ΜΕΛΕΤΗ ΕΡΕΥΝΑ ΚΑΙ ΚΑΤΑΣΚΕΥΗ
ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΧΟΡΔΙΣΜΑΤΟΣ ΚΙΘΑΡΑΣ

από τον
Μακρίδη Άγγελο

Πτυχιακή εργασία που παρουσιάστηκε στους υπεύθυνους καθηγητές
Γρηγόρη Νικολάου, Κωνσταντίνο Αλαφοδήμο

Σχολή Τεχνολογικών Εφαρμογών – Τμήμα Αυτοματισμού
ΤΕΙ Πειραιά
Αιγάλεω
Μάρτιος 2013

ΜΕΛΕΤΗ ΕΡΕΥΝΑ ΚΑΙ ΚΑΤΑΣΚΕΥΗ
ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΧΟΡΔΙΣΜΑΤΟΣ ΚΙΘΑΡΑΣ

ΜΕΛΕΤΗ ΕΡΕΥΝΑ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΧΟΡΔΙΣΜΑΤΟΣ ΚΙΘΑΡΑΣ

Ι. Εισαγωγή

Το θέμα αυτής της πτυχιακής εργασίας είναι η κατασκευή ενός μικροϋπολογιστικού συστήματος που όχι μόνο δείχνει την συχνότητα στην οποία βρίσκεται μια συγκεκριμένη χορδή κιθάρας αλλά και το αυτόματο χόρδισμά της στη σωστή συχνότητα μέσω ενός ηλεκτρικού κινητήρα ο οποίος θα ελέγχεται από έναν προγραμματισμένο μικροϋπολογιστή.

Το σήμα, μέσω ενός καλωδίου, θα μεταφέρεται από την κιθάρα στην είσοδο της κατασκευής όπου θα ενισχύεται τόσο ώστε να μπορεί "φανεί" σαν είσοδος στους ακροδέκτες του μικροϋπολογιστή. Στη συνέχεια θα υπολογίζεται η συχνότητα με την οποία πάλλεται η συγκεκριμένη χορδή που ταλαντώνεται εκείνη τη στιγμή και μέσω του αποθηκευμένου κώδικα στη μνήμη του μικροϋπολογιστή, θα συγκρίνεται με τη σωστή συχνότητα. Τέλος, ο μικροϋπολογιστής θα δίνει εντολή στον ηλεκτροκινητήρα την κατεύθυνση που πρέπει να περιστραφεί (δεξιόστροφα ή αριστερόστροφα) με αποτέλεσμα το σωστό χόρδισμα της χορδής.

Όλη η διαδικασία θα ελέγχεται από τον χρήστη με τη χρήση κουμπιών και μιας οθόνης υγρών κρυστάλλων που θα εμφανίζει σε ποιο σημείο του menu βρίσκεται και τι λειτουργία εκτελείται εκείνη τη στιγμή.

Σκοπός αυτής της κατασκευής είναι το σωστό χόρδισμα της κιθάρας στο κανονικό ή σε διαφορετικό από το κανονικό χόρδισμα γρήγορα και έγκυρα ακόμα και σε περιβάλλον που υπάρχει θόρυβος. Επίσης, δίνεται η δυνατότητα της εύκολης αλλαγής χορδών μέσω συγκεκριμένης επιλογής στο πρόγραμμα με την συνεχή περιστροφή μέσω του ηλεκτρικού κινητήρα ο οποίος θα περιστρέφεται δεξιόστροφα ή αριστερόστροφα, αναλόγως αν ο χρήστης θέλει να αφαιρέσει την

παλιά ή κομμένη χορδή ή να τοποθετήσει μια καινούρια.

Η κατασκευή αυτή απευθύνεται σε μουσικούς που ασχολούνται με την κιθάρα και χρειάζονται ένα γρήγορο και εύκολο τρόπο να χορδίσουν την κιθάρα τους με ακρίβεια είτε στην καθημερινή επαφή με το όργανο – μελέτη ή και διασκέδαση- ,είτε λόγω πίεσης χρόνου, όπως λόγου χάρη σε μια συναυλία ή κάποια πρόβα. Επίσης, ένα επιπλέον πρόβλημα που λύνει η κατασκευή αυτή είναι η δύστροπη διαδικασία της αντικατάσταση μιας χορδής, όπως αναφέρθηκε και προηγουμένως, καθώς είναι από αυτές τις διαδικασίες στην κιθάρα που απαιτούν χρόνο και έχουν σχετική ταλαιπωρία λόγω της συνεχής περιστροφής των κλειδιών της κιθάρας με το χέρι.

ΙΙ. Θεωρητικό υπόβαθρο

Για να γίνει κατανοητό το περιεχόμενο αυτής της πτυχιακής εργασίας και κατασκευής θα πρέπει πρώτα να αναφερθούν κάποια βασικά σημεία περί συχνότητας, μουσικής νότας, αρμονικής συχνότητας και χορδής. Φυσικά δεν μπορεί να γίνει πλήρης ανάλυση όλων των παραπάνω παραμέτρων σε μία μόνο εργασία. Αυτό θα απαιτούσε πληθώρα τόμων και βιβλιογραφίας και στο τέλος θα υπήρχε ακόμα αμφιβολία αν έχει όντως καλυφθεί ολοκληρωτικά το θέμα. Σε αυτή την εργασία θα υποδειχθούν όσα θεωρητικά στοιχεία χρειάστηκαν για την κατασκευή του μικροϋπολογιστικού αυτού συστήματος χορδίσματος κιθάρας και κάποια άλλα για την αποσαφήνιση κάποιων άλλων σημείων.

Α. Συχνότητα

Στη συγκεκριμένη εργασία με τον όρο συχνότητα εννοείται η συχνότητα του κύματος η οποία στα μαθηματικά ορίζεται ως

$$f = \upsilon / \lambda$$

όπου:

f η συχνότητα του κύματος,

υ η ταχύτητα του κύματος και

λ το μήκος του κύματος.

Στη περίπτωσή μας, η f είναι η συχνότητα με την οποία πάλλεται η χορδή, υ η ταχύτητα του ήχου που υπολογίζεται στα 345 m/s σε θερμοκρασία δωματίου και λ το μήκος κύματος της χορδής. Η συχνότητα φυσικά μετριέται σε Hertz (Hz).

Γενικά το ανθρώπινο αυτί μπορεί να ακούσει συχνότητες από 20Hz έως 20.000Hz. Όσο αυξάνεται η συχνότητα ο ήχος γίνεται αντιληπτός ως πιο ψηλός ή "πρίμος" και όσο μειώνεται πιο χαμηλός ή "μπάσος".

Οι συχνότητες που είναι έξω από το ακουστικό πεδίου του ανθρώπου δεν είναι απαραίτητο ότι δεν γίνονται αντιληπτές από τον άνθρωπο. Για παράδειγμα, ένα υπογούφερ σαν αυτά που χρησιμοποιούνται στους κινηματογράφους ή ακόμα και στους οικιακούς χώρους μπορεί να παράξει συχνότητες 19, 15, ακόμα και 10 Hz, δηλαδή να πάλλεται ο κώνος του ηχείου 10 φορές το δευτερόλεπτο. Αυτός ο ήχος δεν γίνεται αντιληπτός από το ανθρώπινο αυτί αλλά από το σώμα του

καθώς αισθάνεται τον παλμό που συντονίζεται ο αέρας μέσω του ηχείου.

Σε γενικές γραμμές η συχνοτική έκταση ενός πιάνου, οι συχνότητες που καλύπτει ένα πιάνο, αντικατοπτρίζει ικανοποιητικά το σύνολο των συχνοτήτων που μπορούν να αναπαραχθούν από ένα μουσικό όργανο. Φυσικά υπάρχουν εξαιρέσεις όπως κάποια εκκλησιαστικά όργανα που μπορούν να παράξουν συχνότητες των 8Hz ή άλλα πνευστά που υπερβαίνουν την υψηλότερη συχνότητα που μπορεί να παράξει το πιάνο. Όμως, καθώς η συντριπτική πλειοψηφία των μουσικών οργάνων καλύπτεται από την συχνοτική έκταση ενός πιάνου, το συγκεκριμένο όργανο χρησιμοποιείται ως μέτρο σύγκρισης για το "ηχητικό ύψος" του κάθε οργάνου.

Β. Μουσική νότα

Δεν θα ήταν δυνατόν βέβαια οι μουσικοί σε όλο τον κόσμο να μιλούσαν για τις νότες του μουσικού τους οργάνου με την τιμή της συχνότητας που αντιστοιχεί. Οι μουσικές νότες όπως αυτές τις αντιλαμβάνονται οι μουσικοί και οι ονομασίες τους χρησιμοποιούνταν πριν ακόμα έρθει στην επιφάνεια η επιστημονική τους τεκμηρίωση, τοποθέτηση και συγκεκριμένη συχνοτική έκταση. Κοινός τόπος είναι το πιάνο ως μουσικό όργανο που όπως ειπώθηκε παραπάνω, τα μουσικά όργανα τοποθετούνται ως προς το τονικό ύψος.

Το πιάνο λοιπόν, έχει 88 πλήκτρα με το άκρως αριστερό πλήκτρο να είναι το πιο "μπάσο" με συχνότητα 27,50Hz και ονομασία A0 και το άκρως δεξιό πλήκτρο να είναι είναι το πιο "πρίμο" με συχνότητα 4186,01Hz και ονομασία C8. Η χαρακτηριστική νότα που κουρδίζονται τα όργανα μιας ορχήστρας (αλλά και όλα τα όργανα στην συντριπτική πλειοψηφία τους) είναι η A4 στα 440,00Hz. Ένα μικρό όργανο που υποδεικνείει αυτή τη νότα είναι το γνωστό "διαπασών" το οποίο όταν πάλλεται παράγει αυτή ακριβώς της συχνότητα.

Το πως θεσπίστηκαν αυτές οι συχνότητες και η ιστορική αναδρομή δεν είναι το αντικείμενο αυτής της εργασίας. Παρ' όλα αυτά θα ήταν παράλειψη να μην δοθεί ο τύπος με τον οποίο μπορεί να βρεθεί η συχνότητα μιας νότας όταν γνωρίζουμε πόσες βήματα/νότες απέχει από μία άλλη ή από την "βασική" νότα A4.

$$f_n = f_0 * (\alpha)^n$$

όπου:

f_0 η συχνότητα της νότας που γνωρίζουμε

n ο αριθμός των βημάτων/νοτών από τη μία νότα στην άλλη, θετικός ή αρνητικός

f_n η συχνότητα που θέλουμε να βρούμε που απέχει n νότες/βήματα απόσταση

a η δωδέκατη ρίζα του 2, $(2)^{1/12}$ είναι κατά προσέγγιση ο αριθμός 1,059463094359

Ο παρακάτω πίνακας παραθέτει τις συχνότητες ενός πιάνου 88 πλήκτρων με τον αριθμό των πλήκτρων και την επιστημονική ονομασία τους.

Αριθμός πλήκτρου	Επιστημονική ονομασία	Συχνότητα (Hz)
88	C8	4186.01
87	B7	3951.07
86	A♯7/B ♭ 7	3729.31
85	A7	3520.00
84	G♯7/A ♭ 7	3322.44
83	G7	3135.96
82	F♯7/G ♭ 7	2959.96
81	F7	2793.83
80	E7	2637.02
79	D♯7/E ♭ 7	2489.02
78	D7	2349.32
77	C♯7/D ♭ 7	2217.46
76	C7	2093.00
75	B6	1975.53
74	A♯6/B ♭ 6	1864.66
73	A6	1760.00
72	G♯6/A ♭ 6	1661.22
71	G6	1567.98
70	F♯6/G ♭ 6	1479.98
69	F6	1396.91
68	E6	1318.51
67	D♯6/E ♭ 6	1244.51
66	D6	1174.66
65	C♯6/D ♭ 6	1108.73
64	C6	1046.50

Αριθμός πλήκτρου	Επιστημονική ονομασία	Συχνότητα (Hz)
63	B5	987.767
62	A♯5/B ♭ 5	932.328
61	A5	880.000
60	G♯5/A ♭ 5	830.609
59	G5	783.991
58	F♯5/G ♭ 5	739.989
57	F5	698.456
56	E5	659.255
55	D♯5/E ♭ 5	622.254
54	D5	587.330
53	C♯5/D ♭ 5	554.365
52	C5	523.251
51	B4	493.883
50	A♯4/B ♭ 4	466.164
49	A4 A440	440.000
48	G♯4/A ♭ 4	415.305
47	G4	391.995
46	F♯4/G ♭ 4	369.994
45	F4	349.228
44	E4	329.628
43	D♯4/E ♭ 4	311.127
42	D4	293.665
41	C♯4/D ♭ 4	277.183
40	C4	261.626
39	B3	246.942
38	A♯3/B ♭ 3	233.082
37	A3	220.000
36	G♯3/A ♭ 3	207.652
35	G3	195.998
34	F♯3/G ♭ 3	184.997
33	F3	174.614
32	E3	164.814
31	D♯3/E ♭ 3	155.563
30	D3	146.832

Αριθμός πλήκτρου	Επιστημονική ονομασία	Συχνότητα (Hz)
29	C♯3/D♭3	138.591
28	C3	130.813
27	B2	123.471
26	A♯2/B♭2	116.541
25	A2	110.000
24	G♯2/A♭2	103.826
23	G2	97.9989
22	F♯2/G♭2	92.4986
21	F2	87.3071
20	E2	82.4069
19	D♯2/E♭2	77.7817
18	D2	73.4162
17	C♯2/D♭2	69.2957
16	C2	65.4064
15	B1	61.7354
14	A♯1/B♭1	58.2705
13	A1	55.0000
12	G♯1/A♭1	51.9131
11	G1	48.9994
10	F♯1/G♭1	46.2493
9	F1	43.6535
8	E1	41.2034
7	D♯1/E♭1	38.8909
6	D1	36.7081
5	C♯1/D♭1	34.6478
4	C1	32.7032
3	B0	30.8677
2	A♯0/B♭0	29.1352
1	A0	27.5000

Αυτός ο πίνακας έχει χρωματιστεί αντίστοιχα για να μοιάζει με τα πλήκτρα του πιάνου. Έτσι, στην στήλη του αριθμού του πλήκτρου, όποιο πλαίσιο είναι χρωματισμένο με μαύρο, σημαίνει ότι αυτή η νότα είναι η δίεση της προηγούμενης (ή η ύφεση της επόμενης, είναι ένα και το αυτό).

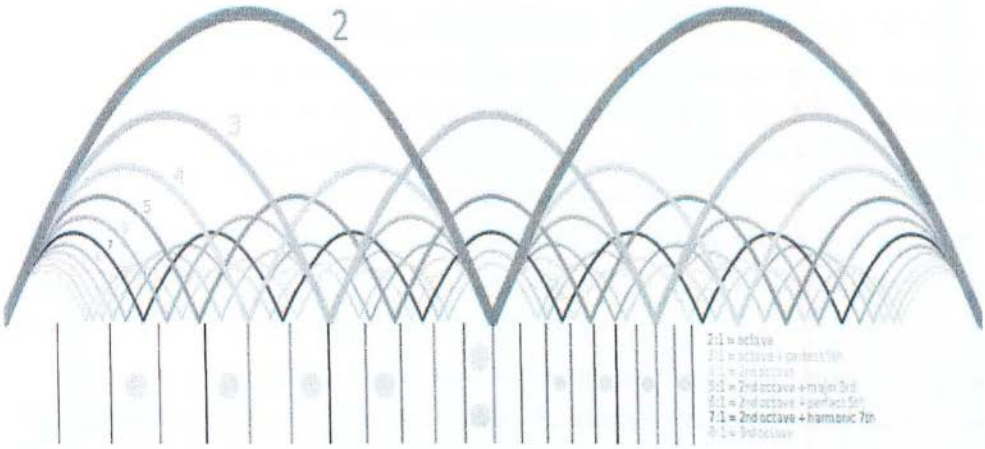
Παρατηρείται ότι η διαφορά της συχνότητας μιας νότας από την επόμενη αυξάνεται λογαριθμικά. Έτσι για παράδειγμα, η διαφορά από την νότα A0 στην ακριβώς επόμενη A#0 είναι 1,6352 Hz ενώ η διαφορά της νότας B7 με την επόμενη και τελευταία νότα του πιάνου C8 είναι 234,94 Hz. Φυσικά, καθώς η αύξηση είναι λογαριθμική, σε κάθε διπλασιασμό της συχνότητας μιας νότας βρίσκεται η ίδια νότα αλλά στην αμέσως μεγαλύτερη οκτάβα, για παράδειγμα η νότα A1 έχει συχνότητα 55,00 Hz, η νότα A2 110,00 Hz, η νότα A3 220,00 Hz κ.ο.κ.

Γ. Αρμονική συχνότητα

Αρμονική συχνότητα ενός κύματος είναι η συχνότητα που εμφανίζεται σε ακέραια πολλαπλάσια από την θεμελιώδη συχνότητα. Αν δηλαδή η θεμελιώδη συχνότητα είναι η f, οι αρμονικές συχνότητες θα έχουν συχνότητες 2f,3f, 4f κ.ο.κ. Πρακτική εφαρμογή βρίσκουμε στα μουσικά όργανα όπως είναι η κιθάρα, το φλάουτο, ακόμα και το τύμπανο, παρά σε κάποια γεννήτρια τόνων σε ένα εργαστήριο.

Συγκεκριμένα στη κιθάρα, όταν παιχτεί η νότα E2 η οποία έχει συχνότητα 82,4069 Hz οι αρμονικές θα έχουν συχνότητα 164,8138 Hz, 247,2207Hz, 329,6276 Hz κ.ο.κ. Αυτό πρακτικά σημαίνει ότι για να μετρηθεί η συχνότητα της νότας της κιθάρας που παίζεται μια συγκεκριμένη στιγμή πρέπει πρώτα το σήμα να μην φέρει και τις αρμονικές του συχνότητες αλλά μόνο την θεμελιώδη συχνότητα, αυτή δηλαδή που πραγματικά παίχτηκε.

Αυτές οι αρμονικές, δυσκολεύουν την μέτρηση για την εύρεση της συχνότητας καθώς οι νέες αυτές, σε σχέση με την θεμελιώδη, συχνότητες έχουν μεγάλο πλάτος, αρκετό για να "περάσουν" στην μέτρηση ως οι κύριες. Το παρακάτω σχήμα δείχνει τις αρμονικές συχνότητες και ο λόγος σε σχέση με την θεμελιώδη συχνότητα.



Από το παραπάνω σχήμα παρατηρείται ότι οι αρμονικές συχνότητες μπορούν να αποπροσανατολίσουν μια μέτρηση συχνότητας και πρέπει να χρησιμοποιηθεί πολύ προσεκτικά ένα ζωνοπερατό φίλτρο (band-pass filter) είτε στον τομέα του software είτε του hardware. Βέβαια, δεν έχουν όλες οι αρμονικές το κατάλληλο πλάτος για να αλλοιώσουν μια μέτρηση συχνότητας. Εμπειρικά, σε μια μέτρηση συχνότητας στην κιθάρα, εμφανίζονταν συχνότητες που υποδείκνυαν την ύπαρξη και επιρροή της μέχρι και της 5ης αρμονικής δηλαδή συχνοτήτων τριών νοτών ψηλότερα από την θεμελιώδη συχνότητα.

Δ. Χορδή Κιθάρας

Η κιθάρα έχει συνολικά έξι χορδές. Υπάρχουν βέβαια και άλλες υλοποιήσεις με δώδεκα ή με επτά χορδές αλλά η συντριπτική πλειοψηφία των κιθάρων κατασκευάζονται με θέσεις για έξι χορδές. Θα πρέπει πρώτα να διευκρινιστεί ότι ως πρώτη χορδή θεωρείται η χορδή που βρίσκεται πρώτη από τα δεξιά στο μπράτσο της κιθάρας και όχι η πρώτη από τα αριστερά, που συνηθίζεται να χορδίζεται πρώτα. Η πρώτη, λοιπόν, χορδή έχει συνήθως πάχος από 0,08mm με 0,13mm, ανάλογα με την χρήση και το ηχόχρωμα που θέλει να βγάλει ο μουσικός, και αυξάνεται από τη δεύτερη χορδή μέχρι και την έκτη η οποία μπορεί να έχει πάχος από 0,38mmμέχρι και 0,56mm. Όσο μεγαλύτερο είναι το πάχος της χορδής, τόσο πιο μπάσο γίνεται το ηχόχρωμα της χορδής ενώ όσο μικρότερο είναι το πάχος, το ηχόχρωμα γίνεται πιο ψιλό και πρίμο.

Οι νότες στις οποίες χορδίζονται οι χορδές είναι οι εξής: Στην νότα Ε2 χορδίζεται η έκτη

χορδή, δηλαδή στα 82,41Hz, η πέμπτη χορδή στην νότα A2 110,00Hz, τέταρτη στην νότα D3 που έχει συχνότητα 146,83Hz, η τρίτη χορδή στην νότα G3 στα 196 Hz, η δεύτερη χορδή στην νότα B3 στη συχνότητα 246,94Hz και τέλος η πρώτη χορδή που είναι η E4 στα 329,63Hz. Μια πιο προσεκτική ματιά θα δείξει ότι όλες οι χορδές έχουν τέσσερις νότες απόσταση η μία από την άλλη με εξαίρεση την δεύτερη χορδή που από την τρίτη έχει απόσταση μόνο τρεις νότες. Αυτό το χόρδισμα λέγεται standard tuning γιατί είναι το πιο συνηθισμένο. Υπάρχουν και άλλα χορδίσματα όπως το drop D tuning, το open G tuning όπως και άλλα τα οποία διαφοροποιούνται από το standard tuning αλλά κυμαίνονται στο ίδιο συχνοτικό ύψος.

Η χορδές δεν μπορούν να χορδιστούν σε οποιαδήποτε συχνότητα. Μετά από κάποιο συχνοτικό ύψος, οι χορδές δεν αντέχουν και σπάνε. Εμπειρικά αυτό συμβαίνει όταν χορδιστεί μια χορδή από τέσσερις νότες και πάνω, ψηλότερα από το standard tuning. Για παράδειγμα, αν η έκτη χορδή που στο standard tuning χορδίζεται στην νότα E2 με συχνότητα 82,41 Hz, χορδιστεί πάνω από την νότα A2 που βρίσκεται στα 110,00 Hz, είναι πολύ πιθανό να σπάσει σε πολύ μικρό χρονικό διάστημα. Σε όλα λοιπόν τα χορδίσματα, είτε είναι το standard tuning είτε είναι κάποιο άλλο, οι χορδές δεν υπερβαίνουν αυτές τις συχνότητες αλλά τις περισσότερες φορές παίρνουν τιμές ίσες ή μικρότερες από αυτές του standard tuning.

Κατά την μέτρηση, λοιπόν, της συχνότητας μιας χορδής της κιθάρας, ένα βαθυπερατό φίλτρο στην συγκεκριμένη συχνότητα που είναι 4 νότες πάνω από το standard tuning της χορδής που χορδίζεται, όχι μόνο είναι απαραίτητο και αποτελεσματικό αλλά είναι και λογικό καθώς η χορδή δεν θα μπορούσε να αναπαράξει ποτέ τέτοια συχνότητα. Με αυτόν τον τρόπο μπορεί αυτό το φίλτρο να φιλτράρει τις διάφορες αρμονικές συχνότητες και να εγγυηθεί ένα ακριβές αποτέλεσμα.

ΙΙΙ. Μεθοδολογία

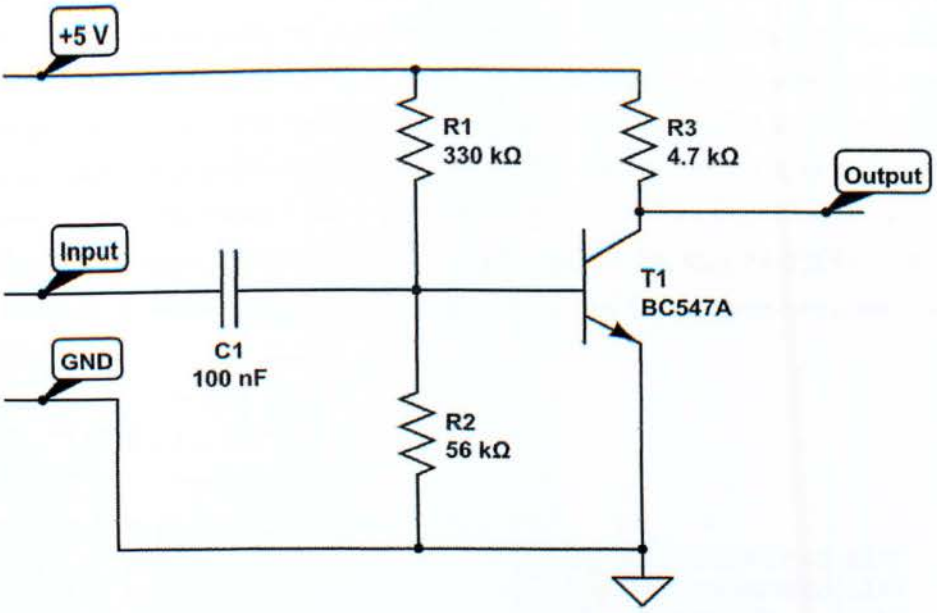
A. Hardware

Σε αυτή την ενότητα θα παρουσιαστούν τα υλικά και ο λόγος που επιλέχθηκαν, η συνδεσμολογία, το κύκλωμα και η γενικότερη μεθοδολογία της κατασκευής όσον αφορά το hardware. Με αυτόν τον τρόπο θα παρουσιαστεί βήμα προς βήμα η διαδικασία της λειτουργίας από την είσοδο του σήματος της κιθάρας στην κατασκευή μέχρι και την κίνηση του servo κινητήρα για το αυτόματο χόρδισμα της χορδής.

1. Κύκλωμα ενίσχυσης

Η μεταφορά του σήματος από την κιθάρα στην κατασκευή γίνεται μέσω ενός καλωδίου κιθάρας με μονοφωνικό "καρφί" ή αλλιώς male audio jack των 6,35 mm. Αυτό του τύπου το καλώδιο είναι που χρησιμοποιείται από τη συντριπτική πλειοψηφία των ηλεκτρικών και ηλεκτροακουστικών κιθάρων. Το ηλεκτρικό σήμα δημιουργείται στην κιθάρα μέσω των μαγνητών στην ηλεκτρική κιθάρα ή μέσω κάποιου μικροφώνου ή πιεζοηλεκτρικού κυκλώματος στην ηλεκτροακουστική κιθάρα. Όμως το ηλεκτρικό σήμα αυτό το οποίο περιέχει και την πληροφορία της συχνότητας της χορδής που πάλλεται, είναι πολύ μικρό, της τάξεως των 50-60 mV. Για αυτό τον λόγο χρειάστηκε ένα κύκλωμα ενίσχυσης το οποίο θα ενισχύει το σήμα από την κιθάρα σε ένα σήμα που θα μπορεί να "δει" ο μικροϋπολογιστής σαν είσοδο. Στην προκειμένη περίπτωση το σήμα από περίπου 50 mV ενισχύεται στα 2,5 Volts και στη συνέχεια μεταφέρεται σε μια από τις εισόδους του μικροϋπολογιστή.

Παρακάτω δίδεται το κύκλωμα ενίσχυσης:

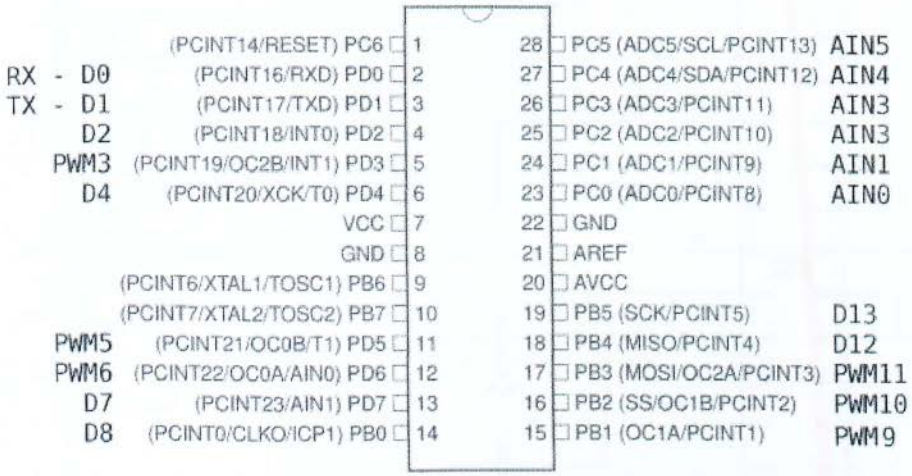


Χρησιμοποιήθηκε ο πυκνωτής C1 για να μπλοκάρει το DC σήμα και να περάσει μόνο το AC και στη συνέχεια ένας διαιρέτης τάσης αποτελούμενος από τις αντιστάσεις R1 και R2 για το σωστό biasing που συμβάλει στη σωστή λειτουργία του τρανζίστορ καθ' όλη τη διάρκεια της μέτρησης. Για τρανζίστορ χρησιμοποιήθηκε ο γνωστός BJT (bipolar junction transistor) τύπου NPN, BC547 ο οποίος χρησιμοποιείται συχνά σε ενισχυτικές και προ-ενισχυτικές κατασκευές. Μέσω της αντίστασης R3 τροφοδοτείται ο BC547 έτσι ώστε το σήμα να ενισχύεται αλλά να μην φτάνει σε παραμόρφωση και clipping. Για τη σύνδεση του καλωδίου της κιθάρας με την κατασκευή χρησιμοποιήθηκε ένα "θηλυκό" βύσμα audio jack 6,35mm το οποίο έχει δύο ακροδέκτες: Ο ένας ακροδέκτης φέρει το σήμα της κιθάρας και ο δεύτερος ενώνεται στην γείωση. Το ενισχυμένο, πλέον, σήμα οδηγείται σε μία από τις εισόδους του μικροϋπολογιστή.

2. Μικροϋπολογιστής

Ο μικροϋπολογιστής που επιλέχθηκε για την κατασκευή αυτή είναι ο ATMEGA 328P-PU της εταιρίας ATMEL. Είναι ένας 8-bit μικροϋπολογιστής με συνολικά 28 ακροδέκτες εκ των οποίων, στο αναπτυξιακό σύστημα που υλοποιήθηκε, 14 χρησιμοποιούνται ως ψηφιακοί είσοδοι/έξοδοι και 6 ως αναλογικοί. Έχει μνήμη SRAM 2K Bytes, Flash Memory 32KB, EEPROM 1KB και η

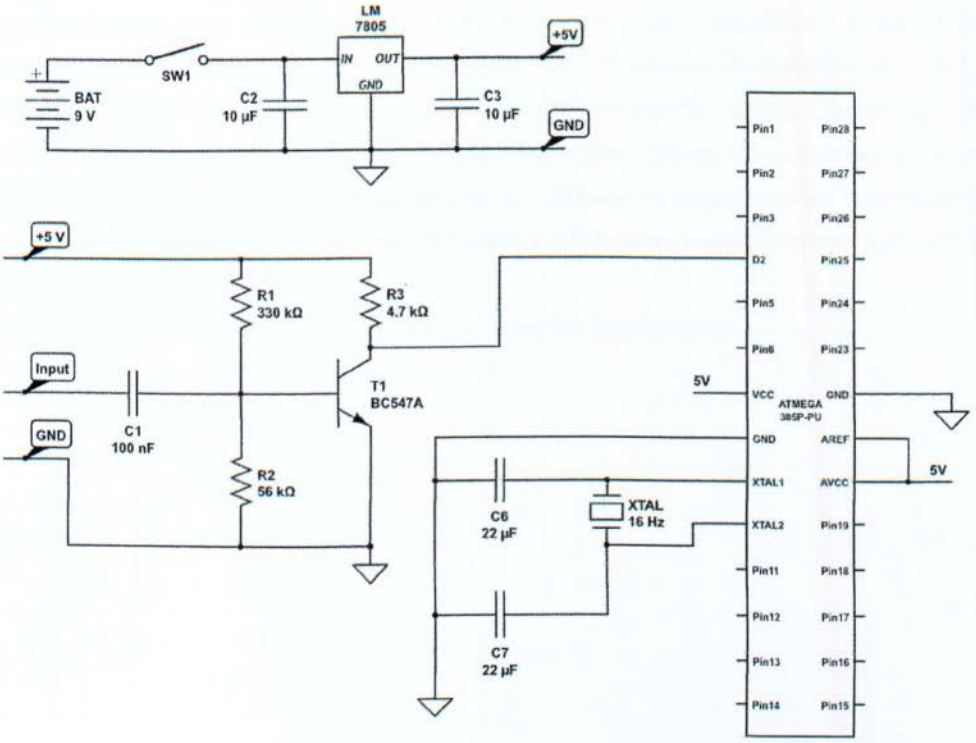
ταχύτητα του ρολογιού είναι στα 16MHz. Η τάση λειτουργίας του κυμαίνεται στα στις τιμές 1,8 V – 5,5 V. Ο προγραμματισμός του έγινε στο αναπτυξιακό σύστημα Arduino Uno R3 το οποίο συνδέεται με τον Η/Υ με μια σύνδεση USB. Για να λειτουργήσει το αναπτυξιακό σύστημα χρειάζεται τροφοδοσία 7-12V (προτεινόμενη) την οποία μπορεί να δεχθεί μέσω του καλωδίου USB, από εξωτερική τροφοδοσία μέσω του 2,1mm βύσματος ρεύματος DC ή από μπαταρία. Το παρακάτω σχήμα παρουσιάζει τον ATMEGA 328P-PU με την ονομασία/λειτουργία των ακροδεκτών του. Σημειώνεται ότι η συμπληρωματικές ονομασίες στα άκρα του σχεδίου είναι αυτές που ονομάτισε το αναπτυξιακό σύστημα Arduino Uno R3 και θα χρησιμοποιούνται από εδώ και στο εξής.



Οι ψηφιακοί είσοδοι/έξοδοι είναι στους ακροδέκτες 2-6 και 11-19 με τις ονομασίες RX-D0, TX-D1, D2, PWM2 (D3), D4, PWM5 (D5), PWM6 (D6), D7, D8, PWM9 (D9), PWM10 (D10), PWM11 (D11), D12 και D13 αντίστοιχα. Οι αναλογικές είσοδοι είναι στους ακροδέκτες 23-28 με τις ονομασίες AIN0-5. Φυσικά όπου δεν υπάρχει συμπληρωματική ονομασία, χρησιμοποιείται η αρχική ονομασία του ATMEGA 328P-PU.

Για να λειτουργήσει ο ATMEGA 328P-PU εκτός αναπτυξιακού, πρέπει οι ακροδέκτες 8 και

22 να γειωθούν και οι 7, 21 και 22 να συνδεθούν στην τάση. Για να λειτουργήσει επίσης χρειάζεται στους ακροδέκτες 9 και 10 η προσθήκη εξωτερικού ρολογιού 16MHz το οποίο θα συνδέεται στην γείωση μέσω δύο πυκνωτών της τάξεως των 22pF. Στην κατασκευή χρησιμοποιήθηκε ως πηγή ρεύματος μια μπαταρία 9V, άρα χρησιμοποιήθηκε ο regulator LM7805 για να μετατρέψει τα 9V σε 5V όπως επίσης και δύο πυκνωτές τις τάξεως των 10uF που θα παρεμβάλλονται ανάμεσα στις εξόδους του. Παρακάτω δίδεται το σχήμα το οποίο περιλαμβάνει και το προηγούμενο στάδιο, του κυκλώματος ενίσχυσης.



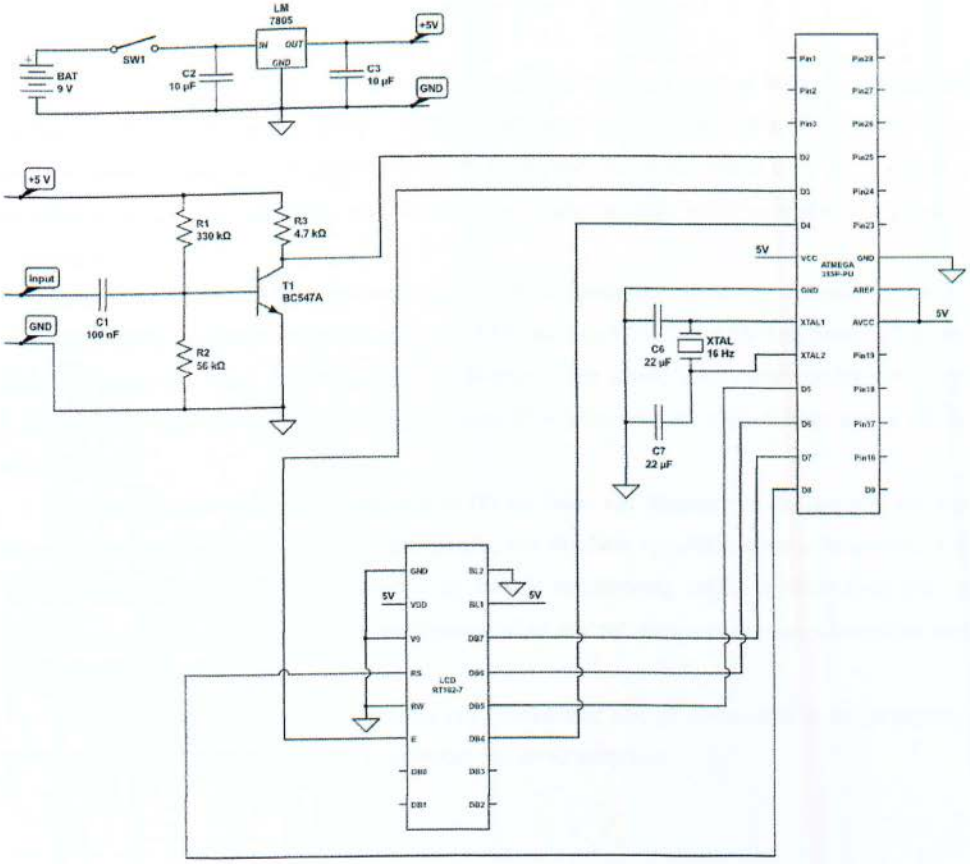
3. Οθόνη υγρών κρυστάλλων (LCD)

Για να έχει τη δυνατότητα ο χρήστης να αλληλεπιδρά με τον μικροϋπολογιστή και με την κατασκευή γενικότερα, χρησιμοποιήθηκε μια οθόνη υγρών κρυστάλλων. Πρόκειται για το μοντέλο RT162-7 με 16 ακροδέκτες. Στον παρακάτω πίνακα παρουσιάζονται οι ακροδέκτες αναλυτικά.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Gnd	Vdd	V0	RS	RW	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	BL1	BL2

Ο ακροδέκτης 1 θα γειωθεί όπως και ο ακροδέκτης 5 η λειτουργία του οποίου δεν θα χρησιμοποιηθεί. Επίσης γειώνεται ο ακροδέκτης 3 ο οποίος ρυθμίζει το contrast της οθόνης καθώς τα αποτελέσματα ήταν ικανοποιητικά με αυτή την ρύθμιση και ο ακροδέκτης 16 επειδή θα χρησιμοποιηθεί η λειτουργία του οπίσθιου φωτισμού. Στα +5V θα οδηγηθεί ο ακροδέκτης 2 και 15 που ενεργοποιούν την οθόνη υγρών κρυστάλλων και τον οπίσθιο φωτισμό αντίστοιχα. Οι ακροδέκτες 4 και 6 οδηγούνται στις ψηφιακές εισόδους/εξόδους D9 και D3 αντιστοίχως απ' όπου θα χρησιμοποιηθούν από τον μικροϋπολογιστή για την ρύθμιση των παραμέτρων και οι ακροδέκτες 11-14 θα οδηγηθούν στις ψηφιακές εισόδους/εξόδους D4-D7 από τις οποίες θα γίνεται η μεταφορά των δεδομένων.

Παρακάτω δίδεται το σχήμα με το κύκλωμα όπως έχει διαμορφωθεί.



4. Push-buttons

Η επιλογή και ο χειρισμός του menu γίνεται μέσω από τέσσερα push-buttons με τα οποία ο χρήστης διαχειρίζεται το πρόγραμμα. Αυτά, παρεμβάλλονται ανάμεσα στις ψηφιακές εισόδους D10-D13 και στην τάση λειτουργίας. Όταν πατηθεί ένα push-button, 5V διαρρέουν το κύκλωμα μεταφέροντας έτσι την πληροφορία της επιλογής του menu. Για κάθε push-button φυσικά υπάρχει και μια αντίσταση των 10kΩ η οποία λειτουργεί ως pull-up resistor, στέλνοντας λογικά "0" στις ψηφιακές εισόδους του μικροϋπολογιστή.

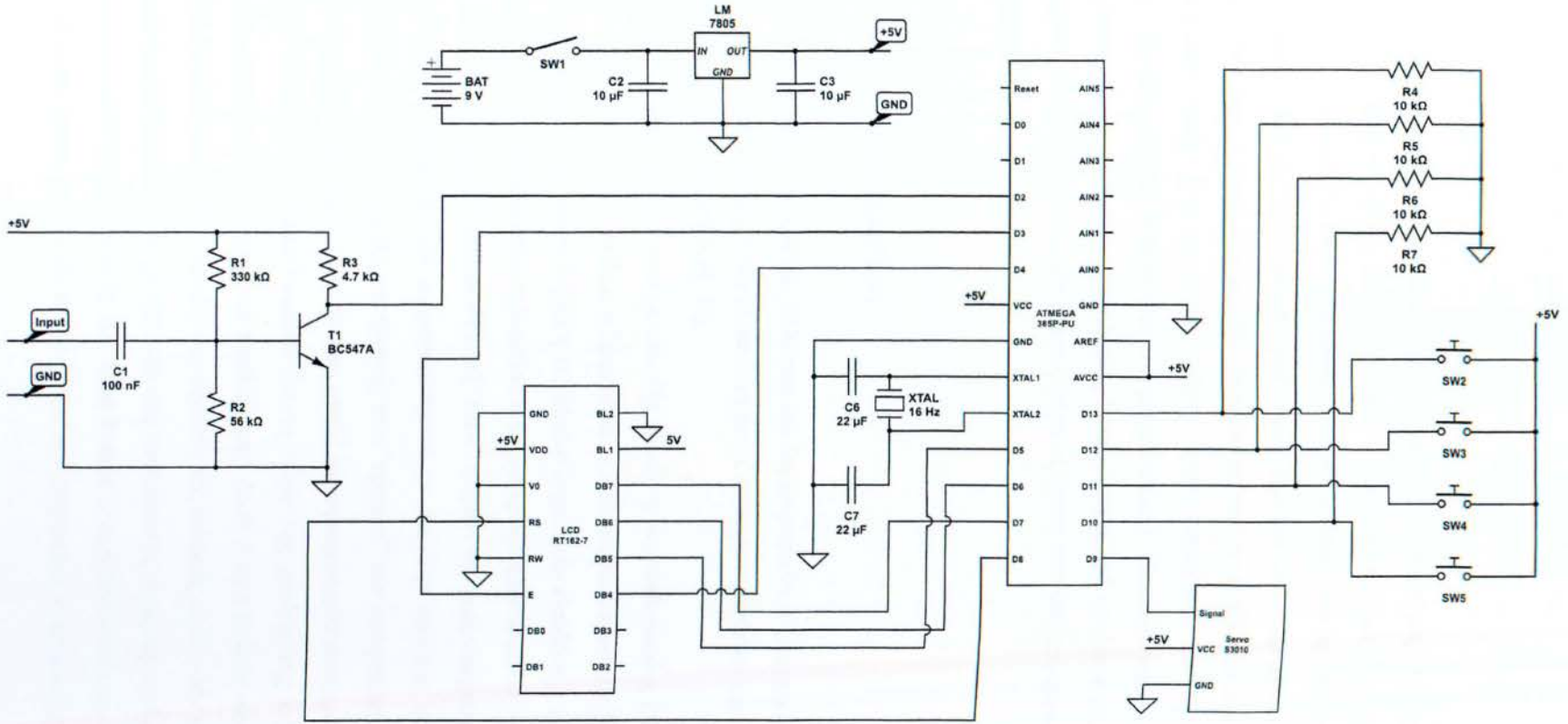
5. Κινητήρας Servo

Ο κινητήρας servo είναι το τελικό στάδιο της υλοποίησης και σε αυτό πραγματοποιείται το αυτόματο χόρδισμα της χορδής. Ο servo έχει τρεις ακροδέκτες: την τροφοδοσία, την γείωση και το σήμα το οποίο δέχεται από τον μικροϋπολογιστή. Ο μικροϋπολογιστής στέλνει στον servo παλμούς ανά δευτερόλεπτο και αναλόγως την ταχύτητα των παλμών περιστρέφεται αριστερόστροφα ή δεξιόστροφα.

Στην κατασκευή αυτή χρησιμοποιήθηκε ο servo κινητήρας s3010 της εταιρίας futaba. Ο κινητήρας αυτός έχει βαθμίδες λειτουργίας στα 4,8V και στα 6,0V. Στα 4,8V έχει ροπή 5,2kgr-cm (δηλαδή "σηκώνει" 5,2kgr αν το μπράτσο του βραχίονα έχει μήκος 1cm) και στα 6,0V έχει ροπή 6,5kgr-cm. Επίσης, ο χρόνος περιστροφής σε γωνία 60 μοιρών στα 4,8V είναι 0,20sec και στα 6,0V είναι 0,16sec.

Ο κινητήρας συνδέεται στον ακροδέκτη D9 απ' όπου και δέχεται την πληροφορία για την περιστροφή του. Για τις ανάγκες της περιστροφής του κλειδιού της κιθάρας κατασκευάστηκε και τοποθετήθηκε στον βραχίονα του servo ένας ειδικός αντάπτορας, ειδικά σχεδιασμένος για τα κλειδιά της κιθάρας, όχι μόνο της συγκεκριμένης αλλά και της πλειοψηφίας των ηλεκτρικών και ηλεκτροακουστικών κιθάρων.

Παρακάτω δίνεται το τελικό κύκλωμα της κατασκευής από το πρώτο στάδιο της ενίσχυσης του σήματος μέχρι και το τελικό, της προσθήκης του servo κινητήρα.


LM
7805
IN
OUT
GND
+5V
GND
SW1
BAT
9 V
C2
10 µF
C3
10 µF
+5V
R1
330 kΩ
R3
4.7 kΩ
Input
GND
C1
100 nF
R2
56 kΩ
T1
BC547A
Reset
D0
D1
D2
D3
D4
VCC
GND
XTAL1
XTAL2
D5
D6
D7
D8
AIN5
AIN4
AIN3
AIN2
AIN1
AIN0
AREF
AVCC
D13
D12
D11
D10
D9
ATMEGA
328P-PU
C6
22 µF
XTAL
16 Hz
C7
22 µF
R4
10 kΩ
R5
10 kΩ
R6
10 kΩ
R7
10 kΩ
SW2
SW3
SW4
SW5
GND
VDD
V0
RS
RW
E
DB0
DB1
BL2
BL1
DB7
DB6
DB5
DB4
DB3
DB2
LCD
RT162-7
5V
Signal
VCC
GND
Servo
S3010

B. Software

Σε αυτή την ενότητα θα παρουσιαστεί ο τομέας του software που χρησιμοποιήθηκε για τον προγραμματισμό του ATMEGA328P-PU και η γενικότερη μεθοδολογία για την σύγκριση του σήματος της χορδής με την συχνότητα που επιλέχθηκε από τον χρήστη, για τον προγραμματισμό του menu της οθόνης υγρών κρυστάλλων και τον χειρισμό του servo κινητήρα. Ο προγραμματισμός έγινε στην γλώσσα του arduino (η γλώσσα του αναπτυξιακού συστήματος) η οποία είναι βασισμένη στην C++ η οποία ανήκει στην οικογένεια των γλωσσών με αντικειμενοστραφή προγραμματισμό (object-oriented programming). Θα παρουσιαστούν οι μόνο βασικοί πυλώνες στους οποίους στηρίχθηκε ο κώδικας διότι η λεπτομερή ανάλυση του κώδικα θα γίνει στο κεφάλαιο IV, όπου θα γίνει μια βήμα-προς-βήμα εξήγηση των συγκεκριμένων σημείων και επιλογών που έγιναν.

1. Υπολογισμός και σύγκριση συχνότητας

Όπως έχει ειπωθεί, η χορδή της κιθάρας πάλλεται και δημιουργεί ένα ηλεκτρικό σήμα μικρής τάσης αλλά συγκεκριμένης συχνότητας. Στην υλοποίηση αυτή, το σήμα ενισχύεται και μεταφέρεται στην ψηφιακή είσοδο του ATMEGA328P-PU D2.

Ο υπολογισμός της συχνότητας γίνεται με τον εξής τρόπο: η ψηφιακή είσοδος D2 λειτουργεί ως interrupt pin αν ενεργοποιηθεί στον κώδικα η λειτουργία των interrupts και τεθεί η D2 ως πόρτα που θα δέχεται τα interrupts. Όταν λοιπόν δεχθεί η D2 κάποιο σήμα στην είσοδό της, παύεται προς στιγμή η ροή του κώδικα και ενεργοποιείται η function που έχει προγραμματιστεί να "τρέξει" στην περίπτωση αυτή. Σε αυτή την function, μια μεταβλητή "time" παίρνει την τιμή του χρόνου που έχει περάσει από την έναρξη λειτουργίας του μικροϋπολογιστή σε ms. Την επόμενη φορά που θα ενεργοποιηθεί η function αυτή, η τιμή της μεταβλητής που "κρατάει" τον προηγούμενο χρόνο θα αφαιρεθεί από την τρέχουσα τιμή και η απόλυτη τιμή αυτής της αφαίρεσης θα αποθηκευτεί σε μία νέα μεταβλητή "T". Θα πρέπει να υπενθυμιστεί ότι το σήμα της χορδής της κιθάρας είναι ημιτονοειδές και περιοδικά παίρνει χαμηλές και υψηλές τιμές. Αυτή η τιμή δηλαδή που κρατάει η μεταβλητή "T" δεν είναι άλλη από την περίοδο του σήματος της κιθάρας σε ms. Με αυτόν τρόπο υπολογίζεται η περίοδος ταλάντωσης της χορδής που στη συνέχεια της συνάρτησης διαιρείται δια 1.000.000 για να υπολογιστεί η συχνότητα σε Hz. Αυτή λοιπόν η τιμή αποθηκεύεται σε μια float μεταβλητή "freq" η οποία πλέον μπορεί να συγκριθεί με την συχνότητα-νότα που έχει επιλεχθεί από τον χρήστη.

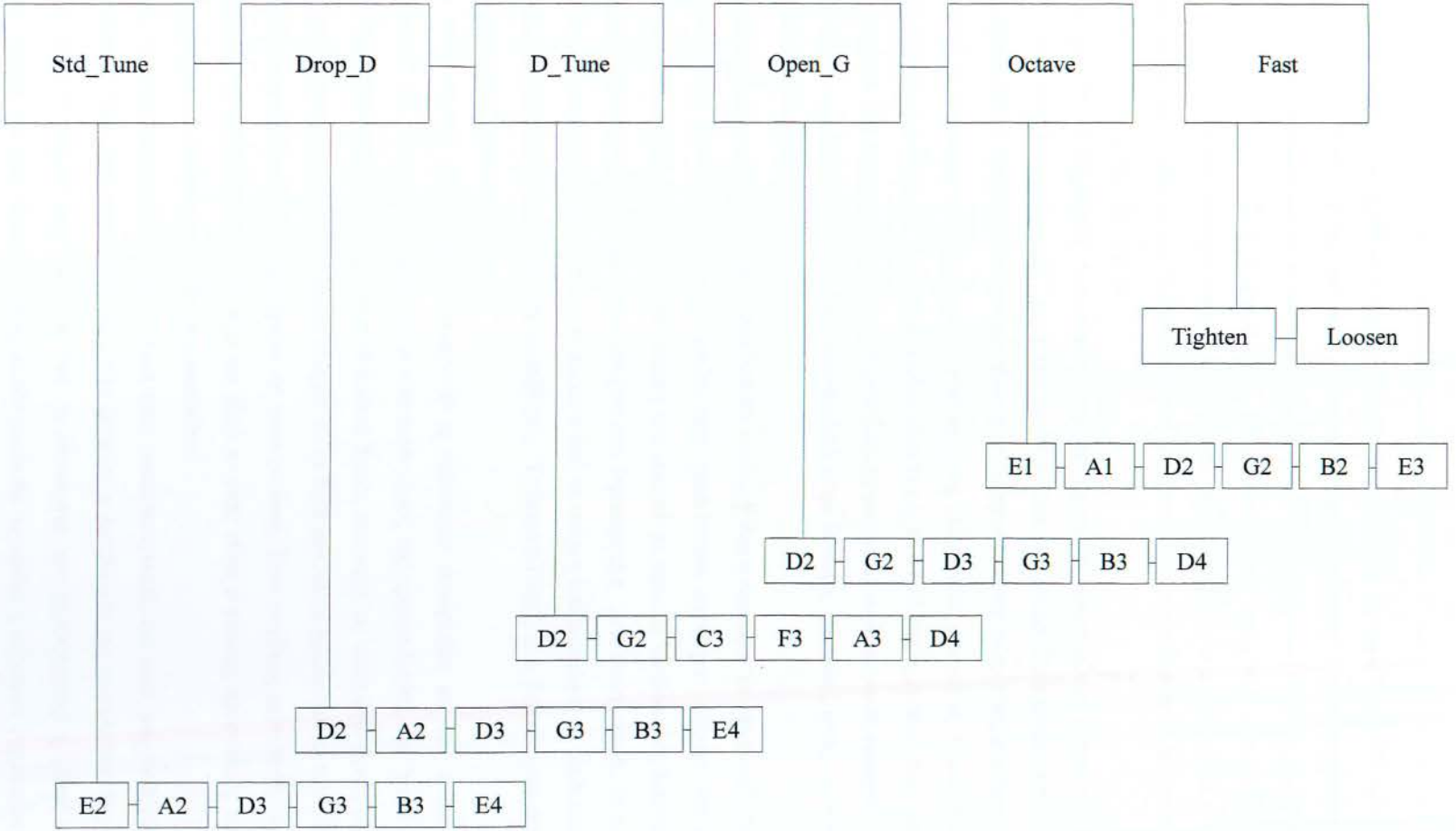
Η σύγκριση της τρέχουσας συχνότητας με την επιθυμητή πραγματοποιείται με ένα if statement. Αν η τρέχουσα συχνότητα βρίσκεται στην επιθυμητή τιμή τότε μέσω της οθόνης υγρών κρυστάλλων γνωστοποιείται στον χρήστη ότι η χορδή έχει την σωστή συχνότητα και το πρόγραμμα διακόπτει την διαδικασία της σύγκρισης και είναι έτοιμο να δεχθεί νέες εντολές. Θα πρέπει να διευκρινιστεί ότι η χορδή δεν πρέπει να εκλαμβάνεται ως ψηφιακή γεννήτρια ήχων. Τουναντίον, το σήμα της κιθάρας αν και ημιτονοειδές, δεν είναι σταθερό και η διακύμανση είναι μεγάλη, ως αναλογικό σύστημα που είναι. Άρα η επιθυμητή τιμή είναι ένα εύρος συχνοτήτων με περιθώριο λάθους της τάξεως του 0,9%, ή αλλιώς, με ακρίβεια 99,1%. Αυτό πρέπει να θεωρηθεί φυσιολογικό διότι περιορίζεται από την φύση αυτής καθ' εαυτής της χορδής ως αναλογικό σύστημα.

2. Προγραμματισμός menu και οθόνης υγρών κρυστάλλων

Ο προγραμματισμός του menu της κατασκευής και ο προγραμματισμός της οθόνης υγρών κρυστάλλων είναι οι δυο όψεις του ίδιου νομίσματος. Δεν θα μπορούσε να υπάρξει menu χωρίς κάποια οθόνη να μπορεί να προβάλει το αντικείμενο που επιλέχθηκε, και μια οθόνη χωρίς menu δεν θα είχε σημαντικό λόγο ύπαρξης σε ένα τέτοιο κύκλωμα καθώς θα μπορούσε εύκολα να αντικατασταθεί από κάποιες φωτεινές ενδείξεις (LED) ή από κάποια οθόνη επτά τμημάτων (seven segment display) .

Το menu είναι δύο διαστάσεων και χειρίζεται με τα τέσσερα κουμπιά left, right, up (esc), down (enter). Στην αρχή του προγράμματος, ορίζονται οι ονομασίες των αντικειμένων (items) του menu και στη συνέχεια, στην συνάρτηση setup ορίζεται η σχέση που έχει το ένα item με το άλλο, δηλαδή αν ένα item είναι δεξιά από ένα άλλο ή αριστερά ή είναι κάτω ή από πάνω. Για παράδειγμα, κάτω από το item "Std_Tune" που είναι το υπομενού για το χόρδισμα της κιθάρας στο συνηθισμένο κούρδισμα, βρίσκονται άλλα έξι items το ένα δίπλα από το άλλο, ένα για κάθε χορδή δηλαδή ένα για κάθε συχνότητα/νότα του συγκεκριμένου χορδίσματος. Η περιήγηση σε αυτά γίνεται με τα κουμπιά δεξιά ή αριστερά. Στην περίπτωση που ο χρήστης πιέσει το κουμπί down (enter), το πρόγραμμα ελέγχει αν υπάρχει κάποιο άλλο υπομενού/item για να προχωρήσει σε αυτό και αν δεν υπάρχει, το πρόγραμμα μεταφέρεται στην function που αναγνωρίζει ποιο item επιλέχθηκε και προβαίνει στην ενέργεια που έχει προγραμματιστεί.

Το παρακάτω σχέδιο παρουσιάζει σχηματικά την μορφή του menu με όλα τα items που χρησιμοποιήθηκαν. Αποτελείται από πέντε διαφορετικά χορδίσματα και μια επιπλέον επιλογή για την γρήγορη αντικατάσταση μιας χορδής με το servo κινητήρα να περιστρέφεται με τη μέγιστη ταχύτητά του.


Std_Tune
Drop_D
D_Tune
Open_G
Octave
Fast
Tighten
Loosen
E1
A1
D2
G2
B2
E3
D2
G2
D3
G3
B3
D4
D2
G2
C3
F3
A3
D4
D2
A2
D3
G3
B3
E4
E2
A2
D3
G3
B3
E4

Στο πάνω μέρος του σχεδίου είναι οι ονομασίες των items των χορδισμάτων και των λειτουργιών, τα οποία items συνδέονται με άλλα items σε μορφή υπομενού ή καλύτερα σε σχέση item-γονέα με item-παιδιού. Πρέπει να αναφερθεί πως η κύλιση του menu είναι κυκλική, δηλαδή τα items που βρίσκονται στο δεξί ή αριστερό άκρο, συνδέονται μεταξύ τους, κάτι που κάνει τον χειρισμό πιο εύκολο και πιο γρήγορο. Επίσης όλα τα items-παιδιά συνδέονται με τον γονέα και με κατακόρυφο τρόπο, έτσι ώστε να μην είναι αναγκαία η μετάβαση στο πρώτο item του υπομενού για να μεταφερθεί ο χρήστης στο item-γονέα.

Ο έλεγχος και η ενημέρωση του display της οθόνης υγρών κρυστάλλων γίνεται με μια function με ονομασία "menuchanged". Αυτή λειτουργεί με ένα block από if statements απ' όπου καθώς έχει προσωρινά συγκρατήσει σε μια μεταβλητή την πληροφορία της ονομασίας του item που βρίσκεται αυτή τη στιγμή το πρόγραμμα, το συγκρίνει με τις υπάρχουσες ονομασίες. Όταν κάποιο if statement εμφανίσει λογικό "1" σε συγκεκριμένο σημείο της LCD οθόνης εμφανίζει την πλήρη ονομασία του item. Για λόγους αισθητικής αλλά και λειτουργικότητας τα item-γονέα εμφανίζονται στην πρώτη σειρά της LCD οθόνης και τα item-παιδιού στην δεύτερη, απεικονίζοντας με αυτόν τον τρόπο την δισδιάστατη δομή του menu.

Τη λειτουργία της ανάγνωσης των push-buttons αναλαμβάνει η function "readbuttons". Αυτή η συνάρτηση εκτός από απλά να αναγνωρίζει ποιο push-button πατήθηκε, ελέγχει και την περίπτωση του debouncing δηλαδή την περίπτωση που μπορεί να υπάρξει λανθασμένη διάγνωση για το πότε πατήθηκε το push-button λόγω θορύβου που δημιουργεί η μεταλλική επαφή. Σε αυτό το σημείο λοιπόν μετράται ο χρόνος που έχει περάσει από τη στιγμή που πατήθηκε το push-button και αν αυτός ο χρόνος υπερβεί την τιμή της μεταβλητής "DebounceTime", τότε βεβαιώνεται ότι το συγκεκριμένο κουμπί όντως πατήθηκε.

Την προηγούμενη συνάρτηση συμπληρώνει η παρακάτω συνάρτηση με την ονομασία "navigateMenus". Η συγκεκριμένη είναι το εκτελεστικό μέρος της προηγούμενης και "κινεί" το menu προς την κατεύθυνση που ορίζει το push-button. Εκτός από αυτή τη λειτουργία όμως, αν το κουμπί που πατήθηκε είναι το down (enter) ελέγχει αν το item στο οποίο βρισκόταν το menu έχει item-παιδί. Αν ναι, τότε το menu μεταφέρεται σε αυτό το item. Στην αντίθετη περίπτωση (όπως είπαμε και πριν) το πρόγραμμα αναγνωρίζει ότι αυτό το item είναι ο τελικός προορισμός και η επιλογή του χρήστη και καλείται η συνάρτηση menuUsed.

Τέλος, η συνάρτηση menuUsed είναι αυτή στην οποία αναγνωρίζεται ποιο item επιλέχθηκε και είτε γίνεται η σύγκριση της συχνότητας, είτε ξεκινάει η λειτουργία της αντικατάστασης της χορδής. Σε αυτό το σημείο (αν πρόκειται για τη λειτουργία του χόρδισματος) η μεταβλητή "right_freq" παίρνει την τιμή της συχνότητας με την οποία θα συγκριθεί η τρέχουσα συχνότητα και υπολογίζεται το εύρος των τιμών που θα θεωρείται ως σωστό και αποδεκτό. Στη συνέχεια με

κάποια if statements αποκόπτονται από την μέτρηση οι αρμονικές συχνότητες που παράγονται από την τρέχουσα συχνότητα και πραγματοποιείται η μέτρηση, υπολογίζεται το error το οποίο και τροφοδοτείται ως είσοδος στον ακροδέκτη signal του servo κινητήρα.

3. Χειρισμός κινητήρα servo

Όπως ειπώθηκε προηγουμένως, ο servo έχει τρεις ακροδέκτες: της τάσης, της γείωσης και του σήματος. Τα δεδομένα που αποστέλλονται από από την ψηφιακή έξοδο D9 του ATMEGA328P-PU στον τρίτο ακροδέκτη του servo είναι αυτά που θα απασχολήσουν την ενότητα αυτή.

Στον ακροδέκτη του σήματος αποστέλλεται παλμός κάποιων ms από τον μικροϋπολογιστή για να καθορίσει την περιστροφή του. Παλμός 1500us ακινητοποιεί τον servo κινητήρα, ενώ τιμές μεγαλύτερες από αυτή τον περιστρέφουν αριστερά και τιμές μικρότερες αυτής, δεξιά. Όσο πιο κοντά στο κέντρο των 1500us είναι οι παλμοί που στέλνονται, τόσο πιο αργή είναι η περιστροφή. Πλήρης ταχύτητα επιτυγχάνεται στις τιμές 1000us και 2000us, 500us δηλαδή μακριά από το κέντρο ισορροπίας.

Στον servo λοιπόν, στην συνάρτηση "menuUsed" στέλνεται στον servo κινητήρα παλμό ίσο με το παλμό ισορροπίας, μειωμένο όμως με την τιμή του σφάλματος "error". Η τιμή του error υπολογίζεται όταν αφαιρέσουμε από την σωστή τιμή της συχνότητας που έχει επιλεχθεί την τρέχουσα συχνότητα. Δηλαδή,

$$error = right\_freq - freq$$

όπου ***error*** είναι το σφάλμα, ***right_freq*** η σωστή τιμή της συχνότητας που έχει επιλεχθεί και ***freq*** η τρέχουσα συχνότητα. Στη συνέχεια ;όπως ειπώθηκε, τροφοδοτείται ο servo κινητήρας με παλμό ίσο με 1500us – error.

Κατά τις μετρήσεις όμως θεωρήθηκε σκόπιμο να προστεθεί ένας ελεγκτής P ο οποίος θα δώσει μεγαλύτερη ταχύτητα στην ανάδραση και τον μηδενισμό του σφάλματος. Καθώς η κίνηση του servo κινητήρα σε τιμές 1,2 ή 5us πάνω ή κάτω από τον παλμό ακινητοποίησης ήταν πολύ μικρή, η προσθήκη του Kp με τον οποίο θα πολλαπλασιάζεται το σφάλμα κρίθηκε αναγκαία. Με αυτόν τον τρόπο προκύπτει το παρακάτω κομμάτι του κώδικα:

error=right_freq-super_freq;

myservo.writeMicroseconds(1500 – error*Kp);

Η επιλογή της τιμής του Kp είναι πολύ κρίσιμη καθώς μικρότερη τιμή από την σωστή θα έχει λίγο ή καθόλου επίδραση στο σύστημα ενώ μια μεγαλύτερη τιμή θα προκαλούσε αστάθεια με τον servo κινητήρα να περιστρέφεται ατέρμονα δεξιόστροφα και αριστερόστροφα, προσπαθώντας να μηδενίσει το σφάλμα.

Τέλος, προστίθεται ένα μικρό delay για να δώσει τον κατάλληλο χρόνο στον servo να καλύψει το σφάλμα της μέτρησης. Πάλι η τιμή της καθυστέρησης πρέπει να έχει σωστή τιμή έτσι ώστε η κίνηση του servo να μην διαρκέσει περισσότερο από το επιθυμητό που μπορεί ακόμα και να σπάσει την χορδή, αλλά και ούτε να διαρκέσει λιγότερο που κάτι τέτοιο θα έφερνε αναποτελεσματικότητα στην σωστή χόρδιση της χορδής.

IV. Προβλήματα – Λύσεις

A. Hardware

1. Τροποποίηση servo κινητήρα για πλήρη περιστροφή

Ο servo κινητήρας για να χορδίσει σωστά μια χορδή, πρέπει να έχει τη δυνατότητα περιστροφής 360^0. Οι περισσότεροι όμως έχουν διαθέτουν ένα stop στα γρανάζια τους όπως επίσης και ένα ποτενσιόμετρο το οποίο συνδέεται μηχανικά με τα γρανάζια και περιστρέφεται ταυτόχρονα με τον βραχίονα του servo κινητήρα. Με αυτόν τον τρόπο μπορεί να υπολογίσει τις μοίρες που βρίσκεται ανά πάσα στιγμή ο βραχίονας σε σχέση με το κέντρο. Επειδή συνήθως οι servo κινητήρες χρησιμοποιούνται σε κατασκευές μοντελισμού όπως τηλεκατευθυνόμενα αυτοκίνητα, πλοία και αεροπλανάκια, ένα φυσικό "stop" είναι πολύ χρήσιμο. Για παράδειγμα για την περιστροφή του άξονα των μπροστινών τροχών ενός τηλεκατευθυνόμενου αυτοκινήτου ένα stop είναι πολύ χρήσιμο, αλλά στην προκειμένη περίπτωση, εμποδίζει την σωστή χόρδιση.

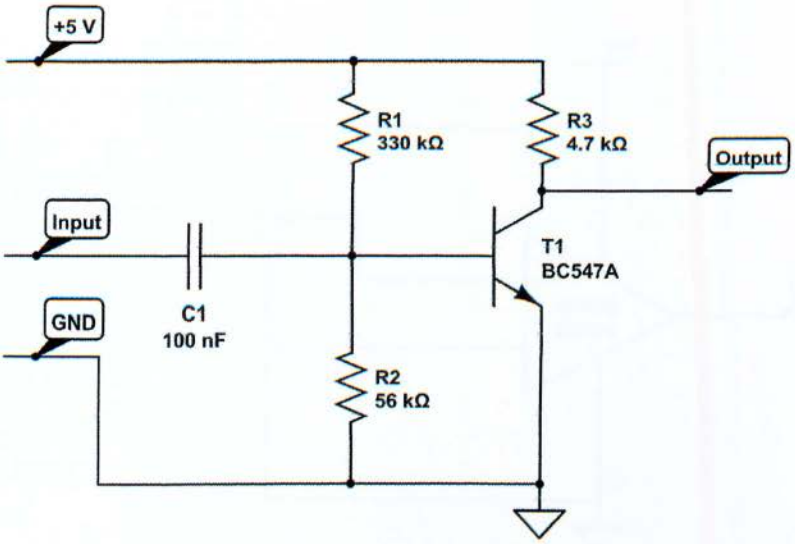
Για την κατασκευή αυτή, όπως έχει ειπωθεί και στα προηγούμενα κεφάλαια χρησιμοποιήθηκε ο servo κινητήρας της εταιρίας Futaba, μοντέλο S3010. Για την τροποποίηση χρειάστηκε να ανοιχθεί ο κινητήρας και να πριονιστεί το μέρος του γραναζιού που λειτουργούσε ως stop. Επίσης, αφαιρέθηκε και το σημείο της μηχανικής σύνδεσης του γραναζιού με το ποτενσιόμετρο έτσι με την οποιαδήποτε κίνηση του γραναζιού δεν περιστρέφεται και το ποτενσιόμετρο και στην ουσία το ακυρώνει.

Σε σχέση με άλλες μεθόδους τροποποίησης των servo κινητήρων, αυτή θεωρήθηκε πιο εύκολη γιατί δεν χρειάστηκε να αφαιρεθεί τελείως το ποτενσιόμετρο, αλλά να μην περιστρέφεται μαζί με το γρανάζι του βραχίονα. Στη συνέχεια στάλθηκε από τον μικροϋπολογιστή παλμός 1500us, ο οποίος θεωρητικά ακινητοποιεί τον βραχίονα και μέσω αυτής της τιμής ρυθμίστηκε το ποτενσιόμετρο με το χέρι για να μην περιστρέφεται. Δηλαδή, ορίστηκε ο παλμός 1500us, ως παλμός ακινητοποίησης περιστρέφοντας το ποτενσιόμετρο του servo κινητήρα μέχρις ότου το μοτέρ να παραμείνει ακίνητο.

2. Ενίσχυση σήματος κιθάρας

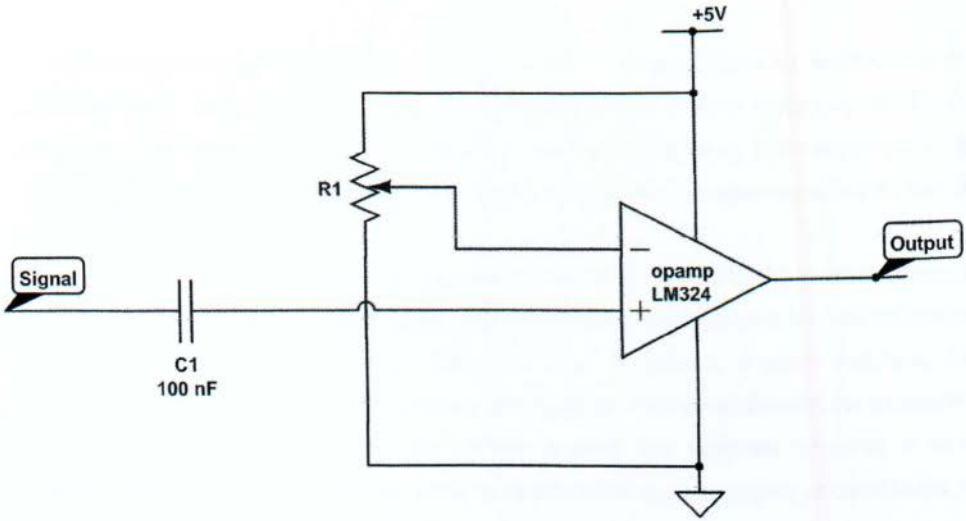
Για την ενίσχυση του σήματος της κιθάρας επιλέχθηκε ο transistor BC547 ο οποίος

τροφοδοτείται από το σήμα της κιθάρας μέσω ενός σωστού biasing κυκλώματος και στη συνέχεια το σήμα στέλνεται ενισχυμένο στην είσοδο D2 του ATMEGA328P-PU. Όμως, μαζί με το σήμα, ενισχύονταν και οι αρμονικές του σήματος και δυσκόλευαν τη μέτρηση και τη σύγκριση της συχνότητας. Για να απομονωθούν οι αρμονικές συχνότητες χρησιμοποιούνται 3 if statements και 3 float μεταβλητές που αν μπορούσαν να αποφευχθούν θα έκαναν τον κώδικα πιο ελαφρύ και πιο γρήγορο.



Έτσι, δοκιμάστηκε η λύση του συγκριτή τάσης. Οι αρμονικές συχνότητες έχουν από τη φύση τους μικρότερο πλάτος Α από την κυρίως συχνότητα. Αν υπήρχε η δυνατότητα θέσπισης ενός ορίου ως προς το πλάτος του σήματος, και μόνο οι δυνατότερες συχνότητες να ενισχύονταν θα έλυνε το πρόβλημα των αρμονικών. Σε αυτό το σκεπτικό λοιπόν, υλοποιήθηκε το παρακάτω σχήμα. Το σήμα της κιθάρας περνάει πρώτα από έναν πυκνωτή 100nF και στη συνέχεια οδηγείται στην είσοδο + του τελεστικού ενισχυτή LM324 που λειτουργεί ως συγκριτής τάσης. Η δεύτερη είσοδος του συγκριτή είναι συνδεδεμένη με την τάση μέσω ενός ποτενσιόμετρου. Η διαδικασία της ρύθμισης του ποτενσιόμετρου είναι η εξής: ενώ το κύκλωμα είναι συνδεδεμένο κατά αυτόν τον τρόπο, προγραμματίζεται ο μικροϋπολογιστής να δείξει την τρέχουσα συχνότητα μέσω της σειριακής θύρας. Καθώς λοιπόν παίζεται η χορδή και εμφανίζονται οι ενδείξεις, ρυθμίζεται η αντίσταση του ποτενσιόμετρου σε real-time. Όσο μειώνεται η αντίσταση του ποτενσιόμετρου, τόσο περισσότερο

ρεύμα θα εισέρχεται στην είσοδο "-" του συγκριτή και τόσο λιγότερες αρμονικές συχνότητες θα περνάνε μέσα από τον συγκριτή. Η διαδικασία ολοκληρώνεται όταν στην έξοδο του συγκριτή περνάει μόνο η θεμελιώδης συχνότητα η οποία και είναι ενισχυμένη στον κόρο που είναι τα +5V. Αυτό δεν αποτελεί πρόβλημα βέβαια διότι ο μικρουπολογιστής υπολογίζει τη μόνο συχνότητα του σήματος, είτε αυτό είναι ημιτονικό σήμα είτε είναι τετραγωνικός παλμός.



Παρ' όλα τα σωστά αποτελέσματα και τη θετική επιρροή που είχε στον κώδικα αυτή η υλοποίηση δεν πραγματοποιήθηκε για ένα μόνο λόγο: το πλήθος των αποτελεσμάτων. Καθώς μειωνόταν το πλάτος Α του σήματος, η τάση στον ακροδέκτη "-" του συγκριτή ήταν μεγαλύτερη από την τάση του σήματος και η έξοδος του LM324 παρουσίαζε λογικό "1", ανεξάρτητα φυσικά αν ήταν η θεμελιώδης συχνότητα που είχε εξασθενήσει με τον χρόνο ή κάποια αρμονική της. Στην υλοποίηση ενίσχυσης του σήματος μέσω του transistor BC547A υπήρχαν πολλές αρμονικές συχνότητες μαζί με την θεμελιώδη μεν, αλλά το πλήθος και η διάρκεια των μετρήσεων ήταν πολύ μεγαλύτερες. Με την ενίσχυση μέσω transistor, ο μικροϋπολογιστής δέχονταν δεδομένα μέχρι και 6 δευτερόλεπτα μετά το παίξιμο της χορδής. Στην αντίστοιχη του συγκριτή η διάρκεια των αποτελεσμάτων ήταν μικρότερη από ένα δευτερόλεπτο.

Για αυτό το λόγο, αποφασίστηκε η ενίσχυση του σήματος μέσω του transistor BC547A. Θεωρήθηκε καλύτερο να υπάρχει μεγάλος όγκος δεδομένων τον οποίο βέβαια με τη χρήση if statements να διαχειριστούν, παρά λίγα δεδομένα, αν και σωστά. Τα περισσότερα δεδομένα δίνουν μεγαλύτερη ροή κατά τη διαδικασία της σύγκρισης της συχνότητας και τον μηδενισμό του σφάλματος από τον servo κινητήρα ενώ τα λιγότερα δεδομένα θα επέβαλαν το συνεχές παίξιμο της χορδής από τον χρήστη μέχρις ότου μηδενιστεί το σφάλμα.

3. Ροπή servo κινητήρα

Για να περιστρέψει το κλειδί της κιθάρας ο servo κινητήρας πρέπει να διαθέτει και την κατάλληλη ροπή. Στην αρχή της κατασκευής χρησιμοποιήθηκε ο servo κινητήρας της Futaba S3003 με ροπή 3,2kgr*cm στα 4,8V και 4,1kgr*cm στα 6V (στην φυσική, η δύναμη μετριέται σε Newton, αλλά στον χώρο των servo κινητήρων έχει αντικατασταθεί για πρακτικούς λόγους από τα kgr, δηλαδή πόσα κιλά "σηκώνει").

Ενώ οι μετρήσεις και οι δοκιμές πραγματοποιούνταν με το Futaba S3003 το οποίο φαινόταν να περιστρέφει τα κλειδιά της κιθάρας χωρίς κάποια δυσκολία, παρατηρήθηκε ότι όταν καλούνταν να κάνει μια μικρή περιστροφή της τάξεως των 5 με 10 μοιρών, ο servo κινητήρας δεν ανταποκρινόταν, αν και ακουγόταν ότι το μοτέρ λειτουργούσε. Για να επιβεβαιωθεί ότι πραγματικά υπήρχε κίνηση στον κινητήρα αλλά δεν αρκούσε η ροπή του, αφηνόταν στιγμιαία ο servo κινητήρας πάνω στο κλειδί της κιθάρας χωρίς να συγκρατείται από τον χρήστη με αποτέλεσμα να περιστρέφεται γύρω από τον εαυτό του.

Για αυτό το λόγο ο servo κινητήρας αντικαταστάθηκε από δυνατότερο της ίδιας εταιρίας. Πρόκειται για τον Futaba S3010 με ροπή 5,2kgr*cm στα 4,8V και 6,5kgr*cm στα 6V. Εκτός από την υπεροχή του στη ροπή, ο S3010 είναι κατά $0,03sec/60^0$ πιο γρήγορος από τον S3003 αλλά και 4 γραμμάρια πιο βαρύς, γεγονός όμως που δεν απασχολεί σημαντικά την συγκεκριμένη κατασκευή.

B. Software

1. Απομόνωση αρμονικών συχνοτήτων

Το σήμα της κιθάρας, μαζί με την κύρια συχνότητα της χορδής όταν πάλλεται φέρει και τις

αρμονικές της οι οποίες δυσκολεύουν τη διαδικασία της μέτρησης και της σύγκρισης. Εφ' όσον δεν κατέστη δυνατόν να απομονωθούν οι αρμονικές στο μέρος του hardware, η λύση πρέπει να προέλθει από το μέρος του software και συγκεκριμένα στον κώδικα.

Η διαδικασία της απομόνωσης των αρμονικών δεν έγινε σε ένα αυστηρώς θεωρητικό πλαίσιο περί πλάτους και εμφάνισης των αρμονικών συχνοτήτων αλλά σε συνδυασμό με το συγκεκριμένο σύστημα στο οποίο εφαρμόζεται αυτό που είναι η χορδή της κιθάρας. Για παράδειγμα, η έκτη χορδή της κιθάρας E2 που πάλλεται με συχνότητα 82,41 Hz εμφανίζει την δεύτερη αρμονική συχνότητά της, η οποία έχει και το μεγαλύτερο πλάτος από τις αρμονικές, στα 164,81 Hz. Δεν θα υπήρχε λόγος να απομονωθεί μόνο το συγκεκριμένο εύρος των συχνοτήτων από 163 Hz έως 165 Hz διότι η έκτη χορδή από την φύση της μπορεί να τεντωθεί μέχρι τέσσερις νότες πάνω από το συνηθισμένο χόρδισμα, δηλαδή μέχρι τα 110 Hz. Μετά από αυτή τη συχνότητα, η χορδή συνήθως σπάει.

Συνολικά απομονώνονται τρεις ομάδες αρμονικών συχνοτήτων με τρία if statements. Η πρώτη ομάδα απομονώνει τις αρμονικές που εμφανίζονται τέσσερις νότες πάνω από τη συχνότητα που πάλλεται. Για παράδειγμα αν χορδίζεται η έκτη χορδή στην νότα E2 με συχνότητα 82,41 Hz, με ένα if statement πρέπει να απομονωθούν συχνότητες κοντά στη νότα A2 με συχνότητα. Έτσι, ο κώδικας διαμορφώνεται ως εξής:

```
right_freq = 82,41;
harm4_plus = right_freq * 4/3 + 2;
harm4_minus = right_freq * 4/3 - 2;

if (now_freq > harm4_plus || now_freq < harm4_minus) {...}
```

όπου ***right_freq*** είναι η συχνότητα που επιλέχθηκε από τον χρήστη (σε αυτή την περίπτωση είναι η νότα E2 στα 82,41Hz), ***harm4_plus*** είναι το πάνω όριο του φίλτρου, ***harm4_minus*** το κάτω όριο του φίλτρου και ***now_freq*** η τρέχουσα συχνότητα.

Παρατηρείται ότι αν η τρέχουσα συχνότητα βρίσκεται στο εύρος αυτό των συχνοτήτων τότε θα απομονωθεί και δεν θα συμπεριληφθεί στην διαδικασία της σύγκρισής της με την επιθυμητή συχνότητα. Πρέπει να υπενθυμιστεί ότι η απόσταση μιας νότας από μία άλλη εκφράζεται λογαριθμικά. Ο αριθμός δε 4/3 δεν είναι άλλος από τον λόγο της τέταρτης νότας ως προς την

πρώτη, σε αυτή την περίπτωση της νότας A2 ως προς την νότα E2, δηλαδή:

E2 x 4/3 = 82,4069 x 4/3 ≈ 110 Hz = A2

όπως όμως ειπώθηκε προηγουμένως, η έκτη χορδή, όπως και κάθε χορδή δεν μπορεί να χορδιστεί πάνω από μία συγκεκριμένη συχνότητα η οποία εμπειρικά βρίσκεται τέσσερις νότες πάνω από το συνηθισμένο χόρδισμα της χορδής. Άρα θα μπορούσε το if statement να τροποποιηθεί και να μην περιέχει το πάνω όριο του φίλτρου αλλά μόνο το κάτω διότι η χορδή από την φύση της δεν μπορεί να ξεπεράσει αυτή τη συχνότητα. Έτσι ο κώδικας παίρνει την παρακάτω μορφή,

```
right_freq = 82,41;
harm4_minus = right_freq * 4/3 - 2;

if (now_freq < harm4_minus ){ ... }
```

η οποία εκτός από πιο απλή και πιο γρήγορη, εκφράζει και την φυσική λειτουργία και αντοχή της χορδής.

Η δεύτερη ομάδα αρμονικών που απομονώνονται είναι οι συχνότητες βρίσκονται τρεις νότες πάνω από την επιλεγμένη συχνότητα η οποία βρίσκεται στα φυσικά όρια αντοχής της χορδής. Ο λόγος της τρίτης νότας ως προς την πρώτη είναι 5/4, για παράδειγμα αν ο χρήστης επιλέξει να χορδίσει την πέμπτη χορδή στη νότα A2 που βρίσκεται στα 110 Hz, το if statement θα απομονώσει συχνότητες που βρίσκονται κοντά στη συχνότητα 110*5/4 =137,5 Hz και είναι κοντά στην νότα C#3. Έτσι, ο κώδικας διαμορφώνεται ως εξής:

```
right_freq = 110,00;
harm3_plus = right_freq * 5/4 + 2;
harm3_minus = right_freq * 5/4 - 2;

if (now_freq > harm3_plus || now_freq < harm3_minus) { ... }
```

όπου ***right_freq*** είναι η συχνότητα που επιλέχθηκε από τον χρήστη (σε αυτή την περίπτωση είναι η νότα A2 στα 110,00 Hz), ***harm3_plus*** είναι το πάνω όριο του φίλτρου, ***harm3_minus*** το κάτω όριο του φίλτρου και ***now_freq*** η τρέχουσα συχνότητα.

Η τρίτη και τελευταία ομάδα αρμονικών συχνοτήτων είναι συχνότητες που εμφανίζονται πέντε νότες κάτω από την επιλεγμένη συχνότητα. Όπως και στην πρώτη κατηγορία συχνοτήτων, δεν θα απομονωθεί μόνο ένα μικρό εύρος συχνοτήτων αλλά όλες οι συχνότητες κάτω από αυτή την νότα. Στην ουσία είναι ένα υψιπερατό φίλτρο συχνοτήτων το οποίο έχει πολλές ομοιότητες με το πρώτο if statement. Ο λόγος για τον οποίο απομονώνονται όλες οι συχνότητες κάτω από αυτή τη συχνότητα οφείλεται και πάλι στην φύση της χορδής. Η χορδή όταν βρίσκεται 5 νότες κάτω από την νότα του συνηθισμένου χορδίσματος, είναι πολύ χαλαρή και ο ήχος είναι αδιευκρίνιστος. Όταν επίσης η χορδή είναι πολύ χαλαρή, κατά την ταλάντωσή της αγγίζει το μπράτσο της κιθάρας και ο ήχος περιπλέκεται ακόμα περισσότερο. Κρίθηκε λοιπόν καλύτερο να λειτουργεί ως υψιπερατό φίλτρο και ο κώδικας πήρε την παρακάτω μορφή:

```
right_freq = 110,00;
harm_half = right_freq / 1.5;

if (now_freq > harm_half){ ... }
```

όπου ***right_freq*** είναι η συχνότητα που επιλέχθηκε από τον χρήστη (σε αυτό το παράδειγμα είναι η νότα A2 στα 110,00 Hz), ***harm_half*** είναι η συχνότητα του υψιπερατού φίλτρου και ***now_freq*** η τρέχουσα συχνότητα.

2. Ελεγκτής P

Κατά τη διάρκεια των δοκιμών της συμπεριφοράς του συστήματος στην σύγκριση της συχνότητας και χόρδιση της χορδής παρατηρήθηκε ότι η ανάδραση μόνο με την τιμή του σφάλματος δεν ήταν ήταν αρκετή. Όταν το σφάλμα ήταν μικρό τότε η απόδοση του servo κινητήρα ήταν ανεπαρκής και αργή. Αυτό οφείλεται αφ' ενός στον τρόπο με τον οποίο δέχεται εντολές από

τον μικροϋπολογιστή και αφ' ετέρου από τον τρόπου που λειτουργεί ο ίδιος ο servo κινητήρας. Παρακάτω δίνεται ο κώδικας χωρίς την χρήση του ελεγκτή:

```
error = right_freq - now_freq;
myservo.writeMicroseconds(1500 – error);
```

Ο servo κινητήρας όπως έχει ειπωθεί, λειτουργεί με διάφορες ταχύτητες. Όταν δεχθεί παλμό 1500us ο servo κινητήρας ακινητοποιείται, με παλμό 2000us περιστρέφεται με τη μέγιστη ταχύτητά του αριστερόστροφα και με 1000us περιστρέφεται με τη μέγιστη ταχύτητά του δεξιόστροφα. Όλες οι ενδιάμεσες τιμές είναι διαβαθμίσεις στην ταχύτητά του που σημαίνει ότι με παλμό 1501us ή με παλμό 1497us θα περιστραφεί με πολύ μικρή ταχύτητα. Κατά την χόρδιση της χορδής και την ρύθμιση της συχνότητας από τον servo κινητήρα αυτό αποτελεί πρόβλημα διότι ένα σφάλμα της τάξεως των 5 Hz, που για την έκτη χορδή μεταφράζεται σε απόκλιση μίας ολόκληρης νότας, για τον κινητήρα μεταφράζεται ως 5us με αποτέλεσμα μια πολύ συγκρατημένη ταχύτητα περιστροφής. Επίσης, όταν το σφάλμα είναι μεγαλύτερης τάξεως, ο servo κινητήρας θα χρειαστεί ίσως και δεκάδες δευτερόλεπτα για να μηδενίσει το σφάλμα. Για αυτό τον λόγο η χρησιμοποίηση ενός ελεγκτή P κρίθηκε απαραίτητη.

Ένας άλλος λόγος που η ανάδραση χωρίς ελεγκτή απορρίφθηκε είναι ο τρόπος με τον οποίο ο servo κινητήρας λαμβάνει τις εντολές από τον μικροϋπολογιστή. Μια προσεκτική ματιά στον κώδικα θα αποκαλύψει την αδυναμία της εντολής ***myservo.writeMicroseconds(us);*** να δεχθεί δεκαδικούς αριθμούς. Για παράδειγμα αν η τρέχουσα συχνότητα είναι 82,82 Hz και η επιθυμητή συχνότητα είναι 82,41 Hz τότε το error είναι της τάξεως των 0,41 Hz. Για να μπορέσει να μηδενιστεί το σφάλμα αυτό θα χρειαζόταν περιστροφή του κλειδιού της κιθάρας τουλάχιστον 90^0. Ο μικροϋπολογιστής όμως θα δώσει παλμό 1499us που θα περιστρέφει το κλειδί κατά 1^0.

Για τους δύο αυτούς λόγους εισάγεται στο σύστημα ελεγκτής P ο οποίος θα μηδενίζει το σφάλμα με ταχύτητα και ακρίβεια. Ο προηγούμενος κώδικας μετατρέπεται στον παρακάτω:

```
error = right_freq - now_freq;
myservo.writeMicroseconds(1500 – Kp * error);
```

όπου ***right_freq*** είναι η συχνότητα που επιλέχθηκε από τον χρήστη, ***now_freq*** η τρέχουσα συχνότητα, ***error*** το σφάλμα της μέτρησης και ***Kp*** η παράμετρος του P ελεγκτή. Η σωστότερη μορφή του ελεγκτή θα ήταν:

```
error = right_freq – now_freq;
P = Kp * error;
myservo.writeMicroseconds(1500 – P);
```

Δεν θεωρήθηκε, όμως, σκόπιμο να χρησιμοποιήσουμε ακόμα μια float μεταβλητή για λόγους οικονομίας στη χρήση της μνήμης του μικροϋπολογιστή και στην ταχύτητα του κώδικα.

3. Σύνδεση item – συχνότητας

Στο κεφάλαιο ΙΙΙ, της μεθοδολογίας και στην ενότητα Β του software αναλύθηκε ο τρόπος χειρισμού του menu μέσω των push-buttons. Όταν λοιπόν ένα συγκεκριμένο item επιλεγόταν, το πρόγραμμα έτρεχε την συνάρτηση menuUsed στην οποία πραγματοποιείται η σύγκριση της συχνότητας και ο μηδενισμός του σφάλματος. Η συνάρτηση όμως δεν γνωρίζει την επιθυμητή συχνότητα-νότα με την οποία είναι συνδεδεμένο το συγκεκριμένο item. Η μόνη πληροφορία που μπορεί να αναγνωσθεί είναι η ονομασία του item που επιλέχθηκε για να τρέξει η συγκεκριμένη συνάρτηση.

Η πιο εύκολη λύση ήταν η δημιουργία μια σειρά από if statements η οποία θα συνέδεε το όνομα του item με την επιθυμητή συχνότητα και στη συνέχεια θα προχωρούσε στη διαδικασία της απομόνωσης των αρμονικών συχνοτήτων, έπειτα στη σύγκριση της συχνότητας και τέλος στον μηδενισμό του σφάλματος μέσω του servo κινητήρα. Αυτό θα έπρεπε να επαναληφθεί για κάθε συχνότητα που αντιπροσωπεύεται από κάποιο item. Τα item που φέρουν μοναδιαίες νότες, δηλαδή νότες που δεν είναι ίδιες με άλλα items είναι συνολικά 16. Για την απομόνωσης των αρμονικών συχνοτήτων, τη σύγκριση της συχνότητας και τον μηδενισμό του σφάλματος απαιτούνται 35 γραμμές κώδικα. Άρα συνολικά θα απαιτούνταν 16 x 35 = 560 γραμμές κώδικα χωρίς τα αρχικά if statements.

Η λύση του switch - case θα έδειχνε μια πιο "τακτοποιημένη" μορφή της παραπάνω έκδοσης

αλλά δεν θα έλυνε το πρόβλημα αφού η επανάληψη θα ήταν μονόδρομος.

Για αυτό τον λόγο υιοθετήθηκε μια πιο σύνθετη λύση η οποία μέσω ενός πίνακα συνδέει την πληροφορία του item που χρησιμοποιήθηκε και της επιθυμητής συχνότητας βάσει της οποίας θα γίνει η σύγκριση της τρέχουσας συχνότητας.

Στην αρχή ορίστηκαν οι συχνότητες με την επιστημονική ονομασία τους με την χρήση του #define για μεγαλύτερη ευκολία στην ανάγνωση του κώδικα. Το define σε αντίθεση με τον const δεν χρησιμοποιεί την ram του μικροϋπολογιστή.

```
#define A1 55.00        //55.0000
#define A2 110.00       //110.000
#define A3 220.00       //220.000
#define A4 440.00       //440.000
#define B2 123.47       //123.471
#define B3 246.94       //246.942
#define C3 130.81       //130.813
#define D2 73.42        //73.4162
#define D3 146.83       //146.832
#define D4 293.66       //293.665
#define E1 41.20        //41.2034
#define Eb2 77.78       //77.7817
#define E2 82.41        //82.4069
#define E3 164.81       //164.814
#define E4 329.63       //329.628
#define F3 174.61       //174.614
#define G2 98.00        //97.9989
#define G3 196.00       //195.998
```


Στη συνέχεια ορίστηκαν οι ονομασίες των items, και καθώς η βιβλιοθήκη που χρησιμοποιήθηκε για τη διαχείριση του menu προβλέπει την χρήση κάποιου "shortkey" δίπλα από την ονομασία , προστέθηκαν και shortkeys σε όλα τα items. Το shortkey ορίζεται όταν μετά την ονομασία του item ακολουθήσει "," και στη συνέχεια ένας χαρακτήρας και δεν είναι υποχρεωτικό να το φέρουν όλα τα items.

Παρακάτω είναι το μέρος του κώδικα στο οποίο ορίζονται οι ονομασίες των items και των shorkeys:

```
MenuItem Std_Tune = MenuItem("1");
MenuItem Std_6E2 = MenuItem("1A",'A');
MenuItem Std_5A2 = MenuItem("1B",'B');
MenuItem Std_4D3 = MenuItem("1C",'C');
MenuItem Std_3G3 = MenuItem("1D",'D');
MenuItem Std_2B3 = MenuItem("1E",'E');
MenuItem Std_1E4 = MenuItem("1F",'F');
MenuItem Drop_D = MenuItem("2");
MenuItem DD_6D2 = MenuItem("2G",'G');
MenuItem DD_5A2 = MenuItem("2B",'B');
MenuItem DD_4D3 = MenuItem("2C",'C');
MenuItem DD_3G3 = MenuItem("2D",'D');
MenuItem DD_2B3 = MenuItem("2E",'E');
MenuItem DD_1E4 = MenuItem("2F",'F');
MenuItem D_Tune = MenuItem("3");
MenuItem D_6D2 = MenuItem("3G",'G');
MenuItem D_5B2 = MenuItem("3H",'H');
MenuItem D_4C3 = MenuItem("3I",'I');
MenuItem D_3F3 = MenuItem("3J",'J');
MenuItem D_2A3 = MenuItem("3K",'K');
MenuItem D_1D4 = MenuItem("3L",'L');
MenuItem Open_G = MenuItem("4");
MenuItem OG_6D2 = MenuItem("4G",'G');
MenuItem OG_5G2 = MenuItem("4M",'M');
MenuItem OG_4D3 = MenuItem("4C",'C');
MenuItem OG_3G3 = MenuItem("4D",'D');
MenuItem OG_2B3 = MenuItem("4E",'E');
MenuItem OG_1D4 = MenuItem("4L",'L');
MenuItem Octave = MenuItem("5");
MenuItem Oct_6E1 = MenuItem("5N",'N');
MenuItem Oct_5A1 = MenuItem("5O",'O');
MenuItem Oct_4D2 = MenuItem("5G",'G');
MenuItem Oct_3G2 = MenuItem("5M",'M');
```

MenuItem Oct_2B2 = MenuItem("5H",'H');
MenuItem Oct_1E3 = MenuItem("5P",'P');

Παρατηρείται ότι τα shortkeys ακολουθούν το αγγλικό αλφάβητο με το "A" να ορίζει το πρώτο item και το "P" το τελευταίο. Υπενθυμίζεται ότι εφ' όσον τα shorkeys ορίστηκαν ως μεταβλητές *char* μπορεί να υπάρξει αντιστοιχία με τον πίνακα ascii όπου το "A"είναι ο αριθμός 65, ο "B" ο αριθμός 66 κ.ο.κ.

Στη συνέχεια δημιουργείται ένας πίνακες που φέρει τις συχνότητες-νότες, που ορίστηκαν με *#define* προηγουμένως.

float official_freq[] ={E2, A2, D3, G3, B3, E4, D2, B2, C3, F3, A3, D4, G2, E1, A1, E3} ;

Παρατηρείται ότι τα περιεχόμενο του πίνακα *official_freq[]* αναλογούν στα shortkeys που ορίστηκαν για τα items Για παράδειγμα η τιμή του *official_freq[0]* είναι η συχνότητα που αντιστοιχεί στο item με shortkey "A", η τιμή του *official_freq[1]* είναι η συχνότητα που αντιστοιχεί στο item με shortkey "B", κ.ο.κ.

Στην συνάρτηση *menuUsed*, η οποία καλείται όταν επιλεχθεί ένα συγκεκριμένο item και στην οποία πραγματοποιείται η απομόνωση των αρμονικών συχνοτήτων, η σύγκριση της συχνότητας και ο μηδενισμός του σφάλματος, συνδυάζονται τα shortkeys με τον πίνακα *official_freq[]*.

Στις πρώτες γραμμές της menuUsed τοποθετείται ο παρακάτω κώδικας:

test_shortkey = used.item.getShortkey();
right_freq=official_freq[test_shortkey-'A'];

Στην πρώτη γραμμή, η μεταβλητή *"test_shortkey"* παίρνει την τιμή του shortkey του item που επιλέχθηκε. Αυτή η τιμή είναι από τον *char* 'A' που είναι ο αριθμός 65 μέχρι τον 'P' που είναι 80. Στην δεύτερη γραμμή ορίζεται η συχνότητα με την οποία θα γίνει η σύγκριση, δηλαδή η επιλεγμένη συχνότητα με την ονομασία *"right_freq"*. Η *"right_freq"* θα πάρει μία τιμή από τον πίνακα *official_freq[]* ο οποίος φέρει τις συχνότητες-νότες. Το αποτέλεσμα της αφαίρεσης που επιτελείται μέσα στον πίνακα θα ορίσει ποιο, κατά σειρά, από τα περιεχόμενα του πίνακα θα επιλεχθεί. Επειδή όμως υπάρχει αντιστοιχία μεταξύ των περιεχόμενων του πίνακα και των shortkeys, η αφαίρεση της

τιμής του shortkey κατά 'A' δηλαδή με τον αριθμό 65 έχει ως αποτέλεσμα τον αριθμό με τον οποίο έγινε η αρχική αντιστοιχία του πίνακα. Για παράδειγμα, αν επιλέχθηκε το item με shortkey τον *char* 'A', το οποίο έχει ως επιλεγμένη νότα την E2 με συχνότητα 82,41 Hz, το αποτέλεσμα της αφαίρεσης θα είναι

$$'A' - 'A' = 65 - 65 = 0$$

άρα θα έχουμε

$$official\_freq[test\_shortkey-'A'] = official\_freq[0] = E2 = 82,41Hz$$

Με ανάλογο τρόπο καθορίζονται και οι υπόλοιπες συχνότητες-νότες καθώς υπάρχει αντιστοιχία των shortkeys με τον πίνακα *official_freq[]*.

Ο κώδικας λοιπόν με την παραπάνω διαδικασία γίνεται πιο μικρός σε μέγεθος, πιο γρήγορος και πιο εύχρηστος σε όποιον θέλει να τον διαχειριστεί.

IV. Κώδικας

```
//Στην αρχή ορίζουμε τις βιβλιοθήκες που χρησιμοποιήσαμε
#include <MenuBackend.h>    //MenuBackend. βιβλιοθήκη για τα αντικείμενα και υην υλοποίηση
του menu από τον Alexander Brevig
#include <LiquidCrystal.h> //η βιβλιοθήκη εμπεριέχεται στο Arduino IDE
#include <Servo.h>         //η βιβλιοθήκη εμπεριέχεται στο Arduino IDE



//η παρακάτω συνάρτηση βοήθησε κατα τη διάρκεια των
//δοκιμών. Επιστρέφει κάθε φορά που καλείται το υπόλοιπο διαθέσιμο της μνημης RAM
int availableMemory() {
  // Use 1024 with ATmega168
  int size = 2048;
  byte *buf;
  while ((buf = (byte *) malloc(--size)) == NULL);
    free(buf);
  return size;
}



//ορισμός του servo κινητήρα
Servo myservo;



//ορισμός μεταβλητών για τον υπολογισμό της συχνότητας
long time;
long old_time;
int data;
float T;
```

```
float freq;


//ορισμός μεταβλητών για την σύγκριση της συχνότητας
float now_freq;
float super_freq;


//ορισμός μεταβλητών για τη σύνδεση item_used και επιθυμητής συχνότητας
char test_shortkey;
float right_freq;


//ορισμός μεταβλητών το μηδενισμό του σφάλματος
float error;
int Kp=9;


//ορισμός μεταβλητών για την απομόνωση των αρμονικών συχνοτήτων
float harm4_minus;
float harm3_plus;
float harm3_minus;
float harm_half;


//ορισμός με define των επιστημονικών ονομασιών των νοτών με τις συχνότητές τους.
// η ακρίβεια είναι δύο δεκαδικών
#define A1 55.00     //55.0000
#define Ab2 103.83   //103.826
#define A2 110.00    //110.000
#define A3 220.00    //220.000
#define A4 440.00    //440.000
#define B2 123.47    //123.471
#define Bb3 233.08   //233.082
```

```
#define B3 246.94   //246.942
#define C2 65.41    //65.4064
#define C3 130.81   //130.813
#define D2 73.42    //73.4162
#define Db3 138.59  //138.591
#define D3 146.83   //146.832
#define D4 293.66   //293.665
#define E1 41.20    //41.2034
#define Eb2 77.78   //77.7817
#define E2 82.41    //82.4069
#define E3 164.81   //164.814
#define Eb4 311.13  //311.127
#define E4 329.63   //329.628
#define F3 174.61   //174.614
#define G2 98.00    //97.9989
#define Gb3 185.00  //184.997
#define G3 196.00   //195.998
#define G4 391.99   //391.995


//πίνακας που συνδέει τα shortkeys των items με τις νότες
float official_freq[] ={E2, A2, D3, G3, B3, E4, D2, G2, C3, F3, A3, D4, E1, A1, B2, E3/*, C2, G4,
Eb2, Ab2, Db3, Gb3, Bb3, Eb4*/} ;


//ορισμός μεταβλητών για τα push-buttons
const int buttonPinLeft = 12;     // pin για το αριστερό κουμπί
const int buttonPinRight = 13;    // pin για το δεξί κουμπί
const int buttonPinEsc = 11;      // pin για το Esc κουμπί
const int buttonPinEnter = 10;    // pin για το Enter κουμπί


//ορισμός μεταβλητών για τη γνωστοποίηση
//ποιο push-button πατήθηκε
```

```
int lastButtonPushed = 0;

int lastButtonEnterState = LOW;   // η προηγούμενη μέτρηση για το Enter input pin
int lastButtonEscState = LOW;   // η προηγούμενη μέτρηση για το Esc input pin
int lastButtonLeftState = LOW;   // η προηγούμενη μέτρηση για το αριστερό input pin
int lastButtonRightState = LOW;   // η προηγούμενη μέτρηση για το δεξί input pin


//ορισμός μεταβλητών για το debounce
long lastEnterDebounceTime = 0;
long lastEscDebounceTime = 0;
long lastLeftDebounceTime = 0;
long lastRightDebounceTime = 0;
long debounceDelay = 100;


// Σύνδεση LCD με:
// rs(pin 4 στην lcd) στο pin 8
// rw γείωση
// enable(pin 6 στην lcd) στο pin 3
// d4, d5, d6, d7 στα pins 4, 5, 6, 7
LiquidCrystal lcd(8, 3, 4, 5, 6, 7);


//Μεταβλητές menu
MenuBackend menu = MenuBackend(menuUsed,menuChanged);


  MenuItem Std_Tune = MenuItem("1");
    MenuItem Std_6E2 = MenuItem("1A",'A');
    MenuItem Std_5A2 = MenuItem("1B",'B');
    MenuItem Std_4D3 = MenuItem("1C",'C');
    MenuItem Std_3G3 = MenuItem("1D",'D');
    MenuItem Std_2B3 = MenuItem("1E",'E');
```

```
    MenuItem Std_1E4 = MenuItem("1F",'F');
  MenuItem Drop_D = MenuItem("2");
    MenuItem DD_6D2 = MenuItem("2G",'G');
    MenuItem DD_5A2 = MenuItem("2B",'B');
    MenuItem DD_4D3 = MenuItem("2C",'C');
    MenuItem DD_3G3 = MenuItem("2D",'D');
    MenuItem DD_2B3 = MenuItem("2E",'E');
    MenuItem DD_1E4 = MenuItem("2F",'F');
  MenuItem D_Tune = MenuItem("3");
    MenuItem D_6D2 = MenuItem("3G",'G');
    MenuItem D_5G2 = MenuItem("3H",'H');
    MenuItem D_4C3 = MenuItem("3I",'I');
    MenuItem D_3F3 = MenuItem("3J",'J');
    MenuItem D_2A3 = MenuItem("3K",'K');
    MenuItem D_1D4 = MenuItem("3L",'L');
  MenuItem Open_G = MenuItem("4");
    MenuItem OG_6D2 = MenuItem("4G",'G');
    MenuItem OG_5G2 = MenuItem("4H",'H');
    MenuItem OG_4D3 = MenuItem("4C",'C');
    MenuItem OG_3G3 = MenuItem("4D",'D');
    MenuItem OG_2B3 = MenuItem("4E",'E');
    MenuItem OG_1D4 = MenuItem("4L",'L');
  MenuItem Octave = MenuItem("5");
    MenuItem Oct_6E1 = MenuItem("5M",'M');
    MenuItem Oct_5A1 = MenuItem("5N",'N');
    MenuItem Oct_4D2 = MenuItem("5G",'G');
    MenuItem Oct_3G2 = MenuItem("5H",'H');
    MenuItem Oct_2B2 = MenuItem("5O",'O');
    MenuItem Oct_1E3 = MenuItem("5P",'P');

/*

τα παρακάτω items παραλήφθηκαν λόγω έλλειψης ram
παρ' όλα αυτά μπορούν να χρησιμοποιηθούν σε
```

άλλη υλοποίηση ή να γίνουν enable τα παρακάτω
και disable τα παραπάνω

```
  MenuItem Drop_C = MenuItem("6");
    MenuItem DC_6C2 = MenuItem("6Q",'Q');
    MenuItem DC_5G2 = MenuItem("6H",'H');
    MenuItem DC_4C3 = MenuItem("6I",'I');
    MenuItem DC_3F3 = MenuItem("6J",'J');
    MenuItem DC_2A3 = MenuItem("6K",'K');
    MenuItem DC_1D4 = MenuItem("6L",'L');
   MenuItem New_Std = MenuItem("7");
    MenuItem NStd_6C2 = MenuItem("7Q",'Q');
    MenuItem NStd_5G2 = MenuItem("7H",'H');
    MenuItem NStd_4D3 = MenuItem("7C",'C');
    MenuItem NStd_3A3 = MenuItem("7K",'K');
    MenuItem NStd_2E4 = MenuItem("7F",'F');
    MenuItem NStd_1G4 = MenuItem("7R",'R');
   MenuItem Eb_Tune = MenuItem("8");
    MenuItem Eb_6Eb2 = MenuItem("S",'S');
    MenuItem Eb_5Ab2 = MenuItem("T",'T');
    MenuItem Eb_4Db3 = MenuItem("U",'U');
    MenuItem Eb_3Gb3 = MenuItem("V",'V');
    MenuItem Eb_2Bb3 = MenuItem("W",'W');
    MenuItem Eb_1Eb4 = MenuItem("X",'X');
*/
   MenuItem Fast = MenuItem("9");
    MenuItem Loose = MenuItem("9Y",'Y');
    MenuItem Tight = MenuItem("9Z",'Z');


//η συνάρτηση που καλείται από interrupt
//σε αυτήν υπολογίζεται η τρέχουσα συχνότητα
void fun()
{
time=micros();
```

```
T=abs(old_time-time);
old_time=time;
data=1;
freq=1000000/T;
}



//συνάρτηση setup
void setup()
{

// ενεργοποίηση interrupt από την είσοδο D2
//το οποίο interrupt θα τρέξει την συνάρτηση fun.
//το interrupt θα τρέχει όταν το σήμα στην D2 αυξάνει (RISING)
 attachInterrupt(0,fun,RISING);


// ο servo κινητήρας θα ελέγχεται από το pin D9
 myservo.attach(9);

//τίθεται ο servo σε ακινησία (παλμός 1470us)
 myservo.writeMicroseconds(1470);


//ορισμός των μεταβλητών των pushbuttons ως είσοδοι -inputs
 pinMode(buttonPinLeft, INPUT);
 pinMode(buttonPinRight, INPUT);
 pinMode(buttonPinEnter, INPUT);
 pinMode(buttonPinEsc, INPUT);


//ορισμός της lcd οθόνης ως οθόνη 16 σειρών και 2 γραμμών
 lcd.begin(16, 2);
```

```
//ορισμός menu
menu.getRoot().add(Std_Tune);
Std_Tune.addRight(Drop_D).addRight(D_Tune).addRight(Open_G).addRight(Octave)/*.addRight
(Drop_C).addRight(New_Std).addRight(Eb_Tune)*/.addRight(Fast).addRight(Std_Tune)/*.addRig
ht(Coffee)*/;

Std_Tune.addAfter(Std_6E2);
Std_6E2.addRight(Std_5A2).addRight(Std_4D3).addRight(Std_3G3).addRight(Std_2B3).addRigh
t(Std_1E4).addRight(Std_6E2);
Std_5A2.addBefore(Std_Tune);Std_4D3.addBefore(Std_Tune);Std_3G3.addBefore(Std_Tune);Std
_2B3.addBefore(Std_Tune);Std_1E4.addBefore(Std_Tune);Std_6E2.addBefore(Std_Tune);

Drop_D.addAfter(DD_6D2);
DD_6D2.addRight(DD_5A2).addRight(DD_4D3).addRight(DD_3G3).addRight(DD_2B3).addRi
ght(DD_1E4).addRight(DD_6D2);
DD_5A2.addBefore(Drop_D);DD_4D3.addBefore(Drop_D);DD_3G3.addBefore(Drop_D);DD_2
B3.addBefore(Drop_D);DD_1E4.addBefore(Drop_D);DD_6D2.addBefore(Drop_D);

D_Tune.addAfter(D_6D2);
D_6D2.addRight(D_5G2).addRight(D_4C3).addRight(D_3F3).addRight(D_2A3).addRight(D_1D
4).addRight(D_6D2);
D_5G2.addBefore(D_Tune);D_4C3.addBefore(D_Tune);D_3F3.addBefore(D_Tune);D_2A3.add
Before(D_Tune);D_1D4.addBefore(D_Tune);D_6D2.addBefore(D_Tune);


Open_G.addAfter(OG_6D2);
OG_6D2.addRight(OG_5G2).addRight(OG_4D3).addRight(OG_3G3).addRight(OG_2B3).addRi
ght(OG_1D4).addRight(OG_6D2);
OG_5G2.addBefore(Open_G);OG_4D3.addBefore(Open_G);OG_3G3.addBefore(Open_G);OG_
2B3.addBefore(Open_G);OG_1D4.addBefore(Open_G);OG_6D2.addBefore(Open_G);

Octave.addAfter(Oct_6E1);
Oct_6E1.addRight(Oct_5A1).addRight(Oct_4D2).addRight(Oct_3G2).addRight(Oct_2B2).addRig
```

```
ht(Oct_1E3).addRight(Oct_6E1);
  Oct_5A1.addBefore(Octave);Oct_4D2.addBefore(Octave);Oct_3G2.addBefore(Octave);Oct_2B2.
addBefore(Octave);Oct_1E3.addBefore(Octave);Oct_6E1.addBefore(Octave);


/*

τα παρακάτω items παραλήφθηκαν λόγω έλλειψης ram
παρ' όλα αυτά μπορούν να χρησιμοποιηθούν σε άλλη υλοποίηση
 Drop_C.addAfter(DC_6C2);
  DC_6C2.addRight(DC_5G2).addRight(DC_4C3).addRight(DC_3F3).addRight(DC_2A3).addRig
ht(DC_1D4).addRight(DC_6C2);
  DC_5G2.addBefore(Drop_C);DC_4C3.addBefore(Drop_C);DC_3F3.addBefore(Drop_C);DC_2A
3.addBefore(Drop_C);DC_1D4.addBefore(Drop_C);DC_6C2.addBefore(Drop_C);

 New_Std.addAfter(NStd_6C2);
 NStd_6C2.addRight(NStd_5G2).addRight(NStd_4D3).addRight(NStd_3A3).addRight(NStd_2E4)
.addRight(NStd_1G4).addRight(NStd_6C2);
 NStd_5G2.addBefore(New_Std);NStd_4D3.addBefore(New_Std);NStd_3A3.addBefore(New_Std
);NStd_2E4.addBefore(New_Std);NStd_1G4.addBefore(New_Std);NStd_6C2.addBefore(New_Std
);

 Eb_Tune.addAfter(Eb_6Eb2);
  Eb_6Eb2.addRight(Eb_5Ab2).addRight(Eb_4Db3).addRight(Eb_3Gb3).addRight(Eb_2Bb3).add
Right(Eb_1Eb4).addRight(Eb_6Eb2);
 Eb_5Ab2.addBefore(Eb_Tune);Eb_4Db3.addBefore(Eb_Tune);Eb_3Gb3.addBefore(Eb_Tune);Eb
_2Bb3.addBefore(Eb_Tune);Eb_1Eb4.addBefore(Eb_Tune);Eb_6Eb2.addBefore(Eb_Tune);
*/
 Fast.addAfter(Loose);
 Loose.addRight(Tight).addRight(Loose);
 Loose.addBefore(Fast);Tight.addBefore(Fast);
```

```
//αρχικός χαιρετισμός
 lcd.setCursor(0,0);
 lcd.print("TEI Automatismou");
 lcd.setCursor(0,1);
 lcd.print("Guitar AutoTuner");

}


void loop()
{

 readButtons();  //συνάρτηση ανάγνωσης των pushbuttons
 navigateMenus();  //χειρισμός menu


 //ακινητοποίηση servo κινητήρα όταν δεν γίνεται ο μηδενισμός του σφάλματος
 myservo.writeMicroseconds(1470);
}


//συνάρτηση για την ενημέρωση της lcd οθόνης σε σχέση με την τρέχουσα
//λειτουργία του menu
void menuChanged(MenuChangeEvent changed){


 //η μεταβλητή newMenuItem παίρνει την ονομασία της νέας θέσης του menu
 MenuItem newMenuItem=changed.to;


 lcd.setCursor(0,1);

 if(newMenuItem==menu.getRoot()){
    lcd.setCursor(0,0);
```

```
  lcd.clear();
  lcd.print("Main Menu      ");
}else if(newMenuItem==Std_Tune){
 lcd.setCursor(0,0);
 lcd.clear();
 lcd.print("<Stand. tuning >");
}else if(newMenuItem==Std_6E2){
  lcd.print("6th string  E2  ");
}else if(newMenuItem==Std_5A2){
  lcd.print("5th string  A2  ");
}else if(newMenuItem==Std_4D3){
  lcd.print("4th string  D3  ");
}else if(newMenuItem==Std_3G3){
  lcd.print("3th string  G3  ");
}else if(newMenuItem==Std_2B3){
  lcd.print("2th string  B3  ");
}else if(newMenuItem==Std_1E4){
  lcd.print("1th string  E4  ");
}else if(newMenuItem==Drop_D){
 lcd.setCursor(0,0);
 lcd.clear();
 lcd.print("<Drop D        >");
}else if(newMenuItem==DD_6D2){
  lcd.print("6th string  D2  ");
}else if(newMenuItem==DD_5A2){
  lcd.print("5th string  A2  ");
}else if(newMenuItem==DD_4D3){
  lcd.print("4th string  D3  ");
}else if(newMenuItem==DD_3G3){
  lcd.print("3th string  G3  ");
}else if(newMenuItem==DD_2B3){
  lcd.print("2th string  B3  ");
}else if(newMenuItem==DD_1E4){
  lcd.print("1th string  E4  ");
```

```
  }else if(newMenuItem==D_Tune){
    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("<D tuning     >");
  }else if(newMenuItem==D_6D2){
     lcd.print("6th string  D2  ");
  }else if(newMenuItem==D_5G2){
     lcd.print("5th string  G2  ");
  }else if(newMenuItem==D_4C3){
     lcd.print("4th string  C3  ");
  }else if(newMenuItem==D_3F3){
     lcd.print("3th string  F3  ");
  }else if(newMenuItem==D_2A3){
     lcd.print("2th string  A3  ");
  }else if(newMenuItem==D_1D4){
     lcd.print("1th string  D4  ");
  }else if(newMenuItem==Open_G){
    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("<Open G       >");
  }else if(newMenuItem==OG_6D2){
     lcd.print("6th string  D2  ");
  }else if(newMenuItem==OG_5G2){
     lcd.print("5th string  G2  ");
  }else if(newMenuItem==OG_4D3){
     lcd.print("4th string  D3  ");
  }else if(newMenuItem==OG_3G3){
     lcd.print("3th string  G3  ");
  }else if(newMenuItem==OG_2B3){
     lcd.print("2th string  B3  ");
  }else if(newMenuItem==OG_1D4){
     lcd.print("1th string  D4  ");
  }else if(newMenuItem==Octave){
  lcd.setCursor(0,0);
```

```
 lcd.clear();
 lcd.print("<Octave down  >");
}else if(newMenuItem==Oct_6E1){
   lcd.print("6th string  E1  ");
}else if(newMenuItem==Oct_5A1){
   lcd.print("5th string  A1  ");
}else if(newMenuItem==Oct_4D2){
   lcd.print("4th string  D2  ");
 }else if(newMenuItem==Oct_3G2){
   lcd.print("3th string  G2  ");
 }else if(newMenuItem==Oct_2B2){
   lcd.print("2th string  B2  ");
 }else if(newMenuItem==Oct_1E3){
    lcd.print("1th string  E3  ");



/*
τα παρακάτω items παραλήφθηκαν λόγω έλλειψης ram
παρ' όλα αυτά μπορούν να χρησιμοποιηθούν σε
άλλη υλοποίηση ή να γίνουν enable τα παρακάτω
και disable τα παραπάνω



  }else if(newMenuItem==Drop_C){
    lcd.setCursor(0,0);
    lcd.clear();
    lcd.print("<Drop C       >");
  }else if(newMenuItem==DC_6C2){
     lcd.print("6th string  C2  ");
  }else if(newMenuItem==DC_5G2){
     lcd.print("5th string  G2  ");
  }else if(newMenuItem==DC_4C3){
```

```
  lcd.print("4th string  C3  ");
}else if(newMenuItem==DC_3F3){
  lcd.print("3th string  F3  ");
}else if(newMenuItem==DC_2A3){
  lcd.print("2th string  A3  ");
}else if(newMenuItem==DC_1D4){
  lcd.print("1th string  D4  ");
}else if(newMenuItem==New_Std){
 lcd.setCursor(0,0);
 lcd.clear();
 lcd.print("<New Std tuning>");
}else if(newMenuItem==NStd_6C2){
  lcd.print("6th string  C2  ");
}else if(newMenuItem==NStd_5G2){
  lcd.print("5th string  G2  ");
}else if(newMenuItem==NStd_4D3){
  lcd.print("4th string  D3  ");
}else if(newMenuItem==NStd_3A3){
  lcd.print("3th string  A3  ");
}else if(newMenuItem==NStd_2E4){
  lcd.print("2th string  E4  ");
}else if(newMenuItem==NStd_1G4){
  lcd.print("1th string  G4  ");
}else if(newMenuItem==Eb_Tune){
 lcd.setCursor(0,0);
 lcd.clear();
 lcd.print("<Eb/D# tuning >");
}else if(newMenuItem==Eb_6Eb2){
  lcd.print("6th string  Eb2  ");
}else if(newMenuItem==Eb_5Ab2){
  lcd.print("5th string  Ab2  ");
}else if(newMenuItem==Eb_4Db3){
  lcd.print("4th string  Db3  ");
}else if(newMenuItem==Eb_3Gb3){
```

```
    lcd.print("3th string  Gb3  ");
  }else if(newMenuItem==Eb_2Bb3){
    lcd.print("2th string  Bb3  ");
  }else if(newMenuItem==Eb_1Eb4){
    lcd.print("1th string  Eb4  ");

*/
  }else if(newMenuItem==Fast){
   lcd.setCursor(0,0);
   lcd.clear();
   lcd.print("<Fast Spin    >");
  }else if(newMenuItem==Loose){
     lcd.print(" Loosen string  ");
  }else if(newMenuItem==Tight){
     lcd.print(" Tighten string ");
  }
 }


 //συνάρτηση για την εκτέλεση των εντολών που έχουν προγραμματιστεί
 //δηλαδή την σύγκριση της συχνότητας και τον μηδενισμό του σφάλματος
 //ή την λειτουργία του servo κινητήρα για την αντικατάσταση της χορδής
 void menuUsed(MenuUseEvent used){

 //η μεταβλητή test_shortkey κρατάει την τιμή του shortkey του item που επιλέχθηκε
 test_shortkey = used.item.getShortkey();


 //αναγνωρίζεται αν πρόκεειται για αντικατάστση χορδής (που δεν χρειάζεται
 //σύγκριση συχνότητας)....

  if (test_shortkey=='Y'||test_shortkey=='Z'){
    while (digitalRead(buttonPinEsc) == LOW){
     if  (test_shortkey=='Y')  myservo.writeMicroseconds(2000);
```

```
    else myservo.writeMicroseconds(1000); }
}
//...η αν πρόκειται και σύγκριση συχνότητας
else {

// οι παρακάτω 6 γραμμές εξηγούνται αναλυτικά στο κεφάλαιο IV και ενότητα Β.
 right_freq=official_freq[test_shortkey-'A'];
 harm4_minus=right_freq*4/3-2;
 harm3_plus=right_freq*5/4+2;
 harm3_minus=right_freq*5/4-2;
 harm_half=right_freq/1.9;
 int count=0;

// ορίζεται η μεταβλητή range από την οποία θα ορίζεται αν η τράχουσα συχνότητα βρίσκεται
//σε ένα λογικό πλαίσιο ακρίβειας
 float range=(right_freq-right_freq/1.06)/77;
 lcd.setCursor(0,1);
 lcd.print("Tuning...      ");
data=0;

//διαδικασία της σύγκρισης συχνότητας και τον μηδενισμο του σφάλματος
// η διαδικασία διακόπτεται μόνο από το push-button Esc ή αν μηδενιστεί το σφάλμα
while (digitalRead(buttonPinEsc) == LOW){


//ελέγχεται έχει γίνει interrupt και αν ναι, η μεταβλητή now_freq
//παίρνει την τιμή της freq από την συνάρτηση fun έτσι ώστε να ελέγχεται μία συχνότητα
//και να μην αλλάζει η τιμή της λόγω Interrupt.
 if (data==1){ now_freq=freq;

//απομόνωση αρμονικών μιας οκταβας κάτω και πάνω από την 4η νοτα
if (now_freq>harm_half&&now_freq<harm4_minus){
//απομόνωση αρμονικών 3ης νότα
  if (now_freq>harm3_plus||now_freq<harm3_minus){
```

```
//δημιουργία super_freq
      if (count<7) {if (count==0){super_freq=now_freq;}goto tune;} else
//απομόνωση συχνοτήτων έξω από το εύρος συχνοτήτων που έχει διαμορφωθεί
  if (now_freq<super_freq+15&&now_freq>super_freq-15)
{


tune:count++;


//σύγκριση συχνότητας
if      ((super_freq>=right_freq-range&&super_freq<=right_freq+range)||(now_freq>=right_freq-
range&&now_freq<=right_freq+range))  //apo edw kai pera to exw ligo sto flou mexri na valw to
moter
  {
  myservo.writeMicroseconds(1470);
  data=0;
  for (int i=0;i<7;i++){
  lcd.setCursor(0,1);
  lcd.print("Tuning Complete!");
  lcd.noDisplay();
  delay(400);
  lcd.display();
  delay(400);}
  menu.moveRight();
  break;
  }

 float a=0.4*now_freq;     //δημιουργία ενός απλού moving average
 float b=0.6*super_freq;   //με 40% την τρέχουσα συχνότητα και
 super_freq=a+b;           //60% τις προηγούμενες τιμές


//εύρεση και μηδενισμός σφάλματος
```

```
//μέσω ελεγκτή P και ψευδο-ελεγκτή I
error=right_freq-super_freq;
if (count>5){
 if (error<2)
   {myservo.writeMicroseconds(1470-2.5*error*Kp);}
 else
   {myservo.writeMicroseconds(1470-error*Kp);}
delay(150);}


data=0;
}else myservo.writeMicroseconds(1470);     //teleftaio if
}else myservo.writeMicroseconds(1470);    //3 armoniki
}else myservo.writeMicroseconds(1470);   //4 armoniki kai half
}else myservo.writeMicroseconds(1470);
}myservo.writeMicroseconds(1470);       //...while

}}


//συνάρτηση ανάγνωσης των pushbuttons
void  readButtons(){
 int reading;
 int buttonEnterState=LOW;
 int buttonEscState=LOW;
 int buttonLeftState=LOW;
 int buttonRightState=LOW;

//κουμπί Enter


reading = digitalRead(buttonPinEnter);
```

```
if (reading != lastButtonEnterState) {

lastEnterDebounceTime = millis();

        }

if ((millis() - lastEnterDebounceTime) > debounceDelay)
{

buttonEnterState=reading;
lastEnterDebounceTime=millis();

}

lastButtonEnterState = reading;


//κουμπί Enter


reading = digitalRead(buttonPinEsc);
if (reading != lastButtonEscState) {

lastEscDebounceTime = millis();

}

if ((millis() - lastEscDebounceTime) > debounceDelay)
{

buttonEscState = reading;
lastEscDebounceTime=millis();
}
```

```
lastButtonEscState = reading;


//κουμπί Down

reading = digitalRead(buttonPinRight);

if (reading != lastButtonRightState)
{

lastRightDebounceTime = millis();

}

if ((millis() - lastRightDebounceTime) > debounceDelay)
{

buttonRightState = reading;
lastRightDebounceTime =millis();

}
lastButtonRightState = reading;


//Up button

reading = digitalRead(buttonPinLeft);
if (reading != lastButtonLeftState)
{

lastLeftDebounceTime = millis();

}
```

```
 if ((millis() - lastLeftDebounceTime) > debounceDelay) {


 buttonLeftState = reading;
 lastLeftDebounceTime=millis();

 }
 lastButtonLeftState = reading;



 //στοιχεια με το ποια pushbuttons έχουν πατηθεί
 if (buttonEnterState==HIGH){
 lastButtonPushed=buttonPinEnter;

 }else if(buttonEscState==HIGH){
 lastButtonPushed=buttonPinEsc;

 }else if(buttonRightState==HIGH){
 lastButtonPushed=buttonPinRight;

 }else if(buttonLeftState==HIGH){
 lastButtonPushed=buttonPinLeft;

 }else{
 lastButtonPushed=0;
 }
 }

void navigateMenus() {
 MenuItem currentMenu=menu.getCurrent();

 switch (lastButtonPushed){
```

```
    case buttonPinEnter:
    //αν αυτό το menu έχει ένα item-παιδί τότε
    //τίθεται σε λειτουγία η συναρτηση menuUsed...
     if(!(currentMenu.moveDown())){
       menu.use();
    //...αλλιώς το menu συνεχίζει κανονικά
     }else{
      menu.moveDown();
     }
     break;
   case buttonPinEsc:
     menu.moveUp();
     break;
   case buttonPinRight:
     menu.moveRight();
     break;
   case buttonPinLeft:
     menu.moveLeft();
     break;
  }

 lastButtonPushed=0;
}
```