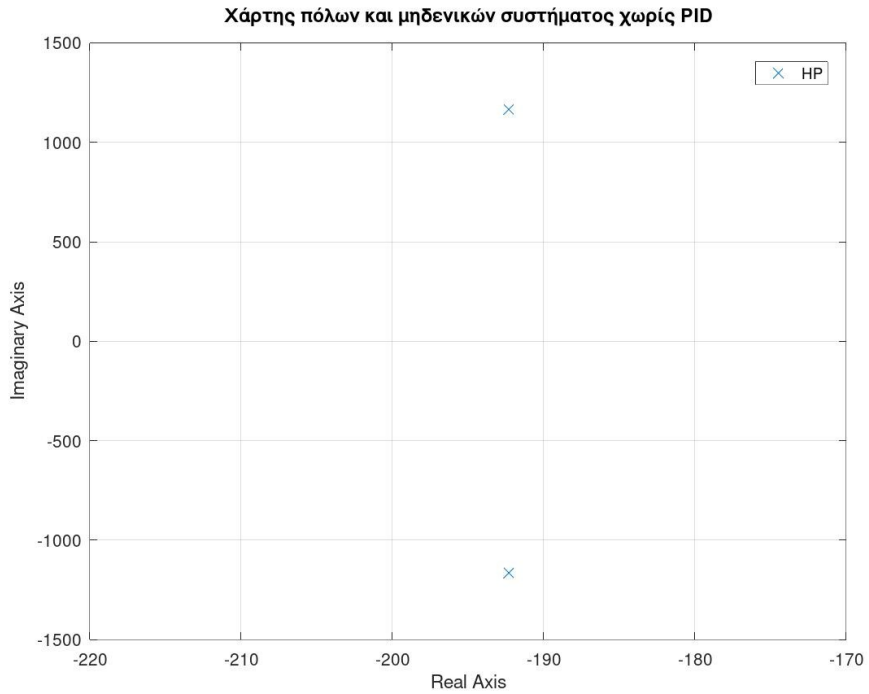
Τα αποτελέσματα του script εύρεσης ευστάθειας είναι:

K_p	3.1705
T_i ή K_i	0.0017046 ή 1859.9
T_d ή K_d	0.016601 ή 0.052634
Αποτέλεσμα ευστάθειας κυκλώματος χωρίς PID	bool=1 (Ευσταθές)
Αποτέλεσμα ευστάθειας κυκλώματος με PID	bool2=1 (Ευσταθές)

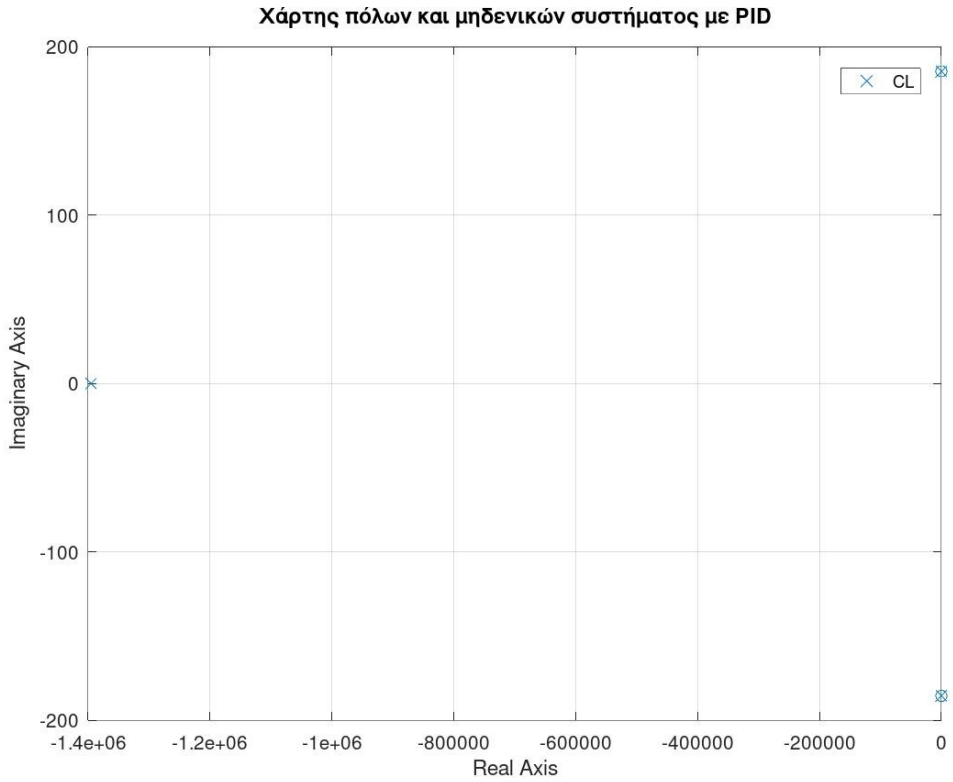
Πίνακας 2: Αποτελέσματα Ευστάθειας και PID

Οι πόλοι και τα μηδενικά του συστήματος χωρίς PID: p1=-192.31 + 1165.4i και p2=-192.31 – 1165.4i δεν έχει μηδενικά.

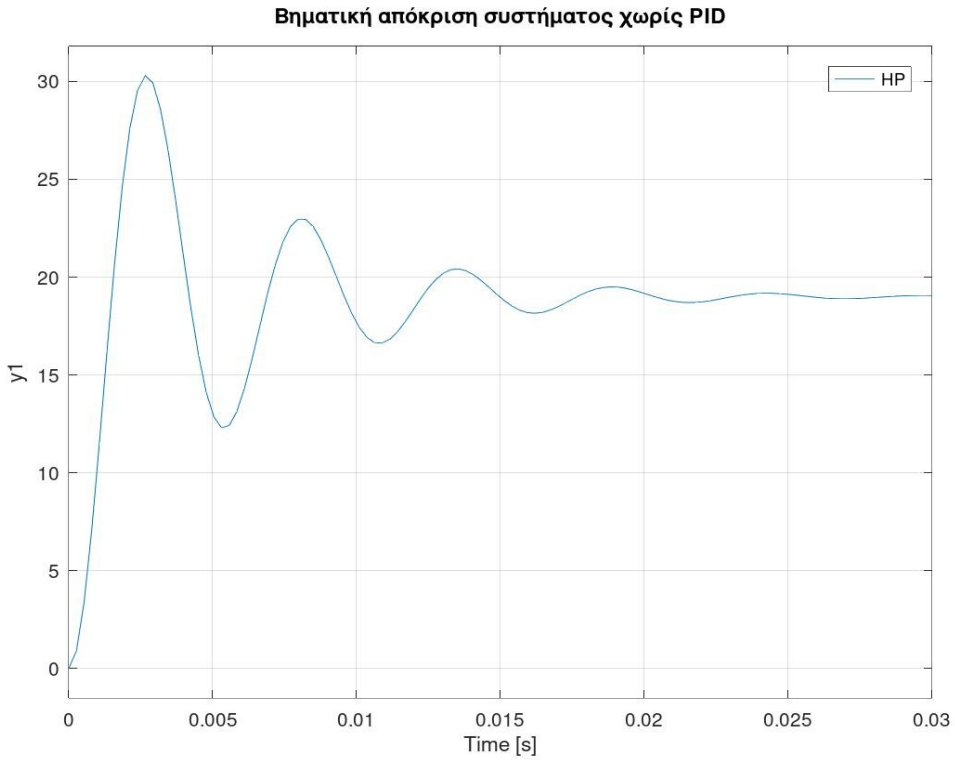
Οι πόλοι και τα μηδενικά του συστήματος με PID: p1=-1.3951e+06 + 0i,p2=-30.607 + 185.48i, p3=-30.607 – 185.48i μηδενικά z1=-30.126 + 185.58i και z2=-30.126 – 185.58i.



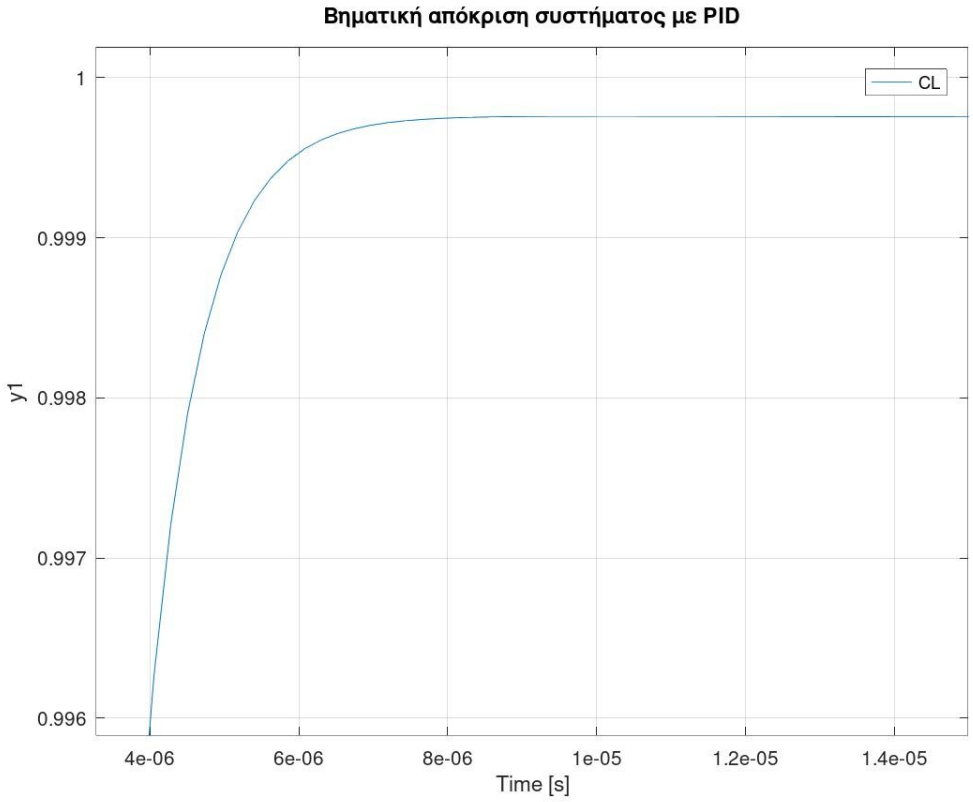
Εικόνα 2.5: Οι πόλοι του Συστήματος Βρίσκονται στην Αριστερή πλευρά του Άξονα άρα το Σύστημα είναι Ευσταθές



Εικόνα 2.6: Οι πόλοι του Συστήματος Βρίσκονται στην Αριστερή πλευρά του Άξονα άρα το Σύστημα είναι Ευσταθές

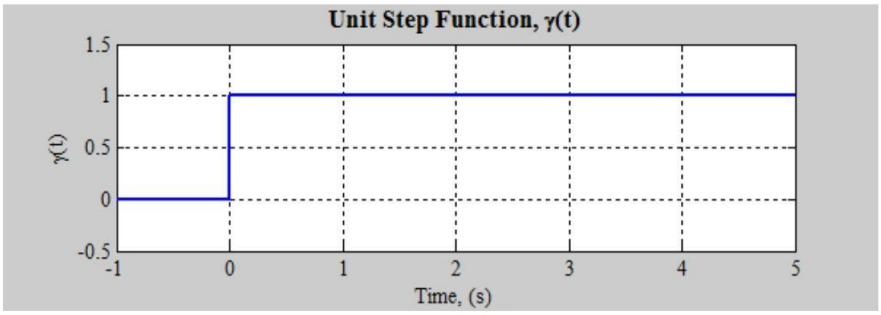


Εικόνα 2.7: Το Σύστημα Έχει στην Αρχή Ταλαντώσεις και μετά από κάποιο Χρόνο Σταθεροποιείται στην Τιμή της Τάσης Εισόδου Vi



Εικόνα 2.8: Το Σύστημα Μέσα σε Πολύ Λίγο Χρόνο Σταθεροποιείται Λόγω του PID

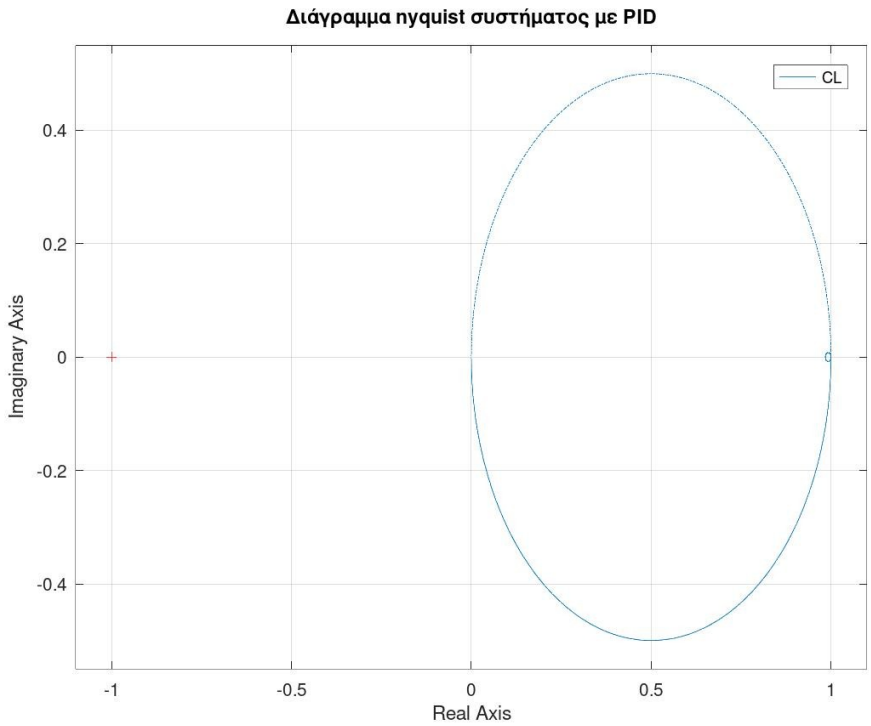
$$\gamma(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$



Εικόνα 2.9: Βηματική Συνάρτηση

Βηματική απόκριση είναι το αποτέλεσμα όταν σε ένα σύστημα βάζουμε ως είσοδο τη βηματική συνάρτηση.

Μια συνάρτηση μεταφοράς G(s) είναι ευσταθής αν το διάγραμμα Nyquist της περιστρέφεται γύρω από το 0 N = Z φορές όπου Z ο αριθμός των μηδενικών της G(s) στο δεξιό μιγαδικό ημιεπίπεδο. Από την εντολή [pCL,zCL]=pzmap(CL), παίρνουμε τους πόλους και τα μηδενικά του συστήματος με PID: z1=-30.118 + 185.55i και z2=-30.118 – 185.55i. Παρατηρούμε ότι δεν έχει μηδενικά δεξιά του μιγαδικού ημιεπίπεδου, αλλά δεν το καθιστά ασταθές.



Εικονα 2.10: Το Σύστημα Είναι Ευσταθές Γιατί Περιστρέφεται Γύρω από το Μηδέν Όσα Είναι και Μηδενικά στο δεξί Μιγαδικό Ημιεπίπεδο δηλαδή 0

Για να βρεθεί η ευστάθεια ενός συστήματος G(s) μέσω διαγράμματος Bode θα χρειαστούν να βρεθούν τρία μεγέθη: το περιθώριο κέρδους Gain Margin ή G_m , το περιθώριο φάσης Phase Margin P_m και το crossover frequency συχνότητα διασταύρωσης ή ω_c. Η συχνότητα διασταύρωσης βρίσκεται από το διάγραμμα Bode μεγέθους ή magnitude όταν αυτό γίνει ίσο με το μηδέν:

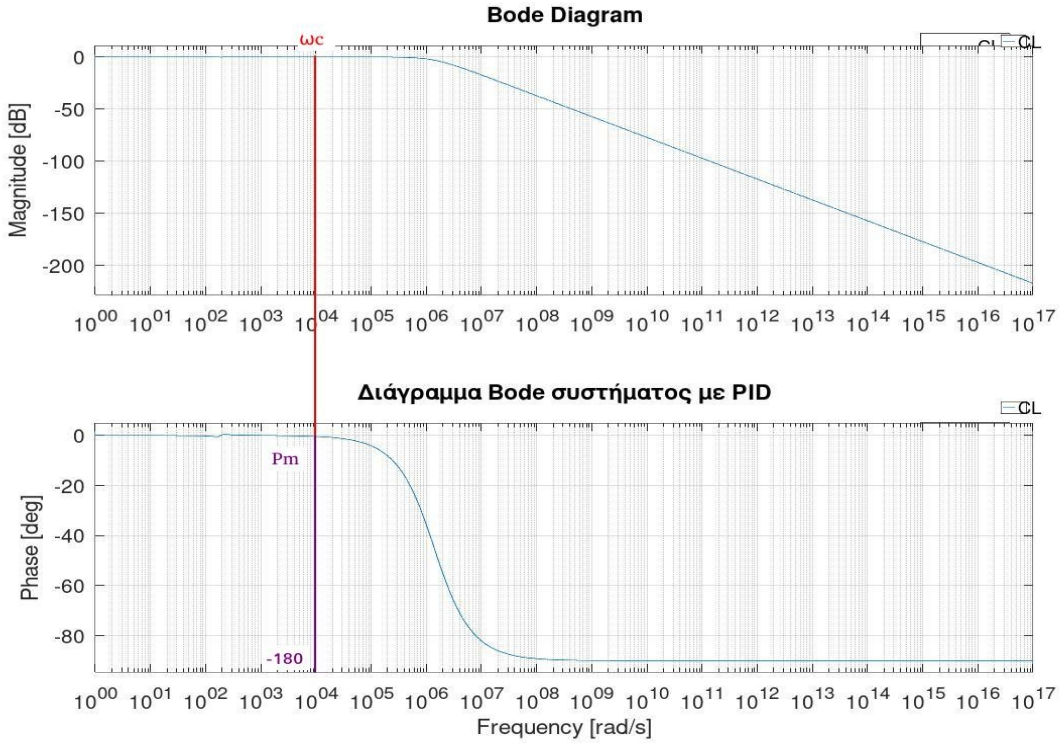
$$\left|G(j\omega_c)\right| = 0\,dB \tag{2.34}$$

Το P_m είναι η γωνία του crossover frequency, μετράει πόσο πάνω από τις -180^o μοίρες βρίσκεται ακόμα η φάση (phase) που μπορεί να προσθέσει στον controller πριν το σύστημα γίνει ασταθές. Όσο πιο θετικές τιμές τόσο πιο ευσταθές είναι, αν είναι αρνητικές το σύστημα είναι ασταθές.

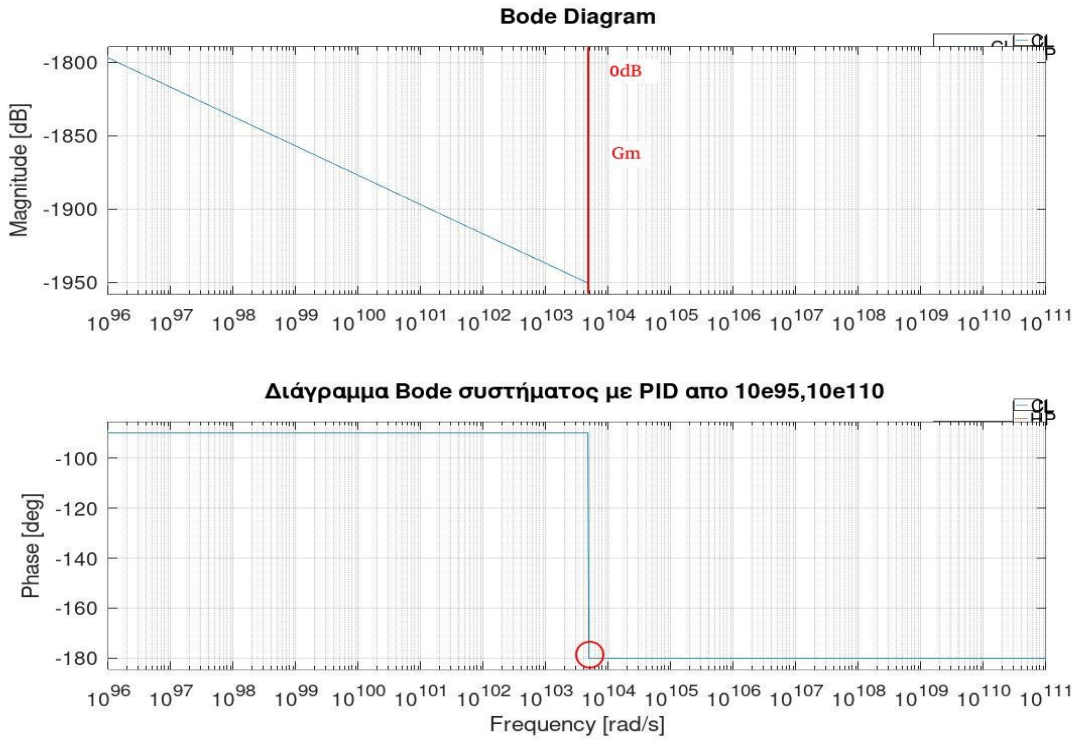
$$P_m = \measuredangle G(j\omega_c) - (-180^o) \tag{2.35}$$

Το G_m είναι το κέρδος που μπορεί να προστεθεί ώστε το magnitude να φτάσει στα 0dB, όταν η φάση περάσει τις -180^o ώστε να μην γίνει το σύστημα ασταθές. Όσο πιο θετική τιμή τόσο πιο ευσταθές είναι το σύστημα καθώς επίσης και θα υπάρχουν λιγότερες ταλαντώσεις. Ενώ αν είναι αρνητική η τιμή το σύστημα είναι ασταθές.

$$G_m = -\left|G(j\omega_c)\right| \tag{2.36}$$



Εικόνα 2.11: Σε αυτό το διάγραμμα Bode φαίνεται το ως καθώς και το Pm



Εικόνα 2.12: Σε αυτό το διάγραμμα Bode φαίνεται το Gm

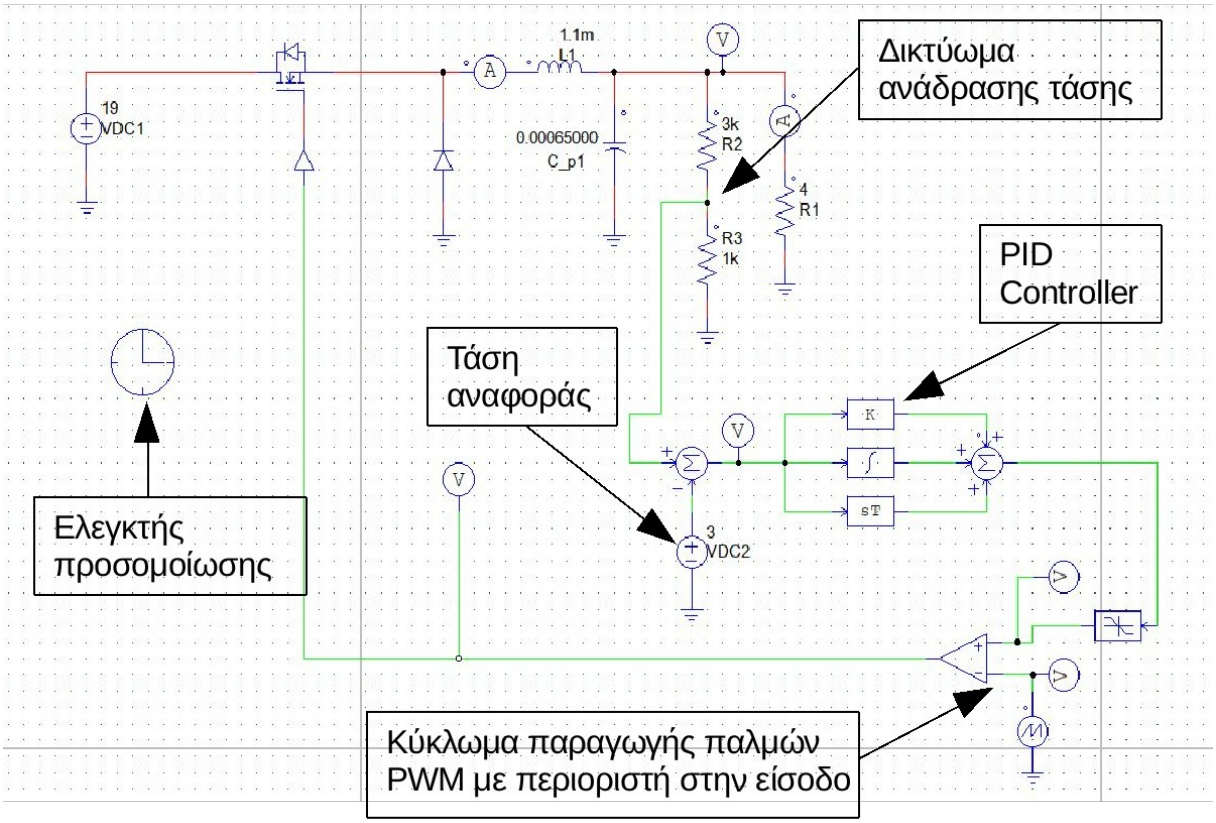
Από τα διαγράμματα Bode Εικόνα 2.11, Εικόνα 2.12 και από την εντολή [mag, pha] =bode(CL,{10e-1,10e170}); βρίσκουμε τις τιμές των (2.34) και (2.35) και έχουμε:

$$P_m = \angle G(j\omega_c) - (-180^o) = 0 + 180^o = 180^o \quad \text{και} \quad G_m = -\left|G(j\omega_c)\right| = 1950\,dB$$

Από τα παραπάνω διαγράμματα συμπεραίνουμε ότι το σύστημα μαζί με τον PID είναι ευσταθές.

2.2.2 Προσομοίωση σε PSIM

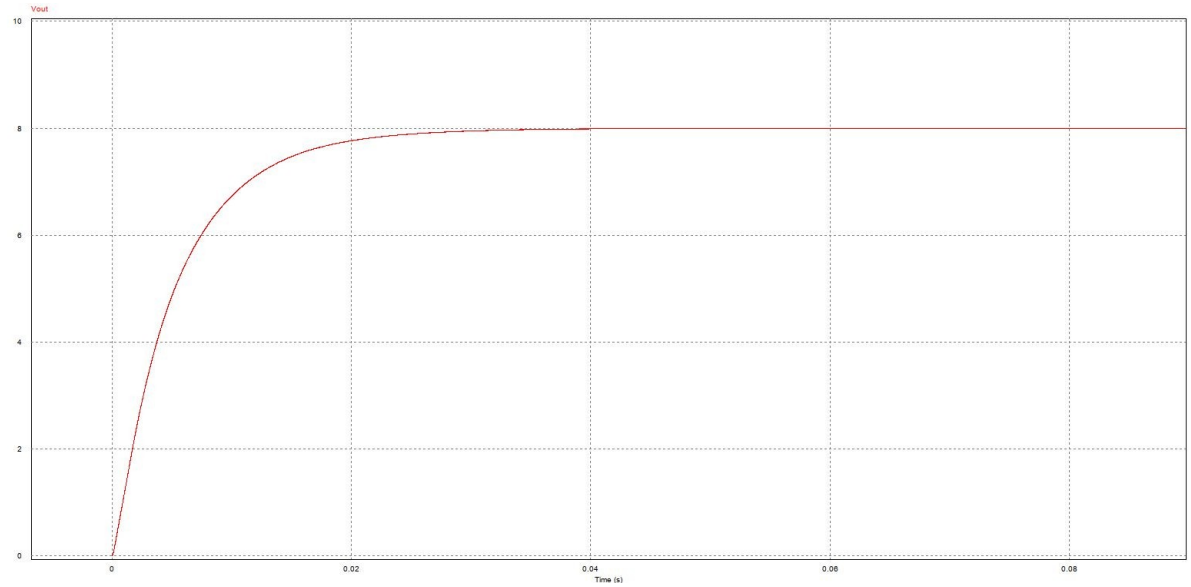
Το PSIM είναι ένα πρόγραμμα προσομοίωσης ηλεκτρονικών κυκλωμάτων ισχύος και μοντέλων. Στο πρόγραμμα σχεδιάζουμε το κύκλωμα του Buck και του PID, δεν μπορεί να προσομοιώσει τον κώδικα του μικροελεγκτή οπότε τα αποτελέσματα που θα πάρουμε είναι προσεγγιστικά. Το πρόγραμμα χρησιμοποιείται σαν βοηθητικό εργαλείο θεωρητικών μετρήσεων. Στη συνέχεια προσομοιώνουμε το μοντέλο του κυκλώματος για να πάρουμε το διάγραμμα Bode.



Εικόνα 2.13: Κύκλωμα Buck Converter και PID Controller στο PSIM

Στην προσομοίωση ο PID παίρνει το σφάλμα που είναι αποτέλεσμα της αφαίρεσης ενός μέρους της τάσης εξόδου από τον διαιρέτη τάσης ή δικτύωμα ανάδρασης τάσης και την τάση αναφοράς, μετά υπολογίζει το πόσο χρειάζεται να αυξομειωθεί η τάση εξόδου για να έχουμε μηδενικό σφάλμα.

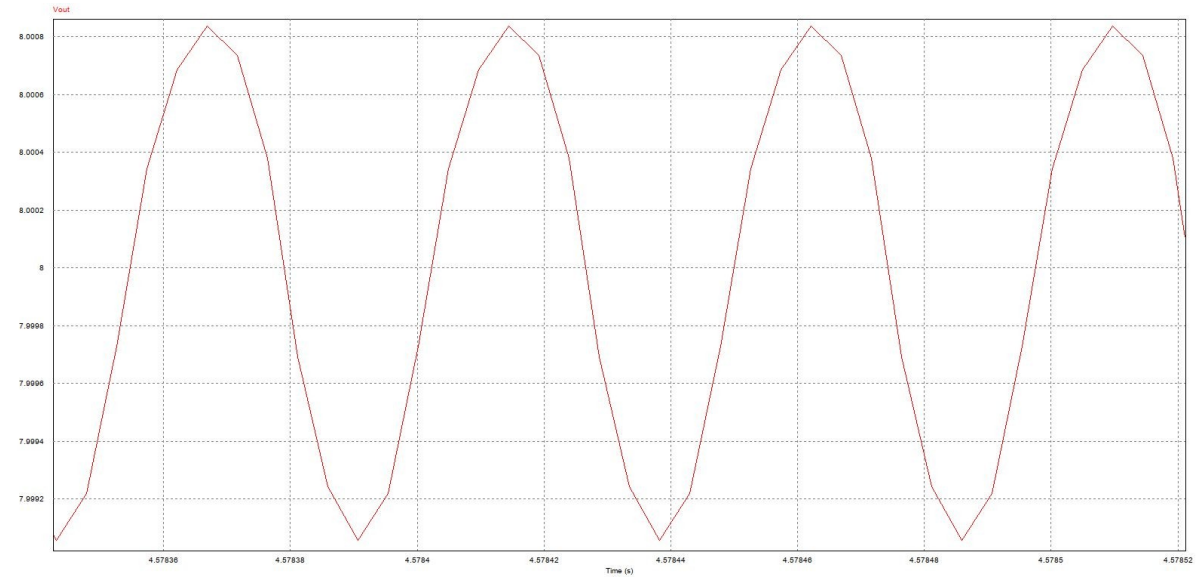
Το αποτέλεσμα περνάει από τον περιοριστή γιατί το αποτέλεσμα του ελεγκτή μπορεί να είναι πολύ μεγάλο για συγκριθεί με την τριγωνική τάση για να παράγει PWM παλμούς, που στο τέλος οδηγούν το MOSFET. Ως τάση αναφοράς είναι τα 2V και περιμένουμε 8V στην έξοδο με ρεύμα 2A.



Εικόνα 2.14: Τάση Εξόδου Προσομοίωσης

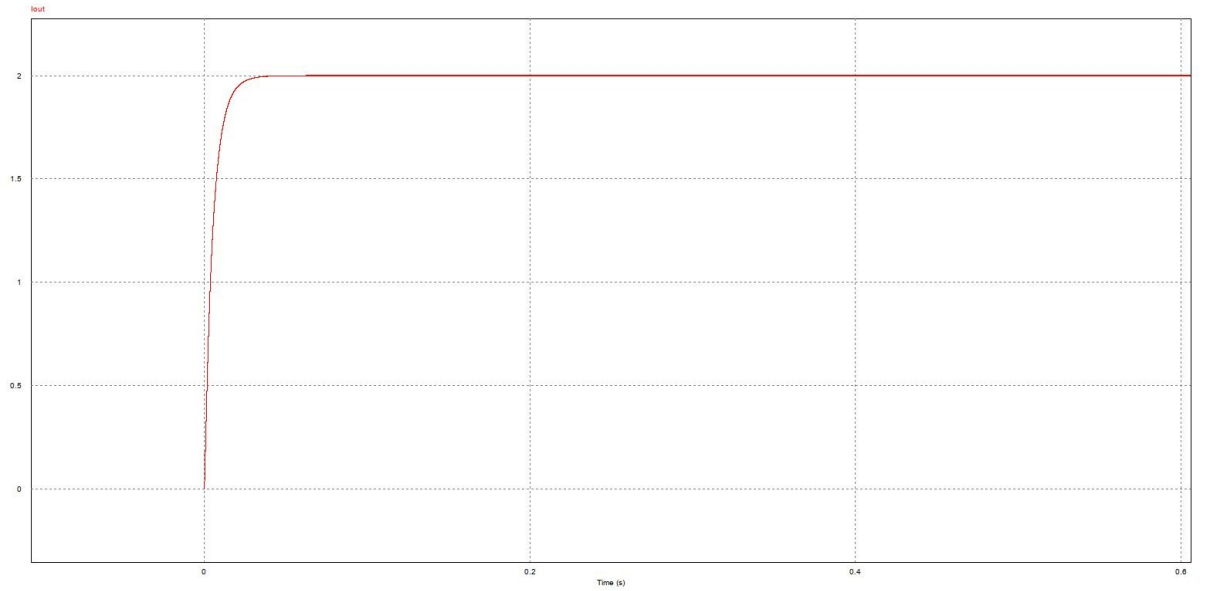
Παρατηρούμε ότι στην έξοδο παίρνουμε την τιμή των 8V βάζοντας ως τάση αναφοράς 2V που είναι αυτό που θέλαμε. Ο λόγος που διαιρούμε την τάση εξόδου δια 4 είναι γιατί ο μικροελεγκτής μπορεί να μετρήσει τάσεις μέχρι τα 5V. Αν είναι μεγαλύτερες θα καεί. Άρα έχουμε:

$$V_{out} = V_{ref} * 4 \tag{2.37}$$



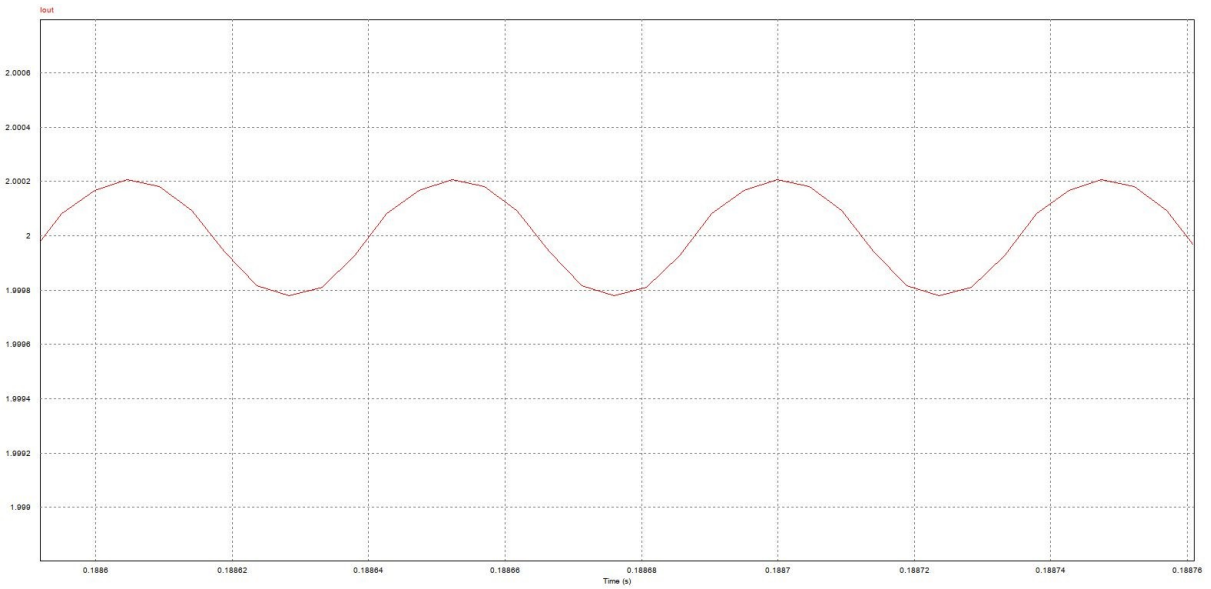
Εικόνα 2.15: Μεγενθυμένη Τάση Εξόδου

Από την παραπάνω εικόνα φαίνεται το V_{ripple} το οποίο είναι ίσο με 1.8mV πολύ πιο μικρό από αυτό που βάλαμε στο Octave. Αυτό συμβαίνει γιατί στο PSIM δεν βάλαμε πραγματικά μοντέλα, όμως δεν δημιουργεί κάποιο πρόβλημα.



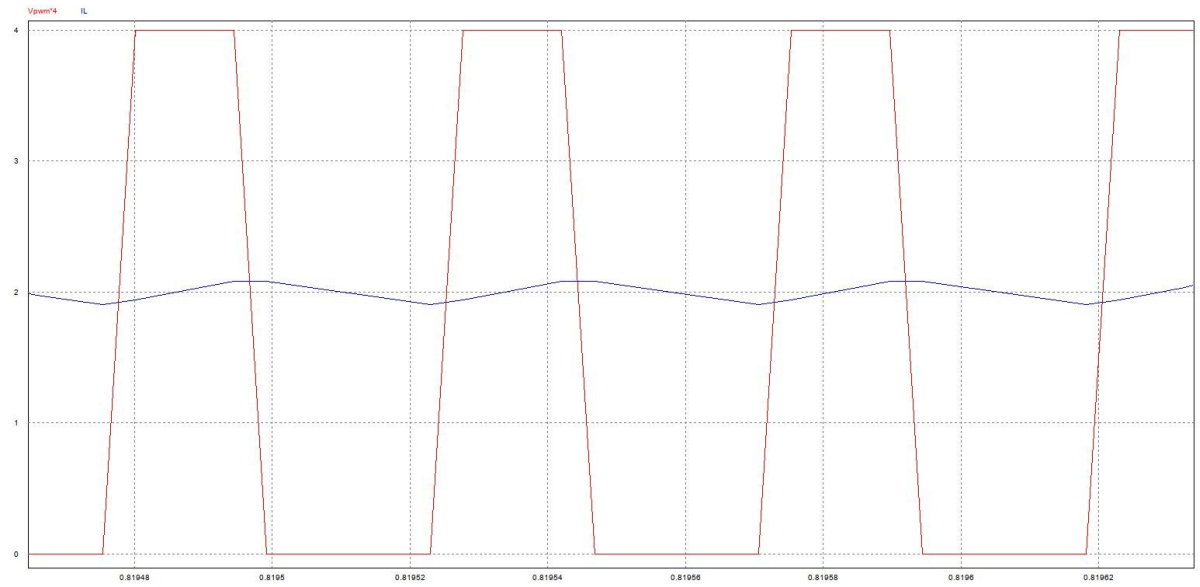
Εικόνα 2.16: Ρεύμα Εξόδου Προσομοίωσης

Το ρεύμα στην έξοδο είναι αυτό που περιμέναμε 8V/4Ω=2A.



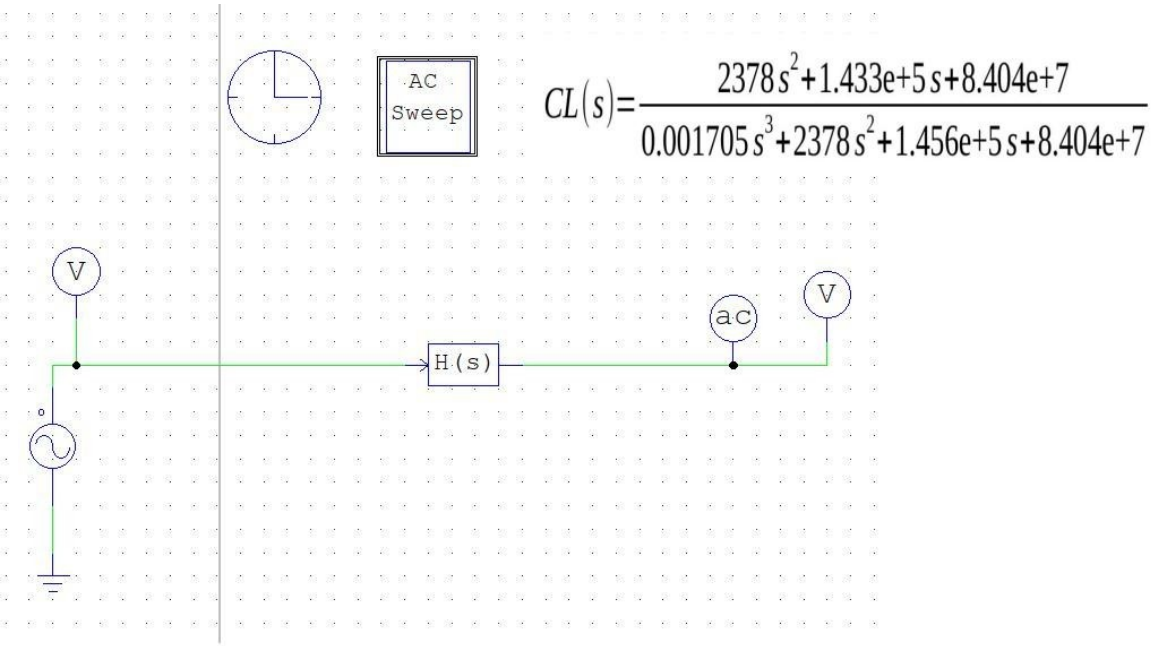
Εικόνα 2.17: Μεγενθυμένο Ρεύμα Εξόδου

Παρατηρούμε ότι το I_{ripple} είναι και αυτό μικρότερο από ότι θέλαμε 4mA εξαιτίας του παραπάνω λόγου.



Εικόνα 2.18: Ρεύμα Στο Πηνίο Σε Σύγκριση με τον Παλμό PWM

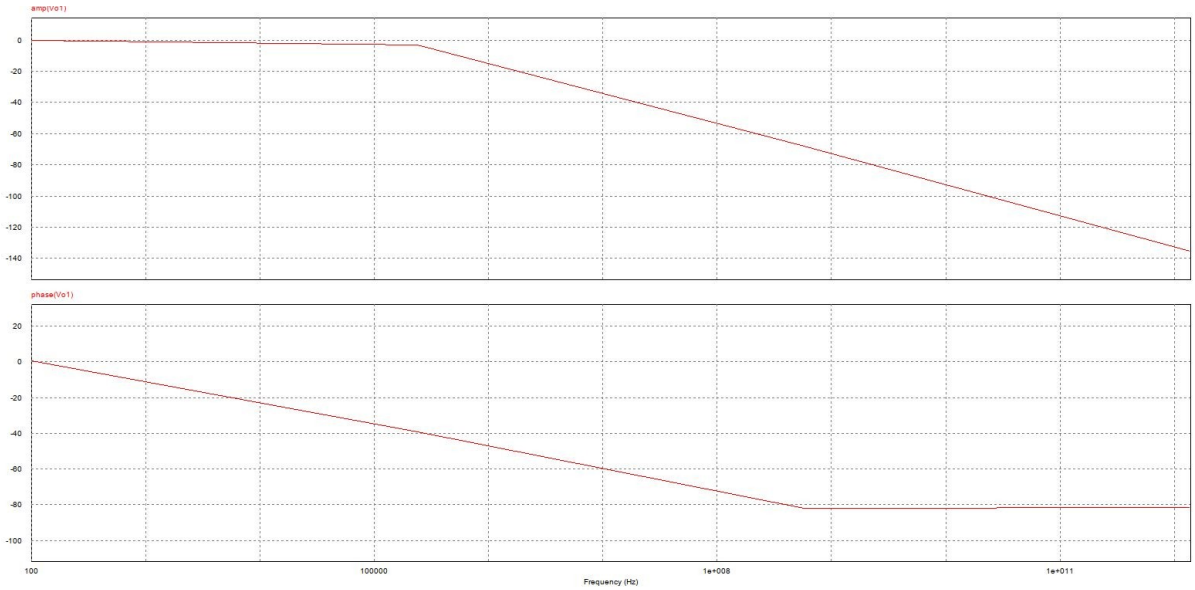
Το ρεύμα στο πηνίο φορτίζει όταν ο PWM παλμός είναι ON και ξεφορτίζει όταν είναι OFF.



Εικόνα 2.19: Μοντέλο του Κυκλώματος με PID

Παρατηρούμε ότι το διάγραμμα Bode εικόνα 2.1 κέρδους είναι παρόμοιο με εκείνο της εικόνας 2.18 Το διάγραμμα της φάσης, αν και τα νούμερα είναι σωστά, το σχήμα είναι διαφορετικό. Αυτό ευθύνεται στο μη επαρκές πλήθος σημείων προσομοίωσης, εφόσον όσα περισσότερα σημεία τόσο πιο λεπτομερές είναι το διάγραμμα.

Από τα παραπάνω συμπεραίνουμε ότι το κύκλωμα είναι ευσταθές και θεωρητικά θα λειτουργεί σωστά.



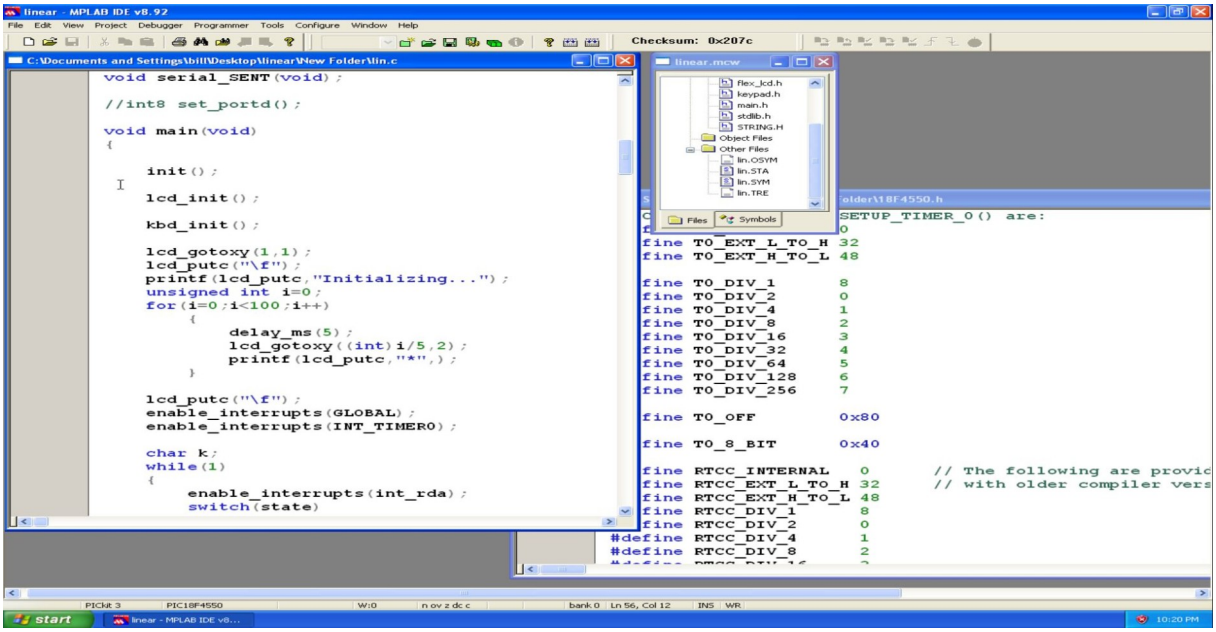
Εικόνα 2.20: Διάγραμμα Bode Προσομοίωσης Μοντέλου Κυκλώματος

Κεφάλαιο 3 Προγραμματισμός Μικροελεγκτή ως Διακοπτικό Τροφοδοτικό

Σε αυτό το κεφαλαίο θα εξηγηθεί ο κώδικας του μικροελεγκτή καθώς και η εφαρμογή που χρησιμοποιήθηκε και το μέσο προγραμματισμού. Ο κώδικας θα εξηγηθεί σε υποενότητες για καλύτερη κατανόηση.

3.1 Εφαρμογή και τρόπος Προγραμματισμού

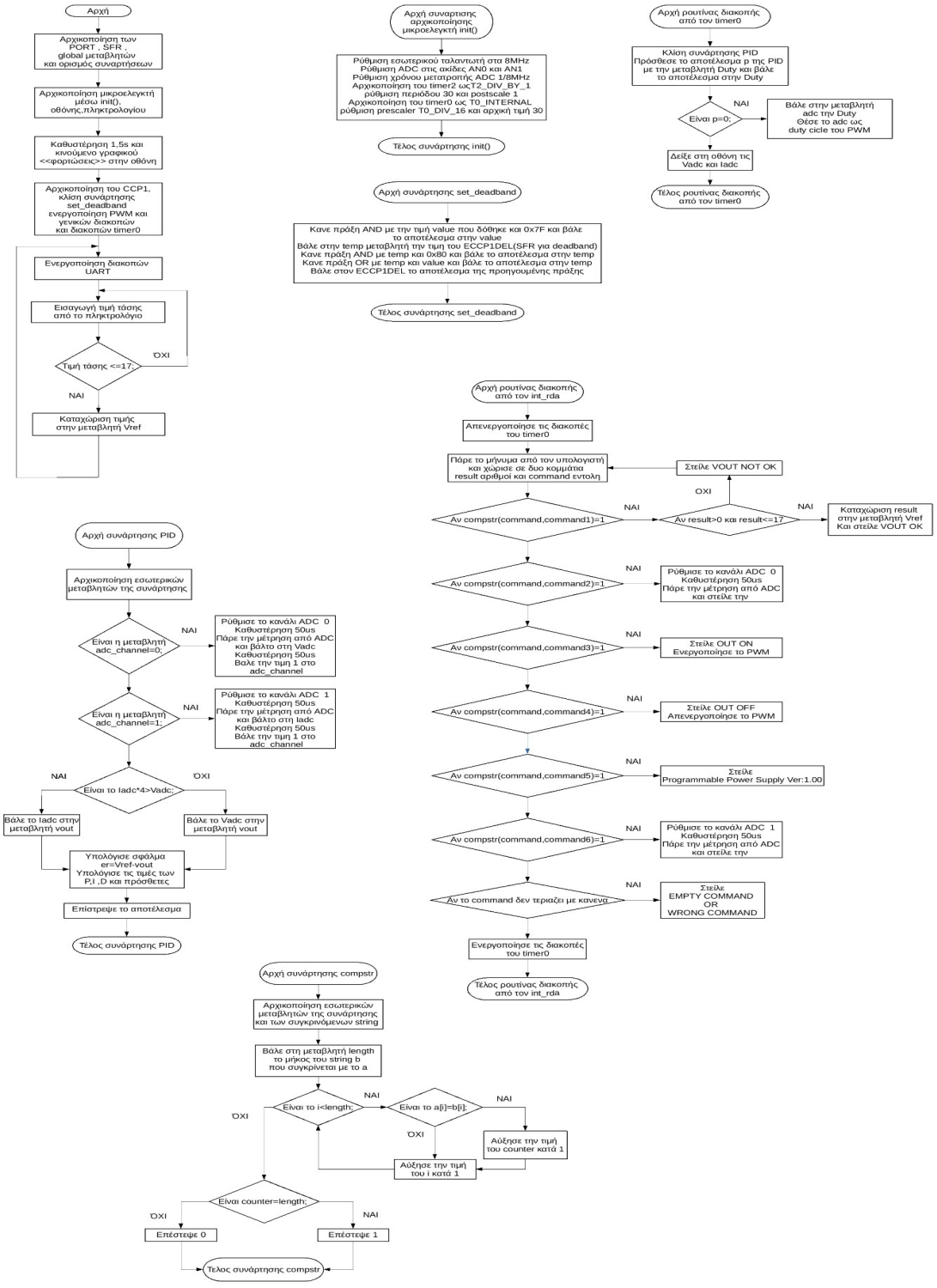
Η εφαρμογή που χρησιμοποιήθηκε είναι το MPLAB 8.92 της Microchip με γλώσσα κώδικα C της CCS PIC PCWHD v5.008. Επιλέχθηκε αυτή η εφαρμογή λόγω της εξοικείωσης μας στον προγραμματισμό και της σταθερότητας της να προγραμματίζει μικροελεγκτές της σειράς που επιλέξαμε. Το μέσο που χρησιμοποιήθηκε για να προγραμματίσουμε τον μικροελεγκτή είναι ο PICkit3 της MICROCHIP .



Εικόνα 3.1: Πρόγραμμα MPLAB

Εικόνα 3.2: Προγραμματιστής PICkit3

3.2 Flowchart Προγράμματος



Εικόνα 3.3: Διάγραμμα Ροής Διακοπτικού Τροφοδοτικού

3.3 Κύριο Πρόγραμμα

Το κύριο πρόγραμμα αρχικά καλεί τη συνάρτηση init() η οποία αρχικοποιεί τον μικροελεγκτή, στη συνέχεια καλεί τη συνάρτηση για την αρχικοποίηση της οθόνης και του πληκτρολογίου.

Κατόπιν, γράφει στην πρώτη σειρά της οθόνης Initializing... και κάθε 15ms βάζει έναν αστερίσκο * στη δεύτερη σειρά της οθόνης.
Αυτό γίνεται γιατί κατά τη διάρκεια των 1,5 δευτερολέπτων του γραφικού που τρέχει, καθυστερεί τον μικροελεγκτή ώστε να δώσει χρόνο στο κύκλωμα ισχύος του buck converter να έρθει σε σταθερή κατάσταση. Αν δεν υπήρχε η καθυστέρηση μπορεί να καιγόντουσαν οι ακίδες του μικροελεγκτή ή ο οδηγός του MOSFET.
Το γραφικό προγραμματίστηκε για να μην φαίνεται ότι ο μικροελεγκτής έχει κολλήσει.

Στη συνέχεια ρυθμίζουμε τον CCP1 να λειτουργεί σαν Half Bridge PWM και έχουμε δυο παλμούς συμπληρωματικούς. Καλεί την set_deadband(4) ώστε να υπάρχει καθυστέρηση μεταξύ των παλμών. Ο CCP1 είναι ένας SFR ο οποίος μπορεί να λειτουργήσει σαν PWM, συγκριτής και για σύλληψη τιμών.

Μετά, το πρόγραμμα “παγιδεύει” μέσα σε έναν ατέρμονα φθόγγο την ενεργοποίηση των διακοπών int_rda και την εισαγωγή χαρακτήρων από το πληκτρολόγιο. Ο λόγος που γίνεται αυτό είναι για να τρέχει το πρόγραμμα συνέχεια ώστε οποιαδήποτε στιγμή μπορεί να εισαχθεί η τιμή της τάσης. Η int_rda ενεργοποιείται συνέχεια γιατί όταν είναι η στιγμή να διακόψει απενεργοποιείται, άρα χρειάζεται να ενεργοποιείται συνέχεια.
Ο λόγος που απενεργοποιείται θα εξηγηθεί σε επομένη ενότητα.

Για να εισαχθεί η τάση εξόδου από το πληκτρολόγιο χρησιμοποιήθηκε η switch-case, είναι μια εντολή με την οποία το πρόγραμμα επιλέγει καταστάσεις που έχουμε ορίσει, μια για κάθε περίπτωση case. Στο πρόγραμμα είναι τρεις για κάθε ψηφίο.

```
void main(void)
{

    init();
    lcd_init();
    lcd_gotoxy(1,1);
    lcd_putc("\f");
    printf(lcd_putc,"Initializing...");
    int i=0;
    for(i=0;i<100;i++)
        {
            delay_ms(15);
            lcd_gotoxy((int)i/6,2);
            printf(lcd_putc,"*",);
        }

    lcd_putc("\f");
    setup_ccp1(CCP_PWM_HALF_BRIDGE|CCP_PWM_H_H);
    set_deadband(4);
    enable_interrupts(global);
    enable_interrupts(INT_TIMER0);

    while(1)
    {
        enable_interrupts(int_rda);
        switch(state)
                {
                case 1:
                       k=kbd_getc();
                       while (k!=0)
                       {
                           lcd_gotoxy(1,2);
                           printf(lcd_putc,"      ");
                           N1= k&0b00001111;
                           lcd_gotoxy(1,2);
                           printf(lcd_putc,"%d",N1);
                           state = 2;
                           break;
                       }
                 break;
                 case 2:
                     k=kbd_getc();
                     while (k!=0)
                     {
                          N2= k&0b00001111;
                              lcd_gotoxy(2,2);
                          printf(lcd_putc,"%d",N2);
                          state = 3;
                          break;
                     }
                  break;
                  case 3:
                     k=kbd_getc();
                     while (k!=0)
                     {
                          N3 = k&0b00001111;
                              lcd_gotoxy(3,2);
                          printf(lcd_putc,"%Ld",N3);
                              lcd_putc("\f");
                          N3 = N1*10+N2+N3/10;
                          if(N3>17)
                          {
                             lcd_gotoxy(1,2);
                             printf(lcd_putc,"Out of bounds");
                             delay_ms(1000);
                             lcd_gotoxy(1,2);
                             printf(lcd_putc,"               ");
                          }
                          else
                          {
                             N4=N3;
                             lcd_gotoxy(1,2);
                             printf (lcd_putc,"%3.2fV",N3);
                             Vout=N3;
                             Vref=Vout*0.1757;
                          }
                  N1=0;
                  N2=0;
                  N3=0;
                  state = 1;
                  break;
                    }

                }


            }
}
```

Εικόνα 3.4: Κώδικας Main *Κ*ύριας *Κ*εντρικής Συνάρτησης

Με την k=kbd_getc() παίρνει το πρόγραμμα το πλήκτρο που πατήθηκε και με τον συνδυασμό της while(k!=0) δεν δέχεται άλλο πάτημα πλήκτρου μέχρι να ολοκληρωθεί η πράξη λογικού AND ή N1=k&0b00001111 (ένα για κάθε κατάσταση), που μετατρέπει από ASCII σε integer αριθμό την k και θα δείξει το ψηφίο στην οθόνη με την εντολή printf(lcd_putc,"%d",N1).
Αυτό γίνεται τρεις φορές και στην τρίτη γίνεται ο έλεγχος με την if(N3>17) αν η N3 είναι η μεγαλύτερη του 17. Αν δεν είναι, η τιμή της N3 καταχωρείται στη Vref και αρχίζει η διαδικασία από την αρχή, αν είναι μεγαλύτερη γράφει στην οθόνη OUT OF BOUNDS εκτός ορίων για 1 δευτερόλεπτο και αρχίζει η διαδικασία από την αρχή.

3.4 Αρχικοποίηση Μικροελεγκτή και Ρύθμιση Εσωτερικών Διακοπών

Με τη συνάρτηση init το πρόγραμμα αρχικοποιεί τον μικροελεγκτή. Στην αρχή ρυθμίζει τη συχνότητα του εσωτερικού ταλαντωτή στα 8MHz που είναι η ταχύτητα του μικροελεγκτή. Μετά ρυθμίζει τα κανάλια και το εύρος της μετρούμενης τάσης του ADC και τη ταχύτητα μετατροπής. Τα κανάλια που επιλέξαμε είναι το AN0 και AN1 με εύρος από VSS μέχρι VDD δηλαδή από 0 έως 5V και ταχύτητα μετατροπής 1/8MHZ.

```
void init(void)
{
    setup_oscillator(OSC_8MHZ);

    setup_adc_ports( AN0_TO_AN1 | VSS_VDD  );
    setup_adc ( ADC_CLOCK_INTERNAL );

    setup_timer_2(T2_DIV_BY_1,30,1);

    setup_timer_0(T0_INTERNAL | T0_DIV_16);
    set_timer0(70);
}
```

Εικόνα 3.5: Κώδικας in*it*()

Στη συνέχεια, ρυθμίζει τον χρονιστή timer2 με τον οποίο επιλέγουμε την περίοδο του PWM. Επιλέγουμε κλιμάκωση εσωτερικής ταχύτητας (prescaler) T2_DIV_BY_1 δηλαδή διαιρούμε την ταχύτητα δια 1, μετά βάζουμε την τιμή της περιόδου στο πρόγραμμα 30 και τέλος επιλέγουμε κλιμάκωση στην έξοδο δηλαδή πόσο πολλαπλασιάζουμε το αποτέλεσμα (postscaler) (στην περίπτωση που χρησιμοποιείται σαν PWM δεν επηρεάζει καθόλου το πρόγραμμα άρα μπαίνει 1). Η συχνότητα του PWM θα αναφερθεί σε επομένη ενότητα.

Τέλος, ρυθμίζεται ο timer0 με τον οποίο θα γίνονται εσωτερικές διακοπές. Επιλέγουμε να λειτουργεί σαν χρονιστής με κλιμάκωση δια 16 και θέτουμε την αρχική τιμή του 70. Οι παραπάνω τιμές δεν μπήκαν τυχαία , μπήκαν αφότου έγιναν πειράματα, επίσης η εσωτερική ταχύτητα των χρονιστών είναι αυτή του συστήματος δια 4 Fosc/4.

Κάθε φορά που ο χρονιστής Timer1 διακόπτει, θέτει την περίοδο του PWM ίση με τη μεταβλητή Duty, που έχει αρχικοποιηθεί στην αρχή του προγράμματος, μετά καλεί την συνάρτηση pid και αποθηκεύει το αποτέλεσμα στη μεταβλητή p. Στη συνέχεια προσθέτει την p στην Duty και αποθηκεύει στην Duty. Αυτό το προγραμματίσαμε ώστε να βρούμε την Duty για την οποία η p είναι 0. Όταν το p είναι ίσο με το 0 τότε καταχωρεί την τιμή της Duty στην μεταβλητή DCON δηλαδή βρίσκει το duty cycle στο οποίο περιμένουμε την τάση εξόδου που θέλουμε. Τέλος, δείχνει στην οθόνη τις τιμές της Vadc και Iadc στην δεύτερη γραμμή, είναι τάση και ένταση της εξόδου αντίστοιχα.

```
#INT_TIMER0
void timer0_int(void)
    {
        set_pwm1_duty(Duty);

        p=PID(Kp,Ki,Kd,Vref);

        Duty=Duty+p;


        if(p==0)
        {
            DCON=Duty;
            set_pwm1_duty(DCON);
        }

        lcd_gotoxy(1,2);
        printf(lcd_putc,"%fV",5.7*Vadc*4.88e-3);

        lcd_gotoxy(8,2);
        printf(lcd_putc,"%1.4fA",Iadc*4.88e-3);

    }
```

Εικόνα 3.6: Κώδικας Χρονιστή Timer1

Για να βρούμε την περίοδο και τη συχνότητα του Timer1 κάνουμε τις παρακάτω πράξεις:

$$T_{Timer\,1} = (65536 - PR1) * Tprescaler = (65536 - 70) * \left(8\,MHz * \frac{1}{4} * \frac{1}{16}\right)^{-1} = 0.5\,s \quad (3.1)$$

ή

$$F_{Timer\,1} = \frac{1}{T_{Timer\,1}} = 1.9\,Hz \quad (3.2)$$

3.5 Προγραμματισμός PID Controller

Η συνάρτηση PID δέχεται τέσσερις μεταβλητές όταν καλείται: την Kp,Ki,Kd και Vref, που είναι οι μεταβλητές για PID και Vref που είναι τάση εξόδου που θέλουμε. Στη συνέχεια αρχικοποιεί τις μεταβλητές όπου θα αποθηκευτούν τα αποτελέσματα τον πράξεων. Ο ADC έχει 13 κανάλια μέτρησης αλλά έναν δίαυλο μετατροπής.

Για αυτό βάλαμε δυο if με τις οποίες γίνεται η επιλογή του καναλιού. Αν το adc_channel είναι 0 τότε μετράει την τάση της εξόδου, αν είναι 1 μετράει την ένταση. Τα delay μπήκαν ώστε ο μικροελεγκτής να προλαβαίνει να αλλάξει το κανάλι και να γίνει η μετατροπή.

Κατόπιν, ελέγχει αν η ένταση περνάει το 1A, αν το περνάει τότε το σφάλμα er θα παρθεί από την Vref-Iadc αλλιώς από την Vref-Vadc.

Ο ADC είναι 10-bit που σημαίνει η μετατροπή έχει ένα αποτέλεσμα μεταξύ 0 και 1024 (2^{10}=1024) . Για να μετατρέπει από 10-bit σε αναλογικό αριθμό (float) θα πολλαπλασιάσουμε το αποτέλεσμα της μετατροπής με το βήμα του μετατροπέα δηλαδή $4.88e^{-3}$ (5/1024=$4.88e^{-3}$). Το 5,7 που πολλαπλασιάζεται με το Iadc είναι για να φέρουμε το ρεύμα στην κλίμακα της τάσης.

Στη συνέχεια, υπολογίζει το σφάλμα er και τις μεταβλητές του PID. Το $3e^{-3}$ είναι ο χρόνος που χρειάζεται ο ADC να κάνει μια μετατροπή. Αφότου υπολογίσει τις μεταβλητές αποθηκεύει το σφάλμα er και την I για την επόμενη κλίση της συνάρτησης ώστε να γίνουν οι πράξεις της ολοκλήρωσης και της παραγώγισης.

Τέλος, επιστρέφει το αποτέλεσμα δια 8, αυτό προγραμματίστηκε γιατί τα αποτελέσματα ήταν πολύ μεγάλα και προκαλούσαν προβλήματα στο PWM.

```
signed int16 PID(float Kp,float Ki,float kd,float Vref)
{
    int16 vout;
    int16 er;
    float prev_er;
    float inter_er_prev;

    float proportional;
    float integral;
    float derivative;

    signed int16 pid;
    byte adc_channel=0;
    if(adc_channel==0)
        {
            set_adc_channel ( 0 );
            delay_us(50);
            Vadc=read_adc();
            delay_us(50);

            adc_channel=1;
        }
    if(adc_channel==1)
        {
            set_adc_channel ( 1 );
            delay_us(50);
            Iadc=read_adc();
            delay_us(50);

            adc_channel=0;
        }
    if(Iadc*4.88e-3>1)
        {
            vout=5.7*Iadc*4.88e-3;//5.7=Feedack resistor ratio
        }
    else
        {
            vout=Vadc;
        }

    er = Vref-vout;

    proportional=Kp*er;
    integral=inter_er_prev+er*3e-6;
    derivative=(er-prev_er)/3e-6;

    prev_er=er;
    inter_er_prev=integral;

    pid=proportional+integral*Ki+derivative*Kd;

    return(pid/8);

}
```

Εικόνα 3.7: Κώδικας συνάρτησης PID

Με την set_deadband ρυθμίζουμε τον χρόνο ανάμεσα στους συμπληρωματικούς PWM παλμούς. Ο SFR που χειρίζεται τον χρόνο είναι ο ECCP1DEL που είναι 8-bit. Το 7bit είναι αν θέλουμε το PWM να κλείνει αυτόματα όταν γίνει κάποιο γεγονός που ορίσαμε και 6-0 bit είναι ο χρόνος του delay. Η συνάρτηση βάζει στα 7 χαμηλότερα bit την τιμή του χρόνου καθυστέρησης χωρίς να πειράζει το 8ο. Δηλαδή, στην περίπτωση αυτή βάλαμε την value=4 έχουμε με την σειρά τις πράξεις του κώδικα:

$$value = 4 \wedge 7\, F_h = 100_b$$
$$temp = FB7_h$$
$$temp = FB7_h \wedge 80_h = 10000000_b$$
$$temp = 10000000_b \vee 100 = 10000100_b$$
$$ECCP1DEL = 10000100_b$$

Έτσι, βάζουμε την τιμή που θέλουμε στον ECCP1DEL. Στο εγχειρίδιο χρήσης του προγράμματος αναφέρεται η CCP_DELAY, όμως στους οδηγούς (drivers) δεν υπάρχει τέτοια ρύθμιση, για αυτό προγραμματίστηκε αυτή η συνάρτηση.

```
void set_deadband(int value)
{
    int temp;

    value &= 0X7F;
    temp = ECCP1DEL;
    temp &= 0X80;
    temp |= value;
    ECCP1DEL = temp;
}
```

Εικόνα 3.8: Κώδικας Συνάρτησης set_deadband

Τέλος, η συχνότητα του PWM θα βρεθεί από τους τύπους:

$$T_{pwm} = 4 * T_{OSC} * (PR2 + 1) * T2\,Prescaler = 4 * 1.25 * 10^{-7} * (30 + 1) * 1 = 15.5\,us \quad (3.3)$$

Με T_{osc} την συχνότητα λειτουργίας του μικροελεγκτή, PR2 η τιμή της που βάλαμε και T2Prescalaler η κλιμάκωση που βάλαμε.

Και από τον παρακάτω τύπο παίρνουμε την καθυστέρηση (delay) μεταξύ των δυο PWM παλμών:

$$Delay = 4 * T_{OSC} * ECCP\,1\,DEL = 4 * 1.25 * 10^{-7} * 132_d = 66\,us \quad (3.4)$$

Από την (3.3) και την (3.4) έχουμε:

$$T_{PWM} = -Delay - T_{pwm} = 66 - 15.5 = 51.1\,us \quad (3.5)$$

$$\text{ή} \quad F_{PWM} = \frac{1}{T_{PWM}} = 19.5\,KHz \quad (3.6)$$

3.6 Προγραμματισμός Σύνδεσης Η/Υ με τον Μικροελεγκτή

Για την σύνδεση μεταξύ μικροελεγκτή και Η/Υ χρησιμοποιήθηκε η μονάδα του UART και το πρωτόκολλο σειριακής επικοινωνίας RS232.

```
#USE rs232(ERRORS,baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,stop=1,bits=8)
```

Εικόνα 3.9: Δήλωση λειτουργίας σειριακής θύρας

Με την παραπάνω εντολή δηλώνουμε την ταχύτητα της σύνδεσης 9600 baud, χωρίς parity είναι η προστασία των πλαισίων (frame), xmit είναι η ακίδα μετάδοσης πλαισίων, rcv η ακίδα λαμβανομένων πλαισίων, bits ο αριθμός των bits δεδομένων που θα χρησιμοποιηθούν στην επικοινωνία και stop ο αριθμός των bits για να τελειώσει η μεταφορά ή η λήψη.

Πριν αναλύσουμε τον προγραμματισμό της συνάρτησης επικοινωνίας μεταξύ μικροελεγκτή και Η/Υ serial_interrupt, θα εξηγηθεί η compstr που χρησιμοποιείται εκτεταμένα μέσα στην συνάρτηση.

Η compstr δέχεται δυο πίνακες χαρακτήρων τον a και b, ο a είναι οι χαρακτήρες που θέλουμε να συγκρίνουμε με αυτούς του b. Στη συνέχεια αρχικοποιεί τις μεταβλητές που θα χρησιμοποιηθούν για την σύγκριση. Μετά με την εντολή strlen παίρνει το μέγεθος του b.

Για να βρει αν οι δυο πίνακες είναι ίδιοι θα τρέξει μια for όσες φορές είναι το length, μέσα στον βρόγχο θα γίνει ο έλεγχος ανά θέση του πίνακα. Αν κάποια θέση του πίνακα a είναι ίση με την αντίστοιχη του b τότε αυξάνουμε τον μετρητή counter κατά 1.

```
int compstr(char a[],char b[])
{
    int i;
    int counter=0;
    int length=0;
    length = strlen(b);

    for(i=0;i<length;i++)
    {
        if(a[i]==b[i])
        {
            counter++;
        }
    }

    if(counter==length)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

Εικόνα 3.10: *Κώδικας* Συνάρτησης compstr

Αφότου τελειώσει η for γίνεται ένας τελευταίος έλεγχος, αν ο counter είναι ίσος με το length τότε επιστρέφει 1 που σημαίνει ότι είναι ίσα, αν δεν είναι τότε επιστρέφει 0.

Η συνάρτηση serial_interrupt καλείται κάθε φορά που ενεργοποιείται η διακοπή της int_rda δηλαδή όταν είναι να σταλεί κάποιο πακέτο εντολή από τον H/Y και τελειώνει με την αποστολή απάντησης από τον μικροελεγκτή.

Σε αυτό το σημείο θα εξηγηθεί η δομή των εντολών, ο μικροελεγκτής έχει προγραμματιστεί να αναγνωρίζει 6 εντολές command με την σειρά:

xVOUT θέτει την τάση εξόδου ίση με το x, VINF στέλνει την τιμή της τάσης εξόδου, ON ανοίγει την έξοδο, OFF κλείνει την έξοδο, INFO στέλνει πληροφορίες για το τροφοδοτικό και IINF στέλνει την τιμή του ρεύματος εξόδου. Δεν ακολουθήθηκε κάποιο στάνταρντ στη σύνταξη των εντολών όπως το SCPI.

Όταν κληθεί, αρχικά σταματάει τις διακοπές τις int_rda ώστε αν σταλεί κάποιο πακέτο κατά την διάρκεια ενός άλλου να μην δημιουργηθεί πρόβλημα μεταξύ τους. Αυτός είναι και ο λόγος που στην main ενεργοποιείται συνέχεια η int_rda. Κατόπιν με την εντολή gets παίρνει το πακέτο από τον H/Y και το αποθηκεύει στον πινάκα temp, στη συνέχεια με τη εντολή strtol χωρίζει τον temp σε νούμερα result αν υπάρχουν και χαρακτήρες command.

```
#int_rda
void serial_interrupt()
{
    disable_interrupts(int_rda);
    gets(temp);
    result=strtol(temp,&command,10);


    if(compstr(command,command1))
    {
        if(result>0&&result<=18)
        {
            printf("VOUT OK");
            Vout=result;
            Vref=Vout*0.1757;
            putc('\n');
            putc('\r');
        }
        else
        {
            printf("VOUT NOT OK");
            putc('\n');
            putc('\r');
        }

    }

    if(compstr(command,command2))
    {
        set_adc_channel ( 0 );
        delay_us(50);
        printf("%f",(read_adc()*4.88759e-3)/(0.1757));
        delay_us(50);
        putc('\n');
        putc('\r');

    }
    if(compstr(command,command3))
    {
        printf("OUT ON");
        set_pwm1_duty(DCON);
        putc('\n');
        putc('\r');
    }
```

Εικόνα 3.11: Κώδικας serial_interrupt Μέρος 1ο

Στη συνέχεια συγκρίνει με κάθε εντολή command που αναφερθήκαν παραπάνω, αν κάποια if έχει αποτέλεσμα 1 με την εντολή που στείλαμε από τον Η/Υ τότε τρέχει τις εντολές μέσα στη if, αν δεν ταιριάζει με καμιά τρέχει η τελευταία if που στέλνει στον Η/Υ EMPTY COMMAND OR WRONG COMMAND άδεια εντολή ή λάθος εντολή.

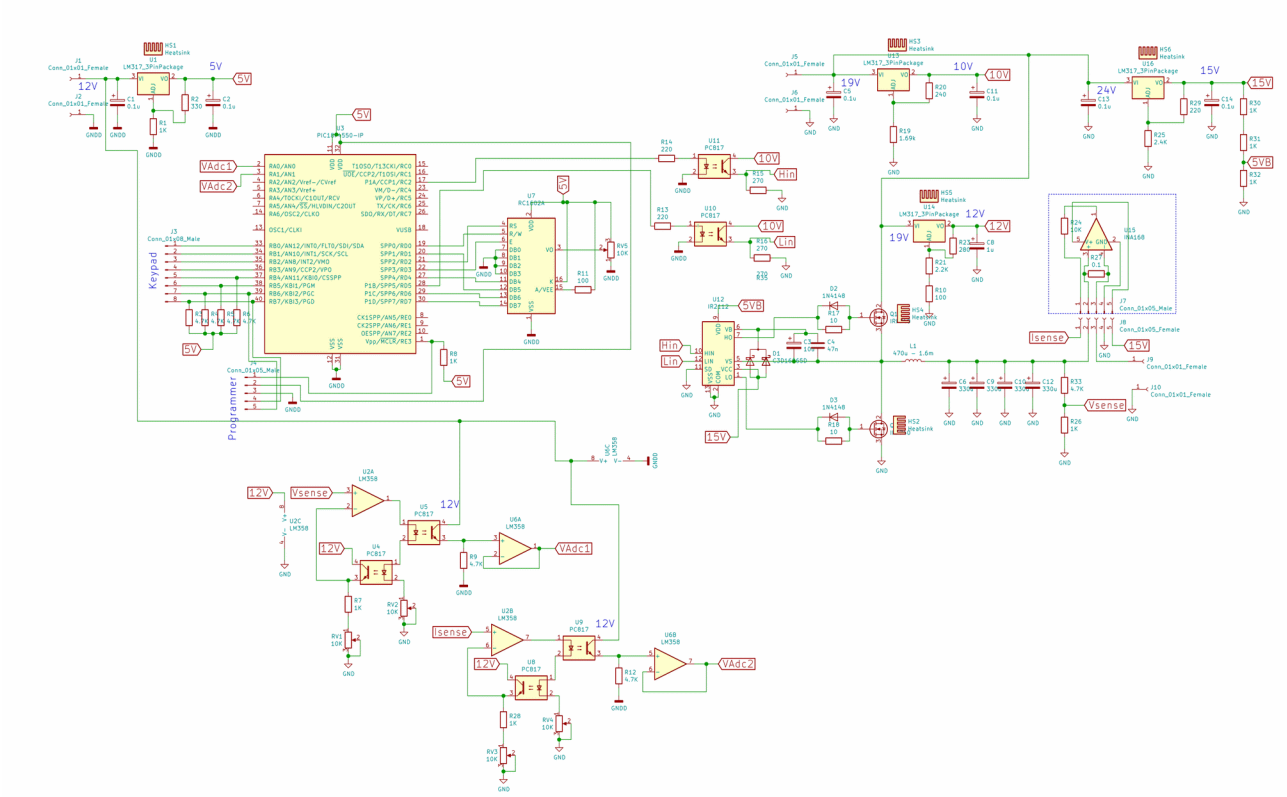
```
    if(compstr(command,command4))
    {
        printf("OUT OFF");
        set_pwm1_duty(0);
        putc('\n');
        putc('\r');
    }
    if(compstr(command,command5))
    {

        printf("Programmable Power Supply Ver:1.00");
        putc('\n');
        putc('\r');

    }

    if(compstr(command,command6))
    {
        set_adc_channel ( 1 );
        delay_us(50);
        printf("%f",(read_adc()*4.88759e-3));
        delay_us(50);
        putc('\n');
        putc('\r');

    }
    if(!compstr(command,command1)&&!compstr(command,command2)
        &&!compstr(command,command3)&&!compstr(command,command4)
            &&!compstr(command,command5)&&!compstr(command,command6))
    {
        printf("EMPTY COMMAND OR WRONG COMMAND");
        putc('\n');
        putc('\r');
    }
}
```

Εικόνα 3.12: Κώδικας serial_interrupt Μέρος 2ο

Κεφάλαιο 4 Πειραματικές Διατάξεις Διακοπτικού Τροφοδοτικού

Σ' αυτό το κεφάλαιο θα αναλυθούν οι πειραματικές διατάξεις που αποτελούν τον Buck Converter, τα αποτελέσματα και τα συμπεράσματα από το κύκλωμα.

4.1 Κυκλώματα τροφοδοτικού και περιφερειακών διατάξεων



Εικόνα 4.1: Κύκλωμα Διακοπτικού Τροφοδοτικού Buck Converter

4.1.1 Κύκλωμα Μικροελεγκτή

Το κύκλωμα του μικροελεγκτή αποτελείται:

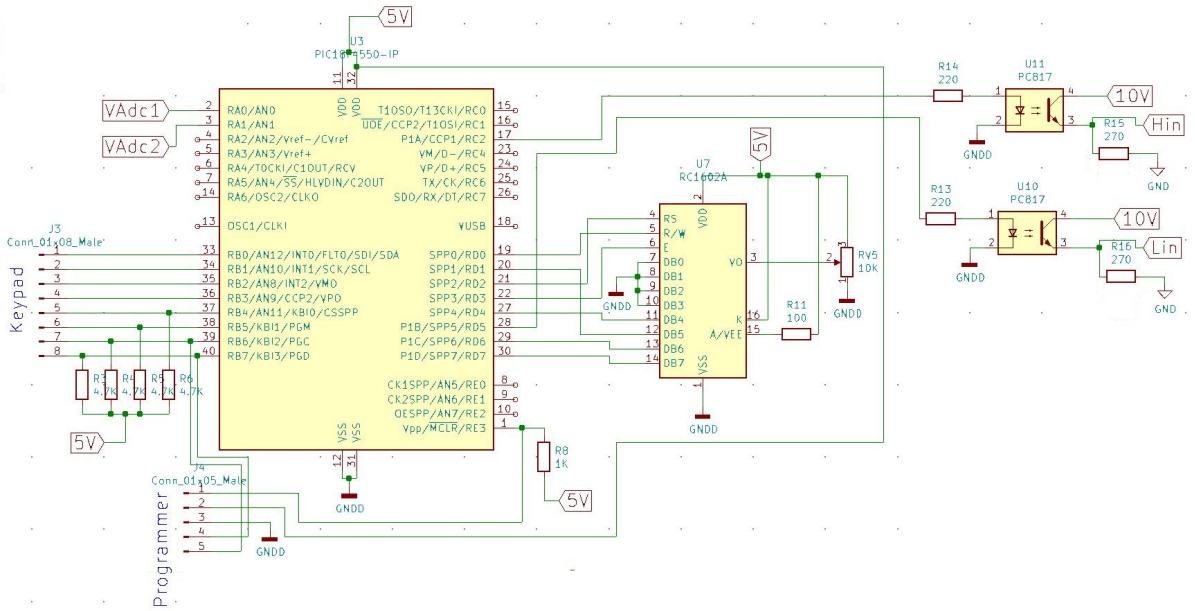
1) Από τον μικροελεγκτή PIC18F4550 που επιλέχθηκε εξαιτίας της εξοικείωσης που έχουμε καθώς επίσης και των δυνατοτήτων του.

2) Από την LCD οθόνη που είναι η BC1602A-YPLEH μεγέθους 2 σειρών από 16 χαρακτήρες (οποιαδήποτε οθόνη LCD 2x16 μπορεί να χρησιμοποιηθεί). Η οθόνη είναι συνδεμένη στη θύρα PORTD με την σύνδεση:
RS - RD2, R/W - RD0, E - RD3, DB4 - RD4, DB5 - RD5, DB6 - RD6, DB7 – RD7.

3) Το πληκτρολόγιο είναι το KEY16-GM-472 αποτελείται από 16 πλήκτρα, 4 σειρές και 4 στήλες (οποιοδήποτε πληκτρολόγιο 4χ4 μπορεί να χρησιμοποιηθεί). Συνδέεται στη θύρα PORTB με κάθε ακίδα του πληκτρολογίου σε αντίστοιχη ακίδα της θύρας: 1η – RB0, 2η – RB1, 3η – RB2, 4η – RB3, 5η – RB4, 6η – RB5, 7η – RB6, 8η – RB7.

4) Η κανονική κατάσταση των ακροδεκτών είναι η υψηλή (high), άρα πρέπει να κατέβουν στην χαμηλή (low) ώστε να πάρουμε το πάτημα του πλήκτρου, για αυτό συνδέθηκαν στις ακίδες RB4 - RB7 με pull-up αντίσταση.

5) Ο προγραμματιστής PICkit3 συνδέεται στις ακίδες RB6 και RB7 που δεν προκαλούν κανένα πρόβλημα στη χρήση που πληκτρολογίου.



Εικόνα 4.2: Κύκλωμα Μικροελεγκτή

Οι photocoupler έχουν μπει στο κύκλωμα για να έχει γαλβανική απομόνωση το κύκλωμα ισχύος από το κύκλωμα έλεγχου, να μην περνάνε παράσιτα από το ένα στο άλλο. Οι photocoupler είναι της σειράς PC817 επιλεχτήκαν για τα χαρακτηριστικά τους, για το εύρος των εφαρμογών που μπορεί να χρησιμοποιηθούν καθώς επίσης και την ευκολία χρήσης τους.
Οι τιμές των αντιστάσεων R14 και R15 βρεθήκαν από τον νόμο του Ohm και εκπληρώνουν τη σχέση:

$$I_{R14,15} \leqslant I_{outMicro} = I_{R14,15} \leqslant 25\,mA \qquad (4.1)$$

$$\text{Άρα έχουμε} \quad R_{14,15} = \frac{V_{pwm} - V_f}{I_{outMicro}} = \frac{5 - 1.2}{25\,mA} = 152\,\Omega \qquad (4.2)$$

Με V_f η τάση που χρειάζεται η φωτοδίοδος να άγει από το datasheet του PC817 ,$I_{outMicro}$ το μέγιστο ρεύμα εξόδου του μικροελεγκτή και V_{pwm} η μέγιστη τάση του PWM.
Όμως, δεν θέλουμε να πιέσουμε την έξοδο του μικροελεγκτή στις μέγιστες τιμές της, γι' αυτό μετά από πείραμα βάλαμε:

$$R_{14,15} = 220\,\Omega \qquad (4.3)$$

$$\text{Από την (4.1) και (4.2) έχουμε:} \quad I_{R14,15} = \frac{V_{pwm} - V_f}{R_{14,15}} = 17\,mA \qquad (4.4)$$

Που είναι το αποτέλεσμα που θέλαμε.
Από το datasheet του PC817 έχουμε ότι το:

$$CTR \simeq 130\,\% \quad \text{για} \quad I_F = 17\,mA \qquad (4.5) \text{ και } (4.6)$$

Το CTR είναι ο λόγος του ρεύματος συλλέκτη προς το ρεύμα της φωτοδιόδου και εκφράζει την ενίσχυση του ρεύματος ενός τρανζίστορ, επίσης από τον τύπο:

$$CTR=\frac{I_C}{I_F}*100\,\%\Rightarrow I_C=I_F*CTR\Rightarrow I_C=17\,mA*1.3=22.1\,mA \tag{4.7}$$

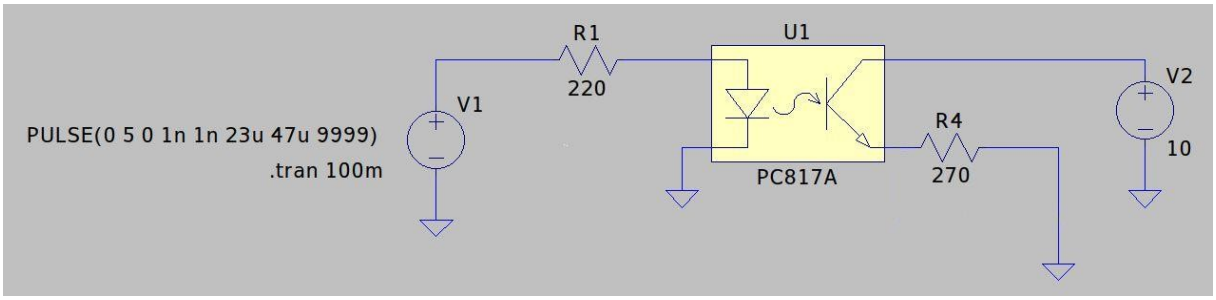
Η ελάχιστη τάση που μπορεί να αναγνωρίσει ο οδηγός του MOSFET είναι τα 5V, άρα το τρανζίστορ του optocoupler πρέπει να έχει στην έξοδο του τουλάχιστον 5V, εμείς βάλαμε 6V για να έχουμε ένα παράθυρο σφάλματος, άρα έχουμε:

$$R_C=\frac{V_C}{I_C}\Rightarrow R_C=\frac{6}{22.1\,mA}\simeq 270\,\Omega \tag{4.8}$$

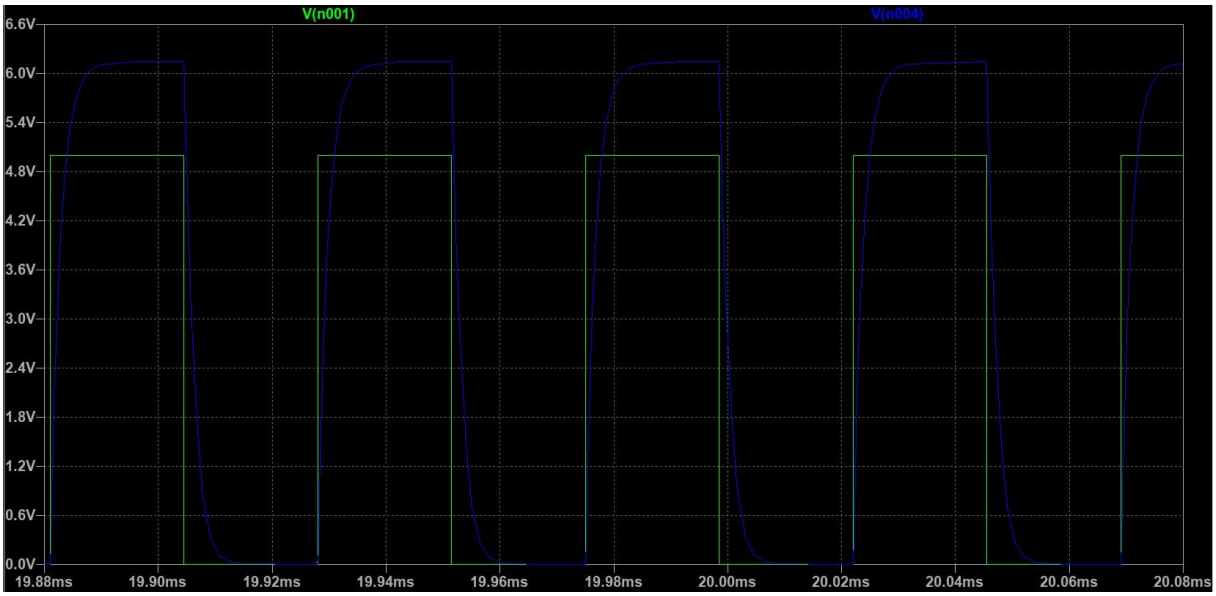
Με τάση εισόδου μεγαλύτερη των 6V, το κύκλωμα των optocoupler έχει στην έξοδο του 6V. Να σημειωθεί εδώ ότι οι τάσεις 5V που τροφοδοτεί τον μικροελεγκτή και η 10V που τροφοδοτεί τους optocoupler τροφοδοτούνται από διαφορετικά τροφοδοτικά. Τα τροφοδοτικά θα εξηγηθούν σε επομένη ενότητα.

Οι ακίδες RA0 και RA1 χρησιμοποιούνται σαν ADC που είναι συνδεδεμένες στο κύκλωμα ανάδρασης, το οποίο θα εξηγηθεί σε επομένη ενότητα.

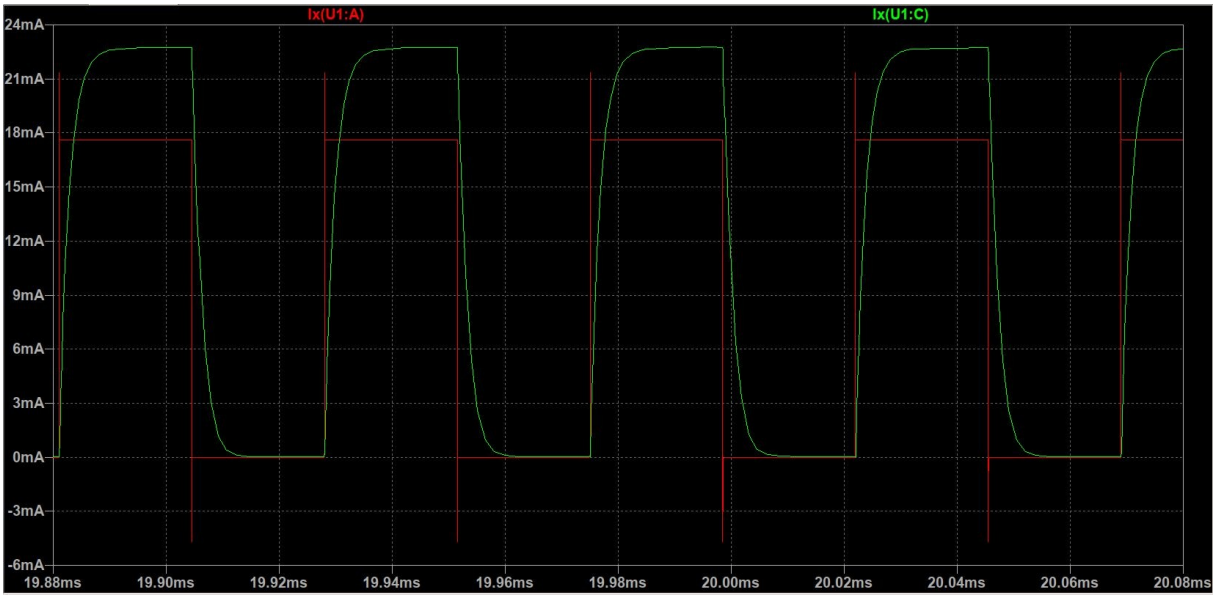
Χρησιμοποιώντας το πρόγραμμα προσομοίωσης Ltspice XVII σχεδιάσαμε και προσομοιώσαμε το κύκλωμα του optocoupler.



Εικόνα 4.3: Κύκλωμα optocoupler στο LTspice με V1 ο Παλμός PWM

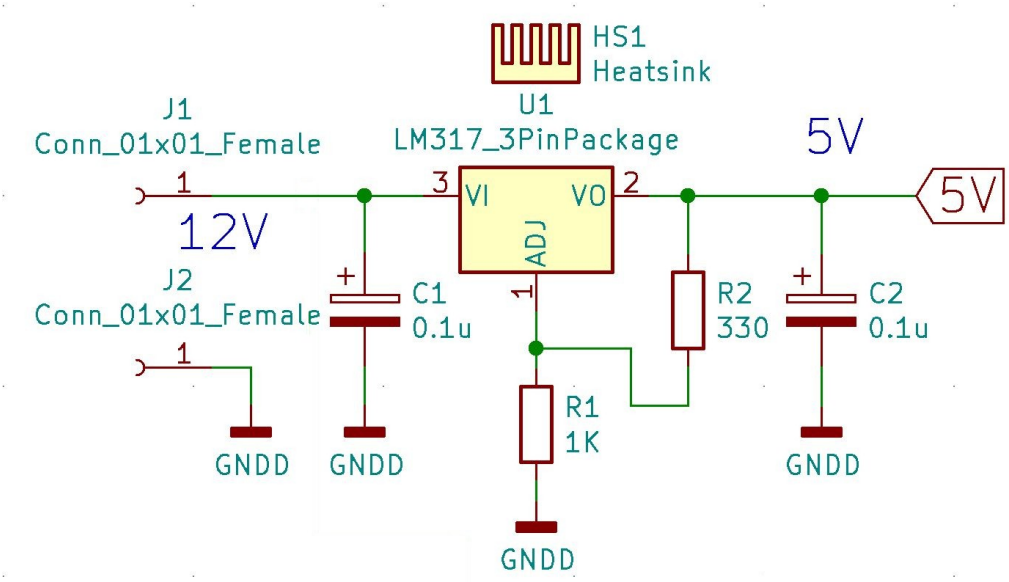


Εικόνα 4.4: Παλμοί Τάσης Εισόδου V(n001) και Τάσης Εξόδου V(n004) Optocoupler



Εικόνα 4.5: Παλμοί Έντασης Εισόδου V(n001) και Έντασης Εξόδου V(n004) Optocoupler

Από τις εικόνες 4.4 και 4.5 παρατηρούμε ότι τα θεωρητικά νούμερα που βρήκαμε είναι πολύ κοντά στα αποτελέσματα της προσομοίωσης, εκτός από την καθυστέρηση στην έξοδο του optocoupler, το οποίο δεν είναι παράξενο, εξαρτάται από την αντίσταση εξόδου.



Εικόνα 4.6: Τροφοδοτικό Μικροελεγκτή

Ο μικροελεγκτής τροφοδοτείται με 5V τα οποία του τα παρέχει ένας γραμμικός ρυθμιστής τάσης ο LM317. Ο ίδιος τροφοδοτείται με 12V. Από το datashet του LM317 έχουμε:

$$V_{out} = V_{ref}\left(1+\frac{R1}{R2}\right) \tag{4.9}$$

Για V_{out} =5V, V_{ref} =1.25V είναι η εσωτερική τάση αναφοράς και θέτοντας αυθαίρετα R2=220Ω (συνήθως η R2 τίθεται 240Ω). Από την (4.9) έχουμε και λύνοντας ως προς R1:

$$R1 = R2 * (\frac{V_{out}}{V_{ref}} - 1) = 330 * (\frac{5}{1.25} - 1) \simeq 1\,k\Omega \quad (4.10)$$

Ο LM317 θέλουμε να μας παρέχει τουλάχιστον 500mA στην έξοδο του, όμως η ισχύς που θα χρειάζεται να κατεβάσει την τάση και την ένταση του ρεύματος μετατρέπουν την ισχύ σε ζέστη που αν ξεπεράσει κάποια θερμοκρασία μπορεί να κάψει τον LM317.
Άρα από τον νόμο του Ohm και του Watt έχουμε:

$$P = V * I = (V_{IN} - V_{out}) * I_{outmax} = (12 - 5) * 500\,mA = 3.5\,W \quad (4.11)$$

Το οποίο είναι παραπάνω από 1W που θα μπορούσαμε να αφήσουμε χωρίς ψήκτρα (heatsink). Λύνοντας τον παρακάτω τύπο ως προς το $\Theta_{casetoAmbient}$ θα βρούμε το μέγεθος της ψήκτρας:

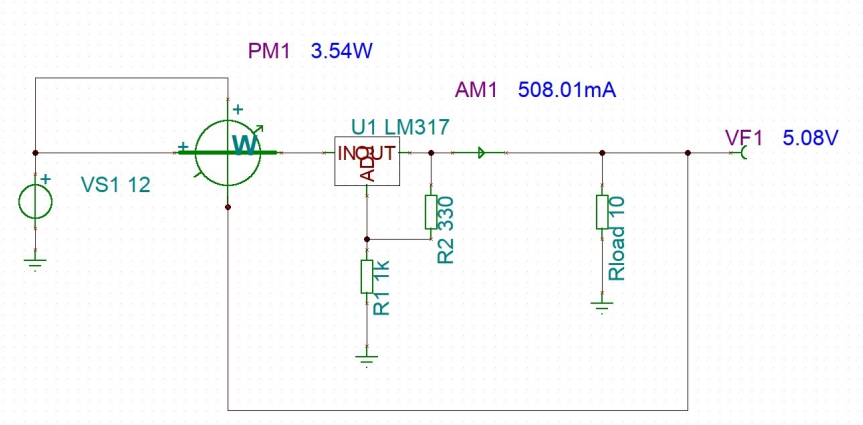
$$T_{junctionMax} - T_{Ambient} = (\Theta_{JunctiontoCase} + \Theta_{casetoAmbient}) * P \Rightarrow$$

$$\Theta_{casetoAmbient} = \frac{T_{junctionMax} - T_{Ambient}}{P} - \Theta_{JunctiontoCase} = \frac{125 - 30}{3.5} - 5 = 22{,}2\,\frac{C^o}{W} \quad (4.12)$$

Με $T_{junctionMax}$ η μέγιστη θερμοκρασία που μπορεί να λειτουργήσει ο LM317, $T_{ambient}$ η θερμοκρασία του περιβάλλοντος που θα λειτουργεί, $\Theta_{juctiontoCase}$ η θερμική αντίσταση σημείο του τύπου πακέτου του ολοκληρωμένου και $\Theta_{casetoAmbient}$ η θερμική αντίσταση σημείο περιβάλλοντος. Τα οποία βρέθηκαν από το datasheet του LM317.

Από την (4.12) συμπεραίνουμε ότι θα χρειαστεί να βάλουμε μια ψήκτρα με θερμική αντίσταση τουλάχιστον 22.2C^o/W αφήνοντας ένα παράθυρο σφάλματος 20% θα βάλουμε ψήκτρα με θερμική αντίσταση 17.76 C^o/W τουλάχιστον.
Στην διάθεση μας έχουμε ψήκτρες παρόμοιες με την EA-T220-38E, έχουν θερμική αντίσταση 10.4 C^O/W που μας καλύπτει.

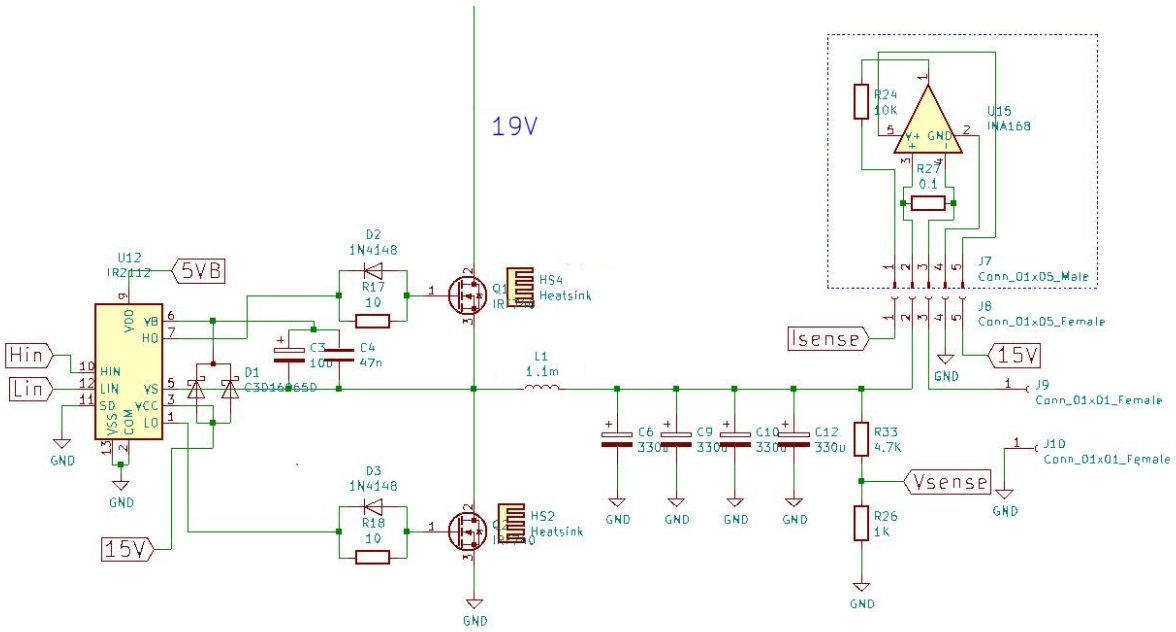
Σχεδιάσαμε το κύκλωμα του τροφοδοτικού στο πρόγραμμα προσομοίωσης TI TINA για να ελέγξουμε αν τα θεωρητικά νούμερα που βρήκαμε συμπίπτουν με αυτά της προσομοίωσης.



Εικόνα 4.7: Κύκλωμα Τροφοδοτικού Μικροελεγκτή Στο Πρόγραμμα TI TINA

4.1.2 Κύκλωμα Ισχύος Buck Converter και Driver/Οδηγού MOSFET

Ο οδηγός του MOSFET που επιλέχθηκε είναι ο IR2112, είναι από τους πιο διαδεδομένους οδηγούς και υπάρχουν πλήθος εφαρμογών που τον καθιστά εύκολο στην χρήση. Τα MOSFET στο κύκλωμα λειτουργούν σαν διακόπτες, για να μπορέσουν να άγουν φορτίζονται στιγμιαία από ένα μεγάλο ρεύμα για να φορτίσει τον παρασιτικό πυκνωτή εισόδου (Gate), το οποίο δεν μπορεί να παρέχει η θύρα του μικροελεγκτή ή έξοδος του optocoupler. Επίσης οι θύρες του μικροελεγκτή μπορούν να παρέχουν μέχρι 5V, σε άλλες περιπτώσεις θα ήταν αρκετό αλλά για να άγει πλήρως χρειάζεται παραπάνω τάση. Καθώς επίσης υπάρχουν αναστρέφοντα ρεύματα που δεν μπορεί να διαχειριστεί ο μικροελεγκτής. Αυτό καθιστά τον οδηγό αναγκαίο στο κύκλωμα.



Εικόνα 4.8: Κύκλωμα Ισχύος Synchronous Buck Converter και Οδηγού MOSFET

Για να μπορεί να παρέχει ο οδηγός τάση μεγαλύτερη αυτής της τροφοδοσίας VCC=15V στο MOSFET Q1 έχει συνδεθεί ένα κύκλωμα bootstaping (εκκίνησης) αποτελούμενο από την δίοδο C3D16056D και τους πυκνωτές C3, C4 που είναι συνδεμένοι παράλληλα. Οι πυκνωτές φορτίζουν καθόλη τη διάρκεια του κύκλου λειτουργίας και ξεφορτίζουν όταν είναι ON το Q1.
Οι παλμοί έλεγχου συνδέονται στον οδηγό στις ακίδες HIN και LIN, το HIN είναι οι παλμοί High side (υψηλή μεριά) ελέγχουν το Q1 και το LIN είναι Low side (χαμηλή μεριά) ελέγχουν το Q2. Τα 5V που πηγαίνουν VDD ορίζουν την τάση ON του οδηγού.

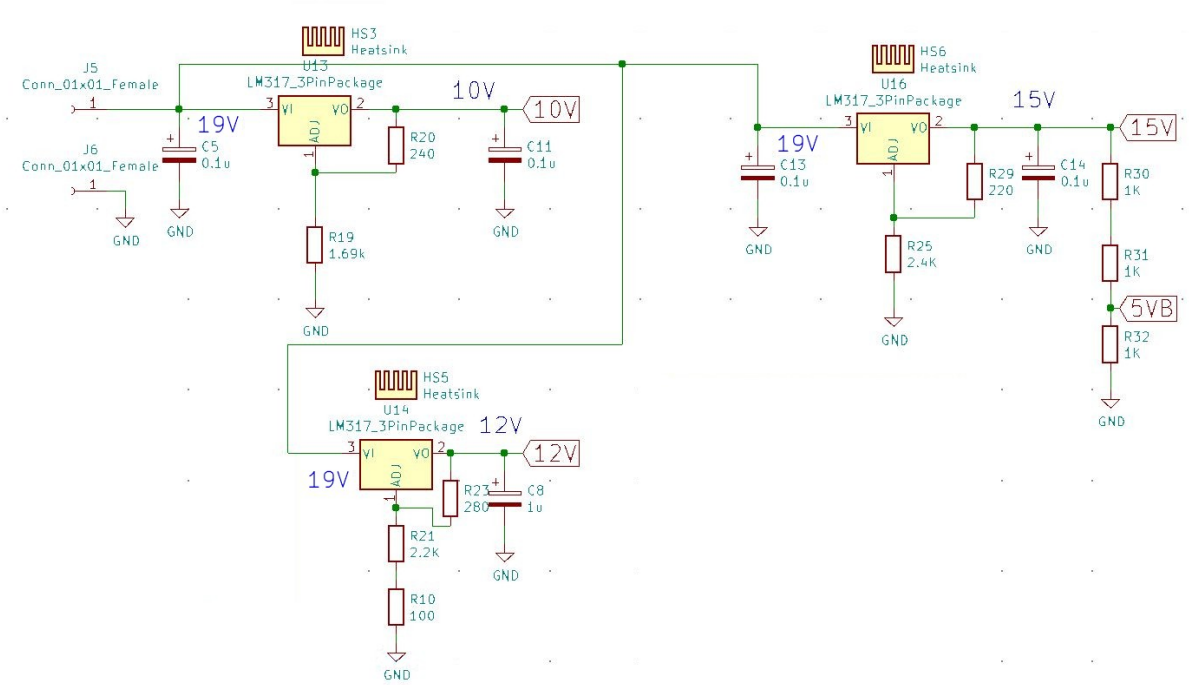
Τα MOSFET του κυκλώματος είναι τα IRF740 επιλεχτήκαν γιατί είναι MOSFET ισχύος και αντέχουν την τάση και ένταση του κυκλώματος καθώς και ότι μπορούν να λειτουργήσουν χωρίς πρόβλημα στην συχνότητα λειτουργίας των 21Khz επίσης, είναι καναλιού N που είναι ταχύτερα από τα P. Οι δίοδοι στις εισόδους των MOSFET είναι για την προστασία από ανάστροφα ρεύματα και οι αντιστάσεις των 10Ω είναι για να μην τραβήξουν απότομα μεγάλα ρεύματα και κάψουν τον οδηγό.

Οι ψήκτρες υπολογίζονται από τις (4.11) και (4.12) έχουμε:

$$P=(V_{IN}-V_{outmax})*I_{outmax}=(19-17)*1=2\,W \tag{4.13}$$

$$\Theta_{casetoAmbient}=\frac{T_{junctionMax}-T_{Ambient}}{P}-\Theta_{JunctiontoCase}=\frac{150-30}{2}-1=59\,\frac{C^o}{W} \tag{4.14}$$

Έχοντας στην διάθεση μας ψήκτρες παρόμοιες με την HSE-B2111-038 που έχουν θερμική αντίσταση 20 C^O/W, μας υπερκαλύπτει ακόμα και το 20% που είναι 47.2C^O/W.
Για να προσεγγιστεί η μεγάλη τιμή του πηνίου συνδέθηκαν σε σειρά 3 πηνία των 330uH με ολική αυτεπαγωγή 990mH, τύπου παρομοίου με 2318-V-RC.
Στο κύκλωμα βάλαμε 4 πυκνωτές χωρητικότητας 330uF παράλληλα με συνολική χωρητικότητα 1320mF για να έχουμε μεγαλύτερη ευστάθεια εξόδου τύπου REA331M1HBK-1320P.
Τα κυκλώματα της ανάδρασης τάσης και έντασης θα εξηγηθούν σε επομένη ενότητα.



Εικόνα 4.9: Κύκλωμα τροφοδοτικών του Κυκλώματος Ισχύος

Τα τροφοδοτικά του κυκλώματος ισχύος ακολούθησαν τον σχεδιασμό του τροφοδοτικού του μικροελεγκτή, θα αναλυθούν από αριστερά προς δεξιά. Από τις (4.10), (4.11) και (4.12) έχουμε για U13:

$$R19 = R20 * \left(\frac{V_{outU13}}{V_{ref}} - 1\right) = 240 * \left(\frac{10}{1.25} - 1\right) \simeq 1.69\,k\Omega \qquad (4.15)$$

$$P_{U13} = \left(V_{IN} - V_{outU13}\right) * \left(I_C * 4\right) = \left(19 - 10\right) * 88.8\,mA = 0.8\,W \qquad (4.16)$$

Το I_C πολλαπλασιάστηκε ώστε να έχουμε να παράθυρο σφάλματος, γιατί τροφοδοτεί τους optocoupler.

$$\Theta_{casetoAmbient} = \frac{T_{junctionMax} - T_{Ambient}}{P_{U13}} - \Theta_{JunctiontoCase} = \frac{125 - 30}{0.8} - 5 = 113.3\,\frac{C^o}{W} \qquad (4.17)$$

Για U16:

$$R25 = R9 * \left(\frac{V_{outU16}}{V_{ref}} - 1\right) = 220 * \left(\frac{15}{1.25} - 1\right) \simeq 2.4\,k\Omega \qquad (4.18)$$

$$Vout2U16 = \frac{R32}{R30 + R31 + R32} * V_{outU16} = \frac{1k}{1k + 1k1 + 1k} * 15 = 5\,V \qquad (4.19)$$

$$P_{U16} = \left(V_{IN} - V_{outU16}\right) * I_{outU16} = \left(19 - 15\right) * 100\,mA = 0.4\,W \qquad (4.20)$$

$$\Theta_{casetoAmbient}=\frac{T_{junctionMax}-T_{Ambient}}{P_{U16}}-\Theta_{JunctiontoCase}=\frac{125-30}{0.4}-5=232.5\frac{C^o}{W} \quad (4.21)$$

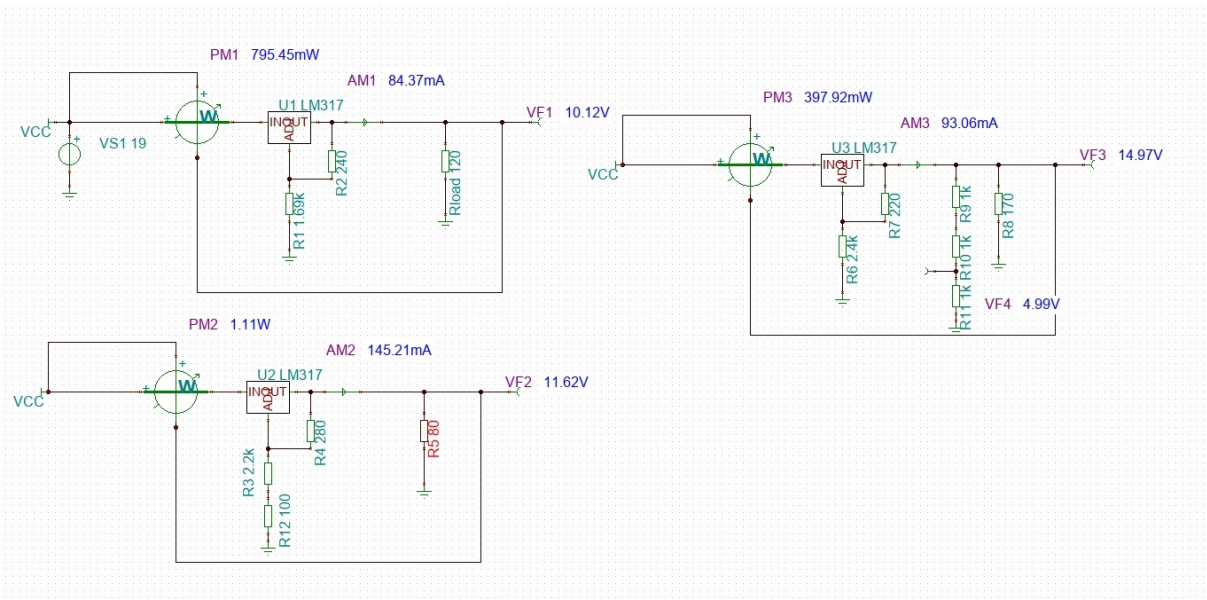
Για U14:

$$RR21,10=R23*(\frac{V_{outU14}}{V_{ref}}-1)=280*(\frac{12}{1.25}-1)\simeq 2.3\,k\Omega\,και\,100\,\Omega \quad (4.22)$$

$$P_{U14}=(V_{IN}-V_{outU14})*I_{outU16}=(19-12)*150\,mA=1.05\,W \quad (4.23)$$

$$\Theta_{casetoAmbient}=\frac{T_{junctionMax}-T_{Ambient}}{P_{U14}}-\Theta_{JunctiontoCase}=\frac{125-30}{1.05}-5=85.5\frac{C^o}{W} \quad (4.25)$$

Από τις (4.16), (4.17), (4.20), (4.21), (4.23) και (4.24) παρατηρούμε ότι δεν ήταν αναγκαίες οι ψύκτρες, αλλά μπήκαν παρόμοιες με HSE-B2111-038 .
Σχεδιάσαμε τα κυκλώματα που τροφοδοτούν το κύκλωμα ισχύος στο πρόγραμμα προσομοίωσης TI TINA.



Εικόνα 4.10: Κύκλωμα Τροφοδοτικού του Κυκλώματος Ισχύος Πρόγραμμα TI TINA

Η προσομοίωση μας δίνει αποτελέσματα τα οποία συμπίπτουν με αυτά των θεωρητικών. Εκτός από τον τελευταίο LM317, που δεν δημιουργεί κάποιο πρόβλημα στο υποκύκλωμα που τροφοδοτεί.

4.1.3 Κύκλωμα Ανάδρασης Τάσης και Έντασης

Το κύκλωμα ανάδρασης αποτελείται από τρία υποκυκλώματα: τον διαιρέτη τάσης που υποβαθμίζει την τάση ώστε να μπορεί να την δέχεται ο optocoupler, το INA169 (στο σχέδιο του κυκλώματος είναι 168, είναι παρόμοια) που είναι ολοκληρωμένο, μετατρέπει την ένταση σε τάση και ο γραμμικός φωτοζεύκτης σήματος.

Ο διαιρέτης τάσης υπονομεύει την τάση, από τον τύπο του Kirchoff για την τάση έχουμε:

$$V_{sense} = \frac{R26}{R26 + R33} * V_{outBuck} \quad (4.26)$$

Άρα για $V_{outBuck}$=17V η μέγιστη έξοδος:

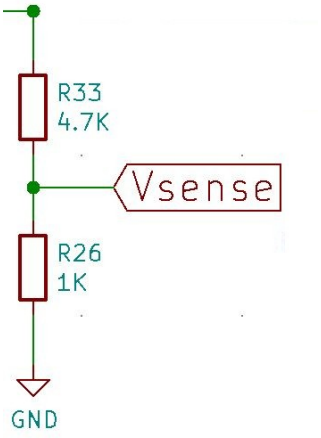
$$V_{sense} = \frac{1k}{1k + 4.7k} * 17 = 3V \quad (4.27)$$

Το υποκύκλωμα της ανάδρασης μετατρέπει την ένταση σε τάση ακολουθώντας τον τύπο:

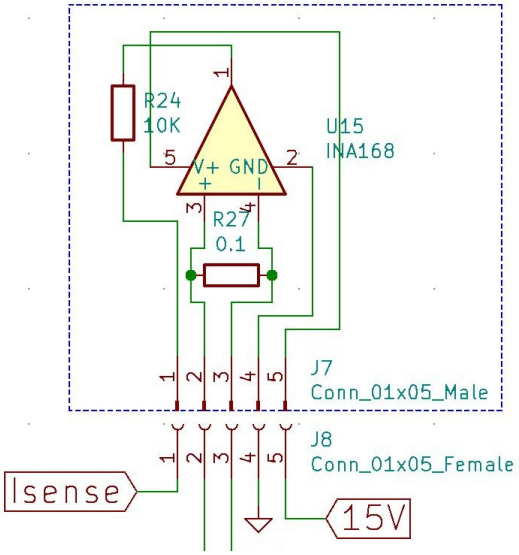
$$Isense = I_{outBuck} * R27 * 1000 \frac{uA}{V} * R24 \quad (4.28)$$

Θέτοντας το $I_{outBuck}$=1A τη μέγιστη ένταση του κυκλώματος έχουμε:

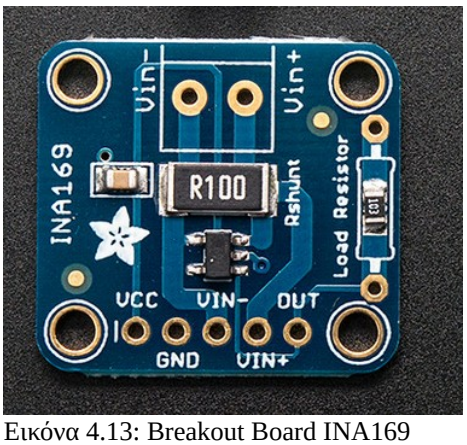
$$Isense = 1 * 0.1 * 1000 \frac{uA}{V} * 10k = 1V \quad (4.29)$$



Εικόνα 4.11: Κύκλωμα Διαιρέτη Τάσης



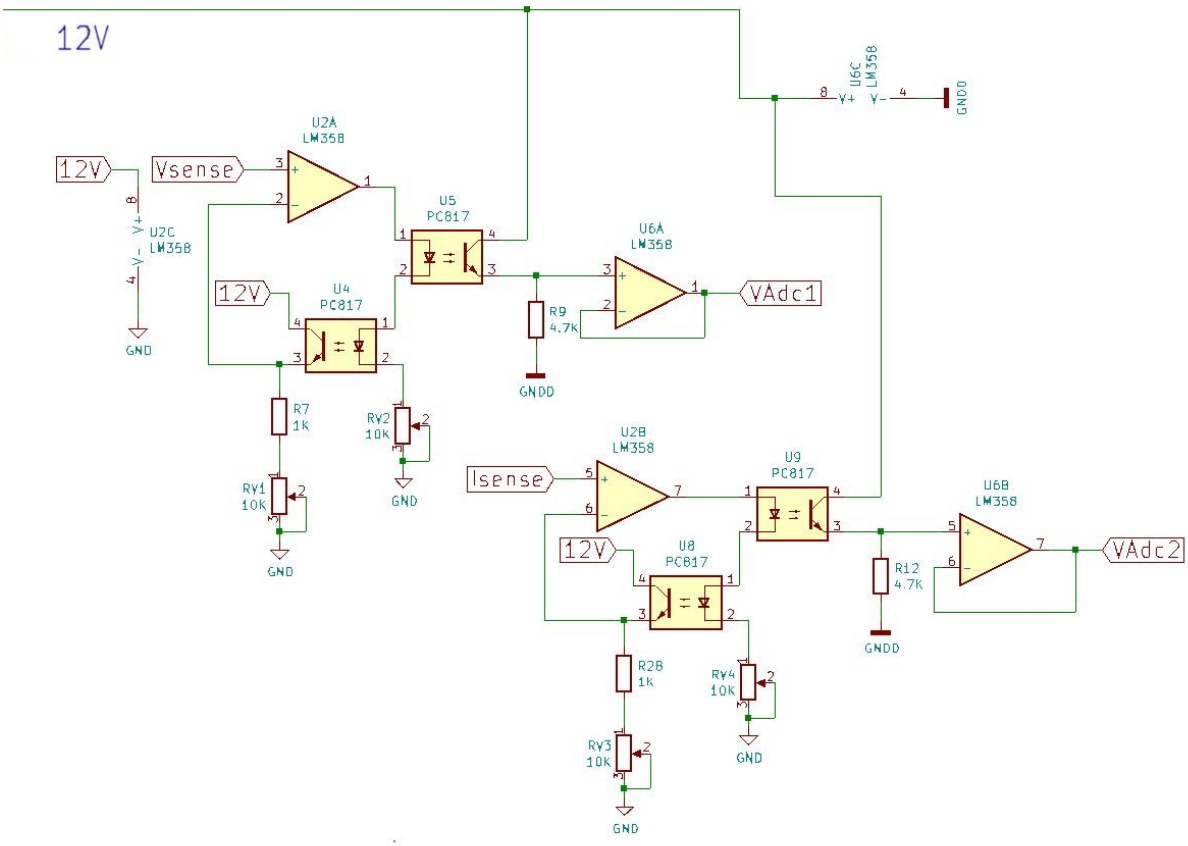
Εικόνα 4.12: Κύκλωμα Ανάδρασης Έντασης



Εικόνα 4.13: Breakout Board INA169

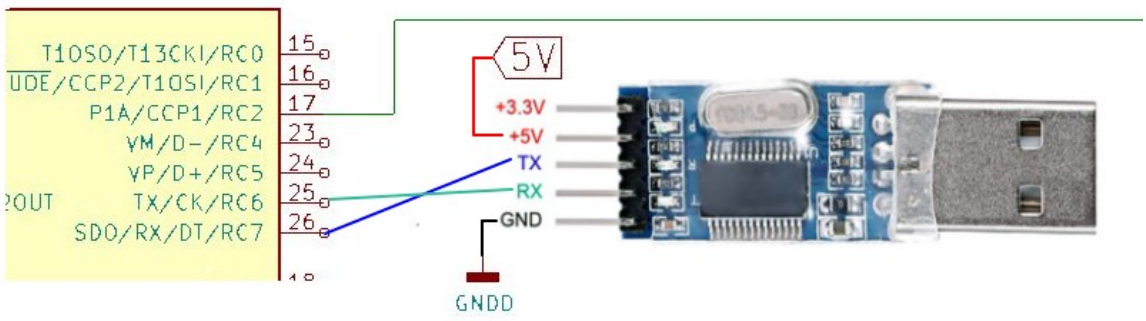
Η R27 είναι αντίσταση – σένσορας. Επάνω της μετριέται η ένταση του ρεύματος από την διαφορά τάσης, η R24 είναι η αντίσταση με τη οποία πολλαπλασιάζεται η τάση εξόδου του INA169 στην περίπτωση μας είναι 1:1. Το κύκλωμα συνδέθηκε με την μορφή εξωτερικού κυκλώματος (breakout-board).

Το σήμα εξόδου (Vsense ή Isense) προκαλεί θετική μετατόπιση της εξόδου των U2A και U2B, αυτό προκαλεί την αγωγή LED στους optocoupler U4 και U5 για Vsense, U8 και U9 για Isense.
Τα U4,U8 παρέχουν ένα σήμα θετικής ανάδρασης. Όταν και οι δύο είσοδοι των τελεστικών ενισχυτών είναι ίσες, οι έξοδοι των U2A και U2B σταματούν να ολοκληρώνουν. Ταυτόχρονα, οι έξοδοι των U5 και U9 παρέχουν ένα θετικό σήμα στους U6A και U6B. Ρυθμίζοντας το RV1 έτσι ώστε RV1 + R9 να είναι περίπου με 4.7K και RV3 + R2 να είναι περίπου με 4.7K, προκαλεί τις εξόδους των U6A και U6B να πλησιάσουν το σήμα τάσης εισόδου.
Περαιτέρω ρύθμιση των RV1 και RV3 θα προσαρμόσει τη διαφορά του CTR έτσι ώστε η τάση εξόδου να ταιριάζει ακριβώς με την τάση εισόδου. Τα ποτενσιόμετρα RV2 και RV4 προσαρμόζουν τις διακυμάνσεις του CTR.



Εικόνα 4.14: Κύκλωμα Γραμμικού Φωτοζεύκτη Σήματος

4.1.4 Κύκλωμα UART σε TTL

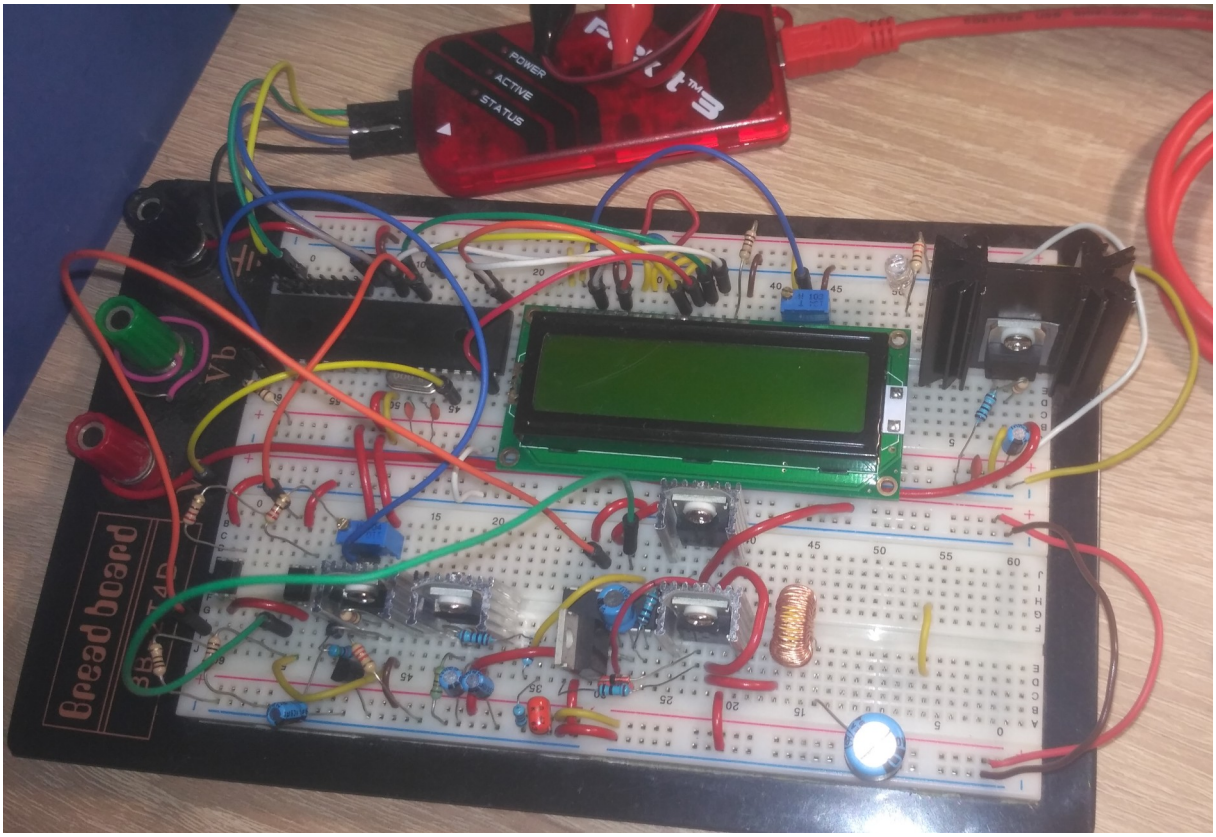


Εικόνα 4.15: Συνδεσμολογία Μεταξύ PL2303 και Μικροελεγκτή

Για την διασύνδεση μεταξύ Η/Υ και μικροελεγκτή χρησιμοποιήθηκε ένα εξωτερικό κύκλωμα (breakout-board) PL2303. Που μετατρέπει το UART σε TTL. Δηλαδή μετατρέπει το σήμα ώστε να το διαβάζει ο Η/Υ μέσω καλωδίου USB. Το PL2303 συνδέεται στην θύρα PORTC του μικροελεγκτή στις ακίδες RC6 με RX και RC7 με TX.

4.2 Αποτελέσματα Κυκλώματος *S*ynchronous Buck Converter σε Ράστερ

Σε αυτό το κεφάλαιο θα αναλυθούν τα αποτελέσματα του κυκλώματος *S*ynchronous Buck Converter που έγιναν πάνω σε ράστερ (διάτρητη πλακέτα που δεν χρειάζεται κολλήσεις)



Εικόνα 4.16: Ημιτελές Κύκλωμα Synchronous Buck Converter

Στη παραπάνω φωτογραφία φαίνεται το κύκλωμα του μικροελεγκτή, των optocoupler και του κυκλώματος ισχύος. Στη συγκεκριμένη διάταξη χρησιμοποιούμε ένα ADC με το οποίο ο μικροελεγκτής διαβάζει την τάση από ένα ποτενσιόμετρο και την μετατρέπει σε παλμούς PWM, που στην συνεχεία μέσω των optocoupler ελέγχουν τον οδηγό.

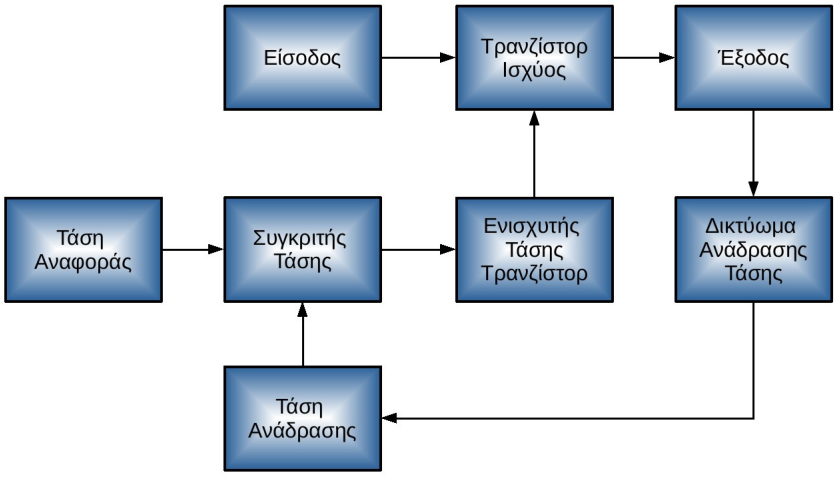
Με την παραπάνω διάταξη είχαμε ικανοποιητικά αποτελέσματα, με εύρος τάσης εξόδου περίπου 0 μέχρι 17V (μπορούσε να ανέβει και άλλο). Όταν συνδέσαμε το κύκλωμα ανάδρασης τάσης και έντασης, δηλαδή τρία ADC, στην οθόνη του μικροελεγκτή παίρναμε μετρήσεις κοντινές σε αυτά που μετρούσαμε με το πολύμετρο, στα χαμηλά ρεύματα υπάρχει σημαντικό σφάλμα, γι’αυτό ευθυνόταν ότι το INA169 μπορεί να μετρήσει ελάχιστη διαφορά τάσης πάνω στην αντίσταση - σένσορα ρεύματος 50mV ή και η μετατροπή από 10-bit σε float αριθμό. Η μετρούμενη τάση ήταν κοντά σ’ αυτή που μετρούσαμε με το πολύμετρο, το σφάλμα ήταν ασήμαντο, δεν μετρούσε σταθερά τρεμόπαιζε.

Στη συνέχεια, προγραμματίσαμε τον μικροελεγκτή με τον PID κωδικό. Η εισαγωγή από το πληκτρολόγιο δούλευε σωστά όπως επίσης και η επικοινωνία με τον Η/Υ καθώς και οι μετρήσεις από τον ADC. Υπήρχε όμως πρόβλημα ταχύτητας μεταξύ της εύρεσης του σφάλματος που είναι ίσο με το μηδέν και της αλλαγής της τάσης εξόδου. Δηλαδή η τάση εξόδου δεν ήταν σταθερή είχε αυξομειώσεις.

Λόγω των πολλών αυξομειώσεων ο μικροελεγκτής κρατούσε την τάση εξόδου σταθερή για λίγα δευτερόλεπτα με αποτέλεσμα ο PID να υπολογίζει αποτελέσματα τα οποία ήταν είτε πολύ μεγάλα είτε πολύ μικρά και κάποια δευτερόλεπτα σταθερός. Έγιναν αρκετές αλλαγές στο πρόγραμμα οι οποίες δεν κατάφεραν να διορθώσουν το πρόβλημα χρονισμού, έτσι αποφασίστηκε να γίνει αλλαγή στο κύκλωμα, από παλμοτροφοδοτικό να αντικατασταθεί με γραμμικό.

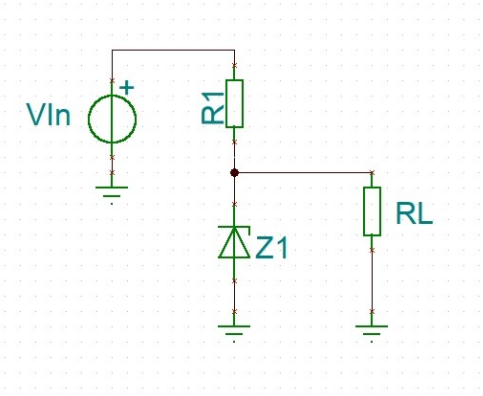
Κεφάλαιο 5 Γραμμικό Τροφοδοτικό

5.1 Αρχή λειτουργίας Γραμμικού Τροφοδοτικού



Εικόνα 5.1: Block Διάγραμμα Γραμμικού Ελεγκτή

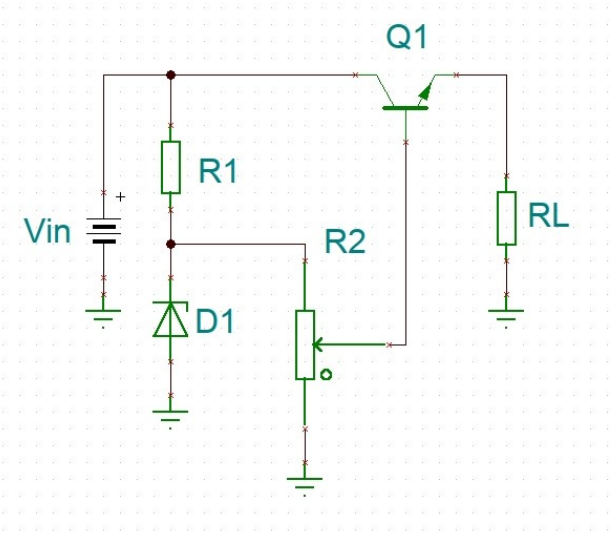
Το γραμμικό τροφοδοτικό είναι πιο απλό στη λειτουργία του από το παλμοτροφοδοτικό καθώς δεν υπάρχουν διακοπτικά στοιχεία. Αντί να ανοιγοκλείνουμε έναν διακόπτη transistor πολλές φορές το δευτερόλεπτο, έχουμε ένα transistor που ελέγχουμε την αγωγιμότητα του με μια DC τάση καθόλη την διάρκεια της λειτουργίας του. Η DC τάση στη βάση του transistor ρυθμίζει την τάση εξόδου, λόγω της συνεχόμενης τάσης έχουμε μικρή απόδοση όπως αναφέρθηκε στο 1ο κεφάλαιο.



Εικόνα 5.2: Σταθεροποιητής τάσης με Δίοδο Zener

Το κύκλωμα που σχεδιάσαμε είναι ένας γραμμικός ελεγκτής, παίρνουμε την DC τάση από ένα οποιοδήποτε DC τροφοδοτικό και την υποβιβάζουμε στην τάση που θέλουμε. Για να κατανοηθεί ο γραμμικός ελεγκτής θα εξηγηθούν δυο κυκλώματα που έχει βασιστεί ο ελεγκτής της εικόνας 5.1.
Στο κύκλωμα της εικόνας 2 έχουμε στην έξοδο σταθερή τάση όση και της διόδου zener .

Αν πολώσουμε ορθά μια zener θα δράσει σαν μια κανονική δίοδος επιτρέποντας να περάσει ρεύμα, αν όμως την πολώσουμε ανάστροφα όταν φτάσει σε ένα επίπεδο ανάστροφης τάσης, η δίοδος αρχίζει να άγει με ρεύμα που ρέει στην ανάποδη φορά, έτσι έχουμε τάση στην έξοδο. Όμως, παίρνουμε μια σταθερή τάση αλλά εμείς θέλουμε μεταβλητή τάση. Επίσης δεν μπορεί να παρέχει αρκετό ρεύμα.



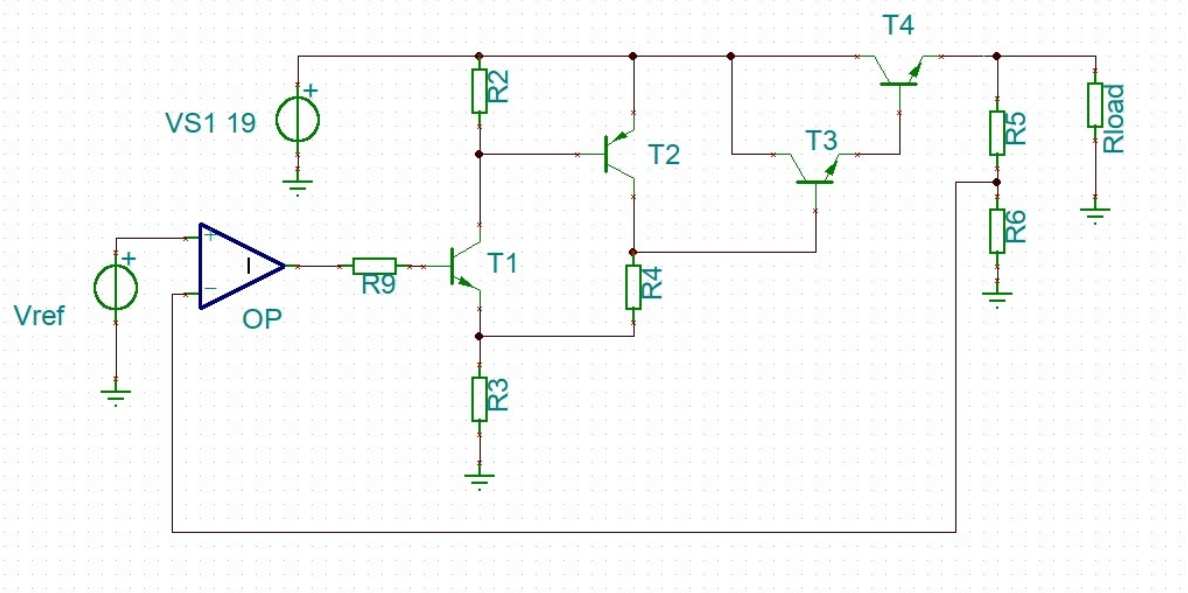
Εικόνα 5.3: Κύκλωμα Μεταβλητού Σταθεροποιητή Τάσης

Για να πάρουμε μεταβλητή τάση στην έξοδο βάζουμε ένα transistor για να παίρνουμε παραπάνω ρεύμα και ένα ποτενσιόμετρο για να μεταβάλουμε την τάση εξόδου. Η τάση εξόδου είναι σταθερή, το εύρος της τάσης εξόδου είναι από μηδέν μέχρι περίπου την τάση της zener. Αν βγάλουμε την zener θα έχουμε μεγαλύτερο εύρος τάσης εξόδου από μηδέν έως περίπου την τάση εισόδου αλλά χάνουμε την σταθερότητα γιατί αν αλλάξει η τάση εισόδου θα αλλάξει ανάλογα και η έξοδος.

Για να έχουμε μεγάλα ρεύματα, μεταβλητή σταθερή έξοδο και έλεγχο σχεδιάσαμε το κύκλωμα του γραμμικού ελεγκτή τάσης της εικόνας 5.4, υπήρχαν και άλλα κυκλώματα που πετυχαίνανε αυτά που θέλαμε άλλα αυτό συνδυαζόταν καλύτερα με μικροελεγκτές.

Το κύκλωμα της εικόνας 5.4 των transistor T1, T2 και των αντιστάσεων R2, R3, R4 λειτουργεί σαν ενισχυτής τάσης γιατί η τάση και η ένταση εξόδου του μικροελεγκτή δεν φτάνει για να ελέγξει το T4 transistor ισχύος. Το T3 συνδέθηκε για να μειώσουμε το ρεύμα που χρειάζεται να ελεγχθεί το T4. Τέλος, ο τελεστικός ενισχυτής OP συνδέθηκε σαν συγκριτής τάσης (η τάση εξόδου του έφτανε αλλά συνδέθηκε για άλλο σκοπό), στη μια είσοδο του έχει την τάση από τον μικροελεγκτή και στην άλλη την τάση ανάδρασης εξόδου. Άγει μόνο όταν και οι δυο τάσεις είναι ίσες. Αντί να έχουμε μια δίοδο zener σαν αναφορά τάσης έχουμε την τάση του μικροελεγκτή.

Ο μικροελεγκτής δίνει τάση αναφοράς στον συγκριτή όταν άγει η τάση αυτή θα αυξηθεί – πολλαπλασιαστεί που θα οδηγήσει το transistor προενισχυτή που ελέγχει το transistor ισχύος. Έτσι ρυθμίζουμε την τάση εξόδου. Για να μπορεί ο μικροελεγκτής να παράγει DC τάση χρησιμοποιήθηκε ένα εξωτερικό DAC, το αντίθετο του ADC, παίρνει ψηφιακά σήματα (0 και 1) και τα μετατρέπει σε αναλογικά.

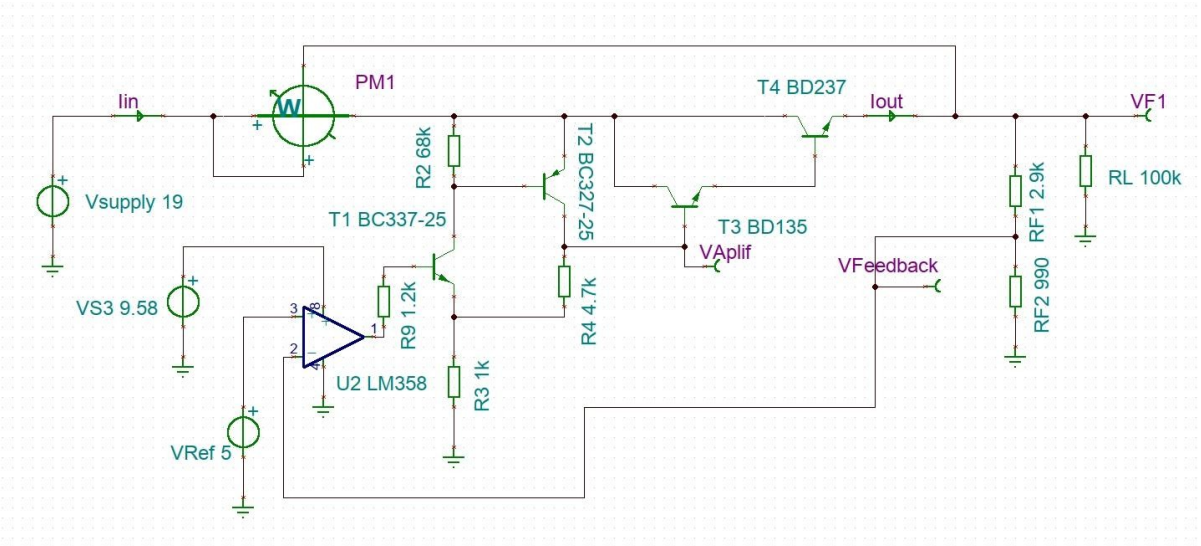


Εικόνα 5.4: Κύκλωμα Γραμμικού Ελεγκτή

5.2 Προσομοιώσεις

5.2.1 Προσομοίωση σε TI TINA

Το πρόγραμμα προσομοίωσης που χρησιμοποιήθηκε για το γραμμικό τροφοδοτικό είναι το TI TINA. Τα αποτελέσματα που θέλουμε να έχουμε είναι η τάση εξόδου να μεταβάλλεται γραμμικά, δηλαδή κάθε αύξηση ή μείωση της τάσης αναφοράς, η τάση εξόδου αυξάνεται ή μειώνεται με το ίδιο βήμα.



Εικόνα 5.5: Κύκλωμα Προσομοίωσης Γραμμικού Τροφοδοτικού

Τα αποτελέσματα της προσομοίωσης είναι κοντά σε αυτό που θέλαμε, το βήμα αλλαγής τάσης είναι σχεδόν ίσο μεταξύ των αλλαγών. Δεν είναι ίσο γιατί τα στοιχεία του κυκλώματος δεν είναι τέλεια. Παρατηρούμε επίσης ότι και η τάση αναφοράς μετά την ενίσχυση ακολουθεί ένα γραμμικό βήμα.

Το βήμα αλλαγής μεταξύ τάσης αναφοράς και τάσης εξόδου είναι ο λόγος μεταξύ τους:

$$V_{step\,1}=\frac{V_{out\,1}}{V_{ref\,1}}=3.94\,V \tag{5.1}$$

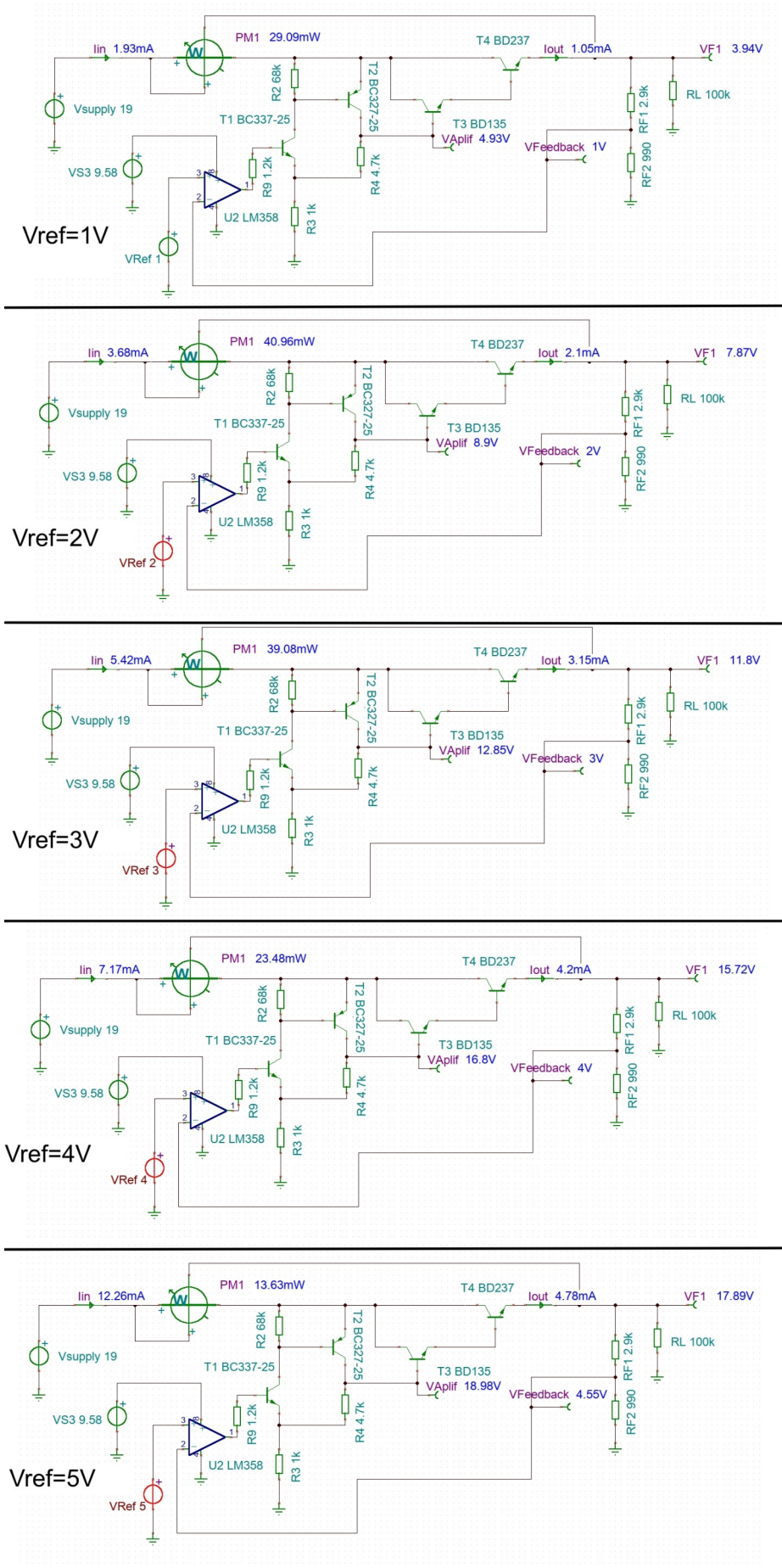
$$V_{step\,2}=\frac{V_{out\,2}}{V_{ref\,2}}=3.935\,V \tag{5.2}$$

$$V_{step\,3}=\frac{V_{out\,3}}{V_{ref\,3}}=3.933\,V \tag{5.3}$$

$$V_{step\,4}=\frac{V_{out\,4}}{V_{ref\,4}}=3.93\,V \tag{5.4}$$

$$V_{step\,5}=\frac{V_{out\,5}}{V_{ref\,5}}=3.578\,V \tag{5.5}$$

Το V_{step5} είναι λιγότερο από τα άλλα βήματα γιατί η τάση στη βάση του transistor ισχύος T4 για να άγει εντελώς πρέπει να είναι μεγαλύτερη από την τάση στον συλλέκτη του, κάτι που στην δικιά μας περίπτωση δεν μπορεί να γίνει.

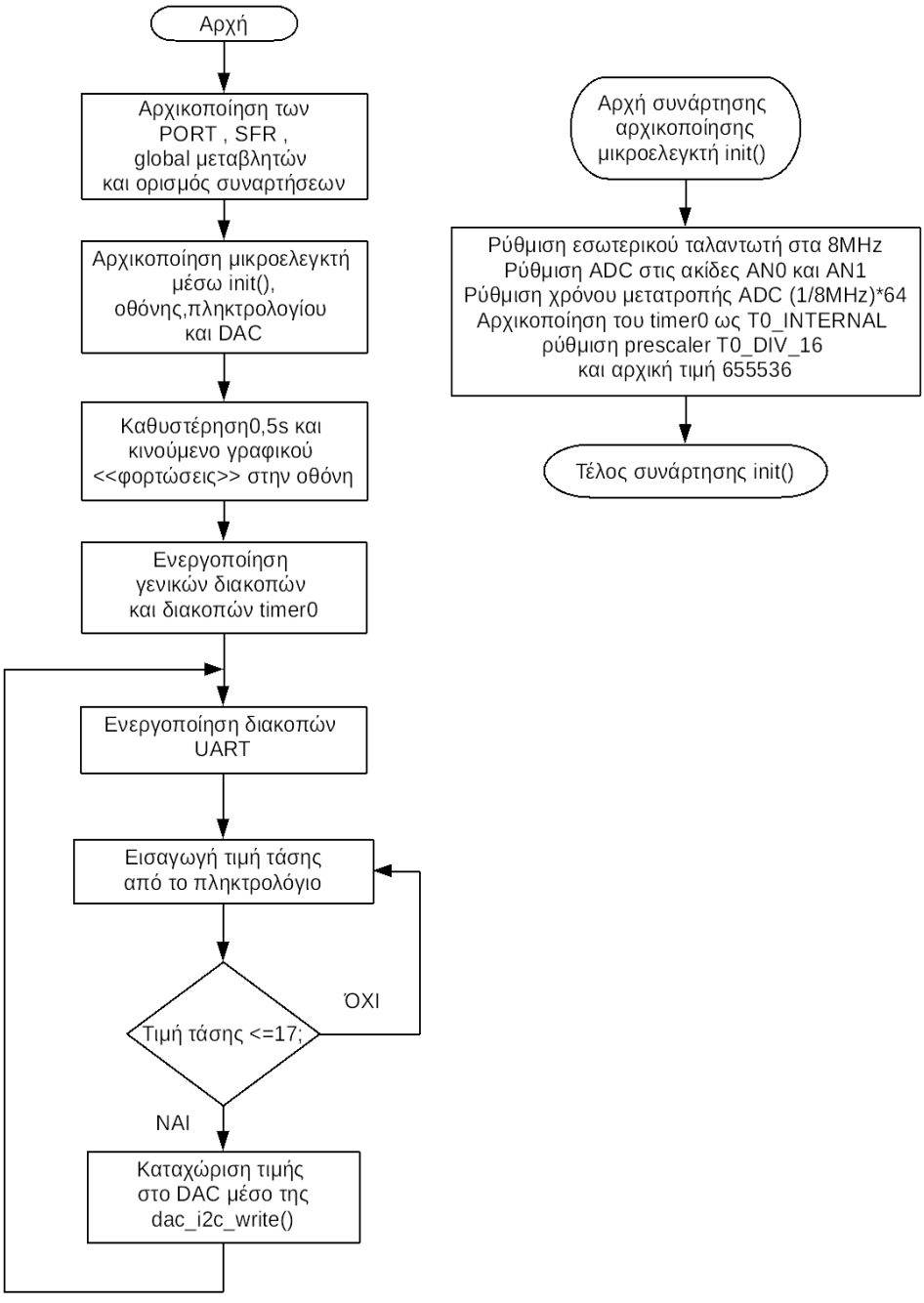


Εικόνα 5.6: Αποτελέσματα Προσομοίωσης Γραμμικού Τροφοδοτικού

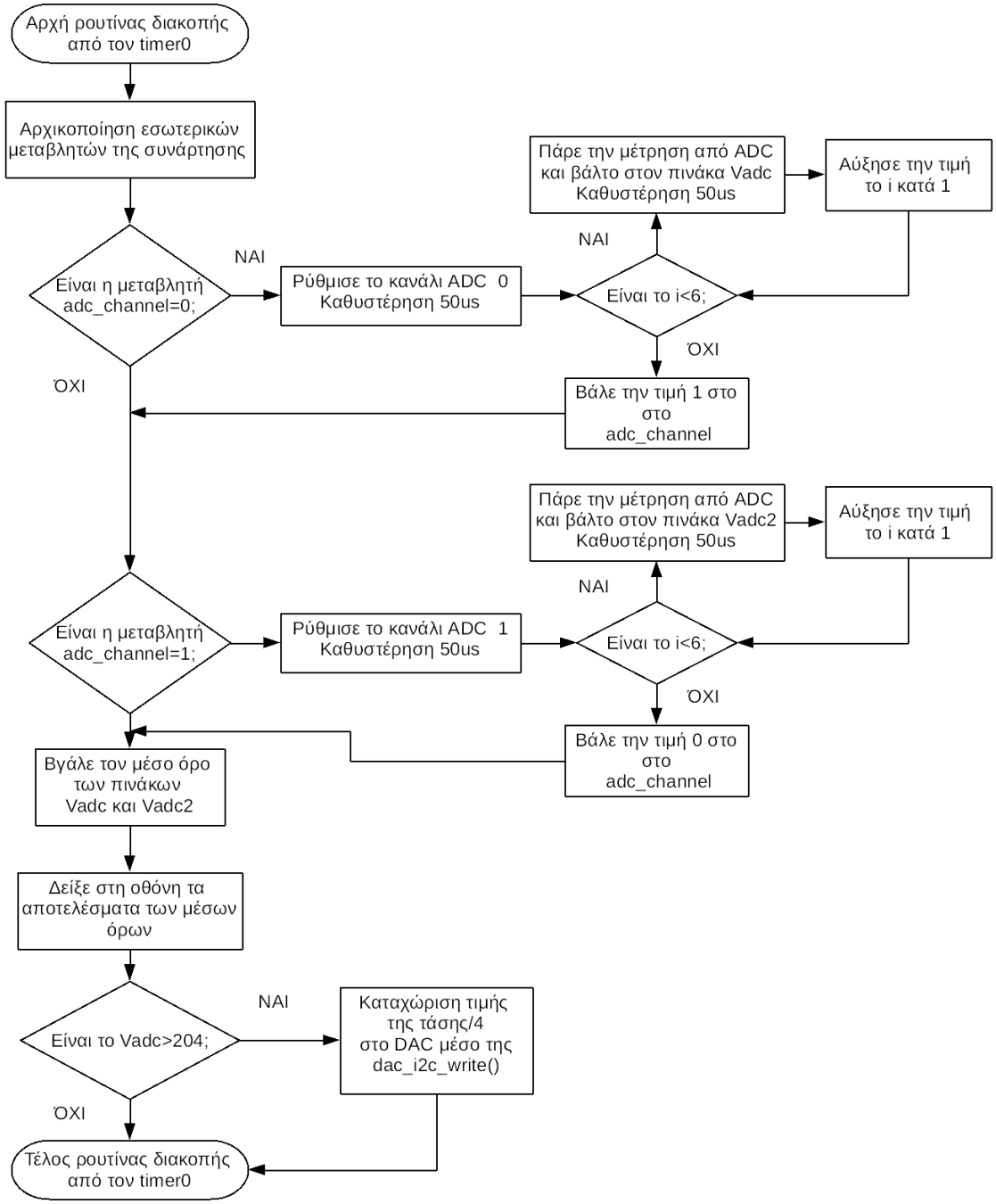
Κεφάλαιο 6 Προγραμματισμός Μικροελεγκτή Σαν Γραμμικό Τροφοδοτικό

Σε αυτό το κεφαλαίο θα εξηγηθεί ο κώδικας του μικροελεγκτή σαν γραμμικό τροφοδοτικό. Ο κώδικας θα εξηγηθεί σε υποκεφάλαια για καλύτερη κατανόηση. Η εφαρμογή και η διαδικασία προγραμματισμού έχουν αναφερθεί στο 3ο κεφαλαίο.

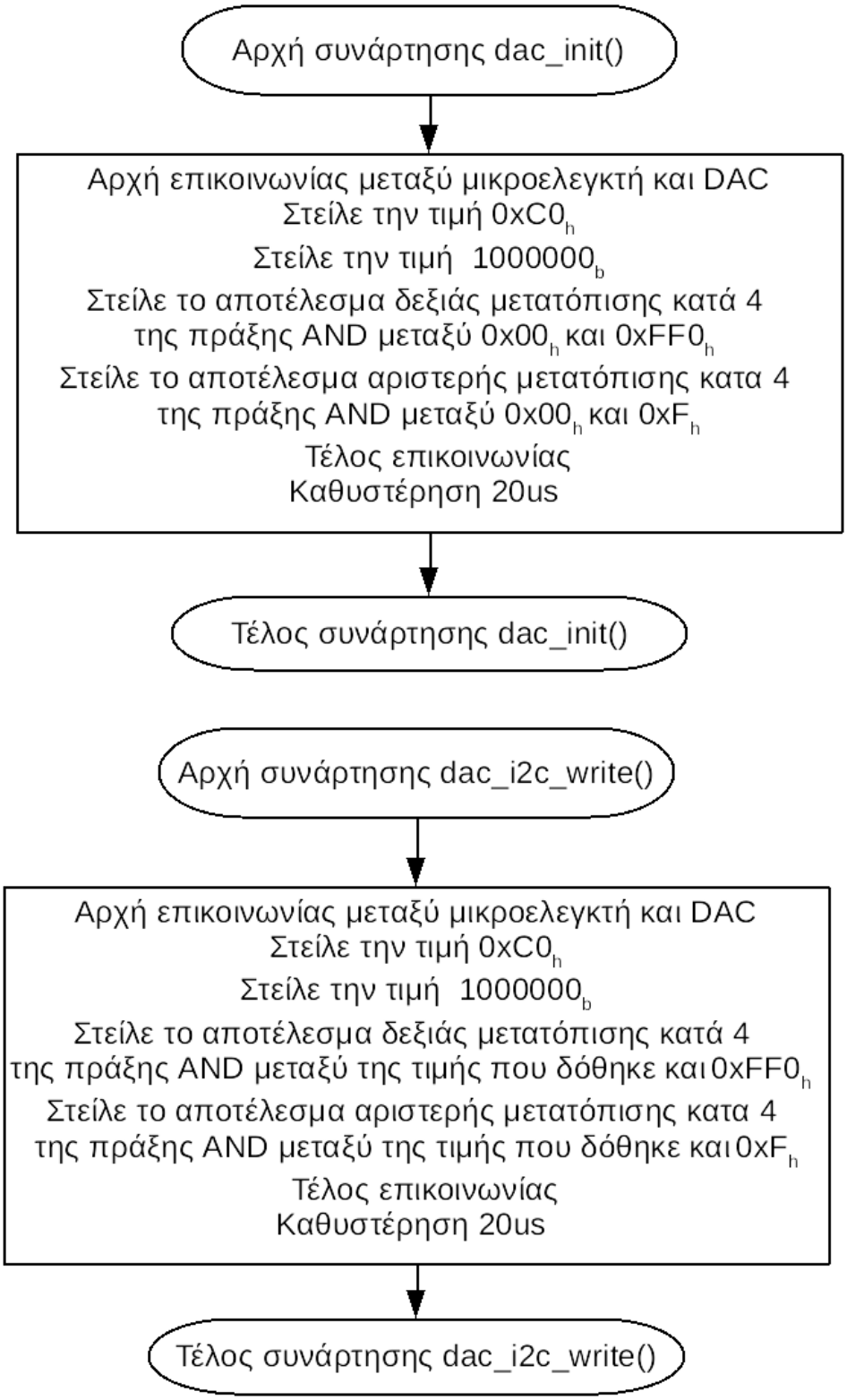
6.1 Flowchart Προγράμματος



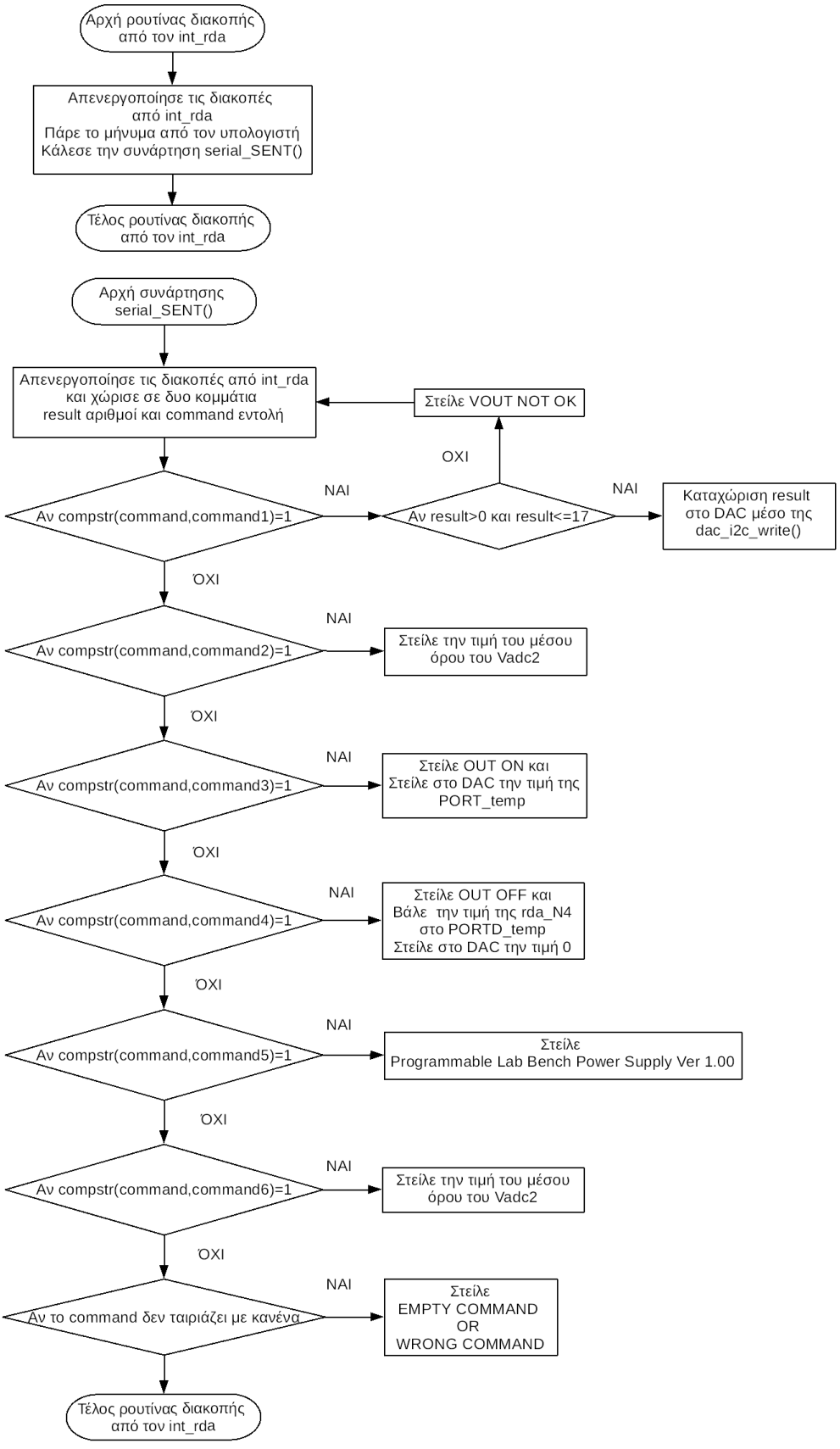
Εικόνα 6.1: Flowchart Κύριου Προγράμματος και Συναρτήσεις init



Εικόνα 6.2: Flowchart Διακοπών από Timer0



Εικόνα 6.3: Flowchart Συναρτήσεων dac_init και dac_i2c_write



Εικόνα 6.4: Flowchart Διακοπών από int_rda και Συνάρτησης serial_SENT

6.2 Κύριο Πρόγραμμα

Το κύριο πρόγραμμα του γραμμικού τροφοδοτικού δεν διαφέρει από του παλμοτροφοδοτικού στο κεφάλαιο 3, για αυτό θα εξηγηθούν μόνο οι διάφορες.

Στο γραμμικό δεν έχει προγραμματιστεί PID ελεγκτής άρα δεν έχει και ccp1 και Timer2. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο (κεφάλαιο 5) για να παράγει DC τάση συνδέθηκε ένα εξωτερικό DAC, για να αρχικοποιήσουμε το DAC γράφτηκε η dac_init και για να στείλουμε την τιμή της τάσης που θέλουμε, γράφτηκε dac_i2c_write, οι οποίες θα εξηγηθούν σε επομένη ενότητα.

Μετά τον έλεγχο, αν η τιμή που πληκτρολογήθηκε είναι μικρότερη του 17 και υπολογιστεί η τιμή που θα σταλεί γίνεται ένας ακόμα έλεγχος αν η τιμή που υπολογίστηκε είναι αρνητική. Αν είναι αρνητική στέλνεται 0 αν δεν είναι στέλνεται αυτό που υπολογίστηκε.

Για να υπολογιστεί η N4_temp διαιρείται με την Voltage_div που είναι η μεταβλητή στην οποία είναι καταχωρημένος ο λόγος του διαιρέτη τάσης. Στη συνέχεια γίνεται μια ακόμα διαίρεση με το step12bit - μεταβλητή που έχει καταχωρηθεί η τιμή του βήματος 12bit ώστε να μετατρέψουμε τον δεκαδικό αριθμό σε 12bit.
Μετά από μετρήσεις της τάσης εξόδου, παρατηρήθηκε ότι υπάρχει μία απόκλιση ίση με 0.07. Για την διόρθωση αυτής της απόκλισης, αφαιρείται 0.07 από την τιμή της μεταβλητής N4_temp.

```
void main(void)
{

    init();

    lcd_init();

    kbd_init();

    dac_int();
    lcd_gotoxy(1,1);
    lcd_putc("\f");
    printf(lcd_putc,"Initializing...");
    unsigned int i=0;
    for(i=0;i<100;i++)
        {
            delay_ms(5);
            lcd_gotoxy((int)i/5,2);
            printf(lcd_putc,"*",);
        }

    lcd_putc("\f");
    enable_interrupts(GLOBAL);
    enable_interrupts(INT_TIMER0);

    char k;
    while(1)
    {
        enable_interrupts(int_rda);
        switch(state)
        {

            case 1:
                k=kbd_getc();

                while (k!=0)
                {
                    lcd_gotoxy(1,1);
                    printf(lcd_putc,"      ");
                    N1= k&0b00001111;
                    lcd_gotoxy(1,1);
                    printf(lcd_putc,"%d",N1);
                    state = 2;
                    break;
                }
                break;
                case 2:
                    k=kbd_getc();
                    while (k!=0)
                    {
                        N2= k&0b00001111;
                        lcd_gotoxy(2,1);
                        printf(lcd_putc,"%d",N2);
                        state = 3;
                        break;
                    }

                    break;
                case 3:
                    k=kbd_getc();
                    while (k!=0)
                    {
                        N3 = k&0b00001111;
                        lcd_gotoxy(3,1);
                        printf(lcd_putc,"%Ld",N3);
                        lcd_putc("\f");
                        N3 =(N1*10+N2+N3/10);
                        if(N3>19)
                        {
                            lcd_gotoxy(1,1);
                            printf(lcd_putc,"Voltage<=17");
                            delay_ms(300);
                            lcd_gotoxy(1,1);
                            printf(lcd_putc,"               ");
                        }
                        else
                        {
                            lcd_gotoxy(1,1);
                            printf (lcd_putc,"%3.2fV",N3);

                            N4_temp=ceil(((N3-0.07)/Voltage_Div)/step12bit);
                            if(N4_temp<0)
                            {
                                N4_temp=0;
                            }
                            N4=N4_temp;
                            dac_i2c_write(N4);
                        }

                        N1=0;
                        N2=0;
                        N3=0;
                        N4=0;
                        state = 1;
                        break;
                    }

        }

    }
}
```

Εικόνα 6.5: Κώδικας Main Κύριας Κεντρικής Συνάρτησης

6.3 Αρχικοποίηση Μικροελεγκτή και Ρύθμιση Εσωτερικών Διακοπών

```
void init(void)
{
    setup_oscillator(OSC_8MHZ);

    setup_adc_ports( AN0_TO_AN1 | VSS_VDD  );
    setup_adc ( ADC_CLOCK_DIV_64 );

    setup_timer_0(T0_INTERNAL | T0_DIV_16);
    set_timer0(65535);

}
```

Εικόνα 6.6: Κώδικας Συνάρτησης init()

Στην συνάρτηση init() του παλμοτροφοδοτικού, όπως αυτή αναλύθηκε στο κεφαλαίο 3, υπάρχουν μικρές αλλαγές. Μειώσαμε το χρονισμό του ADC κατά 64 φορές, άρα έγινε πιο αργός, ώστε να έχουμε καλύτερες μετρήσεις απ' αυτό. Στη συνέχεια, βάλαμε την μέγιστη τιμή στον timer0, μετά από πειράματα καταλήξαμε σε αυτήν, η οποία είναι 65535 και από την (3.1) και (3.2) έχουμε:

$$T_{Timer\,0} = (65536 - PR1) * Tprescaler = 8\,us \qquad (4.1)$$

$$F_{Timer\,0} = \frac{1}{T_{Timer\,0}} = 125\,KHz \qquad (4.2)$$

Ο timer0 μοιάζει με την αρχή της συνάρτησης PID (που περιγράφεται στο κεφαλαίο 3) δηλαδή ο τρόπος με τον οποίο γίνονται οι μετρήσεις από το ADC, η διαφορά είναι ότι και στα δυο κανάλια βάλαμε μια for ώστε να παίρνουμε 6 μετρήσεις και τα αποτελέσματα αποθηκεύονται στους πινάκες Vadc και Vadc2, ένταση και τάση αντίστοιχα.

Κατόπιν βρίσκει τους μέσους όρους των πινάκων και αποθηκεύει τα αποτελέσματα στις μεταβλητές Vadc_mean και Vadc2_mean, ώστε να έχουμε πιο σταθερά αποτελέσματα.

Τα αποτελέσματα φαίνονται στην οθόνη. Τέλος, γίνεται έλεγχος αν η ένταση του ρεύματος είναι μεγαλύτερη του 204_h η 1A. Αν είναι, παίρνει την τελευταία μέτρηση της τάσης και την πολλαπλασιάζει με το 4 ώστε από 10bit να γίνει 12bit και μετά διαιρεί το αποτέλεσμα δια 5 και στέλνεται στο DAC.

```
#INT_TIMER0
void timer0_int(void)
{
    int i;
    if(adc_channel_select==0)
    {
        set_adc_channel(0);
        delay_us(50);
        for(i=0;i<6;i++)
        {
            Vadc[i]=read_adc();
            delay_us(50);
        }
        adc_channel_select=1;
    }
    if(adc_channel_select==1)
    {
        set_adc_channel(1);
        delay_us(50);
        for(i=0;i<6;i++)
        {
            Vadc2[i]=read_adc();
            delay_us(50);
        }
        adc_channel_select=0;
    }
    Vadc_mean=(Vadc[0]+Vadc[1]+Vadc[2]+Vadc[3]+Vadc[4]+Vadc[5])/6;

    Vadc2_mean=(Vadc2[0]+Vadc2[1]+Vadc2[2]+Vadc2[3]+Vadc2[4]+Vadc2[5])/6;



    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.3fA",Vadc_mean*4.88e-3);
    lcd_gotoxy(8,2);
    printf(lcd_putc,"%1.1fV",Vadc2_mean*step10bit*3.9);

    if(Vadc>204)
    {
        dac_i2c_write((Vadc2_mean*4)/5);
    }
}
```

Εικόνα 6.7: Κώδικας Χρονιστή Timer0

6.4 Προγραμματισμός DAC

Για τη σύνδεση μεταξύ μικροελεγκτή και του εξωτερικού DAC χρησιμοποιήθηκε η μονάδα του MSSP και το πρωτόκολλο σειριακής επικοινωνίας I2C.

```
#USE I2C(MASTER,scl=PIN_D0,sda=PIN_D1,FORCE_SW,FAST=400000)
```

Εικόνα 6.8: Δήλωση Λειτουργίας Δίαυλου I2C

Με την παραπάνω εντολή δηλώνουμε ότι ο μικροελεγκτής στέλνει εντολές MASTER, την ταχύτητα της σύνδεσης FAST 400 kbps, scl είναι η ακίδα χρονισμού, sda η ακίδα μετάδοσης και λήψεις bits και FORCE_SW είναι χρήση των ακίδων που ορίσαμε από τον κώδικα.

```
void dac_int(void)
{
    i2c_start();
    i2c_write(0xC0); //Device Address
    i2c_write(0b1000000); //Internal Device Address
    i2c_write((0x00 & 0xFF0) >> 4); //D11 to D4
    i2c_write((0x00 & 0xF) << 4); // D3 to D0
    i2c_stop();
    delay_us(20);
}
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
void dac_i2c_write(unsigned int16 value)
{
    i2c_start();
    i2c_write(0xC0); //Device Address
    i2c_write(0b1000000); //Internal Device Address
    i2c_write((value & 0xFF0) >> 4); //D11 to D4
    i2c_write((value & 0xF) << 4); // D3 to D0
    i2c_stop();
}
```

Εικόνα 6.9: Κώδικες Συναρτήσεων dac_init και dac_i2c_write

Οι κώδικες των συναρτήσεων dac_init και dac_i2c_write είναι σχεδόν ίδιοι, για αυτό θα εξηγηθούν μαζί. Στην αρχή εκτελούν την εντολή i2c_start ώστε να αρχίσει η επικοινωνία μεταξύ του μικροελεγκτή και του DAC, μετά στέλνουμε την τιμή $0xC0_h$ η οποία είναι η διεύθυνση του DAC, μπορεί να υπάρχουν και άλλες εξωτερικές μονάδες που πρέπει να απευθυνόμαστε ξεχωριστά.
Στη συνέχεια, στέλνουν την τιμή 1000000_b που είναι η τιμή της εσωτερικής διεύθυνσης μνήμης, στέλνεται ώστε να ξέρει ο DAC πού να αποθηκεύσει τις τιμές που θα του σταλούν, διαφορετικές διευθύνσεις έχουν άλλες διεργασίες.
Ο DAC είναι 12bit άρα οι τιμές που θα σταλούν θα πρέπει να είναι 12bit για να γίνει αυτό και οι δυο συναρτήσεις θα φτιάξουν δυο 8bit αριθμούς. Ο πρώτος είναι τα bits από 11 έως 4 και ο δεύτερος αριθμός θα είναι τα bits από 3 έως 0. Το dac_init στέλνει πάντα 0 άρα παίρνουμε στην έξοδο του DAC 0 κάθε φορά που αρχικοποιείται. Για την dac_i2c_write παράδειγμα αν βάζαμε την value=4 έχουμε:

$$(10 \wedge 0xFF0_h) \gg 4 = 0000\,0000_b$$
$$(10 \wedge 0xF_h) \ll 4 = 1010\,0000_b$$

Άρα θα σταλεί με την σειρά από D11 σε D0 : 0000 0000 1010 12bit.

6.5 Προγραμματισμός Σύνδεσης Η/Υ με τον Μικροελεγκτή

Η σύνδεση μεταξύ του μικροελεγκτή και Η/Υ είναι ίδια με του παλμοτροφοδοτικού που περιγράφεται στο 3ο κεφαλαίο, με την διαφορά ότι η διακοπή int_rda δεν κάνει τον έλεγχο του μηνύματος που στάλθηκε αλλά καλεί μια συνάρτηση την serial_SENT που κάνει τον έλεγχο που περιγράφεται στο κεφάλαιο 3.

```
#INT_RDA
void serial_interrupt(void)
{

    disable_interrupts(int_RDA);

    gets(temp);

    serial_SENT();

}
```

Εικόνα 6.10: Κώδικας Διακοπής int_rda

```
int compstr(char a[],char b[])
{
    int i;
    int counter=0;
    int length=0;
    length = strlen(b);
    for(i=0;i<length;i++)
    {
        if(a[i]==b[i])
        {
            counter++;
        }
    }
    if(counter==length)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

Εικόνα 6.11: Κώδικας Συνάρτησης compstr

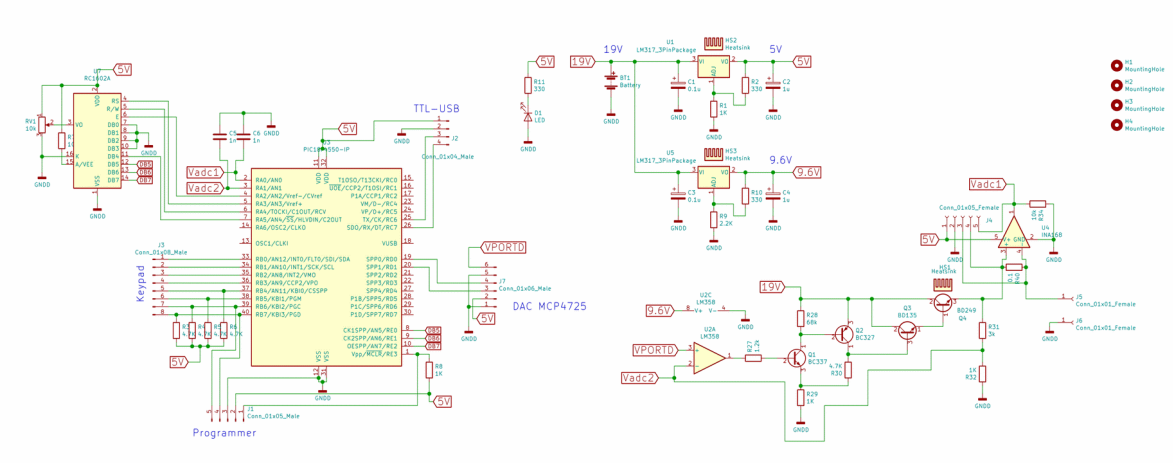
```
void serial_SENT(void)
{
    disable_interrupts(int_RDA);
    result=strtol(temp,&command,10);

    if(compstr(command,command1))
    {
        if(result>=0&&result<=170)
        {
            printf("VOUT OK");
            rda_N4=ceil((((result)/10)/Voltage_Div)/step12bit);
            putc('\n');
            putc('\r');
            printf("%Ld",rda_N4);
            putc('\n');
            putc('\r');
            dac_i2c_write(rda_N4);
            lcd_gotoxy(1,1);
            printf (lcd_putc,"%3.2fV",result/10);

        }
        else
        {
            printf("VOUT NOT OK");
            putc('\n');
            putc('\r');
        }
    }

    else if(compstr(command,command2))
    {

        printf("%f",Vadc2_mean*4.88e-3*3.9);
        putc('\n');
        putc('\r');
    }
    else if(compstr(command,command3))
    {

        printf("OUT ON");
        dac_i2c_write(PORTD_temp);
        putc('\n');
        putc('\r');
    }

    else if(compstr(command,command4))
    {

        printf("OUT OFF");
        PORTD_temp=rda_N4;
        dac_i2c_write(0);
        putc('\n');
        putc('\r');
    }
    else if(compstr(command,command5))
    {

        printf("Programmable Lab");
        putc('\n');
        putc('\r');
        printf("Bench Power Supply Ver 1.00");
        putc('\n');
        putc('\r');

    }
    else if(compstr(command,command6))
    {
        printf("%f",Vadc_mean*4.88e-3);
        putc('\n');
        putc('\r');
    }
    if(!compstr(command,command1)&&!compstr(command,command2)
        &&!compstr(command,command3)&&!compstr(command,command4)
            &&!compstr(command,command5)&&!compstr(command,command6))
    {
        printf("EMPTY COMMAND");
        putc('\n');
        putc('\r');
        printf("OR WRONG COMMAND");
        putc('\n');
        putc('\r');
    }

}
```

Εικόνα 6.12: Κώδικας Συνάρτησης serial_SENT

Κεφάλαιο 7 Πειραματικές Διατάξεις Γραμμικού Τροφοδοτικού

Σ' αυτό το κεφάλαιο θα αναλυθούν οι πειραματικές διατάξεις που αποτελούν το γραμμικό τροφοδοτικό καθώς επίσης τα αποτελέσματα και συμπεράσματα από το κύκλωμα.

7.1 Κυκλώματα τροφοδοτικού και περιφερειακών διατάξεων



Εικόνα 7.1: Κύκλωμα γραμμικού τροφοδοτικού

7.1.1 Κύκλωμα Μικροελεγκτή

Το κύκλωμα του μικροελεγκτή είναι παρόμοια συνδεσμολογημένο με αυτό του παλμοτροφοδοτικού στο κεφάλαιο 4. Ο προγραμματιστής PICkit3, το πληκτρολόγιο, τα ADC και το TTL-USB περιφερειακό έχουν παραμείνει συνδεσμολογημένα με τον ίδιο τρόπο, ενώ η οθόνη συνδέθηκε στις ακίδες των θυρών PORTA και PORTE : RW – RA2, RS – RA3, E – A4, DB4 – RA5, DB5 – RE0, DB6 – RE1, DB7 – RE2. Το εξωτερικό DAC έχει συνδεθεί στην θύρα PORTD με SCL – RD0 με τον χρονιστή clock και SDA – RD1. Προστεθήκαν οι πυκνωτές C5, C6 ώστε να κρατάνε σταθερή την τάση μέτρησης.

Ο μικροελεγκτής για να λειτουργήσει χρειάζεται 5V καθώς και ο τελεστικός ενισχυτής χρειάζεται από 7 μέχρι 10V και αυτές οι τάσεις θα ληφθούν από το LM317. Οι συναρτήσεις που περιγράφουν το κύκλωμα είναι (4.9), (4.10) που μένουν ίδιες, οι (4.11) και (4.12), αλλάζει η V_{IN} άρα έχουμε για 5V έξοδο:

$$P_{5V} = V * I = (V_{IN} - V_{out}) * I_{outmax} = (19-5) * 500\,mA = 7\,W \quad (7.1)$$

$$\Theta_{casetoAmbient} = \frac{T_{junctionMax} - T_{Ambient}}{P} - \Theta_{JunctiontoCase} = \frac{125-30}{7} - 5 = 8.57\,\frac{C^o}{W} \quad (7.2)$$

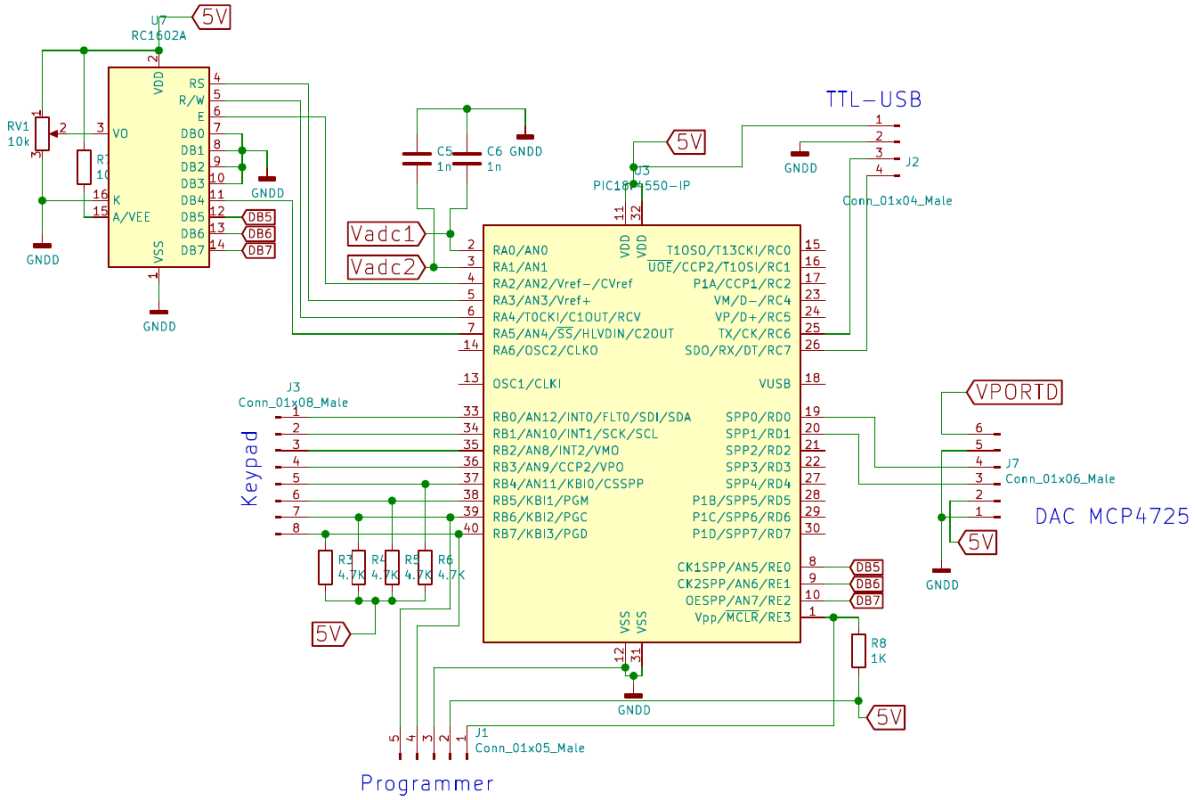
Η ψήκτρα που είχε πάνω του το καλύπτει οριακά. Για την τροφοδοσία του τελεστικού έχουμε:

$$R9 = R10 * \left(\frac{V_{out}}{V_{ref}} - 1\right) = 330 * \left(\frac{10}{1.25} - 1\right) \simeq 2.3\,k\Omega \quad (7.3)$$

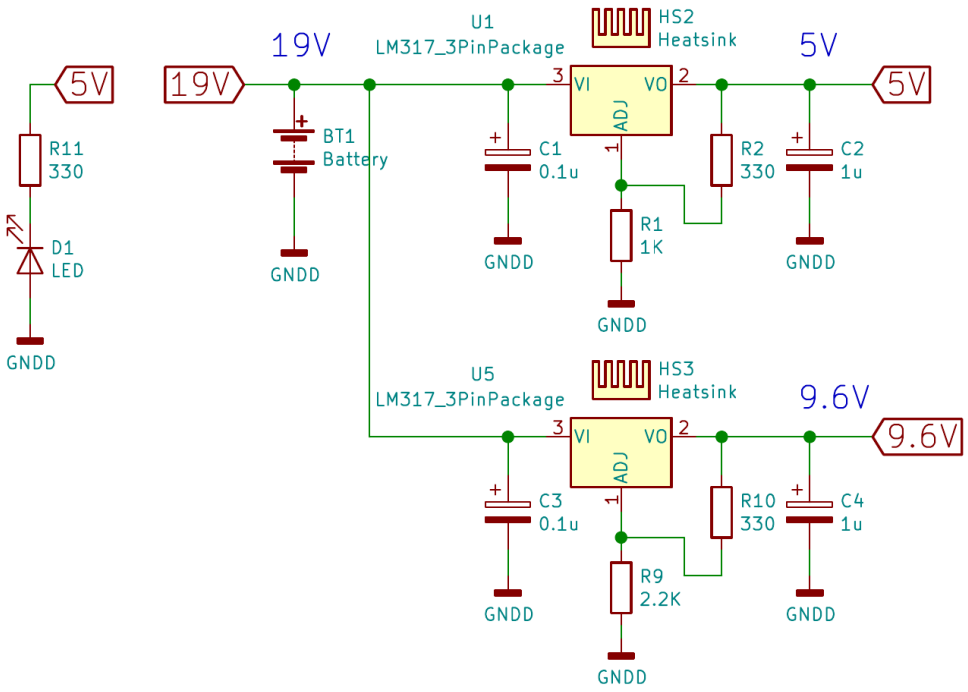
Εμείς βάλαμε R9 ίση με 2.2kΩ που είχε ως αποτέλεσμα να κατέβει η τάση στα 9.6V που δεν δημιούργησε κάποιο πρόβλημα.

$$P_{9.6V} = V * I = (V_{IN} - V_{out}) * I_{outmax} = (19 - 9.6) * 10\,mA = 0{,}094\,W \quad (7.4)$$

Δεν χρειάζεται να μπει ψήκτρα στο συγκεκριμένο LM317 άλλα μπήκε παρόμοια με την HSE-B2111-038 .



Εικόνα 7.2: Κύκλωμα Μικροελεγκτή

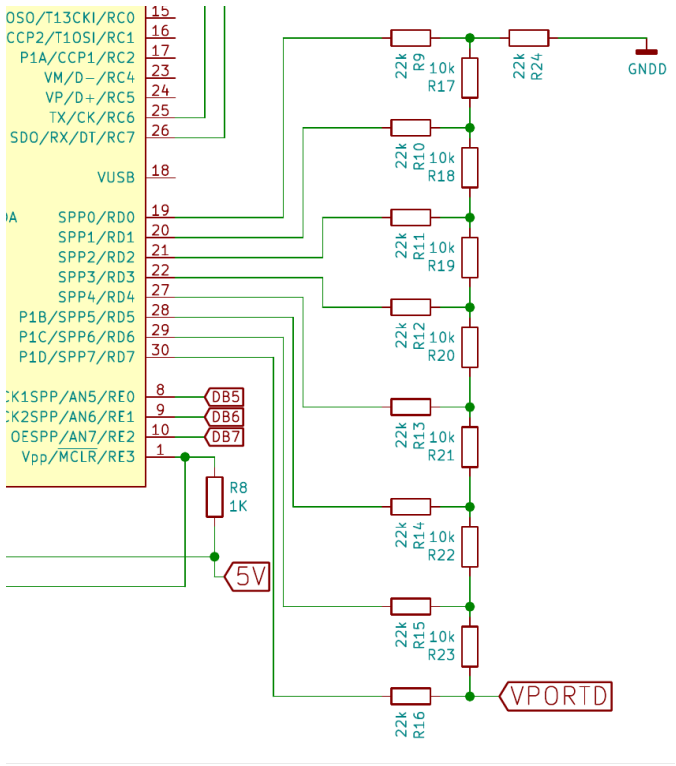


Εικόνα 7.3: Κύκλωμα Τροφοδοσίας

7.1.2 Κύκλωμα R2R Ladder και DAC

Για να παράγει ο μικροελεγκτής συνεχόμενη DC τάση ή άλλα αναλογικά σήματα θα πρέπει να έχει DAC Digital to Alalog Converter δηλαδή από ψηφιακό σήμα σε αναλογικό σήμα. Ένας από τους τρόπους που μπορεί να μετατρέψει έχει αναφερθεί έμμεσα στο 2ο και 3ο κεφάλαιο, δηλαδή με την χρήση PWM και ενός χαμηλοπερατού φίλτρου που είναι μια απλή λύση. Δεν χρησιμοποιήθηκε γιατί δεν θα είχε καλή ανάλυση, θα ήταν αργό και θα χρειάζονταν μεγάλοι πυκνωτές και τέλος, η συχνότητα του φίλτρου θα έπρεπε να είναι πολύ πιο μικρή από αυτή του PWM. Σαν παράδειγμα η φυσική συχνότητα του χαμηλοπερατού φίλτρου του παλμοτροφοδοτικού ήταν 188 Hz ενώ των διακοπών 21kHz.

Στην αρχή χρησιμοποιήθηκε ένα δικτύωμα R2R Ladder που αποτελείται από διακόπτες, οι ακίδες της θύρας PORTD και δυο διαφορετικών τιμών αντιστάσεις τις R και 2R. Εμείς σαν R βάλαμε 10kΩ και 2R 22kΩ κανονικά έπρεπε να ήταν 20 kΩ.



Εικόνα 7.4: Δικτύωμα R2R Ladder

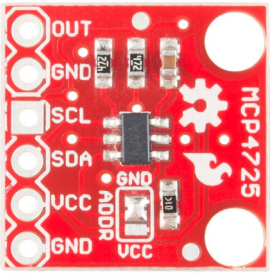
Ο τρόπος που δουλεύει είναι πολύ απλός, κάθε ακίδα λειτουργεί σαν διακόπτης είναι είτε 0V είτε 5V, κάθε ζευγάρι R και 2R λειτουργεί σαν διαιρέτης τάσης. Η τάση που βγάζει κάθε διαιρέτης προστίθεται και παίρνουμε το αναλογικό σήμα.

Στη περίπτωση μας είναι 8bit άρα έχουμε θεωρητικά 255 αναλογικές στάθμες τάσης.
Αν στέλναμε την τιμή $1000\ 0011_b$ που είναι 131_h στο δεκαδικό ή 2.5 σε τάση, οι ακίδες RD0, RD1 και RD7 θα ήταν ανοιχτές και θα παίρναμε τα 2.5V.
Ένας τύπος για να βρίσκουμε την τάση είναι :

$$V_o = V_{CC} * \frac{Value}{2^N} = 5 * \frac{131}{2^8} = 2.55\,V \qquad (7.5)$$

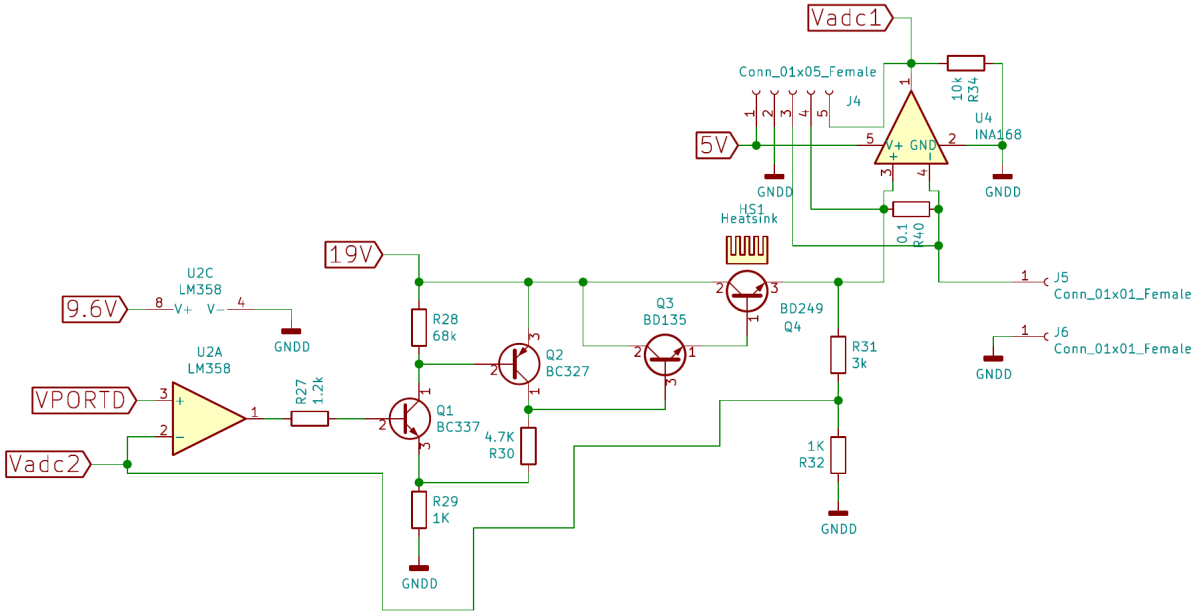
Υπήρχαν όμως προβλήματα με την R2R Ladder όπως η ανακρίβεια των τιμών που είχε στην έξοδο, το μέγεθος και ανοχή των αντιστάσεων που μπορούσε να αλλάξει με τη θερμοκρασία. Να σημειωθεί ότι η ανακρίβεια ευθύνεται και στον κώδικα που είχε γραφτεί τότε.

Γι' αυτό συνδέσαμε έναν εξωτερικό DAC σε μορφή breaoutboard, το MCP4725 που είναι 12bit δηλαδή 4095 στάθμες, η επικοινωνία γίνεται μέσω του πρωτοκόλλου I2C. Είχαμε καλύτερα αποτελέσματα από τη R2R Ladder τα οποία θα παρουσιαστούν σε επομένη ενότητα.



Εικόνα 7.5: Breakout Board του MCP4725

7.1.3 Κύκλωμα Ισχύος Transistor



Εικόνα 7.6: Κύκλωμα Γραμμικού Ρυθμιστή Τάσης

Η τάση εξόδου του DAC συνδέεται στη μη αναστρέφουσα είσοδο το (+) και συγκρίνεται συνεχώς με την τάση ανάδρασης που είναι συνδεμένη στην αναστρέφουσα είσοδο το (-). Ο τελεστικός ενισχυτής δρα ως συγκριτής δηλαδή άγει μόνο όταν η αναστρέφουσα τάση είναι ίση ή μικρότερη της μη αναστρέφουσας και κρατάει σταθερή την έξοδο.

Για να οδηγήσουμε το τρανζίστορ ισχύος Q4 με μέγιστο ρεύμα εξόδου 1A θα πρέπει να έχει στην βάση του 100mA με hfe=100 από το datasheet του Q4, για μειώσουμε το ρεύμα που χρειάζεται θα συνδέσουμε ένα τρανζίστορ μεσαίας ισχύος με συνδεσμολογία Darlington και hfe τουλάχιστον 50 με 100.

Στην δική μας περίπτωση από το datashet του Q3 έχουμε hfe=100, άρα στην βάση του Q4 θα χρειαζόμαστε 100/100=1mA. Το οποίο μπορούν να το διαχειριστούν τα τρανζίστορ Q1 και Q2, που είναι συνδεσμολογημένα σαν ενισχυτής τάσης.

Ο λόγος που χρησιμοποιήθηκε ενισχυτής από τρανζίστορ και όχι από κάποιο τελεστικό ενισχυτή είναι γιατί θέλαμε να αποφύγουμε τις συμμετρικές τάσεις τροφοδοσίας (θετική και αρνητική) καθώς επίσης θέλαμε να μπορεί η έξοδος του ενισχυτή να πηγαίνει από το μηδέν έως την μέγιστη τάση χωρίς ταλαντώσεις στην έξοδο του.

Τα τρανζίστορ Q1 και Q2 έχουν συνδεσμολογηθεί ώστε να ενισχύουν την τάση εξόδου του τελεστικού όσες φορές είναι περίπου ο λόγος [11]:

$$V_{apl}=\frac{R30+R29}{R29}=\frac{5.7}{1}=5.7 \tag{7.6}$$

Στο σχέδιο είναι το BD249 αλλά στο κύκλωμα είναι το BD245C, επιλέχτηκε ώστε να μπορεί να ανταπεξέλθει στην συνεχή ένταση, τάση καθώς και στην ισχύ που πέφτει επάνω του. Το BD135 μπορούσε να ήταν ένα οποιοδήποτε τρανζίστορ μεσαίας ισχύος και επιλέχθηκε γιατί υπήρχε ήδη στην κατοχή μας και γιατί μπορεί να παρέχει το απαραίτητο ρεύμα για να οδηγήσει το BD245C. Τα BC337 και BC327 είναι συμπληρωματικά τρανζίστορ επιλεχτήκαν για το hfe και για την ένταση και τάση που αντέχουν. Τέλος, ο LM358 επιλέχτηκε λόγω του εύρους τάσης που λειτουργεί και voltage output swing from negative rail που φτάνει σχεδόν στο μηδέν.

7.1.4 Κύκλωμα Ανάδρασης Τάσης και Έντασης

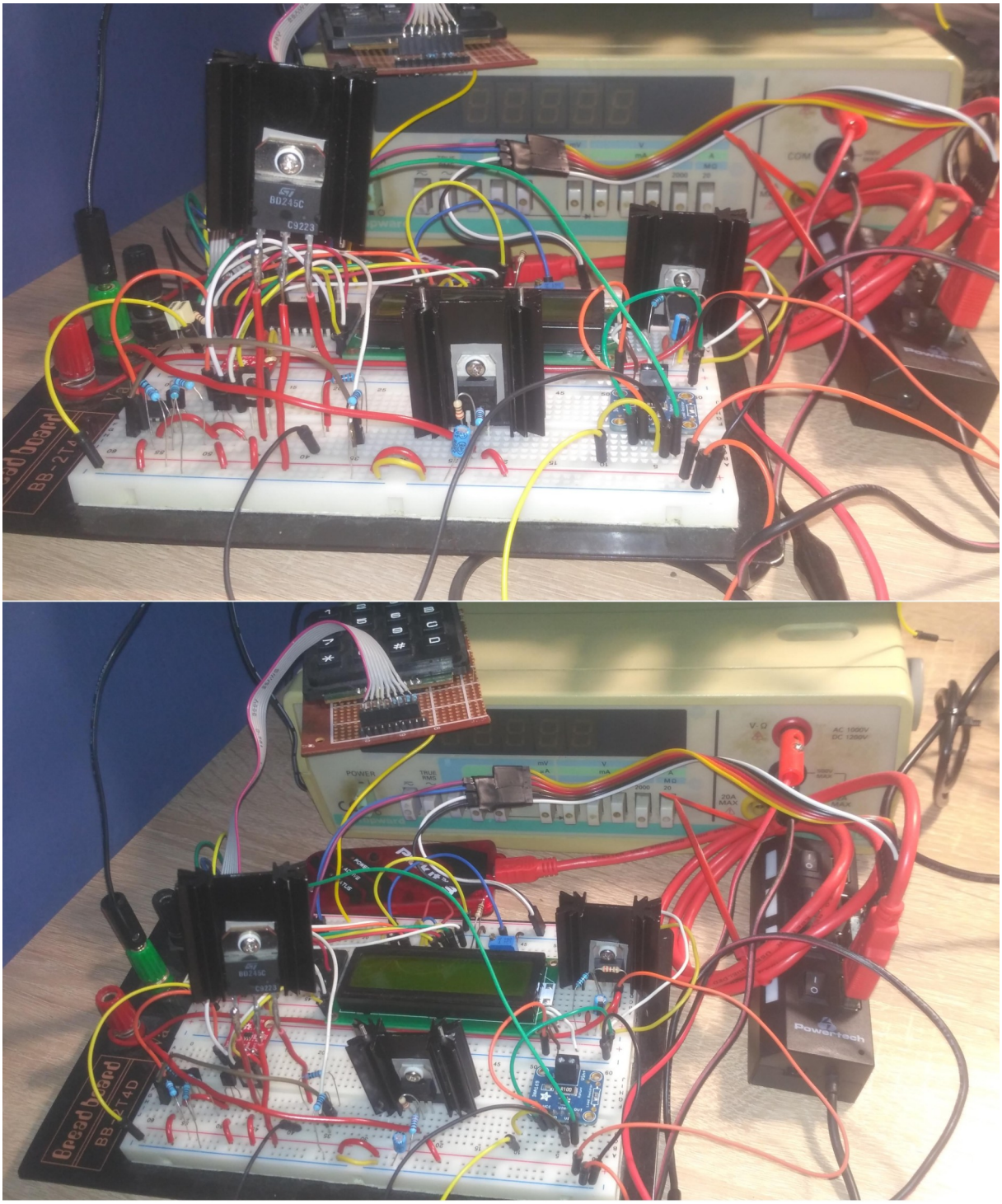
Τα κυκλώματα ανάδρασης τάσης και έντασης είναι παρόμοια με αυτά του παλμοτροφοδοτικού, η μόνη διαφορά είναι ο λόγος του διαιρέτη τάσης άρα και η μέγιστη μετρούμενη τάση , από την (4.26):

$$Vadc2=\frac{R32}{R23+R31}*V_{outLinear}=\frac{1k}{1k+3k}*17=4.25V \tag{7.7}$$

7.1.5 Κύκλωμα Επικοινωνίας με Η/Υ

Το κύκλωμα επικοινωνίας μεταξύ Η/Υ και μικροελεγκτή είναι ακριβώς το ίδιο με αυτό του παλμοτροφοδοτικού στο κεφάλαιο 4.

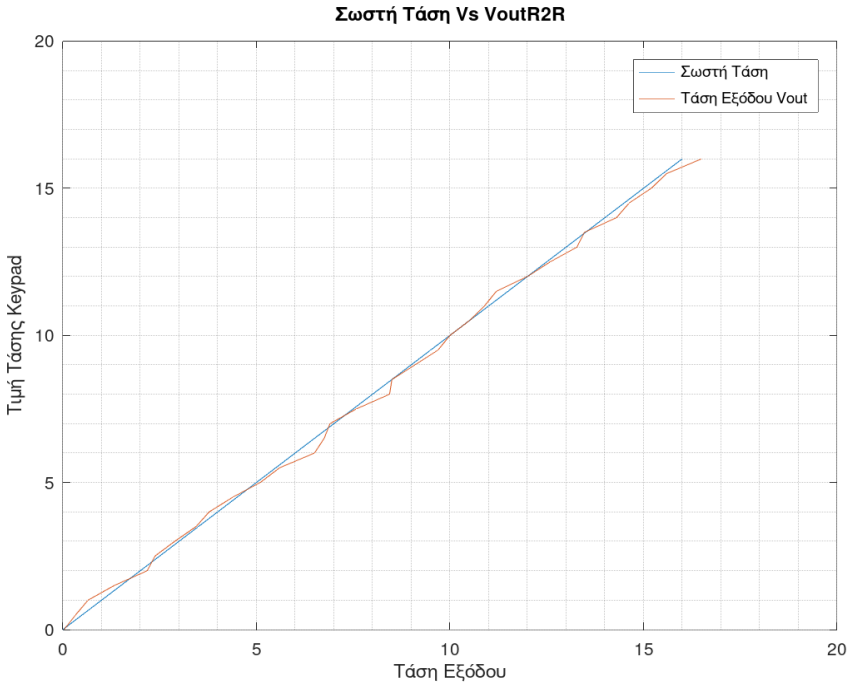
7.2 Πείραμα και Αποτελέσματα σε Breadboard/Ράστερ



Εικόνα 7.7: Γραμμικό Τροφοδοτικό σε Ράστερ

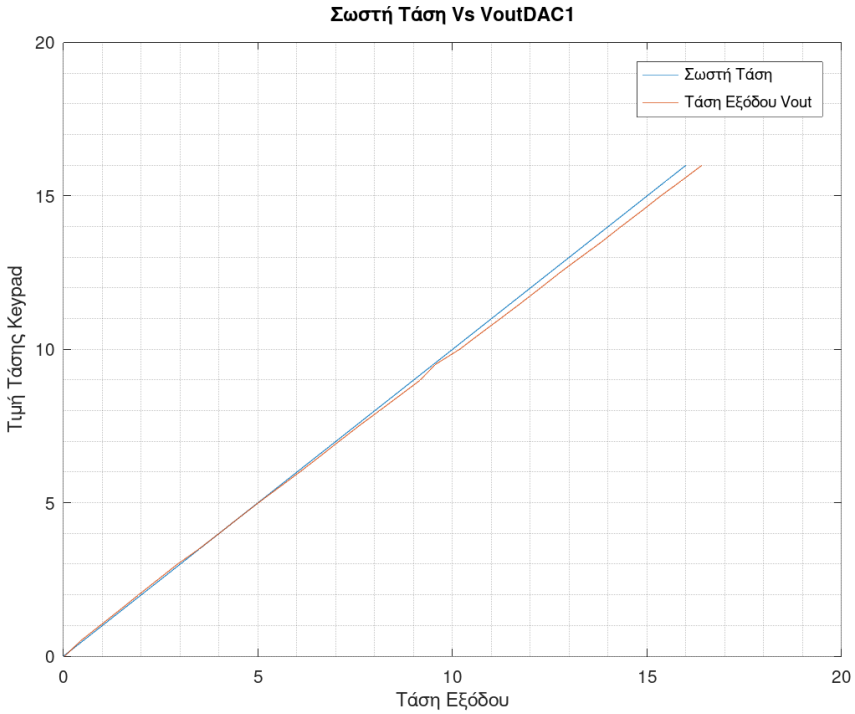
Στην αρχή που είχαμε την R2R Ladder ως μέσο μετατροπής ψηφιακού σε αναλογικό σήμα, τα αποτελέσματα δεν ήταν ακριβή άλλα είχαμε όμως σταθερή τάση στην έξοδο. Η ανακρίβεια οφειλόταν στην χρήση 22kΩ αντιστάσεων αντί για 10kΩ και στον κώδικα, δεν είχαν δοθεί ακριβείς μεταβλητές για την μετατροπή αριθμών όπως στον τελικό κώδικα οι step10bit, step12bit και η Voltage_Div. Επίσης, στα αρχικά πειράματα είχαμε δυο τροφοδοτικά ένα για τον μικροελεγκτή 12V και ένα για το κύκλωμα ισχύος 19V, αυτό έγινε ώστε να μην ζεσταίνεται πολύ το LM317 που τροφοδοτούσε τον μικροελεγκτή.

Επίσης, μια άλλη αλλαγή ήταν ο τρόπος με τον οποίο τροφοδοτούσαμε τον LM358. Στην αρχή ήταν με ένα κύκλωμα της μορφής της εικόνας 5.2 άλλα ζεσταινόταν πολύ και δεν ήταν ικανοποιητικό γι' αυτό το αλλάξαμε σε έναν LM317.



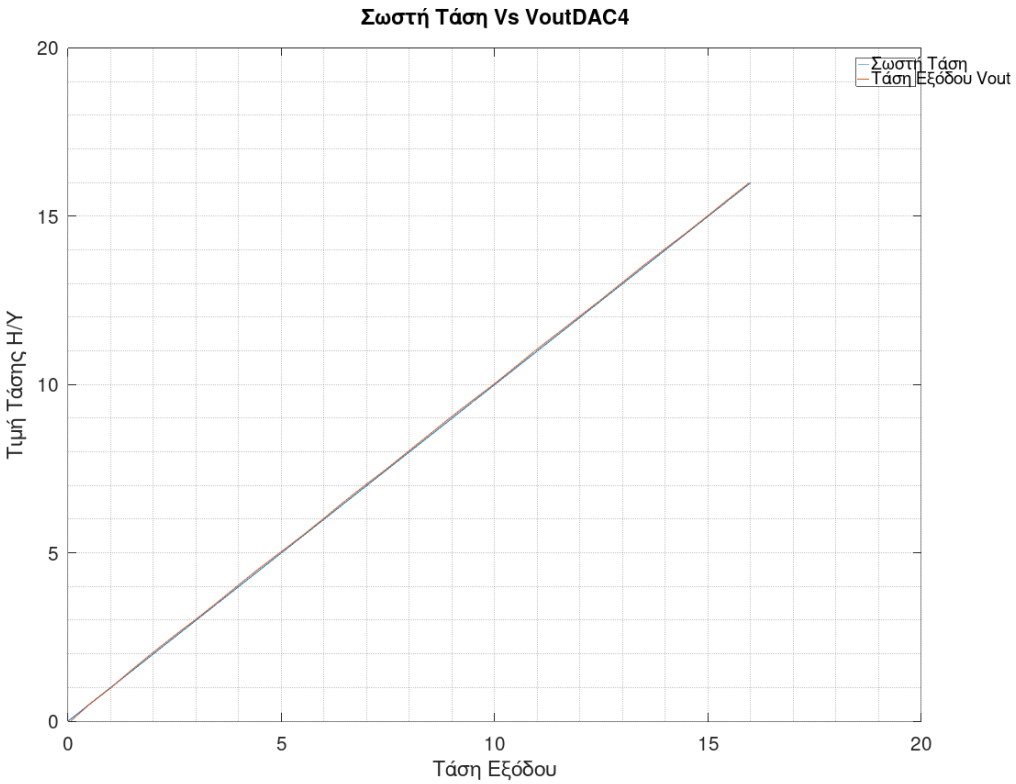
Εικόνα 7.8: Γράφημα Τάσης Αναφοράς με R2R Vs Τάσης που Πληκτρολογήθηκε

Αφότου όμως βάλαμε το MCP4725 τα αποτελέσματα ήταν καλύτερα ακόμα και με την μη χρήση των παραπάνω μεταβλητών.



Εικόνα 7.9: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης που Πληκτρολογήθηκε και χρήση δυο τροφοδοτικών

Παρατηρούμε ότι έχει μικρή απόκλιση, έτσι μετά από αλλαγές έχουμε το τελικό αποτέλεσμα που είναι:



Εικόνα 7.10: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης που Πληκτρολογήθηκε χρήση ενός τροφοδοτικού

Παρατηρούμε πως η τάση που πληκτρολογήσαμε είναι πάρα πολύ κοντά σ’ αυτό το οποίο παίρναμε στην τάση έξοδο.

Τάση Αναφοράς	R2R Ladder	DAC χωρίς μεταβλητές χρήση δυο τροφοδοτικών	DAC χωρίς μεταβλητές χρήση ενός τροφοδοτικού
Σφάλμα	4.6%	2.2%	0.68%

Πίνακας 3: Σφάλμα Τάσης Αναφοράς με Τάση Εξόδου από Πληκτρολόγιο

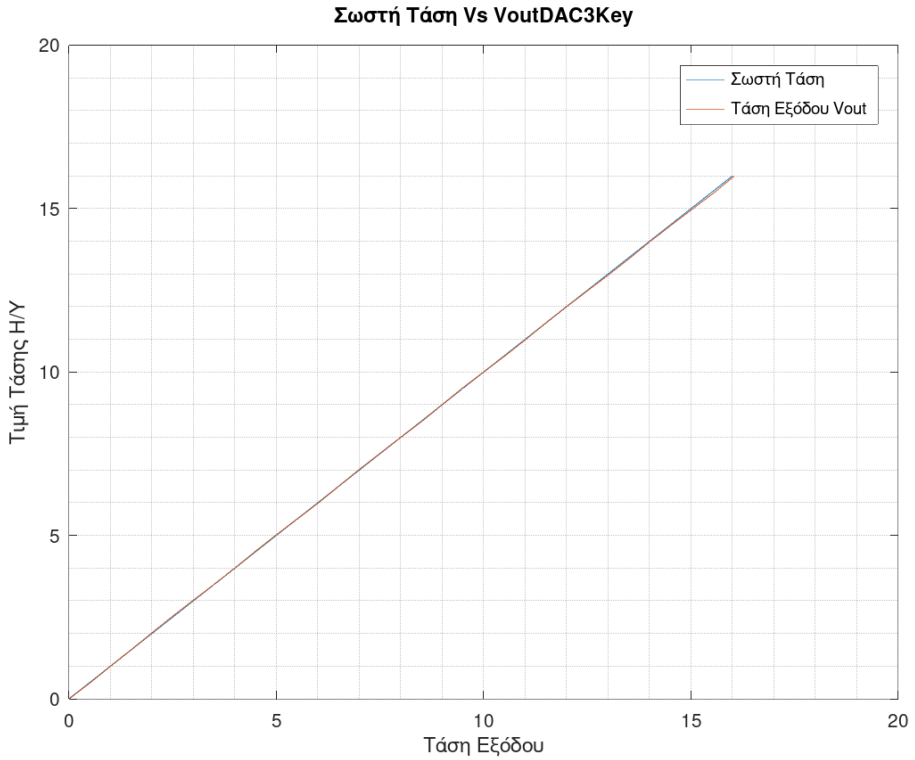
Βλέπουμε από τον παραπάνω πίνακα τη μείωση του σφάλματος από την R2R Ladder να υποδιπλασιάζεται και με την χρήση των μεταβλητών σχεδόν να μηδενίζεται. Είχαμε πολύ καλά αποτελέσματα στο ράστερ, όταν πληκτρολογήσαμε την τιμή που θέλαμε.

Μετά, συνδέσαμε το περιφερειακό εξάρτημα για την επικοινωνία με τον Η/Υ, τα αποτελέσματα ήταν πολύ καλά, να σημειωθεί ότι η τάση αναφοράς παίρνονταν μόνο με το DAC όχι με την R2R Ladder.

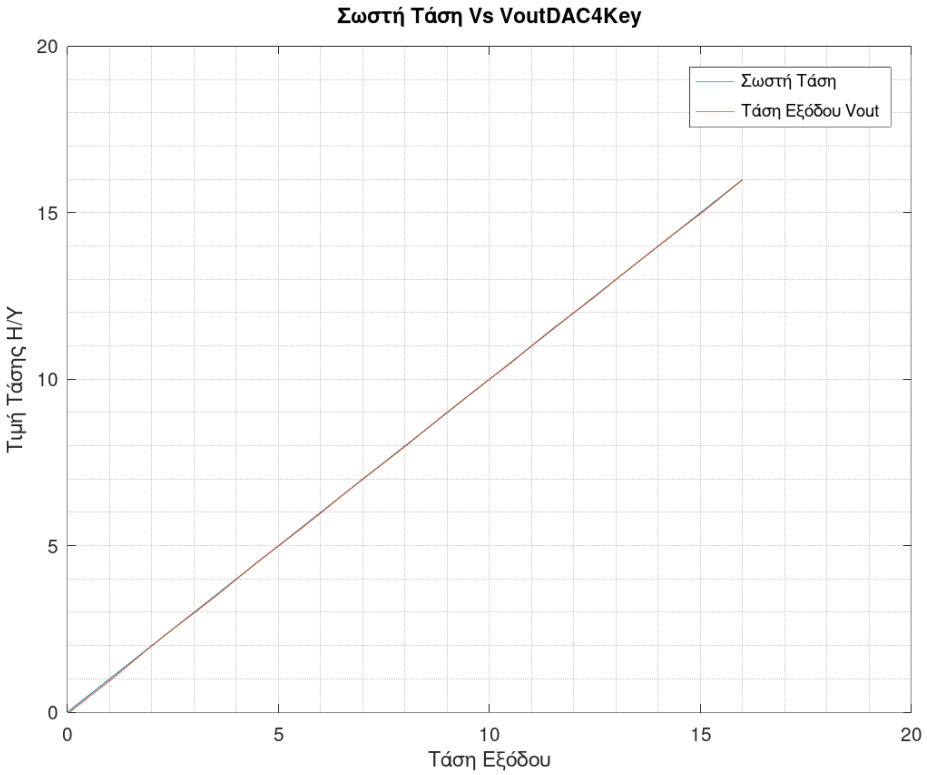
Τάση Αναφοράς	DAC χωρίς μεταβλητές χρήση δυο τροφοδοτικών	DAC χωρίς μεταβλητές χρήση ενός τροφοδοτικού
Σφάλμα	0.35%	0.7%

Πίνακας 4: Σφάλμα Τάσης Αναφοράς με Τάση Εξόδου από Η/Υ

Τα αποτελέσματα είναι διαφορετικά και αυτό οφείλεται στον τρόπο που το πρόγραμμα μετατρέπει τα μηνύματα σε τιμές που στέλνονται στο DAC. Αλλά τα αποτελέσματα είναι πολύ καλά.

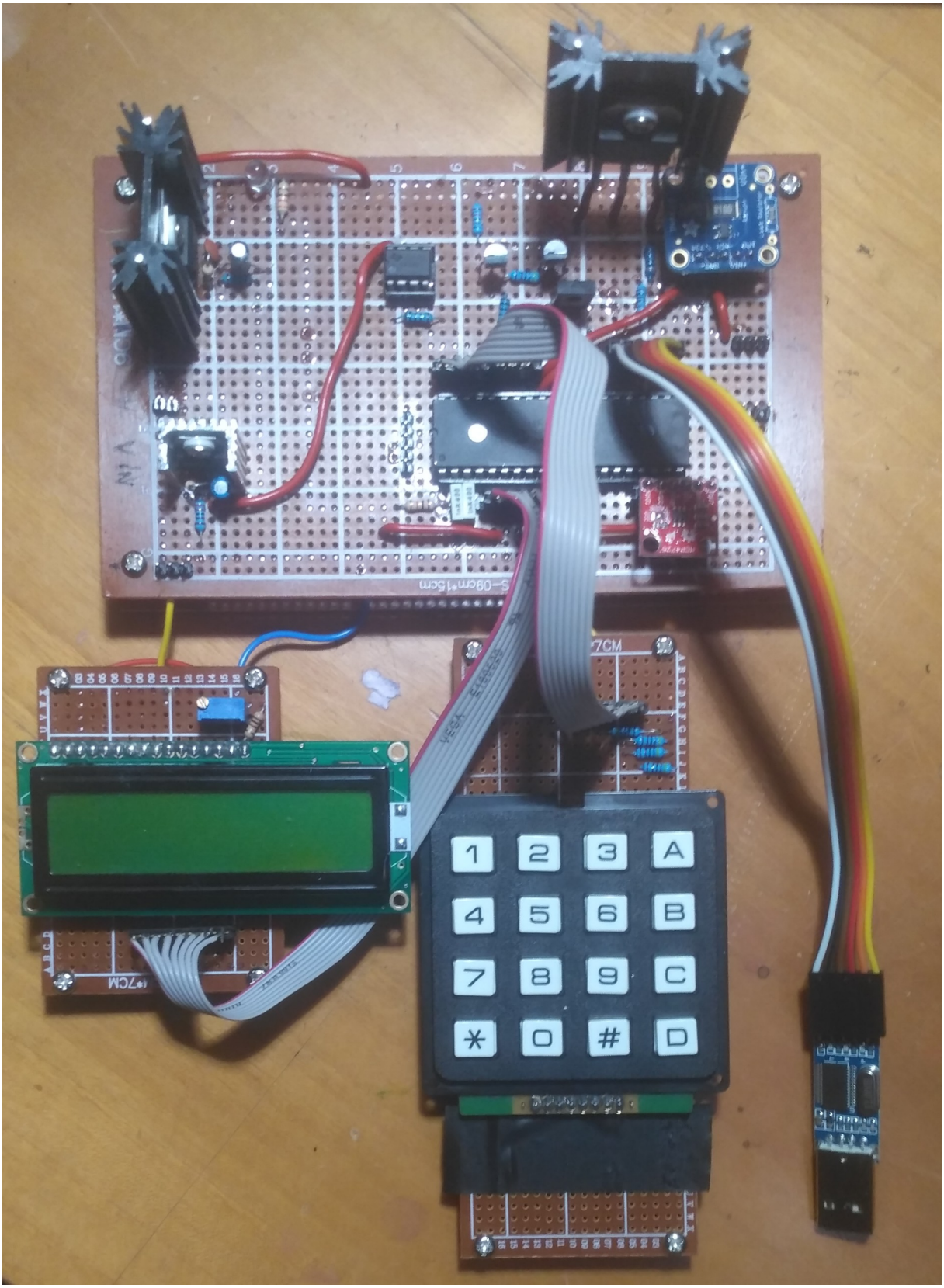


Εικόνα 7.11: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης από Η/Υ και χρήση δυο τροφοδοτικών



Εικόνα 7.12: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης από Η/Υ και χρήση ενός τροφοδοτικού

7.3 Πείραμα και Αποτελέσματα σε Διάτρητη Πλακέτα

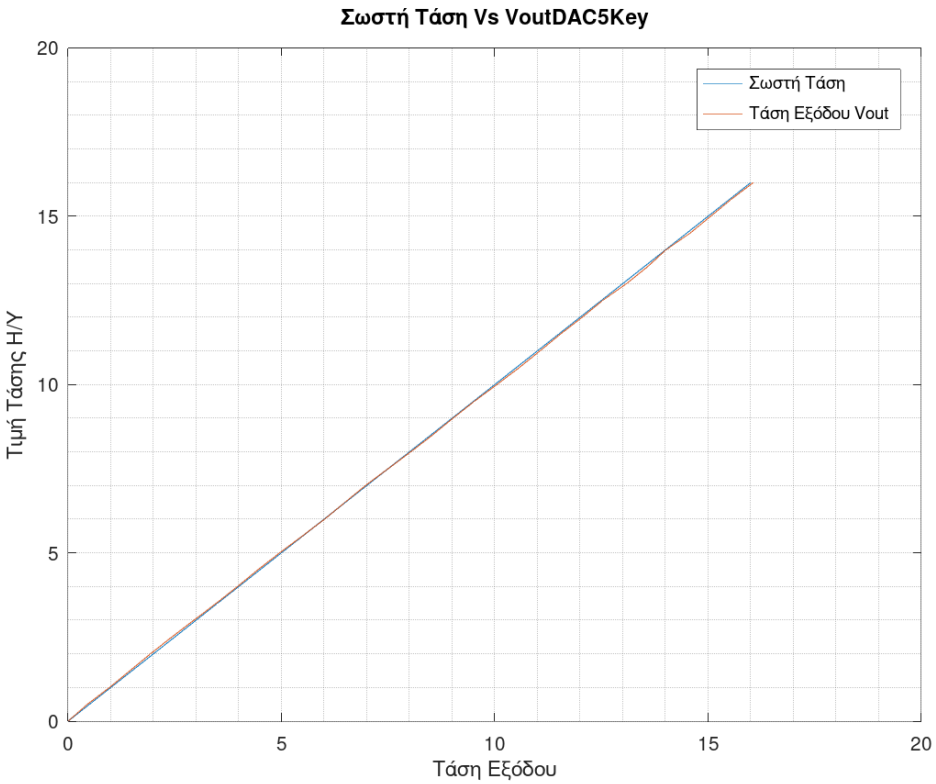


Εικόνα 7.13: Γραμμικό Τροφοδοτικό σε Διάτρητη Πλακέτα

Κατόπιν κολλήσαμε το κύκλωμα σε διάτρητη πλακέτα ώστε να μειώσουμε τον θόρυβο και να ελέγξουμε αν το κύκλωμα λειτουργεί σωστά εκτός του ράστερ.

Τα αποτελέσματα ήταν πολύ ικανοποιητικά, είχαμε ένα μικρό σφάλμα από την εισαγωγή της τιμής της τάσης εξόδου από το πληκτρολόγιο και από τον Η/Υ.

Εικόνα 7.14: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης που Πληκτρολογήθηκε Σε Διάτρητη Πλακέτα



Εικόνα 7.15: Γράφημα Τάσης Αναφοράς με DAC Vs Τάσης Απο Η/Υ Σε Διάτρητη Πλακέτα

Δεν άλλαξε ο κώδικας πολύ για να λειτουργήσει στην πλακέτα πέρα από το -0.07 που δεν χρειαζόταν λόγω μείωσης του θορύβου.

Τάση Αναφοράς	Από πληκτρολόγιο	Από Η/Υ
Σφάλμα	1.15%	1.11%

Πίνακας 5: Σφάλμα Τάσης Αναφοράς με Τάση Εξόδου στη Διάτρητη Πλακέτα

Τα σφάλματα είναι ικανοποιητικά , για τις διαφορές με τα παραπάνω αποτελέσματα μπορεί να ευθύνεται ο κώδικας και οι ανοχές των υλικών.

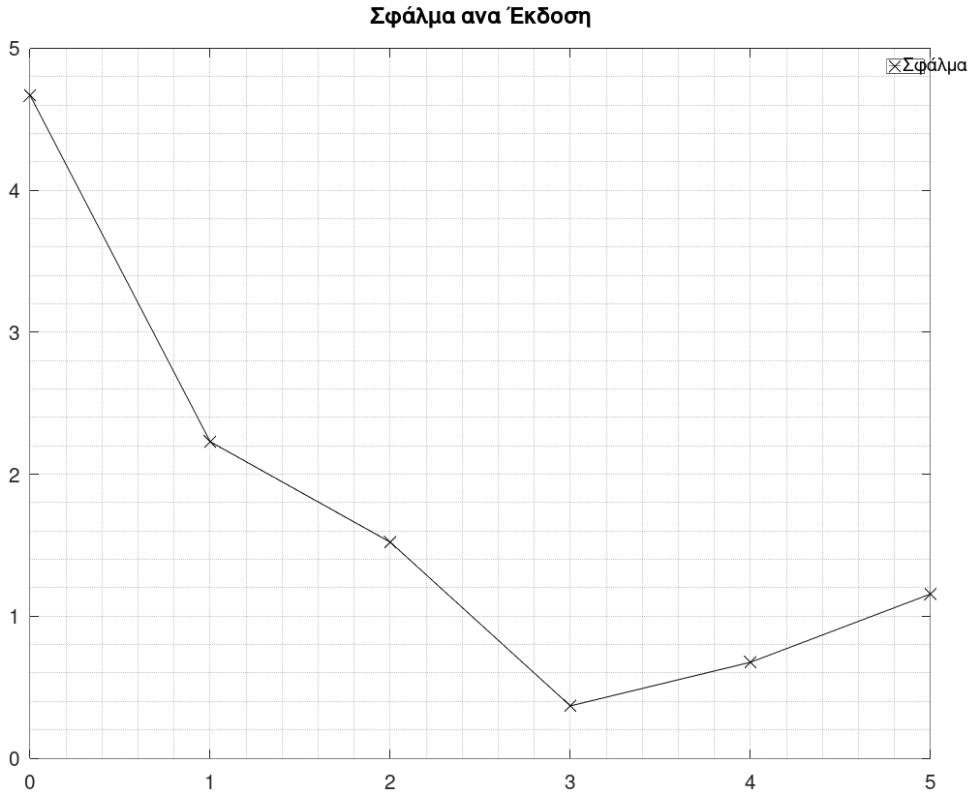
Κεφάλαιο 8 Αποτελέσματα και Βελτιώσεις

8.1 Αποτελέσματα

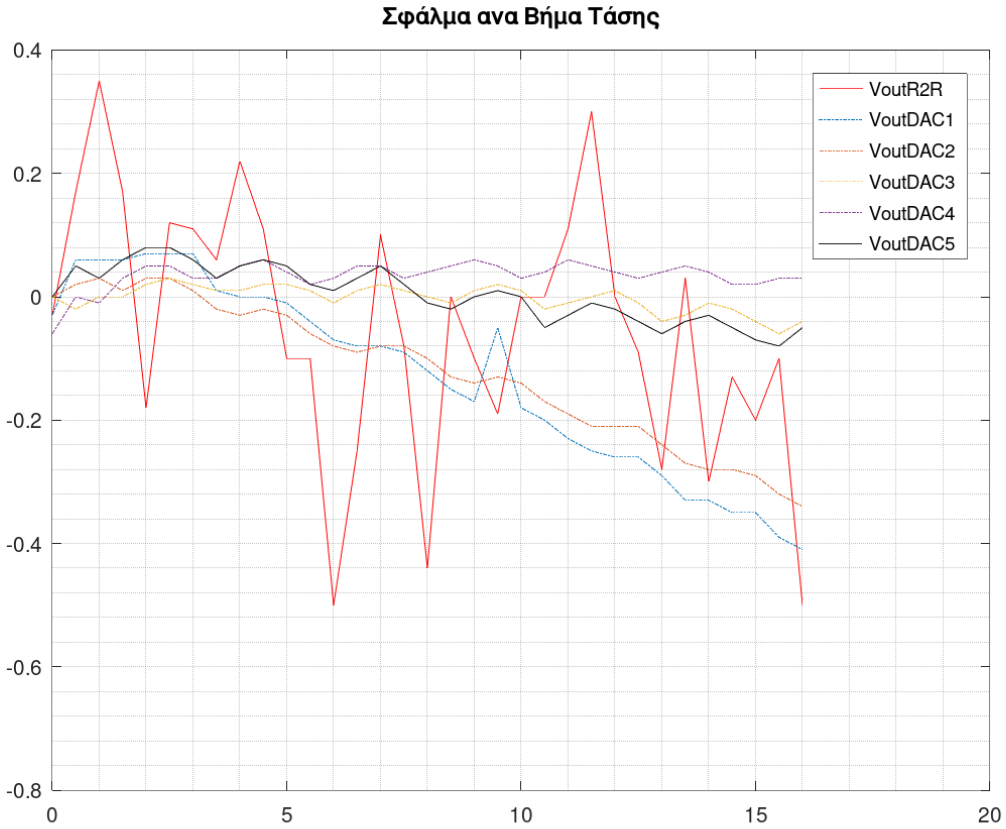
	Vout από keypad , εισαγωγή από πληκτρολόγιο						Vout από H/Y,εισαγωγή από τον H/Y		
	Δυο Τροφοδοτικά Raster				Ένα τροφοδοτικό Raster	Διάτρητη	Δυο Τροφοδοτικά Raster	Ένα τροφοδοτικό Raster	Διάτρητη
Σωστές τιμές Vout	**VoutR2R**	**VOUTDAC v1.0**	**VOUTDAC v2.0**	**VOUTDAC v3.0**	**VOUTDAC v4.0**	**VOUTDAC v5.0**	**VOUTDAC v3.0**	**VOUTDAC v4.0**	**VOUTDAC v5.0**
0	0.03	0.03	0	0	0.06	0	0	0.05	0
0.5	0.33	0.44	0.48	0.52	0.5	0.45	0.52	0.55	0.45
1	0.65	0.94	0.97	1	1.01	0.97	1	1.06	0.97
1.5	1.33	1.44	1.49	1.5	1.47	1.44	1.5	1.53	1.45
2	2.18	1.93	1.97	1.98	1.95	1.92	1.98	2.00	1.93
2.5	2.38	2.43	2.47	2.47	2.45	2.42	2.47	2.5	2.43
3	2.89	2.93	2.99	2.98	2.97	2.94	2.98	3.02	2.95
3.5	3.44	3.49	3.52	3.49	3.47	3.47	3.5	3.53	3.47
4	3.78	4	4.03	3.99	3.95	3.95	4	4.01	3.97
4.5	4.39	4.5	4.52	4.48	4.44	4.44	4.48	4.49	4.45
5	5.1	5.01	5.03	4.98	4.96	4.95	4.98	5.01	4.96
5.5	5.6	5.54	5.56	5.49	5.48	5.48	5.5	5.52	5.49
6	6.5	6.07	6.08	6.01	5.97	5.99	6.02	6.02	6.01
6.5	6.75	6.58	6.59	6.49	6.45	6.47	6.5	6.50	6.49
7	6.9	7.08	7.08	6.98	6.95	6.95	6.98	6.99	6.97
7.5	7.58	7.59	7.58	7.49	7.47	7.48	7.49	7.51	7.5
8	8.44	8.12	8.1	8	7.96	8.01	8	8.02	8.03
8.5	8.5	8.65	8.63	8.51	8.45	8.52	8.52	8.5	8.54
9	9.1	9.17	9.14	8.99	8.94	9.00	9	8.99	9.02
9.5	9.69	9.55	9.63	9.48	9.45	9.49	9.48	9.49	9.52
10	10	10.18	10.14	9.99	9.97	10.0	10	10.0	10.05
10.5	10.5	10.70	10.67	10.52	10.46	10.55	10.52	10.52	10.57
11	10.89	11.23	11.19	11.01	10.94	11.03	11.02	10.99	11.06
11.5	11.2	11.75	11.71	11.5	11.45	11.51	11.5	11.48	11.54
12	12	12.26	12.21	11.99	11.96	12.02	12	12.00	12.05
12.5	12.59	12.76	12.71	12.51	12.47	12.54	12.52	12.52	12.53
13	13.28	13.29	13.24	13.04	12.96	13.06	13.04	13.00	13.09
13.5	13.47	13.83	13.77	13.53	13.45	13.54	13.54	13.50	13.58
14	14.3	14.33	14.28	14.01	13.96	14.03	14.02	14.00	14.01
14.5	14.63	14.85	14.78	14.52	14.48	14.55	14.53	14.51	14.58
15	15.2	15.35	15.29	15.04	14.98	15.07	15.05	15.03	15.06
15.5	15.6	15.89	15.82	15.56	15.47	15.58	15.57	15.52	15.54
16	16.5	16.41	16.34	16.04	15.97	16.05	16.05	16.00	16.06
			Αλλαγή τροφοδοτικού LM358	Χρήση ceil()+12 για την υπολογισμό της εξόδου	Χρήση ενός τροφοδοτικού για όλο το κύκλωμα		Χρήση ceil()+12 για την υπολογισμό της εξόδου	Χρήση ενός τροφοδοτικού για όλο το κύκλωμα	
ΣΦΑΛΜΑ	4.66%	2.22%	1.52%	0.36%	0.67%	1.15%	0.35%	0.7%	1.11%

Εικόνα 8.1: Πίνακας Αποτελεσμάτων Γραμμικού Τροφοδοτικού

Τα αποτελέσματα του γραμμικού τροφοδοτικού είναι πάρα πολύ ικανοποιητικά. Με μέσο όρο σφάλματος 1.01%. Με κάθε version, έκδοση του τροφοδοτικού, είτε αλλάζαμε τον κώδικα είτε το κύκλωμα προσπαθούσαμε να πάρουμε όσο κοντινότερη τιμή εξόδου με αυτή που πληκτρολογούσαμε ή στέλναμε από τον H/Y κάτι που επιτεύχθηκε. Καθώς επίσης πετύχαμε και την επικοινωνία με τον H/Y ώστε να προγραμματίζεται από κάποιο απομακρυσμένο σημείο.



Εικόνα 8.2: Σφάλμα ανά Έκδοση Γραμμικού Τροφοδοτικού



Εικόνα 8.3: Σφάλμα ανά Βήμα Τάσης όλων των Εκδόσεων

Από τα παραπάνω γραφήματα φαίνεται η επίτευξη του ελάχιστου σφάλματος που θέλαμε ώστε να είμαστε πιο κοντά σε αυτό που πληκτρολογούσαμε.

Όπως είχε αναφερθεί, δεν καταφέραμε να πάρουμε αποτελέσματα από το παλμοτροφοδοτικό, αλλά θα αναφερθούν κάποιες προτάσεις γι' αυτό.

8.2 Βελτιώσεις

Αρχικά θα αναφερθούν οι βελτιώσεις που θα μπορούσαν να γίνουν για να καταφέρναμε να υλοποιήσουμε το παλμοτροφοδοτικό:

1) Καλύτερος προγραμματισμός του ελεγκτή PID.
2) Χρήση διαφορετικού οδηγού MOSFET ή χρήση εξωτερικού κυκλώματος ελέγχου.
3) Αντικατάσταση μικροελεγκτή με κάποιον πιο εξειδικευμένο για το συγκεκριμένο κύκλωμα.

Οι βελτιώσεις του γραμμικού τροφοδοτικού είναι:

1) Καλύτερος προγραμματισμός εισαγωγής τιμών τάσης εξόδου από το πληκτρολόγιο και καλύτερος χρονισμός του ADC ή να προγραμματιστεί μέσα στο κύριο πρόγραμμα ώστε να μην τρεμοπαίζει.
2) Χρήση υλικών με καλύτερες ανοχές και τελεστικού που είναι rail to rail, ώστε να έχουμε ένα LM317.
3) Αντικατάσταση του UART-TTL περιφερειακό με τον εσωτερικό USB περιφερειακό, για πιο εύκολη και ταχύτερη σύνδεση.

8.3 Συμπεράσματα

Τα διακοπτικά τροφοδοτικά αν και πιο αποδοτικά και μικρά στο μέγεθος είναι καλύτερα να χρησιμοποιούνται για φορτιστές ή απλά τροφοδοτικά από προγραμματιζόμενα τροφοδοτικά πάγκου. Τα διακοπτικά τροφοδοτικά είναι πολύ εύκολο να αποσταθεροποιούνται και να μας δίνουν άλλο αποτέλεσμα από αυτό που θέλουμε. Αντιθέτως, το γραμμικό είναι πολύ δύσκολο να αποσταθεροποιηθεί μιας και έχει πιο εύκολο έλεγχο από το παλμοτροφοδοτικό δηλαδή χάνουμε από απόδοση και χώρο για να κερδίσουμε σε σταθερότητα. Την διαφορά μεγέθους δυστυχώς δεν μπορέσαμε να την δείξουμε γιατί τα σχεδιάσαμε και τα δυο με μικρό μέγιστο ρεύμα εξόδου.

Άρα, το γραμμικό τροφοδοτικό είναι εύκολο στη σχεδίαση και υλοποίηση με μικρή απόδοση και σταθερό λόγω του εύκολου ελέγχου, για αυτό και είναι πιο διαδεδομένο στα εργαστήρια.

Βιβλιογραφία

[1] Control Systems for Power Electronics : A Practical Guide Mahesh Patil Pankaj Rodey

[2] FPGA Implementation of PID Controller for the Stabilization of a DC-DC “Buck” Converter Eric William Zurita-Bustamante, Jesús Linares-Flores, Enrique Guzmán-Ramírez2 and Hebertt Sira-Ramírez

[3] Modern Contro Engineering Fifth Edition Katsuhiko Ogata

[4] SWITCHMODE POWER SUPPLY HANDBOOK Keith Billings Taylor Morey

[5] Power Supply Cookbook Second Edition Marty Brown

[6] Σύγχρονη Θεωρία Ελέγχου Αναστάσιος Πουλιέζος

[7] Buck Converter Design Jens Ejury Infineon Technologies North America (IFNA) Corp.Design Note DN 2013-01 V0.1 January 2013

[7] SLIDING MODE-DELTA MODULATION GPI CONTROL OF A DC MOTOR THROUGH A BUCK CONVERTER J. Linares-Flores and H. Sira-Rrunfrez'

[8] Mathematical Modelling of Buck Converter Rajvir Kaur Navdeep Kaur

[9] Robust Passivity Based Control of a Buck–Boost–Converter/DC–Motor System: An Active Disturbance Rejection Approach . Linares-Flores, Member, IEEE, J. L. Barahona-Avalos, H. Sira-Ramírez, Senior Member, IEEE, and M. A. Contreras-Ordaz, Member, IEEE

[10] Modern Control Systems ELEVENTH EDITION Richard C. Dorf U, Davis Robert H. Bishop

[11] A Digital DC Power Supply (programmable bench power supply unit), hardware version 3.0

[12] Ιορδάνης Κιοσκερίδης Ανανεώσιμες Πηγές Ενέργειας και εφαρμογές των Ηλεκτρονικών

[13] Ιορδάνης Κιοσκερίδης Ηλεκτρονικά Ισχύος

Παράρτημα Α Κώδικες

Κώδικας εύρεσης τιμών υλικών και ευστάθειας παλμοτροφοδοτικού

```
pkg load control
pkg load symbolic
pkg load gnuplot
pkg load signal
%Εισαγωγή πακέτων
clear
clc

%buck converter
Iomax=2;  %Μέγιστο ρεύμα εξόδου
Vripple=20e-3; %Ripple τάσης εξόδου
Iripple=200e-3;  %Ripple ρεύματος εξόδου
Vd=19;  %Τόση εισόδου
Vout=8;  %Τάση εξόδου μέγιστη~18
Fs=21e3;  %Συχνότητα λειτουργίας παλμών
Ts=1/Fs;  %Περίοδος παλμών
R=Vout/Iomax  %Αντίσταση εξόδου
D=Vout/Vd;  %Duty Cicle σε δευτερόλεπτα
DC=D*100;  %Duty Cicle ποσοστό επί της εκατό
Pulse_width = D*Ts;  %Εύρος παλμών
ul=Vout;
L=(ul*(1-D)*Ts)/Iripple  % Μέγεθος πηνίου
Ilim=(Vout*(1-D)*Ts)/(2*L)  %Κατώτατο όριο ρεύματος
Rc=Vripple/Iripple  %Effective Series Resistance
C=(65e-6)/Rc  %Μέγεθος πυκνωτή
##############################################################################
s = tf ('s');
 HP=(Vd/(L*C))/((s^2)+(1/(R*C))*s+(1/(L*C))) % Συνάρτηση μεταφοράς κυκλώματος
 bool = isstable(HP)  %Εντολή για έλεγχο  ευστάθειας κυκλώματος
 figure(1)
 pzmap(HP)
 title('Χάρτης πόλων και μηδενικόν συστήματος χωρίς PID')
 [pHP, zHP]=pzmap(HP);
 figure(2)
 step(HP)
 title('Βηματική απόκριση συστήματος χωρίς PID')
##############################################################################

 z=(sqrt(L/C))/(2*R) %Βαθμός απόσβεσης
 wn=1/(2*pi*sqrt(L*C)) % Φυσική συχνότητα σε  hz
##############################################################################
 a=1/(L*C) %Μεταβλητή α  προσεγγιστικά
 Kp=((2*z*wn*a*L*C+(wn^2)*L*C-1)/Vd) %Μεταβλητή του P
 Ti=(Vd*Kp)/(L*C*a*(wn^2)) %Μεταβλητή του I
 Td=((L*C)/(Vd*Kp))*(a+2*z*wn-(1/R*C)) %Μεταβλητή του D
 Ki=Kp/Ti %Μεταβλητή του I
 Kd=Kp*Td %Μεταβλητή του D
##############################################################################
 P = Kp*(1+((1/(Ti*s)))+(Td*s)) % Συνάρτηση μεταφοράς PID
 HOL=series(HP,P) %Εντολή που συνδέει δυο συστήματα σε σειρά
 CL=feedback(HOL) %Εντολή που συνδέει δυο συστήματα με ανάδραση
   bool2 = isstable(CL) %Εντολή για έλεγχο  ευστάθειας κυκλώματος με PID
```

```
figure(3)
pzmap(CL)
title('Χάρτης πόλων και μηδενικόν συστήματος με PID')
[pCL,zCL]=pzmap(CL);
figure(4)
bode(CL,{10e-2,10e14})
[mag, pha] =bode(CL,{10e-1,10e170}); %εντολη για τη αποθυκευση τον τιμων του Bode
title('Διάγραμμα Bode συστήματος με PID')
figure(5)
step(CL)
title('Βηματική απόκριση συστήματος με PID')
figure(6)
nyquist(CL)
title('Διάγραμμα nyquist συστήματος με PID')
figure(7)
bode(CL,{10e95,10e110})
title('Διάγραμμα Bode συστήματος με PID απο 10e95,10e110 ')
```

Κώδικας παλμοτροφοδοτικού

```
#include <main.h> //Δήλωση Βιβλιοθηκών
#include <flex_lcd.h>
#include <keypad.h>
#include <STRING.H>
#include <stdlib.h>

#USE rs232(ERRORS,baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,stop=1,bits=8) //Δήλωση
rs232
//PORT DEFINE
//Αρχικοποίηση PORT(πυλών),SFR(Special Function Registers)
#use standard_io ( A )
#use standard_io ( B )
#use standard_io ( C )

#byte PORTA        =0xF80
#byte PORTB        =0xF81
#byte PORTC        =0xF82
#byte PORTD        =0xF83
#byte PORTE        =0xF84

#byte ECCP1DEL      =0xFB7 //SFR για το deadtime

//Gloabal Variables Δήλωση μεταβλητών
int16 Vadc; //Μεταβλητές για μέτρηση τάσης και ρεύματος
int16 Iadc;

float Vout; //Μεταβλητή τάσης από το πληκτρολογίο
float Vref; //Μεταβλητή  καταχώρισης τάσης αναφοράς

float Kp=3.24; //Proportional αναλογικός
float Ki=1.035e1; //Integral ολοκληρωτής
float Kd=6.64e-3; //Derivative διαφορικός

int DCON=0; //Μεταβλητή duty cycle
signed int16 p; // Μεταβλητή καταχώρισης αποτελέσματος συνάρτησης PID
int Duty=512/8; // Μεταβλητή που καταχωρεί αρχική τιμή PWM
```

```
char state=1; //Μεταβλητή επιλογής κατάστασης από την switch
int8 N1=0; //Μεταβλητή δεκάδων
int8 N2=0; //Μεταβλητή μονάδων
float N3=0; //Μεταβλητή αποθήκευσης αποτελέσματος και δεκαδικού ψηφίου
char k; //Μεταβλητή αποθήκευσης χαρακτήρα ASCII

//Δήλωση  εντολών  από Η/Υσαν πινάκες χαρακτήρων char
char *command;
char temp[9];
char command1[5]={"VOUT"};
char command2[5]={"VINF"};
char command3[4]={"ON"};
char command4[4]={"OFF"};
char command5[5]={"INFO"};
char command6[5]={"IINF"};

int32 result;
//Δήλωση συναρτήσεων,οι συναρτήσεις που χρησιμοποιούνται από διακοπές δεν χρειάζεται να δηλωθούν
void init(void);
void set_deadband(int value);
signed int16 PID(float Kp,float Ki,float kd,float Vref);
int compstr(char a[],char b[]);

void main(void)
{
        init();
        lcd_init();
        lcd_gotoxy(1,1);
        lcd_putc("\f");
        printf(lcd_putc,"Initializing...");
        int i=0;
        for(i=0;i<100;i++)
                {
                        delay_ms(15);
                        lcd_gotoxy((int)i/6,2);
                        printf(lcd_putc,"*",);
                }

        lcd_putc("\f");
        setup_ccp1(CCP_PWM_HALF_BRIDGE|CCP_PWM_H_H);
        set_deadband(4);
        enable_interrupts(global);
        enable_interrupts(INT_TIMER0);

        while(1)
        {
                enable_interrupts(int_rda);
                switch(state)
                                {
                        case 1:
                                k=kbd_getc();
                                while (k!=0)
                                {
                                        lcd_gotoxy(1,2);
                                        printf(lcd_putc,"      ");
                                        N1= k&0b00001111;
```

```
                                        lcd_gotoxy(1,2);
                                        printf(lcd_putc,"%d",N1);
                                        state = 2;
                                        break;
                                }
                        break;
                            case 2:
                            k=kbd_getc();
                             while (k!=0)
                             {
                                        N2= k&0b00001111;
                                       lcd_gotoxy(2,2);
                                       printf(lcd_putc,"%d",N2);
                                        state = 3;
                                        break;
                             }
                          break;
                          case 3:
                            k=kbd_getc();
                            while (k!=0)
                            {
                                        N3 = k&0b00001111;
                                         lcd_gotoxy(3,2);
                                          printf(lcd_putc,"%Ld",N3);
                                         lcd_putc("\f");
                                         N3 = N1*10+N2+N3/10;
                                         if(N3>17)
                                         {
                                                lcd_gotoxy(1,2);
                                                 printf(lcd_putc,"Out of bounds");
                                                delay_ms(1000);
                                                lcd_gotoxy(1,2);
                                                 printf(lcd_putc,"               ");
                                        }
                                         else
                                         {
                                                lcd_gotoxy(1,2);
                                                 printf (lcd_putc,"%3.2fV",N3);
                                                 Vout=N3;
                                                 Vref=Vout*0.1757;//Πολλαπλασιασμός με τον λόγο του
                                                                  διαιρέτη τάσης
                                         }
                          N1=0;
                          N2=0;
                          N3=0;
                          state = 1;
                          break;
                           }

                             }

                    }
}

void init(void)
{
        setup_oscillator(OSC_8MHZ);
```

```
	setup_adc_ports( AN0_TO_AN1 | VSS_VDD  );
	setup_adc ( ADC_CLOCK_INTERNAL );

	setup_timer_2(T2_DIV_BY_1,30,1);

	setup_timer_0(T0_INTERNAL | T0_DIV_16);
	set_timer0(70);
}
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
void set_deadband(int value)
{
	int temp;

	value &= 0X7F;
	temp = ECCP1DEL;
	temp &= 0X80;
	temp |= value;
	ECCP1DEL = temp;
}

signed int16 PID(float Kp,float Ki,float kd,float Vref)
{
	int16 vout;
	int16 er;
	float prev_er;
	float inter_er_prev;

	float proportional;
	float integral;
	float derivative;

	signed int16 pid;
	byte adc_channel=0;

	if(adc_channel==0)
		{
			set_adc_channel ( 0 );
			delay_us(50);
			Vadc=read_adc();
			delay_us(50);

			adc_channel=1;
		}
	if(adc_channel==1)
		{
			set_adc_channel ( 1 );
			delay_us(50);
			Iadc=read_adc();
			delay_us(50);

			adc_channel=0;
		}
	if(Iadc*4.88e-3>1)
		{
```

```
                vout=5.7*Iadc*4.88e-3 ;//5.7=Feedack resistor ratio
        }
    else
        {
                vout=Vadc;
        }

    er = Vref-vout;

    proportional=Kp*er;
    integral=inter_er_prev+er*3e-6;
    derivative=(er-prev_er)/3e-6;

    prev_er=er;
    inter_er_prev=integral;

    pid=proportional+integral*Ki+derivative*Kd;

    return(pid/8);

}


#INT_TIMER0
void timer0_int(void)
{
    set_pwm1_duty(Duty);

    p=PID(Kp,Ki,Kd,Vref);

    Duty=Duty+p;


    if(p==0)
    {
        DCON=Duty;
        set_pwm1_duty(DCON);
    }

    lcd_gotoxy(1,2);
    printf(lcd_putc,"%fV",5.7*Vadc*4.88e-3);

    lcd_gotoxy(8,2);
    printf(lcd_putc,"%1.4fA",Iadc*4.88e-3);

}

int compstr(char a[],char b[])
{
    int i;
    int counter=0;
    int length=0;
    length = strlen(b);

    for(i=0;i<length;i++)
    {
        if(a[i]==b[i])
```

```
                {
                        counter++;
                }
        }

        if(counter==length)
        {
                return 1;
        }
        else
        {
                return 0;
        }
}

//////////////////////////////////////////////////////
#int_rda
void serial_interrupt()
{
        disable_interrupts(int_rda);
        gets(temp);
        result=strtol(temp,&command,10);


        if(compstr(command,command1))
        {
                if(result>0&&result<=18)
                {
                        printf("VOUT OK");
                        Vout=result;
                        Vref=Vout*0.1757;
                        putc('\n');
                        putc('\r');
                }
                else
                {
                        printf("VOUT NOT OK");
                        putc('\n');
                        putc('\r');
                }

        }

        if(compstr(command,command2))
        {
                set_adc_channel ( 0 );
                delay_us(50);
                printf("%f",(read_adc()*4.88759e-3)/(0.1757));
                delay_us(50);
                putc('\n');
                putc('\r');

        }

        if(compstr(command,command3))
        {
                printf("OUT ON");
```

```
            set_pwm1_duty(DCON);
            putc('\n');
            putc('\r');
        }

        if(compstr(command,command4))
        {
            printf("OUT OFF");
            set_pwm1_duty(0);
            putc('\n');
            putc('\r');
        }

        if(compstr(command,command5))
        {

            printf("Programmable Power Supply Ver:1.00");
            putc('\n');
            putc('\r');

        }

        if(compstr(command,command6))
        {
            set_adc_channel ( 1 );
            delay_us(50);
            printf("%f",(read_adc()*4.88759e-3));
            delay_us(50);
            putc('\n');
            putc('\r');

        }

        if(!compstr(command,command1)&&!compstr(command,command2)
            &&!compstr(command,command3)&&!compstr(command,command4)
                &&!compstr(command,command5)&&!compstr(command,command6))
        {
            printf("EMPTY COMMAND OR WRONG COMMAND");
            putc('\n');
            putc('\r');
        }
}
```

Κώδικας εύρεσης σφάλματος και γραφημάτων σφάλματος γραμμικού τροφοδοτικού

```
pkg load gnuplot

clf;

VoutRight=[0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5,10,10.5,11,11.5,12,12.5,13,13.
5,14,14.5,15,15.5,16];
VoutR2R=[0.03,0.33,0.65,1.33,2.18,2.38,2.89,3.44,3.78,4.39,5.1,5.6,6.5,6.75,6.9,7.58,8.44,8.5,9.1,
9.69,10,10.5,10.89,11.2,12,12.59,13.28,13.47,14.3,14.63,15.2,15.6,16.5];
VoutDAC1=[0.03,0.44,0.94,1.44,1.93,2.43,2.93,3.49,4,4.5,5.01,5.54,6.07,6.58,7.08,7.59,8.12,8.65,
9.17,9.55,10.18,10.70,11.23,11.75,12.26,12.76,13.29,13.83,14.33,14.85,15.35,15.89,16.41];
```

```
VoutDAC2=[0,0.48,0.97,1.49,1.97,2.47,2.99,3.52,4.03,4.52,5.03,5.56,6.08,6.59,7.08,7.58,8.1,8.63,
9.14,9.63,10.14,10.67,11.19,11.71,12.21,12.71,13.24,13.77,14.28,14.78,15.29,15.82,16.34];
VoutDAC3=[0,0.52,1,1.5,1.98,2.47,2.98,3.49,3.99,4.48,4.98,5.49,6.01,6.49,6.98,7.49,8,8.51,8.99,9
.48,9.99,10.52,11.01,11.5,11.99,12.51,13.04,13.53,14.01,14.52,15.04,15.56,16.04];
VoutDAC3Key=[0,0.52,1,1.5,1.98,2.47,2.98,3.5,4,4.48,4.98,5.5,6.02,6.5,6.98,7.49,8,8.52,9,9.48,10
,10.52,11.02,11.5,12,12.52,13.04,13.54,14.02,14.53,15.05,15.57,16.05];#apo pc
VoutDAC4=[0.06,0.5,1.01,1.47,1.95,2.45,2.97,3.47,3.95,4.44,4.96,5.48,5.97,6.45,6.95,7.47,7.96,8.
45,8.94,9.45,9.97,10.46,10.94,11.45,11.96,12.47,12.96,13.45,13.96,14.48,14.98,15.47,15.97];
VoutDAC4Key=[0.05,0.55,1.06,1.53,2.00,2.5,3.02,3.53,4.01,4.49,5.01,5.52,6.02,6.50,6.99,7.51,8.0
2,8.5,8.99,9.49,10.0,10.52,10.99,11.48,12.00,12.52,13.00,13.50,14.00,14.51,15.03,15.52,16.00];#
apo pc
VoutDAC5=[0,0.45,0.97,1.44,1.92,2.42,2.94,3.47,3.95,4.44,4.95,5.48,5.99,6.47,6.95,7.48,8.01,8.5
2,9.00,9.49,10.0,10.55,11.03,11.51,12.02,12.54,13.06,13.54,14.03,14.55,15.07,15.58,16.05];
VoutDAC5Key=[0,0.45,0.97,1.45,1.93,2.43,2.95,3.47,3.97,4.45,4.96,5.49,6.01,6.49,6.97,7.5,8.03,8
.54,9.02,9.52,10.05,10.57,11.06,11.54,12.05,12.53,13.09,13.58,14.01,14.58,15.06,15.54,16.06];#a
po pc

l=length(VoutRight);
##t=0:1:l-1;

Verror1=VoutRight-VoutR2R;
Verror2=VoutRight-VoutDAC1;
Verror3=VoutRight-VoutDAC2;
Verror4=VoutRight-VoutDAC3;
Verror5=VoutRight-VoutDAC3Key;#apo pc
Verror6=VoutRight-VoutDAC4;
Verror62=VoutRight-VoutDAC4Key;#apo pc
Verror7=VoutRight-VoutDAC5;
Verror72=VoutRight-VoutDAC5Key;#apo pc


Telikoerror=0;
for i= 2:1:l
Relative_error(i)=Verror72(i)/VoutRight(i);#epilogi error
endfor
Relative_error_perc=abs(Relative_error*100);
for i= 2:1:l
Telikoerror=Telikoerror+Relative_error_perc(i);
endfor
Telikoerror_mean=Telikoerror/32




figure(1)
plot(VoutRight,VoutRight,'-',VoutR2R,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutR2R");
#####################################
figure(2)
hold on
errorbar(VoutRight,VoutR2R,Verror1,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
```

```
title("Errorbar Σωστή Τάση Vs VoutR2R");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(3)
plot(VoutRight,VoutRight,'-',VoutDAC1,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC1");
#####################################
figure(4)
hold on
errorbar(VoutRight,VoutDAC1,Verror2,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
title("Errorbar Σωστή Τάση Vs VoutDAC1");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(5)
plot(VoutRight,VoutRight,'-',VoutDAC2,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC2");
#####################################
figure(6)
hold on
errorbar(VoutRight,VoutDAC2,Verror3,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
title("Errorbar Σωστή Τάση Vs VoutDAC2");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(7)
plot(VoutRight,VoutRight,'-',VoutDAC3,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC3");
#####################################
figure(8)
hold on
```

```
errorbar(VoutRight,VoutDAC3,Verror4,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης Keypad");
title("Errorbar Σωστή Τάση Vs VoutDAC3");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(9)
plot(VoutRight,VoutRight,'-',VoutDAC3Key,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC3Key");
#####################################
figure(10)
hold on
errorbar(VoutRight,VoutDAC3Key,Verror5,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
title("Errorbar Σωστή Τάση Vs VoutDAC3Key");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(11)
plot(VoutRight,VoutRight,'-',VoutDAC4,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC4");
#####################################
figure(12)
hold on
errorbar(VoutRight,VoutDAC4,Verror6,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
title("Errorbar Σωστή Τάση Vs VoutDAC4");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#####################################
#####################################
figure(13)
plot(VoutRight,VoutRight,'-',VoutDAC5,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC5");
```

```
#######################################
figure(14)
hold on
errorbar(VoutRight,VoutDAC5,Verror7,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
title("Errorbar Σωστή Τάση Vs VoutDAC5");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#######################################
#######################################
figure(15)
plot(VoutRight,VoutRight,'-',VoutDAC4Key,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC4Key");
#######################################
figure(16)
hold on
errorbar(VoutRight,VoutDAC4Key,Verror62,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
title("Errorbar Σωστή Τάση Vs VoutDAC4Key");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#######################################
#######################################
figure(17)
plot(VoutRight,VoutRight,'-',VoutDAC5Key,VoutRight,'-');
grid minor on
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
legend("Σωστή Τάση", "Τάση Εξόδου Vout");
title("Σωστή Τάση Vs VoutDAC5Key");
#######################################
figure(18)
hold on
errorbar(VoutRight,VoutDAC4Key,Verror72,'-');
xlabel("Τάση Εξόδου");
ylabel("Τιμή Τάσης H/Y");
title("Errorbar Σωστή Τάση Vs VoutDAC5Key");
plot(VoutRight,VoutRight,'-');
grid minor on
legend("Σφάλμα Τάσης Εξόδου Vout","Σωστή Τάση");
hold off
#######################################
#######################################
```

Κώδικας Γραμμικού Τροφοδοτικού

```
#include <main.h>
#include <flex_lcd.h>
#include <keypad.h>
#include <MATH.h>
#include <STRING.H>
#include <stdlib.h>


#USE
rs232(ERRORS,baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,stop=1,bits=8,TIMEOUT=250)

#USE I2C(MASTER,scl=PIN_D0,sda=PIN_D1,FORCE_SW,FAST=400000)

//PORT DEFINE
#use standard_io ( A )
#use standard_io ( B )
#use standard_io ( C )

#byte PORTA        =0xF80
#byte PORTB        =0xF81
#byte PORTC        =0xF82
#byte PORTD        =0xF83
#byte PORTE        =0xF84



//Gloabal Variables
unsigned long Vadc[6];
unsigned long Vadc2[6];
unsigned long Vadc_mean;
unsigned long Vadc2_mean;
unsigned short adc_channel_select=0;


float VCC=5.13;
float step10bit=VCC/1023;
float step12bit=VCC/4095;
float Voltage_Div=3.99;

int N1=0;
int N2=0;
float N3=0;
unsigned long N4=0;
float N4_temp;
char state=1;

char *command;
char temp[9];

char command1[5]={"VOUT"};
char command2[5]={"VINF"};
char command3[4]={"ON"};
```

```
char command4[4]={"OFF"};
char command5[4]={"INF"};
char command6[5]={"IINF"};
int COUNTER=0;
float result;
unsigned long rda_N4=0;
int PORTD_temp=0;

void init(void);
void dac_int(void);
int compstr(char a[],char b[]);
void serial_SENT(void);
void dac_i2c_write(unsigned int16 value);

void main(void)
{

	init();

	lcd_init();

	kbd_init();

	dac_int();
	lcd_gotoxy(1,1);
	lcd_putc("\f");
	printf(lcd_putc,"Initializing...");
	unsigned int i=0;
	for(i=0;i<100;i++)
		{
			delay_ms(5);
			lcd_gotoxy((int)i/5,2);
			printf(lcd_putc,"*",);
		}

	lcd_putc("\f");
	enable_interrupts(GLOBAL);
	enable_interrupts(INT_TIMER0);

	char k;
	while(1)
	{
		enable_interrupts(int_rda);
		switch(state)
		{

			case 1:
				k=kbd_getc();

				while (k!=0)
				{
					lcd_gotoxy(1,1);
					printf(lcd_putc,"      ");
					N1= k&0b00001111;
					lcd_gotoxy(1,1);
					printf(lcd_putc,"%d",N1);
					state = 2;
```

```
                    break;
                }
                break;
                case 2:
                    k=kbd_getc();
                    while (k!=0)
                    {
                        N2= k&0b00001111;
                        lcd_gotoxy(2,1);
                        printf(lcd_putc,"%d",N2);
                        state = 3;
                        break;
                    }
                    break;
                case 3:
                    k=kbd_getc();
                    while (k!=0)
                    {
                        N3 = k&0b00001111;
                        lcd_gotoxy(3,1);
                        printf(lcd_putc,"%Ld",N3);
                        lcd_putc("\f");
                        N3 =(N1*10+N2+N3/10);
                        if(N3>19)
                        {
                            lcd_gotoxy(1,1);
                            printf(lcd_putc,"Voltage<=17");
                            delay_ms(300);
                            lcd_gotoxy(1,1);
                            printf(lcd_putc,"              ");
                        }
                        else
                        {
                            lcd_gotoxy(1,1);
                            printf (lcd_putc,"%3.2fV",N3);

                            N4_temp=ceil(((N3-
0.07)/Voltage_Div)/step12bit);

                            if(N4_temp<0)
                            {
                                N4_temp=0;
                            }
                            N4=N4_temp;
                            dac_i2c_write(N4);
                        }

                        N1=0;
                        N2=0;
                        N3=0;
                        N4=0;
                        state = 1;
                        break;
                    }

            }

    }
```

```
}
////////////////////////////////////////////////////
////////////////////////////////////////////////////
void init(void)
{
	setup_oscillator(OSC_8MHZ);

	setup_adc_ports( AN0_TO_AN1 | VSS_VDD  );
	setup_adc ( ADC_CLOCK_DIV_64 );

	setup_timer_0(T0_INTERNAL | T0_DIV_16);
	set_timer0(65536);

}
////////////////////////////////////////////////////
////////////////////////////////////////////////////
void dac_int(void)
{
	i2c_start();
	i2c_write(0xC0); //Device Address
	i2c_write(0b1000000); //Internal Device Address
	i2c_write((0x00 & 0xFF0) >> 4); //D11 to D4
	i2c_write((0x00 & 0xF) << 4); // D3 to D0
	i2c_stop();
	delay_us(20);
}
////////////////////////////////////////////////////
////////////////////////////////////////////////////
void dac_i2c_write(unsigned int16 value)
{
	i2c_start();
	i2c_write(0xC0); //Device Address
	i2c_write(0b1000000); //Internal Device Address
	i2c_write((value & 0xFF0) >> 4); //D11 to D4
	i2c_write((value & 0xF) << 4); // D3 to D0
	i2c_stop();
}
////////////////////////////////////////////////////
////////////////////////////////////////////////////

int compstr(char a[],char b[])
{
	int i;
	int counter=0;
	int length=0;
	length = strlen(b);

	for(i=0;i<length;i++)
	{
		if(a[i]==b[i])
		{
			counter++;
		}
	}

	if(counter==length)
```

```
	{
		return 1;
	}
	else
	{
		return 0;
	}
}

//////////////////////////////////////////////////////
//////////////////////////////////////////////////////

#INT_TIMER0
void timer0_int(void)
{
	int i;
	if(adc_channel_select==0)
	{
		set_adc_channel(0);
		delay_us(50);
		for(i=0;i<6;i++)
		{
			Vadc[i]=read_adc();
			delay_us(50);
		}
		adc_channel_select=1;
	}

	if(adc_channel_select==1)
	{
		set_adc_channel(1);
		delay_us(50);
		for(i=0;i<6;i++)
		{
			Vadc2[i]=read_adc();
			delay_us(50);
		}
		adc_channel_select=0;
	}
	Vadc_mean=(Vadc[0]+Vadc[1]+Vadc[2]+Vadc[3]+Vadc[4]+Vadc[5])/6;

	Vadc2_mean=(Vadc2[0]+Vadc2[1]+Vadc2[2]+Vadc2[3]+Vadc2[4]+Vadc2[5])/6;



	lcd_gotoxy(1,2);
	printf(lcd_putc,"%1.3fA",Vadc_mean*4.88e-3);
	lcd_gotoxy(8,2);
	printf(lcd_putc,"%1.1fV",Vadc2_mean*step10bit*3.9);

	if(Vadc>204)
	{
		dac_i2c_write((Vadc2_mean*4)/5);
	}
}
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
```

```
#INT_RDA
void serial_interrupt(void)
{

        disable_interrupts(int_RDA);

        gets(temp);

        serial_SENT();

}

//////////////////////////////////////////////////////
//////////////////////////////////////////////////////

void serial_SENT(void)
{
        disable_interrupts(int_RDA);
        result=strtol(temp,&command,10);

        if(compstr(command,command1))
        {
                if(result>=0&&result<=170)
                {
                        printf("VOUT OK");
                        rda_N4=ceil((((result)/10)/Voltage_Div)/step12bit);
                        putc('\n');
                        putc('\r');
                        printf("%Ld",rda_N4);
                        putc('\n');
                        putc('\r');
                        dac_i2c_write(rda_N4);
                        lcd_gotoxy(1,1);
                        printf (lcd_putc,"%3.2fV",result/10);

                }
                else
                {
                        printf("VOUT NOT OK");
                        putc('\n');
                        putc('\r');
                }
        }

        else if(compstr(command,command2))
        {

                printf("%f",Vadc2_mean*4.88e-3*3.9);
                putc('\n');
                putc('\r');
        }

        else if(compstr(command,command3))
        {

                printf("OUT ON");
```

```
            dac_i2c_write(PORTD_temp);
            putc('\n');
            putc('\r');
        }

        else if(compstr(command,command4))
        {

            printf("OUT OFF");
            PORTD_temp=rda_N4;
            dac_i2c_write(0);
            putc('\n');
            putc('\r');
        }

        else if(compstr(command,command5))
        {

            printf("Programmable Lab");
            putc('\n');
            putc('\r');
            printf("Bench Power Supply Ver 1.00");
            putc('\n');
            putc('\r');

        }
        else if(compstr(command,command6))
        {
            printf("%f",Vadc_mean*4.88e-3);
            putc('\n');
            putc('\r');
        }

        if(!compstr(command,command1)&&!compstr(command,command2)
            &&!compstr(command,command3)&&!compstr(command,command4)
                &&!compstr(command,command5)&&!compstr(command,command6))
        {
            printf("EMPTY COMMAND");
            putc('\n');
            putc('\r');
            printf("OR WRONG COMMAND");
            putc('\n');
            putc('\r');
        }

}
```