ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΑΝΑΠΤΥΞΗ ΣΥΣΚΕΥΗΣ ΨΗΦΙΑΚΗΣ ΚΑΤΑΓΡΑΦΗΣ ΦΩΝΗΣ, ΜΕ ΜΙΚΡΟΕΛΕΓΚΤΗ

Πτυχιακή εργασία του

Αγγελόπουλου Ζαφείριου

Α.Ε.Μ. : 3044

Επιβλέπων : Δρ. Καλόμοιρος Ιωάννης, Αναπληρωτής Καθηγητής

ΣΕΡΡΕΣ, ΜΑΡΤΙΟΣ 2017

Υπεύθυνη Δήλωση

Υπεύθυνη δήλωση : Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του τμήματος Μηχανικών Πληροφορικής Τ.Ε. του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Ευχαριστίες

Αρχικά ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου Δρ. Καλόμοιρο Ιωάννη, του οποίου η αφοσίωση και το πάθος για την επιστήμη αυτή, μου άνοιξε τις πύλες ενός κόσμου στον οποίο βάδιζα στις μύτες των ποδιών μου.

Έπειτα ευχαριστώ τους φίλους μου Στάθη, Δημήτρη και Γιώργο οι οποίοι με τις γνώσεις τους και την ψυχολογική τους συμπαράσταση με οδήγησαν στο επιθυμητό αποτέλεσμα.

Τέλος θα ήθελα να αφιερώσω την πτυχιακή αυτή εργασία στους γονείς μου Άγγελο και Τερέσα οι οποίοι, με την συνεχή υποστήριξή τους και την άνευ όρων αγάπη τους, είναι οι σύμβουλοι της καθημερινής μου ζωής και της σταδιοδρομίας μου.

Περιεχόμενα

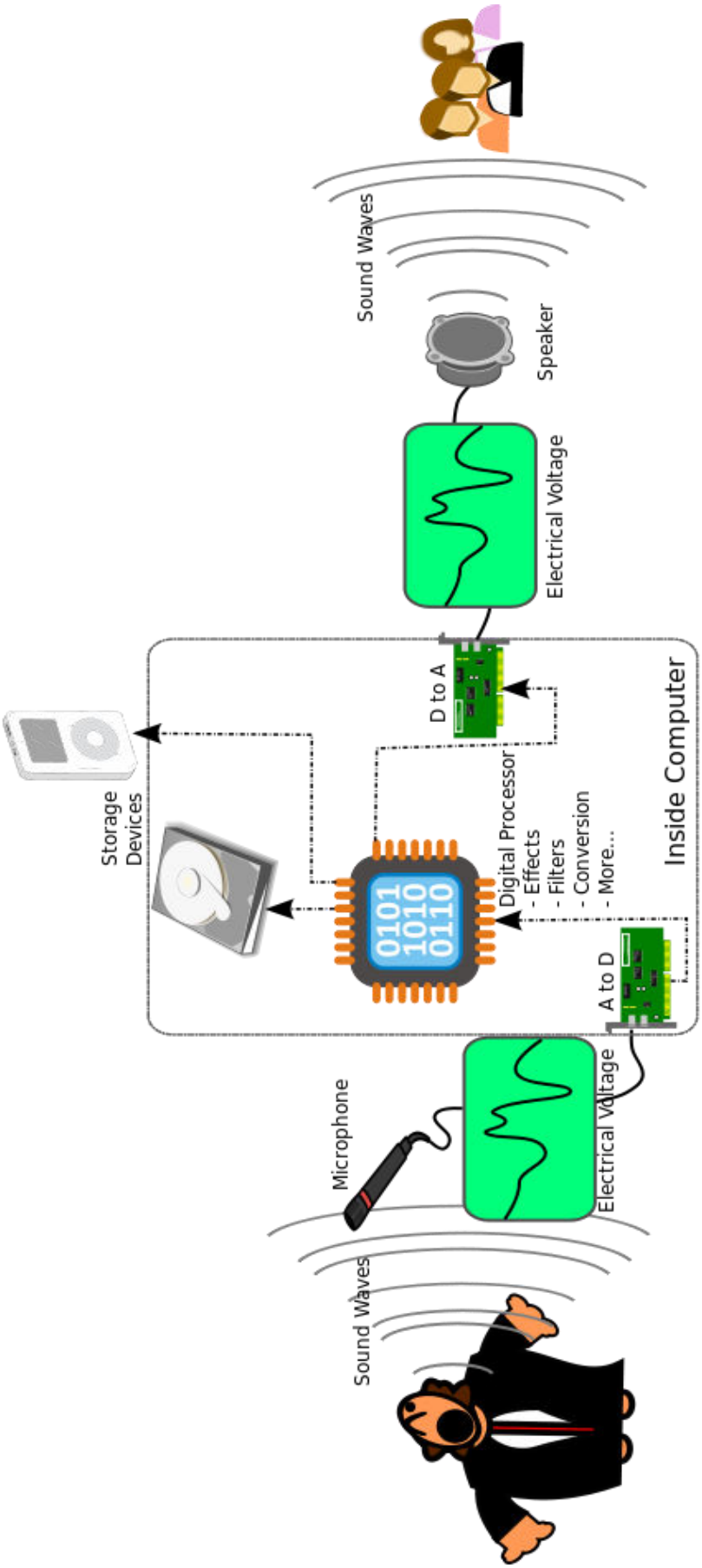
Περίληψη

Στην παρούσα πτυχιακή εργασία περιγράφεται η σχεδίαση και υλοποίηση μιας συσκευής ψηφιακής καταγραφής ήχου. Για την υλοποίησή της χρησιμοποιήθηκε ο μικροελεγκτής PIC16F877 της οικογένειας μικροελεγκτών PIC της εταιρείας Microchip Technology. Οι επιμέρους βαθμίδες είναι η δειγματοληψία του ήχου, η αποθήκευσή του και η αναπαραγωγή του. Η δειγματοληψία επιτυγχάνεται μέσω πυκνωτικού μικροφώνου και μετατροπή του ήχου σε ψηφιακή μορφή, κάνοντας χρήση του μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC – Analog to Digital Converter) που βρίσκεται μέσα στον μικροελεγκτή. Μετέπειτα μέσω του σειριακού πρωτοκόλλου επικοινωνίας σύγχρονης μετάδοσης δεδομένων (SPI – Serial Peripheral Interface), αποθηκεύουμε τα δεδομένα σε μία κάρτα μνήμης τύπου flash (SD - Secure Digital). Με την ολοκλήρωση της αποθήκευσης περνάμε στο στάδιο της αναπαραγωγής, διαβάζοντας τα αποθηκευμένα δεδομένα από την κάρτα SD. Μέσω της βαθμίδας PWM (Pulse Width Modulation) του μικροελεγκτή και ενός χαμηλοπερατού φίλτρου, επιτυγχάνουμε την μετατροπή του ψηφιακού σήματος σε αναλογικό (DAC – Digital to Analog Converter). Με την βοήθεια ενός μικρού ενισχυτή σήματος και ενός μεγαφώνου ολοκληρώνεται η αναπαραγωγή του ήχου.

Κεφάλαιο 1^{ο} – Εισαγωγή στα ψηφιακά συστήματα ήχου

Στην ψηφιακή εποχή στην οποία ζούμε και στην τεχνολογική επανάσταση που βιώσαμε και συνεχίζουμε να βιώνουμε στόχος του ανθρώπου ήταν η μετατροπή του ογκώδες, ενεργοβόρου, ακριβού και σύνθετου αναλογικού κόσμου σε έναν μικρότερο, οικονομικότερο και απλούστερο για τον χρήστη, ψηφιακό κόσμο. Και έτσι από λυχνίες κενού περάσαμε σε τρανζίστορ, από δωματίου διαστάσεων υπολογιστές σε φορητούς υπολογιστές (laptops), από γραμμόφωνα σε φορητές συσκευής αναπαραγωγής ήχου (PMP – Portable Media Player). Δεδομένου ότι ο ήχος ήταν και συνεχίζει να είναι μία από τις μεγαλύτερες βιομηχανίες ψυχαγωγίας, δεν μπορούσε παρά να ακολουθήσει και αυτός την ψηφιακή οδό. Έτσι με αρχή το 1970 ξεκίνησε η επανάσταση, η οποία αντικαθιστώντας τους αναλογικούς τρόπους αναπαραγωγής του ήχου (δίσκους βινυλίου, κασέτες) επέτρεψε την μετάβαση στους ψηφιακής μορφής τρόπους (cd's, mp3 players).

Ψηφιακό σύστημα ήχου είναι οποιαδήποτε τεχνολογική συσκευή η οποία μπορεί να ηχογραφεί, αποθηκεύει, δημιουργεί, επεξεργαστεί, παραποιήσει και να αναπαράγει ήχο ο οποίος είναι κωδικοποιημένος σε ψηφιακή μορφή. Επιπλέον υπάρχουν και πολυπλοκότερες μονάδες ψηφιακών συστημάτων ήχου οι οποίες μπορούν να υλοποιήσουν και κάποιες επιπλέον λειτουργίες όπως συμπίεση και μετάδοση. Η χρήση των συστημάτων αυτών είναι πολύ διαδεδομένη στην καθημερινότητά μας και για τον λόγο αυτό είναι ολοένα μεγαλύτερη η χρήση τους είτε ενσωματωμένα σε άλλα υπολογιστικά συστήματα (π.χ. ηλεκτρονικοί υπολογιστές, κινητά τηλέφωνα) είτε ως συσκευές αποκλειστικά αφιερωμένες στην διαχείριση του ήχου (π.χ. συσκευές ηχογράφησης, mp3 players, πολυεφέ ηλεκτρικών μουσικών οργάνων). Στην συνέχεια θα υπάρξει μια θεωρητική προσέγγιση των διαφόρων βαθμίδων και δυνατοτήτων ενός ψηφιακού συστήματος ήχου. Στην εικόνα 1 θα δοθεί μια σχηματική απεικόνιση ενός τυπικού ψηφιακού συστήματος ήχου με τις βασικότερες βαθμίδες.



Εικόνα 1 – Σύνοψη ενός τυπικού ψηφιακού συστήματος ήχου

1.1 - Ηχογράφηση

Το πρώτο στάδιο ενός ψηφιακού συστήματος ήχου είναι η ηχογράφηση, δηλαδή ή συλλογή του αναλογικού ήχου που παράγεται από ένα φυσικό όν (άνθρωπος ή μουσικό όργανο στις περισσότερες περιπτώσεις) και την εισαγωγή αυτού στο ψηφιακό σύστημα. Η συλλογή του ήχου πραγματοποιείται πάντα με την χρήση μικροφώνου. Υπάρχουν πάρα πολλοί τύποι μικροφώνου οι οποίοι επιλέγονται ανάλογα με την εφαρμογή που θέλουμε να υλοποιηθεί. Το πυκνωτικό μικρόφωνο είναι ένας από τους πιο κοινούς τύπους μικροφώνου που χρησιμοποιούνται στα περισσότερα ψηφιακά συστήματα ήχου. Ο ήχος λόγω της κυματικής του φύσης για να μπορέσει να γίνει αντιληπτός από την ψηφιακή συσκευή πρέπει από αναλογικός που είναι να μετατραπεί σε ψηφιακό. Με την χρήση του μικροφώνου μετατρέπουμε τον ήχο από κύμα σε ηλεκτρικό σήμα. Εφόσον ο ήχος γίνει αντιληπτός από το μικρόφωνο σαν ηλεκτρικό σήμα οδηγείται, σε μία από τις βασικότερες μονάδες ενός ψηφιακού συστήματος ήχου, στον μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC) για την ψηφιοποίηση του.

Ο ADC (Analog to Digital Converter), σύμφωνα με την ονομασία του, αναλαμβάνει την πραγματοποίηση της μετατροπής του ήχου από αναλογικό σήμα σε ψηφιακό. Υπάρχουν πάρα πολλές σχεδιαστικές επιλογές για τον ADC ανάλογα με την πολυπλοκότητα και την ανάγκη της εφαρμογής. Συνήθως τους μετατροπείς τους βρίσκουμε στην αγορά ως ολοκληρωμένα κυκλώματα ή ενσωματωμένους μέσα σε μικροελεγκτές. Ο ADC χωρίζεται σε δύο διαδικασίες, την δειγματοληψία και την κβαντοποίηση. Με την δειγματοληψία γίνεται η περιοδική μετατροπή ενός σήματος τάσης (συνάρτηση του χρόνου) σε σήμα διακριτού χρόνου (ακολουθία πραγματικών αριθμών). Υπάρχει ένας πολύ σημαντικός παράγοντας ο οποίος πρέπει να προσεχθεί στην διαδικασία της δειγματοληψίας. Ο παράγοντας αυτός είναι ή συχνότητα δειγματοληψίας, δηλαδή κάθε πότε πρέπει να γίνεται δειγματοληψία μίας νέας τιμής. Η επιλογή της συχνότητας μπορεί να επηρεάσει σε μεγάλο βαθμό την λειτουργία του μετατροπέα ο οποίος επηρεάζεται και από το υλισμικό που χρησιμοποιείται. Στην συνέχεια με την κβαντοποίηση γίνεται αντικατάσταση του κάθε πραγματικού αριθμού σε ένα πεπερασμένο σύνολο διακριτών τιμών για την αποθήκευση και

επεξεργασία με χρήση αριθμητικών μεθόδων. Αυτές οι διακριτές τιμές αναπαριστούνται με χρήση λέξεων που έχουν σταθερό μέγεθος. Το μήκος των λέξεων αυτών ονομάζεται ανάλυση και αναπαριστά τον αριθμό των διακριτών τιμών που παράγονται στο συνολικό εύρος των αναλογικών τιμών. Έτσι ένας ADC με ανάλυση 8-bit μπορεί να κωδικοποιήσει μια αναλογική είσοδο σε ένα από 256 επίπεδα (2^8 = 256). Είναι προφανές ότι κάνοντας χρήση ενός ADC θα έχουμε ποσοστά θορύβου στο αποκτημένο ψηφιακό σήμα, λόγω της απώλειας πληροφοριών που προκύπτει κατά την διαδικασία κβαντοποίησης. Αυτό γίνεται επειδή οι διακριτές τιμές που παράγουμε είναι μια προσέγγιση των αναλογικών συνεχών τιμών. Η μείωση αυτού του θορύβου μπορεί να ελαττωθεί κάνοντας χρήση μεγαλύτερων αναλύσεων και καλού σχεδιασμού κατά την υλοποίηση των ADC. Έπειτα της μετατροπής του ήχου ο ADC παράγει μία σειρά ψηφιακών λέξεων με 0 και 1 τα οποία οδηγούνται στο αποθηκευτικό μέσο του συστήματος. Για την ροή δεδομένων που παράγεται από τον ADC χρησιμοποιούμε μια μέθοδο διαμόρφωσης που ονομάζεται PCM (Pulse Code Modulation). Το διαμορφωμένο σήμα PCM που αποκτούμε είναι μια ψηφιακή αναπαράσταση του αναλογικού σήματος όπου το πλάτος του σήματος δειγματοληπτείται, κβαντίζεται και μεταδίδεται ως συμβολοσειρά δυαδικών bit. Είναι η πρότυπη (standard) μορφή ψηφιακού ήχου.

1.2 – Αποθήκευση

Στα ψηφιακά συστήματα ήχου η αποθήκευση των αρχείων ήχου πραγματοποιείται είτε σε κάποια εσωτερική μνήμη που το σύστημα έχει (εσωτερικός δίσκος) είτε, σε μεγαλύτερο βαθμό σε αφαιρούμενους τρόπους αποθήκευσης (σε μία μνήμη USB ή σε κάρτας μνήμης όπως μίας SD Card). Στην μουσική βιομηχανία ο ψηφιακός ήχος αποθηκεύεται και εμπορεύεται σε μορφή CD (Compact Disk). Επειδή όμως η χρήση των CDs γίνεται ολοένα και μικρότερη και πια έχει αποκτήσει κυρίως συλλεκτικό χαρακτήρα, η μουσική βιομηχανία έχει στραφεί σε μεγάλο βαθμό και στην χρήση ψηφιακών υπηρεσιών μουσικής όπως είναι το iTunes, μέσω του οποίου αγοράζουμε το ψηφιακό περιεχόμενο και το ακούμε είτε στον υπολογιστή μας είτε στην κινητή συσκευή μας. Εναλλακτικά υπάρχουν υπηρεσίες streaming όπως το Spotify οι οποίες μας παρέχουν μια μεγάλη γκάμα μουσικών επιλογών στην οποία μπορούμε να επιλέξουμε όποιο

τραγούδι η άλμπουμ θέλουμε να ακούσουμε με ένα μηνιαίο κόστος. Η γενικότερη τάση είναι να γίνει απαλοιφή του αποθηκευτικού χώρου ως μέσο και την χρήση online αποθηκευτικών χώρων (cloud) ή υπηρεσιών streaming για την αναπαραγωγή της μουσικής.

Ανάλογα με τον τύπο αποθηκευτικού μέσου που επιλέγεται υπάρχει και η αντίστοιχη διαδικασία αρχικοποίησης και προετοιμασίας του μέσου αποθήκευσης. Υπάρχουν συγκεκριμένοι χρόνοι οι οποίοι πρέπει να τηρηθούν για την διαδικασία εγγραφής ή για την διαδικασία ανάγνωσης. Το μέγεθος του κάθε μπλοκ μνήμης που χρησιμοποιείται είναι και αυτό παραμετροποιήσιμο. Επίσης μεγάλη σημασία έχει με ποια μέθοδο θα γίνει η επικοινωνία του αποθηκευτικού μέσου με τον εκάστοτε επεξεργαστή ή μικροελεγκτή του ψηφιακού συστήματος. Γενικότερα κάθε επιλογή που αφορά τον τρόπο αποθήκευσης μπορεί να επηρεάσει σημαντικά την όψη και λειτουργία της συσκευής.

1.3 – Συμπίεση

Η συμπίεση δεδομένων σε ένα ψηφιακό σύστημα αφορά την κωδικοποίηση της πληροφορίας χρησιμοποιώντας λιγότερα bits σε σχέση με την αρχική μορφή που λαμβάνουμε στην έξοδο του ADC. Η διαδικασία της συμπίεσης γίνεται με στόχο την μείωση του αποθηκευτικού χώρου που χρησιμοποιείται αλλά ταυτόχρονα και την μείωση των απαιτούμενων πόρων κατά την μετάδοση των δεδομένων.

Η συμπίεση χωρίζεται σε δύο κατηγορίες. Την απωλεστική (lossy) και μη απωλεστική (lossless) συμπίεση. Με την μη απωλεστική συμπίεση έχουμε πλήρη ακεραιότητα δεδομένων. Αυτό σημαίνει πως τα αρχικά δεδομένα και τα δεδομένα που αποκτούμε μετά από την συμπίεση και την αποσυμπίεση είναι ακριβώς τα ίδια. Ο τύπος της συμπίεσης αυτός βασίζεται στο γεγονός ότι στον αναλογικό κόσμο το ίδιο φαινόμενο εμφανίζεται πολλαπλές φορές σε σειρά και μπορεί να κωδικοποιηθεί (π.χ. το ίδιο χρώμα σε μια φωτογραφία, ο ίδιος τόνος σε ένα μουσικό απόσπασμα). Με την χρήση της μη απωλεστικής συμπίεσης δεν χάνουμε την ποιότητα του ήχου αλλά ταυτόχρονα έχουμε και μικρή μείωση στον αποθηκευτικό χώρο. Η μη απωλεστική συμπίεση χρησιμοποιείται σε περιπτώσεις στις οποίες θέλουμε να επεξεργαστούμε τον ήχο χωρίς απώλεια δεδομένων με

στόχο την επίτευξη αυξημένης ποιότητας ήχου (που πολλές φορές απαιτείται) αλλά και για την ύπαρξη ενός πρωτοτύπου αντιγράφου (κυρίως στην μουσική βιομηχανία με την ονομασία master records).

Με την απωλεστική συμπίεση έχουμε απώλεια δεδομένων. Προφανώς η απώλεια των δεδομένων αυτή γίνεται με τρόπο έτσι ώστε να μην γίνει αντιληπτή από τον τελικό χρήστη. Υλοποιείται κάνοντας χρήση μεθόδων ψυχοακουστικής για την αφαίρεση των συχνοτήτων και των ήχων που δεν είναι αντιληπτοί από το ανθρώπινο ακουστικό σύστημα. Με την μέθοδο αυτή είτε οι πολύ υψηλές συχνότητες είτε ήχοι που προκύπτουν ταυτόχρονα με ήχους με μεγαλύτερη ένταση ή κωδικοποιούνται με μικρότερη ακρίβεια ή δεν κωδικοποιούνται καθόλου. Συμπεραίνουμε ότι η απωλεστική συμπίεση είναι μία μέθοδος με την οποία έχουμε μεγάλη εξοικονόμηση του μεγέθους των αρχείων και η ακουστική ποιότητα μειώνεται άλλα σε βαθμό που είναι αποδεκτή από τον μέσο χρήστη. Δεν είναι μια μέθοδος η οποία μπορεί να χρησιμοποιηθεί για σκοπούς παραγωγής η επεξεργασία της μουσικής γιατί εκεί είναι απαραίτητη η ύπαρξη όλων των φασμάτων για ένα ποιοτικό αποτέλεσμα.

Πιο γνωστοί τύποι συμπιεσμένων αρχείων ήχου για την μη απωλεστική μέθοδο είναι τα FLAC (Free Lossless Audio Codec) και το WMAL (Windows Media Audio Lossless). Ενώ αντίστοιχα για την απωλεστική μέθοδο είναι τα γνωστά σε όλους mp3 (MPEG-1 Layer III) και τα wma (Windows Media Audio). Όπως παρατηρούμε η απωλεστική μέθοδος είναι ευρέως διαδεδομένη και λόγω του χαμηλότερου μεγέθους αρχείου που προσφέρει αλλά και λόγο της ευκολότερης διάδοσής του στο διαδίκτυο. Πολλά ψηφιακά συστήματα ήχου παρέχουν την δυνατότητα συμπίεσης του ήχου σε ένα συγκεκριμένο τύπο συμπίεσης (π.χ. mp3 ή wma) ενώ ταυτόχρονα υπάρχουν και πιο ανεπτυγμένα συστήματα τα με τα οποία επιλέγουμε εμείς τι κωδικοποίηση θα χρησιμοποιηθεί και τι ακρίβειας θα είναι αυτή.

1.4 – Δημιουργία, Επεξεργασία, Παραποίηση

Τα ψηφιακά συστήματα ήχου είναι, όπως αναλύθηκε ευρέως, διαδεδομένα στον τελικό χρήστη και έχουν πάρα πολλές μορφές και σε αυτόνομες συσκευές αλλά και ως ενσωματωμένες μονάδες σε άλλα μεγαλύτερα ή μικρότερα

υπολογιστικά συστήματα. Πέρα όμως από την “οικιακή” χρήση που μπορεί να έχει ένα ψηφιακό σύστημα ήχου η ανάπτυξη την οποία είχε την οφείλει στις επαγγελματικές εφαρμογές και δυνατότητες που έχει στον χώρο της μουσικής βιομηχανίας και παραγωγής.

Στην βιομηχανία της μουσικής είναι πολύ συνηθισμένο το φαινόμενο της επεξεργασίας και παραποίησης της μουσικής. Συνήθως πραγματοποιείται είτε για την βελτίωση του ακουστικού αποτελέσματος είτε για την διόρθωση λαθών που προέκυψαν κατά την διαδικασία ηχογράφησης (μουσικών και τραγουδιστών). Την επεξεργασία αυτή την αναλαμβάνουν ειδικά λογισμικά προγράμματα σε συνδυασμό με συγκεκριμένο υλισμικό που αποτελείται από ειδικές κάρτες ήχου, διάφορους μίκτες και ισοσταθμιστές. Τα συστήματα αυτά ονομάζονται DAW (Digital Audio Workstation – Σταθμός εργασίας ψηφιακού ήχου). Κάνοντας χρήση των DAW οι μουσικοί παραγωγοί έχουν την δυνατότητα της ηχογράφησης του κάθε μουσικού οργάνου ξεχωριστά και μετά μίξη όλων σε ένα κομμάτι. Με αυτόν τον τρόπο μπορούν, πέρα από το να διορθώσουν, να προσθέσουν ειδικά εφέ στον ήχο και να το εμπλουτίζουν με άλλους ήχους που έχουν δημιουργηθεί ψηφιακά. Τα πιο γνωστά DAW λογισμικά που υπάρχουν είναι το FL Studio, το Ableton Live, το Pro Tools ενώ ταυτόχρονα υπάρχουν και πάρα πολλά άλλα με αντίστοιχες δυνατότητες. Αυτά τα λογισμικά χρησιμοποιούνται και για επαγγελματικό σκοπό σε ειδικά διαμορφωμένα studio αλλά σε πολλές περιπτώσεις υπάρχουν οι πιο απλές εκδόσεις οι οποίες χρησιμοποιούνται κυρίως από ερασιτέχνες.

Πέρα από την επεξεργασία και παραποίηση της μουσικής τα τελευταία χρόνια υπάρχει ιδιαίτερη ανάπτυξη της ηλεκτρονικής μουσικής καθιστώντας έτσι δυνατή την δημιουργία μουσικής μέσω ενός ψηφιακού συστήματος. Είναι ένα είδος μουσικής το οποίο παράγεται εξ’ ολοκλήρου μέσα σε κάποιο studio από μουσικούς παραγωγούς κάνοντας χρήση προηχογραφημένων ήχων οι οποίοι συνθέτονται δημιουργώντας μουσικά κομμάτια. Είναι ένα είδος μουσικής το οποίο, στις μέρες μας, είναι αρκετά διαδεδομένο και ο λόγος της διάδοσής του είναι κυρίως ή “απλότητα” παραγωγής, που καθιστά δυνατό στον καθένα την δημιουργία μουσικών αποσπασμάτων χωρίς την κατοχή ιδιαίτερης μουσικής γνώσης.

1.5 – Αναπαραγωγή

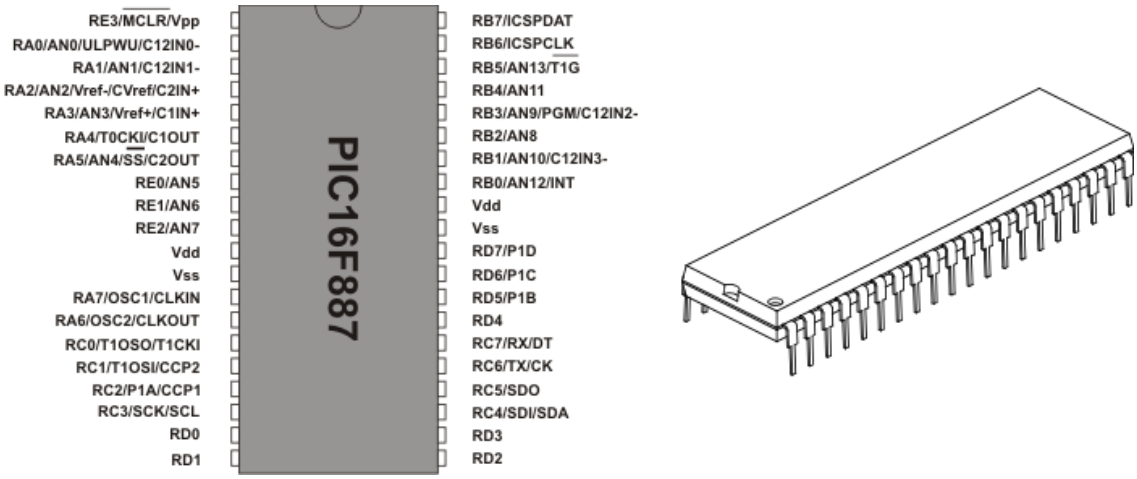
Η αναπαραγωγή είναι το τελευταίο κομμάτι σε ένα ψηφιακό σύστημα ήχου. Με την μετάδοση του ήχου από τον χώρο αποθήκευσης, στον μετατροπέα ψηφιακού σήματος σε αναλογικό (DAC) ολοκληρώνεται η πορεία του ήχου μέσα στο ψηφιακό σύστημα. Στον DAC γίνεται η μετατροπή των διακριτών τιμών ξανά σε αναλογικές τιμές σύμφωνα με τα επίπεδα που είχαν δημιουργηθεί για το κάθε ψηφιακό δείγμα. Για την ολοκλήρωση της διαδικασίας και για να ακούσουμε επιτυχώς την ηχογράφηση που κάναμε, ο πλέον ξανά αναλογικός ήχος φιλτράρεται, εφόσον αυτό είναι απαιτητό ανάλογα της μεθόδου που χρησιμοποιήθηκε, ενισχύεται (γιατί έπειτα της μετατροπής έχουμε χαμηλό πλάτος σήματος το οποίο δεν είναι αρκετό για την αναπαραγωγή) και γίνεται η μετάδοσή του σε ηχεία τα οποία το μετατρέπουν ξανά σε κύμα για να μπορέσει να γίνει αντιληπτό από τον άνθρωπο.

Όπως και στην ηχογράφηση υπάρχει ο ADC που ουσιαστικά υλοποιεί την διαδικασία ψηφιοποίησης έτσι και στην αναπαραγωγή έχουμε τον DAC ο οποίος υλοποιεί την αντίστροφη διαδικασία. Ο DAC (Digital to Analog Converter) στα ψηφιακά συστήματα ήχου μετατρέπει τις διακριτές τιμές που είναι αποθηκευμένες στην μνήμη της συσκευής σε τάση για να μπορέσει έπειτα να γίνει η αναπαραγωγή. Η επιλογή τους βασίζεται, σε μεγάλο βαθμό, σε τρεις βασικές παραμέτρους, την ανάλυση, την μέγιστη συχνότητα δειγματοληψίας και την ακρίβεια. Λόγω της πολυπλοκότητας και της ανάγκης επιλογής βαθμίδων οι οποίες ταιριάζουν απόλυτα μεταξύ τους τα περισσότερα DAC είναι υλοποιημένα ως ολοκληρωμένα κυκλώματα. Ο τελικός στόχος των DAC είναι να ανακατασκευάσουν το αρχικό σήμα. Οι DAC χρησιμοποιούνται ευρέως στην καθημερινότητα μας και υπάρχουν στις περισσότερες ψηφιακές συσκευές. Υπάρχουν πάρα πολλές τεχνικές υλοποίησης DAC ανάλογα με την εφαρμογή με την οποία βρισκόμαστε αντιμέτωποι. Σε εφαρμογές που αφορούν την αναπαραγωγή ήχου μπορούμε να βρούμε DAC μέσα σε CD players, σε mp3 players καθώς και στις κάρτες ήχου των υπολογιστών. Επιπλέον όταν είναι απαραίτητη μία ανώτερη ποιότητα ήχου και αυτό συμβαίνει συνήθως στα hi-fi

συστήματα υπάρχουν ειδικά σχεδιασμένοι DAC οι οποίοι εξομοιώνουν την έξοδο που λαμβάνεται από το ηχοσύστημα και την είσοδο που απαιτείται για την οδήγηση προς τον ενισχυτή, για να παρέχουν την μέγιστη απόδοση ποιότητας κατά την αναπαραγωγή του ήχου. Αντίστοιχα χρήση τους υπάρχει και στις εφαρμογές VoIP όπου στην πλευρά του αποδέκτη της συνομιλίας υπάρχει ένας DAC ο οποίος ανακατασκευάζει το αναλογικό σήμα.

1.6 – Αρχιτεκτονική του PIC16F877

Στην καρδιά του ψηφιακού συστήματος ήχου που υλοποιήθηκε υπάρχει ο μικροελεγκτής της εταιρείας Microchip Technology PIC16F877. Ο μικροελεγκτής αυτός κατατάσσεται στην μεσαία κατηγορία της οικογένειας μικροελεγκτών PIC. Είναι ένας πολύ προχωρημένος, σε σχέση με τους προγενέστερους, μικροελεγκτής ο οποίος με 40 ακροδέκτες, 35 εντολές και ένα αρκετά εκτενές εύρος περιφερειακών, καθιστά τον κύκλωμα αυτό ιδανικό σε εφαρμογές βιομηχανικού ελέγχου καθώς και σε συσκευές καθημερινής χρήσης. Έχει πλέον αποκτήσει και τον τίτλο εκπαιδευτικού μικροελεγκτή λόγω του συνδυασμού υψηλών επιδόσεων, εύκολου προγραμματισμού και χαμηλού κόστους. Στην εικόνα 2 παρουσιάζονται οι ακροδέκτες και η ονομασία τους.



Εικόνα 2 – Ακροδέκτες του μικροελεγκτή PIC16F877

Είναι ένας μικροελεγκτής αρχιτεκτονικής RISC και ακολουθεί το μοντέλο μνήμης Harvard δηλαδή να υπάρχει διαχωρισμός, του χώρου αποθήκευσης, των διαύλων επικοινωνίας, των εντολών και των δεδομένων. Η δομή του PIC μπορεί να χωριστεί σε 2 βασικά μέρη, τον πυρήνα και τις περιφερειακές μονάδες. Στον πυρήνα βρίσκεται η CPU, η μνήμη και οι λειτουργίες διακοπών. Ενώ στις

περιφερειακές μονάδες οι 5 θύρες εισόδου - εξόδου (τρείς θύρες των 8 bit, μία θύρα των 6 bit και μία των 3 bit), μετρητές χρόνου (ένα μετρητής 16 bit και δύο των 8 bit), μετατροπέας αναλογικού σήματος σε ψηφιακό, μονάδα διαμόρφωσης πλάτους, τρείς θύρες σειριακής επικοινωνίας, συγκριτές και μονάδα παραγωγής τάσης αναφοράς. Ξεκινώντας από τον πυρήνα παρατηρούμε πως η CPU αποτελείται από την ALU (Αριθμητική και λογική μονάδα επεξεργασίας – Arithmetic Logic Unit) που εκτελεί τις διάφορες αριθμητικές και λογικές πράξεις. Μαζί με συγκεκριμένους καταχωρητές ειδικής χρήσης υλοποιείται η εκτέλεση εντολών που διαβάζονται από την ROM μνήμη. Η μνήμη χωρίζεται σε τρείς διαφορετικούς τύπους, την ROM, την RAM και την EEPROM. Στην μνήμη ROM αποθηκεύεται το κυρίως πρόγραμμα που εκτελείται και για αυτό και ονομάζεται μνήμη προγράμματος. Έχει μέγεθος 8K (14-bit λέξεων) και αυτό θέτει ένα "όριο" στο μέγεθος προγράμματος που πρόκειται να υλοποιηθεί. Ο προγραμματισμός της μνήμης αυτής πραγματοποιείται με χρήση ειδικού προγραμματιστή. Τα δεδομένα προφανώς και μετά την διακοπή παροχής παραμένουν αποθηκευμένα. Ο προγραμματιστής που χρησιμοποιήθηκε στην παρούσα εργασία, ο οποίος είναι και η τελευταία έκδοση που υπάρχει, είναι ο PICkit 3 επίσης της Microchip Technologies. Στην εικόνα 3 φωτογραφία του προγραμματιστή.

Εικόνα 3 – Ο προγραμματιστής PICkit 3

Η EEPROM μνήμη είναι και αυτή μία μνήμη στην οποία τα δεδομένα αποθηκεύονται και μετά την διακοπή τάσης. Η μόνη διαφορά είναι ότι αυτή η μνήμη αποθηκεύει δεδομένα του προγράμματος εν ώρα εκτέλεσης. Με αυτό τον τρόπο μπορούμε να αποθηκεύουμε με ασφάλεια τα σημαντικά δεδομένα σε μία μόνιμη μνήμη. Το μέγεθος της μνήμης αυτής είναι 256 byte. Τέλος υπάρχει η RAM μνήμη. Στην μνήμη αυτή όλες οι τιμές που δίνουμε στους καταχωρητές είναι προσωρινά διαθέσιμες. Η RAM χωρίζεται σε δύο μέρη. Τους καταχωρητές γενικού σκοπού (GPR) και τους καταχωρητές ειδικών λειτουργιών (SFR). Οι καταχωρητές γενικού σκοπού έχουν μέγεθος 368 byte. Χρησιμοποιούνται κυρίως για την αποθήκευση προσωρινών αποτελεσμάτων π.χ. από μία πράξη ή για την αποθήκευση κάποιου μετρητή. Γενικότερα μπορούν να χρησιμοποιηθούν με οποιοδήποτε τρόπο. Οι SFR αντιθέτως είναι καταχωρητές οι οποίοι, μέσω υλισμικού, έχουν προκαθορισμένες λειτουργίες και συνδέονται άμεσα με συγκεκριμένες βαθμίδες στο κύκλωμα. Έτσι, αναλόγως με τις τιμές που αρχικοποιούνται, ρυθμίζουν ανάλογα τις βαθμίδες αυτές. Η διευκόλυνση που έχει γίνει βέβαια για τους καταχωρητές ειδικού σκοπού είναι ότι και οι ίδιοι καταχωρητές αλλά και τα bit, έχουν ονομαστεί καθιστώντας εύκολη την απομνημόνευσή τους και τη προσπέλασή τους. Η RAM μνήμη χωρίζεται σε 4 μέρη (banks). Για να γίνει εγγραφή θα πρέπει αρχικά να οριστεί σε ποιο bank κοιτάμε και έπειτα να γίνει η απαραίτητη εγγραφή. Στις χαμηλές διευθύνσεις των banks βρίσκουμε τους SFR ενώ στις υπόλοιπες υπάρχουν οι GPR. Στην εικόνα 4 απεικονίζονται τα 4 banks μνήμης με τους SFR και GPR.

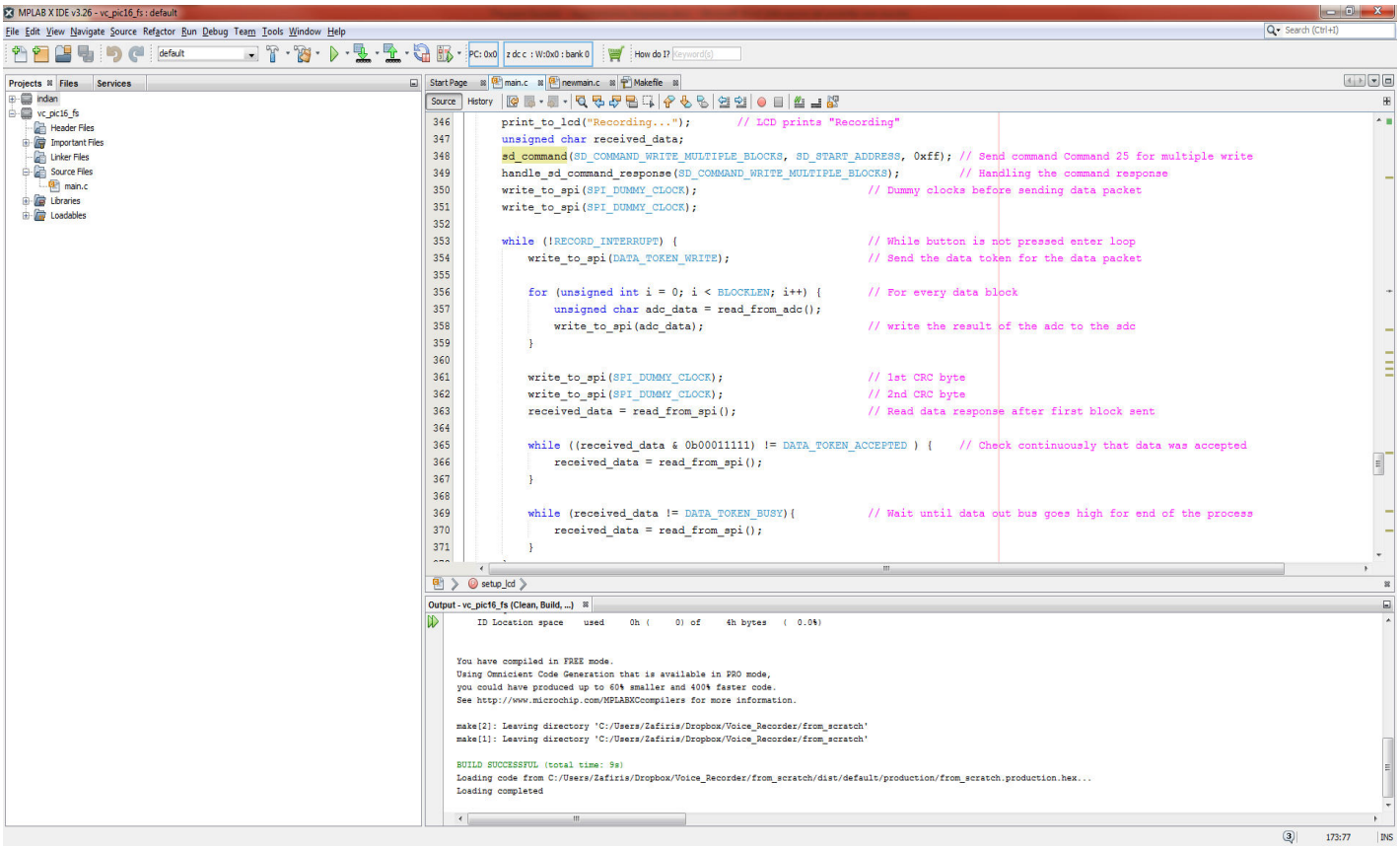
Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name
00h	INDF	80h	INDF	100h	INDF	180h	INDF
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h	WDTCON	185h	SRCON
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h	CM1CON0	187h	BAUDCTL
08h	PORTD	88h	TRISD	108h	CM2CON0	188h	ANSEL
09h	PORTE	89h	TRISE	109h	CM2CON1	189h	ANSELH
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDAT	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	Not Used
0Fh	TMR1H	8Fh	OSCCON	10Fh	EEADRH	18Fh	Not Used
10h	T1CON	90h	OSCTUNE	110h		190h	
11h	TMR2	91h	SSPCON2				
12h	T2CON	92h	PR2				
13h	SSPBUF	93h	SSPADD				
14h	SSPCON	94h	SSPSTAT				
15h	CCPR1L	95h	WPUB				
16h	CCPR1H	96h	IOCB				
17h	CCP1CON	97h	VRCON				
18h	RCSTA	98h	TXSTA				
19h	TXREG	99h	SPBRG				
1Ah	RCREG	9Ah	SPBRGH		General Purpose Registers 96 bytes		General Purpose Registers 96 bytes
1Bh	CCPR2L	9Bh	PWM1CON				
1Ch	CCPR2H	9Ch	ECCPAS				
1Dh	CCP2CON	9Dh	PSTRCON				
1Eh	ADRESH	9Eh	ADRESL				
1Fh	ADCON0	9Fh	ADCON1				
20h	General Purpose Registers 96 bytes	A0h	General Purpose Registers 80 bytes				
7Fh		FFh		17Fh		1EFh	
	Bank 0		**Bank 1**		**Bank 2**		**Bank 3**

Εικόνα 4 – Τα 4 διαφορετικά τμήματα (banks) της RAM μνήμης

Τέλος με την λειτουργία διακοπών, ενώ τρέχει το πρόγραμμα και λάβει ένα interrupt, σταματάει την εκτέλεση του τρέχον προγράμματος και στέλνει την θέση μνήμης του στην μνήμη stack όπου αποθηκεύεται. Πραγματοποιείται η εκτέλεσή του και μόλις ολοκληρωθεί τότε γίνεται ανάκτηση της διεύθυνσης μνήμης του προγράμματος που εκτελούταν πριν από το interrupt και συνεχίζει η ροή της εφαρμογής. Σε συνδυασμό με τις περιφερειακές μονάδες ο μικροελεγκτής PIC16F877 θέτει την βάση για την υλοποίηση πολλών εφαρμογών και για επαγγελματική χρήση αλλά και για εκπαιδευτική και οικιακή χρήση.

1.7 – Περιβάλλον προγραμματισμού MPLAB X IDE

Το περιβάλλον προγραμματισμού MPLAB X IDE που χρησιμοποιήθηκε για τους σκοπούς της εργασίας αυτής είναι η πιο πρόσφατη έκδοση της σειράς MPLAB. Η σχεδίασή του είναι βασισμένη στην, ανοικτού κώδικα, πλατφόρμα NetBeans. Χάρη στην χρήση αυτής της σχεδίασης το MPLAB X μας δίνει πολλές και νέες δυνατότητες εύκολα και γρήγορα. Με την χρήση του MPLAB X μπορεί να γίνει επεξεργασία, αποσφαλμάτωση, και προγραμματισμός όλων των οικογενειών μικροελεγκτών της Microchip. Μία από τις βασικότερες διαφορές σε σχέση με το MPLAB είναι ότι το MPLAB X είναι cross-platform και υποστηρίζει, πέρα των Windows, Mac OS και Linux λειτουργικά συστήματα. Ο μεταγλωττιστής που χρησιμοποιήθηκε είναι ο MPLABXC8 με τον οποίο μπορεί να γίνει υλοποίηση προγραμμάτων σε γλώσσα C για τους 8-bit μικροελεγκτές PIC. Προφανώς με τον MPLAB X μπορεί να γίνει και χρήση άλλων γλωσσών προγραμματισμού πέρα της C. Στην εικόνα 5 βλέπουμε το περιβάλλον διεπαφής με το χρήστη της εφαρμογής



Εικόνα 5 – Διεπαφή του MPLAB X

Με την χρήση του MPLAB X μπορούμε να υλοποιήσουμε ολόκληρα έργα (Projects) για την υλοποίηση εφαρμογών για χρήση με τον μικροελεγκτή.

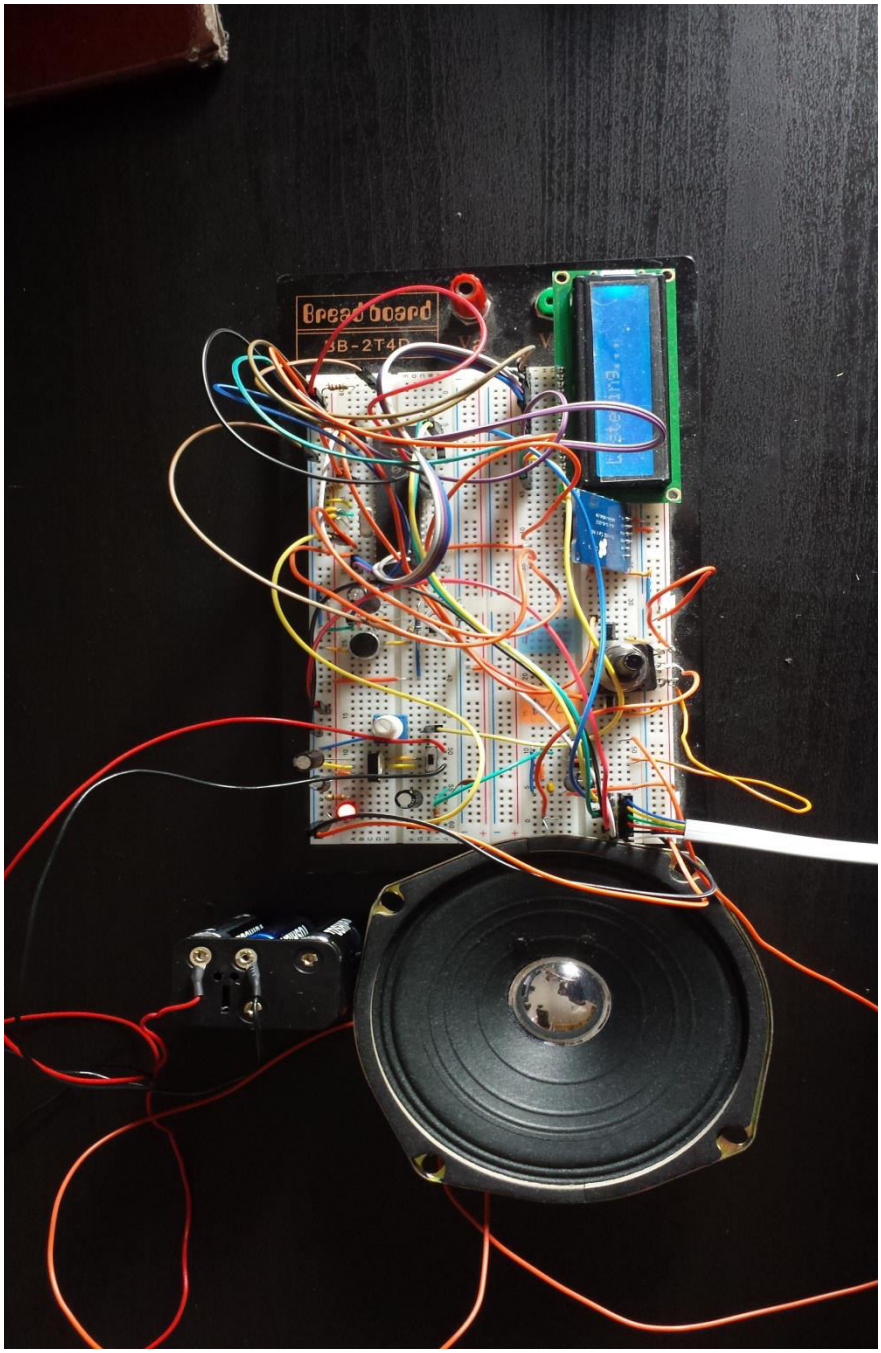
Κάνοντας χρήση του κειμενογράφου γράφουμε τον κώδικα για τις εφαρμογές μας και με το παράθυρο πλοήγησης μπορούμε να οργανώσουμε με ευκολία το έργο μας. Τέλος στο κάτω μέρος έχουμε το παράθυρο Output όπου γίνεται η εμφάνιση των αποτελεσμάτων έπειτα από το build της εφαρμογής. Στο παράθυρο αυτό βλέπουμε τα σφάλματα (errors) καθώς και τις προειδοποιήσεις (warnings) που προκύπτουν. Είναι ένα πλήρες περιβάλλον προγραμματισμού ιδιαίτερα φιλικό προς τον χρήστη για την ανάπτυξη εφαρμογών.

Στο σημείο αυτό, έχοντας ολοκληρώσει την εισαγωγή στα ψηφιακά συστήματα ήχου, θα ακολουθήσουν δύο ακόμη κεφάλαια όπου αναλύονται και επεξηγούνται τα στάδια ανάπτυξης της συσκευής ψηφιακής καταγραφής και αναπαραγωγής ήχου. Στο δεύτερο κεφάλαιο μελετάται το ηλεκτρονικό κύκλωμα που υλοποιήθηκε για τις ανάγκες της εργασίας με μία εκτενή παρουσίαση όλων των βαθμίδων που πήραν μέρος στην υλοποίηση του ψηφιακού αυτού συστήματος. Στο τρίτο θα αναλυθεί η ανάπτυξη της εφαρμογής (software) που υλοποιήθηκε. Στο παράρτημα θα αναρτηθεί όλος ο κώδικας μαζί με τα σχόλιά του ενώ τμήματα του κώδικα θα αναρτούνται κατά την επεξήγησή για να είναι εύκολη η αναφορά τους.

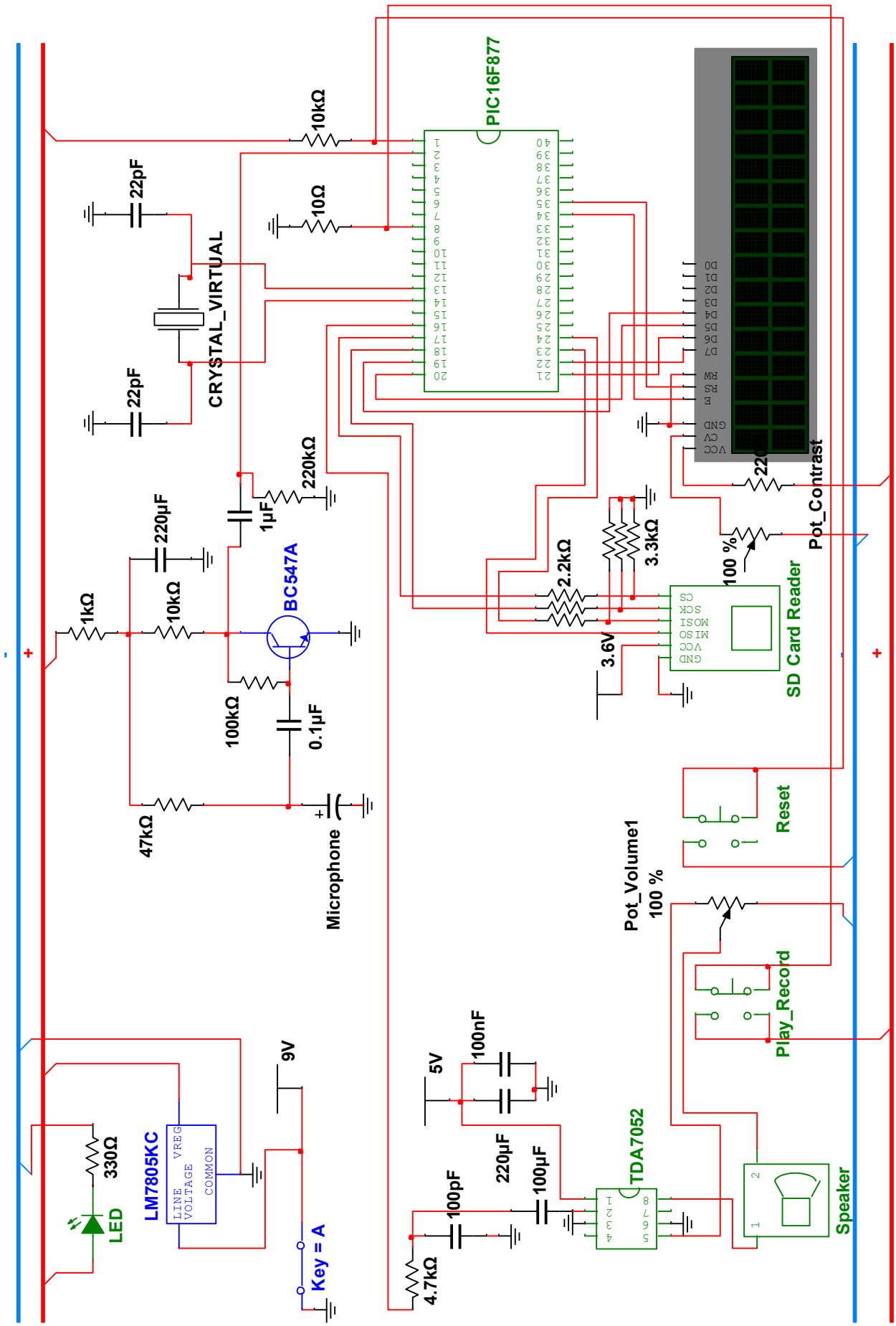
Η εργασία αυτή αναπτύχθηκε με δύο βασικές συνιστώσες. Η πρώτη αφορά το ακαδημαϊκό και εκπαιδευτικό ενδιαφέρον που παρουσιάζει, δεδομένου ότι το πρώτο μέρος της εργασίας ήταν ο σχεδιασμός και η υλοποίηση κάποιων από των σημαντικότερων και ευρέως χρησιμοποιήσιμων ηλεκτρονικών διατάξεων και πρωτοκόλλων επικοινωνίας που υπάρχουν στον χώρο των ενσωματωμένων συστημάτων και το δεύτερο μέρος, στον προγραμματισμό όλου αυτού του κυκλώματος, κάνοντας χρήση διαφόρων μεθόδων προγραμματισμού με στόχο την εναρμόνιση και ορθή λειτουργία αυτού του ηλεκτρονικού "οικοσυστήματος". Η δεύτερη συνιστώσα ήταν η πρόκληση αλλά και το ρίσκο που συνόδευε την εργασία δηλαδή αν θα ήταν εφικτό να υλοποιηθεί η εργασία αυτή με ένα σχετικά μεσαίας κατηγορίας μικροελεγκτή σαν τον PIC16F877 και να έχουμε ένα αποδεκτό ακουστικό αποτέλεσμα.

Κεφάλαιο 2^{ο} – Ανάλυση του κυκλώματος

Στο κεφάλαιο αυτό στόχος είναι να γίνει διαχωρισμός των διαφόρων διατάξεων του κυκλώματος και ανάλυση καθενός δίνοντας έμφαση στα βασικότερα σημεία. Επιπλέον θα προστεθούν κυκλωματικά σχέδια καθώς και φωτογραφίες από την συσκευή. Πριν ξεκινήσει η ανάλυση των κυκλωμάτων θα προστεθεί στην εικόνα 6 μία φωτογραφία και στην εικόνα 7 το κυκλωματικό σχέδιο όλου του κυκλώματος.



Εικόνα 6 – Φωτογραφία όλης της συσκευής



Εικόνα 7 – Κυκλωματικό σχέδιο της συσκευής

2.1 - Τροφοδοσία του κυκλώματος

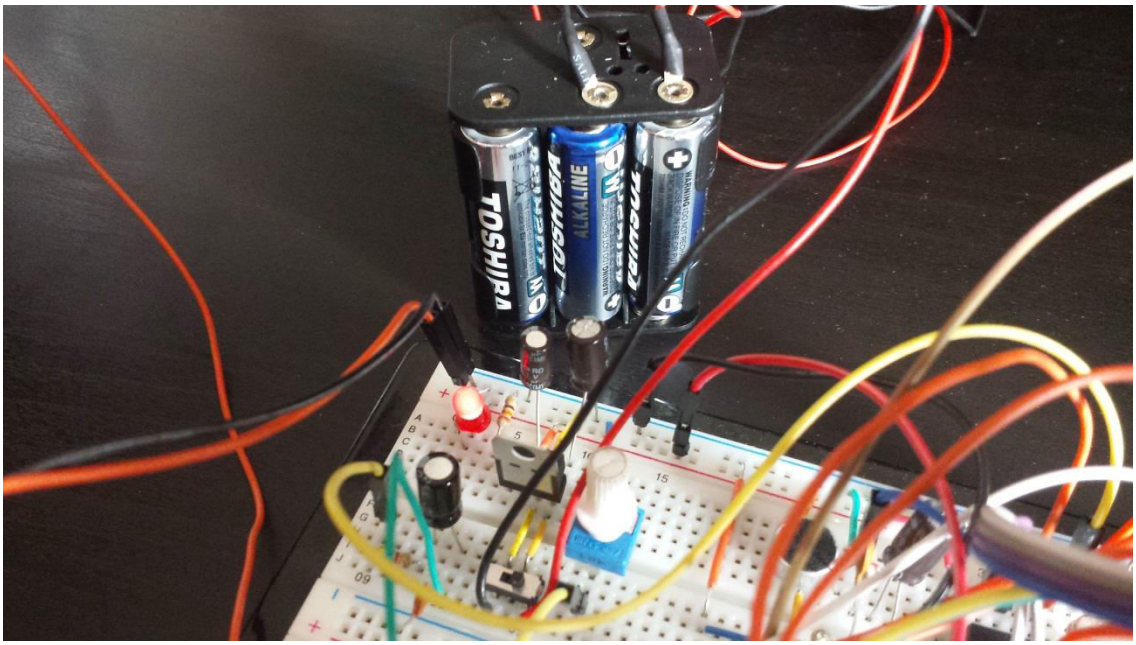
Το κύκλωμα τροφοδοσίας που υλοποιήθηκε αποτελείται από έναν σταθεροποιητή τάσης 5V και συγκεκριμένα τον L7805. Στον πίνακα 1 παρουσιάζονται οι συνδεσμολογίες που υλοποιήθηκαν μεταξύ του σταθεροποιητή τάσης και το κύκλωμα.

Pins No	L7805 Pins	Circuit	Explanation
1	Input Voltage	VDD	Power Supply Positive Pole
2	Ground	VSS	Power Supply Negative Pole/ Negative Power Line
3	Output Voltage	Circuit Input	Positive Power Line

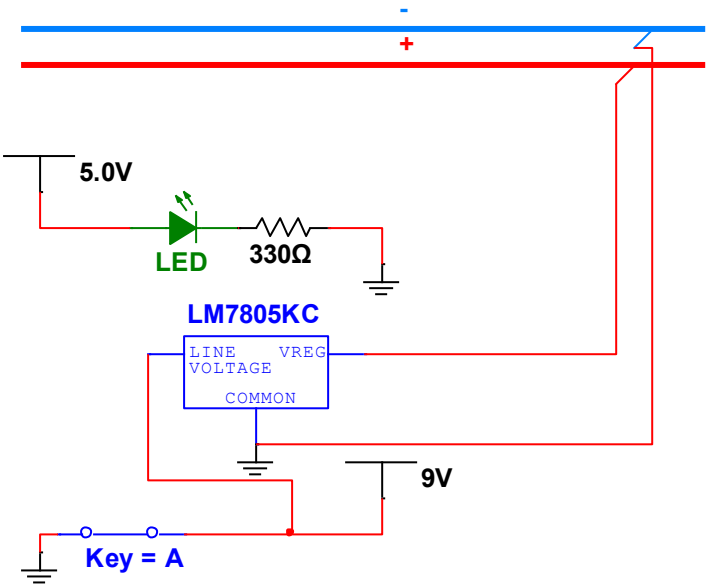
Πίνακας 1 – Πίνακας συνδεσμολογιών του κυκλώματος τροφοδοσίας

Ο πρώτος ακροδέκτης του σταθεροποιητή συνδέεται με τον θετικό πόλο της τροφοδοσίας. Στην συγκεκριμένη περίπτωση χρησιμοποιήθηκαν 6 μπαταρίες AA των 1.5V συνδεδεμένες σε σειρά σε ένα πακέτο με αποτέλεσμα να έχουμε 9V τάσης. Εδώ θα πρέπει να επισημανθεί ότι οι σταθεροποιητές τάσης, για να μπορέσουν να ανταπεξέλθουν την ονομαστική τάση που αναγράφεται στα φύλλα δεδομένων τους, πρέπει να τροφοδοτούνται με μία μεγαλύτερη τάση για να επιτευχθεί το επιθυμητό αποτέλεσμα. Για την συγκεκριμένη περίπτωση η ελάχιστη τάση εισόδου θα πρέπει να είναι στα 7.5V μέχρι και την μέγιστη 12V για τον εν λόγω σταθεροποιητή. Προτιμήθηκε ως τάση εισόδου αυτή των 9V επειδή με την επιλογή αυτή θα διαχέεται λιγότερη θερμότητα σε σύγκριση με την επιλογή των 12V και ταυτόχρονα θα υπάρχει μεγαλύτερη διάρκεια των μπαταριών. Ο δεύτερος ακροδέκτης του σταθεροποιητή είναι αυτός που συνδέεται με τον αρνητικό πόλο της τροφοδοσίας και ταυτόχρονα με την γραμμή τροφοδοσίας όπου θα γίνεται η γείωση όλων των ηλεκτρονικών στοιχείων και κυκλωμάτων που χρησιμοποιούνται στο κύκλωμα. Τέλος με τον τρίτο ακροδέκτη έχουμε την σταθεροποιημένη τάση εξόδου στα 5V και την εισάγουμε στην γραμμή τροφοδοσίας για την πλήρη παροχή όλου του κυκλώματος. Επιπλέον γίνεται και χρήση κάποιων πυκνωτών για την ομαλοποίηση της τάσης και την αποφυγή αιχμών και μιας led λυχνίας για την οπτική επιβεβαίωση λειτουργίας του κυκλώματος τροφοδοσίας.

Στο σημείο αυτό θα γίνει μια αναφορά για την επιλογή χρήσης της μπαταρίας για την τροφοδοσία του κυκλώματος και όχι κάποιου τροφοδοτικού του εμπορίου. Έγινε χρήση διαφόρων τροφοδοτικών του εμπορίου με αποτέλεσμα να υπάρχει συνεχώς στατικός ηλεκτρισμός στο κύκλωμα ο οποίος προκαλούσε δυσλειτουργία του κυκλώματος και κατ' επέκταση θα μπορούσε να καταστρέψει κάποιο από τα κυκλώματα. Αυτό οφειλόταν κυρίως από την κατασκευή των τροφοδοτικών μεταβλητής τάσης οι οποίοι προκαλούν αρμονικούς και είναι η ρυπογόνα πηγή που επιδρά αρνητικά στην ποιότητα της ηλεκτρικής ενέργειας. Αντιθέτως η μπαταρία, παρά το μεγάλο κόστος, είναι η πιο καθαρή μορφή ενέργειας και δεν προκαλεί καμία απολύτως δυσλειτουργία στο κύκλωμα. Επιπλέον προσφέρει την "φορητή αίσθηση" που αρμόζει σε μία συσκευή ηχογράφησης και αναπαραγωγής ήχου. Στην εικόνα 8 θα προστεθεί μία φωτογραφία του τροφοδοτικού και στην εικόνα 9 το κυκλωματικό σχέδιο.



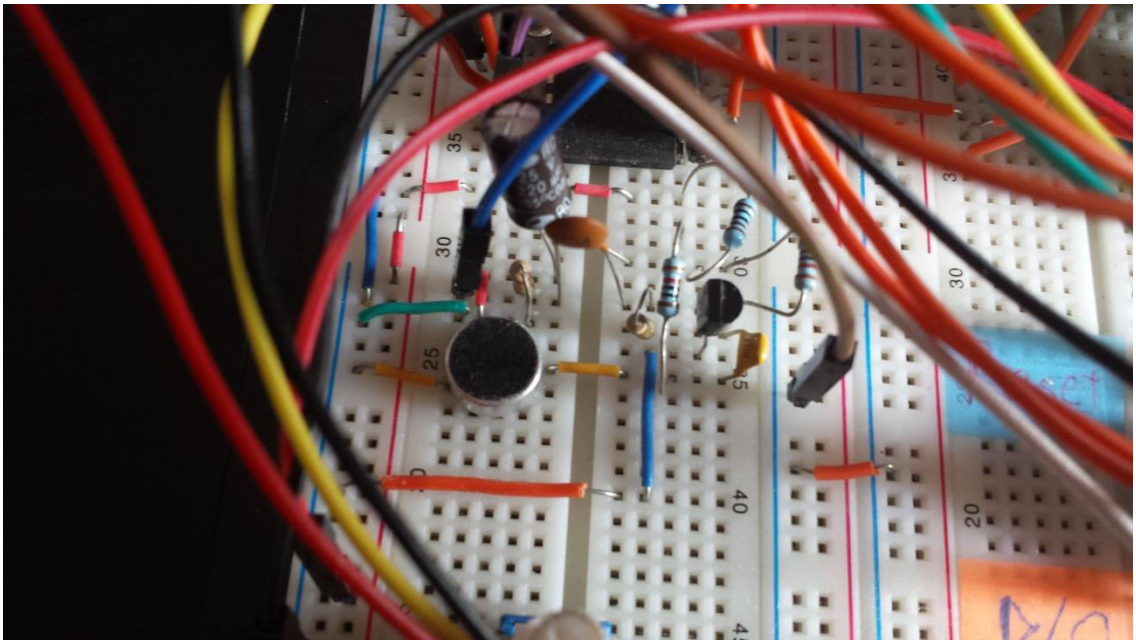
Εικόνα 8 – Φωτογραφία του κυκλώματος τροφοδοσίας



Εικόνα 9 – Κυκλωματικό σχέδιο του κυκλώματος τροφοδοσίας

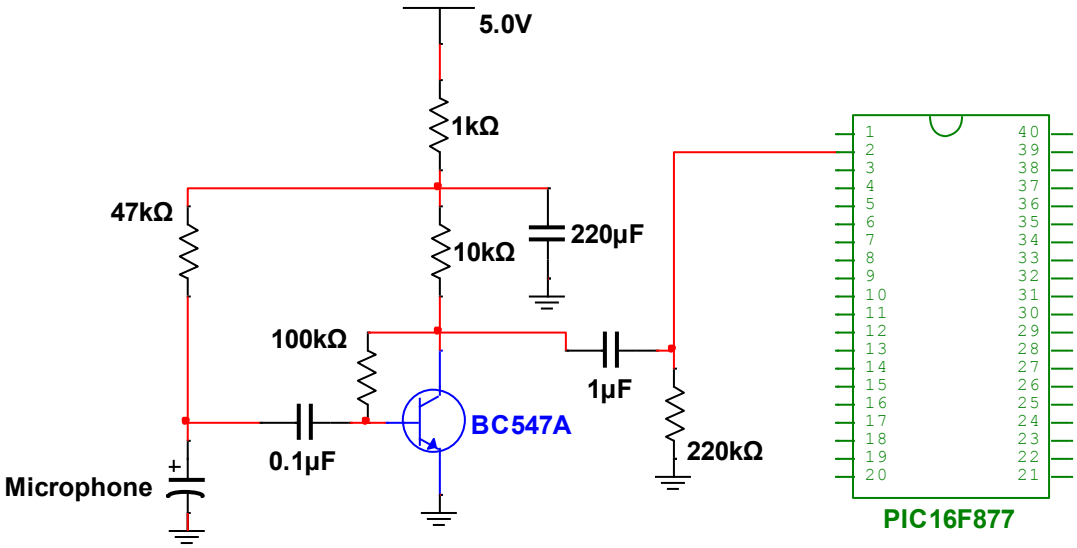
2.2 – Μικρόφωνο και προενισχυτής μικροφώνου

Μία από τις βασικότερες διατάξεις του κυκλώματος ήταν αυτή του μικροφώνου και του προενισχυτή του. Υλοποιήθηκε με στόχο την ενίσχυση της τάσης εισόδου του μικροφώνου για να μπορέσει να γίνει αντιληπτή από τον μικροελεγκτή. Στην εικόνα 10 θα προστεθεί μία φωτογραφία του κυκλώματος προενίσχυσης μικροφώνου.



Εικόνα 10 – Φωτογραφία του προενισχυτή μικροφώνου

Η συλλογή του ήχου πραγματοποιείται με ένα πυκνωτικό μικρόφωνο το οποίο, σύμφωνα με τις μεταβολές πίεσης που προκαλούνται από τα ηχητικά κύματα, μεταβάλει την χωρητικότητα του ενσωματωμένου πυκνωτή που εμπεριέχει. Με αυτόν τον τρόπο γίνεται η μετατροπή των ηχητικών κυμάτων σε ηλεκτρικές ταλαντώσεις. Η επιλογή αυτού του τύπου μικροφώνου έγινε για τους εξής βασικούς λόγους. Πρώτον, έχει μικρό μέγεθος και εύκολα μπορεί να τοποθετηθεί πάνω στο ράστερ δεύτερον, δέχεται τα ηχητικά κύματα από πολλές κατευθύνσεις χωρίς την ανάγκη να μιλάμε απευθείας σε αυτό τρίτον, είναι αρκετά ευαίσθητο, ποιοτικό και επιπλέον πολύ οικονομικό. Η τάση που παράγει ένα πυκνωτικό μικρόφωνο είναι ιδιαίτερα μικρή (max. : 25mV), τάση η οποία δεν μπορεί να γίνει αντιληπτή από τον μικροελεγκτή. Για τον λόγο αυτό γίνεται χρήση του προενισχυτή μικροφώνου. Πριν την ανάλυση του κυκλώματος στην εικόνα 11 θα προστεθεί το κυκλωματικό σχέδιο του προενισχυτή μικροφώνου.



Εικόνα 11 – Κυκλωματικό σχέδιο του προενισχυτή μικροφώνου

Ο προενισχυτής μικροφώνου που χρησιμοποιήθηκε είναι ένας ενισχυτής κοινού εκπομπού (CE – Common Emitter) που ενισχύει μικρά σήματα με στόχο την είσοδό τους σε άλλες βαθμίδες. Για την υλοποίηση του προενισχυτή αυτού χρησιμοποιήθηκε ένα τρανζίστορ (BC547), αντιστάσεις και πυκνωτές. Το BC547 είναι ένα NPN διπολικό τρανζίστορ σε συνδεσμολογία κοινού εκπομπού. Τα NPN τρανζίστορ σε συνδεσμολογία κοινού εκπομπού μπορούν εφόσον χρησιμοποιηθούν στην ενεργό περιοχή να λειτουργήσουν ως ενισχυτές σήματος.

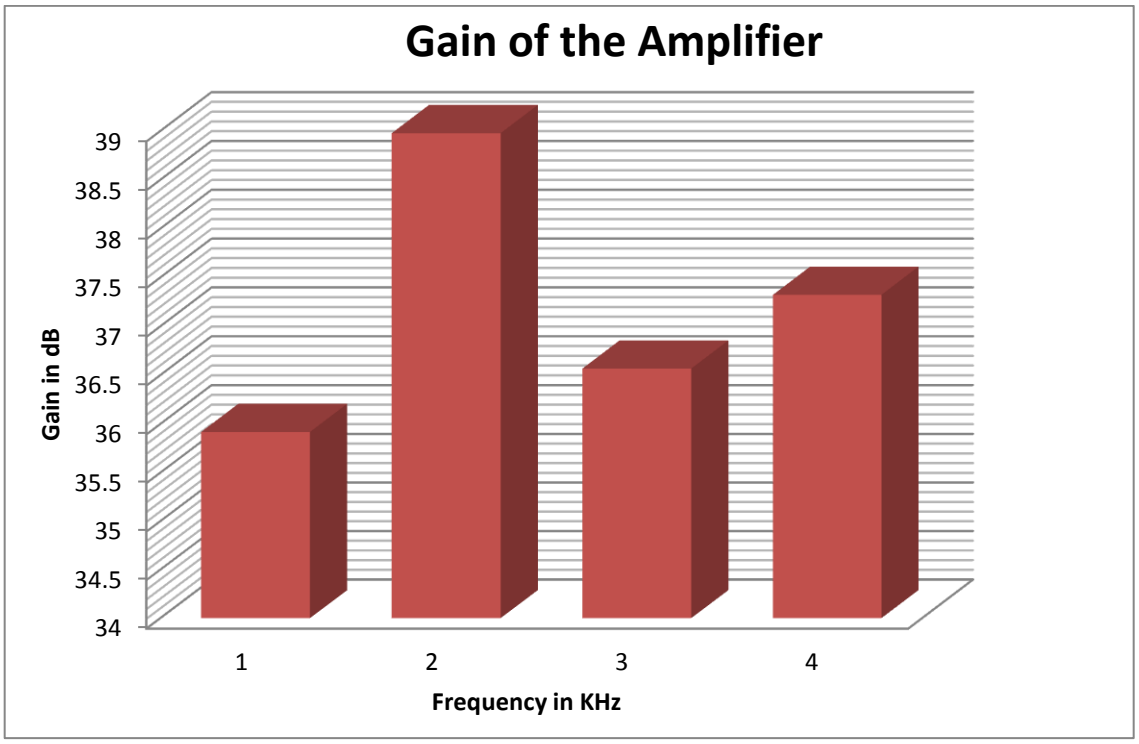
Έτσι οποιοδήποτε μικρό σήμα εφαρμοστεί στην βάση μπορεί να ενισχυθεί έχοντας τον εκπομπό γειωμένο. Για να λειτουργήσει το τρανζίστορ στην ενεργό περιοχή πρέπει η επαφή βάσης εκπομπού να είναι πολωμένη ορθά ενώ η επαφή βάσης συλλέκτη να είναι πολωμένη ανάστροφα.

Στο κύκλωμα γίνεται η σύνδεση του ενός άκρου του μικροφώνου με την γείωση ενώ το άλλο συνδέεται με μία αντίσταση στα 5V. Η χρήση της αντίσταση γίνεται για να είναι εφικτή η σταδιακή φόρτιση του πυκνωτή στο μικρόφωνο και όχι άμεσα. Το ίδιο άκρο του μικροφώνου το συνδέουμε και στην βάση του τρανζίστορ κάνοντας χρήση ενός πυκνωτή ως AC σύζευξη για να έχουμε στην είσοδο της βάσης την πραγματική κυματομορφή. Ο εκπομπός του τρανζίστορ συνδέεται στην γείωση ενώ ο συλλέκτης μέσω μια αντίστασης συνδέεται με τα 5V. Με τον τρόπο αυτό ο συλλέκτης κάνει pull-up την τάση ενώ με τον εκπομπό γειωμένο επιτυγχάνουμε το pull-down της τάσης. Αυτό όμως δεν είναι αρκετό επειδή στην βάση έχουμε το AC σήμα το οποίο εάν το ενισχύσουμε με αυτόν τον τρόπο θα "κόβαμε" την κυματομορφή και θα ενισχύονταν μόνο οι τιμές πάνω από το μηδέν και όχι αυτές κάτω από το μηδέν. Για τον λόγο αυτό προσθέτουμε μία αντίσταση μεταξύ της βάσης και του συλλέκτη για να μπορέσει να γίνει μετατόπιση του σήματος πάνω από τον άξονα του χ και να γίνει ενίσχυση όλης της κυματομορφής. Συνεπώς υλοποιώντας αυτή την συνδεσμολογία επιτυγχάνεται η ενίσχυση του σήματος. Πριν συνδέσουμε την έξοδο του προενισχυτή με την είσοδο του μικροελεγκτή προσθέτουμε ένα πυκνωτή για να έχουμε ένα καθαρό σήμα γύρω από το μηδέν.

Το τελικό αποτέλεσμα του προενισχυτή που σχεδιάσαμε είναι ικανοποιητικό για τις ανάγκες της εργασίας αυτής. Έπειτα από μετρήσεις που έγιναν στον παλμογράφο, μετρήθηκε τάση εισόδου περίπου στα 20mV και στην έξοδο είχαμε ενισχυμένη τάση στα 1,2V. Ο λόγος τάσης εξόδου/εισόδου είναι $V_{OUT}/V_{IN} = 62.5$ και η απολαβή του κυκλώματος υπολογίζοντάς την με τον τύπο $20\log_{10}\left(\frac{V_{OUT}}{V_{IN}}\right)$ είναι 36dB. Με την ενίσχυση αυτή το σήμα μας οδηγείται στη είσοδο του μικροελεγκτή, είναι αρκετά ισχυρό και καθίσταται δυνατή η δειγματοληψία του. Στον πίνακα 2 αποτυπώνεται η απολαβή του προενισχυτή σε σχέση με συγκεκριμένες συχνότητες που επιλέχθηκαν. Στην εικόνα 12 γράφημα του πίνακα αυτού.

Frequency in KHz	Gain in dB
1	35,9176
2	38,97695
3	36,56636
6	37,32213

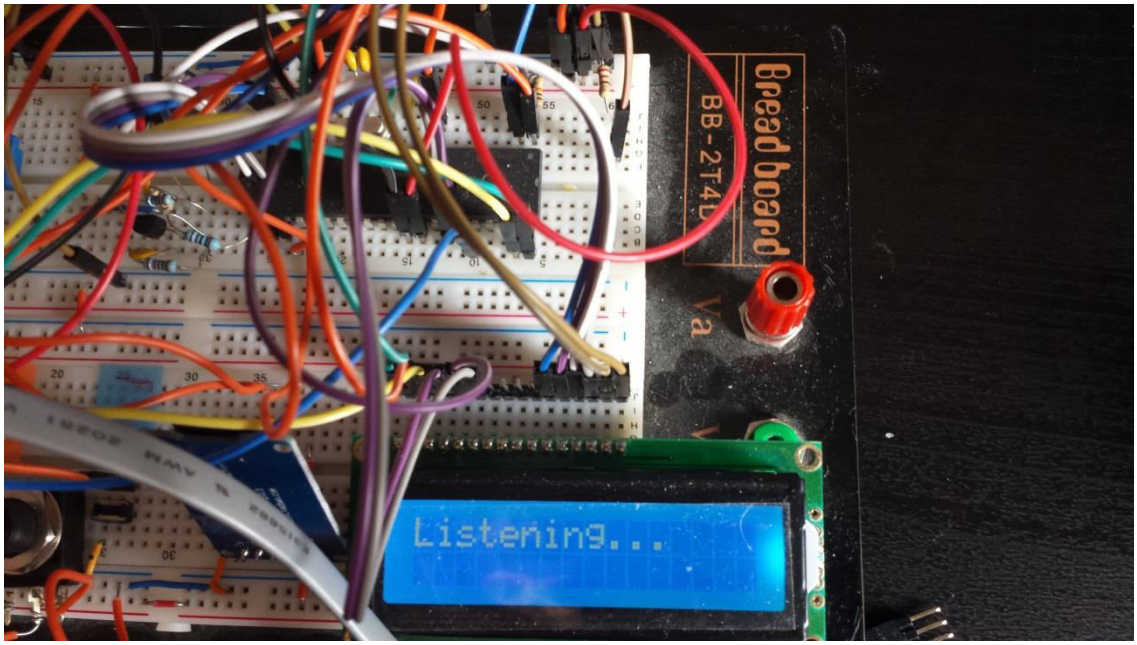
Πίνακας 2 – Απολαβή προενισχυτή



Εικόνα 12 – Γράφημα απολαβής ενισχυτή

2.3 – Σύνδεση μικροελεγκτή με οθόνη LCD

Στο σημείο αυτό θα αναλύσουμε την σύνδεση του μικροελεγκτή με την οθόνη, στην οποία θα έχουμε την δυνατότητα να παρακολουθούμε την κατάσταση της συσκευής ηχογράφησης. Για την πλήρη κατανόηση της συνδεσμολογίας θα πρέπει να αναφερθούν κάποια βασικά τεχνικά χαρακτηριστικά των οθονών LCD (Liquid Crystal Display) εμπλουτισμένα με την εικόνα του κυκλωματικού σχεδίου της διασύνδεσης του μικροελεγκτή με την οθόνη. Στην εικόνα 13 μία φωτογραφία της LCD.



Εικόνα 13 – Φωτογραφία της οθόνης LCD

Η οθόνη LCD 16x2 που θα χρησιμοποιήσουμε είναι μία οθόνη υγρών κρυστάλλων με 16 χαρακτήρες μήκος και 2 γραμμές. Υπάρχουν 2 βασικοί τρόποι συνδεσμολογίας της οθόνης με τον μικροελεγκτή και αυτοί είναι η 4-bit διαμόρφωση και η 8-bit διαμόρφωση. Η βασική διαφορά στους δύο αυτούς τρόπους συνδεσμολογίας είναι ότι η 4-bit διαμόρφωση χρησιμοποιεί 4 ακροδέκτες για την μεταφορά των δεδομένων ενώ η 8-bit διαμόρφωση χρησιμοποιεί και τους 8 ακροδέκτες της οθόνης. Βέβαια η 4-bit διαμόρφωση χρειάζεται να κάνει δύο μεταφορές για την εμφάνιση ενός χαρακτήρα ή την αποστολή μίας εντολής, ενώ στην 8-bit διαμόρφωση χρειάζεται μόνο μία. Αυτό έχει και ως αποτέλεσμα η διαμόρφωση 4-bit να χρειάζεται διπλάσιο χρόνο για την εκτέλεση μιας εντολής. Στην παρούσα εργασία θα δούμε την υλοποίηση της 4-bit διαμόρφωσης κυρίως επειδή οι ανάγκες μας για ταχύτητα ουσιαστικά είναι μηδαμινές ενώ ταυτόχρονα ο σκοπός της οθόνης είναι αποκλειστικά να μας δίνει οπτική αναφορά της κατάστασης της συσκευής. Τέλος με την επιλογή αυτή μειώθηκε η πολυπλοκότητα του κυκλώματος λόγω της χρήσης μόνο των τεσσάρων καλωδίων για την μεταφορά δεδομένων και εντολών.

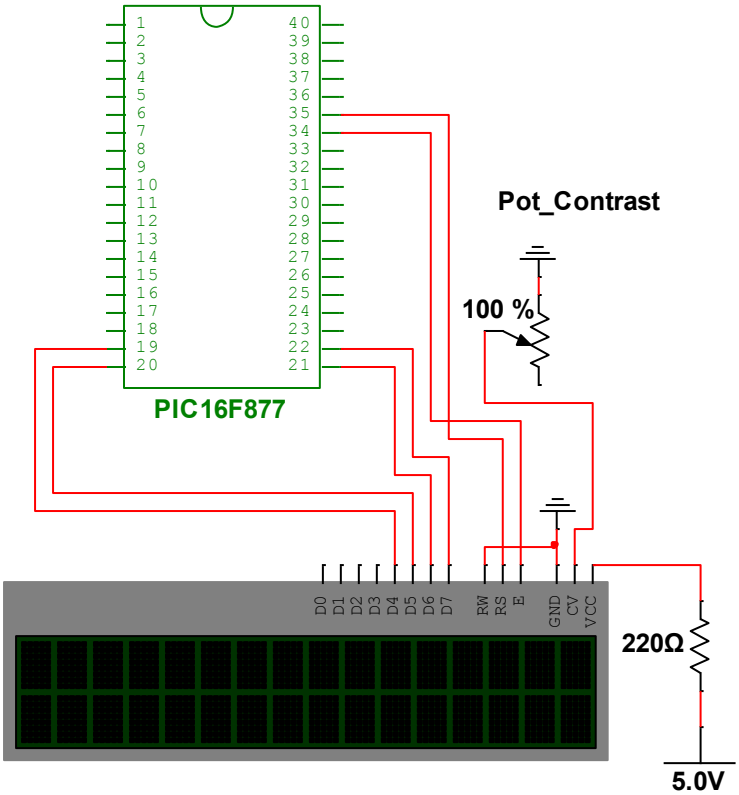
Στον πίνακα 3 αποτυπώνονται οι συνδεσμολογίες που υλοποιήθηκαν μεταξύ του μικροελεγκτή και της οθόνης LCD.

Pins No	LCD Pins	Circuit	Explanation
1	VSS	Gnd	Ground
2	VDD	5V	VDD
3	VE	Pot	Contrast
4	RS	RB2	Register Select
5	R/W	Gnd	Read/Write LCD
6	EN	RB1	Execute
11	DB4	RD0	Data Pin4
12	DB5	RD1	Data Pin5
13	DB6	RD2	Data Pin6
14	DB7	RD3	Data Pin7
15	LED+	5V	BackLight +
16	LED-	Gnd	BackLight -

Πίνακας 3 – Πίνακας συνδεσμολογιών της LCD

Η πρώτη παρατήρηση είναι ότι για δεδομένα χρησιμοποιούμε μόνο τους τέσσερις ακροδέκτες 11 - 14 (DB4 – DB7) από τους οκτώ όπως έγινε ανάλυση στην προηγούμενη παράγραφο και τα συνδέουμε στους ακροδέκτες 19 – 22 (RD0 – RD3) του μικροελεγκτή όπου από εκεί η LCD θα λαμβάνει τα δεδομένα και τις εντολές. Οι ακροδέκτες με νούμερο 2 και 15 συνδέονται στα 5v και αφορούν την παροχή τάσης στην οθόνη (VDD) και πίσω φωτισμός (LED+) ενώ οι ακροδέκτες 1 (VSS) και 16 (LED-) γειώνονται. Ο στόχος του ακροδέκτη με νούμερο 3 (VE) είναι, με χρήση του ποτενσιόμετρου με το οποίο το έχουμε συνδέσει, η ρύθμιση της αντίθεσης της οθόνης δηλαδή το πόσο έντονα θα είναι τα γράμματα που θα εμφανίζονται σε σχέση με τον πίσω φωτισμό. Για την επεξήγηση του ακροδέκτη με νούμερο 4 (RS) πρέπει να επισημάνουμε ότι η LCD έχει 2 καταχωρητές. Είναι ο καταχωρητής εντολών (Command Register) και ο καταχωρητής δεδομένων (Data Register). Όταν στέλνουμε τις διάφορες εντολές για την παραμετροποίηση της οθόνης αυτές κατευθύνονται στον καταχωρητή εντολών όπου επεξεργάζονται για να πραγματοποιηθεί έπειτα η εκτέλεσή τους. Στον καταχωρητή δεδομένων στέλνουμε τα δεδομένα, ουσιαστικά αυτό που θέλουμε να αποτυπωθεί στην οθόνη. Ο ακροδέκτης με νούμερο 4 είναι αυτός ο οποίος καθορίζει αν θα επιλεχθεί ο καταχωρητής εντολών ή αν θα επιλεχθεί ο καταχωρητής δεδομένων. Όταν ο ακροδέκτης RS είναι ίσος με το λογικό μηδέν (RS = 0) τότε θα επιλεχθεί ο καταχωρητής εντολών ενώ όταν είναι ίσος με το λογικό ένα (RS = 1) τότε θα επιλεχθεί ο καταχωρητής δεδομένων. Η επιλογή της τιμής αυτού του καταχωρητή

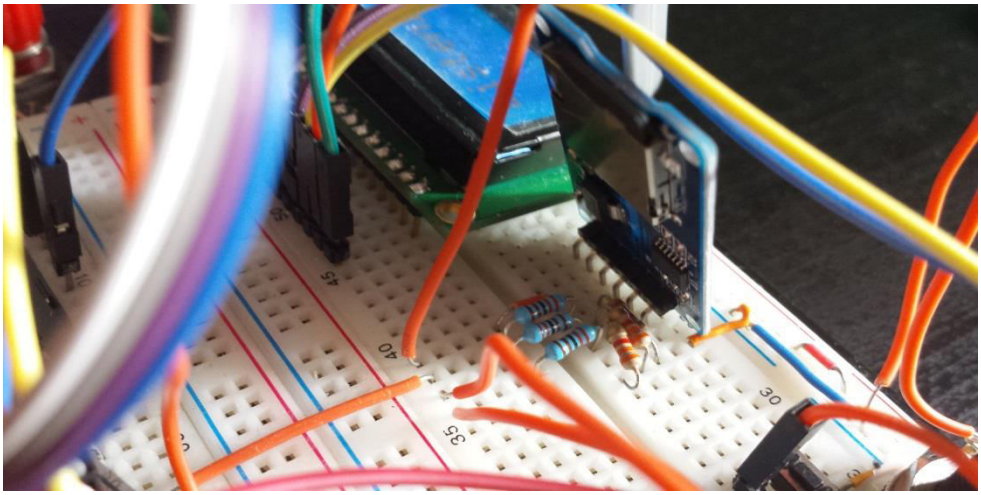
θα γίνει μέσω κώδικα του οποίου η πλήρης ανάλυση πραγματοποιείται στο επόμενο κεφάλαιο. Ο ακροδέκτης νούμερο 5 (R/W) είναι αυτός που καθορίζει αν θα γράψουμε δεδομένα στην LCD ή αν θα διαβάσουμε από αυτή. Όταν ο ακροδέκτης RW είναι ίσος με λογικό μηδέν (RS = 0) τότε δίνουμε εντολή ότι θέλουμε να γράψουμε δεδομένα στην οθόνη ενώ αν ο RS είναι ίσος με το λογικό ένα (RS = 1) τότε αυτό σημαίνει ότι θα διαβάσουμε δεδομένα από την LCD. Στο παρόν project επειδή η χρήση της LCD είναι αποκλειστικά για την οπτικοποίηση της κατάστασης στην οποία βρίσκεται σε κάθε στιγμή η συσκευή, γειώνουμε τον ακροδέκτη νούμερο 5 κάνοντας την LCD αποκλειστικά αποδέκτη δεδομένων. Η χρήση του ακροδέκτη νούμερο 6 (EN) αφορά, εφόσον έχουμε δώσει τιμές στους καταχωρητές (Command & Data Registers) και έχουμε δώσει την απαραίτητη τιμή στον ακροδέκτη 5 (R/W), την αποστολή των εντολών ή δεδομένων που στέλνουμε στην LCD. Για να γίνει η αποστολή των εντολών η δεδομένων που βρίσκονται στην γραμμή δεδομένων (data line) αρχικά ορίζεται ο ακροδέκτης 6 ίσος με το λογικό μηδέν (EN = 0) και μόλις πρέπει να γίνει η εκτέλεση της εντολής τον κάνουμε ίσο με το λογικό ένα (EN = 1) για μερικά χιλιοστά του δευτερολέπτου (ms). Μετά επιστρέφουμε στην αρχική κατάσταση (δλδ. EN = 0). Στην εικόνα 14 έχουμε το κυκλωματικό σχέδιο που απεικονίζει τις συνδεσμολογίες του μικροελεγκτή με την LCD Στο $3^{ο}$ κεφάλαιο θα γίνει εκ νέου αναφορά στην διαδικασία εκτέλεσης λόγω της ύπαρξης υπορουτίνας που αποσκοπεί στην υλοποίηση αυτής της διαδικασίας.



Εικόνα 14 – Κυκλωματικό σχέδιο της οθόνης LCD

2.4 - Σύνδεση μικροελεγκτή με Κάρτα SD

Στο σημείο αυτό θα αναλυθεί, εάν όχι η σημαντικότερη, μία από τις σημαντικότερες βαθμίδες που υλοποιήθηκαν, δηλαδή η σύνδεση του μικροελεγκτή με την κάρτα SD. Για να υλοποιηθεί η βαθμίδα αυτή έγινε χρήση ενός υποδοχέα κάρτας SD ο οποίος μέσω του πρωτοκόλλου SPI επικοινωνεί με τον μικροελεγκτή. Στην εικόνα 15 μία φωτογραφία του υποδοχέα και του κυκλώματος της κάρτας SD.



Εικόνα 15 – Φωτογραφία του υποδοχέα κάρτα SD

Το SPI (Serial Peripheral Interface) είναι ένα από τα πιο ευρέως χρησιμοποιήσιμα πρωτόκολλα επικοινωνίας για την διασύνδεση περιφερειακών μονάδων και μικροελεγκτών μεταξύ τους. Είναι ένα σύγχρονο σειριακό πρωτόκολλο επικοινωνίας μεταξύ ολοκληρωμένων με την δυνατότητα ταυτόχρονης μετάδοσης και λήψης δεδομένων. Χρησιμοποιείται και υλοποιείται σε εφαρμογές που κάνουν χρήση ενσωματωμένων συστημάτων. Δύο είδη συσκευών μπορούν να συνδεθούν στο SPI, μία Master και μία ή περισσότερες Slave συσκευές. Για την υλοποίηση του πρωτοκόλλου κάνουμε χρήση τεσσάρων διαύλων (four-wire serial bus). Οι τρεις δίαυλοι επικοινωνίας είναι κοινοί για όλες τις slave συσκευές και μόνο ο τέταρτος είναι ξεχωριστός για την κάθε συσκευή.

Οι τρεις δίαυλοι είναι :

- MOSI (Master Out Slave In) – αφορά την μετάδοση δεδομένων από τον Master στον Slave
- MISO (Master In Slave Out) – αφορά την μετάδοση των δεδομένων από τον Slave στον Master
- SCLK (Serial CLocK) – αποτελεί την γραμμή χρονισμού των διαύλων MOSI και MISO

Ο τέταρτος δίαυλος SS (Slave Select) είναι αυτός με τον οποίο ο Master επιλέγει με ποιά συσκευή θέλει να επικοινωνήσει κάθε φορά. Η επικοινωνία ελέγχεται από την συσκευή Master η οποία πρέπει να ρυθμιστεί κατάλληλα σε μία συχνότητα που δεν ξεπερνά την συχνότητα λειτουργίας των συσκευών Slave. Κάθε φορά που παράγεται ένα σήμα χρονισμού στην SCLK τότε γίνεται μία ταυτόχρονη εκπομπή και λήψη δεδομένων μεταξύ της Master και Slave συσκευής. Ουσιαστικά τα δεδομένα ολισθαίνουν πάνω στις γραμμές MOSI και MISO σε κάθε πτώση ή άνοδο του παλμού του ρολογιού. Αυτή η αμφίδρομη επικοινωνία λαμβάνει μέρος ακόμη και στην περίπτωση που η εφαρμογή μας απαιτεί αποκλειστικά μονόδρομη μεταφορά δεδομένων.

Με την ολοκλήρωση της θεωρητική προσέγγισης του πρωτοκόλλου SPI θα αναφερθούμε συγκεκριμένα στο κύκλωμα που υλοποιήθηκε. Ο μικροελεγκτής στην λειτουργία SPI μπορεί να κάνει ταυτόχρονη εκπομπή και λήψη 8-bit

δεδομένων. Για την λειτουργία του SPI υπάρχουν τέσσερεις καταχωρητές. Οι καταχωρητές αυτοί είναι οι :

- MSSP Control Register (SSPCON) – καταχωρητής ελέγχου στην λειτουργία SPI
- MSSP Status Register (SSPSTAT) – καταχωρητής κατάστασης στην λειτουργία SPI
- Serial Receive/Transmit Buffer Register (SSPBUF) – καταχωρητής εγγραφής ή ανάγνωσης δεδομένων
- MSSP Shift Register (SSPSR) – καταχωρητής για την ολίσθηση δεδομένων είτε εντός του μικροελεγκτή είτε εκτός

Για την ορθή λειτουργία του SPI πρέπει να αρχικοποιηθούν τα διάφορα bit αυτών των καταχωρητών για να μπορέσει να επιτευχθεί η σωστή επικοινωνία των συσκευών μεταξύ τους. Θα γίνει ανάλυση των αρχικοποιήσεων του μικροελεγκτή στο κεφάλαιο με τον κώδικα όπου ορίζονται οι τιμές των καταχωρητών.

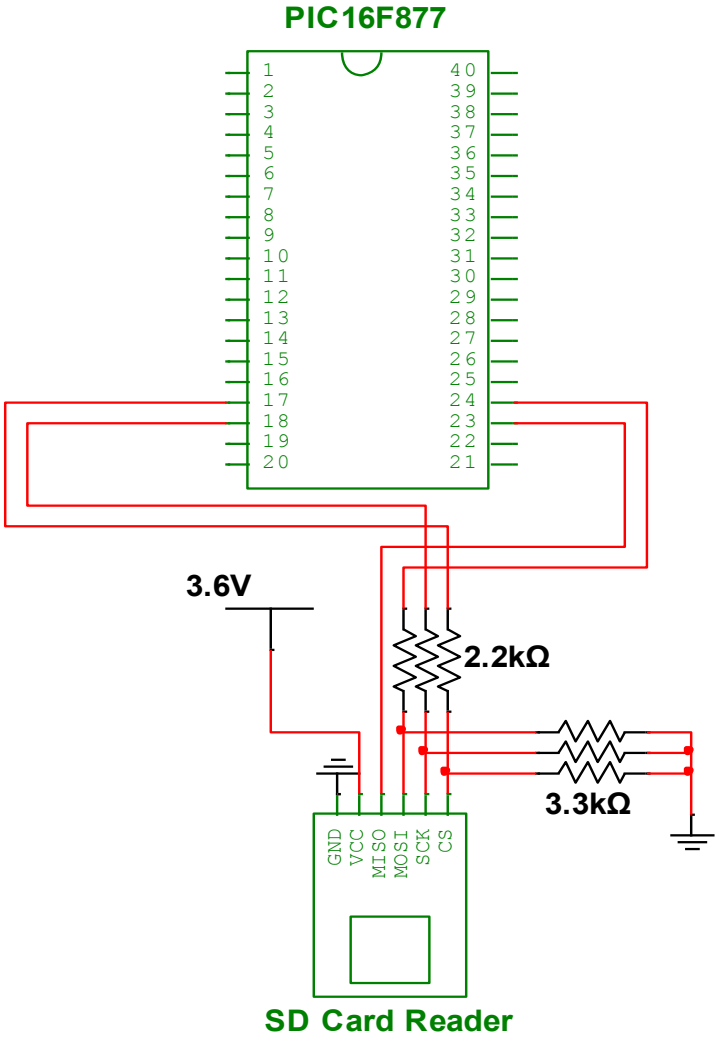
Στον πίνακα 4 δίνονται οι συνδεσμολογίες που υλοποιήθηκαν μεταξύ του μικροελεγκτή και της κάρτας SD.

SD Pins	Circuit	Explanation
VSS	Gnd	Ground
VDD	5V	VDD
SS	RC2 (CS)	Chip Select
CLK	RC3 (CLK)	Serial Clock
MISO	RC4 (SDI)	Master In Slave Out
MOSI	RC5 (SDO)	Master Out Slave In

Πίνακας 4 – Πίνακας συνδεσμολογιών της κάρτας SD

Οι συνδέσεις που γίνονται είναι αρκετά κατανοητές επειδή ο μικροελεγκτής λόγω της διατιθέμενης αρχιτεκτονικής, έχει ήδη αναθέσει τις τρεις βασικές γραμμές (SCLK, MISO,MOSI) στους τρεις ακροδέκτες (Pin 18, Pin 23, Pin24) της PORTC, άρα τα RC3, RC4 και RC5 συνδέονται με τους ακροδέκτες της κάρτας SD CLK, MISO και MOSI αντίστοιχα. Χρησιμοποιείται ο ακροδέκτης RC2 (Pin 17) ως επιλογέας συσκευής SS (Slave Select) και το συνδέουμε με το αντίστοιχο ακροδέκτη της κάρτας SD CS (Chip Select). Σε αυτό το σημείο πρέπει να τονίσουμε ότι η κάρτα SD λειτουργεί στην τάση 3.6V ενώ ο μικροελεγκτής στα 5V.

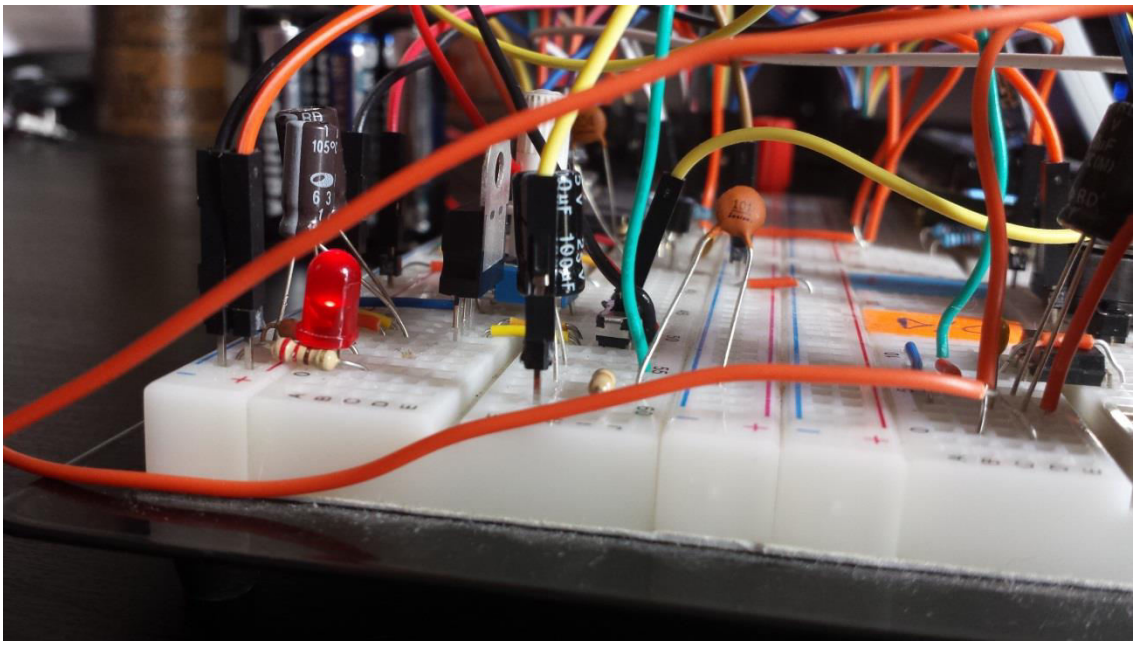
Για τον λόγο αυτό υπάρχει η ανάγκη ύπαρξης ενός διαιρέτη τάσης, ο οποίος υλοποιείται με αντιστάσεις, για να εξομαλύνει την διαφορά αυτή. Ο διαιρέτης τάσης θα τοποθετηθεί στους ακροδέκτες CS, CLK και MOSI των οποίων η τάση πρέπει να εξομαλυνθεί. Ο ακροδέκτης MISO συνδέεται απευθείας με την κάρτα SD. Στον πίνακα παρατηρούμε ότι το VDD της κάρτας συνδέεται με τα 5V του κυκλώματος κάτι το οποίο έρχεται σε αντιπαράθεση με την μέγιστη τάση που δέχεται η κάρτα SD (3.6V). Στην συγκεκριμένη περίπτωση πρέπει να τονισθεί ότι το ολοκληρωμένο που χρησιμοποιούμε για την ανάγνωση της κάρτας SD έχει και αυτό διαιρέτη τάσης στο κύκλωμα τροφοδοσίας του. Εάν δεν υπήρχε διαιρέτης τάσης μέσα στο ολοκληρωμένο θα έπρεπε και στην τροφοδοσία της κάρτας SD να γίνει υλοποίηση ενός αντίστοιχου κυκλώματος. Στην εικόνα 16 το κυκλωματικό σχέδιο της κάρτας SD και την διασύνδεσή της με τον μικροελεγκτή.



Εικόνα 16 – Κυκλωματικό σχέδιο της κάρτας SD

2.5 – Παραγωγή PWM με μικροελεγκτή

Για να οδηγηθεί ο ήχος προς τον ενισχυτή ακουστικού σήματος πρέπει πρώτα τα αποθηκευμένα δεδομένα στην κάρτα SD από ψηφιακά να μετατραπούν σε αναλογικά. Για να επιτευχθεί αυτό είναι απαραίτητη η ύπαρξη ενός μετατροπέα ψηφιακού σήματος σε αναλογικό (DAC – Digital to Analog Conversion). Στον μικροελεγκτή PIC16F877 δεν υπάρχει υλοποιημένη μέσα στο κύκλωμα κάποια βαθμίδα DAC η οποία να είναι σχεδιασμένη αποκλειστικά για αυτό το σκοπό. Μία λύση θα ήταν να αγοραστεί μία άλλη διάταξη DAC του εμπορίου και να συνδεθεί με τον μικροελεγκτή. Αυτό βέβαια θα απαιτούσε να δεσμευτούν άλλοι 3-4 ακροδέκτες για να γίνει η σύνδεση μέσω SPI και να γραφτούν επιπλεόν ρουτίνες κώδικα για την επίτευξη της επικοινωνίας. Για τον λόγο αυτό επιλέχθηκε να γίνει χρήση του PWM (Pulse Width Modulation) που είναι ήδη σχεδιασμένος μέσα στο μικροελεγκτή μας. Έτσι με την χρήση του PWM και ενός χαμηλοπερατού φίλτρου, γίνεται υλοποίηση ενός DAC με τον οποίο γίνεται επίτευξη του επιθυμητού αποτελέσματος. Στην εικόνα 17 μία φωτογραφία του φίλτρου που υλοποιήθηκε.

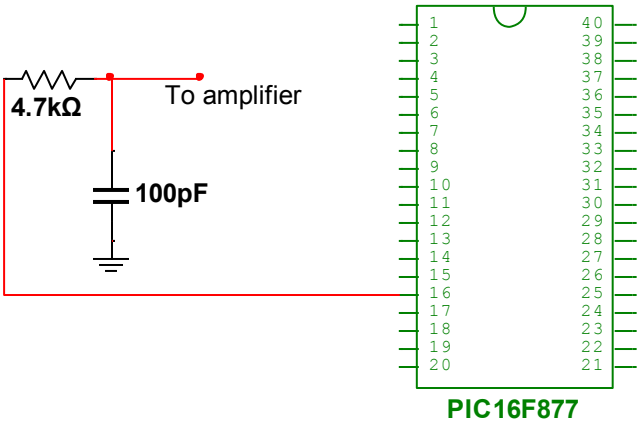


Εικόνα 17 – Φωτογραφία του φίλτρου RC

Η PWM (Διαμόρφωση εύρους παλμών) είναι μία τεχνική διαμόρφωσης που χρησιμοποιείται για την κωδικοποίηση δεδομένων για μετάδοση. Χρησιμοποιείται καθημερινά σε πάρα πολλές εφαρμογές κυρίως σε εφαρμογές που αφορούν τον

έλεγχο της ισχύος που παρέχεται σε ηλεκτρικές συσκευές (π.χ. κυρίως σε μοτέρ). Στο παρόν project θα εκμεταλλευτεί η δυνατότητα αυτής της τεχνικής διαμόρφωσης παράγοντας το αναλογικό σήμα που επιθυμείται, διαμορφώνοντας το ψηφιακό σήμα και εκπέμποντάς το πλέον νέο διαμορφωμένο σήμα σε ένα χαμηλοπερατό φίλτρο για την αφαίρεση των αρμονικών που παράγονται κατά την διαδικασία διαμόρφωσης. Στην τεχνική PWM υπάρχει πάντα μία σταθερή περίοδο στην οποία εναλλάσσεται ο χρόνος της ενεργής κατάστασης σε σχέση με τον χρόνο της ανενεργής κατάστασης. Αυτό που είναι σημαντικό όμως δεν είναι οι απόλυτες τιμές των δύο καταστάσεων αλλά η σχέση που υπάρχει μεταξύ τους. Η σχέση αυτή λέγεται duty cycle (κύκλος καθηκόντων). Υποθέτοντας ότι ο duty cycle είναι του ποσοστού 50%, δηλαδή για την μισή περίοδο στην ενεργή κατάσταση και την άλλη μισή περίοδο στην ανενεργή κατάσταση, έχοντας τάση εισόδου 5V τότε στην έξοδο θα υπάρχει μια ομαλοποιημένη 2.5V τάση. Με τον τρόπο αυτό τα ψηφιακά αποθηκευμένα δεδομένα μετατρέπονται σε αναλογική μορφή έχοντας πλέον την δυνατότητα να γίνει αναπαραγωγή του αποθηκευμένου ήχου.

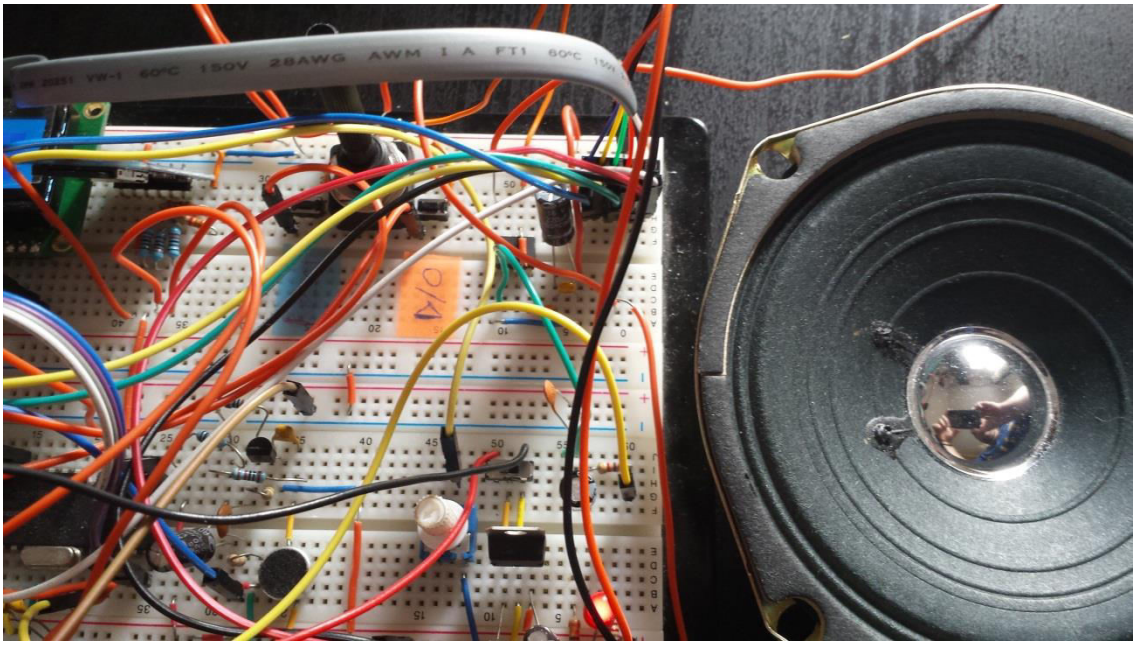
Όσον αφορά το κύκλωμα, αλλά και για τον κώδικα που θα αναλυθεί στο επόμενο κεφάλαιο, για την λειτουργία του DAC δεν πραγματοποιείται κάποια ιδιαίτερη συνδεσμολογία. Όλη η διαδικασία γίνεται στον μικροελεγκτή και το διαμορφωμένο κατά PWM σήμα λαμβάνεται στον ακροδέκτη 16. Έπειτα οδηγείται σε ένα χαμηλοπερατό φίλτρο RC. Η αντίσταση (R) του RC φίλτρου είναι συνδεδεμένη σε σειρά με την έξοδο ενώ ο πυκνωτής (C) παράλληλα σε σχέση με την έξοδο. Ο πυκνωτής δρα σαν αντίσταση για την χαμηλές συχνότητες και τις οδηγεί στην έξοδο. Στις υψηλότερες συχνότητες η αντίσταση του πυκνωτή σταματά να υφίσταται και έτσι λειτουργεί σαν βραχυκύκλωμα και οδηγεί τις συχνότητες στην γείωση αποκόβοντας τες. Για την υλοποίηση του RC φίλτρου κάναμε χρήση μίας αντίστασης 4,7k και ενός πυκνωτή 100pf. Με τις τιμές αυτές έχουμε ένα ικανοποιητικό ακουστικό αποτέλεσμα. Στην εικόνα βλέπουμε 18 το κυκλωματικό σχέδιο του φίλτρου RC που υλοποιήθηκε για την λειτουργία της συσκευής.



Εικόνα 18 – Κυκλωματικό σχέδιο του φίλτρου RC

2.6 – Ακουστικός ενισχυτής

Στο τελευταίο μέρος του κεφαλαίου αυτού θα γίνει επεξήγηση του ακουστικού ενισχυτή που χρησιμοποιήθηκε για την ενίσχυση του ήχου και την οδήγησή του στο μεγάφωνο. Για την υλοποίησή του χρησιμοποιήθηκε το ολοκληρωμένο κύκλωμα ενίσχυσης σήματος TDA7052. Επιλέχθηκε το συγκεκριμένο ολοκληρωμένο και λόγω της απλότητάς του στην συνδεσμολογία αλλά και λόγω της πολύ καλής ενίσχυσης που έχει στην έξοδό του. Στην εικόνα 19 απεικονίζεται το κύκλωμα του ακουστικού ενισχυτή.



Εικόνα 19 – Φωτογραφία του ακουστικού ενισχυτή

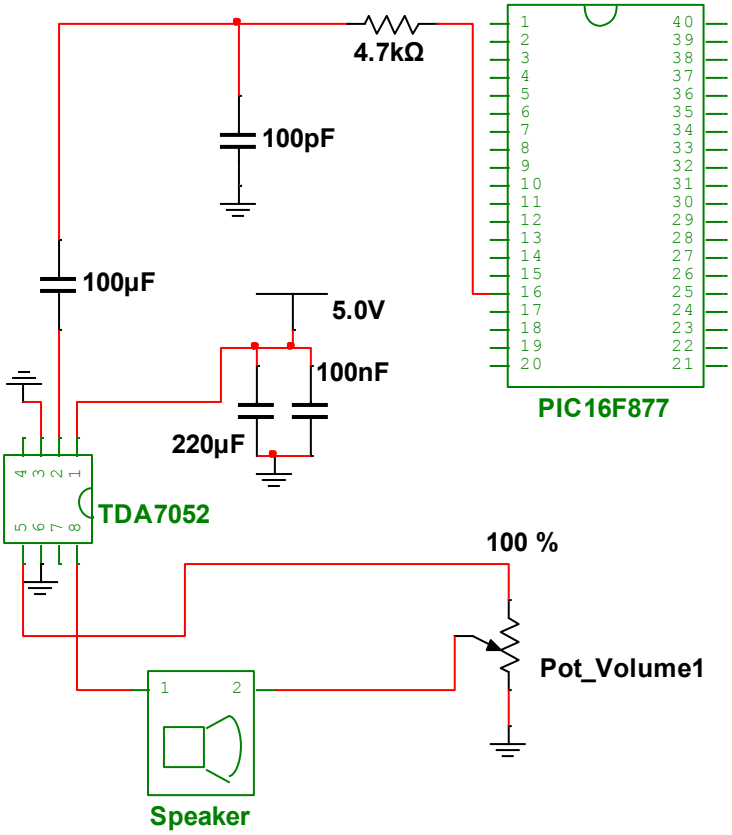
Στον πίνακα 5 αποτυπώνονται οι συνδεσμολογίες που υλοποιήθηκαν για την σωστή λειτουργία του ενισχυτή.

Pins No	TDA7052 Pins	Circuit	Explanation
1	VDD	5V	VDD
2	IN	RC Filter Output	Signal Input
3	GND1	Gnd	Ground
4	N.C.	N.C.	Not Connected
5	OUT1	Pot to SPKR	Potentiometer to Speaker
6	GND2	Gnd	Ground
7	N.C.	N.C.	Not Connected
8	OUT2	SPKR	Speaker

Πίνακας 5 – Πίνακας συνδεσμολογιών ακουστικού ενισχυτή

Η συνδεσμολογία του TDA7052, κατά κύριο λόγο, έγινε ακολουθώντας το φύλλο δεδομένων όπου υπάρχει αντίστοιχο κυκλωματικό σχέδιο της εικόνας 20. Ο ακροδέκτης με νούμερο 1 συνδέεται με την τροφοδοσία ενώ οι ακροδέκτες με νούμερο 3 και 6 συνδέονται με την γείωση. Οι ακροδέκτες νούμερο 4 και 7 δεν είναι συνδεδεμένοι εσωτερικά στο ολοκληρωμένο, συνεπώς δεν θα συνδεθούν πουθενά. Ο ακροδέκτης με νούμερο 2 είναι αυτός στον οποίο οδηγούμε το φιλτραρισμένο ακουστικό σήμα. Πριν το εισάγουμε στον ενισχυτή τοποθετούμε ένα πυκνωτή για να μειώσουμε τον θόρυβο που προκαλείται από το DC σήμα. Οι ακροδέκτες νούμερο 5 και 8 συνδέονται με το ηχείο των 8Ohm. Στον ακροδέκτη με αριθμό 5 προσθέσαμε επιπλέον ένα λογαριθμικό ποτενσιόμετρο με το οποίο κάνουμε τον έλεγχο της έντασης του ήχου.

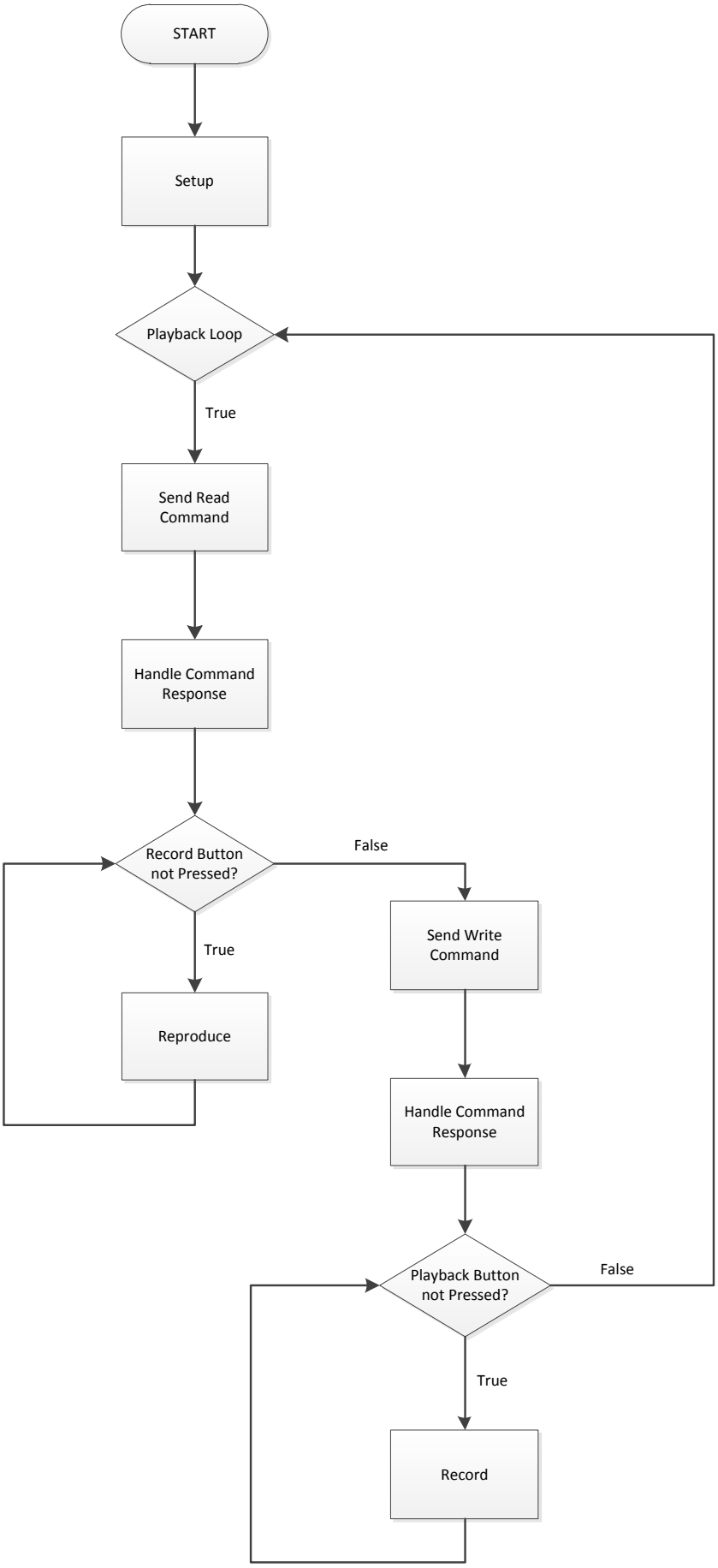
Ο μονοφωνικός ενισχυτής TDA7052 είναι ειδικά σχεδιασμένος για εφαρμογές που χρησιμοποιούν μπαταρίες και επιπλέον στην έξοδό του δίνει 1Watt ισχύ το οποίο είναι πολύ ικανοποιητικό για έναν ενισχυτή τόσο μικρού μεγέθους και απλής υλοποίησης.



Εικόνα 20 – Κυκλωματικό σχέδιο του ακουστικού ενισχυτή

Κεφάλαιο 3^{ο} – Ανάλυση του κώδικα

Στο κεφάλαιο αυτό θα αναλυθεί ο κώδικας που συγγράφηκε για την ορθή λειτουργία της συσκευής και θα δοθεί ένα διάγραμμα ροής του προγράμματος. Ο κώδικας που συγγράφηκε αποτελείται αποκλειστικά από συναρτήσεις από τις οποίες η κάθε μία εκτελεί μια συγκεκριμένη διαδικασία. Στην εικόνα 21 βλέπουμε το διάγραμμα ροής του προγράμματος με τις βασικές λειτουργίες της εφαρμογής.



Εικόνα 21 – Διάγραμμα ροής

3.1 – ADC (Analog to Digital Converter – Μετατροπέας αναλογικού σήματος σε ψηφιακό)

Στην συγκεκριμένη εργασία κάνουμε χρήση του ενσωματωμένου ADC που διαθέτει ο PIC16F877 και χρησιμοποιούμε 8 από τα 10 bit που μας προσφέρει. Χρησιμοποιούμε αυτή την υλοποίηση και για την απλοποίηση της εφαρμογής και λόγω των όχι ιδιαιτέρων απαιτούμενων προδιαγραφών που έχει η συσκευή. Ως τάση αναφοράς δηλώνουμε ως Vref+ την τάση τροφοδοσίας του επεξεργαστή δηλαδή τα 5V και ως Vref- την γείωση. Αυτό σημαίνει ότι θα υπάρχουν 256 διακριτά επίπεδα στο εύρος των 5V τάσης. Άρα η ανάλυση θα είναι 5V/255=0.01960V. Επίσης σημαντικό κομμάτι σε έναν ADC είναι ο χρόνος δειγματοληψίας. Ο χρόνος δειγματοληψίας στον PIC16F877 είναι το άθροισμα του χρόνου απόκτησης και του χρόνου μετατροπής (Acquisition Time + A/D Conversion Time). Ο χρόνος απόκτησης (T_{ACQ}) δίνεται στο φύλλο δεδομένων του PIC και είναι 19,72μs. Στο φύλλο δεδομένων υπάρχει η σημείωση πως ο χρόνος μετατροπής (T_{AD}) πρέπει να είναι τουλάχιστον $12T_{AD}$ για να γίνει μια πλήρη μετατροπή. Για τον υπολογισμό του T_{AD} στο φύλλο δεδομένων μας δίνεται ένας πίνακας στον οποίο, σύμφωνα με το κρύσταλλο που έχουμε (στην παρούσα εργασία 20MHz), επιλέγουμε το ανάλογο T_{AD} (σύμφωνα με τον πίνακα $T_{AD} = 32T_{OSC}$ για 20MHz κρύσταλλο) με στόχο ένα ελάχιστο χρόνο T_{AD} = 1,6μs. Ο T_{OSC} είναι εύκολα υπολογίσιμος έχοντας υπόψη πως ο $T_{OSC}= 1/F_{OSC}$ = 1/20MHz = 0.05μs. Αυτό μας δίνει αυτομάτως τον $T_{AD} = 32T_{OSC}$ = 32*0.05μs = 1.6μs που είναι και ο επιθυμητός ελάχιστος T_{AD} χρόνος. Κάνοντας και τον πολλαπλασιασμό έχουμε ένα συνολικό χρόνο μετατροπής $12T_{AD}$ = 12*1,6μs = 19.2μs. Ολοκληρώνοντας υπολογίζουμε τον συνολικό χρόνο δειγματοληψίας $T_{ACQ} + 12T_{AD}$ = 19.72μs + 19.2μs = 38.92μs ≈ 39μs.

Για τον προγραμματισμό και τον έλεγχο του ADC, που μας παρέχει ο PIC16F877, υπάρχουν 4 καταχωρητές. Οι 2 από τους 4 είναι καταχωρητές ελέγχου. Ο ADCON0 και ο ADCON1 είναι οι καταχωρητές με τους οποίους ρυθμίζουμε τον χρόνο μετατροπής που αναλύσαμε πιο πάνω, την επιλογή των αναλογικών καναλιών, την ρύθμιση των bit των καταχωρητών καθώς και τον έλεγχο της μετατροπής και ενεργοποίησης του μετατροπέα. Έπειτα υπάρχουν οι άλλοι 2 καταχωρητές ο ADRESH και ο ADRESL όπου αποθηκεύεται το

αποτέλεσμα της μετατροπής. Επειδή ο ADC που σχεδιάσθηκε για τον σκοπό της εργασίας αυτής είναι 8-bit, γίνεται χρήση μόνο του ADRESL. Η αρχή και το τέλος της μετατροπής ελέγχεται και διαχειρίζεται από το 3° bit του ADCON0 GO/ $\overline{\text{DONE}}$. Όταν η τιμή του τίθεται ίση με το λογικό ένα (GO = 1) ενεργοποιείται η μετατροπή, έπειτα με την ολοκλήρωσή της γίνεται αυτόματα η απενεργοποίηση από το υλισμικό (GO = 0). Ελέγχοντας συνεχώς το bit αυτό ελέγχεται με ευκολία και η κατάσταση της μετατροπής.

Αρχικά αναλύεται η υπορουτίνα ρύθμισης του ADC, η setup_adc();.

```
/**
 * Sets up the ADC
 */
void setup_adc() {

    TRISA0 = 1;           // PORTA0 Input
    ADCON0 = 0b10000001; // Fosc/32 ADC On
    ADCON1 = 0b10001110; // Port A0 Analog Input Right justified
}
```

Σε αυτή την υπορουτίνα γίνεται η ρύθμιση όλων των παραμέτρων για την σωστή λειτουργία του A/D μετατροπέα. Αρχικά δηλώνεται ότι το 1° pin της PORTA θα είναι είσοδος και συγκεκριμένα είναι η είσοδος για το μικρόφωνο. Οτιδήποτε θα καταγράφει το μικρόφωνο θα εισάγεται στον μικροελεγκτή μέσω του pin αυτού. Έπειτα με τον ADCON0 και τον ADCON1 πραγματοποιείται η ρύθμιση του A/D μετατροπέα. Όπως περιγράφηκε, επιλέγεται ως χρόνος μετατροπής το Fosc/32. Η επιλογή αυτή γίνεται με τον συνδυασμό του ADCON0 και ADCON1. Έπειτα στον ADCON0 επιλέγεται το πρώτο αναλογικό κανάλι από τα οκτώ κανάλια που παρέχει ο μικροελεγκτής και ταυτόχρονα ενεργοποιείται το A/D module. Στον ADCON1 επιλέγεται την ρύθμιση right justified δηλαδή ότι τα 6 MSB (Most Significant Bits) του ADRESH θα διαβαστούν ως μηδέν. Αυτή η επιλογή γίνεται επειδή χρησιμοποιείται αποκλειστικά ο ADRESL όπου αποθηκεύεται το αποτέλεσμα της μετατροπής. Τέλος ορίζεται πόσες αναλογικές εισόδους θα έχει ο μετατροπέας και ταυτόχρονα γίνεται η επιλογή της τάσης αναφοράς. Άρα τίθεται ως αναλογική είσοδος η AN0 και τάση αναφοράς Vref+=VDD και Vref-=VSS. Με αυτές τις ρυθμίσεις ολοκληρώνεται η αρχικοποίηση του A/D μετατροπέα.

Η συνάρτηση με ονομασία read_from_adc(); ενεργοποιεί την διαδικασία μετατροπής.

```
/**
 * Reads from ADC
 */
unsigned char read_from_adc() {
    GO = 1;          // When GO=1 starts A/D conversion
    while(GO);       // Wait until conversion is complete (bit cleared
by hardware when conversion complete)

    return ADRESL;   // Return converted data
}
```

Στον PIC16F877 η A/D μετατροπή επιτυγχάνεται όπως αναλύθηκε με την ενεργοποίηση του GO/ $\overline{\text{DONE}}$ bit. Άρα στην πρώτη γραμμή κώδικα γίνεται ενεργοποίηση της A/D μετατροπής και μετά χρησιμοποιείται μια while μέχρι να ολοκληρωθεί η διαδικασία αυτή. Με την ολοκλήρωση της μετατροπής γίνεται επιστροφή του αποτελέσματός της, που βρίσκεται στον καταχωρητή ADRESL, στην καλούσα συνάρτηση. Έτσι επιτυγχάνεται η συλλογή της κάθε μετατροπής.

3.2 – LCD (Liquid Crystal Display – Οθόνη υγρών κρυστάλλων)

Για να υπάρχει μία οπτική επικοινωνία με τον μικροελεγκτή, προστέθηκε στο κύκλωμα μια οθόνη LCD. Η λειτουργία της είναι αποκλειστικά για την επικοινωνία του μικροελεγκτή προς τον χρήστη. Σε αυτό το υποκεφάλαιο θα αναλυθούν οι εντολές και οι συναρτήσεις που χρησιμοποιήθηκαν για τον έλεγχο της LCD οθόνης. Στον πίνακα 6 θα δοθούν οι βασικότερες εντολές που χρησιμοποιήθηκαν σε αυτήν την εργασία για τον έλεγχο της οθόνης LCD.

Commands No	Commands	Hex Value	Decimal Value
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry Mode	0x06	6
6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15

10	Shift entire display left	0x18	24
11	Shift entire display right	0x1C	30
12	Move cursor left by one character	0x10	16
13	Move cursor right by one character	0x14	20
14	Clear Display (also clear DDRAM content)	0x01	1

Πίνακας 6 – Πίνακας εντολών LCD

Οι εντολές που χρησιμοποιούνται, αποστέλλονται στον ελεγκτή της οθόνης σε 16αδική μορφή. Αρχικά, όπως αναλύθηκε στο δεύτερο κεφάλαιο, γίνεται η αρχικοποίηση της οθόνης στέλνοντας την εντολή με αριθμό 4. Με την εντολή αυτή ρυθμίζεται η διαμόρφωση των 4-bit 2 γραμμών. Εν συνεχεία με την εντολή 8 ορίζουμε πως δεν επιθυμείται η ύπαρξη κέρσορα στην οθόνη και πραγματοποιείται καθαρισμός της οθόνης με την εντολή 14 (από τυχόν προηγούμενα δεδομένα που υπάρχουν). Τέλος στην αρχικοποίηση στέλνεται η εντολή με νούμερο 6. Η εντολή αυτή είναι σημαντική επειδή με αυτήν καθορίζεται με ποια κατεύθυνση θα εμφανίζονται οι χαρακτήρες στην οθόνη, καθώς και αν θα υπάρχει ολίσθηση των χαρακτήρων στην οθόνη. Επιλέχθηκε να μην υπάρχει ολίσθηση των γραμμάτων και να αυξάνεται ο κέρσορας κατά μία θέση κάθε φορά που γίνεται προσθήκη ενός χαρακτήρα. Αυτές είναι οι κύριες εντολές που χρησιμοποιήθηκαν για την αρχικοποίηση της οθόνης. Για την χρήση της LCD πραγματοποιήθηκαν πέντε συναρτήσεις. Η setup_lcd(); είναι αυτή με την οποία γίνεται αρχικοποίηση των καταχωρητών που χρησιμοποιούνται καθώς και τη ρύθμιση της LCD με την αποστολή των απαραίτητων εντολών (commands) για την ορθή λειτουργία της.

```
/**
 * Sets up the LCD
 */
void setup_lcd(){

   TRISD=0;      // PORTD as output
   TRISB2=0;     // PORTB2 pin as output
   TRISB1=0;     // PORTB1 pin as output
   PORTD=0;      // PORTD zero value
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);    // Command instruction set
4bits 2 lines mode
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);
   __delay_ms(20);
   lcd_command(LCD_COMMAND_CURSOR_OFF);      // Command instruction set
cursor off
```

```
   lcd_command(LCD_COMMAND_CLEAR);          // Command instruction
clear screen
   lcd_command(LCD_COMMAND_ENTRY_MODE);     // Command instruction set
entry mode Display Shift off ->
}
```

Στις πρώτες γραμμές υπάρχει αρχικοποίηση των καταχωρητών του μικροελεγκτή που ρυθμίζουν την συμπεριφορά των ακροδεκτών που θα συνδεθούν με την LCD. Οι ακροδέκτες της PORTD του μικροελεγκτή ορίζονται ως ακροδέκτες εξόδου και είναι αυτοί που συνδέονται με τους ακροδέκτες δεδομένων της LCD. Ως ακροδέκτες εξόδου δηλώνονται επίσης οι Pin 1 και Pin 2 της PORTB οι οποίοι θα είναι το Enable (EN) και το Register Select (RS) αντίστοιχα. Έπειτα αρχικοποιείται η PORTD για την αποφυγή προηγούμενων μη επιθυμητών τιμών. Στις επόμενες γραμμές κώδικα γίνεται η αρχικοποίηση της LCD. Στους καταχωρητές της LCD αποστέλλονται : η εντολή για την διαμόρφωση 4-bit 2 γραμμών, η εντολή για την απενεργοποίηση του κέρσορα, η εντολή για τον καθαρισμό της οθόνης και την κατεύθυνση των χαρακτήρων. Η αρχικοποίηση γίνεται με κλήση της συνάρτησης lcd_command();. Το μόνο που πρέπει να τονισθεί, στο σημείο αυτό, είναι πως για να γίνει επιτυχώς η αρχική διαμόρφωση (οποιαδήποτε τύπου και αν είναι αυτή είτε 4-bit 2 γραμμών είτε 8-bit 2 γραμμών) πρέπει η εντολή αρχικοποίησης να σταλεί 3 φορές γιατί διαφορετικά η αρχικοποίηση δεν επιτυγχάνεται.

Η επόμενη συνάρτηση είναι η lcd_strobe();.

```
/**
* Execution of the instructions
*/
void lcd_strobe(){

    EN = 1;          // To execute instructions set EN=1...
    __delay_us(1);   // ... wait for execution...
    EN = 0;          // ... and set EN=0
}
```

Με την συνάρτηση αυτή επιτυγχάνεται η αποστολή όλων των εντολών και δεδομένων στην οθόνη κάνοντας χρήση του bit EN όπως αναλύθηκε στο δεύτερο κεφάλαιο.

Η τρίτη συνάρτηση η οποία θα αναλυθεί είναι η lcd_command (unsigned char); . Με την συνάρτηση αυτή γίνεται η αποστολή των εντολών στην οθόνη.

```
/**
 * Sends the command to the Lcd
 *
 * @param {unsigned char} command    The command passed to the lcd
 */
void lcd_command(unsigned char command){

    RS = 0;                         // When RS = 0 sends command
    __delay_us(1);
    PORTD = (command >> 4);         // First 4 bits to PORTD
    lcd_strobe();
    PORTD = command;                // Last 4 bits to PORTD
    lcd_strobe();
}
```

Αρχικά παρατηρούμε ότι η συνάρτηση αυτή έχει παράμετρο. Η παράμετρος της συνάρτησης είναι η εντολή ή ο 16αδικής μορφής αριθμός που αντιστοιχεί στην εντολή, με στόχο την αποστολή της στην LCD. Στην πρώτη γραμμή ορίζεται το RS=0 για να δηλωθεί ότι στέλνονται εντολές (commands). Έπειτα εξάγονται στην PORTD τα 4 MSB ψηφία και με την lcd_strobe(); στέλνονται στην LCD. Το ίδιο υλοποιείται και με τα 4 LSB ψηφία και πάλι με την lcd_strobe(); αποστέλλονται. Με την διαδικασία αυτή ολοκληρώνεται η αποστολή των εντολών στην LCD. Η αποστολή των δεδομένων στην οθόνη γίνεται με την συνάρτηση lcd_data(unsigned char);.

```
/**
 * Sends the data to the Lcd
 *
 * @param {unsigned char} data    The data passed to the lcd
 */
void lcd_data(unsigned char data){

    RS = 1;                 // When RS = 1 sends data
    __delay_ms(80);
    PORTD = (data >> 4);    // First 4 bits to PORTD
    lcd_strobe();
    PORTD = data;           // Last 4 bits to PORTD
    lcd_strobe();
}
```

Η συνάρτηση αυτή λειτουργεί ακριβώς με την ίδια λογική της lcd_command(unsigned char);. Οι μόνες διαφορές είναι ότι στην παράμετρο στέλνουμε τα δεδομένα προς εκτύπωση και ότι το RS τίθεται ίσο με το λογικό ένα

(RS=1) για να αναγνωρισθούν από την LCD ως δεδομένα προς εκτύπωση. Το υπόλοιπο κομμάτι της συνάρτησης είναι ακριβώς το ίδιο. Η τελευταία συνάρτηση που χρησιμοποιείται για την διαχείριση της LCD είναι η print_to_lcd(const char*);.

```
/**
 * Prints a string to the LCD
 *
 * @param {const char *} string    String to be printed
 */
void print_to_lcd(const char *string) {

    lcd_command(LCD_COMMAND_CLEAR);  // Clear LCD

    while (*string) {                // Sends the string to the LCD
character by character
        lcd_data(*string++);         // It makes the string point to
the next memory address
    }
}
```

Στην συνάρτηση αυτή ως παράμετρο έχουμε την διεύθυνση του πρώτου χαρακτήρα στην στοιχειοσειρά (string) χαρακτήρων που θα εκτυπωθεί στην οθόνη. Έτσι η συνάρτηση ξέρει σε ποιο σημείο θα βρει τον πρώτο χαρακτήρα που θα εκτυπώσει στην οθόνη. Στην πρώτη γραμμή πραγματοποιείται ο καθαρισμός της οθόνης. Εν συνεχεία ξεκινάει μια while με την οποία υλοποιείται η εκτύπωση της στοιχειοσειράς που στέλνεται. Στην συνθήκη της while αναφέρεται ότι όσο υπάρχει τιμή που αντιστοιχεί στον πίνακα ASCII το statement της while είναι true. Δηλαδή όσο υπάρχει χαρακτήρας που αντιστοιχεί στον ASCII πίνακα το αποτέλεσμα της while είναι true και πάντοτε εκτελείται. Με την εκτέλεση της while γίνεται η εκτύπωση της τιμής στην οποία δείχνει ο δείκτης στην τρέχουσα επανάληψη και με την ολοκλήρωση της εκτύπωσης, χρησιμοποιώντας τον τελεστή ++, η μεταβλητή string δείχνει στην επόμενη διεύθυνση μνήμης έτσι ώστε, όταν θα ξαναμπεί στην while, θα εκτυπωθεί ο επόμενος χαρακτήρας. Όταν έρθει ο χαρακτήρας '\0' που αντιστοιχεί στο τέλος της στοιχειοσειράς τότε το statement είναι false και σταματάει η εκτέλεση της while. Αυτό προκύπτει γιατί στην γλώσσα προγραμματισμού C ο χαρακτήρας τέλους '\0' είναι false. Έτσι επιτυγχάνουμε την εκτύπωση της στοιχειοσειράς στην LCD. Με τις συναρτήσεις αυτές και τις κατάλληλες συνδεσμολογίες γίνεται εφικτή η χρήση της LCD οθόνης στην εργασία αυτή.

3.3 – SPI (Serial Peripheral Interface)

Η επικοινωνία με την κάρτα SD για την υλοποίηση της εγγραφής και ανάγνωσης των δεδομένων επιτυγχάνεται κάνοντας χρήση του πρωτοκόλλου SPI. Όπως περιγράφηκε στο 2^{ο} κεφάλαιο η επικοινωνία υλοποιείται με χρήση τεσσάρων διαύλων. Οι 2 είναι οι δίαυλοι δεδομένων ένας από τον μικροελεγκτή προς την κάρτα και ένας από την κάρτα προς τον μικροελεγκτή (MOSI και MISO αντίστοιχα). Ο ένας από τους άλλους δύο είναι ο δίαυλος χρονισμού (SCLK) και ο τελευταίος αφορά ποιά slave (SS) συσκευή είναι επιλεγμένη κάθε φορά. Για τον επιτυχή έλεγχο του SPI υλοποιήθηκαν τρείς συναρτήσεις. Η πρώτη που θα αναλυθεί είναι η setup_spi();.

```
/**
 * Sets up the SPI
 */
void setup_spi(){

    TRISC4 = 1;                         // PORTC4 4bit input
    CS     = 1;                         // Chip Select is high
    RC3    = 0;                         // Clock is low
    RC5    = 0;                         // Data out low
    TRISC2 = TRISC3 = TRISC5 = 0;       // PORTC 2,3,5 bit output
    SSPCON = 0b00100010;                // SPI Master Mode Fosc/64, Clock
Polarity idle state=low level
    SSPEN  = 1;                         // Configuration of SCK,SDO,SDI,
!SS as serial port pins
    SMP    = 1;                         // SSPSTAT 7bit input data sampled
at end of data output time
    CKE    = 1;                         // SSPSTAT 6bit transmit from
transition high -> low
    CKP    = 0;                         // Idle state for clock is a low
level

}
```

Η συνάρτηση αυτή όπως όλες τις setup συναρτήσεις αρχικοποιεί και ρυθμίζει τους καταχωρητές για την σωστή λειτουργία του SPI. Αρχικά τίθεται ότι το PORTC4 είναι bit εισόδου δηλαδή θα δέχεται δεδομένα και αντιστοιχεί με την MISO γραμμή του SPI. Έπειτα ορίζεται ότι τα PORTC2,3,5 θα είναι bit εξόδου και αντιστοιχούν με την SS γραμμή, την SCLK και την MOSI γραμμή. Αρχικοποιούνται οι καταχωρητές εξόδου με low state για την SCLK και MOSI γραμμή ενώ το SS=1 για να γίνει επιλογή της μοναδικής slave συσκευής που διαθέτουμε. Με την αρχικοποίηση του SSPCON=ob00100010 επιλέγεται η λειτουργία Master δηλαδή ότι ο χρονισμός ελέγχεται από τον μικροελεγκτή και η μεταφορά μπορεί να γίνει

οποιαδήποτε στιγμή απαιτηθεί από αυτόν. Επιπλέον ορίζουμε Fosc/64 που είναι η συχνότητα SPI που χρησιμοποιούμε για την αρχικοποίηση της κάρτας SD. Για τον προγραμματισμό του SPI πρέπει να γίνει επιλογή της SPI λειτουργία μεταφοράς (SPI transfer mode). Η επιλογή της σχετίζεται με το χρονικό διάστημα της μετάδοσης ή της συλλογής των δεδομένων, είτε στον άνοδο προς κάθοδο παλμό είτε αντίστροφα. Συγκεκριμένα υπάρχουν 4 transfer mode των οποίων οι λειτουργίες αναλύονται στον πίνακα 7.

SPI Mode	CPOL (Clock Polarity)	CPHA (Clock Phase)	CKP(CPOL in PIC)	CKE(NotCPHA in PIC)	Explanation
0	0	0	0	1	Base Value of the clock is zero, Data captured from low to high transition and transmission from high to low transition
1	0	1	0	0	Base Value of the clock is zero, Data captured from high to low transition and transmission from low to high transition
2	1	0	1	1	Base Value of the clock is one, Data captured from low to high transition and transmission from high to low transition
3	1	1	1	0	Base Value of the clock is one, Data captured from low to high transition and transmission from high to low transition

Πίνακας 7 – SPI Modes

Με την επιλογή του CKE=1 και CKP=0 ενεργοποιήθηκε το SPI Mode 0 όπου η τιμή του παλμού ρολογιού είναι μηδέν ενώ η συλλογή των δεδομένων γίνεται από την κάθοδο προς την άνοδο του παλμού και η μετάδοση των δεδομένων από την άνοδο προς την κάθοδο του παλμού. Τέλος με την ρύθμιση SSPEN=1 γίνεται ενεργοποίηση των SCK, SDO, SDI, και $\overline{SS}$ ως ακροδέκτες για την σειριακή μεταφορά, ενώ με την SMP=1 ορίζεται ότι η δειγματοληψία των δεδομένων εισόδου θα γίνεται στο τέλος των δεδομένων εξόδου. Με την ρύθμιση των

καταχωρητών αυτών έχουμε ετοιμάσει τον μικροελεγκτή μας για την επικοινωνία μέσω SPI.

Η επικοινωνία γίνεται κάνοντας εγγραφή και ανάγνωση του διαύλου SPI. Έχει γίνει υλοποίηση 2 συναρτήσεων μία για την κάθε λειτουργία. Η write_to_spi(unsigned char); κάνει την εγγραφή στον δίαυλο.

```
void write_to_spi(unsigned char data) {

    SSPBUF = data;    // Send the data to SSPBUF
    while (BF == 0); // Wait until receive of data is complete
}
```

Ως παράμετρο στέλνονται τα επιθυμητά δεδομένα για την εγγραφή τους στον δίαυλο. Στον καταχωρητή SSPBUF αποθηκεύονται τα δεδομένα και με την χρήση μιας while και του BF bit αναμένεται μέχρι να γεμίσει ο SSPBUF με τα δεδομένα που αποστάλθηκαν.

Η read_from_spi(); κάνει την διαδικασία ανάγνωσης από τον δίαυλο SPI.

```
/**
 * Reads data from SPI bus
 */
unsigned char read_from_spi() {
    SSPBUF = SPI_DUMMY_CLOCK; // Send dummy data to slave
    while (BF == 0);          // Wait that data is received
    return SSPBUF;            // Read the data that slave has sent
back
}
```

Στην επικοινωνία μέσω SPI πάντα πρέπει να γίνεται αμφίδρομη επικοινωνία μεταξύ master και slave. Αυτό σημαίνει ότι για να υλοποιηθεί η διαδικασία ανάγνωσης δεδομένων πρέπει να γίνει υποχρεωτικά μια εγγραφή. Για τον λόγο αυτό πραγματοποιείται, στην πρώτη γραμμή της συνάρτησης, μια εγγραφή dummy δεδομένων στον MOSI δίαυλο με στόχο την ανάγνωσή τους από τον MISO δίαυλο (της κάρτας SD). Έπειτα χρησιμοποιώντας εκ νέου το bit BF αναμένεται η ολοκλήρωση της λήψης των δεδομένων που διαβάστηκαν από τον δίαυλο SPI, τα οποία επιστρέφονται στην καλούσα συνάρτηση επιτρέποντας την επικοινωνία μεταξύ μικροελεγκτή και κάρτας SD μέσω πρωτοκόλλου SPI.

3.4 – SD Card (Secure Digital Card)

Η SD Card ή εν συντομία SDC είναι ένας τύπος αποθηκευτικού χώρου ο οποίος αναπτύχθηκε για χρήση στην κινητή τηλεφωνία. Δεν είναι τυχαίο γεγονός πως κάθε κινητή συσκευή (εκτός λίγων εξαιρέσεων) φέρνει θήκη για την τοποθέτηση Micro SDC. Λόγω του μικρού της μεγέθους σε σχέση με τον αποθηκευτικό χώρο που προσφέρει έχει γίνει πολύ δημοφιλής σε πάρα πολλές συσκευές και για τον λόγο αυτό έχει πολύ μεγάλη υποστήριξη σαν αποθηκευτικό μέσο και στην χρήση της στα ενσωματωμένα συστήματα. Πρόγονος της SDC ήταν η MMC και τώρα κυκλοφορούν κυρίως οι Micro SDC γιατί προσφέρουν τις ίδιες δυνατότητες σε πολύ μικρότερο χώρο. Προσοχή απαιτεί η χρήση τους επειδή το εύρος τάσης στο οποίο λειτουργούν είναι από τα 2.7V έως τα 3.6V, πράγμα που αποτελεί κίνδυνο γιατί αν τροφοδοτηθούν με 5V θα καταστραφούν άμεσα.

Για να γίνει εφικτός ο έλεγχος της SDC χρησιμοποιείται το πρωτόκολλο SPI του οποίου η λειτουργία αναλύθηκε στο δεύτερο κεφάλαιο. Σε αυτό το σημείο θα αναλυθεί η επικοινωνία με την SDC και θα γίνει αναφορά στις εντολές που χρησιμοποιήθηκαν για την λειτουργία αυτή. Για την επικοινωνία χρησιμοποιούνται εντολές, σε κάθε απεσταλμένη εντολή υπάρχει και η αντίστοιχη απόκριση. Οι εντολές είναι συντομογραφημένες ανάλογα με την λειτουργία τους για να είναι εύκολα αναγνωρίσιμες. Η κάθε εντολή έχει και έναν κωδικό ο οποίος είναι CMD<n> (όπου CMD είναι συντομογραφία του Command, και όπου <n> είναι ένας αριθμός από το 0 έως το 63). Οι απαντήσεις που παίρνουμε από την κάρτα SD είναι μορφής R1, R1b, R3, R7. Στον πίνακα 8 θα αναρτηθεί ο πίνακας με τις βασικότερες εντολές γενικού τύπου που αφορούν την επικοινωνία με την SDC όπως αρχικοποίηση, εγγραφή, ανάγνωση που χρησιμοποιούνται στην παρούσα εργασία.

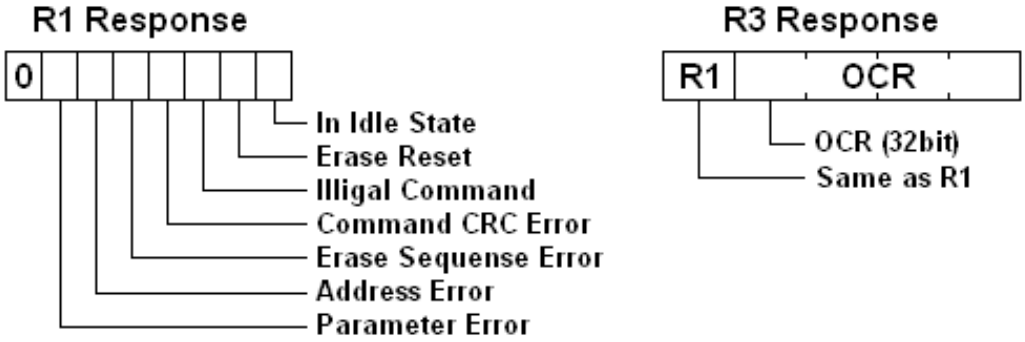
Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.

CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.
***1: ACMD<n> means a command sequence of CMD55-CMD<n>.**					
***2: Rsv(0)[31], HCS[30], Rsv(0)[29:0]**					
***3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]**					

Πίνακας 8 - Πίνακας εντολών κάρτας SD

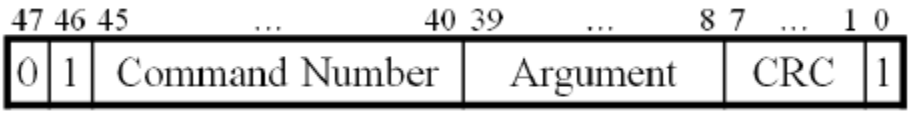
Οι περισσότερες εντολές όπως αναγράφεται στον πίνακα δέχονται ως απαντήσεις το R1. Η ανάλυση των bit του R1 απεικονίζεται στην εικόνα 22. Εάν το R1 = 0x00 αυτό σημαίνει πως η εντολή που στάλθηκε εκτελέστηκε επιτυχώς. Οποιοδήποτε λάθος παρουσιαστεί στην εντολή θα μας επιστραφεί στην απόκριση με αποτέλεσμα διάφορο του 0x00 και ένα ή περισσότερα bit του R1 θα έχουν αλλάξει τιμή από λογικό μηδέν σε λογικό ένα. Υπάρχουν άλλοι 3 τύποι αποκρίσεων. Οι R3 και R7 αφορούν αποκρίσεις δύο αποκλειστικών εντολών, την CMD58 και την CMD8, και έχουν ως απόκριση την R1+OCR (32bit). Ο OCR (Operating Conditions Register) είναι καταχωρητής που δίνει την κατάσταση

λειτουργίας της κάρτας. Η R1b απόκριση αφορά κάποιες εντολές οι οποίες απαιτούν περισσότερο χρόνο για να εμφανιστούν. Για τον λόγο αυτό υπάρχει R1b που είναι μια R1 απόκριση με την ακολουθία ενός busy flag για τον οποίο αναμένουμε μέχρι να ολοκληρωθεί η διαδικασία.



Εικόνα 22 – Ανάλυση αποκρίσεων εντολών κάρτας SD

Στο σημείο αυτό θα γίνει αναφορά στις εντολές που χρησιμοποιούνται στην εργασία αυτή. Οι εντολές που στέλνουμε στην SDC έχουν ένα συγκεκριμένο πλαίσιο το οποίο πρέπει να τηρηθεί για να αποσταλούν με επιτυχία. Το πλαίσιο αυτό αποτελείται από τρία μέρη. Τον κωδικό εντολής, το ορισμό της εντολής και τέλος το CRC (Cyclic Redundancy Code) που είναι μία μέθοδος ανίχνευσης και διόρθωσης λάθους. Στην εικόνα 23 εμφανίζεται και σχηματικά το πλαίσιο μέσα του οποίου γίνεται η αποστολή εντολών.



Εικόνα 23 – Πλαίσιο εντολών κάρτας SD

Για να γίνει αποστολή των εντολών στην κάρτα SD έγινε συγγραφή της συνάρτησης sd_command(unsigned char, unsigned long int, unsigned char);. Με την συνάρτηση αυτή ουσιαστικά υλοποιούμε το πλαίσιο εντολών της εικόνας 23 για να γίνει αναγνώρισή του από την SDC.

```
/**
 * Sends a command to the SD Card
 * @param {unsigned char}       command      Command Code
 * @param {unsigned long int}   argument     Argument of the Command
 * @param {unsigned char}       crc          CRC Value
 */
```

```
void sd_command(unsigned char command, unsigned long int argument,
unsigned char crc) {

    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(0b01000000 | command);             // OR the command
because Bits 7-8 are always 01
    write_to_spi((unsigned char) (argument >> 24)); // send the four-
byte argument in byte chunks
    write_to_spi((unsigned char) (argument >> 16));
    write_to_spi((unsigned char) (argument >>  8));
    write_to_spi((unsigned char) (argument));
    write_to_spi(crc);
}
```

Στην συνάρτηση αυτή παρατηρείται η παρουσία τριών παραμέτρων. Προφανώς οι παράμετροι αυτοί δεν είναι άλλοι παρά τα τρία μέρη του πλαισίου εντολών δηλαδή ο κωδικός εντολής, ο ορισμός της εντολής και το CRC. Στην πρώτη γραμμή στέλνεται ένα dummy clock για να ξεκινήσει η διαδικασία. Αρχικά αποστέλλεται ο κωδικός της εντολής. Ο κωδικός εντολής έχει μέγεθος 6 bit. Για τον λόγο αυτό στο πλαίσιο εντολών τα πρώτα 2 bit είναι πάντα "01" για να ακολουθήσει ο 6-bit κωδικός. Στέλνεται στη SDC, κάνοντας την χρήση μιας OR, πρώτα το "01" και μετά το command που έρχεται μέσω της παραμέτρου. Στην συνέχεια στέλνεται ο 32-bit ορισμός της εντολής (δηλαδή 4 byte) χωρισμένο σε 4 κομμάτια (byte chunks) μιας και με το SPI είναι εφικτή η εγγραφή μόνο 1 byte την φορά. Άρα αρχικά στέλνονται τα MSB bits φτάνοντας τελικώς στα LSB bits. Στο τέλος κάνουμε την εγγραφή του CRC για να ολοκληρωθεί το πλαίσιο της εντολής. Μια σημείωση πως στο SPI mode της κάρτας SD το CRC δεν ελέγχεται και για τον λόγο αυτό τις περισσότερες φορές αποστέλλεται μια οποιαδήποτε τιμή εκτός εάν χρειαστεί να σταλεί κάτι συγκεκριμένο.

Η κάρτα SD όταν ενεργοποιείται κάνει boot στο SD mode. Στα πλαίσια των αναγκών του project αυτού πρέπει να αρχικοποιηθεί κατάλληλα για να λειτουργήσει στο SPI mode. Για την αρχικοποίηση πρέπει να ακολουθηθούν συγκεκριμένα βήματα. Αρχικά θα πρέπει να γίνουν 74 ή περισσότεροι παλμοί ρολογιού στην κάρτα για να μπει στην βασική λειτουργία εκκίνησης. Η πρώτη ενέργεια είναι να σταλεί η CMD0 εντολή με την οποία γίνεται επαναφορά (reset) της SDC και είσοδός της στην κατάσταση αδράνειας (idle mode) ενώ ταυτόχρονα τίθεται το CS ίσο με το λογικό μηδέν (CS=0) για να λειτουργήσει η κάρτα στην λειτουργία SPI. Στην κατάσταση αδράνειας η SDC μπορεί να δεχθεί μόνο

συγκεκριμένες εντολές. Για να γίνει η αρχικοποίηση της κάρτας πρέπει να σταλεί η εντολή αρχικοποίησης που είναι το CMD1. Με την αποδοχή της εντολής αυτής η κάρτα είναι έτοιμη να δεχθεί οποιαδήποτε άλλη εντολή είτε για εγγραφή είτε για ανάγνωση. Η τρίτη και τελευταία εντολή που χρησιμοποιείται, όντας και αυτή μέρος της διαδικασίας αρχικοποίησης, είναι η CMD16 με την οποία ορίζεται το block size (μπλοκ μέγεθος) των δεδομένων στην τιμή των 512 byte το οποίο είναι ένα από τα πιο ευρέως χρησιμοποιημένα block size στις SDC. Στο σημείο αυτό πρέπει να τονισθεί, όπως αναφέρθηκε και στο δεύτερο κεφάλαιο, πως στην εργασία αυτή δεν χρησιμοποιείται κάποιο file system (σύστημα αρχείων) όπως FAT16, FAT32, NTFS κτλ. Επίσης τα δεδομένα που αποθηκεύονται στην κάρτα είναι μορφής raw (ακατέργαστα) και δεν έχουν υποστεί κανένα είδος συμπίεσης, όπως mp3 ή wma, με αποτέλεσμα η αναπαραγωγή να υλοποιείται μόνο στην ίδια την συσκευή και όχι σε κάποια άλλη.

Η συνάρτηση setup_sd(); είναι η υλοποίηση της αρχικοποίησης που μόλις αναλύθηκε.

```
/**
 * Sets up the SD Card
 */
void setup_sd() {

    unsigned char received_data;
    unsigned int counter = 0;
    CS = 1;

    for (unsigned int i = 0; i < 74; i++) {              // Apply 74
clock pulses to SD Card
        write_to_spi(SPI_DUMMY_CLOCK);
    }

    CS = 0;
    __delay_ms(1);
    sd_command(SD_COMMAND_GO_IDLE_STATE, SD_START_ADDRESS, 0x95); //
Software Reset
    received_data = read_from_spi();

    while ((received_data != 1) && (counter < 1000)) {   // Reads from
SPI until received data = 1
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000) {                               // If
received data != 1 command fails
        print_to_lcd("Cmd 0 failed");
        while(1);
    }
    sd_command(SD_COMMAND_INIT, SD_START_ADDRESS, 0xff); //
Initialization process
```

```
    received_data = read_from_spi();

    counter = 0;
    while((received_data != 0) && (counter < 1000)) {    // Reads from
SPI until received data = 0
        received_data = read_from_spi();
        sd_command(SD_COMMAND_INIT, SD_START_ADDRESS, 0xff);
        received_data = read_from_spi();
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000){
        print_to_lcd("Cmd 1 failed");                   // If received
data != 0 command fails
        while(1);
    }
    sd_command(SD_COMMAND_SETBLOCKLEN, BLOCKLEN, 0xff); // Set block
size
    received_data = read_from_spi();
    counter = 0;
    while ((received_data != 0) && (counter < 1000)) {  // Reads from
SPI until received data = 0
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000) {
        print_to_lcd("Cmd 16 failed");                  // If received
data != 0 command fails
        while(1);
    }
    print_to_lcd("SD Card is A-OK");
    __delay_ms(1000);
    SSPCON = 0b00100001;                                // Set Fosc/16
for reading process
}
```

Στην αρχή γίνεται δήλωση 2 μεταβλητών που θα χρησιμοποιηθούν από την συνάρτηση στην συνέχεια, η πρώτη για την αποθήκευση δεδομένων (received_data) και η δεύτερη για την χρήση στις δομές επανάληψης (counter). Έπειτα τίθεται το CS=1 και αρχίζει η αποστολή των 74 παλμών, επιτρέποντας την κάρτα να μπει στην βασική λειτουργία εκκίνησης. Εν συνεχεία τίθεται το CS=0 και στέλνεται το CMD0 (Go Idle State). Η κάρτα αρχικοποιείται επιτυχώς στην λειτουργία SPI και εφόσον ενεργοποιηθεί στέλνει ως απόκριση το R1 = 0x01 το οποίο είναι το bit απόκρισης στην βασική λειτουργία (idle state). Αν δεν γίνει σωστά η αρχικοποίηση και δεν ληφθεί στο χρονικό διάστημα που ορίστηκε, με την χρήση του counter, τότε στην οθόνη εμφανίζεται το μήνυμα λάθους που μας ενημερώνει ότι η εντολή CMD0 απέτυχε και σταματάει η εκτέλεση του προγράμματος. Η κάρτα στο σημείο αυτό βρίσκεται στο idle state και για τον λόγο αυτό δεν μπορεί να δεχθεί όλες τις εντολές. Για να μπορέσει να δεχτεί τις

εντολές εγγραφής και ανάγνωσης πρέπει να σταλεί η CMD1 (SD Command Init) με την οποία ξεκινάει η διαδικασία αρχικοποίησης. Με την ίδια λογική, όπως και με την CMD0, στέλνεται η CMD1 στην SDC και γίνεται έλεγχος αν το R1=0 στο χρονικό πλαίσιο που έχει οριστεί. Εφόσον το R1=0 τότε σημαίνει ότι η διαδικασία αρχικοποίησης ολοκληρώθηκε και η κάρτα είναι έτοιμη να δεχθεί οποιαδήποτε εντολή. Σε αντίθετη περίπτωση εμφανίζεται και πάλι μήνυμα με την ένδειξη ότι η εντολή CMD1 απέτυχε και σταματάει η εκτέλεση του προγράμματος. Η τελευταία εντολή που αποστέλλεται για την αρχικοποίηση της SD είναι η CMD16 (Set Block Length) με την οποία ορίζεται το μπλοκ μέγεθος των δεδομένων. Στην παρούσα εργασία κάνουμε χρήση block size 512 byte. Με παρόμοιο τρόπο στέλνεται στην SDC η CMD16, ελέγχεται εάν η απόκριση είναι R1=0 μέσα στο χρονικό πλαίσιο που έχει οριστεί και εφόσον η διαδικασία ολοκληρωθεί επιτυχώς τότε εμφανίζεται στην οθόνη μήνυμα επιβεβαίωσης και ρυθμίζεται Fosc/16 που είναι η συχνότητα SPI για την διαδικασία ανάγνωσης που ακολουθεί. Σε αντίθετη περίπτωση εμφανίζεται το μήνυμα αποτυχίας για την CMD16 και σταματάει η εκτέλεση του προγράμματος. Η σημασία αυτής της συνάρτησης είναι μεγάλη δεδομένου ότι εφόσον εκτελεστεί, σημαίνει ότι η κάρτα είναι έτοιμη για εγγραφή, ανάγνωση και ταυτόχρονα ότι δεν έχει κάποιο λειτουργικό πρόβλημα.

Η συνάρτηση handle_sd_command_response(unsigned char); που θα αναλυθεί είναι αυτή που χρησιμοποιείται για την διαχείριση των απαντήσεων που δεχόμαστε από την SDC. Η χρήση της συνάρτησης αυτής αποτελεί επιβεβαίωση ότι η εκτέλεση των εντολών ολοκληρώθηκε επιτυχώς, ελέγχοντας την απόκριση που δεχόμαστε από την SDC.

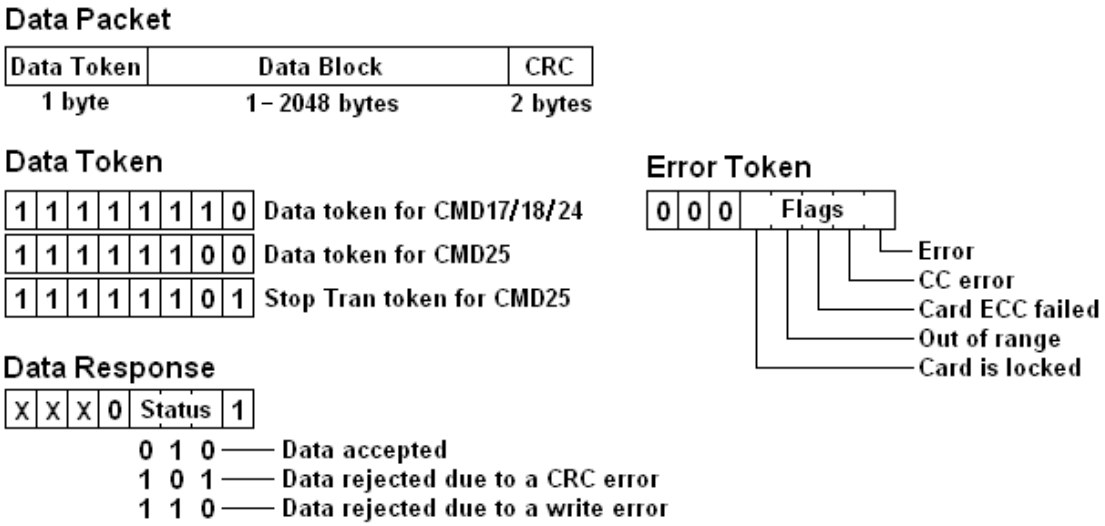
```
/**
 * SD command response handler
 *
 * @param {unsigned char}  command    Command Code
 */
void handle_sd_command_response(unsigned char command) {

    unsigned char received_data;
    switch(command) {
        case SD_COMMAND_READ_MULTIPLE_BLOCKS:       // For read case
        case SD_COMMAND_WRITE_MULTIPLE_BLOCKS:      // For write case
            received_data = read_from_spi();
            while (received_data != SD_COMMAND_RESPONSE_SUCCESS) { //
If received data not succesful read again
                received_data = read_from_spi();
            }
            break;
```

```
        case SD_COMMAND_STOP_TRANSMISSION:              // For stop
tran case
            received_data = read_from_spi();            // Stuff byte
has to be ignored
            received_data = read_from_spi();            // Previously
stuff byte so read again
            while (received_data != SD_COMMAND_RESPONSE_SUCCESS) { //
If received data not succesful read again
                received_data = read_from_spi();
            }

            received_data = read_from_spi();
            while (received_data != DATA_TOKEN_BUSY) {  // Wait until
data out bus goes high for end of the process
                received_data = read_from_spi();
            }
            break;
    }
}
```

Η συνάρτηση αυτή δέχεται παράμετρο. Το όρισμα της παραμέτρου είναι ο κωδικός εντολής για την οποία πρέπει να γίνει έλεγχος απόκρισης. Στην αρχή του κώδικα δηλώνεται μία μεταβλητή η οποία θα αποθηκεύει τα δεδομένα. Έπειτα γίνεται υλοποίηση μιας switch case με την οποία ανάλογα με τον κωδικό εντολής που λαμβάνεται, τρέχει το αντίστοιχο κομμάτι κώδικα. Όπως παρατηρείται για την διαδικασία πολλαπλής εγγραφής (Write Multiple Blocks, CMD25) και πολλαπλής ανάγνωσης (Read Multiple Blocks, CMD18) ο έλεγχος είναι ο ίδιος επειδή η απόκριση για αυτές τις δύο εντολές είναι η ίδια (δηλαδή R1). Αρχικά γίνεται ανάγνωση των δεδομένων και μετά ελέγχεται αν το αποτέλεσμα είναι R1=0x00. Εφόσον η απάντηση είναι ορθή τότε σταματάει ο έλεγχος, σταματάει το case της switch και τελειώνει η εκτέλεση της συνάρτησης. Εφόσον η τιμή του R1 είναι διαφορετική τότε γίνεται συνεχώς ανάγνωση μέχρι να ολοκληρωθεί η αποστολή της εντολής και να ληφθεί η σωστή απόκριση από την SDC. Υπάρχει και μία άλλη εντολή που στέλνεται στην SD και είναι αυτή της διακοπής ανάγνωσης (Stop Tran, CMD12). Η εντολή αυτή στέλνεται στην SDC όταν θέλουμε να τερματίσουμε την ανάγνωση. Όπως αναλύθηκε νωρίτερα η CMD12 έχει ως απόκριση όχι το R1 αλλά το R1b. Στο σημείο αυτό θα πρέπει να γίνει έλεγχος ότι το R1=0x00 και στην συνέχεια πρέπει να αναμένεται η ολοκλήρωση της διαδικασία, ελέγχομτας πότε η MISO γραμμή γίνει 0xff. Παρατηρείται ότι γίνονται δύο αναγνώσεις, αυτό προκύπτει από το γεγονός ότι στην CMD12 εντολή το πρώτο byte που λαμβάνεται αποτελείται από λάθος δεδομένα και το

φαινόμενο αυτό ονομάζεται stuff byte. Για τον λόγο αυτό πραγματοποιείται χωρίς έλεγχο μία ακόμη ανάγνωση για να αποκτηθεί η σωστή τιμή. Συνεχίζοντας την εκτέλεση γίνεται ο έλεγχος για το R1 και όπως στο προηγούμενο case αναμένεται μέχρι να εμφανιστεί η τιμή R1=0x00 για να συνεχίσει η εκτέλεση. Με την ανάγνωση της τιμής αυτής περνάμε στον έλεγχο του busy flag. Γίνεται και πάλι ανάγνωση των δεδομένων και έλεγχος για το πότε θα εμφανιστεί στην D0 γραμμή της SD η τιμή 0xff που σηματοδοτεί το τέλος του busy flag. Τέλος γίνεται ολοκλήρωση της εκτέλεσης της switch και επιστρέφει η εκτέλεση στην καλούσα συνάρτηση.

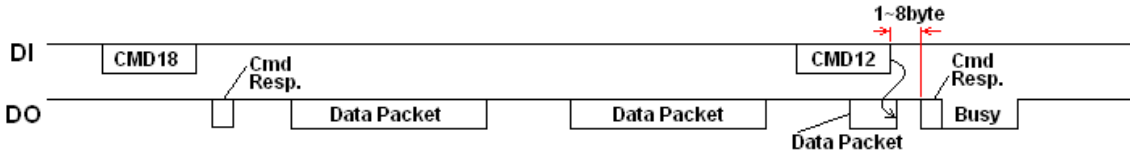
Για την επίτευξη είτε εγγραφής είτε ανάγνωσης στην SDC πρέπει να ξέρουμε πέρα από τις απαντήσεις που δίνουν οι εντολές και τις αποκρίσεις από τα πλαίσια των πακέτων δεδομένων που πρέπει να σταλούν ή να ληφθούν. Στην εικόνα 24 απεικονίζονται σχηματικά. Και το πακέτο δεδομένων αποτελείται από τρία μέρη. Το πρώτο μέρος αποτελείται από την κεφαλίδα (data token) των δεδομένων μεγέθους 1byte. Υπάρχουν 3 διαφορετικά data token για τις διάφορες εντολές που αποστέλλονται στην κάρτα. Το πρώτο συνδέεται με την εντολή CMD18 που αφορά την ανάγνωσης της SDC, το δεύτερο και το τρίτο συνδέονται με την εντολή CMD25 που αφορά την εγγραφή στην κάρτα SD. Έπειτα υπάρχει το μπλοκ δεδομένων και στο τέλος το CRC μεγέθους 2byte. Εδώ θεωρείται αναγκαίο να τονιστεί πως στην εικόνα αναγράφεται ότι το πακέτο δεδομένων είναι 2048byte, ενώ κατά την αρχικοποίηση τέθηκαν ως μέγεθος μπλοκ δεδομένων τα 512byte.



Εικόνα 24 – Ανάλυση κεφαλίδων και απόκριση των πακέτων δεδομένων

Η απόκριση δεδομένων (Data Response) που λαμβάνεται αφορά μόνο την διαδικασία εγγραφής και πρέπει να είναι πάντα τιμής 0x05 (0b00000101) για να βεβαιωθεί ότι τα δεδομένα έγιναν αποδεκτά από την κάρτα. Σε διαφορετική περίπτωση υπάρχουν άλλες τιμές ανάλογα με το λάθος που προέκυψε.

Θα ολοκληρωθεί η θεωρητική προσέγγιση της SDC και των εντολών της περιγράφοντας την λειτουργία ανάγνωσης και την λειτουργία εγγραφής. Στην εικόνα 25 υπάρχει η σχηματική ανάλυση της πολλαπλής ανάγνωσης αρχείων. Η εντολή CMD18 είναι η εντολή που μας επιτρέπει την ανάγνωση πολλαπλών πακέτων δεδομένων από μια συγκεκριμένη διεύθυνση. Η διεύθυνση ορίζεται στα δεδομένα της εντολής. Μόλις η κάρτα δεχθεί την εντολή στο δίαυλο DI (Data In) τότε στον δίαυλο DO (Data Out) εμφανίζεται η απόκριση της εντολής. Διαβάζονται πολλαπλά πακέτα έως ότου στον δίαυλο DI εμφανιστεί η εντολή CMD12 η οποία αιτείται από την κάρτα την παύση αποστολής πακέτων. Εφόσον η εντολή αυτή γίνει αποδεκτή σταματάει η αποστολή πακέτων.



Εικόνα 25 – Ανάλυση πολλαπλής ανάγνωσης πακέτων κάρτας SD

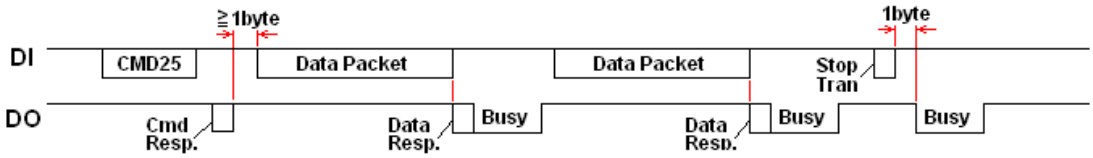
Με αντίστοιχο σκελετό σαν αυτό της εικόνας 25 υλοποιήθηκε η συνάρτηση read_from_sd(); με την οποία πραγματοποιείται η διαδικασία ανάγνωσης από την κάρτα SD.

```
/**
 * Read from SD
 */
void read_from_sd() {
    SSPCON =0b00100001;                                          // Set
Fosc/16 for reading process
    print_to_lcd("Listening...");                                // LCD
prints "Listening"
    unsigned char received_data;
    sd_command(SD_COMMAND_READ_MULTIPLE_BLOCKS, SD_START_ADDRESS,
0xff); // Send command Command 18 for multiple read
    handle_sd_command_response(SD_COMMAND_READ_MULTIPLE_BLOCKS);
// Handling the command response

    while (!RECORD_INTERRUPT) {                                  //
While button is not pressed enter loop
        received_data = read_from_spi();                         //
Read first appeared data from spi
```

```
        while (received_data != DATA_TOKEN_READ) {                //
Read continuously data until read data token appears
            received_data = read_from_spi();
        }

        for (unsigned int i = 0; i < BLOCKLEN; i++) {             // For
every data block
            received_data = read_from_spi();                      //
Read the data from sdc

            __delay_us(16.5);
            CCPR2L = received_data;                               // And
send them to the PWM for reproduction
        }

        write_to_spi(SPI_DUMMY_CLOCK);                            // 1st
CRC byte
        write_to_spi(SPI_DUMMY_CLOCK);                            // 2nd
CRC byte
    }

    sd_command(SD_COMMAND_STOP_TRANSMISSION, 0, 0xff);           //
Button was pressed so send stop transmission command
    handle_sd_command_response(SD_COMMAND_STOP_TRANSMISSION);    //
Handling the command response

    start_recording();                                            //
Start the recording process
}
```

Αρχικά ρυθμίζεται η συχνότητας SPI για την διαδικασία ανάγνωσης, έπειτα εκτυπώνεται στην οθόνη το μήνυμα ότι ξεκινάει η αναπαραγωγή της εγγραφής. Δηλώνεται η μεταβλητή στην οποία θα αποθηκεύονται τα δεδομένα. Η διαδικασία ανάγνωσης ξεκινάει στέλνοντας την εντολή CMD18 στην κάρτα SD και καλώντας την handle_sd_command_response ελέγχεται η απόκριση από την SDC. Εφόσον η εντολή γίνει δεκτή ξεκινάει η ανάγνωση. Υλοποιείται μια while στην οποία ελέγχεται εάν έχει πατηθεί το πλήκτρο που ξεκινάει την διαδικασία εγγραφής. Όσο το πλήκτρο δεν έχει πατηθεί εκτελείται ο κώδικας της while. Μέσα στην while γίνεται ανάγνωση των δεδομένων και έλεγχος μέχρι να εμφανιστεί στον δίαυλο το data token read (0b11111110) της εντολής CMD18. Έπειτα γίνεται αποστολή των δεδομένων στο PWM μέσω του καταχωρητή CCPR2L διαβάζοντας συνεχόμενα block από την κάρτα. Τέλος για να ολοκληρωθεί το πλαίσιο δεδομένων προσθέτουμε και τα 2 byte του CRC. Αυτή η διαδικασία συνεχίζει μέχρι να πατηθεί το πλήκτρο που σηματοδοτεί την αρχή της εγγραφής. Με το πάτημα του πλήκτρου στέλνεται άμεσα στην SD μία CMD12 για να σταματήσει να μας αποστέλλει δεδομένα, γίνεται ο έλεγχος της απόκρισης μέσω της handle_sd_command_response και καλείται η συνάρτηση που αναλαμβάνει την διαδικασία εγγραφής. Με τον τρόπο αυτό επιτυγχάνεται η ανάγνωση της SDC.

Ίδια λογική της CMD18 ακολουθείται και με την εντολή CMD25 η οποία μόλις εμφανιστεί στον DI δίαυλο και γίνει επιτυχώς δεκτή από την απόκριση που εμφανίζεται στο DO δίαυλο, αρχίζει την εγγραφή των δεδομένων. Η εγγραφή των δεδομένων γίνεται στο δίαυλο DI και σε κάθε τέλος πακέτου (μεγέθους 512byte όπως έχει οριστεί) εμφανίζεται στον δίαυλο DO η απόκριση του πακέτου δεδομένων. Ελέγχεται ότι η εγγραφή έχει γίνει σωστά κάνοντας του απαραίτητους ελέγχους και συνεχίζει η εγγραφή των επιθυμητών πακέτων δεδομένων. Για την παύση αποστολής των δεδομένων στέλνεται το STOP TRAN TOKEN με το οποίο ενημερώνεται η SDC ότι δεν θα δεχθεί άλλα δεδομένα. Εφόσον το STOP TRAN TOKEN γίνει δεκτό τότε σταματάει η διαδικασία εγγραφής δεδομένων. Στην εικόνα 26 γίνεται η σχηματική αναπαράσταση της διαδικασίας αυτής.



Εικόνα 26 - Ανάλυση πολλαπλής εγγραφής πακέτων κάρτας SD

Με την συνάρτηση write_to_sd(); γίνεται η εγγραφή των δεδομένων στην SDC όπως φαίνεται και στην εικόνα 26.

```
/**
 * Write to SD
 */
void write_to_sd() {

    SSPCON = 0b00100000;                // Set Fosc/4 for writing
process
    print_to_lcd("Recording...");       // LCD prints "Recording"
    unsigned char received_data;
    sd_command(SD_COMMAND_WRITE_MULTIPLE_BLOCKS, SD_START_ADDRESS,
0xff); // Send command Command 25 for multiple write
    handle_sd_command_response(SD_COMMAND_WRITE_MULTIPLE_BLOCKS);
// Handling the command response
    write_to_spi(SPI_DUMMY_CLOCK);                          // Dummy
clocks before sending data packet
    write_to_spi(SPI_DUMMY_CLOCK);

    while (!RECORD_INTERRUPT) {                             // While
button is not pressed enter loop
        write_to_spi(DATA_TOKEN_WRITE);                     // Send
the data token for the data packet

        for (unsigned int i = 0; i < BLOCKLEN; i++) {       // For
every data block
            unsigned char adc_data = read_from_adc();
            write_to_spi(adc_data);                         // write
the result of the adc to the sdc
```

```
        }

        write_to_spi(SPI_DUMMY_CLOCK);                          // 1st CRC
byte
        write_to_spi(SPI_DUMMY_CLOCK);                          // 2nd CRC
byte
        received_data = read_from_spi();                        // Read
data response after first block sent

        while ((received_data & 0b00011111) != DATA_TOKEN_ACCEPTED ) {
// Check continuously that data was accepted
            received_data = read_from_spi();
        }

        while (received_data != DATA_TOKEN_BUSY){               // Wait
until data out bus goes high for end of the process
            received_data = read_from_spi();
        }
    }

    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(DATA_TOKEN_STOP_TRAN);                         // Button
was pressed so send stop tran token
    write_to_spi(SPI_DUMMY_CLOCK);                              // Dummy
byte after stop tran token

    received_data = read_from_spi();
    while (received_data != DATA_TOKEN_BUSY){                   // Wait
until data out bus goes high for end of the process
        received_data = read_from_spi();
    }
}
```

Αρχικά γίνεται αλλαγή της συχνότητας SPI για την διαδικασία εγγραφής κάνοντάς την ίση με Fosc/4. Αυτή η αλλαγή πραγματοποιείται επειδή κατά την υλοποίηση της συσκευής παρατηρήθηκε ότι εάν παρέμενε η ίδια συχνότητα στην αναπαραγωγή και στην ηχογράφηση θα προέκυπτε λανθασμένο και κακό ακουστικό αποτέλεσμα. Εν συνεχεία γίνεται εκτύπωση μηνύματος στην οθόνη ότι ξεκινάει η εγγραφή και δηλώνεται μεταβλητή για την αποθήκευση των δεδομένων. Αποστέλλεται η εντολή CMD25 και μέσω της handle_sd_command_response γίνεται ο έλεγχος της απόκρισης που λαμβάνεται. Στην συνέχεια στέλνονται 2 dummy bytes όπως απεικονίζεται στην εικόνα 26 πριν γίνει η εγγραφή του πακέτου. Έπειτα υπάρχει, όπως και με την ανάγνωση, η while με την οποία ελέγχεται εάν πατήθηκε το πλήκτρο. Επειδή τώρα η συσκευή βρίσκεται στο στάδιο εγγραφής πρέπει ουσιαστικά να υλοποιηθεί το πλαίσιο που υπάρχει στην εικόνα 24. Δηλαδή θα πρέπει να σταλούν αρχικά το data token έπειτα τα δεδομένα και τέλος τα 2 CRC bytes. Αντίστοιχα στον κώδικα θα

γραφτεί, αρχικά στην SDC, το data token write (0b11111100). Τα δεδομένα που λαμβάνονται από τον μετατροπέα A/D εγγράφονται ανά μπλοκ στην κάρτα SD και έπειτα εγγράφουμε τα 2 CRC bytes. Μετά από την εγγραφή του κάθε μπλοκ ελέγχεται αν τα δεδομένα αυτά έγιναν αποδεκτά από την κάρτα μνήμης. Αυτό υλοποιείται ελέγχοντας την απόκριση που λαμβάνουμε μετά την αποστολή του κάθε πακέτου δεδομένων. Εφόσον τα τελευταία 5 bit της απάντησης αντιστοιχούν σε 0b00101 τότε η εγγραφή των δεδομένων ήταν επιτυχής. Έτσι πραγματοποιείται η εγγραφή των δεδομένων μέχρι να πατηθεί το πλήκτρο. Με το πάτημα του πλήκτρου σταματάει η εκτέλεση της while και αποστέλλονται 2 dummy bytes. Έπειτα για να καταλάβει η SDC ότι δεν θα δεχθεί άλλα δεδομένα στέλνεται το stop transfer data token (0b11111101) και ένα dummy clock που απαιτείται. Έπειτα αναμένεται η ολοκλήρωση της διαδικασίας μέχρι να εμφανιστεί στο DO το 0xff από το busy state της SDC. Στο σημείο αυτό ολοκληρώνεται η ανάλυση των συναρτήσεων που αφορούν την κάρτα SD.

3.5 – PWM (Pulse Width Modulation)

Για την αναπαραγωγή του ήχου είναι αναγκαίο να χρησιμοποιηθεί ένας μετατροπέας ψηφιακού σήματος σε αναλογικό (DAC). Επειδή ο συγκεκριμένος μικροελεγκτής δεν έχει τέτοια βαθμίδα υλοποιημένη χρησιμοποιείται ο PWM που υπάρχει, σε συνδυασμό με ένα RC φίλτρο. Με αυτή την υλοποίηση δημιουργείται ένας απλός DAC για τις απαιτήσεις αυτής της εργασίας. Για να γίνει ενεργοποίηση ή απενεργοποίηση του PWM πρέπει να γίνει μία εγγραφή στον CCPCON2 καταχωρητή. Έπειτα εφόσον είναι ενεργοποιημένος, στον καταχωρητή CCPR2L, εμφανίζεται το διαμορφωμένο κατά PWM σήμα και οδηγείται στο φίλτρο. Η μοναδική ενέργεια που πρέπει να γίνει για την λειτουργία του PWM είναι η αρχικοποίηση κάποιων καταχωρητών που αφορούν την περίοδο και τον χρονισμό.

Με την setup_pwm(); γίνεται ο έλεγχος και η ρύθμιση του PWM.

```
/**

 * Sets up the PWM
 */
void setup_pwm() {

    TRISC1  = 0;            // PORTC1 Output
    T2CKPS1 = 0;            // Prescaler is 1
    T2CKPS0 = 0;            // Prescaler is 1
```

```
    PR2     = 0x50;          // Sets the PWM period
    TMR2ON  = 1;             // Timer 2 is On
    CCP2CON = 0b00001100;    // Selection of PWM mode
}
```

Αρχικά δηλώνεται ότι η PORTC1 είναι έξοδος. Ουσιαστικά ο ακροδέκτης αυτός θα παρέχει το κατά PWM διαμορφωμένο σήμα. Στην λειτουργία PWM στον PIC για να ορισθεί η περίοδος του PWM απαιτείται η ενεργοποίηση του χρονιστή TMR2ON και δίνεται τιμή στον PR2 καταχωρητή. Δίνοντας τιμή στον PR2 δηλώνουμε ουσιαστικά την περίοδο του PWM γιατί ο TMR2ON ξεκινάει από το 0x00h μέχρι να φτάσει την τιμή του PR2. Επιπλέον ορίζεται ότι δεν θα χρησιμοποιηθεί prescaler ορίζοντας τα T2CKPS1:0=0. Με τον CCP2CON=0b00001100 γίνεται ενεργοποίηση του PWM.

3.6 – Η main() και υπόλοιπες συναρτήσεις

Έχοντας ολοκληρώσει την ανάλυση των βασικών συναρτήσεων που υλοποιούν τις βασικές λειτουργίες θα δούμε και τις συναρτήσεις από τις οποίες διαχειρίζονται. Αρχικά αναλύεται η main(); του προγράμματος.

```
void main(void) {
    setup();                 // Start the setup process
    while(1) {
        start_playback();    // Start Playback process
    }
}
```

Η main(); πολύ απλά ξεκινάει καλώντας την συνάρτηση setup(); και έπειτα με μία ατέρμων while καλεί την συνάρτηση start_playback(); με την οποία ξεκινάει η διαδικασία αναπαραγωγής.

```
/**
 * Program Configuration
 */
void setup() {

    setup_lcd();     // Sets up the LCD
    setup_adc();     // Sets up the ADC
    setup_spi();     // Sets up the SPI
    setup_sd();      // Sets up the SD Card
    setup_pwm();     // Sets up the PWM
}
```

Με την setup(); πραγματοποιείται η κλήση όλων των υπολοίπων setup συναρτήσεων που αναλύθηκαν στα προηγούμενα υποκεφάλαια. Ουσιαστικά γίνεται η αρχική διαμόρφωση του προγράμματος και αρχικοποιούνται όλοι οι καταχωρητές και όλες οι βαθμίδες που χρησιμοποιούνται.

```
/**
 * Starts Playback
 */
void start_playback() {

    CCP2CON = 0b00001100; // Enable PWM mode

    print_to_lcd("Play Mode"); // LCD prints "Play Mode"
    __delay_ms(1000);
    read_from_sd();
}
```

Με την start_playback(); γίνεται η αρχή της αναπαραγωγής. Αρχικά ενεργοποιείται ο PWM έτσι ώστε να γίνει εφικτή η αναπαραγωγή. Έπειτα εκτυπώνεται μήνυμα στην οθόνη ότι ενεργοποιήθηκε η λειτουργία αναπαραγωγής και καλεί την συνάρτηση read_from_sd(); με την οποία διαβάζει τα δεδομένα από την SDC. Στην ίδια λογική λειτουργίας είναι και η συνάρτηση εγγραφής start_recording();.

```
/**
 * Start Recording
 */
void start_recording() {
    CCP2CON = 0b00000000; // Disable PWM mode

    print_to_lcd("Record Mode"); // LCD prints "Record Mode"
    __delay_ms(1000);
    write_to_sd();
}
```

Η start_recording(); λόγω ότι το πρόγραμμα σχεδιάστηκε με αυτόν τον τρόπο καλείται και έρχεται πάντα μετά από την διαδικασία ανάγνωσης. Για τον λόγο αυτό στην πρώτη γραμμή κώδικα απενεργοποιείται ο PWM και εμφανίζεται στην οθόνη ότι ενεργοποιήθηκε η λειτουργία εγγραφής. Έπειτα γίνεται η κλήση της write_to_sd(); και ξεκινάει η ηχογράφηση.

Εδώ ολοκληρώνεται η ανάλυση όλων των συναρτήσεων που υλοποιήθηκαν για την εργασία αυτή. Η λογική του προγράμματος είναι πάντα να ενεργοποιείται και να ξεκινάει αρχικά η αναπαραγωγή και όποτε ο χρήστης θελήσει μπορεί να περάσει στην λειτουργία εγγραφής με το πάτημα ενός πλήκτρου. Με την ίδια ενέργεια μπορεί να επιλέγει όποια από τις δύο λειτουργίες επιθυμεί. Ακολουθήθηκε η υλοποίηση αυτή για να υπάρχει λογική δομή στον κώδικα αλλά ταυτόχρονα να γίνει χρήση όσο λιγότερων πλήκτρων ήταν δυνατό για την επίτευξη απλότητας του υλισμικού.

Κεφάλαιο 4^{ο} – Συμπεράσματα και Μελλοντικές προοπτικές

Στην εργασία αυτή έγινε σχεδιασμός και προγραμματισμός μίας συσκευής με στόχο την ηχογράφηση και αναπαραγωγή του ήχου που παράγει η ανθρώπινη φωνή. Το αποτέλεσμα ήταν αρκετά ικανοποιητικό λαμβάνοντας υπόψη ότι ο PIC16F877 που χρησιμοποιήθηκε είναι ένας μεσαίας κατηγορίας μικροελεγκτής ενώ συνήθως σε εφαρμογές που αφορούν τον ήχο, γίνεται χρήση μικροελεγκτών μεγαλύτερων δυνατοτήτων. Επίσης η ανάλυση του ADC που χρησιμοποιήθηκε ήταν 8 bit δηλαδή 256 διαφορετικά επίπεδα ήχου και δεν υπήρχε βαθμίδα DAC αλλά μία κατά προσέγγιση υλοποίησή της κάνοντας χρήση του PWM και του RC φίλτρου όπως αναλύθηκε. Έπειτα από διάφορες δοκιμές παρατηρήθηκε ότι, παρά τις χαμηλές προδιαγραφές της συσκευής, παράγεται αρκετά καλή ποιότητα ήχου και στην ομιλητική φωνή αλλά και σε πολλά διαφορετικά μουσικά αποσπάσματα που ηχογραφήθηκαν. Επίσης, όπως αναφέρθηκε στο πρώτο κεφάλαιο, η συσκευή αυτή παρουσιάζει ακαδημαϊκό ενδιαφέρον δεδομένου ότι αγγίζει πολλούς διαφορετικούς τομείς της πληροφορικής αλλά και άλλων τομέων. Ολοκληρώνεται η εργασία με την ελπίδα να διευρύνει τις γνώσεις του αναγνώστη όπως έγινε με εκείνες του μελετητή της.

Προφανώς όντας μία προσπάθεια υλοποίησης μιας συσκευής από την αρχή υπάρχουν αρκετά σημεία τα οποία μπορούν να βελτιωθούν δίνοντας έτσι στο project αυτό μια μελλοντική προοπτική βελτίωσης. Μία από τις βασικότερες που θα μπορούσε να υλοποιηθεί στην συσκευή αυτή θα ήταν να γίνεται η εγγραφή του ήχου στην κάρτα SD με χρήση κάποιας γνωστής μεθόδου συμπίεσης όπως mp3 και επιπλέον η κάρτα SD να έχει κάποιο format, έτσι ώστε να είναι προσβάσιμη και από άλλα υπολογιστικά συστήματα. Αυτό πολύ πιθανόν να απαιτούσε χρήση κάποιου μεγαλύτερου μικροελεγκτή για να υπάρχει μεγαλύτερη υπολογιστική ισχύ με στόχο επίσης την βελτίωση και της ηχογράφησης, χρησιμοποιώντας μεγαλύτερες αναλύσεις ADC αλλά και κάποια βαθμίδας DAC για την βελτίωση της αναπαραγωγής του ήχου. Άλλες δυνατότητες θα ήταν, η πλοήγηση στην μνήμη, η δυνατότητα pause και γενικότερα περισσότερες επιλογές για τον χρήστη. Τέλος μια επίσης σημαντική βελτίωση θα ήταν η υποστήριξη πέρα των standard

SD card, νεοτέρων εκδόσεων όπως είναι οι SD HC (High Capacity) και οι SD XC (Extra Capacity).

Χάρη στις βελτιώσεις που μπορούν να υλοποιηθούν πάνω σε μία τέτοια συσκευή υπάρχει αρκετά μεγάλο περιθώριο περαιτέρω μελέτης των ενσωματωμένων ψηφιακών συστημάτων ήχου. Κλείνουμε έχοντας την ικανοποίηση της δημιουργίας μίας πλήρως λειτουργικής συσκευής που υλοποιεί τις βασικές λειτουργίες ηχογράφησης και αναπαραγωγής με ένα πολύ ικανοποιητικό ακουστικό αποτέλεσμα.

Βιβλιογραφία

Albert Paul Malvino Ph.D., Βασική Ηλεκτρονική, Εκδόσεις Τζιόλα 2007

Κωνσταντίνος Καλοβρέκτης, Βασικές Δομές Ενσωματωμένων συστημάτων, Varmar Publications

Parag Havaldar & Gerard Medioni, Συστήματα Πολυμέσων, Εκδόσεις Broken Hill 2012

Han-Way Huang, PIC Microcontroller: An Introduction to Software and Hardware Interfacing, Εκδόσεις Thomson Delmar Learning 2005

Tim Wilmshurst, Designing Embedded Systems with Pic Microcontrollers, Εκδόσεις Newnes Elvesier 2010

Dogan Ibrahim, Sd Card Projects using the PIC Microcontroller, Εκδόσεις Newnes Elvesier 2010

PIC16F87XA Data Sheet, Microchip Technologies

HI-TECH C for PIC10/12/16 User's Guide, Microchip Technologies

http://www.avrfreaks.net/forum/lcd-4bit-vs-8bit

http://www.microcontroller-project.com/16x2-lcd-working.html

http://www.microcontroller-project.com/lcd-in-4bit-mode-with-pic-microcontroller.html

https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

http://www.eln.teilam.gr/sites/default/files/Protocola%20Diasyndeshs.pdf

http://www.sciencedirect.com/science/article/pii/S1877705812002792

http://www.allaboutcircuits.com/technical-articles/turn-your-pwm-into-a-dac/

https://en.wikipedia.org/wiki/Pulse-width_modulation

https://en.wikipedia.org/wiki/Low-pass_filter

http://www.learningaboutelectronics.com/Articles/Low-pass-filter-calculator.php#answer1

https://en.wikipedia.org/wiki/Analog-to-digital_converter

https://en.wikipedia.org/wiki/Quantization_(signal_processing)

https://electrosome.com/adc-pic-microcontroller-hi-tech-c/

https://learn.sparkfun.com/tutorials/analog-to-digital-conversion

http://www.nxp.com/documents/data_sheet/TDA7052.pdf

http://www.ohio.edu/people/uijtdeha/ee3954_fall13_10_adc.pdf

https://www.8051projects.net/lcd-interfacing/commands.php

http://www.dinceraydin.com/lcd/commands.htm

http://elm-chan.org/docs/mmc/mmc_e.html

http://elm-chan.org/docs/spi_e.html

https://en.wikipedia.org/wiki/Digital_audio

http://www.eln.teilam.gr/sites/default/files/PCM.pdf

https://en.wikipedia.org/wiki/Digital-to-analog_converter

https://en.wikipedia.org/wiki/Line_level

https://en.wikipedia.org/wiki/Digital_recording

https://en.wikipedia.org/wiki/Digital_audio_workstation

https://en.wikipedia.org/wiki/Data_compression

https://en.wikipedia.org/wiki/Computer_data_storage

http://www.microchip.com/forums/m744278.aspx

http://www.c4learn.com/c-programming/c-bitwise-right-shift/

https://www.youtube.com/watch?v=SToBPCajwc0

https://www.youtube.com/watch?v=9SclQIWkOtk

Παράρτημα Α – Κώδικας της Εφαρμογής

```
/*
 * File:   main.c
 * Author: Zafiris
 *
 * Created on 29 Απριλίου 2016, 12:49 μμ
 */

#pragma config FOSC = HS        // Oscillator Selection bits (HS
oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT
disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config CP = OFF         // FLASH Program Memory Code
Protection bits (Code protection off)
#pragma config BOREN = OFF       // Brown-out Reset Enable bit (BOR
disabled)
#pragma config LVP = OFF        // Low Voltage In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used
for programming)
#pragma config CPD = OFF        // Data EE Memory Code Protection
(Code Protection off)
#pragma config WRT = OFF        // FLASH Program Memory Write Enable
(Unprotected program memory may not be written to by EECON control)


#include <xc.h>

#define _XTAL_FREQ 20e6
#define CS RC2
#define RECORD_INTERRUPT RE0
#define RS RB2
#define EN RB1
#define SPI_DUMMY_CLOCK 0xff
#define SD_START_ADDRESS 0
#define SD_COMMAND_READ_MULTIPLE_BLOCKS 18
#define SD_COMMAND_STOP_TRANSMISSION 12
#define SD_COMMAND_WRITE_MULTIPLE_BLOCKS 25
#define SD_COMMAND_RESPONSE_SUCCESS 0x00
#define SD_COMMAND_GO_IDLE_STATE 0
#define SD_COMMAND_INIT 1
#define SD_COMMAND_SETBLOCKLEN 16
#define BLOCKLEN 512
#define DATA_TOKEN_READ 0xfe
#define DATA_TOKEN_BUSY 0xff
#define DATA_TOKEN_WRITE 0xfc
#define DATA_TOKEN_STOP_TRAN 0xfd
#define DATA_TOKEN_ACCEPTED 0x05
#define LCD_COMMAND_4BITS_2LINES 0x28
#define LCD_COMMAND_CURSOR_OFF 0x0C
#define LCD_COMMAND_ENTRY_MODE 0x06
#define LCD_COMMAND_CLEAR 0x01

void setup_adc();
void setup_pwm();
void setup_spi();
void print_to_lcd(const char*);
void write_to_spi(unsigned char);
unsigned char read_from_spi();
```

```
void sd_command(unsigned char, unsigned long int, unsigned char);
void setup_sd();
void setup();
void handle_sd_command_response(unsigned char);
void start_playback();
void write_to_sd();
void start_recording();
void read_from_sd();
unsigned char read_from_adc();
void lcd_command(unsigned char);
void lcd_data(unsigned char);
void lcd_strobe();
void setup_lcd();

void main(void) {
    setup();                // Start the setup process
    while(1) {
        start_playback();   // Start Playback process
    }
}

/**
 * Sets up the ADC
 */
void setup_adc() {

    TRISA0 = 1;           // PORTA0 Input
    ADCON0 = 0b10000001; // Fosc/32 ADC On
    ADCON1 = 0b10001110; // Port A0 Analog Input Right justified
}

/**
 * Sets up the PWM
 */
void setup_pwm() {

    TRISC1  = 0;              // PORTC1 Output
    T2CKPS1 = 0;              // Prescaler is 1
    T2CKPS0 = 0;              // Prescaler is 1
    PR2     = 0x50;           // Sets the PWM period
    TMR2ON  = 1;              // Timer 2 is On
    CCP2CON = 0b00001100;     // Selection of PWM mode
}

/**
 * Sets up the SPI
 */
void setup_spi(){

    TRISC4 = 1;                     // PORTC4 4bit input
    CS     = 1;                     // Chip Select is high
    RC3    = 0;                     // Clock is low
    RC5    = 0;                     // Data out low
    TRISC2 = TRISC3 = TRISC5 = 0;   // PORTC 2,3,5 bit output
    SSPCON = 0b00100010;            // SPI Master Mode Fosc/64, Clock
Polarity idle state=low level
    SSPEN  = 1;                     // Configuration of SCK,SDO,SDI,
!SS as serial port pins
    SMP    = 1;                     // SSPSTAT 7bit input data sampled
at end of data output time
    CKE    = 1;                     // SSPSTAT 6bit transmit from
transition high -> low
```

```
    CKP    = 0;                      // Idle state for clock is a low
level

}

/**
* Execution of the instructions
*/
void lcd_strobe(){

    EN = 1;          // To execute instructions set EN=1...
    __delay_us(1);   // ... wait for execution...
    EN = 0;          // ... and set EN=0
}

/**
 * Sends the data to the Lcd
 *
 * @param {unsigned char} data    The data passed to the lcd
 */
void lcd_data(unsigned char data){

    RS = 1;               // When RS = 1 sends data
    __delay_ms(80);
    PORTD = (data >> 4); // First 4 bits to PORTD
    lcd_strobe();
    PORTD = data;         // Last 4 bits to PORTD
    lcd_strobe();
}

/**
 * Sends the command to the Lcd
 *
 * @param {unsigned char} command    The command passed to the lcd
 */
void lcd_command(unsigned char command){

    RS = 0;                       // When RS = 0 sends command
    __delay_us(1);
    PORTD = (command >> 4);       // First 4 bits to PORTD
    lcd_strobe();
    PORTD = command;              // Last 4 bits to PORTD
    lcd_strobe();
}

/**
 * Sets up the LCD
 */
void setup_lcd(){

   TRISD=0;      // PORTD as output
   TRISB2=0;     // PORTB2 pin as output
   TRISB1=0;     // PORTB1 pin as output
   PORTD=0;      // PORTD zero value
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);   // Command instruction set
4bits 2 lines mode
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);
   __delay_ms(20);
   lcd_command(LCD_COMMAND_4BITS_2LINES);
   __delay_ms(20);
```

```
   lcd_command(LCD_COMMAND_CURSOR_OFF);     // Command instruction set
cursor off
   lcd_command(LCD_COMMAND_CLEAR);          // Command instruction
clear screen
   lcd_command(LCD_COMMAND_ENTRY_MODE);     // Command instruction set
entry mode Display Shift off ->
}                                           // -> Increment Address
Counter

/**
 * Prints a string to the LCD
 *
 * @param {const char *} string    String to be printed
 */
void print_to_lcd(const char *string) {

    lcd_command(LCD_COMMAND_CLEAR);  // Clear LCD

    while (*string) {                // Sends the string to the LCD
character by character
        lcd_data(*string++);         // It makes the string point to
the next memory address
    }
}

/**
 * Writes data to SPI bus
 *
 * @param {unsigned char} data      Data to write to SPI
 */
void write_to_spi(unsigned char data) {

    SSPBUF = data;   // Send the data to SSPBUF
    while (BF == 0); // Wait until receive of data is complete
}

/**
 * Reads data from SPI bus
 */
unsigned char read_from_spi() {
    SSPBUF = SPI_DUMMY_CLOCK; // Send dummy data to slave
    while (BF == 0);          // Wait that data is received
    return SSPBUF;            // Read the data that slave has sent
back
}

/**
 * Sends a command to the SD Card
 * @param {unsigned char}       command      Command Code
 * @param {unsigned long int}   argument     Argument of the Command
 * @param {unsigned char}       crc          CRC Value
 */
void sd_command(unsigned char command, unsigned long int argument,
unsigned char crc) {

    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(0b01000000 | command);             // OR the command
because Bits 7-8 are always 01
    write_to_spi((unsigned char) (argument >> 24)); // send the four-
byte argument in byte chunks
    write_to_spi((unsigned char) (argument >> 16));
    write_to_spi((unsigned char) (argument >>  8));
    write_to_spi((unsigned char) (argument));
```

```
    write_to_spi(crc);
}

/**
 * Sets up the SD Card
 */
void setup_sd() {

    unsigned char received_data;
    unsigned int counter = 0;
    CS = 1;

    for (unsigned int i = 0; i < 74; i++) {              // Apply 74
clock pulses to SD Card
        write_to_spi(SPI_DUMMY_CLOCK);
    }

    CS = 0;
    __delay_ms(1);
    sd_command(SD_COMMAND_GO_IDLE_STATE, SD_START_ADDRESS, 0x95); //
Software Reset
    received_data = read_from_spi();

    while ((received_data != 1) && (counter < 1000)) {   // Reads from
SPI until received data = 1
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000) {                               // If
received data != 1 command fails
        print_to_lcd("Cmd 0 failed");
        while(1);
    }
    sd_command(SD_COMMAND_INIT, SD_START_ADDRESS, 0xff); //
Initialization process
    received_data = read_from_spi();

    counter = 0;
    while((received_data != 0) && (counter < 1000)) {    // Reads from
SPI until received data = 0
        received_data = read_from_spi();
        sd_command(SD_COMMAND_INIT, SD_START_ADDRESS, 0xff);
        received_data = read_from_spi();
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000){
        print_to_lcd("Cmd 1 failed");                   // If received
data != 0 command fails
        while(1);
    }
    sd_command(SD_COMMAND_SETBLOCKLEN, BLOCKLEN, 0xff); // Set block
size
    received_data = read_from_spi();
    counter = 0;
    while ((received_data != 0) && (counter < 1000)) {  // Reads from
SPI until received data = 0
        received_data = read_from_spi();
        counter++;
    }
    if (counter >= 1000) {
        print_to_lcd("Cmd 16 failed");                  // If received
data != 0 command fails
```

```
        while(1);
    }
    print_to_lcd("SD Card is A-OK");
    __delay_ms(1000);
    SSPCON = 0b00100001;                                // Set Fosc/16
for reading process
}

/**
 * Program Configuration
 */
void setup() {

    setup_lcd();    // Sets up the LCD
    setup_adc();    // Sets up the ADC
    setup_spi();    // Sets up the SPI
    setup_sd();     // Sets up the SD Card
    setup_pwm();    // Sets up the PWM
}

/**
 * SD command response handler
 *
 * @param {unsigned char}  command    Command Code
 */
void handle_sd_command_response(unsigned char command) {

    unsigned char received_data;
    switch(command) {
        case SD_COMMAND_READ_MULTIPLE_BLOCKS:       // For read case
        case SD_COMMAND_WRITE_MULTIPLE_BLOCKS:      // For write case
            received_data = read_from_spi();
            while (received_data != SD_COMMAND_RESPONSE_SUCCESS) { //
If received data not succesful read again
                received_data = read_from_spi();
            }
            break;

        case SD_COMMAND_STOP_TRANSMISSION:              // For stop
tran case
            received_data = read_from_spi();            // Stuff byte
has to be ignored
            received_data = read_from_spi();            // Previously
stuff byte so read again
            while (received_data != SD_COMMAND_RESPONSE_SUCCESS) { //
If received data not succesful read again
                received_data = read_from_spi();
            }

            received_data = read_from_spi();
            while (received_data != DATA_TOKEN_BUSY) {  // Wait until
data out bus goes high for end of the process
                received_data = read_from_spi();
            }
            break;
    }
}

/**
 * Starts Playback
 */
void start_playback() {
```

```
    CCP2CON = 0b00001100; // Enable PWM mode

    print_to_lcd("Play Mode"); // LCD prints "Play Mode"
    __delay_ms(1000);
    read_from_sd();
}

/**
 * Write to SD
 */
void write_to_sd() {

    SSPCON = 0b00100000;                // Set Fosc/4 for writing
process
    print_to_lcd("Recording...");       // LCD prints "Recording"
    unsigned char received_data;
    sd_command(SD_COMMAND_WRITE_MULTIPLE_BLOCKS, SD_START_ADDRESS,
0xff); // Send command Command 25 for multiple write
    handle_sd_command_response(SD_COMMAND_WRITE_MULTIPLE_BLOCKS);
// Handling the command response
    write_to_spi(SPI_DUMMY_CLOCK);                          // Dummy
clocks before sending data packet
    write_to_spi(SPI_DUMMY_CLOCK);

    while (!RECORD_INTERRUPT) {                             // While
button is not pressed enter loop
        write_to_spi(DATA_TOKEN_WRITE);                     // Send
the data token for the data packet

        for (unsigned int i = 0; i < BLOCKLEN; i++) {       // For
every data block
            unsigned char adc_data = read_from_adc();
            write_to_spi(adc_data);                         // write
the result of the adc to the sdc
        }

        write_to_spi(SPI_DUMMY_CLOCK);                      // 1st CRC
byte
        write_to_spi(SPI_DUMMY_CLOCK);                      // 2nd CRC
byte
        received_data = read_from_spi();                    // Read
data response after first block sent

        while ((received_data & 0b00011111) != DATA_TOKEN_ACCEPTED ) {
// Check continuously that data was accepted
            received_data = read_from_spi();
        }

        while (received_data != DATA_TOKEN_BUSY){           // Wait
until data out bus goes high for end of the process
            received_data = read_from_spi();
        }
    }

    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(SPI_DUMMY_CLOCK);
    write_to_spi(DATA_TOKEN_STOP_TRAN);                     // Button
was pressed so send stop tran token
    write_to_spi(SPI_DUMMY_CLOCK);                          // Dummy
byte after stop tran token

    received_data = read_from_spi();
```

```
    while (received_data != DATA_TOKEN_BUSY){                // Wait
until data out bus goes high for end of the process
        received_data = read_from_spi();
    }
}

/**
 * Reads from ADC
 */
unsigned char read_from_adc() {
    GO = 1;          // When GO=1 starts A/D conversion
    while(GO);       // Wait until conversion is complete (bit cleared
by hardware when conversion complete)

    return ADRESL;  // Return converted data
}

/**
 * Start Recording
 */
void start_recording() {
    CCP2CON = 0b00000000; // Disable PWM mode

    print_to_lcd("Record Mode"); // LCD prints "Record Mode"
    __delay_ms(1000);
    write_to_sd();
}

/**
 * Read from SD
 */
void read_from_sd() {
    SSPCON =0b00100001;                                       // Set
Fosc/16 for reading process
    print_to_lcd("Listening...");                             // LCD
prints "Listening"
    unsigned char received_data;
    sd_command(SD_COMMAND_READ_MULTIPLE_BLOCKS, SD_START_ADDRESS,
0xff); // Send command Command 18 for multiple read
    handle_sd_command_response(SD_COMMAND_READ_MULTIPLE_BLOCKS);
// Handling the command response

    while (!RECORD_INTERRUPT) {                               //
While button is not pressed enter loop
        received_data = read_from_spi();                      //
Read first appeared data from spi
        while (received_data != DATA_TOKEN_READ) {            //
Read continuously data until read data token appears
            received_data = read_from_spi();
        }

        for (unsigned int i = 0; i < BLOCKLEN; i++) {         // For
every data block
            received_data = read_from_spi();                  //
Read the data from sdc

            __delay_us(16.5);
            CCPR2L = received_data;                           // And
send them to the PWM for reproduction
        }

        write_to_spi(SPI_DUMMY_CLOCK);                        // 1st
CRC byte
```

```
        write_to_spi(SPI_DUMMY_CLOCK);                          // 2nd
CRC byte
    }

    sd_command(SD_COMMAND_STOP_TRANSMISSION, 0, 0xff);          //
Button was pressed so send stop transmission command
    handle_sd_command_response(SD_COMMAND_STOP_TRANSMISSION);   //
Handling the command response

    start_recording();                                          //
Start the recording process
}
```

Παράρτημα Β – Εικόνες και Πίνακες