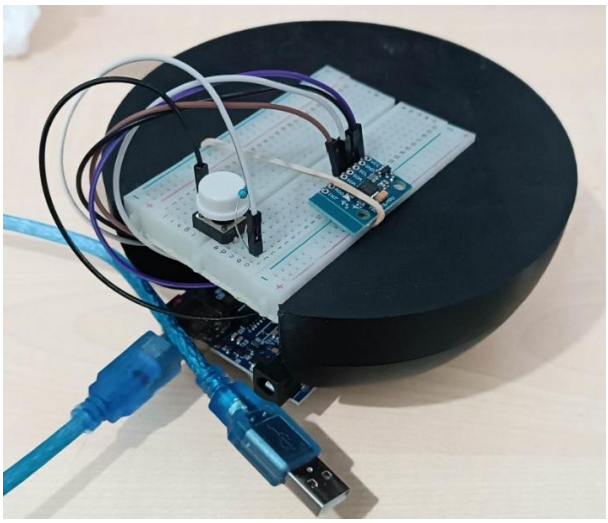
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός και Ανάπτυξη Μουσικού Οργάνου με τη χρήση του μικροελεγκτή Arduino»



Του φοιτητή
Αρσενίου Σεραφείμ
Αρ. Μητρώου: 164806

Επιβλέπων
Χριστίνα Βολιώτη
Έκτακτο Εκπαιδευτικό Προσωπικό

Ημερομηνία 10/09/2023

Τίτλος Δ.Ε. Σχεδιασμός και Ανάπτυξη Μουσικού Οργάνου με τη χρήση του μικροελεγκτή Arduino

Κωδικός Δ.Ε. 22286

Ονοματεπώνυμο φοιτητή/τών Αρσενίου Σεραφείμ

Ονοματεπώνυμο εισηγητή Βολιώτη Χριστίνα

Ημερομηνία ανάληψης Δ.Ε. 24-10-2022

Ημερομηνία περάτωσης Δ.Ε. 10-09-2023

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

«Στην οικογένεια μου και σε όσους ήταν και είναι

δίπλα μου σε κάθε δύσκολη στιγμή της ζωής μου.»

Πρόλογος

Η απόφαση εκπόνησης του συγκεκριμένου θέματος πτυχιακής εργασίας δημιουργήθηκε λόγω του ενδιαφέροντος που μου προκάλεσε το μάθημα «Οργάνωση και Αρχιτεκτονική Υπολογιστικών Συστημάτων» που διδάσκεται στο τμήμα, όσον αφορά το πρακτικό κομμάτι του Arduino. Παρ' ότι η ιδέα μου αρχικά ήταν διαφορετική, κατόπιν συνεννόησης με την Κα. Βολιώτη και λόγω του ότι ήθελα να εξερευνήσω καινούριες τεχνολογίες και να ασχοληθώ πρακτικά με το κομμάτι του Arduino, κατέληξα να επιλέξω το συγκεκριμένο θέμα. Η προσέγγιση που ακολούθησα για να επιτύχω το ακόλουθο αποτέλεσμα δεν είναι η μοναδική. Οι δυνατότητες για ανάπτυξη παρόμοιων project είναι πάρα πολλές. Αναλογίζοντας τις μουσικές γνώσεις και την εμπειρία του καθενός, την προσέγγιση και το είδος μουσικής, τα αποτελέσματα που μπορούν να παραχθούν είναι αμέτρητα.

Περίληψη

Η συγκεκριμένη πτυχιακή έχει σαν στόχο το σχεδιασμό και την ανάπτυξη ενός μουσικού οργάνου που έχει σχήμα σφαίρας (music ball). Ο έλεγχος αυτού πραγματοποιείται με τη χρήση του μικροελεγκτή Arduino. Με αυτόν, ο χρήστης θα έχει τη δυνατότητα να ενεργοποιεί (trigger) διαφορετικά ηχητικά εφέ λαμβάνοντας σαν δεδομένα κάποια φυσικά χαρακτηριστικά, όπως είναι η θέση και η περιστροφή του αντικειμένου. Αυτό έχει σαν αποτέλεσμα να μετατραπεί η μπάλα σε midi controller. Στόχος είναι να δημιουργηθεί ένα εργαλείο που θα διευκολύνει μουσικούς παραγωγούς να συνθέτουν και να επεξεργάζονται τη μουσική τους σε πραγματικό χρόνο. Για την πραγματοποίηση της πραγματοποιήθηκε μελέτη διάφορων εξαρτημάτων και λογισμικών. Η διαδικασία παραγωγής ήχου βασίστηκε πολύ στον πειραματισμό. Η διαδικασία βασίστηκε στην δημιουργία, επεξεργασία μεμονωμένων ήχων, και μετέπειτα μίξη αυτών των ήχων μεταξύ τους, με σκοπό να παραχθεί ένας ήχος με πολλά επίπεδα/στρώματα (layered). Στη συνέχεια πραγματοποιήθηκε μελέτη που αφορά τη μοντελοποίηση και εκτύπωση τρισδιάστατων υλικών, με σκοπό την εκτύπωση ενός αντικειμένου που αποτελεί στήριγμα για το music box. Η παραγωγή του ήχου γίνεται με το πρόγραμμα Max/MSP.

Design and Development of a Musical Instrument Using the Arduino Microcontroller

«Serafim Arseniou»

Abstract

This thesis aims to design and develop a musical instrument in the shape of a music ball. The control of this instrument is achieved using the Arduino microcontroller. With it, the user will be able to trigger different audio effects based on certain physical characteristics, such as the position and rotation of the object. This will transform the ball into a MIDI controller. The goal is to create a tool that facilitates music producers in composing and processing their music in real-time. The process involved the study of various components and software, and sound production heavily relied on experimentation. The sound production process involved creating and processing individual sounds and later mixing them to produce a multi-layered sound. Additionally, a study on 3D modeling and printing was conducted to create a structure to contain the music box. The sound production is done using the Max/MSP software.

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω την καθηγήτρια μου, κυρία Βολιώτη για την ευκαιρία που μου έδωσε για να συνεργαστώ μαζί της, και για όλη την καθοδήγηση και τις συμβουλές που μου παρείχε κατά τη διάρκεια εκπόνησης της παρούσας πτυχιακής εργασίας. Επίσης θα ήθελα να ευχαριστήσω όλους τους φίλους και συμφοιτητές που με συμβούλευαν και μου παρείχαν τις πληροφορίες και τον εξοπλισμό που χρειαζόμουν την στιγμή που τις χρειαζόμουν. Τέλος την οικογένεια μου, για όσα έχει προσφέρει καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Η/Υ	Ηλεκτρονικός Υπολογιστής
3D	Τρισδιάστατος
I/O	Input/Output
IDE	Integrated Development Environment
USB	Universal Serial Bus
PWM	Pulse-width modulation
ICSP	In-Circuit Serial Programming
IMU	Inertial Measurement Unit
AC	Alternating Current
DC	Direct Current
RFID	Radio Frequency Identification
MPU	Microprocessing Unit
I2C	Inter-Integrated Circuit
MIDI	Musical Instrument Digital Interface
ITOA	Integer to ASCII
STL	Stereolithography
WI-FI	Wireless Fidelity

Κεφάλαιο 1ο: Εισαγωγή

1.1 Γενικά

Η βασική ιδέα της παρούσας πτυχιακής εργασίας είναι η εξερεύνηση και η αξιοποίηση των δυνατοτήτων του μικροελεγκτή Arduino σε συνδυασμό με διάφορους αισθητήρες και modules. Οι μικροελεγκτές γενικά χρησιμοποιούνται για την εκτέλεση απλών λειτουργιών στην καθημερινή ζωή. Για να δημιουργηθεί όσο πιο εύκολα και αποδοτικά ένα πλήρως λειτουργικό και ολοκληρωμένο έργο, είναι απαραίτητη η μελέτη ποικίλων μικροελεγκτών. Η επιλογή συγκεκριμένα του Arduino έγινε, καθώς ο συγκεκριμένος μικροελεγκτής παρ' ότι είναι απλός στην χρήση του, παρέχει πολλές δυνατότητες. Μπορεί να διαχειρίζεται ταυτόχρονα πολλαπλές εισόδους, να τις επεξεργάζεται αποτελεσματικά και ταυτόχρονα να παράγει αντίστοιχες εξόδους.

Για να πραγματοποιηθεί λειτουργική αλληλεπίδραση, είναι απαραίτητο να γίνει κατανοητός ο τρόπος διαχείρισης και αξιοποίησης των εξωτερικών συσκευών και λογισμικών με τα οποία θα επικοινωνεί. Πρέπει να γνωρίζουμε τον τρόπο ροής των δεδομένων και να επεξεργαζόμαστε αποδοτικά αυτά τα δεδομένα προκειμένου να επιτύχουμε τις επιθυμητές εξόδους.

Κάθε module που χρησιμοποιείται απαιτεί ξεχωριστή αντιμετώπιση, τόσο σε επίπεδο λογισμικού όσο και σε επίπεδο υλικού (hardware). Πριν από την αγορά τους, πρέπει να γίνει προσεκτική μελέτη και θεωρητική ανάλυση κάθε τμήματος ξεχωριστά, προκειμένου να εξασφαλιστεί ότι η ταυτόχρονη σύνδεση και αλληλεπίδραση των διαφορετικών modules είναι εφικτή και ότι το αποτέλεσμα θα είναι το επιθυμητό. [1]

Κατά τη συγγραφή του απαιτούμενου κώδικα πρέπει να δοθεί ιδιαίτερη προσοχή στην αλληλεπίδραση των διαφόρων modules μεταξύ τους. Αυτά τα modules το πιο πιθανό είναι να βασίζονται σε ξεχωριστές βιβλιοθήκες και θα πρέπει όταν επικοινωνούν μεταξύ τους να λειτουργούν με αποδοτικό τρόπο, χωρίς να παρεμβάλλει το ένα με το άλλο. [2]

Η φυσική κατασκευή που θα αποτελεί το τελικό προϊόν πρέπει να είναι κατά το δυνατόν πρακτική. Δεν πρέπει να είναι μεγάλη, ούτε να καταλαμβάνει πολύ χώρο ή να είναι βαριά. Οι οδηγίες χρήσης της συγκεκριμένης εφαρμογής πρέπει να είναι κατανοητές καθώς και η χρήση του συγκεκριμένου μουσικού οργάνου να είναι εύκολη και άμεση.

Η αξιοπιστία των συστημάτων βασίζεται στη σωστή λειτουργία τόσο του hardware, όσο και στη χρησιμοποίηση αυτού μέσω του λογισμικού (software). [3] Τέλος, για την ορθή λειτουργία του συστήματος ορισμένα ζητήματα ηλεκτρονικής και κυκλωμάτων θα πρέπει να αντιμετωπιστούν, έτσι ώστε να βεβαιωθούμε ότι η τροφοδοσία των υλικών γίνεται σωστά, ώστε να αποφευχθούν τυχόν ανεπιθύμητες βλάβες που μπορούν να καταστρέψουν είτε κάποιο από τα modules, είτε τον μικροελεγκτή Arduino.

1.2 Στόχοι

Στόχος της πτυχιακής είναι να δημιουργηθεί ένα εργαλείο, το οποίο μπορεί στα χέρια κάποιου εξειδικευμένου μουσικού να αποτελέσει μεγάλη βοήθεια. Το εργαλείο αυτό, όπως έχει προαναφερθεί είναι ένα μουσικό όργανο με τη χρήση του Arduino, και εντάσσεται στον τομέα της φυσικής υπολογιστών (physical computing) και του δημιουργικού προγραμματισμού (creative computing). [4] [5]

Φυσική Υπολογιστών είναι ο τομέας της φυσικής υπολογιστών που ασχολείται με τον συνδυασμό της ψηφιακής τεχνολογίας και των φυσικών συστημάτων. Σε αυτόν τον τομέα, οι υπολογιστές και/ή οι μικροελεγκτές (όπως το Arduino) χρησιμοποιούνται για τη δημιουργία συστημάτων που αλληλοεπιδρούν με το περιβάλλον τους μέσω αισθητήρων. [6] Αυτό μπορεί να περιλαμβάνει τη δημιουργία διαδραστικών εγκαταστάσεων, φυσικών κατασκευών ή έργων τέχνης που ανταποκρίνονται σε φυσική κίνηση, ήχο, φως και άλλα φυσικά σήματα. [7]

Δημιουργική Πληροφορική είναι ο τομέας της πληροφορικής που επικεντρώνεται στη χρήση της πληροφορικής και της τεχνολογίας για την έκφραση και τη δημιουργία τέχνης και μουσικής. Σε αυτόν τον τομέα, οι δημιουργοί χρησιμοποιούν τεχνολογικά εργαλεία και περιβάλλοντα προγραμματισμού για να δημιουργήσουν μουσικά έργα, ηχητικά εφέ, μουσικές εφαρμογές και ψηφιακά μουσικά όργανα. Με τη χρήση του Arduino, μπορεί να δημιουργηθούν προσαρμοσμένα μουσικά όργανα, εφέ ήχου, μουσικές εγκαταστάσεις και άλλα συστήματα που συνδυάζουν τη μουσική με την τεχνολογία. [8]

Το Arduino αποτελεί το επίκεντρο της πτυχιακής. Με αυτό θα επιχειρήσουμε να δημιουργήσουμε και επεξεργαστούμε ένα μικρό αριθμό από εναυσμάτων, τα οποία θα οδηγήσουν στη δημιουργία και τον έλεγχο ήχων και μελωδιών, τον προγραμματισμό διάφορων λειτουργιών συσχετιζόμενων με τη μουσική, και όλα αυτά σε πραγματικό χρόνο.

Θα σχεδιάσουμε ένα 3D μοντέλο το οποίο θα αποτελέσει προστατευτικό περίβλημα για το Arduino, θα προστατέψει τα καλώδια και ταυτόχρονα θα είναι μια φιλική προς τον χρήστη κατασκευή, για να μπορεί εύκολα, κατανοητά και άνετα να χρησιμοποιεί το συγκεκριμένο εργαλείο.

1.3 Διάρθρωση κειμένου

Ο τρόπος με τον οποίο έγινε η συγγραφή της πτυχιακής εργασίας ακολουθεί την λογική πορεία βάσει της οποίας έγινε ο συνολικός σχεδιασμός αυτής. Στο τρέχον Κεφάλαιο γίνεται μία πρώτη εισαγωγή στο ζήτημα που μας απασχολεί. Στο Κεφάλαιο 2 γίνεται αναφορά στον μικροελεγκτή Arduino που θα χρησιμοποιήσουμε και ό,τι αφορά αυτόν. Στο Κεφάλαιο 3 αναλύουμε τα τεχνικά χαρακτηριστικά και τη συνδεσμολογία του hardware που θα χρησιμοποιήσουμε. Στο Κεφάλαιο 4 παραθέτουμε και περιγράφουμε κομμάτια κώδικα, άμεσα συσχετιζόμενα με την εργασία που δημιουργούμε. Στο Κεφάλαιο 5 αναφέρουμε το λογισμικό MAX/MSP και περιγράφουμε τις λειτουργίες και χρήσεις του. Επίσης παραθέτουμε και αναλύουμε μερικά από τα διαφορετικά αρχεία (patches) που δημιουργήσαμε προσπαθώντας να εξοικειωθούμε με την εφαρμογή. Στο Κεφάλαιο 6 περιγράφουμε αναλυτικά την βασική εφαρμογή αλληλεπίδρασης με το Arduino, που αποτελεί και την βασική εφαρμογή αναπαραγωγής και επεξεργασίας μουσικής, μέσω του music box που κατασκευάσαμε. Στο Κεφάλαιο 7 αναφέρουμε κάποια βασικά χαρακτηριστικά που αφορούν την τρισδιάστατη εκτύπωση καθώς και περιγράφουμε τη διαδικασία μοντελοποίησης της κατασκευής μέσα στην οποία περικλείεται το Arduino με τα συνοδευόμενα του. Στο Κεφάλαιο 8 καταλήγουμε σε κάποια συμπεράσματα, αναφέρουμε τις προκλήσεις που αντιμετωπίσαμε, καθώς επίσης προτείνουμε και αναφέρουμε μελλοντικές βελτιώσεις με τις οποίες θα θέλαμε να ασχοληθούμε μελλοντικά.

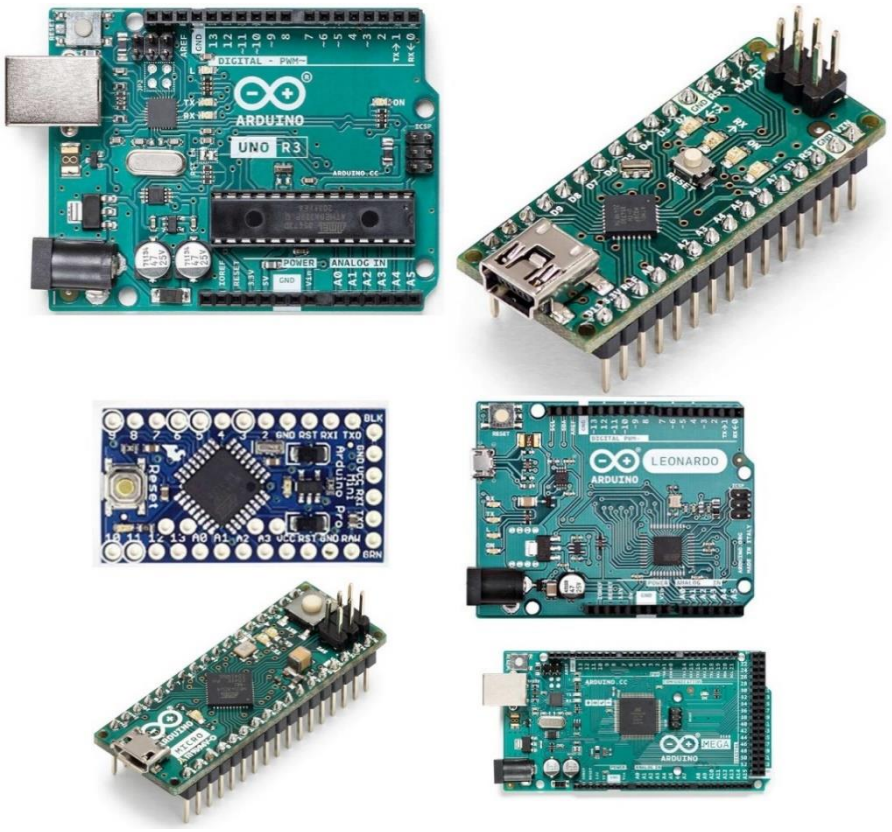
Κεφάλαιο 2ο: Ο μικροελεγκτής Arduino

2.1 Εισαγωγή

Το ακόλουθο κεφάλαιο αφορά τον μικροελεγκτή Arduino, κάποιες γενικές πληροφορίες για αυτόν. Αργότερα θα μιλήσουμε συγκεκριμένα για το μοντέλο Arduino Uno Rev3 και το λογισμικό που χρησιμοποιούμε για να τον προγραμματίσουμε.

2.2 Γενική περιγραφή και δυνατότητες

Το Arduino είναι μία κατηγορία hardware που αποτελείται από ένα φυσικό προγραμματιζόμενο κύκλωμα (συχνά αναφέρεται ως μικροελεγκτής) για τη δημιουργία ηλεκτρονικών έργων. [1] Αποτελεί δηλαδή μια μητρική πλακέτα που έχει πάνω της pins εισόδου/εξόδου (I/O pins), η οποία μπορεί να προγραμματιστεί, χρησιμοποιώντας ένα περιβάλλον ανάπτυξης IDE, που τρέχει σε ηλεκτρονικό υπολογιστή και χρησιμοποιείται για τη συγγραφή και μεταφόρτωση κώδικα στο φυσικό κύκλωμα. Σχεδιάζεται και παράγεται από την ομώνυμη εταιρία Arduino, ωστόσο τα σχέδια και οι πληροφορίες για το υλικό είναι μη προστατευόμενα από πνευματικά δικαιώματα και διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους. Προφανώς, οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες. [9]



Εικόνα 2.1: Arduino Uno (R3), Arduino Nano, Arduino Pro, Arduino Mini [10]

Υπάρχουν πολλές επίσημες εκδόσεις οι οποίες μπορούν να συνδεθούν και να χρησιμοποιηθούν με μία πληθώρα από μικροελεγκτών και αισθητήρων. Διαθέτουν μία σειρά από αναλογικές και ψηφιακές εισόδους και εξόδους, οι οποίες διαφέρουν σε πλήθος ανάλογα και με το εκάστοτε μοντέλο. Βασικό τους χαρακτηριστικό είναι ότι διαθέτουν κανάλια σειριακής επικοινωνίας, συμπεριλαμβανομένου και του USB, έτσι ώστε να "φορτώνονται" τα επιθυμητά προγράμματα στον μικροεπεξεργαστή. [2]

Έτσι, ένα Arduino μπορεί να χρησιμοποιηθεί είτε για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων μέσω των pins που διαθέτει, είτε να συνδεθεί και να επικοινωνεί με breadboards και μεγαλύτερα κυκλώματα. Μπορεί επίσης να συνδεθεί με κάποιο Η/Υ, και να αλληλοεπιδράσει με προγράμματα και γλώσσες όπως είναι το Processing, το MAX/MSP, Jitter, Pure Data κ.ά. Τα Arduino και τα Arduino-compatible boards χρησιμοποιούν επίσης shields, πλακέτες επέκτασης (modules), που συνδέονται κανονικά στα pin headers του Arduino, ώστε να προσδίδουν επιπλέον λειτουργικότητες.

2.3 Arduino Uno Rev3

Το Arduino Uno Rev3 είναι το Arduino που θα χρησιμοποιήσουμε εμείς ως βάση στην εργασία μας. Ο λόγος είναι ότι, τελικά, καλύπτει τις ανάγκες μας και μπορεί να ανταπεξέλθει στις απαιτήσεις που προέκυψαν μετά από τη θεωρητική ανάλυση τόσο του Arduino, όσο και των εκάστοτε modules από τα οποία απαρτίζεται το project.

Πρόκειται για την τελευταία έκδοση του μοντέλου Arduino Uno που είναι βασισμένη στον μικροελεγκτή ATmega328P. [11] Διαθέτει 14 ψηφιακές ακροδέκτες I/O (από τους οποίους 6 μπορούν να χρησιμοποιηθούν ως αναλογο-ψηφιακές εξόδους PWM), 6 αναλογικές εισόδους, ένα resonator 16 MHz (CSTCE16M0V53-R0), μια σύνδεση USB, ένα ηλεκτρικό βύσμα, μια κεφαλίδα ICSP και ένα κουμπί επαναφοράς. Περιλαμβάνει ό,τι χρειάζεται για να λειτουργήσει, αρκεί να συνδεθεί σε έναν υπολογιστή με ένα καλώδιο USB ή να τροφοδοτηθεί με έναν προσαρμογέα AC-to-DC ή μπαταρία 9V συνδεδεμένη στο κατάλληλο power jack. Ένα βασικό του πλεονέκτημα είναι ότι είναι σχετικά φθηνός, οπότε μπορεί εύκολα να αντικατασταθεί αν προκύψει οποιαδήποτε βλάβη στο κύκλωμα. [12]

2.4 Arduino Modules

Arduino modules ή shields ονομάζονται επιπλέον πλακέτες που συνδέονται απευθείας στο Arduino και προσφέρουν επιπλέον λειτουργικότητες και δυνατότητες στο σύστημα. Αυτά τα modules μπορούν να είναι αισθητήρες, ενσωματωμένα κυκλώματα, επέκταση διεπαφών I/O, οθόνες, κινητήρες, επικοινωνιακά μέσα όπως ασύρματα ανταλλακτικά, και πολλά άλλα.

Τα Arduino modules συνδέονται στα αντίστοιχα pins του Arduino board και μπορούν να χρησιμοποιηθούν για να επεκτείνουν τις δυνατότητες του συστήματος, επιτρέποντας την ανάγνωση αισθητήρων, τον έλεγχο εξόδων, την επικοινωνία με άλλες συσκευές και πολλές άλλες λειτουργίες. Τα περισσότερα modules συνοδεύονται από libraries που παρέχουν έτοιμο γραμμένο κώδικα για την εύκολη ανάπτυξη και χρήση τους μέσω του Arduino IDE. [13] Για τα libraries θα μιλήσουμε σε επόμενες ενότητες.

Οι πιο δημοφιλείς κατηγορίες αισθητήρων για το Arduino καταμετρούν θερμοκρασία και υγρασία (KY-015), πίεση και θερμοκρασία (BMP180 / BMP280), ήχο (KY-038), συγκέντρωση αερίων στο χώρο (MQ-2/MQ-7), απόσταση και υπερήχους (HC-SR04), RFID (MFRC-522), επιτάχυνση και περιστροφή

(MPU-6050), σήματα συχνοτήτων Bluetooth (HC-05 / HC-06), κίνηση μέσω υπέρυθρης ακτινοβολίας (PIR). [14]

2.5 Arduino IDE

Το IDE (Integrated Development Environment) είναι ένα λογισμικό περιβάλλον προγραμματισμού που παρέχει τις εργαλειοθήκες και τις λειτουργίες που απαιτούνται για την ανάπτυξη λογισμικού. Είναι ένας χώρος εργασίας όπου οποιοσδήποτε μπορεί να γράψει, μεταγλωττίσει, επεξεργαστεί και διαχειριστεί τον κώδικα που αφορά ένα συγκεκριμένο πρόγραμμα.

Ένα IDE συνήθως περιλαμβάνει έναν κειμενογράφο για την επεξεργασία του κώδικα, έναν μεταγλωττιστή για τη μετατροπή του κώδικα σε εκτελέσιμο πρόγραμμα, έναν αποσφαλματωτή (debugger) για τον εντοπισμό και τη διόρθωση σφαλμάτων, καθώς και διάφορα εργαλεία για τη διαχείριση του κώδικα, τη δημιουργία παραμέτρων και τη δοκιμή του προγράμματος. [15]

Η χρήση ενός IDE διευκολύνει τον προγραμματιστή, καθώς του παρέχει ένα συνολικό περιβάλλον για την ανάπτυξη και τη διαχείριση του κώδικα του προγράμματος. Το IDE συχνά προσφέρει επίσης λειτουργίες αυτόματης συμπλήρωσης κώδικα, έλεγχο σύνταξης, οπτική αναπαράσταση της δομής του προγράμματος και άλλες εργαλειοθήκες που βοηθούν στην αποτελεσματική ανάπτυξη του λογισμικού. [16]

Το Arduino IDE αποτελεί ένα ευέλικτο και εύχρηστο εργαλείο που παρέχει τη δυνατότητα να δημιουργηθεί και αποσταλεί εκτελέσιμος κώδικας στο Arduino. Μέσω του Arduino IDE, μπορούμε να συντάξουμε κώδικα λογισμικού σε γλώσσα προγραμματισμού βασισμένη στη C/C++, χρησιμοποιώντας τις βιβλιοθήκες και τις λειτουργίες που παρέχονται από την εταιρία Arduino.

Το περιβάλλον παρέχει επίσης δυνατότητες για την παρακολούθηση της κατάστασης της σύνδεσης, την επεξεργασία του κώδικα με βοήθεια του ενσωματωμένου κειμενογράφου, και την ανάγνωση/εμφάνιση των μηνυμάτων λάθους κατά τον προγραμματισμό. [17] Συνολικά, το Arduino IDE είναι το κύριο εργαλείο που χρησιμοποιείται από τους προγραμματιστές για να δημιουργήσουν και να φορτώσουν τον κώδικα στις πλακέτες Arduino.

2.6 Libraries

Στον τομέα της πληροφορικής, οι βιβλιοθήκες είναι συλλογές προγραμμάτων που περιέχουν έτοιμο κώδικα που μπορεί να χρησιμοποιηθεί για την επίλυση συγκεκριμένων προβλημάτων ή για την υλοποίηση συγκεκριμένων λειτουργιών. Ο κώδικας αυτός συνήθως παρέχεται δωρεάν από τους δημιουργούς/προγραμματιστές της κάθε βιβλιοθήκης και μπορεί να περιλαμβάνει συναρτήσεις, κλάσεις και άλλα αναγκαία στοιχεία για την επίλυση συγκεκριμένων προγραμματιστικών προβλημάτων. [18]

Οι βιβλιοθήκες καθιστούν τον προγραμματισμό πιο εύκολο και αποδοτικό, καθώς ο προγραμματιστής μπορεί να χρησιμοποιήσει αυτές τις έτοιμες λειτουργίες και δομές δεδομένων που έχουν ήδη υλοποιηθεί και έχουν δοκιμαστεί. Αυτό επιτρέπει στους προγραμματιστές να επικεντρωθούν στην ανάπτυξη του κύριου τούς προγράμματος, αντί να ανακαλύπτουν και υλοποιούν ξανά τις βασικές λειτουργίες της κάθε βιβλιοθήκης.

Για την δημιουργία έργων που έχουν βάση το Arduino υπάρχουν βιβλιοθήκες που δημιουργήθηκαν από τους κατασκευαστές αισθητήρων και περιφερειακών συσκευών ή γενικότερα από την Arduino

κοινότητα. Αυτές οι βιβλιοθήκες περιλαμβάνουν προγραμματιστικούς κώδικες που παρέχουν έτοιμες συναρτήσεις και δομές δεδομένων για να διευκολύνουν τον προγραμματισμό των Arduino. [19]

Οι βιβλιοθήκες παρέχουν έναν τρόπο για τον προγραμματιστή να επεκτείνει τις δυνατότητες του Arduino. Ο προγραμματιστής μπορεί να εισάγει αυτές τις βιβλιοθήκες στο πρόγραμμά του και να επαναχρησιμοποιήσει, τροποποιήσει η αναπτύξει αυτές, με σκοπό να επικοινωνήσει αποτελεσματικότερα με τις συνδεδεμένες συσκευές. Αυτό μειώνει τον χρόνο προγραμματισμού και καθιστά ευκολότερη την ανάπτυξη προτύπων εφαρμογών, προσφέροντας παράλληλα αξιοπιστία σε αυτό που παρέχουν.

Κεφάλαιο 3ο: Hardware

3.1 Εισαγωγή

Το ακόλουθο κεφάλαιο αφορά τον το κατασκευαστικό κομμάτι της εργασίας, και συγκεκριμένα αυτό που έχει να κάνει με την συνδεσμολογία του κυκλώματος του Arduino. Στον μικροελεγκτή Arduino θα συνδέσουμε ένα επιταχυνσιόμετρο (accelerometer), και ένα κουμπί. Για να γίνει αυτό εφικτό απαιτείται η χρήση μιας breadboard πλακέτας.

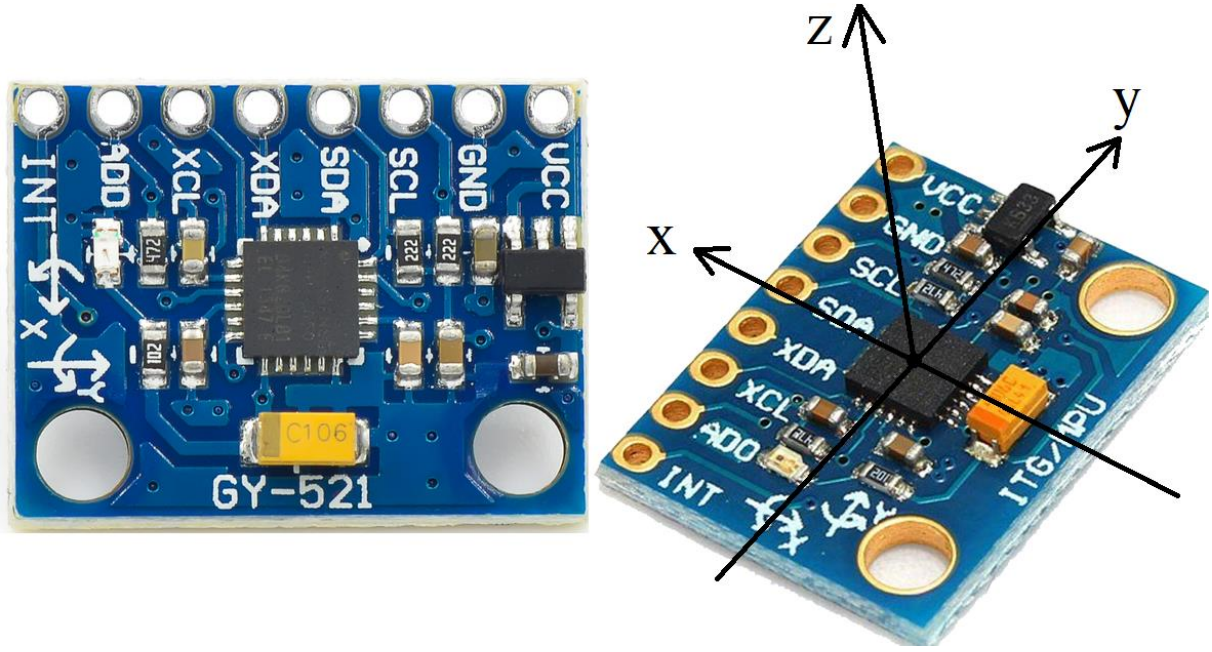
Για το κομμάτι που αφορά την 3D εκτύπωση που πραγματοποιήσαμε θα μιλήσουμε σε επόμενο κεφάλαιο.

3.2 Ο αισθητήρας MPU-6050

3.2.1 Γενικά

Ο αισθητήρας MPU-6050 περιλαμβάνει έναν επιταχυνσιομετρητή και ένα γυροσκόπιο τριών αξόνων σε ένα μοναδικό ενσωματωμένο κύκλωμα. Θεωρείται πολύ ακριβές, καθώς περιλαμβάνει μετατροπή αναλογικού σήματος σε ψηφιακό με τη χρήση 16 bits. Έτσι, μπορεί και καταγράφει ταυτόχρονα μετρήσεις και για τους τρεις άξονες x, y και z. [20]

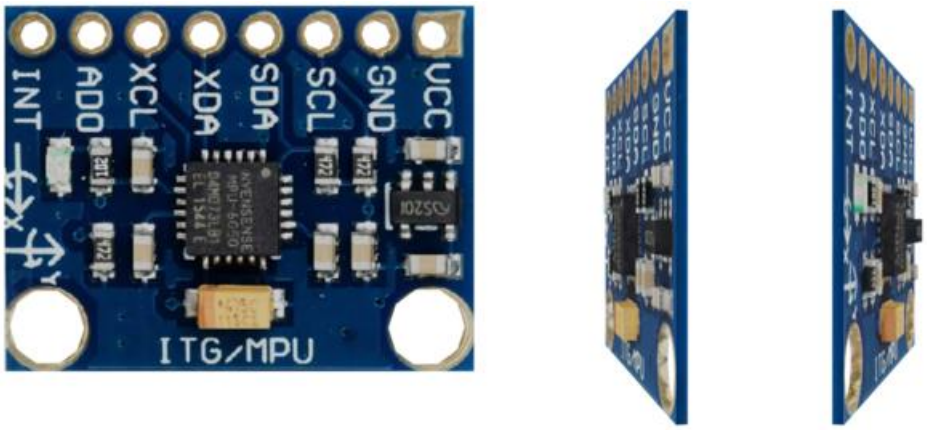
Αποτελεί ένα από τα πρώτα συστήματα Motion Tracking στον κόσμο, που σχεδιάστηκε με στόχο την χαμηλή κατανάλωση, το χαμηλό κόστος και τις υψηλές απαιτήσεις απόδοσης των έξυπνων τηλεφώνων, των tablet. Ο αισθητήρας χρησιμοποιεί το δίαυλο I2C για να συνδεθεί με το Arduino. [21]



Εικόνα 3.1: ο αισθητήρας MPU-6050 [22] [23]

Όταν περιστρέφουμε το MPU-6050, αυτό παράγει τιμές, ανάλογα με την διεύθυνση του άξονα που πραγματοποιήσαμε περιστροφή.

Όταν περιστρέφουμε το επιταχυνσιόμετρο όπως φαίνεται στην Εικόνα 3.2, τότε μεταβάλλουμε τις τιμές του άξονα Xx'. Η ελάχιστη και μέγιστη δυνατή τιμή είναι -1 και 1 αντίστοιχα, όταν πραγματοποιηθεί περιστροφή 90 μοιρών αριστερόστροφα και δεξιόστροφα.

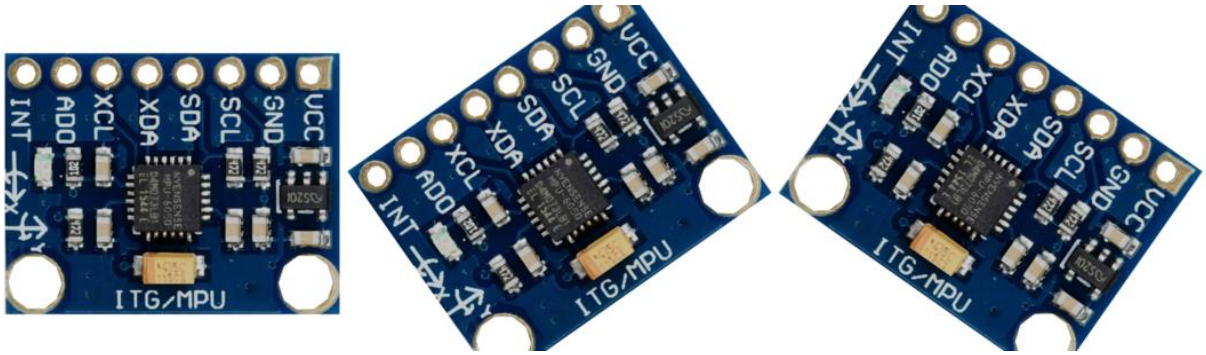


Εικόνα 3.2: Περιστροφή κατά τον άξονα Xx'

Όταν περιστρέφουμε το επιταχυνσιόμετρο όπως φαίνεται στην Εικόνα 3.3, τότε μεταβάλλουμε τις τιμές του άξονα Yy'. Η ελάχιστη και μέγιστη δυνατή τιμή είναι -1 και 1 αντίστοιχα, όταν πραγματοποιηθεί περιστροφή 90 μοιρών προς τα πάνω και προς τα κάτω.

Εικόνα 3.3: Περιστροφή κατά τον άξονα Yy'

Όταν περιστρέφουμε το επιταχυνσιόμετρο όπως φαίνεται στην Εικόνα 3.4, τότε μεταβάλλουμε τις τιμές του άξονα Zz'. Η ελάχιστη και μέγιστη δυνατή τιμή είναι -1 και 1 αντίστοιχα, όταν πραγματοποιηθεί περιστροφή 90 μοιρών αριστερόστροφα και δεξιόστροφα.



Εικόνα 3.4: Περιστροφή κατά τον άξονα Zz'

Οι Εικόνες 3.2-3.4 αποτελούν στιγμιότυπα οθόνης που παρήχθησαν με το λογισμικό δημιουργίας τρισδιάστατων γραφικών αντικειμένων που ονομάζεται Blender. [24] [25]

3.2.2 MPU-6050 Pins

Το MPU-6050 περιλαμβάνει 8 ακίδες (pins), τα οποία είναι τα εξής:

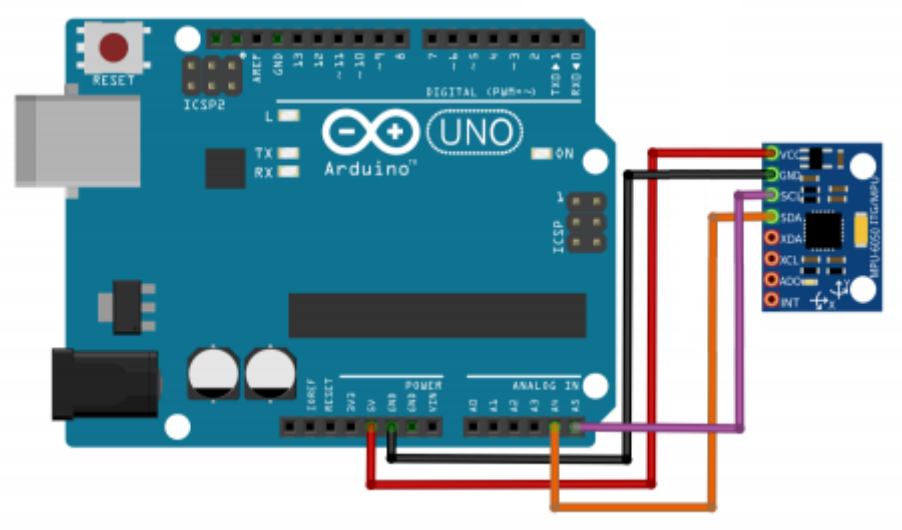
- VCC: pin τροφοδοσίας. Συνδέεται με την πηγή τάσης για την τροφοδοσία του MPU-6050.
- GND: pin γείωσης (Ground). Συνδέεται με το έδαφος ή την αρνητική πλευρά της τροφοδοσίας.
- SCL: pin ρολογιού I2C (Serial Clock). Χρησιμοποιείται για τον συγχρονισμό των μεταδόσεων δεδομένων μεταξύ του MPU-6050 και του μικροελεγκτή μέσω του πρωτοκόλλου I2C.
- SDA: pin δεδομένων του I2C (Serial Data). Χρησιμοποιείται για την ανταλλαγή δεδομένων μεταξύ του MPU-6050 και του μικροελεγκτή ή άλλων συσκευών μέσω του πρωτοκόλλου I2C.
- XDA: pin εξόδου δεδομένων (Auxiliary Serial Data). Μπορεί να χρησιμοποιηθεί για τη σύνδεση με άλλα I2C modules και η χρήση του είναι προαιρετική.
- XCL: pin εξόδου ρολογιού (Auxiliary Serial Clock). Μπορεί να χρησιμοποιηθεί για τη σύνδεση με άλλα I2C modules και η χρήση του είναι προαιρετική.
- AD0: pin επιλογής διεύθυνσης/εξόδου δεδομένων. Χρησιμοποιείται για την επιλογή της διεύθυνσης I2C του MPU-6050 και ως έξοδος δεδομένων για εξωτερική χρήση.
- INT: pin διακοπής (Interrupt). Χρησιμοποιείται για να υποδείξει ότι δεδομένα είναι διαθέσιμα για ανάγνωση από τον μικροελεγκτή. [26] [27]

Για τις ανάγκες της εργασίας μας, από αυτά τα pins, εμείς θα χρησιμοποιήσουμε τα πρώτα τέσσερα.

3.2.3 Συνδεσμολογία MPU-6050 με το Arduino

Για να πετύχουμε την ορθή σύνδεση μεταξύ του Arduino και του αισθητήρα MPU-6050, πρέπει να πραγματοποιήσουμε τις ακόλουθες συνδέσεις:

- Το VCC pin του MPU-6050 με το pin 5V του Arduino
- Το GND pin του MPU-6050 με το pin GND του Arduino
- Το SDA pin του MPU-6050 με το pin A4 (ή SDA) του Arduino
- Το SCL pin του MPU-6050 με το pin A5 (ή SCL) του Arduino [21]

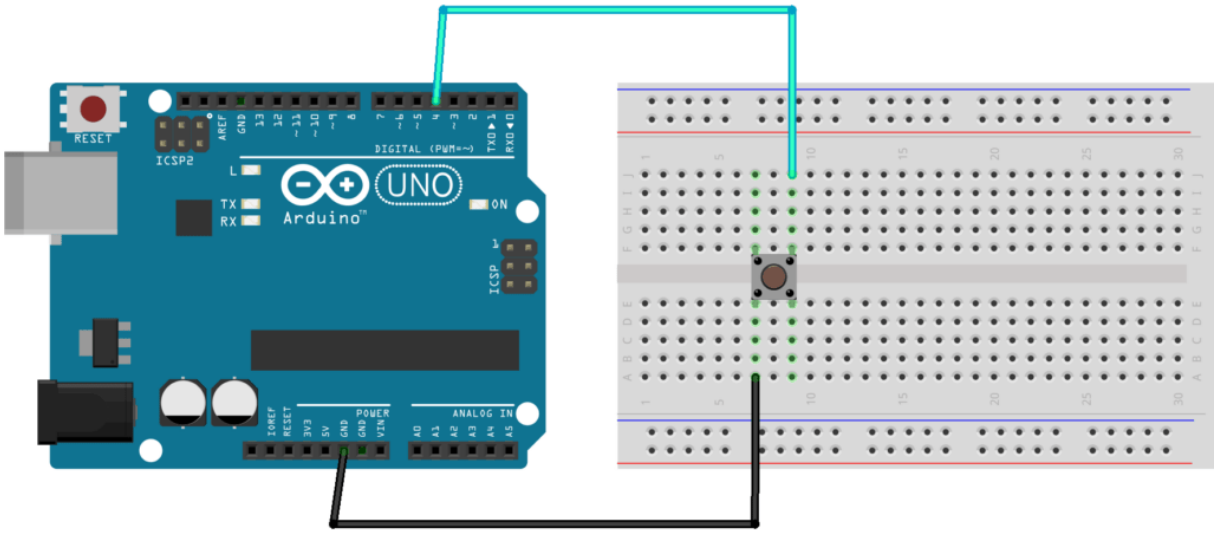


Εικόνα 3.5: Συνδεσμολογία Arduino - MPU-6050 [21]

3.3 Συνδεσμολογία κουμπιού με το Arduino

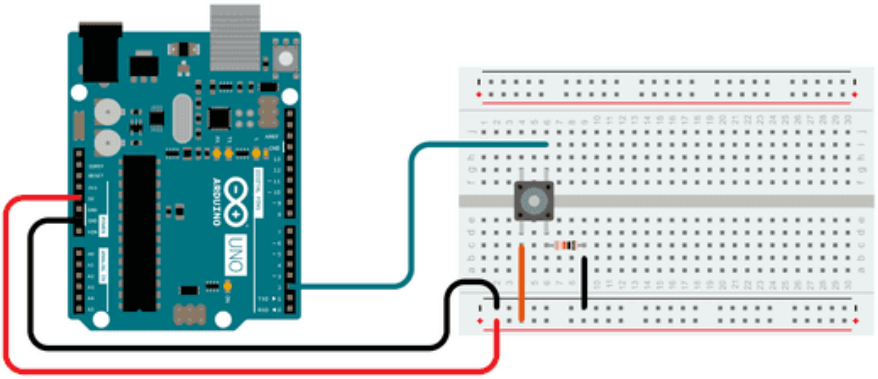
Για την συγκεκριμένη εργασία θεωρήσαμε απαραίτητη την προσθήκη ενός κουμπιού για την πιο αποτελεσματική χρήση του λογισμικού που δημιουργήσαμε. Η συγκεκριμένη σύνδεση μπορεί να επιτευχθεί και χωρίς τη χρήση breadboard. [28]

Για την επιτυχή σύνδεση πρέπει να συνδέσουμε το ένα άκρο του κουμπιού σε οποιοδήποτε ψηφιακό ακροδέκτη εισόδου στο Arduino (π.χ., ακροδέκτης 4), και το άλλο άκρο του κουμπιού στον ακροδέκτη γείωσης (GND) στο Arduino, όπως φαίνεται στην Εικόνα 3.2.



Εικόνα 3.6: Συνδεσμολογία Arduino – Push Button [29]

Αξίζει να σημειωθεί πως η σύνδεση του κουμπιού απευθείας με το Arduino δεν προτείνεται, και πως συνίσταται η χρήση κάποιας αντίστασης (resistor).

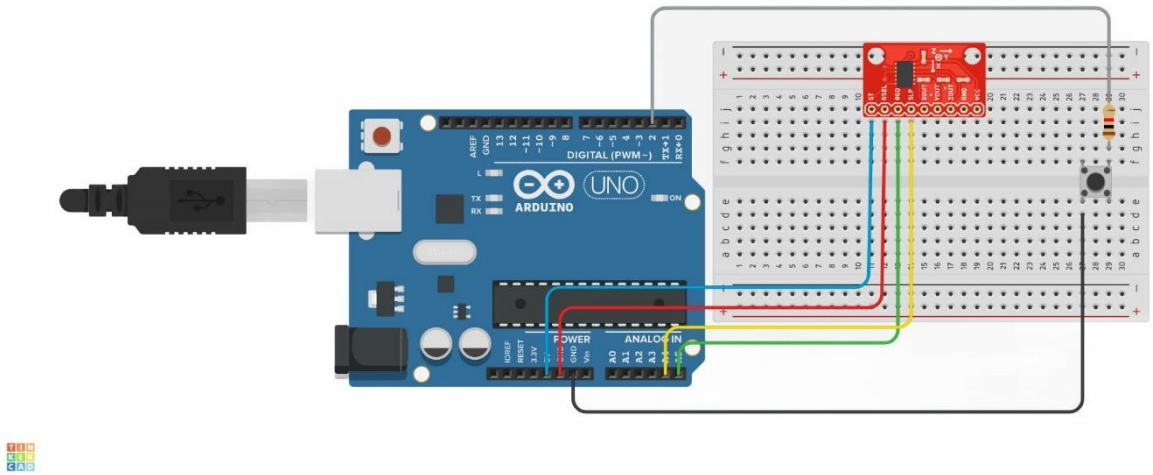


Εικόνα 3.7: Συνδεσμολογία Arduino – Push Button με τη χρήση resistor [30]

3.4 Συνδεσμολογία ταυτόχρονης σύνδεσης

Η τελική κατασκευή απαιτεί την ταυτόχρονη σύνδεση του αισθητήρα MPU-6050 και του κουμπιού, οπότε η χρήση ενός breadboard είναι απαραίτητη.

Όπως αναφέρθηκε και στην ενότητα 3.2.3 πραγματοποιούνται οι συνδέσεις VCC - 5V, GND – GND, SDA - A4, SCL - A5. Παρακάτω, όπως φαίνεται και στην Εικόνα 3.4 έχει συνδεθεί μαζί με το MPU-6050, επιπλέον ένα κουμπί, χρησιμοποιώντας την κατάλληλη αντίσταση στο ένα από τα δύο άκρα του κουμπιού, ώστε να εγγυηθούμε ότι δεν θα δημιουργηθεί κάποια πιθανή βλάβη, όταν το κουμπί δεν χρησιμοποιείται.



Εικόνα 3.8: Κύκλωμα εργασίας [31]

Κεφάλαιο 4ο: Software

4.1 Γενικά

Στο συγκεκριμένο κεφάλαιο μεταβαίνουμε από το σκέλος της κατασκευής, στο σκέλος που αφορά το προγραμματιστικό κομμάτι. Το λογισμικό της εργασίας χωρίζεται σε 2 σκέλη. Το 1ο αφορά το Arduino και την αλληλεπίδραση του με τον αισθητήρα MPU-6050 (προγραμματισμός Arduino IDE). Το 2ο αφορά την μετέπειτα επεξεργασία των δεδομένων που στέλνει το Arduino, τα οποία λαμβάνει ένα λογισμικό που ονομάζεται MAX/MSP. Για το MAX/MSP θα μιλήσουμε αναλυτικά στο επόμενο κεφάλαιο

Η συγγραφή του κώδικα που ακολουθεί, αποτελεί επεξεργασία κώδικα, τον οποίο λάβαμε από libraries που παρέχονται δωρεάν στο διαδίκτυο.

4.2 Arduino Code

4.2.1 Accelerometer

Ο κώδικας που φαίνεται στην Εικόνα 4.1 αποτελεί επεξεργασία του κώδικα της βιβλιοθήκης που έχει συντάξει ο χρήστης Electronic Cats, ο οποίος διατίθεται δωρεάν στο επίσημο site του Arduino. [32] [33] Συνοπτικά ο κώδικας διαβάζει τις τιμές των τριών αξόνων x, y και z από τον αισθητήρα MPU-6050 και τις εκτυπώνει στη σειριακή οθόνη κάθε 200 milliseconds.

```
#include <basicMPU6050.h>
basicMPU6050<> imu;

void setup(){
  imu.setup();
  imu.setBias();
  Serial.begin(38400);
}

void loop() {
  imu.updateBias();

  Serial.print("      x: "); //print x values
  Serial.print( imu.ax() );
  Serial.print(" ");
  Serial.print("      y: "); //print y values
  Serial.print( imu.ay() );
  Serial.print(" ");
  Serial.print("      z: "); //print z values
  Serial.print( imu.az() );
  Serial.println();

  delay(200);

  }
```

```
01:03:44.099 ->   x: 0.51   y: -0.17    z: 0.86
01:03:44.271 ->   x: 0.52   y: 0.13    z: 0.76
01:03:44.517 ->   x: 0.33   y: 0.42    z: 0.75
01:03:44.703 ->   x: 0.00   y: 0.47    z: 0.73
01:03:44.908 ->   x: -0.26   y: 0.25    z: 0.86
01:03:45.128 ->   x: -0.49   y: -0.51    z: 0.63
01:03:45.346 ->   x: -0.37   y: -0.86    z: 0.30
01:03:45.532 ->   x: -0.16   y: -0.92    z: 0.37
01:03:45.750 ->   x: 0.13   y: -0.79    z: 0.57
01:03:45.921 ->   x: 0.46   y: -0.33    z: 0.79
01:03:46.152 ->   x: 0.52   y: 0.19    z: 0.82
01:03:46.369 ->   x: 0.29   y: 0.41    z: 0.85
01:03:46.555 ->   x: -0.32   y: 0.07    z: 0.90
01:03:46.775 ->   x: -0.39   y: -0.55    z: 0.72
```

Εικόνα 4.1: Κώδικας ανίχνευσης τιμών του MPU-6050 – Ενδεικτικό output στη σειριακή οθόνη.

Αυτός ο κώδικας χρησιμοποιεί τη βιβλιοθήκη basicMPU6050 για να επικοινωνήσει με έναν αισθητήρα κίνησης MPU-6050 και μετέπειτα δημιουργούν ένα αντικείμενο “imu” τύπου basicMPU6050. Η

συνάρτηση setup εκτελείται όταν τροφοδοτείται το Arduino με ρεύμα. Αυτή αρχικά θέτει τον αισθητήρα MPU-6050 σε μία αρχική κατάσταση και ορίζει τις τιμές βάσει των οποίων θα μετρά την παραμόρφωση κατά την εκτέλεση του προγράμματος. Τέλος αρχικοποιεί την σειριακή επικοινωνία με ταχύτητα 38400 bits/sec.

Η συνάρτηση loop εκτελείται αυτόματα μετά την setup συνάρτηση και επαναλαμβάνεται η εκτέλεση της μέχρι την απενεργοποίηση του προγράμματος. Αυτή ενημερώνει συνεχώς τις τιμές παραμόρφωσης του αισθητήρα. Στη συνέχεια ακολουθεί μια σειρά εντολών print και println, οι οποίες έχουν συνταχθεί με τρόπο ώστε να εμφανίζονται με όσο το δυνατόν πιο ευανάγνωστο και κατανοητό τρόπο, οι τιμές των τριών αξόνων x, y και z του αισθητήρα MPU-6050 στη σειριακή οθόνη.

Τέλος προκαλείται μια καθυστέρηση που διαρκεί 200 ms μέχρι να ξεκινήσει η επόμενη επανάληψη του loop. Αξίζει εδώ να σημειωθεί ότι η τιμή των 200 ms δεν αντιπροσωπεύει τιμή εφαρμογών «πραγματικού χρόνου». Έχει δοθεί τέτοιος αριθμός ώστε ο χρήστης που βλέπει την εμφάνιση στην οθόνη να μπορεί να διαβάσει εύκολα το αποτέλεσμα στην οθόνη. Για ανάγνωση των αποτελεσμάτων συνίσταται ακόμα μεγαλύτερη τιμή, σε βαθμό που να αγγίζει και τα δύο δευτερόλεπτα. Στο τελικό πρόγραμμα ωστόσο ο χρόνος έχει ρυθμιστεί στα 10 ms ώστε να πλησιάζει κατά πολύ σε συνθήκες πραγματικού χρόνου.

4.2.2 Button

Ο κώδικας που φαίνεται στην Εικόνα 4.2 αποτελεί επεξεργασία του κώδικα της βιβλιοθήκης που έχει συντάξει ο Michael Adams, ο οποίος διατίθεται δωρεάν στο επίσημο site του Arduino. [34] [35] Συνοπτικά αυτό που κάνει είναι να διαβάζει την κατάσταση ενός κουμπιού που είναι συνδεδεμένο και ανιχνεύει όταν αυτό πατιέται. Όταν το κουμπί πατιέται, εκτυπώνει ένα μήνυμα στη σειριακή οθόνη.

Στην αρχή αρχικοποιούνται δύο τιμές, οι οποίες αναπαριστούν τις δύο διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί το κουμπί. Συγκεκριμένα η μεταβλητή BUTTON αναπαριστά την τρέχουσα κατάσταση του κουμπιού, ενώ η μεταβλητή «exBUTTON» την προηγούμενη. Η συνάρτηση setup έχει τον ρόλο να ρυθμίζει το pin 2 ως είσοδο με ενεργοποιημένη αντίσταση. Το κουμπί που θέλουμε να ενεργοποιήσουμε πρέπει δηλαδή να είναι συνδεδεμένο στο συγκεκριμένο pin. Στην συνέχεια ενεργοποιούμε την σειριακή επικοινωνία με ταχύτητα 9600 bits/sec.

Στη συνάρτηση loop, διαβάζουμε την τρέχουσα κατάσταση του pin 2 (οπού είναι συνδεδεμένο το κουμπί) και στην συνέχεια το αποθηκεύει στη μεταβλητή «BUTTON». Ακολουθεί μια δομή if στην οποία ελέγχεται αν το κουμπί είναι πατημένο ή όχι. Η πρώτη δομή if ελέγχει εάν το κουμπί βρίσκεται στην κατάσταση HIGH (μη πατημένο), όπου αν είναι αληθής, τότε ορίζει την «exBUTTON» σε 1, υποδηλώνοντας ότι το κουμπί ήταν προηγουμένως πατημένο. Η δεύτερη δομή if ελέγχει εάν το κουμπί βρίσκεται στην κατάσταση LOW (πατημένο) και αν η «exBUTTON» είναι ίση με 1. Αν και οι δύο συνθήκες είναι αληθείς, τότε εκτελείται ο κώδικας μέσα στη δομή if (Εικόνα 4.2 , γραμμή 15»).

Εκεί εκτυπώνεται το μήνυμα «Hello!» και το «exBUTTON» γίνεται πλέον 0, υποδηλώνοντας ότι το κουμπί δεν θεωρείται προηγούμενα πατημένο. Στη συνέχεια προκαλείται μια μικρή καθυστέρηση 50 ms πριν ξεκινήσει να λειτουργεί η συνάρτηση loop από την αρχή.

```
int BUTTON;
int exBUTTON;

void setup() {
  pinMode(2, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  BUTTON = digitalRead(2);
  if (BUTTON == HIGH) {
    exBUTTON = 1;
  }
  if (BUTTON == LOW and exBUTTON == 1) {
    Serial.println("Hello!");
    exBUTTON = 0;}
  delay(50);
}
```

Εικόνα 4.2: Κώδικας ενεργοποίησης κουμπιού

Η χρήση της μεταβλητής «exBUTTON» βοηθά στο να διασφαλίζεται ότι η πίεση του κουμπιού ανιχνεύεται μόνο μία φορά ανά πάτημα, αποφεύγοντας την επαναλαμβανόμενη ανίχνευση κατά τη διάρκεια πατημένου κουμπιού.

4.2.3 Final Code

Ο τελικός κώδικάς Arduino που συντάξαμε αποτελεί μια μίξη των δύο παραπάνω προγραμμάτων. Γενικά ο κώδικας διαβάζει τις τιμές των αξόνων x και y από τον αισθητήρα MPU6050 και τις εμφανίζει στη σειριακή οθόνη. Επίσης, ανιχνεύει την κατάσταση ενός κουμπιού και εμφανίζει μήνυμα στην οθόνη όταν το κουμπί πατηθεί.

Όπως και πριν, με τη χρήση της βιβλιοθήκη basicMPU6050 δημιουργούμε ένα αντικείμενο imu τύπου basicMPU6050. Αρχικοποιείται ο αισθητήρας MPU-6050, ρυθμίζονται οι τιμές της αρχικής παραμόρφωσης του, ενώ ταυτόχρονα ορίζεται το pin 2 ως είσοδος. Η σειριακή επικοινωνία αρχικοποιείται με ταχύτητα 38400 bits/sec, και οι τιμές παραμόρφωσης του αισθητήρα ενημερώνονται συνεχώς.

Οι διαφορές στο πρόγραμμα ξεκινούν εδώ. Όπως και στο πρόγραμμα της ενότητας 4.2.1. εμφανίζεται μήνυμα στην οθόνη. Στη συγκεκριμένη περίπτωση όμως, οι εντολές εμφάνισης χρησιμοποιούνται για να εμφανίσουν τις τιμές των αξόνων x και y (όχι z) στη σειριακή οθόνη, χωρίς να συνοδεύεται από κείμενο. Οι τιμές των αξόνων χωρίζονται με ένα κενό διάστημα και στη συνέχεια προστίθεται μια κενή γραμμή. Η διαδικασία αυτή τη φορά πραγματοποιείται με delay 10 ms, γεγονός που κάνει το αποτέλεσμα της σειριακής οθόνης να εναλλάσσεται γρήγορα και να είναι ταυτόχρονα σχετικά δυσανάγνωστο για τον χρήστη.

Η συγκεκριμένη εφαρμογή εμφανίζει το αποτέλεσμα με τρόπο που δεν είναι φιλικό προς το χρήση. Είναι φτιαγμένη όμως με τρόπο, που τα δεδομένα είναι δοσμένα με τέτοιο τρόπο, ώστε να μπορούν να ληφθούν από το πρόγραμμα MAX/MSP (γεγονός που θα συζητήσουμε αναλυτικά στο κεφάλαιο 6).

Ταυτόχρονα με την εμφάνιση στην οθόνη εκτελείται και μια δομή if, η οποία ελέγχει την κατάσταση του pin 2. Αν το pin αυτό είναι σε χαμηλή κατάσταση (κουμπί πατημένο), τότε εκτυπώνεται το γράμμα «A» στη σειριακή οθόνη. Ακολουθεί μια καθυστέρηση 500 ms για να αποφευχθεί η επαναλαμβανόμενη ανίχνευση κατά την πατημένη κατάσταση του κουμπιού. Αυτή η τιμή αποστέλλεται επίσης στο MAX/MSP. Η όλη διαδικασία επαναλαμβάνεται έως ότου κλείσει η εφαρμογή.

Στην ακόλουθη Εικόνα 4.3 βλέπουμε τον κώδικα της συγκεκριμένης εφαρμογής καθώς και ένα ενδεικτικό αποτέλεσμα της σειριακής οθόνης

```
#include <basicMPU6050.h>
basicMPU6050<> imu;

void setup(){
  imu.setup();
  imu.setBias();
  pinMode(2, INPUT_PULLUP);
  Serial.begin(38400);
}

void loop() {
  imu.updateBias();

  Serial.print( imu.ax() );
  Serial.print(" ");
  Serial.print( imu.ay() );
  Serial.print(" ");
  Serial.println();\
  delay(10);

    if (digitalRead(2) == LOW) {
    Serial.println("A");
    delay(500);
    }
  }
```

```
19:32:50.907 -> 0.16 -0.35
19:32:50.907 -> 0.21 -0.29
19:32:50.940 -> 0.25 -0.19
19:32:50.940 -> 0.25 -0.15
19:32:50.940 -> 0.23 -0.31
19:32:50.940 -> 0.16 -0.41
19:32:50.972 -> 0.12 -0.36
19:32:51.006 -> 0.15 -0.27
19:32:51.006 -> 0.24 -0.06
19:32:51.006 -> 0.30 0.10
19:32:51.006 -> 0.26 0.14
```

Εικόνα 4.3: Τελική έκδοση του κώδικα που χρησιμοποιείται για το music-box

Κεφάλαιο 5ο: MAX/MSP

5.1 Γενικά

Το MAX/MSP είναι ένα περιβάλλον προγραμματισμού που χρησιμοποιείται για τη δημιουργία και την ανάπτυξη διαδραστικών ήχων, μουσικής και πολυμέσων εφαρμογών. Αναπτύχθηκε από την εταιρεία Cycling '74 και παρέχει ένα γραφικό περιβάλλον που επιτρέπει στους χρήστες να δημιουργήσουν προγράμματα, γνωστά ως patches. [36] Μέσω της γραφικής διεπαφής που παρέχει το πρόγραμμα, οι χρήστες μπορούν να συνδέσουν αντικείμενα μεταξύ τους, ορίζοντας τη ροή των δεδομένων και των εντολών.

Οι χρήστες του MAX/MSP μπορούν να δημιουργήσουν προγράμματα που να παράγουν οπτικά αποτελέσματα, όπως γραφήματα, σχήματα, και πολυμέσα (εικόνα και βίντεο). Μπορούν να κάνουν πολύ απλά πράγματα όπως να αναπαράγουν ένα βίντεο, ή να επιλέξουν με τι ρυθμό θα αναπαράγεται αυτό το βίντεο. Μπορούν επίσης αλληλοεπιδρούν με διάφορες πηγές εισόδου, όπως κάμερα ή μικρόφωνο, μουσικά όργανα, αισθητήρες κίνησης και άλλες συσκευές, και να ελέγχουν αυτά τα πολυμέσα. Ανάλογα με την έμπνευση και τη δημιουργικότητα του καθενός μπορούν να παραχθούν πολλά και διαφορετικά αποτελέσματα.

Το MAX/MSP αποτελεί μια αξιόλογη εφαρμογή επειδή μας επιτρέπει να δημιουργούμε προγράμματα και να καθορίζουμε τη λειτουργικότητά τους μέσω γραφικής αναπαράστασης. Αντί να χρησιμοποιούμε γλώσσα προγραμματισμού με κώδικα για να περιγράψουμε τις λειτουργίες της εφαρμογής, το MAX/MSP μας επιτρέπει να χτίζουμε τα προγράμματά χρησιμοποιώντας τη γραφική διεπαφή. Κάθε αντικείμενο που δημιουργούμε αντιπροσωπεύει ένα τμήμα λειτουργικότητας και εκτελεί κάποια ενέργεια ή παρέχει κάποια λειτουργία.

Κάθε αντικείμενο έχει (συνήθως), τουλάχιστον μια υποδοχή εισόδου (inlet), και μια υποδοχή εξόδου (outlet), μέσω των οποίων μπορούμε να περάσουμε μεταβλητές, η να συνδέσουμε τα αντικείμενα μεταξύ τους. Συνδέοντας αυτά μεταξύ τους, καθορίζουμε τη ροή των δεδομένων και τις ενέργειες που θέλουμε να εκτελεστούν. Αυτό το γραφικό προγραμματιστικό περιβάλλον επιτρέπει σε μη προγραμματιστές ή σε ανθρώπους με περιορισμένες γνώσεις προγραμματισμού να δημιουργήσουν προγράμματα μέσω οπτικής αναπαράστασης και σύνδεσης αντικειμένων.

5.2 Patches, Patchers, Subpatch

Οι όροι patch και patcher και subpatch αναφέρονται σε διάφορα στοιχεία του οπτικού περιβάλλοντος του MAX/MSP. Ένα **patch** αναφέρεται σε ένα ατομικό αρχείο στο MAX/MSP. Είναι μια αυτόνομη αρχείο που περιέχει οπτικά αντικείμενα, γνωστά ως «boxes» (κουτιά), συνδεδεμένα με patch cords (καλώδια σύνδεσης). Ένα patch αντιπροσωπεύει ένα συγκεκριμένο κομμάτι λειτουργικότητας ή μια μουσική σύνθεση που δημιουργείται με το MAX/MSP. [37]

Ένας διαφορετικός όρος που χρησιμοποιείται είναι ο patcher, ο οποίος μπορεί να χρησιμοποιηθεί με διαφορετική σημασία την κάθε φορά, ανάλογα τη χρήση του και των συμφραζόμενων. **Patcher** μπορεί να χαρακτηριστεί ο συνδυασμός δύο διαφορετικών patches που αλληλοεπιδρούν μεταξύ τους και μπορεί να υπάρχουν (όχι απαραίτητα) στο ίδιο αρχείο. Ένα patcher είναι ουσιαστικά ένας χώρος εργασίας μέσα στον οποίο μπορούμε να δημιουργούμε και να οργανώνουμε διάφορα patch. Μπορούμε να το φανταστούμε σαν ένα καμβά που πάνω τοποθετούμε patches. Έτσι, η διαφορά μεταξύ «patch»

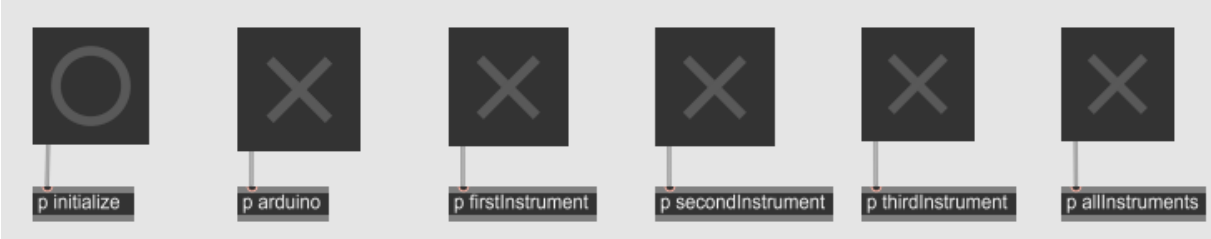
και «patcher» είναι ότι ένα patcher μπορούμε να το φανταστούμε σαν ένα περιβάλλον, στο οποίο μπορούμε να δημιουργούμε και επεξεργαζόμαστε τα patches.

Στο MAX/MSP, ο όρος «patcher» χρησιμοποιείται για να υπογραμμίσει την ιδέα ότι δεν δημιουργούμε απλά απομονωμένα patches, αλλά εργαζόμαστε εντός ενός μεγαλύτερου περιβάλλοντος που μας επιτρέπει να δημιουργούμε και να οργανώνουμε πολλαπλά patches μαζί για την κατασκευή πολύπλοκων συστημάτων ή συνθέσεων. Ένα patch που χρησιμοποιείται πολλές φορές μέσα σε έναν patcher δεν απαιτείται να γραφτεί περισσότερες από μια φορές, καθώς το MAX/MSP επιτρέπει επαναχρησιμοποίηση κώδικα.

Τέλος αξίζει να αναφερθεί ο όρος subpatch. Ένα **subpatch** είναι ένα αυτοτελές εσωτερικό patch που αποτελεί μέρος ενός μεγαλύτερου patcher. Συνήθως ενθυλακώνουμε συγκεκριμένα patches, μετατρέποντας τα σε subpatches. Με αυτό τον τρόπο μπορούμε να οργανώσουμε πιο αποδοτικά και να δομήσουμε ένα μεγαλύτερου patch. Τα subpatches διευκολύνουν στην αναγνωσιμότητα και τη συντήρηση του κώδικα, καθώς μας επιτρέπουν να επαναχρησιμοποιήσουμε και να οργανώσουμε τα διάφορα patches ανεξάρτητα από το μεγάλο patcher. Αυτό είναι ιδιαίτερα σημαντικό, ειδικά όταν ασχολούμαστε με μεγαλύτερα και πιο πολύπλοκα έργα.

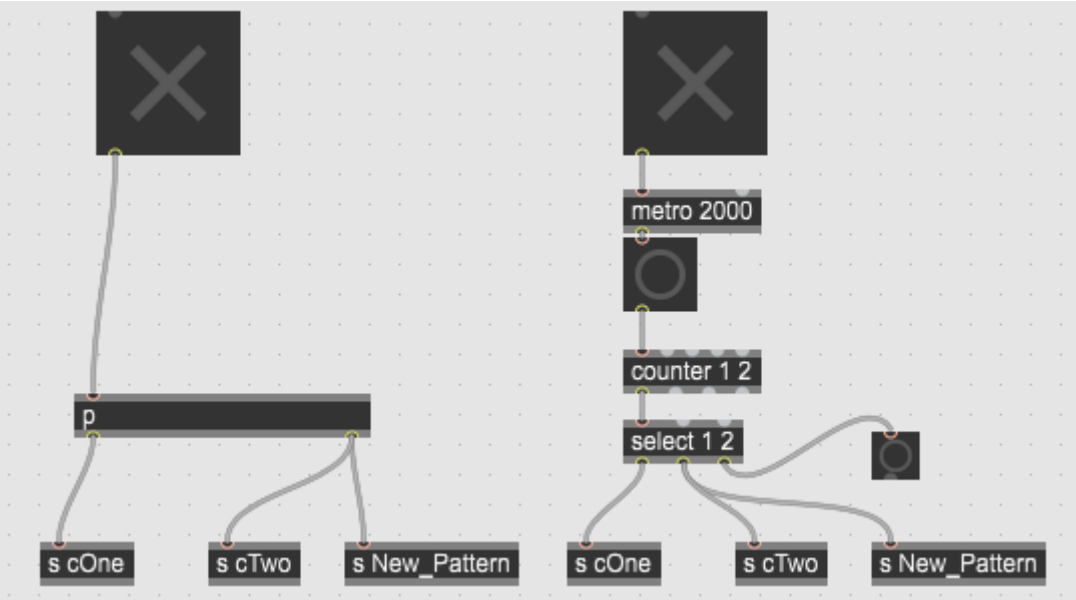
Ένα patch θεωρείται subpatch ενός patcher, αν εκτελεί κάποια υπολειτουργία αυτού μέσα σε αυτόν. Για τον χρήστη αυτό μπορεί να μην είναι άμεσα εμφανές. Για να κάνουμε πιο ευδιάκριτο το χαρακτηριστικό που περιγράψαμε και να «μετατρέψουμε» το κομμάτι κώδικα σε subpatch, επιλέγουμε τα αντικείμενα που θέλουμε να οργανώσουμε, κάνουμε δεξί κλικ και επιλέγουμε τη λειτουργία «group objects» (εναλλακτικά πιέζουμε τα πλήκτρα ctrl + g). Το αποτέλεσμα φαίνεται καθαρά στην Εικόνα 6.6, οπού βλέπουμε ότι το patch περικλείεται από ένα κουτί.

Αυτό γίνεται για να είναι πιο ευανάγνωστος ο Patcher. Στην Εικόνα 5.1 βλέπουμε ένα στιγμιότυπο από τον Main Patcher που αποτελεί την βασική εφαρμογή που δημιουργήσαμε. Σε αυτή την εικόνα βλέπουμε ενδεικτικά πως φαίνονται οι subpatchers από την οπτική του κεντρικού Patcher. Αν κάνουμε κλικ πάνω στον patcher δύο φορές αυτός θα ανοίξει και θα βλέπουμε αναλυτικά το περιεχόμενό του.



Εικόνα 5.1: Subpatchers view from Patcher perspective

Εμφώλευση μπορεί να γίνει και σε ένα συγκεκριμένο κομμάτι ενός patch, κάτι που γίνεται για τους ίδιους λόγους. Ένα τέτοιο παράδειγμα φαίνεται στην εικόνα που ακολουθεί. Encapsulation/Decapsulation μπορεί να πραγματοποιηθεί επιλέγοντας τα αντικείμενα που θέλουμε να εμφωλεύσουμε και πιέζοντας τα πλήκτρα (ctrl + shift + e για εμφώλευση, ή ctrl + shift + d για από-εμφώλευση).



Εικόνα 5.2: Encapsulated vs De-encapsulated patch

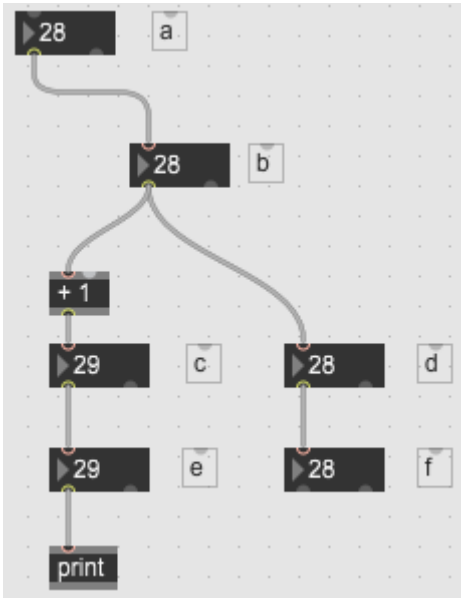
Κλείνοντας, αξίζει να σημειώσουμε ότι όλοι οι παραπάνω όροι θεωρούνται σωστοί. Ένα patch μπορεί να χαρακτηριστεί patcher, η subpatch, ανάλογα με την σημασία και τα συμφραζόμενα που υπάρχουν στην κάθε πρόταση. Συνήθως όταν αναφερόμαστε σε ένα patcher, αναφερόμαστε στο σύνολο μιας εφαρμογής. Ωστόσο ένας patcher μπορεί να αποτελεί subpatch ενός μεγαλύτερου patcher που πραγματοποιεί μεγαλύτερη πιο σύνθετη λειτουργία από αυτόν.

5.3 Patch Examples

Στη συγκεκριμένη ενότητα θα αναλύσουμε τη λειτουργικότητα διάφορων patches, για να καταλάβουμε κάποια βασικά χαρακτηριστικά που αφορούν τη λειτουργία του προγράμματος MAX/MSP, και εν τέλει να μπορέσουμε να κατανοήσουμε τη λειτουργικότητα του τελικού patcher που δημιουργήσαμε.

5.3.1 Cords, mathematical operations and print

Στο ακόλουθο παράδειγμα που φαίνεται στην Εικόνα 5.3, έχουμε την δυνατότητα να επεξεργαστούμε την τιμή του αντικειμένου «a», το οποίο αναπαριστά έναν ακέραιο. Αυτό στέλνει την τιμή του στον επόμενο ακέραιο «b», που συνδέονται μεταξύ τους με ένα cord. Στην συνέχεια αποστέλλεται η τιμή στα αντίστοιχα πεδία «c» και «d». Στην περίπτωση του «c» η τιμή έχει αυξηθεί κατά μια μονάδα. Μετέπειτα αποστέλλονται οι τιμές από τα «c» και «d», στα αντίστοιχα «e» και «f», ενώ τέλος εμφανίζεται στην κονσόλα η τιμή του «e» με τη χρήση του αντικειμένου «print». Η διαδικασία επαναλαμβάνεται κάθε φορά που ο χρήστης μεταβάλει την τιμή του «a».



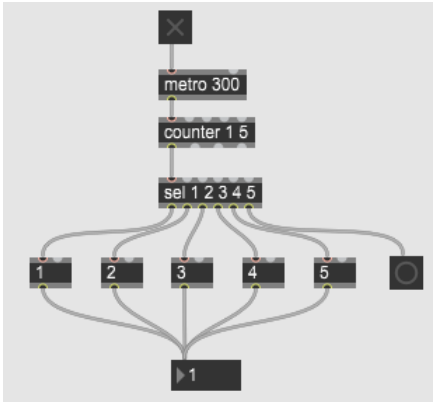
Εικόνα 5.3: Patch που δείχνει εκτέλεση μαθηματικών πράξεων, και εμφάνιση αυτών.

5.3.2 Toggle, Metro

Το ακόλουθο patch πραγματοποιεί την λειτουργία όταν το αντικείμενο toggle είναι ενεργοποιημένο, να στέλνει σειριακά τα μηνύματα 1-5 στο αντικείμενο integer που βρίσκεται στο τέλος του patch.

Στο ακόλουθο παράδειγμα (Εικόνα 5.4), έχουμε τη δυνατότητα να ενεργοποιήσουμε το toggle. Όταν το toggle είναι απενεργοποιημένο δεν εκτελεί τίποτα. Όταν όμως είναι ενεργοποιημένο στέλνει μηνύματα «1».

Στο ακόλουθο παράδειγμα συγκεκριμένα, όταν το toggle είναι ενεργοποιημένο, εκτελείται η μια διαδικασία τύπου «for» (με τη χρήση του αντικειμένου counter) που μετρά σειριακά από το ένα μέχρι το πέντε, κάθε 300 ms (metro 300). Σε κάθε επανάληψη αποστέλλεται «bang» μήνυμα από το αντίστοιχο outlet του selector (αν και μόνο η τιμή που λάβει από τον counter υπάρχει στον selector). Στην περίπτωση που λάβει τιμή που δεν υπάρχει μέσα στον selector, τότε στέλνεται μήνυμα «bang» από το τελευταίο outlet του selector.

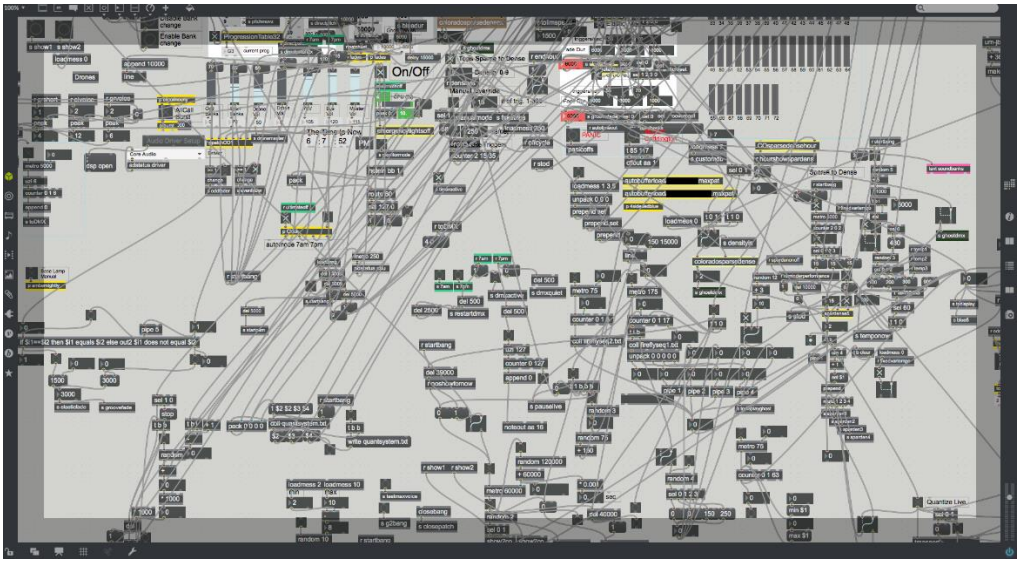


Εικόνα 5.4: Patch που δείχνει τη λειτουργία του metro.

Το μήνυμα «bang» που αποστέλλεται από τον selector, πηγαίνει στο ανάλογο inlet των ακόλουθων αριθμητικών αντικειμένων. Αυτά τα αντικείμενα με τη σειρά τους αποστέλλουν το περιεχόμενο τους στο integer αντικείμενο που φαίνεται στο τέλος της Εικόνας 5.4. Αυτό αλλάζει το περιεχόμενο του ανάλογα με ποια είναι η τελευταία τιμή που έχει λάβει από τα integer αντικείμενα που βρίσκονται πάνω από αυτό.

5.3.3 Receive / Send

Το ακόλουθο patch είναι απλό και κατανοητό και πιο εύκολο στη δημιουργία του, ωστόσο η χρησιμότητα του είναι πολύ μεγάλη. Τα receive και send μηνύματα υπάρχουν για να αποφευχθούν οι πολύ μεγάλοι patchers που είναι δυσνόητοι και καθιστούν την επεξεργασία τους και την επεκτασιμότητα τους δύσκολη. Ένας τέτοιος patcher φαίνεται στην ακόλουθη εικόνα.

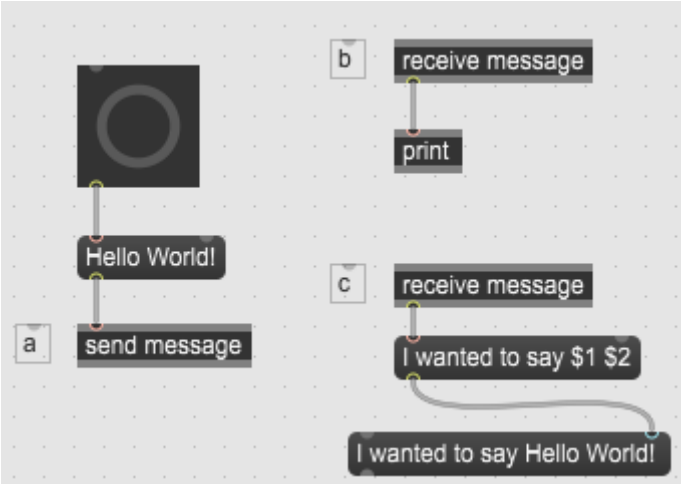


Εικόνα 5.5: Παράδειγμα προς αποφυγή patcher, που δεν χρησιμοποιεί τα κατάλληλα receive/send μηνύματα. [38]

Το ακόλουθο patch έχει σαν ρόλο να λαμβάνει το μήνυμα «Hello World», και να το στέλνει σε δύο σημεία ταυτόχρονα.

Αυτό εκτελείται όταν ο χρήστης πιέζει το bang. Αυτό διαβάζει το μήνυμα «Hello World!», το οποίο και αποστέλλει χρησιμοποιώντας το αντικείμενο «send». Η λέξη message στα αντικείμενα send και receive πρέπει να είναι επιλεγμένη κατάλληλα, ώστε ο χρήστης να ξέρει από ποια σημεία στέλνει και σε ποια σημεία λαμβάνει αυτά τα μηνύματα.

Ένα μήνυμα μπορεί να σταλεί σε περισσότερα από ένα σημεία όπως φαίνεται και στην Εικόνα 5.6. Στο σημείο «a» γίνεται η αποστολή, ενώ στα σημεία «b» και «c» γίνεται η λήψη. Στην περίπτωση «b» γίνεται απλώς εκτύπωση του μηνύματος «Hello World», ενώ στην περίπτωση του «c» πραγματοποιείται μια επεξεργασία που παράγει το μήνυμα «I wanted to say Hello World!».



Εικόνα 5.6: Ενδεικτική χρήση receive/send μηνυμάτων.

5.3.4 Random Sequencer

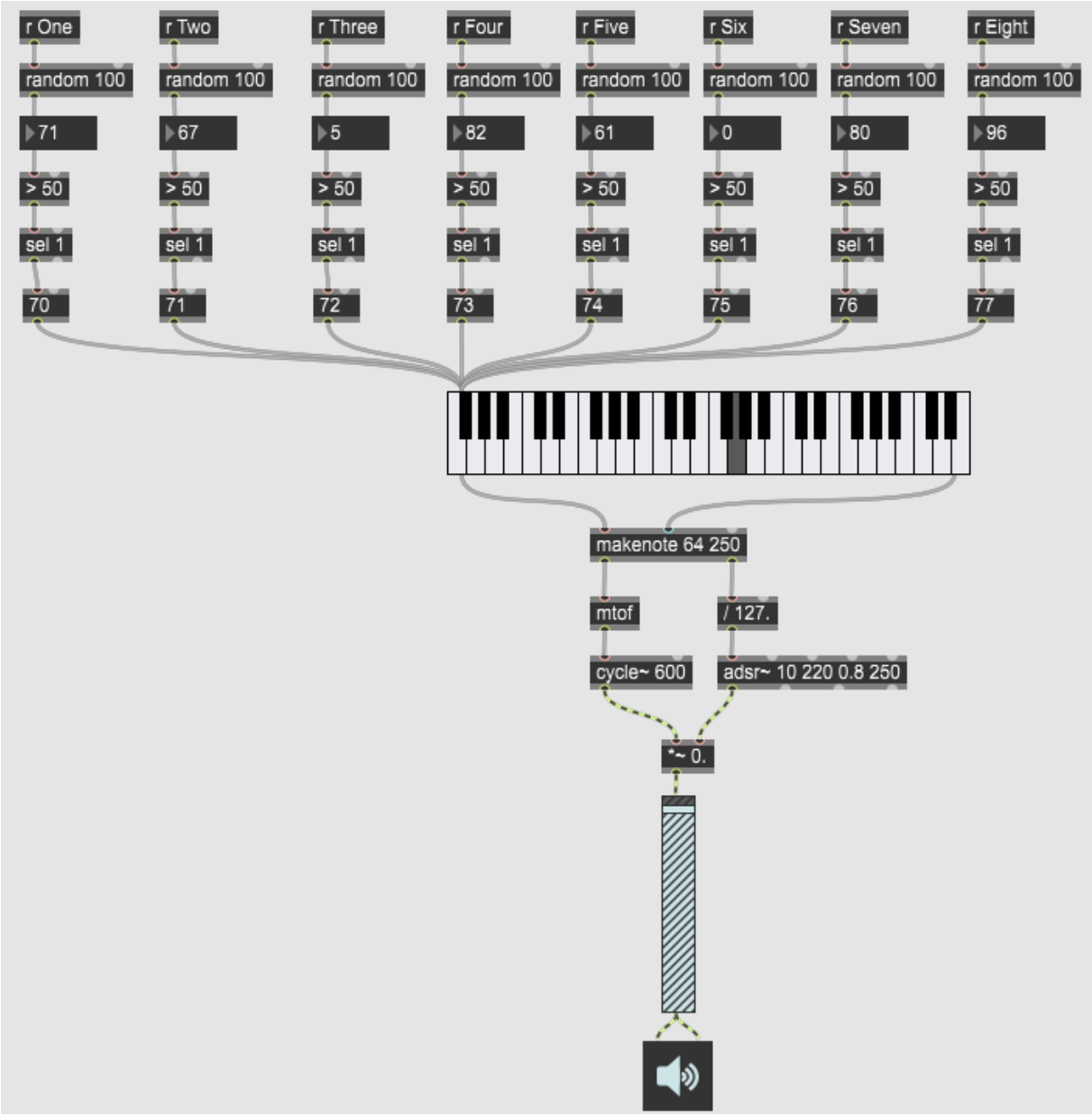
Το patch της συγκεκριμένης ενότητας είναι ένα από τα πολλά patch που δημιουργήσαμε για να εξοικειωθούμε και να πειραματιστούμε με την εφαρμογή MAX/MSP. Ίσως ο ήχος που παράγει να μην είναι επαγγελματικός και ακουστικά να μην είναι τόσο ελκυστικός, ωστόσο μας δίδαξε σχεδόν όσα χρειαζόμαστε για να δημιουργήσουμε την βασική εφαρμογή μας.

Στο επόμενο patch θα δείξουμε για πρώτη φορά ένα patch που παράγει ήχο. Ο ήχος αυτός παράγεται από ένα midi πιάνο. Αρχικά φτιάξαμε ένα patch που θυμίζει το patch της ενότητας 5.3.2, με την διαφοροποίηση ότι τα outlet της select στέλνουν 8 send μηνύματα (One, Two, Three, Four, Five, Six, Seven, Eight).

Στην ακόλουθη φωτογραφία φαίνονται τα receive αντικείμενα, με τα οποία γίνεται η λήψη των send μηνυμάτων που αναφέραμε. Όλα έχουν την ίδια λειτουργικότητα. Όταν λαμβάνουν το μήνυμα, παράγουν ένα τυχαίο αριθμό από το μηδέν μέχρι και το 99. Στην συνέχεια συγκρίνουν τον συγκεκριμένο αριθμό με το 50 και ελέγχουν αν είναι μεγαλύτερος. Αν η συνθήκη είναι αληθής και ο αριθμός είναι μεγαλύτερος του 50, τότε αποστέλλεται μήνυμα ανάλογο με τον αριθμό της κάθε περίπτωσης (70, 71, 72, 73, 74, 75, 76, 77).

Το μήνυμα προκαλεί το να πατηθεί το αντίστοιχο midi-to-note πλήκτρο του πιάνο, [39] και μετέπειτα να εκτελεστεί το «makenote». Αυτό εξάγει ένα μήνυμα «note-on» που συνοδεύεται από μια τιμή ταχύτητας (velocity) και ακολουθείται από ένα μήνυμα «note-off» μετά από ένα καθορισμένο χρονικό διάστημα. [40]

Στη συνέχεια συναντάμε για πρώτη φορά το αντικείμενο «cycle~». Το αντικείμενο αυτό παράγει μια συνεχή κυματομορφή που επαναλαμβάνεται περιοδικά. Ο χρόνος της περιόδου μπορεί να ρυθμιστεί από τον χρήστη. Καθώς η κυματομορφή επαναλαμβάνεται συνεχώς, μπορεί να δημιουργήσει μια συνεχή χρονική αλληλουχία ήχου, όπως μια συνεχής τονική νότα. [41]

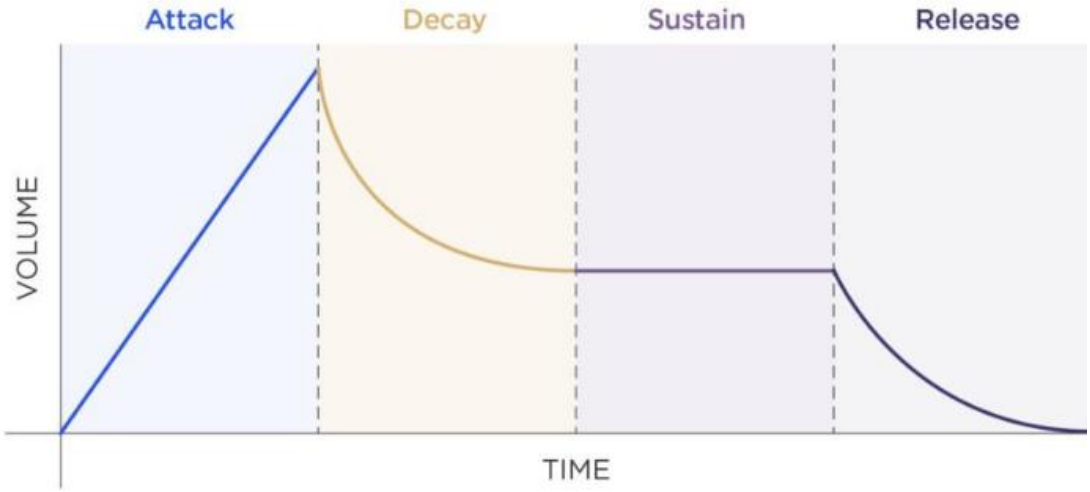


Εικόνα 5.7: Random Sequencer

Στην Εικόνα 5.7 το αντικείμενο «makenote» καθορίζει την ταχύτητα (velocity) με την οποία παράγεται η νότα. Η ταχύτητα με την οποία παράγεται η νότα στο συγκεκριμένο παράδειγμα είναι 64, ενώ η διάρκεια για να ξεκινήσει να εξασθενεί είναι τα 250 ms. Το «adsr~» εξάγει ένα σήμα Attack-Decay-Sustain-Release (ADSR). Τα Attack, Decay, Sustain και Release (ADSR) είναι φάσεις που χρησιμοποιούνται σε συνθεσάιζερ και ηλεκτρονικά μουσικά όργανα για να περιγράψουν την εξέλιξη της έντασης ήχου μετά από μια συνθετική εντολή ή μια νότα. Αυτές οι φάσεις αναφέρονται στις ακόλουθες έννοιες: [42]

- Attack (Επίθεση): Είναι η φάση κατά την οποία η ένταση του ήχου αυξάνεται από το μηδέν στη μέγιστη τιμή της.
- Decay (Υποχώρηση): Είναι η φάση κατά την οποία η ένταση του ήχου μειώνεται από τη μέγιστη τιμή σε ένα σταθερό επίπεδο (το επίπεδο συντήρησης).

- Sustain (Συντήρηση): Είναι η φάση κατά την οποία η ένταση του ήχου παραμένει στο σταθερό επίπεδο μέχρι την απελευθέρωση της νότας.
- Release (Απελευθέρωση): Είναι η φάση κατά την οποία η ένταση του ήχου μειώνεται από το επίπεδο συντήρησης στο μηδέν, μετά την απελευθέρωση της νότας. Αντιπροσωπεύει τον χρόνο κατά τον οποίο ο ήχος σβήνει και εξαφανίζεται μετά την απελευθέρωση της νότας.



Εικόνα 5.8: ADSR γράφημα. [43]

Αυτές οι φάσεις χρησιμοποιούνται για να δημιουργήσουν πλούσιους και δυναμικούς ήχους σε συνθετικά συστήματα και μουσικές εφαρμογές.

Στη συνέχεια η τιμή του ηχητικού κύματος πολλαπλασιάζεται με μια πολύ μικρή τιμή ώστε να φιλτραριστεί και να αποφευχθεί οποιοδήποτε πιθανό clipping. Το να χρησιμοποιούνται τέτοια μέτρα κρίνεται απαραίτητο για να μειωθεί ο κίνδυνος μικροτραυματισμών στα αυτιά.

Το συγκεκριμένο patch δημιουργήσαμε, για να επεξηγήσουμε αρχικά πως λειτουργεί το αντικείμενο “random” και για να δείξουμε ενδεικτικά πως μπορεί να παραχθεί ήχος. Υπάρχει ωστόσο και ένας ακόμη λόγος, για τον οποίο θα μιλήσουμε αναλυτικά στο επόμενο κεφάλαιο.

5.3.5 Επίλογος

Στο συγκεκριμένο κεφάλαιο δημιουργήσαμε εφαρμογές που κάποιος θα μπορούσε κάλλιστα να έχει δημιουργήσει χρησιμοποιώντας μια άλλη πιο διαδεδομένη γλώσσα προγραμματισμού. Το MAX/MSP αποτελεί μια εφαρμογή δημιουργίας λογισμικού, ωστόσο ξεχωρίζει από αυτές για διάφορους λόγους.

Αρχικά παρέχει ένα γραφικό περιβάλλον προγραμματισμού που επιτρέπει στους χρήστες να δημιουργούν προγράμματα, αντί να τους αναγκάζει να γράφουν κώδικα. Αυτό το καθιστά πιο προσβάσιμο και εύκολο στη χρήση από μη προγραμματιστές και ανθρώπους με περιορισμένη εμπειρία προγραμματισμού. Συγκριτικά με τις κοινές γλώσσες προγραμματισμού εξειδικεύεται στη δημιουργία ηχητικών συνθέσεων, μουσικών εφέ και αλληλεπιδραστικών εφαρμογών, κάτι που με μια κοινή εφαρμογή μπορεί να απαιτεί μεγαλύτερη τεχνογνωσία, εμπειρία και χρόνο. [44] Το MAX/MSP παρέχει επίσης δυνατότητες επέκτασης μέσω διάφορων extensions που δημιουργούνται από την κοινότητα που ασχολείται με αυτό.

Στο επόμενο κεφάλαιο δημιουργούμε μια εφαρμογή που συνδυάζει όσα σχεδόν μάθαμε μέχρι τώρα και προσθέτει κάποια επιπλέον αντικείμενα.

Κεφάλαιο 6ο: Music Box Patcher

6.1 Γενικά

Η φιλοσοφία τόσο του patch της ενότητας 5.3.4, όσο και του patcher του συγκεκριμένου κεφαλαίου είναι η δημιουργία μουσικής από τον άνθρωπο, σε συνδυασμό με τον παράγοντα της τύχης, που δίδεται από δεδομένα που λαμβάνει ένας ηλεκτρονικός υπολογιστής. Γι' αυτό και πολλές φορές η δημιουργία μουσικής με τέτοιο τρόπο ονομάζεται generative music.

Το «generative» μεταφράζεται στα ελληνικά ως «δημιουργικό» ή «παραγωγικό». Αναφέρεται σε κάτι που παράγεται ή δημιουργείται αυτόματα, χωρίς την άμεση παρέμβαση ή επίβλεψη από τον άνθρωπο. Στον τομέα της μουσικής, ο όρος «generative» χρησιμοποιείται για να περιγράψει μουσικά συστήματα ή διαδικασίες που δημιουργούν μουσικό περιεχόμενο αυτόνομα, χρησιμοποιώντας αλγόριθμους που συνήθως περιλαμβάνουν τύχη.

Η ιδέα της εφαρμογής είναι η παραγωγή generative music, η οποία ταυτοχρόνως θα είναι ambient. Η ambient μουσική είναι ένα είδος μουσικής που συνήθως χαρακτηρίζεται από ήρεμη, ατμοσφαιρική και αιθέρια αίσθηση. Σκοπός αυτού του είδους μουσικής είναι να δημιουργήσει ένα συνδυασμό διαφορετικών μεταξύ τους, αλλά μονότονων ήχων, που να αμβλύνει τις αισθήσεις και να δημιουργεί μια αίσθηση χαλάρωσης και ατμόσφαιρας.

Ο ήχος της ambient μουσικής συχνά αποτελείται από πολυστρωματικές ηχητικές αναταράξεις, μακράς διάρκειας ήχους, περιβάλλοντα θόρυβο, περιπλανώμενες μελωδίες και εφέ ήχου. Οι συνθέσεις της ambient μουσικής συχνά είναι απαλές και αργές, χωρίς έντονες δομές ή ρυθμικά μοτίβα.

6.2 Περιγραφή του Patcher

Στον συγκεκριμένο patcher, υπάρχουν πολλά subpatches, που πραγματοποιούν πολλές διαφορετικές λειτουργίες, ωστόσο αλληλοεπιδρούν μεταξύ τους, και λειτουργούν συνεργατικά, για να παράξουν το τελικό αποτέλεσμα.

Υπάρχει το subpatch που αρχικοποιεί, ή επαναφέρει την εφαρμογή σε μια συγκεκριμένη κατάσταση που έχουμε ορίσει (Initialize Subpatch). Υπάρχει το subpatch που ο ρόλος του είναι η αλληλεπίδραση με το Arduino και η αποστολή των δεδομένων στα σωστά σημεία (Arduino Subpatch).

Υπάρχουν επίσης τέσσερα subpatches που ενεργοποιούν συγκεκριμένα μέρη της εφαρμογής. Αυτά έχουν ρόλο να στέλνουν μήνυμα σε άλλους subpatchers, ώστε να παράξουν ήχο (firstInstrument, secondInstrument, thirdInstrument, allInstruments).

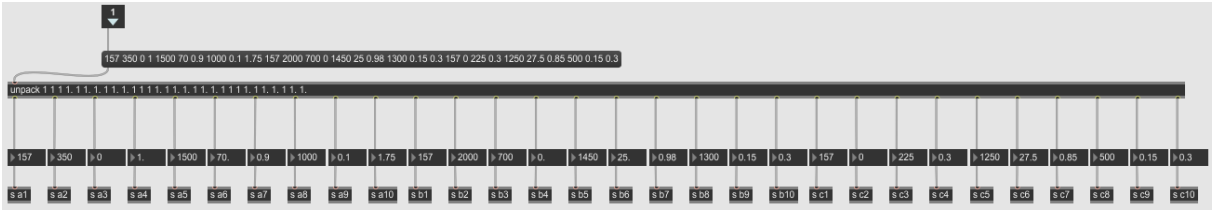
Τέλος ακολουθούν εννιά subpatchers που έχουν το ρόλο να καθορίζουν πως θα αναπαραχθούν οι ήχοι που οι προηγούμενοι τέσσερις subpatchers έχουν προκαλέσει (trigger). Ουσιαστικά οι τέσσερις subpatchers ορίζουν ότι κάποιες νότες πρέπει να αναπαραχθούν, ενώ οι υπόλοιποι εννιά ορίζουν τον τρόπο με τον οποίο πρέπει να αναπαραχθούν.

Όλα αυτά θα τα δούμε αναλυτικά στις επόμενες ενότητες.

6.3 Ανάλυση των διαφορετικών Subpatches

6.3.1 Initialize Subpatch

Το patcher που φαίνεται στην Εικόνα 6.1 έχει σαν ρόλο να αρχικοποιεί τις τιμές των διαφόρων μεταβλητών της εφαρμογής. Το subpatch εκτελείται κάθε φορά που ο χρήστης πιέζει το bang που φαίνεται στην Εικόνα 5.1.



Εικόνα 6.1: Initialize Subpatch

Στο πρώτο αντικείμενο εμπεριέχεται ένα μήνυμα με αριθμούς που αναπαριστούν τις διαφορετικές τιμές που θέλουμε να στείλουμε σε όλες τις μεταβλητές που υπάρχουν στο πρόγραμμα. Στην συνέχεια πραγματοποιείται το unpack, το οποίο λαμβάνει το παραπάνω String και το χωρίζει σε διακριτές τιμές τύπου integer ή float (1 για integer, και 1. για float). Στη συνέχεια γίνεται εμφάνιση αυτών των αριθμών στον χρήστη με τη χρήση integer αντικειμένων, και τέλος γίνεται αποστολή στα κατάλληλα πεδία με τα αντίστοιχα αντικείμενα send.

Οι μεταβλητές που μεταβάλλονται με το συγκεκριμένο patch είναι τριάντα, οπότε το message αντικείμενο περιέχει τόσους αριθμούς, και το ίδιο ισχύει και για το unpack αντικείμενο. Το σε ποια σημεία γίνεται η λήψη των συγκεκριμένων μηνυμάτων θα το δούμε σε επόμενες ενότητες.

6.3.2 Arduino Subpatch

6.3.2.1 Λήψη των τιμών x, y

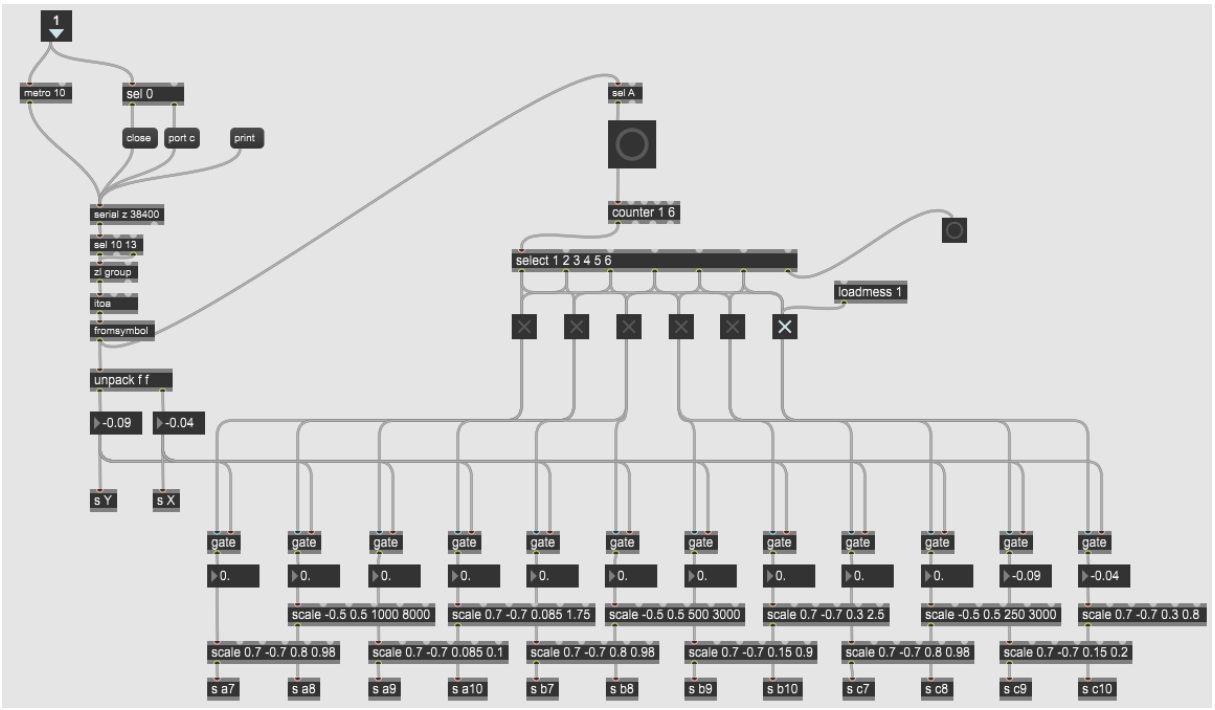
Ξεκινώντας την περιγραφή από το αριστερό μέρος της Εικόνας 6.2, βλέπουμε ότι όταν ενεργοποιηθεί το toggle του συγκεκριμένου patch θα ξεκινήσουν να πραγματοποιούνται δύο γεγονότα.

Το **πρώτο** και πιο απλό αφορά την επιλογή του serial port, στην οποία θα γίνει η επικοινωνία με το Arduino. Στην περίπτωση μας η επικοινωνία γίνεται με την USB serial port «C». Αν ο χρήστης πιέσει το print μπορεί να δει τις διαθέσιμες πόρτες επικοινωνίας για να επιλέξει την αντίστοιχη πόρτα που ταιριάζει στη δική του περίπτωση.

Όταν ένα toggle είναι ενεργοποιημένο, αποστέλλει μηνύματα με την τιμή 1 ανά συνεχή χρονικά διαστήματα. Όταν είναι απενεργοποιημένο, δεν αποστέλλει μηνύματα. Αυτό έχει σαν αποτέλεσμα όταν το trigger είναι ενεργοποιημένο, να επιλέγεται το port C σαν πόρτα επικοινωνίας, ενώ όταν το trigger απενεργοποιείται, να σταματάει η επικοινωνία με τη συγκεκριμένη πόρτα.

Το **δεύτερο** γεγονός που πραγματοποιείται αφορά τι είδους επεξεργασία προκαλείται με την επικοινωνία με το Arduino. Τέλος αρχικοποιεί την σειριακή επικοινωνία με ταχύτητα 38400 bits/sec, ενώ το αντικείμενο «metro 10» υπάρχει για το συγχρονισμό αποστολής/λήψης δεδομένων μεταξύ του Arduino και του MAX/MSP.

Ένα σημαντικό ASCII string είναι το «13 10», το οποίο αντιπροσωπεύει έναν χαρακτήρα αλλαγής γραμμής (carriage return). Ο κώδικας ζητά από το Arduino να εξάγει χαρακτήρες αλλαγής γραμμής. Όταν πραγματοποιείται αυτό, το αντικείμενο «sel», εξάγει ένα bang από το αριστερό outlet και τα υπόλοιπα δεδομένα από το δεξί outlet. Τα δεδομένα αυτά, όπως είδαμε σε ενότητα 4.2.3, είναι 2 τιμές που αναπαριστούν τις τιμές χ και y περιστροφής του επιταχυνσιομέτρου. Με την εντολή «zl group» συνδυάζουμε αυτές μέσα σε μια λίστα, ενώ αργότερα μετατρέπουμε τη λίστα των integers σε symbols (itoa - integer to ASCII), και στη συνέχεια τα σύμβολα σε μηνύματα που να μπορούν να επεξεργαστούν από το MAX/MSP (fromsymbol). [45]



Εικόνα 6.2: Arduino Subpatch

Τέλος το μήνυμα χωρίζεται σε μέρη, ώστε οι τιμές να μπορούν να επεξεργαστούν ανεξάρτητα η μια από την άλλη , με σκοπό να σταλούν στα κατάλληλα σημεία (Y, X, a7, a8, a9, a10, b7, b8, b9, b10, c7, c8, c9, c10).

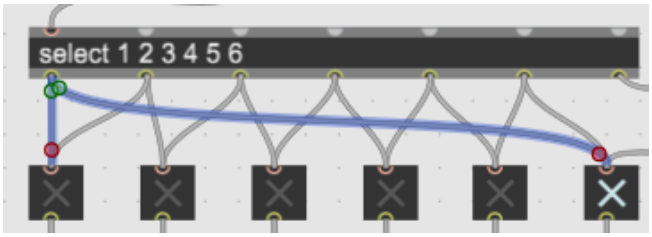
Τα δυο αυτά γεγονότα είναι αλληλεξαρτώμενα και λειτουργούν παράλληλα.

6.3.2.2 Λειτουργικότητα κουμπιού του Arduino.

Παράλληλα με τα γεγονότα που περιγράψαμε στην προηγούμενη ενότητα, ισχύει και το παρακάτω. Το αντικείμενο «select A» περιμένει να λάβει μήνυμα «A» για να στείλει κάποιο μήνυμα, κάτι που θα πραγματοποιηθεί όταν ο χρήστης πιέσει το κουμπί που υπάρχει συνδεδεμένο στο Arduino. Αυτό θα έχει σαν επίδραση να πατηθεί το bang που φαίνεται στην Εικόνα 6.2.

Κάθε φορά που το bang πιέζεται ξεκινάει η λειτουργία του αντικειμένου «counter». Το αντικείμενο αυτό αποτελεί ένα μετρητή που μετράει τιμές κάθε φορά που λαμβάνει ένα bang (στη δική μας περίπτωση μετρά τιμές από το 1 έως το 6). Κάθε φορά που λαμβάνει bang, στέλνει τη νέα τιμή στην έξοδό του.

Αυτό που θέλουμε να πέτυχουμε είναι η ταυτόχρονη ενεργοποίηση και απενεργοποίηση δύο διαδοχικών toggle αντικειμένων. Για να το πετύχουμε αυτό, έχουμε συνδέσει κάθε outlet του αντικειμένου select με το αντίστοιχο inlet από κάθε toggle αντικειμένου, που βρίσκεται ακριβώς από κάτω του, και ταυτόχρονα με το προηγούμενο του (όπως φαίνεται στην Εικόνα 6.3). Με αυτό τον τρόπο καταφέρνουμε κάθε φορά που πιέζουμε το κουμπί, σειριακά να ενεργοποιούμε και να απενεργοποιούμε τα διαφορετικά toggle που υπάρχουν, με αποτέλεσμα να επεξεργαζόμαστε και να στέλνουμε δεδομένα σε διαφορετικά σημεία τη φορά.



Εικόνα 6.3: Συνδεσμολογία των cords του counter

Μια ακόμη προϋπόθεση για να λειτουργήσει αυτό το αντικείμενο select σωστά, είναι ένα συγκεκριμένο από τα toggle αντικείμενα να είναι εξ' αρχής πατημένο. Αυτό που βολεύει να είναι ενεργοποιημένο είναι το τελευταίο κατά σειρά που υπάρχει. Αυτό ισχύει γιατί κατά το πρώτο πάτημα που θα πραγματοποιήσει ο χρήστης, θα ενεργοποιηθεί το πρώτο κατά σειρά toggle αντικείμενο. Πρακτικά αυτό σημαίνει ταυτόχρονη ενεργοποίηση του πρώτου toggle αντικείμενο και απενεργοποίηση του τελευταίου, το οποίο είναι αυτό ακριβώς που επιθυμούμε.

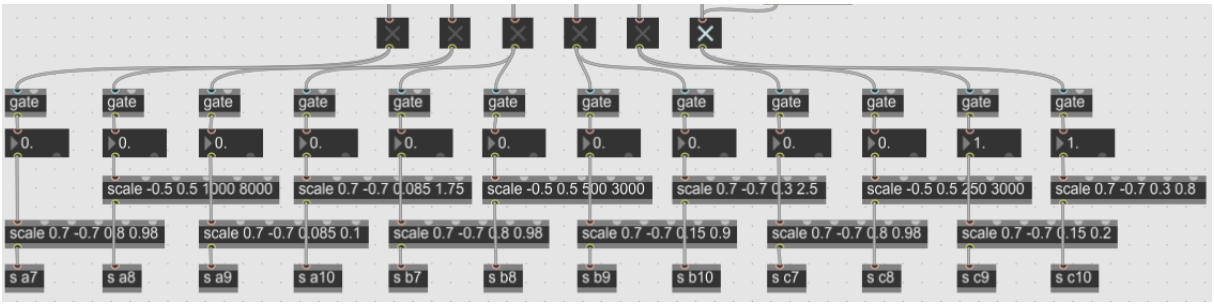
Η προϋπόθεση που περιγράψαμε πληρείται με τη χρήση της εντολής «loadmess». Αυτή είναι εντολή εκτελείται κατά την εκτέλεση του συνολικού patcher.

6.3.2.3 Τελική επεξεργασία τιμών πριν την αποστολή.

Στις προηγούμενες δύο ενότητες είδαμε πως γίνεται η λήψη την δεδομένων από το Arduino καθώς και πως φέραμε αυτά σε μορφή που να μπορούμε να τα επεξεργαστούμε και να τα χρησιμοποιήσουμε. Σε αυτή την ενότητα προσπαθούμε να βρούμε ένα αποδοτικό τρόπο να κάνουμε το μουσικό όργανο λειτουργικό και εύχρηστο.

Πριν ξεκινήσουμε να εξηγούμε τι απεικονίζει η κάθε φωτογραφία καλό είναι να ξέρουμε τη κάνει το αντικείμενο gate. Αυτό λειτουργεί ως ένας διακόπτης που επιτρέπει ή αποκλείει την διέλευση των δεδομένων, ανάλογα με την κατάστασή του. Η gate έχει δύο εισόδους: μία για τον έλεγχο της κατάστασης (τιμή on/off) και μία για τα δεδομένα που θέλουμε να στείλουμε. Για να μπορέσουμε να περάσουμε δεδομένα μέσα από το «gate», είναι προϋπόθεση το αριστερό inlet του «gate» να είναι αριθμός μικρότερος του -1 ή μεγαλύτερος του 1.

Στην Εικόνα 6.2 η συνδεσμολογία των καλωδίων δεν είναι ευδιάκριτη. Γι' αυτό το λόγο έχουμε χωρίσει τον patcher σε κομμάτια, ώστε να μπορέσουμε να εξηγήσουμε τις υπολειτουργίες του πιο εύκολα και αποδοτικά. Σύμφωνα με όσα είπαμε παραπάνω και κοιτώντας την Εικόνα 6.4, παρατηρούμε ότι με κάθε ενεργοποιημένο toggle, στέλνονται δεδομένα, σε ταυτόχρονα 2 «gate» τη φορά.



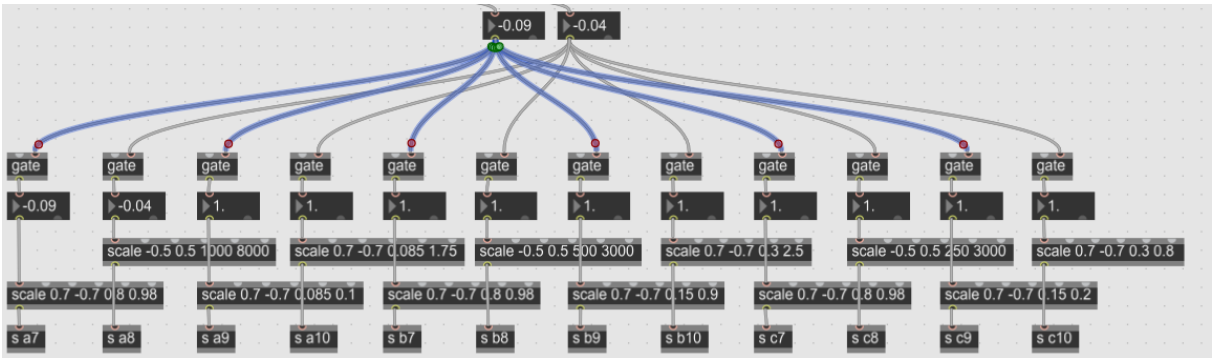
Εικόνα 6.4: Toggle-to-gate Cords

Στην Εικόνα 6.5 βλέπουμε τη συνδεσμολογία μεταξύ των float τιμών που λαμβάνουμε από το Arduino, με τα αντίστοιχα «gate». Παρατηρούμε ότι οι συνδέσεις που πραγματοποιούνται, πραγματοποιούνται εναλλάξ. Τα μπλε cords (μέσα από τα οποία αποστέλλεται το output του άξονα y του επιταχυνσιομέτρου) συνδέονται με τα αντικείμενα «gate» που οδηγούν στα αντικείμενα «send» a7, a9, b7, b9, c7 και c9. Τα υπόλοιπα (μέσα από τα οποία αποστέλλεται το output του άξονα χ του επιταχυνσιομέτρου) οδηγούν στα αντικείμενα «send» a8, a10, b8, b10, c8, c10.

Αν συνδυάσουμε τις πληροφορίες που μάθαμε στην Εικόνα 6.4 με τις πληροφορίες της Εικόνας 6.5, θα προσέξουμε ότι με κάθε ένα toggle που είναι ενεργοποιημένο, στέλνεται μια πληροφορία χ σε ένα σημείο και μια πληροφορία y σε ένα άλλο. Οι δυνατές περιπτώσεις που μπορεί να συναντήσουμε είναι οι εξής:

1. a7, a8 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.
2. a9, a10 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.
3. b7, b8 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.
4. b9, b10 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.
5. c7, c8 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.
6. c9, c10 ενεργοποιημένα και όλα τα υπόλοιπα απενεργοποιημένα.

Με αυτό τον τρόπο μπορούμε να στέλνουμε δεδομένα σε συγκεκριμένα σημεία της εφαρμογής κάθε φορά.



Εικόνα 6.5: Arduino Output-to-gate Cords

Μετά τις εντολές gate βλέπουμε τις τιμές που στέλνει το Arduino στα αντικείμενα float που ακολουθούν. Σε αυτές πραγματοποιείται μια επεξεργασία με τη χρήση του αντικειμένου «scale». Το αντικείμενο «scale» αντιστοιχεί ένα εύρος εισόδου από ακέραιες ή δεκαδικές τιμές σε ένα εύρος εξόδου. Οι παράμετροι που δέχεται είναι συνήθως τέσσερις. Κατά σειρά συμβολίζουν τα min-input, max-input,

min-output, max-output. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για να μετατρέψει μια τιμή από το εύρος 0-1 σε ένα άλλο εύρος, όπως 0-100.

Αν θέλαμε να μετατρέψουμε την λειτουργία του αντικειμένου scale αυτού σε μαθηματική εξίσωση, η εξίσωση θα ήταν η εξής:

$$x = (inlet - a) * \frac{d - c}{b - a} + c$$

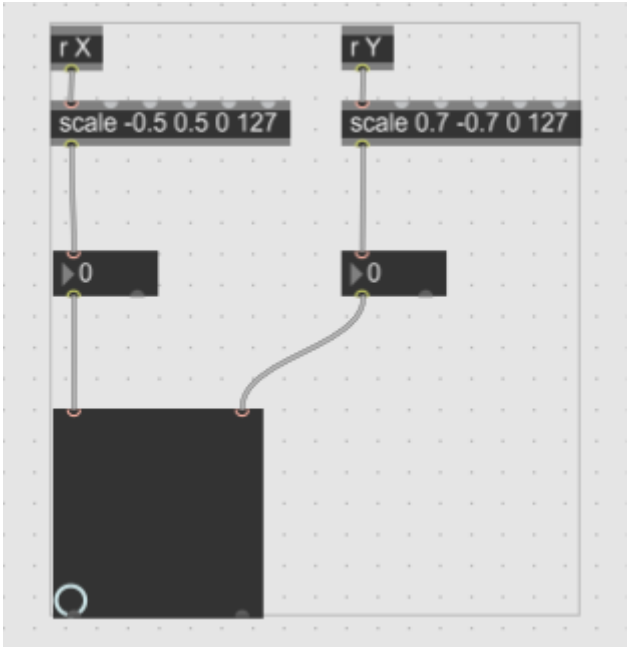
όπου:

- x: είναι το output του outlet του αντικειμένου «scale»
- inlet: είναι η τιμή του inlet που δέχεται το αντικείμενο «scale»
- a: min-input
- b: max-input
- c: min-input
- d: max-output

6.3.3 Arduino Subpatch 2

Το ακόλουθο subpatch φαίνεται στην Εικόνα 6.6 δεν έχει κάποια ιδιαίτερη χρησιμότητα που να αφορά την καλύτερη χρήση, ή αποδοτικότητα της εφαρμογής. Είναι ωστόσο μια ωραία οπτικοποίηση, της περιστροφής του Arduino. Με αυτό γίνεται πιο κατανοητό στον χρήστη, το τι ακριβώς πραγματοποιείται κατά την εκτέλεση της εφαρμογής.

Το συγκεκριμένο subpatch λαμβάνει τις τιμές χ και y, που στέλνει το patch που φαίνεται στην Εικόνα 6.2. Στη συνέχεια γίνεται επεξεργασία των τιμών στο διάστημα 0-127 με τη χρήση του αντικειμένου «scale». Ενώ στη συνέχεια βλέπουμε τον αριθμό στο integer αντικείμενο και μια αναπαράσταση της περιστροφής με τη χρήση του αντικειμένου «pictslider».



Εικόνα 6.6: Arduino Output Visualization for user

6.3.4 Instrument Subpatches

Τα ακόλουθα 4 subpatches, όπως αναφέραμε σε προηγούμενη ενότητα δίνουν το έναυσμα ώστε να αναπαραχθεί ήχος, ωστόσο δεν συμβάλουν στο πώς αναπαράγεται αυτός (με ποια μορφή).

6.3.4.1 First Instrument Subpatch

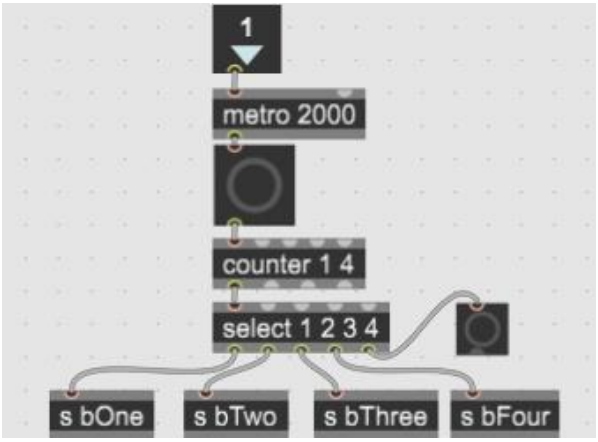
Το ακόλουθο subpatch έχει το ρόλο της αναπαραγωγής του πρώτου μουσικού οργάνου που δημιουργήσαμε.

Εικόνα 6.7: First Instrument Subpatch

Όταν ο χρήστης ενεργοποιήσει το συγκεκριμένο subpatch μέσω του ανάλογου toggle που βρίσκεται στον Main Patcher, θα ξεκινήσει και η λειτουργία της εφαρμογής. Κάθε 500 ms (μισό δευτερόλεπτο) στέλνεται ένα bang μήνυμα που ενεργοποιεί ένα counter αντικείμενο. Αυτό με τη σειρά του ενεργοποιεί ένα select αντικείμενο που στέλνει σειριακά τα μηνύματα One, Two Three, Four, Five, Six, Seven και Eight.

6.3.4.2 Second Instrument Subpatch

Το ακόλουθο subpatch έχει το ρόλο της αναπαραγωγής του δευτέρου μουσικού οργάνου που δημιουργήσαμε.

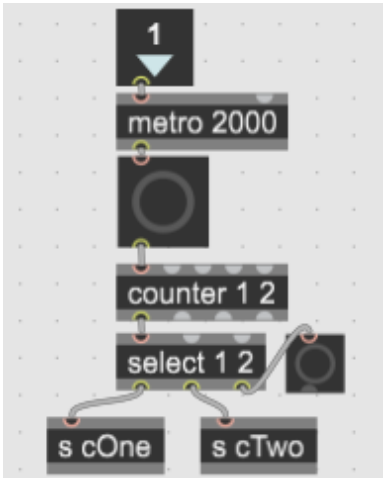


Εικόνα 6.8: Second Instrument Subpatch

Όταν ο χρήστης ενεργοποιήσει το συγκεκριμένο subpatch μέσω του ανάλογου toggle που βρίσκεται στον Main Patcher, θα ξεκινήσει και η λειτουργία της εφαρμογής. Κάθε 2000 ms (δύο δευτερόλεπτα) αποστέλλεται ένα bang μήνυμα που ενεργοποιεί ένα counter αντικείμενο. Αυτό με τη σειρά του ενεργοποιεί ένα select αντικείμενο που στέλνει σειριακά τα μηνύματα bOne, bTwo bThree, bFour.

6.3.4.3 Third Instrument Subpatch

Το ακόλουθο subpatch έχει το ρόλο της αναπαραγωγής του τρίτου μουσικού οργάνου που δημιουργήσαμε.

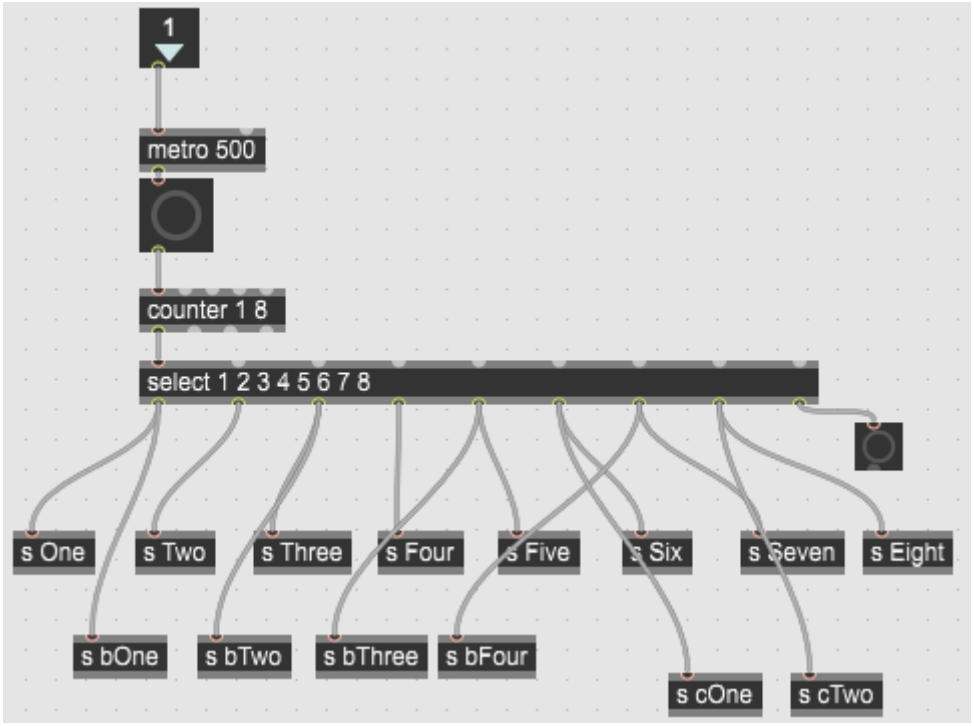


Εικόνα 6.9: Third Instrument Subpatch

Όταν ο χρήστης ενεργοποιήσει το συγκεκριμένο subpatch μέσω του ανάλογου toggle που βρίσκεται στον Main Patcher, θα ξεκινήσει και η λειτουργία της εφαρμογής. Κάθε 2000 ms (δύο δευτερόλεπτα) στέλνεται ένα bang μήνυμα που ενεργοποιεί ένα counter αντικείμενο. Αυτό με τη σειρά του ενεργοποιεί ένα select αντικείμενο που στέλνει σειριακά τα μηνύματα cOne, cTwo.

6.3.4.4 All Instruments Subpatch

Το ακόλουθο subpatch έχει το ρόλο της αναπαραγωγής των τριών μουσικών οργάνων που δημιουργήσαμε ταυτόχρονα.



Εικόνα 6.10: All Instruments Subpatch

Στην συγκεκριμένη περίπτωση η συχνότητα αναπαραγωγής του κάθε οργάνου είναι διαφορετική. Αυτό συμβαίνει καθώς κάθε όργανο εξυπηρετεί διαφορετικό σκοπό. Κάποιο αποτελεί τη βασική μελωδία, ενώ κάποιο άλλο είναι δευτερεύον κι υπάρχει για «χρωματισμό». Αυτό έχει σαν συνέπεια, κάθε outlet του αντικειμένου «select» έχει διαφορετικό αριθμό από συνδεδεμένα cords.

Όταν ο χρήστης ενεργοποιήσει το συγκεκριμένο subpatch μέσω του ανάλογου toggle που βρίσκεται στον Main Patcher, θα ξεκινήσει και η λειτουργία της εφαρμογής. Κάθε 500 ms (μισό δευτερόλεπτο) στέλνεται ένα bang μήνυμα που ενεργοποιεί ένα counter αντικείμενο. Αυτό με τη σειρά του ενεργοποιεί ένα select αντικείμενο που στέλνει σειριακά συγκεκριμένα μηνύματα.

Η διαφορά ξεκινάει εδώ, όπου με κάθε μήνυμα «bang» που στέλνεται, διαφοροποιείται το τελικό μήνυμα/έναυσμα που στέλνεται (και κατ' επέκταση το μουσικό όργανο που αναπαράγεται). Παρατηρούμε στην Εικόνα 6.10 ότι στον πρώτο χρόνο στέλνονται τα μηνύματα One και bOne, στον δεύτερο το Two, στον τρίτο τα Three και bTwo, στον τέταρτο το Four, στον πέμπτο το Five και bThree, στον έκτο το Six και cOne, στον έβδομο το Seven και bFour, και στον όγδοο το Eight και cTwo.

Στον ακόλουθο πίνακα κάνουμε δείχνουμε τη συχνότητα αποστολής αυτών των μηνυμάτων καθενός από τα μουσικά όργανα. Το αποτέλεσμα αποστολής αυτών των μηνυμάτων θα το δούμε στην επόμενη ενότητα, που είναι και η τελική που αφορά το προγραμματιστικό κομμάτι της εφαρμογής.

Πίνακας 1: Συχνότητα εντολής/εναύσματος αναπαραγωγής μουσικών οργάνων.

	1	2	3	4	5	6	7	8
First Instrument	X	X	X	X	X	X	X	X
Second Instrument	X		X		X		X	
Third Instrument						X		X

6.4 Main Patch

Το συγκεκριμένο patch έχει σαν ρόλο να ενώνει όλα τα προηγούμενα subpatches που παρουσιάσαμε μεταξύ τους. Μέσα από αυτό δίνουμε όλες τις βασικές διαφορετικές εντολές όπως η ενεργοποίηση της επικοινωνίας με το Arduino, η αναπαραγωγή των διαφορετικών μουσικών οργάνων και ηχητική επεξεργασία αυτών των οργάνων σε πραγματικό χρόνο. Η ενεργοποίηση αυτών των subpatches γίνεται από τα διαφορετικά toggles που φαίνονται στην Εικόνα 5.1.

Όπως αναφέραμε και στην εισαγωγή του συγκεκριμένου κεφαλαίου η μουσική που παράγεται από τον συγκεκριμένο patcher είναι generative/ambient. Εξηγήσαμε ότι για το τελικό ακουστικό αποτέλεσμα επηρεάζεται από τον παράγοντα της τύχης. Μέχρι στιγμής (σε όλα τα προηγούμενα subpatches) που είδαμε, παρατηρήσαμε ότι οι εντολές που στέλνονται προκαλούν μια διαδοχή send μηνυμάτων που δεν εξαρτώνται από κάποιο παράγοντα πιθανοτήτων. Αυτός είναι και ο λόγος που περιγράψαμε ότι αυτά τα subpatches απλώς προκαλούν το έναυσμα αναπαραγωγής του ήχου, αλλά δεν παίζουν κάποιο ρόλο στο πως αναπαράγεται αυτός ο ήχος. Το πως γίνεται πραγματικά η αναπαραγωγή του ήχου θα το δούμε στους ακόλουθους subpatchers.

6.4.1 First Instrument Piano

Το συγκεκριμένο subpatch λαμβάνει τα μηνύματα One, Two Three, Four, Five, Six, Seven και Eight, από το «First Instrument Subpatch» ή το «All Instruments Subpatch», αναλόγως ποιο είναι ενεργοποιημένο.

Κάθε ένα από τα μηνύματα που λαμβάνεται παράγει ένα τυχαίο αριθμό μεταξύ των τιμών 0 και 99. Αργότερα ελέγχεται το γεγονός αν αυτοί οι αριθμοί είναι μικρότεροι από συγκεκριμένες σταθερές. Αν ο αριθμός που έχει παραχθεί είναι μικρότερος από την σταθερά που έχει δοθεί, τότε θα σταλεί ο αριθμός midi νότας που αναγράφεται (60, 62, 64, 67, 69) στο αντικείμενο «keyslider» (πιάνο) που φαίνεται στην Εικόνα 6.11. [46] Από την εικόνα παρατηρούμε ότι η κάθε νότα έχει διαφορετική πιθανότητα αναπαραγωγής. Πιο συγκεκριμένα για τις νότες ισχύουν:

Πίνακας 2: Στατιστικά στοιχεία δεδομένων του πρώτου μουσικού οργάνου.

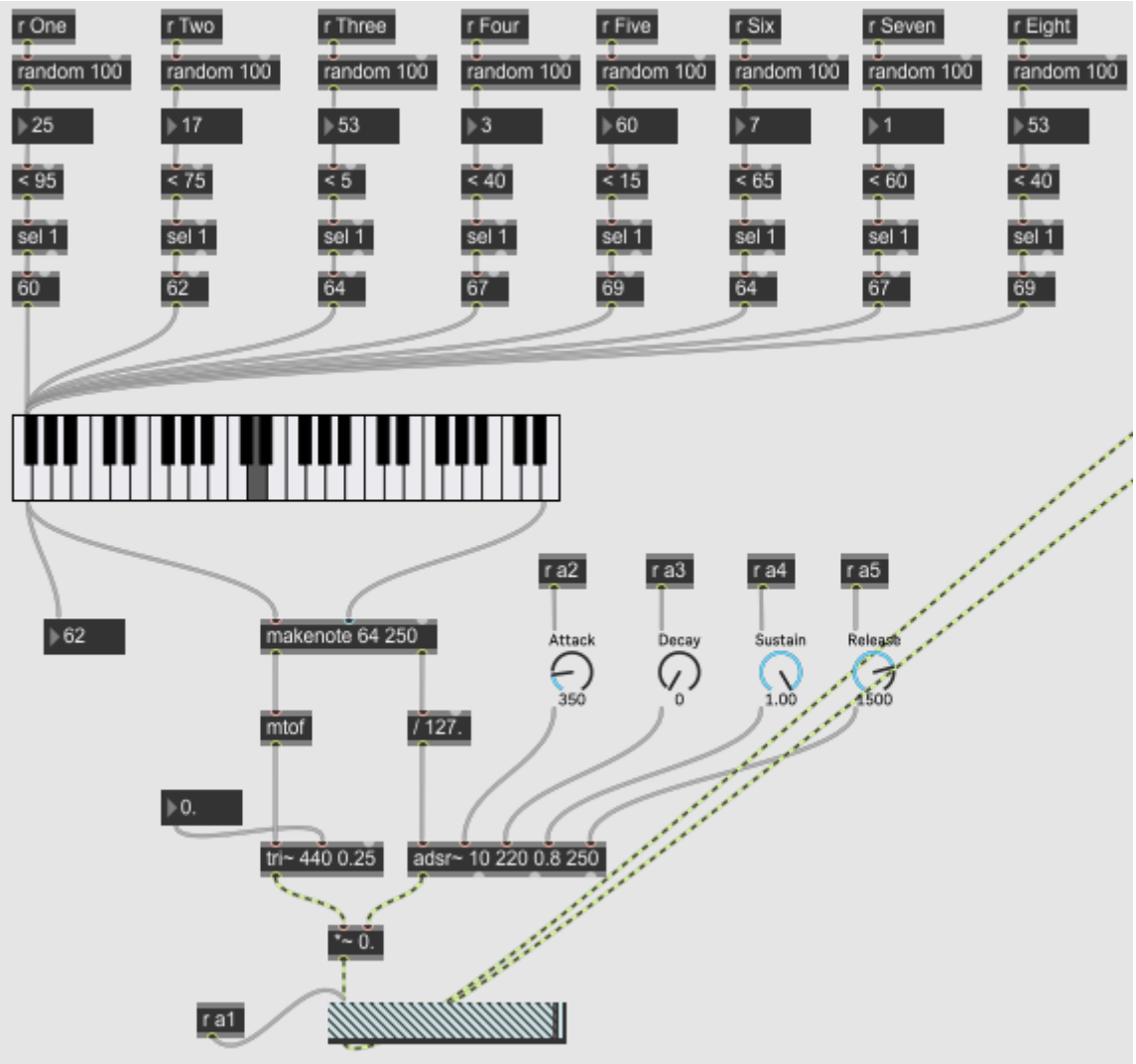
Αριθμός Νότας	Αριθμός MIDI	Όνομα Νότας	Πιθανότητα Αναπαραγωγής
1	60	Ντο4 (C4)	95%
2	62	Ρε4 (D4)	75%
3	64	Μι4 (E4)	5%
4	67	Σολ4 (G4)	40%

5	69	Λα4 (A4)	15%
6	64	Μι4 (E4)	65%
7	67	Σολ4 (G4)	60%
8	69	Λα4 (A4)	40%

Το αντικείμενο «makenote» παράγει κάθε φορά νότα με ταχύτητα 64 και διάρκεια εξασθένησης τα 250 ms. Στη συνέχεια, με τη χρήση του «mtof» αντικειμένου (midi-to-frequency) πραγματοποιείται μετατροπή της MIDI νότας σε συχνότητα. Το «tri~» παράγει ένα τριγωνικό κύμα του οποίου οι συνιστώσες συχνοτήτων αντιστέκονται στο aliasing (απώλεια πληροφορίας κατά τη διαδικασία δειγματοληψίας και ανακατασκευής του σήματος). [47]

Το ακουστικό αποτέλεσμα της κάθε νότας επηρεάζεται επίσης από το αντικείμενο «adsr» του οποίου τη λειτουργία έχουμε αναπτύξει στην ενότητα 5.3.4. Οι μεταβλητές που δέχεται το «adsr» αρχικοποιούνται από τον «Initialize Subpatcher» της ενότητας 6.3.1, ωστόσο μπορούν να τροποποιηθούν και κατά την εκτέλεση της εφαρμογής.

Τα σήματα που παράγουν τα «tri~» και «adsr~» πολλαπλασιάζονται μεταξύ τους, στη συνέχεια περνάνε μέσα από το «gain~» αντικείμενο το οποίο φαίνεται στην Εικόνα 6.11 σαν γραφικό slider και μας επιτρέπει να πειράζουμε την ένταση του παραγόμενου ήχου. Κατόπιν το σήμα περνάει μέσα απ’ το Reverb Subpatch για περαιτέρω επεξεργασία, του οποίου τη λειτουργία θα αναπτύξουμε στην ενότητα 6.4.7.



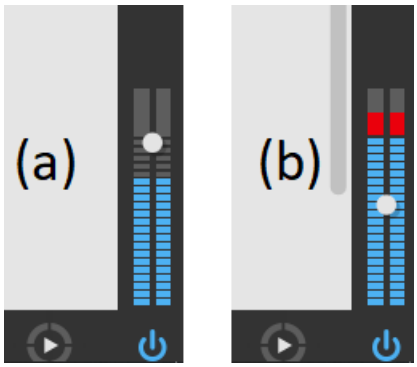
Εικόνα 6.11: Final Patch First Instrument Piano

Με τη χρήση του αντικειμένου random προκαλούμε τυχαιότητα στην αναπαραγωγή του ήχου. Κάθε νότα έχει συγκεκριμένη πιθανότητα αναπαραγωγής. Έτσι, με κάθε επανάληψη ο παραγόμενος ήχος είναι (σχεδόν) πάντα διαφορετικός.

6.4.2 First Instrument Filter graph

Στο MAX/MSP, το αντικείμενο «audio meter/gain» που βρίσκεται στο κάτω δεξιά μέρος του κεντρικού παραθύρου, αντιπροσωπεύει ένα μετρητή ήχου και έλεγχο ενίσχυσης ήχου. Όταν ο μετρητής εμφανίσει κόκκινο χρώμα, αυτό υποδεικνύει ότι η εισερχόμενη ή εξερχόμενη ένταση του ήχου ξεπερνά τα επιτρεπτά όρια.

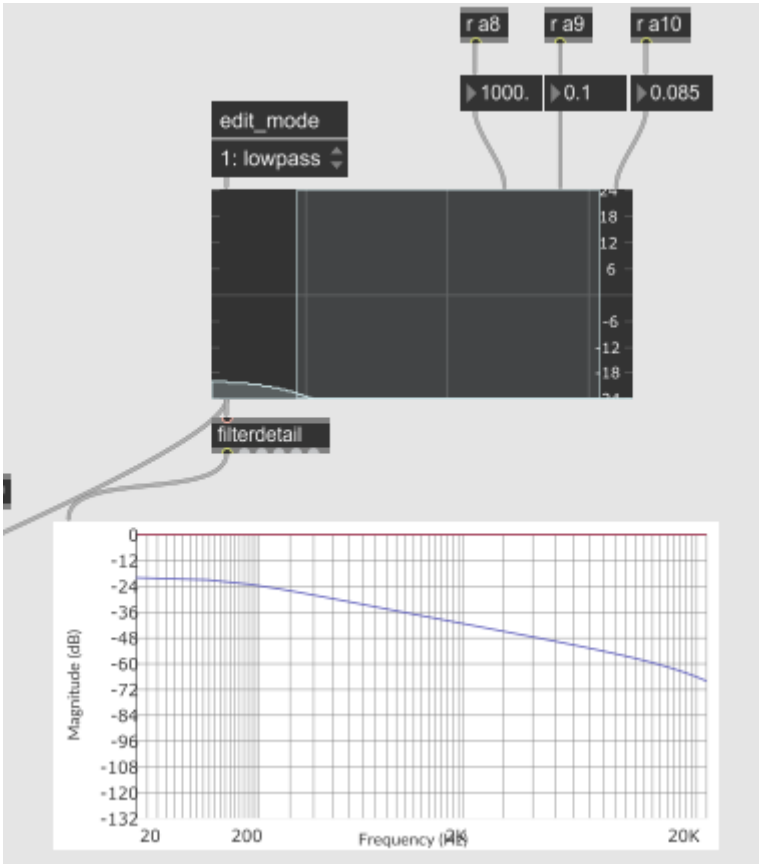
Ο κόκκινος χρωματισμός του μετρητή υποδεικνύει πιθανώς υπερβολικά δυνατό ή παραμορφωμένο σήμα. Αυτό μπορεί να συμβαίνει ανάλογα με την ρύθμιση της ενίσχυσης ήχου ή τον έλεγχο της σήμανσης (clipping) στο σύστημα ήχου. Σε κάθε ανάλογη περίπτωση είναι σημαντικό να ρυθμίσουμε την έξοδο του σήματος έτσι ώστε να μην υπερβαίνουμε τα επιτρεπτά όρια, προκαλώντας παραμόρφωση ή απώλεια ποιότητας του ήχου.



Εικόνα 6.12: audio meter/gain

Γι' αυτό ακριβώς το λόγο έχουμε προσθέσει ένα φίλτρο στο συγκεκριμένο patcher, ώστε ο ήχος να μην ξεπερνάει τα επιτρεπτά όρια. Το αντικείμενο που μας παρέχει αυτό το φίλτρο ονομάζεται «filtergraph». Υπάρχουν διαφόρων ειδών φίλτρα που παρέχει το filtergraph, όπως το lowpass, highpass, bandpass, bandstop, peaknotch, lowshelf, highshelf, resonant, allpass. [41]

Στη συγκεκριμένη περίπτωση χρησιμοποιούμε το lowpass filter, το οποίο μπορούμε να παραμετροποιήσουμε σε πραγματικό χρόνο, λαμβάνοντας τις τιμές από το Arduino και χρησιμοποιώντας τα μηνύματα a8, a9 και a10 που στέλνουμε, όπως είδαμε στην Ενότητα 6.3.2.3.



Εικόνα 6.13: Final Patch First Instrument Filter Graph

6.4.3 Second Instrument Piano

Το συγκεκριμένο subpatch λαμβάνει τα μηνύματα bOne, bTwo bThree και bFour, από το «Second Instrument Subpatch» ή το «All Instruments Subpatch», αναλόγως ποιο είναι ενεργοποιημένο.

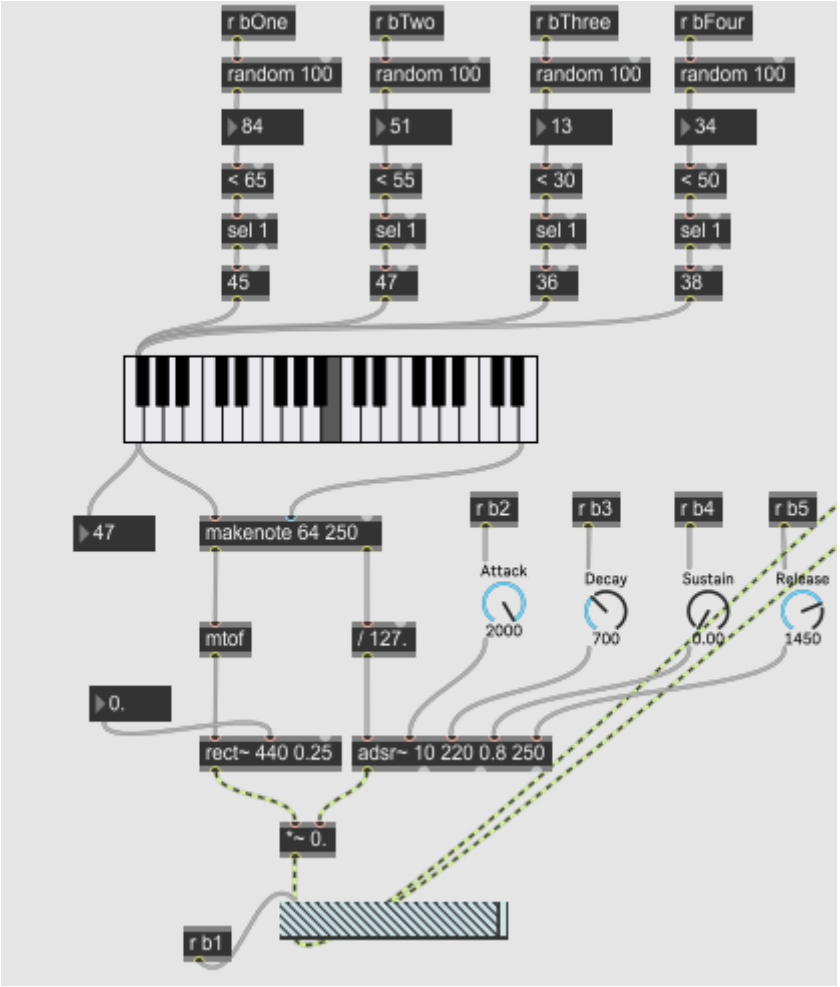
Κάθε ένα από τα μηνύματα που λαμβάνεται παράγει ένα τυχαίο αριθμό μεταξύ των τιμών 0 και 99. Αργότερα ελέγχεται το γεγονός αν αυτοί οι αριθμοί είναι μικρότεροι από συγκεκριμένες σταθερές. Αν ο αριθμός που έχει παραχθεί είναι μικρότερος από την σταθερά που έχει δοθεί, τότε θα σταλεί ο αριθμός midi νότας που αναγράφεται (45, 47, 36, 38) στο αντικείμενο «keyslider» (πιάνο) που φαίνεται στην Εικόνα 6.14. Από την εικόνα παρατηρούμε ότι η κάθε νότα έχει διαφορετική πιθανότητα αναπαραγωγής. Πιο συγκεκριμένα για τις νότες ισχύουν:

Πίνακας 3: Στατιστικά στοιχεία δεδομένων του δευτέρου μουσικού οργάνου.

Αριθμός Νότας	Αριθμός MIDI	Όνομα Νότας	Πιθανότητα Αναπαραγωγής
1	45	Λα 2 (A2)	65%
2	47	Σι 2 (B2)	55%
3	36	Ντο 2 (C2)	30%
4	38	Ρε 2 (D2)	50%

Το αντικείμενο «makenote» παράγει κάθε φορά νότα με ταχύτητα 64 και διάρκεια εξασθένησης τα 250 ms. Στη συνέχεια, με τη χρήση του «mtof» αντικειμένου (midi-to-frequency) πραγματοποιείται μετατροπή της MIDI νότας σε συχνότητα. Το «rect~» παράγει ένα τετραγωνικό κύμα του οποίου οι συνιστώσες συχνοτήτων αντιστέκονται στο aliasing.

Το ακουστικό αποτέλεσμα της κάθε νότας επηρεάζεται επίσης από το αντικείμενο «adsr». Τα σήματα που παράγουν τα «rect~» και «adsr~» πολλαπλασιάζονται μεταξύ τους, στη συνέχεια περνάνε μέσα από το «gain~» αντικείμενο, που μας επιτρέπει να πειράζουμε την ένταση του παραγόμενου ήχου. Κατόπιν το σήμα περνάει μέσα απ' το Reverb Subpatch για περαιτέρω επεξεργασία.

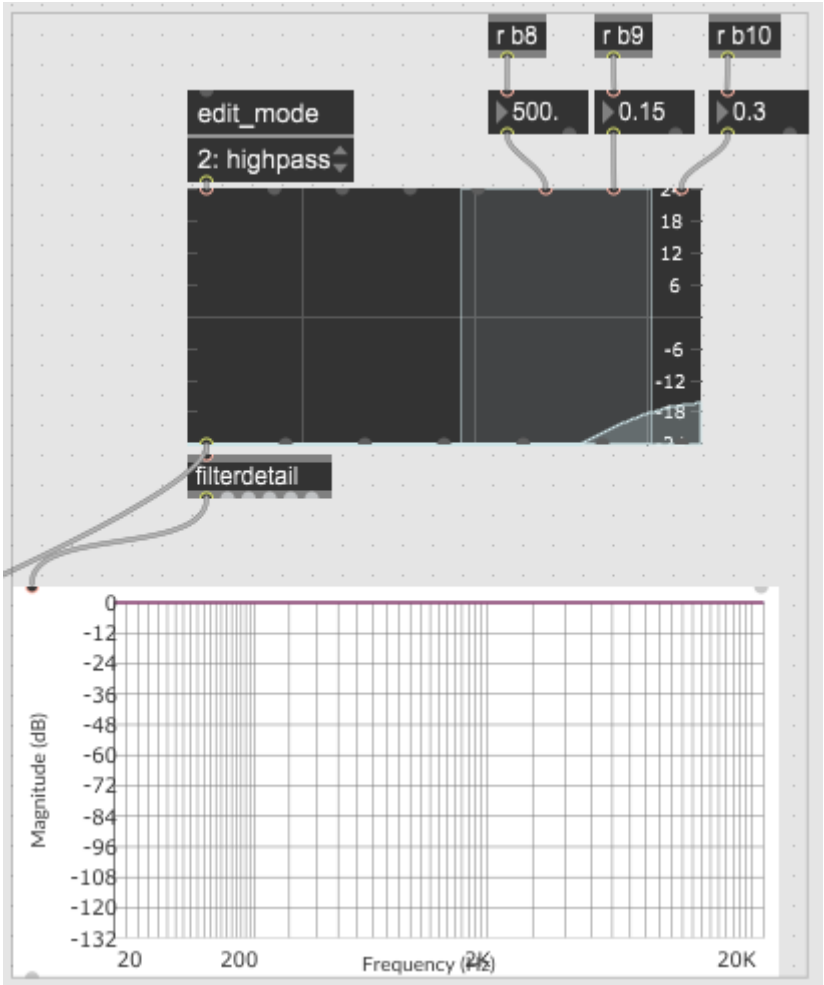


Εικόνα 6.14: Final Patch Second Instrument Piano

Όπως και πριν, με τη χρήση του αντικειμένου random προκαλούμε τυχαιότητα στην αναπαραγωγή του ήχου. Κάθε νότα έχει συγκεκριμένη πιθανότητα αναπαραγωγής. Έτσι, με κάθε επανάληψη ο παραγόμενος ήχος είναι (σχεδόν) πάντα διαφορετικός.

6.4.4 Second Instrument Filter graph

Στο συγκεκριμένο patcher έχουμε προσθέσει ένα φίλτρο, ώστε ο ήχος να μην ξεπερνάει τα επιτρεπτά όρια. Το αντικείμενο που μας παρέχει αυτό το φίλτρο ονομάζεται «filtergraph». Στη συγκεκριμένη περίπτωση χρησιμοποιούμε το highpass filter, το οποίο μπορούμε να παραμετροποιήσουμε σε πραγματικό χρόνο, λαμβάνοντας τις τιμές από το Arduino και χρησιμοποιώντας τα μηνύματα b8, b9 και b10 που στέλνουμε, όπως είδαμε στην Ενότητα 6.3.2.3.



Εικόνα 6.15: Final Patch Second Instrument Filter

6.4.5 Third Instrument Piano

Το συγκεκριμένο subpatch λαμβάνει τα μηνύματα cOne και cTwo από το «Third Instrument Subpatch» ή το «All Instruments Subpatch», αναλόγως ποιο είναι ενεργοποιημένο.

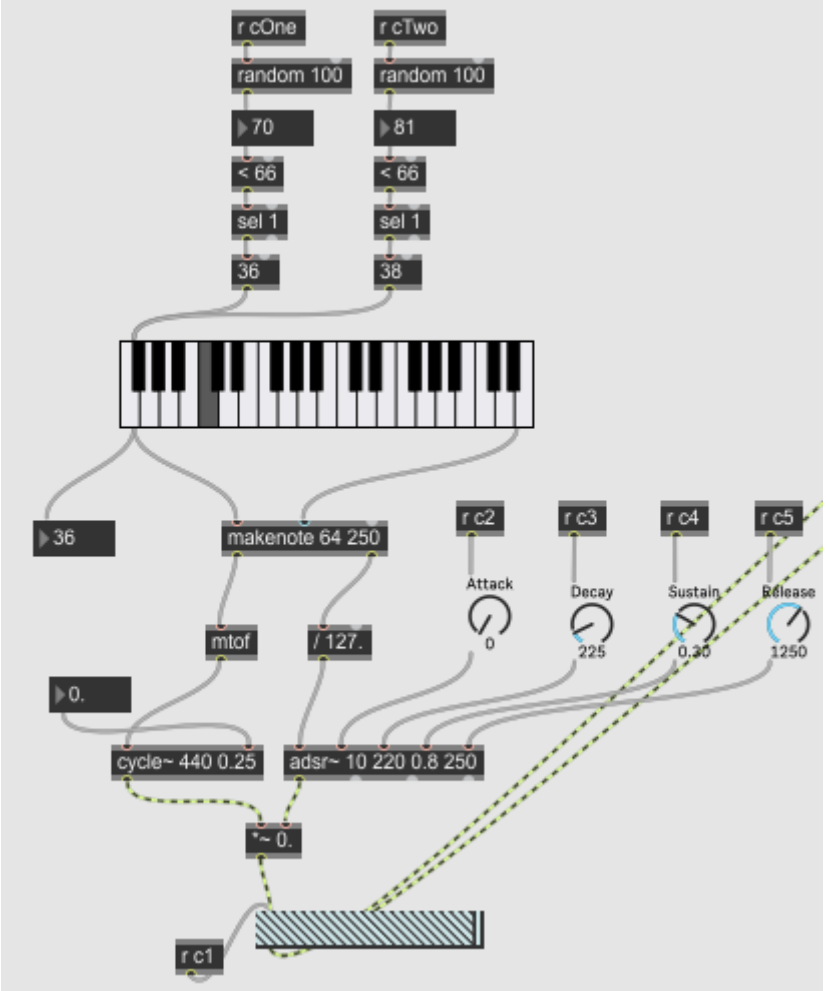
Κάθε ένα από τα μηνύματα που λαμβάνεται παράγει ένα τυχαίο αριθμό μεταξύ των τιμών 0 και 99. Αργότερα ελέγχεται το γεγονός αν αυτοί οι αριθμοί είναι μικρότεροι από συγκεκριμένες σταθερές. Αν ο αριθμός που έχει παραχθεί είναι μικρότερος από την σταθερά που έχει δοθεί, τότε θα σταλεί ο αριθμός midi νότας που αναγράφεται (36, 38) στο αντικείμενο «keyslider» (πιάνο) που φαίνεται στην Εικόνα 6.16. Πιο συγκεκριμένα για τις νότες ισχύουν:

Πίνακας 4: Στατιστικά στοιχεία δεδομένων του τρίτου μουσικού οργάνου.

Αριθμός Νότας	Αριθμός MIDI	Όνομα Νότας	Πιθανότητα Αναπαραγωγής
1	36	Ντο 2 (C2)	66%
2	38	Ρε 2 (D2)	66%

Το αντικείμενο «makenote» παράγει κάθε φορά νότα με ταχύτητα 64 και διάρκεια εξασθένησης τα 250 ms. Στη συνέχεια, με τη χρήση του «mtof» αντικειμένου (midi-to-frequency) πραγματοποιείται μετατροπή της MIDI νότας σε συχνότητα. Το «cycle~» παράγει ένα κύμα ήχου τύπου κύκλου (sine wave). Η ταλάντωση που παράγεται από το «cycle~» είναι μονοφωνική, που σημαίνει ότι παράγεται μία συχνότητα και ένας ήχος κάθε φορά.

Το ακουστικό αποτέλεσμα της κάθε νότας επηρεάζεται επίσης από το αντικείμενο «adsr». Τα σήματα που παράγουν τα cycle~ και adsr~ πολλαπλασιάζονται μεταξύ τους, στη συνέχεια περνάνε μέσα από το gain~ αντικείμενο, που μας επιτρέπει να πειράζουμε την ένταση του παραγόμενου ήχου. Κατόπιν το σήμα περνάει μέσα απ' το Reverb Subpatch για περαιτέρω επεξεργασία.

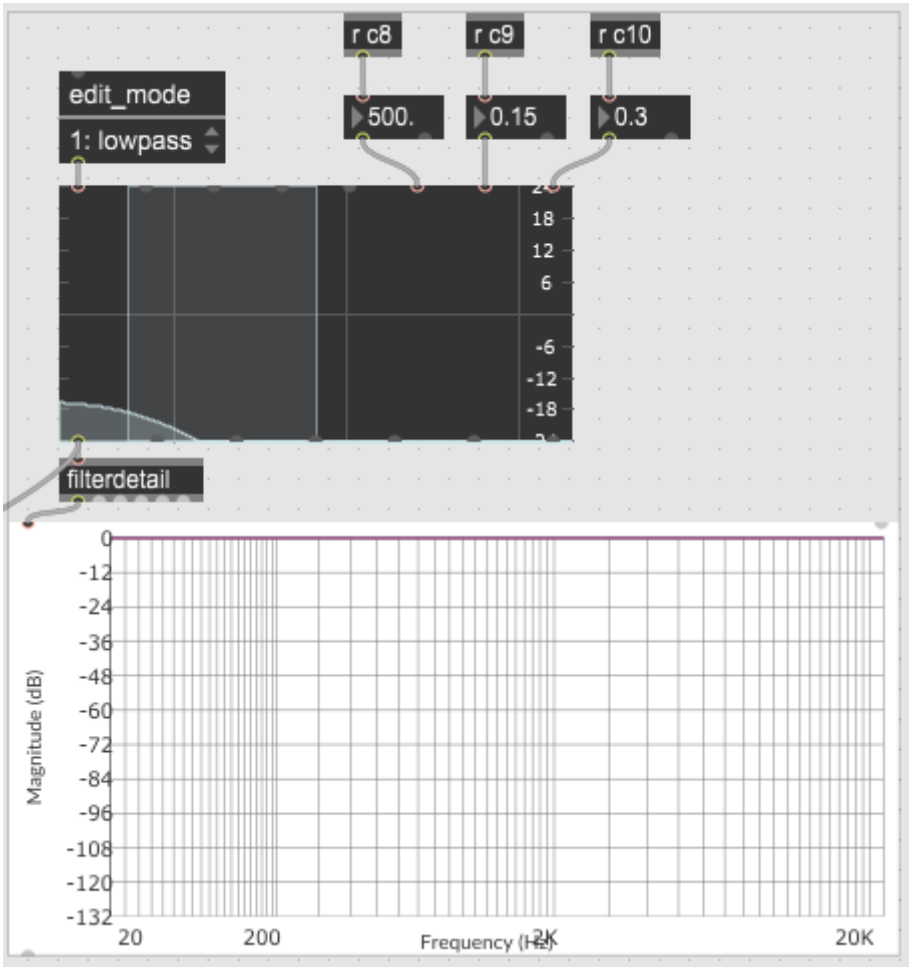


Εικόνα 6.16: Final Patch Third Instrument Piano

Και σε αυτή την περίπτωση, με τη χρήση του αντικειμένου random προκαλούμε τυχαιότητα στην αναπαραγωγή του ήχου. Κάθε νότα έχει συγκεκριμένη πιθανότητα αναπαραγωγής. Έτσι, με κάθε επανάληψη ο παραγόμενος ήχος είναι (σχεδόν) πάντα διαφορετικός.

6.4.6 Third Instrument Filter Graph

Στο συγκεκριμένο patcher έχουμε προσθέσει ένα φίλτρο, ώστε ο ήχος να μην ξεπερνάει τα επιτρεπτά όρια. Το αντικείμενο που μας παρέχει αυτό το φίλτρο ονομάζεται «filtergraph». Στη συγκεκριμένη περίπτωση χρησιμοποιούμε το lowpass filter, το οποίο μπορούμε να παραμετροποιήσουμε σε πραγματικό χρόνο, λαμβάνοντας τις τιμές από το Arduino και χρησιμοποιώντας τα μηνύματα c8, c9 και c10 που στέλνουμε, όπως είδαμε στην Ενότητα 6.3.2.3.



Εικόνα 6.17: Final Patch Third Instrument Filter

6.4.7 Αντήχηση (Reverb)

Όταν ο ήχος αντηχεί σε έναν χώρο, ανακλάται από τους τοίχους, τα δάπεδα και τα αντικείμενα και φτάνει πίσω στον ακροατή. Αντήχηση ονομάζεται το αποτέλεσμα από αυτές τις ανακλάσεις του ήχου σ' αυτό το χώρο.

Το reverb στο MAX/MSP και σε άλλα προγράμματα επεξεργασίας ήχου είναι ένα ηχητικό εφέ που προσομοιώνει την αντήχηση ήχων σε χώρους σε ανοικτό ή κλειστό περιβάλλον. Το reverb επιδιώκει να αναπαράγει αυτές τις αντανακλάσεις και να προσθέσει αυτές στον αρχικό ανεπεξέργαστο ήχο που αναπαράγεται. Γενικώς χρησιμοποιείται γιατί προσθέτει πλούσιο χαρακτήρα και βάθος σε μουσικά κομμάτια ή ηχογραφήσεις. Το reverb μπορεί να προστεθεί σε ηχογραφήσεις, μίξεις ήχου, μουσικά κομμάτια ή οποιαδήποτε άλλη ηχητική παραγωγή.

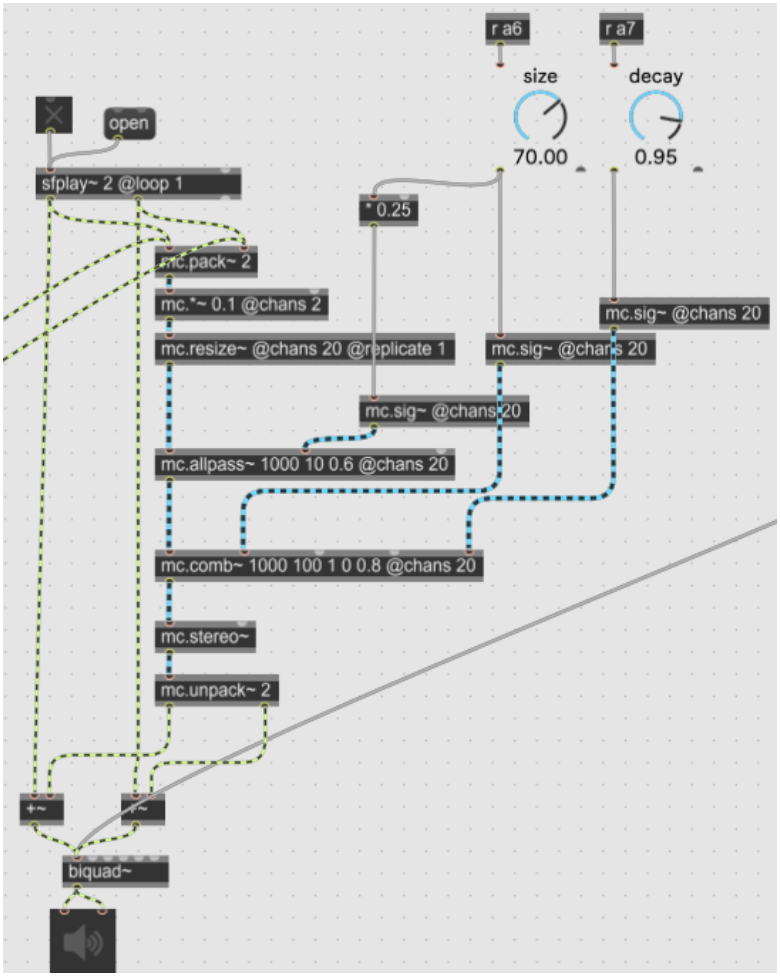
Στα πλαίσια του εφέ reverb, το «size» και το «decay» είναι δύο παράμετροι που επηρεάζουν τον χαρακτήρα της ηχητικής αντήχησης. Ας δούμε την κάθε μία ξεχωριστά:

1. Παράμετρος «size»: Αυτή η παράμετρος ελέγχει το μέγεθος του εικονικού χώρου ή αίθουσας αντήχησης που προσομοιώνεται. Αυξάνοντας την τιμή του size, δημιουργείται η εντύπωση ότι ο ήχος αντηχεί σε έναν μεγαλύτερο χώρο, ενώ μειώνοντας την τιμή, ο ήχος αντηχεί σε έναν μικρότερο χώρο. Η παράμετρος size επηρεάζει τη διάρκεια και την ένταση των αντανακλάσεων του ήχου.
2. Παράμετρος «decay»: Αυτή η παράμετρος ελέγχει τον χρόνο που απαιτείται για την εξασθένιση των αντανακλάσεων του ήχου. Αυξάνοντας την τιμή του decay, οι αντανακλάσεις διαρκούν περισσότερο, δημιουργώντας έναν μακρύτερο χρόνο αντήχησης. Μειώνοντας την τιμή του decay, οι αντανακλάσεις εξασθενούν πιο γρήγορα, δημιουργώντας έναν συντομότερο χρόνο αντήχησης. Η παράμετρος decay επηρεάζει τον χαρακτήρα του ήχου και το πόσο «μακριά» ακούγεται ο ήχος.

Στην Εικόνα 6.18 που ακολουθεί βλέπουμε το subpatcher που προσομοιώνει το εφέ της αντήχησης. Κάθε ένα από τα First, Second και Third Instrument Piano Subpatches στέλνει 2 outputs (ένα για κάθε κανάλι εξόδου του ήχου, αριστερό και δεξί). Αυτά φαίνονται να εισέρχονται από το αριστερό μέρος της εικόνας.

Το συγκεκριμένο subpatch μπορεί να λειτουργήσει και σαν αυτόνομο, αν στη θέση του αντικειμένου «sfplay~» προσθέσουμε ένα μουσικό κομμάτι ή ήχο κάνοντας drag-and-drop ή πιέζοντας το κουμπί «open», και κατόπιν ενεργοποιήσουμε το «toggle» που υπάρχει. Έτσι προσθέτουμε reverb στο κομμάτι που ανοίξαμε.

Εναλλακτικά, στέλνουμε input με διαφορετικό τρόπο, όπως στην περίπτωσή μας μέσω των cords που είναι συνδεδεμένα στο «mc.pack». Το αντικείμενο «ms.pack~ 2» δημιουργεί ένα «πακέτο» με δύο κανάλια ήχου. Τα κανάλια αυτά περιέχουν το output καθενός από τα First, Second και Third Instrument Piano Subpatches.



Εικόνα 6.18: Final Patch First Instrument Reverb

Στη συνέχεια μειώνουμε την ένταση του σήματος με τη χρήση του πολλαπλασιαστή «mc.*~0.1». Το αντικείμενο «mc.resize~» χρησιμοποιείται για την αλλαγή του αριθμού των καναλιών ενός πολυκάναλου σήματος ήχου. Με την παράμετρο «@chans 20», ορίζουμε τον αριθμό 20 για τα κανάλια εξόδου, ενώ με την παράμετρο «@replicate 1», κάθε κανάλι εξόδου θα αντιγράψει τα δεδομένα του αρχικού σήματος. Αυτό σημαίνει ότι το πολυκάναλο σήμα ήχου θα αποτελείται από πολλαπλά αντίγραφα των ίδιων δεδομένων κατά μήκος των καναλιών εξόδου.

Το αντικείμενο «mc.allpass~» παρέχει ένα allpass φίλτρο στην έξοδο του ήχου. Για τις δικές μας απαιτήσεις έχουμε παραμετροποιήσει τις τιμές των παραμέτρων που δέχεται το φίλτρο. Η πρώτη καθορίζει τη συχνότητα δειγματοληψίας του allpass φίλτρου. [41] Στη συγκεκριμένη περίπτωση, η συχνότητα δειγματοληψίας είναι 1000 Hz. Η δεύτερη τιμή αναφέρεται στο μέγεθος του allpass φίλτρου. Όσο μεγαλύτερη η τιμή, τόσο περισσότερες καθυστερήσεις εισάγονται από το φίλτρο. Στην περίπτωση μας, το μέγεθος είναι 10. Η Τρίτη τιμή καθορίζει τον συντελεστή ανάκλασης του allpass φίλτρου. Επηρεάζει την αναμετάδοση και τον χρόνο καθυστέρησης του σήματος, που στη συγκεκριμένη περίπτωση έχει τιμή 0.6. Και σε αυτή την περίπτωση η έξοδος αποτελείται από 20 κανάλια,

Το αντικείμενο «mc.comb~» παρέχει ένα comb φίλτρο στην έξοδο του ήχου. Για τις δικές μας απαιτήσεις έχουμε παραμετροποιήσει τις τιμές των παραμέτρων που δέχεται το φίλτρο. Η πρώτη καθορίζει τη χρονική καθυστέρηση (delay) του φίλτρου. Στην συγκεκριμένη περίπτωση, η χρονική

καθυστέρηση είναι 1000 ms. Η δεύτερη τιμή αναφέρεται στο μέγεθος (gain) του φίλτρου. Όσο μεγαλύτερη η τιμή, τόσο περισσότερες αντιγραφές του σήματος θα γίνουν πριν το συνδυάσουμε με το αρχικό σήμα. Στην περίπτωση μας, το μέγεθος είναι 100. Η τρίτη τιμή καθορίζει τον συντελεστή αναμετάδοσης των καθυστερήσεων του ήχου του φίλτρου. Η τιμή 1 σημαίνει ότι η αναμετάδοση είναι πλήρης. Η τέταρτη τιμή καθορίζει τη χρονική καθυστέρηση εκκίνησης του φίλτρου. Στην συγκεκριμένη περίπτωση, η χρονική καθυστέρηση εκκίνησης του φίλτρου είναι 0 ms. Η πέμπτη τιμή καθορίζει τον συντελεστή απόκρισης (feedback coefficient) του φίλτρου. Στη συγκεκριμένη περίπτωση ο συντελεστής απόκρισης είναι 0.8. Και σε αυτή την περίπτωση η έξοδος αποτελείται από 20 κανάλια.

Ταυτόχρονα, με τη χρήση των αντικειμένων «live.dial» που φαίνονται στο πάνω δεξιά μέρος της εικόνας, μπορούμε να εξάγουμε συγκεκριμένους αριθμούς. Τις τιμές αυτών των αριθμών λαμβάνουμε από το Arduino, χρησιμοποιώντας τα μηνύματα a6 και a7.

Στην περίπτωση του **size** επεξεργαζόμαστε τον αριθμό σε που λαμβάνουμε από το χρήστη με 2 τρόπους. Στο πρώτο μέρος, μικραίνουμε την τιμή του σήματος με τον πολλαπλασιαστή 0.25, ενώ με τη χρήση του αντικειμένου «mc.sig~» τον μετατρέπουμε σε σήμα 20 καναλιών και περνάμε το σήμα στη μεταβλητή delay του «mc.allpass» φίλτρου που φαίνεται στην Εικόνα 6.18. Στο δεύτερο μέρος μετατρέπουμε κατ' ευθείαν αυτό τον αριθμό σε σήμα 20 καναλιών και περνάμε το σήμα στην μεταβλητή delay του «mc.comb» φίλτρου.

Στην περίπτωση του **delay**, μετατρέπουμε κατ' ευθείαν τον αριθμό που λαμβάνουμε από το χρήστη σε σήμα 20 καναλιών και περνάμε το σήμα στην μεταβλητή του συντελεστή απόκρισης του «comb» φίλτρου.

Το αντικείμενο «mc.stereo~» χρησιμοποιείται για τη μετατροπή ενός εισερχόμενου σήματος σε στερεοφωνικό σήμα. Με το αντικείμενο «mc.unpack~ 2» γίνεται διαχωρισμός σε 2 κανάλια. Στη συνέχεια γίνεται πρόσθεση των σημάτων μεταξύ τους, όπως φαίνεται στη φωτογραφία. Ταυτόχρονα το σήμα περνάει από τα φίλτρα που αναλύσαμε στις Ενότητες 6.4.2, 6.4.4 και 6.4.6. Τέλος το σήμα περνάει από ένα biquad φίλτρο, και στη συνέχεια αναπαράγεται.

6.5 Επίλογος

Κλείνοντας το συγκεκριμένο κεφάλαιο παρατηρήσαμε ότι ακολουθώντας μια συγκεκριμένη μεθοδολογία και χρησιμοποιώντας απλές μεθόδους που θυμίζουν προγράμματα κλασσικού προγραμματισμού, μπορούμε να δημιουργήσουμε ένα μεγάλο αριθμό διαφορετικών υποπρογραμμάτων. Αυτά τα υποπρογράμματα έχουν τη δική τους ανεξάρτητη λειτουργία, ωστόσο συνδυασμένα μεταξύ τους μπορούν να δημιουργήσουν ένα ευρύτερο ενιαίο πρόγραμμα.

Για τη παραγωγή μουσικής μπορούν να χρησιμοποιηθούν διαφορετικά και πιο δημοφιλή προγράμματα. Ωστόσο αυτά χρησιμοποιούνται για διαφορετικούς λόγους και παρέχουν διαφορετικές λειτουργίες. Το FL Studio για παράδειγμα, είναι παρέχει ένα περιβάλλον σύνθεσης μουσικής, που διαθέτει ένα γραφικό περιβάλλον με πλούσιες δυνατότητες σύνθεσης, επεξεργασίας ήχου και ενορχήστρωσης και χρησιμοποιεί μία πιο παραδοσιακή προσέγγιση με μενού και πλήκτρα πλοήγησης.

Η διαφορά είναι ότι αυτού του τύπου οι εφαρμογές χρησιμοποιούνται περισσότερο για την δημιουργία, επεξεργασία και μίξη μουσικών κομματιών, ενώ το MAX/MSP μπορεί να χρησιμοποιηθεί για την επεξεργασία μουσικής σε πραγματικό χρόνο. Τα εργαλεία που παρέχει, προσφέρουν τόσο μεγάλη ευελιξία και παραμετροποίηση, που με αυτά μπορούμε να παράγουμε μια τεράστια ποικιλία μουσικών ειδών.

Κεφάλαιο 7ο: 3D Printing

7.1 Γενικά

Στο συγκεκριμένο κεφάλαιο θα μιλήσουμε για το 3D printing. Στην συνέχεια θα μιλήσουμε για τα μέρη ενός 3D εκτυπωτή, τα πιο συνηθισμένα υλικά που χρησιμοποιούνται για την 3D εκτύπωση και τις χρήσεις αυτών των αντικειμένων, και θα μιλήσουμε για τη διαδικασία ακολουθήσαμε προκειμένου να εκτυπώσουμε ένα 3D αντικείμενο, για να χρησιμοποιήσουμε ως βάση για το music box που δημιουργήσαμε.

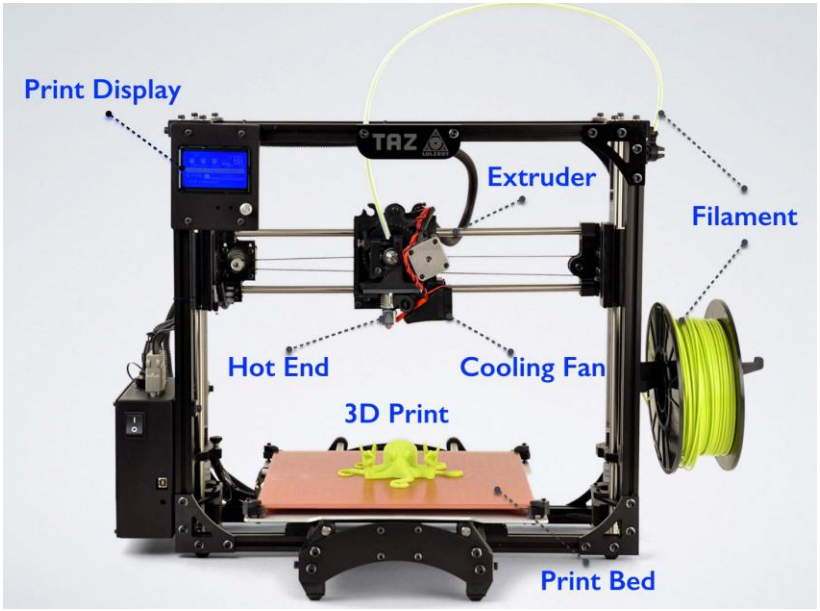
7.2 Τι είναι το 3D printing;

3D εκτύπωση είναι η διαδικασία δημιουργίας τρισδιάστατων φυσικών αντικειμένων, τα οποία προέρχονται από ψηφιακά μοντέλα. Η συσκευή που δημιουργεί αυτά τα φυσικά αντικείμενα είναι ο 3D εκτυπωτής. Αντί να εκτυπώνει επίπεδες εικόνες ή κείμενο σε χαρτί, ένας 3D εκτυπωτής κατασκευάζει αντικείμενα στον πραγματικό τρισδιάστατο χώρο. Αυτό είναι εφικτό, καθώς ο εκτυπωτής κατασκευάζει το αντικείμενο «στρώμα πάνω σε στρώμα», χρησιμοποιώντας συνήθως υλικά όπως πλαστικό ή μέταλλο.

Οι 3D εκτυπωτές είναι χρήσιμοι σε οικιακούς χρήστες και σε επαγγελματίες. Χρησιμοποιούνται σε διάφορους τομείς, όπως η αρχιτεκτονική, η ιατρική, η εκπαίδευση, η κατασκευή πρωτοτύπων, πολύπλοκων γεωμετρικών μορφών, μοντέλα αρχιτεκτονικής. [48] Με αυτόν μπορούν επίσης να κατασκευαστούν κοσμήματα, μινιατούρες μεγάλων αντικειμένων, φιγούρες ή ανταλλακτικά και μικροεξαρτήματα πάσης φύσεως.

7.3 Τα μέρη ενός 3D εκτυπωτή.

Για να κατανοήσουμε τα μέρη ενός 3D εκτυπωτή θα πρέπει να κάνουμε εικόνα τους τρεις άξονες εκτύπωσης. Οι άξονες x, y αφορούν την κίνηση που γίνεται οριζόντια, ενώ ο άξονας z αφορά την κίνηση στον κατακόρυφο άξονα. Κάθε άξονας ελέγχεται από ένα μοτέρ.



Εικόνα 7.1: Τα μέρη ενός 3D εκτυπωτή [49]

Ο extruder είναι το εξάρτημα που τραβάει ή σπρώχνει το νήμα ανάλογα με τις ανάγκες τις εκτύπωσης. Το νήμα προωθείται προς το Hot End, το ζεστό μέρος, όπου θερμαίνεται από το θερμαινόμενο block σε κατάλληλες θερμοκρασίες (170-240c) μέχρι να λιώσει. Το λιωμένο νήμα βγαίνει από το στόμιο σε μικρότερη διάμετρο 0,3-0,6 χιλιοστά και τοποθετείται πάνω στο πάτωμα της εκτύπωσης. Ο έλεγχος της διαδικασίας εκτύπωσης γίνεται από τον υπολογιστή και η σύνδεση του μηχανήματος και του υπολογιστή γίνεται μέσα από την κεντρική πλακέτα του εκτυπωτή. [50]

7.4 Υλικά 3D εκτύπωσης

Υπάρχουν διάφορα υλικά με τα οποία μπορεί να πραγματοποιηθεί εκτύπωση. Μερικά από αυτά είναι:

- Πολυμερές υλικό (PLA) το οποίο κατασκευάζεται από ανανεώσιμους πόρους (αραβόσιτο, σακχαρότευτλα). Το υλικό αυτό είναι μη τοξικό, βιοδιασπώμενο και δεν επιβαρύνει το περιβάλλον.
- Πολυμερές πλαστικό (ABS) πολύ ανθεκτικό υλικό που χρησιμοποιείται σε αρκετές εφαρμογές της βιομηχανίας. Είναι και αυτό μη τοξικό, η θέρμανση του θέλει κατά προτίμηση αερισμένο δωμάτιο.
- Άλλα υλικά, όπως Nylon, Wood, Metal, Flex και άλλα. Αυτά παρουσιάζουν ποικίλα χαρακτηριστικά όπως αντοχή, όψη και ελαστικότητα. Το nylon και το flex απαιτούν υψηλότερες θερμοκρασίες και μεγαλύτερη εξειδίκευση. Είναι πιο εύκαμπτα και κατάλληλα για συγκεκριμένα σχέδια. Τα wood και metal προέρχονται από ανακυκλώσιμο ξύλο και ρινίσματα μετάλλου αντίστοιχα και μας δίνουν την εικόνα ξύλινου ή μεταλλικού αντικειμένου.

Άλλα υλικά που υπάρχουν είναι επίσης τα HIPS, PC, TPU, TPE, PETG, ASA, PP, Glass Fiber, SLA.

7.5 Tinkercad

Υπάρχουν πολλές εφαρμογές 3D εκτύπωσης, κάθε μια από τις οποίες έχει τα δικά της χαρακτηριστικά και πλεονεκτήματα. Ορισμένες από αυτές είναι οι Ultimaker Cura, PrusaSlicer, Simplify3D, Autodesk Fusion 360, Tinkercad. Κάθε μια από αυτές έχει τα δικά της χαρακτηριστικά, καλύπτει διαφορετικές ανάγκες και απευθύνεται σε άτομα με διαφορετικά επίπεδα εμπειρίας.

Εμείς για τις ανάγκες τις εργασίας μας χρησιμοποιήσαμε το Tinkercad. Αυτή είναι μια δωρεάν online εφαρμογή σχεδίασης 3D που μας επιτρέπει να δημιουργούμε απλά και εύκολα 3D μοντέλα. Αποτελεί ένα ιδανικό εργαλείο για αρχάριους που ενδιαφέρονται να εξοικειωθούν με τη 3D σχεδίαση. Η διεπαφή του Tinkercad είναι απλή και φιλική προς τον χρήστη, παρέχει έτοιμα τρισδιάστατα αντικείμενα, καθιστώντας εύκολη τη δημιουργία και επεξεργασία 3D μοντέλων.

Ένα από τα μεγάλα πλεονεκτήματα του Tinkercad είναι η δυνατότητα να εξάγουμε τα 3D μοντέλα που σχεδιάσαμε ως αρχεία STL, τα οποία μπορούμε να χρησιμοποιήσουμε για να εκτυπώσουμε σε εκτυπωτές 3D, το οποίο είναι και αυτό που έχουμε ως στόχο να κάνουμε. [51]

7.6 Modeling

Τα βασικά δύο σχήματα που χρησιμοποιήσαμε από την έτοιμη συλλογή αντικειμένων που προσφέρει το Tinkercad είναι ο κύβος και η μισή σφαίρα (Half Sphere). Με διάφορους μετασχηματισμούς αυτών των σχημάτων (περιστροφή, μεγέθυνση, στρέβλωση, προσθήκη σχημάτων μεταξύ τους ή αφαίρεση αυτών) καταφέραμε να δημιουργήσουμε το τελικό μοντέλο που χρησιμοποιήσαμε στην εργασία μας.

Η κατασκευή χωρίζεται σε τρία μέρη: το κάτω μέρος (τη βάση του μοντέλου), το πάνω μέρος (σαν καπάκι), και ένα επιπλέον σχήμα που δημιουργήθηκε για να προστατεύει τα καλώδια που προεξέχουν. Στις επόμενες ενότητες περιγράφουμε όλους τους μετασχηματισμούς που πραγματοποιήσαμε, με σκοπό να δημιουργήσουμε το τελικό αντικείμενο. Οι διαστάσεις που αναφέρονται στις επόμενες ενότητες είναι όλες μετρημένες σε χιλιοστά, και οι εικόνες που ακολουθούν (στιγμιότυπα οθόνης) έχουν ληφθεί από την online εφαρμογή Tinkercad.

Σημείωση: Σε ορισμένα από τα σημεία στα οποία καταγράφονται οι διαστάσεις των σχημάτων, οι μετρήσεις των διαστάσεων δεν είναι ακριβείς, καθώς έχουν πραγματοποιηθεί στρογγυλοποιήσεις. Οι διαστάσεις αυτές δεν έχουν στρογγυλό αριθμό, λόγω της καμπυλότητας των σχημάτων.

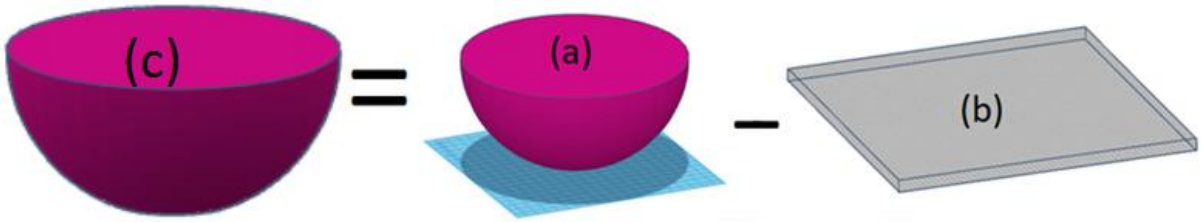
7.6.1 Δημιουργία κάτω μέρους του βασικού μοντέλου

Το σχήμα που δημιουργούμε στη συγκεκριμένη ενότητα (σχήμα g) αποτελεί το σχήμα που θα στηρίζεται πάνω στο δάπεδο στο οποίο θα είναι τοποθετημένο το music box. Για τη δημιουργία του χρησιμοποιούμε δύο ημισφαίρια σχήματα διαφορετικών διαστάσεων, με σκοπό να τα αφαιρέσουμε μεταξύ τους ώστε να δημιουργήσουμε το επιθυμητό αποτέλεσμα.

Επιπλέον έχουμε δημιουργήσει μια επίπεδη βάση στο κάτω μέρος (αντί για την κυκλική που ήδη υπάρχει) ώστε το αντικείμενο να μένει σταθερό. Με την προσθήκη βάρους πάνω στο αντικείμενο (Arduino και breadboard), είναι πολύ πιθανό το κέντρο βάρους να μετατοπίζεται, με αποτέλεσμα το αντικείμενο να μην μένει σταθερό, γεγονός που θέλουμε να αποφύγουμε.

7.6.1.1 Σχεδιασμός εξωτερικού σχήματος, με βάση στο κάτω μέρος

Σχεδιάζουμε το εξωτερικό μέρος του σχήματος που αποτελεί τη βάση για το music box, από το οποίο αργότερα θα αφαιρέσουμε το εσωτερικό. Από το σχήμα (a) αφαιρούμε το (b) με σκοπό να υπάρξει μια επίπεδη επιφάνεια του σχήματος, όπως φαίνεται στο σχήμα (c) της ακόλουθης εικόνας.



Εικόνα 7.2: Μετασχηματισμός/Δημιουργία σχήματος (c). Στήριγμα του music box.

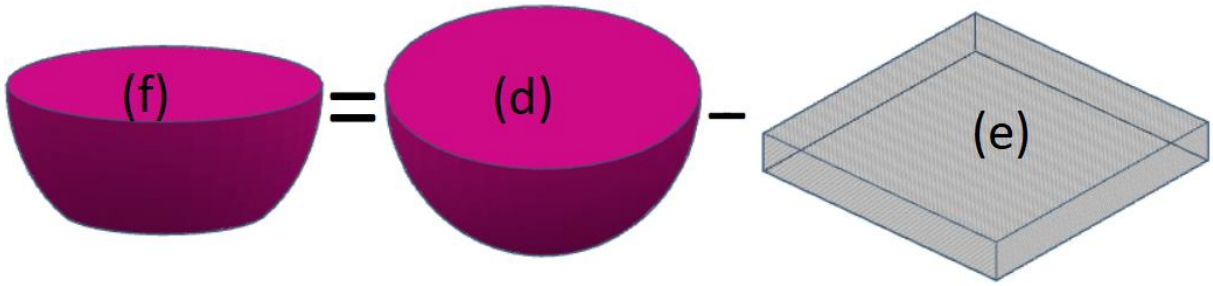
(a) ← 200x200x100, (b) ← 200x200x7.5

(c) ← (a) – (b)

(c) ← 200x200x92.5

7.6.1.2 Σχεδιασμός εσωτερικού σχήματος, με τη βάση στο κάτω μέρος, για να είναι πιο βαρύ το σχήμα στο κάτω μέρος

Με την ίδια διαδικασία σχεδιάζουμε το εσωτερικό σχήμα που προορίζεται για αφαίρεση από το σχήμα (c) της εικόνας 7.2.



Εικόνα 7.3: Μετασχηματισμός/Δημιουργία σχήματος (f). Σχήμα που προορίζεται για μετασχηματισμό άλλου σχήματος.

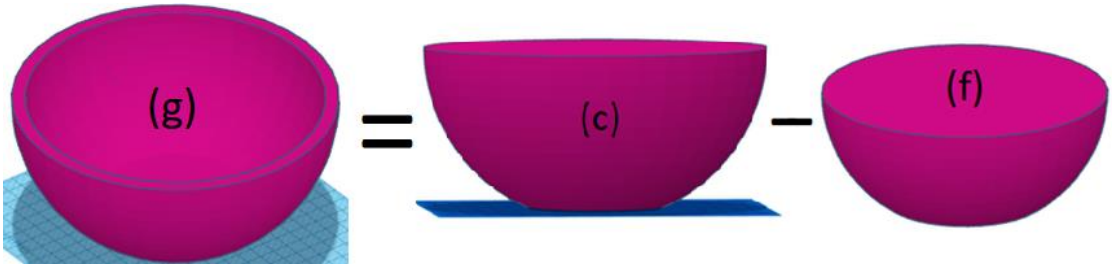
(d) ← 185x185x100, (e) ← 185x185x25

(f) ← (d) - (e)

(f) ← 185x185x75

7.6.1.3 Μετασχηματισμός για τη δημιουργία του κάτω μέρους του βασικού μοντέλου

Εν τέλει αφαιρούμε τα σχήματα (c) και (f) των δύο προηγούμενων ενοτήτων για να δημιουργήσουμε το κούφιο ημισφαίριο που φαίνεται στην ακόλουθη εικόνα.



Εικόνα 7.4: Μετασχηματισμός/Δημιουργία σχήματος (g), στήριγμα του music box.

(g) ← (c) - (f)

(g) ← 200x200x92.5

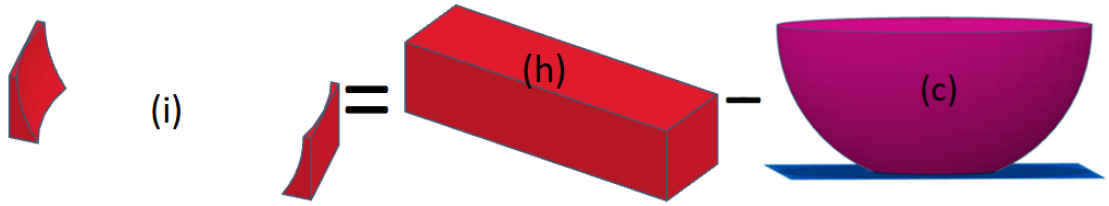
7.6.2 Δημιουργία πάνω μέρους του βασικού μοντέλου

Το σχήμα που μοντελοποιούμε στη συγκεκριμένη ενότητα ακουμπάει πάνω από σχήμα (g) της προηγούμενης ενότητας. Έχει σχεδιαστεί με τρόπο, ώστε να μπορεί να στηρίζεται το Arduino και το breadboard. Ακολουθεί μια ακολουθία μετασχηματισμών σχημάτων, που αποσκοπούν στην δημιουργία σχήματος που θα προσαρμόζεται πάνω στο σχήμα (x) της επόμενης ενότητας 7.6.3, ακριβώς όπως ένα καπάκι.

Αρχικά σχεδιάζουμε το βοηθητικό σχήμα (j), που μεταξύ άλλων, θα συμβάλει στη δημιουργία του σχήματος (x). Στη συνέχεια θα σχεδιάσουμε τα σχήματα (q) και (v). των οποίων η προσθήκη θα μας οδηγήσει στη δημιουργία του βασικού σχήματος (y) του πάνω μέρους του music box.

7.6.2.1 Σχεδιασμός βοηθητικού σχήματος (j)

Με τους ακόλουθους μετασχηματισμούς δημιουργούμε το σχήμα (j).

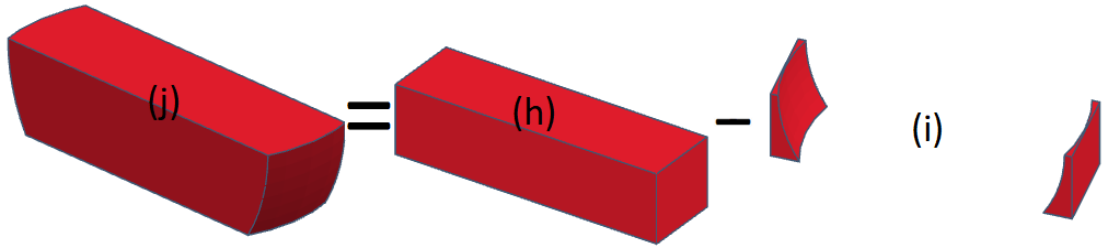


Εικόνα 7.5: Μετασχηματισμός/Δημιουργία σχήματος (i)

(h) ← 200x56x50

(i) ← (h) - (c)

(i) ← 200x56x50



Εικόνα 7.6: Μετασχηματισμός/Δημιουργία σχήματος (j), βοηθητικό σχήμα 1.

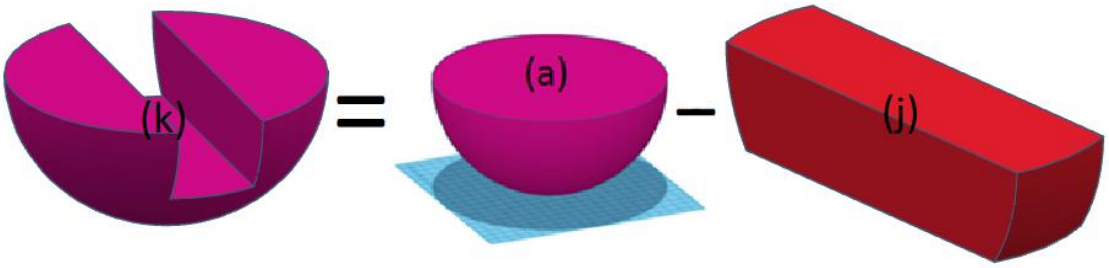
(j) ← (h) - (i)

(j) ← 200x56x50

7.6.2.2 Δημιουργία βοηθητικού σχήματος (q)

Οι ακόλουθοι μετασχηματισμοί αποσκοπούνε στην δημιουργία του σχήματος (q) που θα μας βοηθήσει στην δημιουργία του σχήματος (u). Σε μια τροποποιημένη μορφή του σχήματος της συγκεκριμένης ενότητας, στηρίζεται πάνω το Arduino

Δημιουργούμε ένα κενό σημείο, στο οποίο θα τοποθετούνται τα Arduino και breadboard.

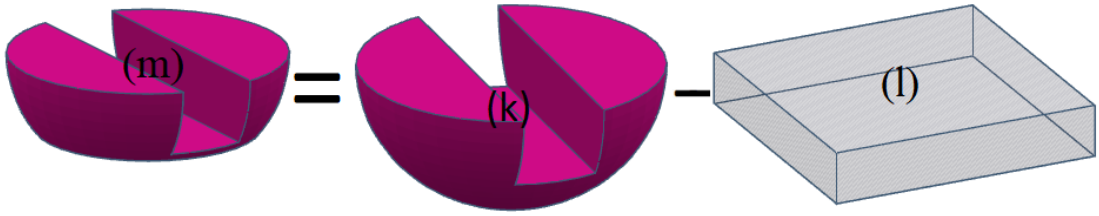


Εικόνα 7.7: Μετασχηματισμός/Δημιουργία σχήματος (k)

(k) ← (a) - (j)

(k) ← 200x191.25x92.5

Αφαιρούμε το κάτω μέρος του σχήματος για να μειώσουμε το βάρος και να αποφύγουμε intersect με το σχήμα στο οποίο πρόκειται να τοποθετηθεί πάνω.



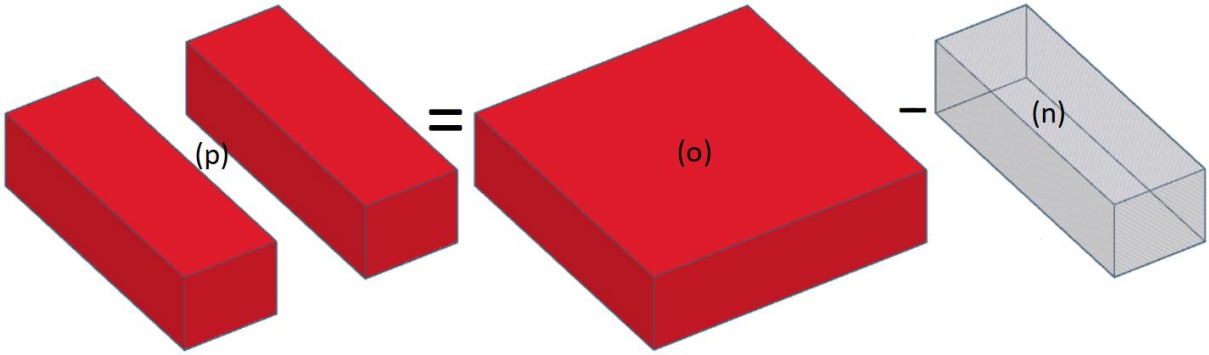
Εικόνα 7.8: Μετασχηματισμός/Δημιουργία σχήματος (m)

(l) ← 200x200x37.5

(m) ← (k) - (l)

(m) ← 200x191.25x55

Δημιουργία σχήματος για επιπλέον αφαίρεση από το σχήμα (m).



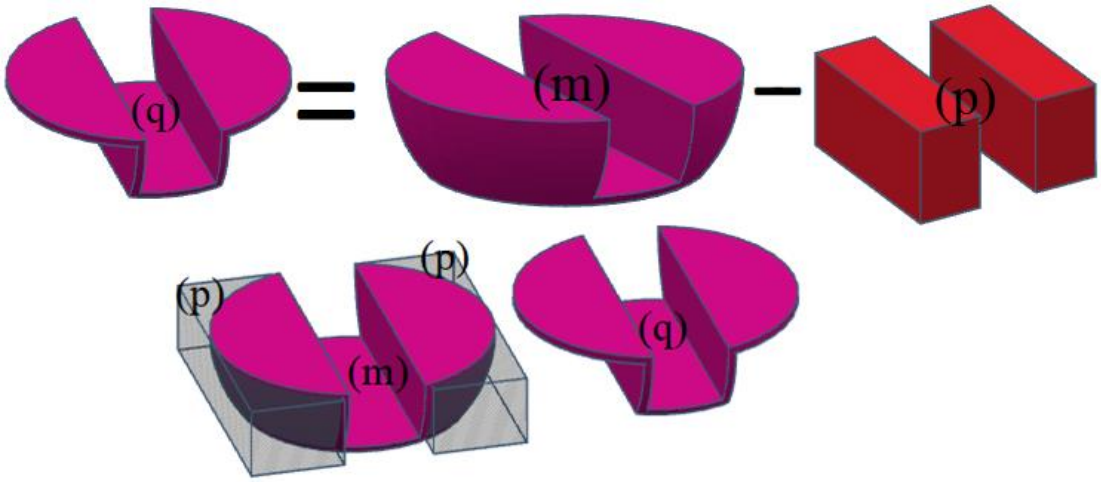
Εικόνα 7.9: Μετασχηματισμός/Δημιουργία σχήματος (p)

(n) ← 200x66x45, (o) 200x200x45

(p) ← (o) - (n)

(p) ← 200x200x45

Αφαίρεση μέρους του σχήματος (m) για να μειώσουμε το βάρος και να αποφύγουμε intersect με το σχήμα στο οποίο πρόκειται να τοποθετηθεί πάνω.



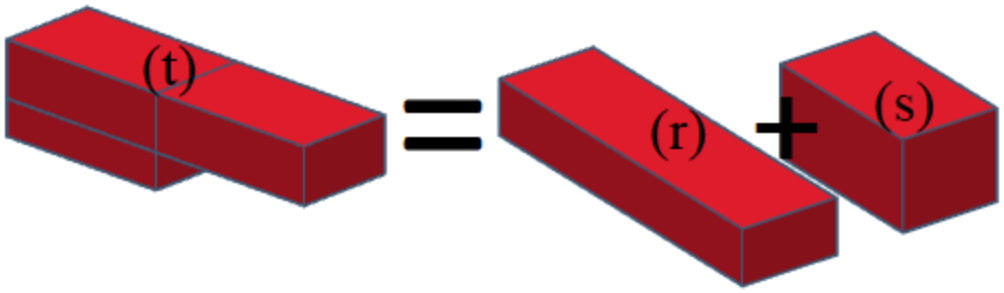
Εικόνα 7.10: Μετασχηματισμός/Δημιουργία σχήματος (q)

(q) ← (m) - (p)

(q) ← 200x191.25x55

7.6.2.3 Δημιουργία βοηθητικού σχήματος (v)

Στο μοντέλο που δημιουργούμε στην συγκεκριμένη ενότητα στηρίζεται πάνω το breadboard. Ταυτόχρονα δημιουργούμε μια τρύπα, μέσα από την οποία θα εξέρχεται το καλώδιο τροφοδοσίας του Arduino, που θα συνδέεται με τον Η/Υ στον οποίο δουλεύουμε.

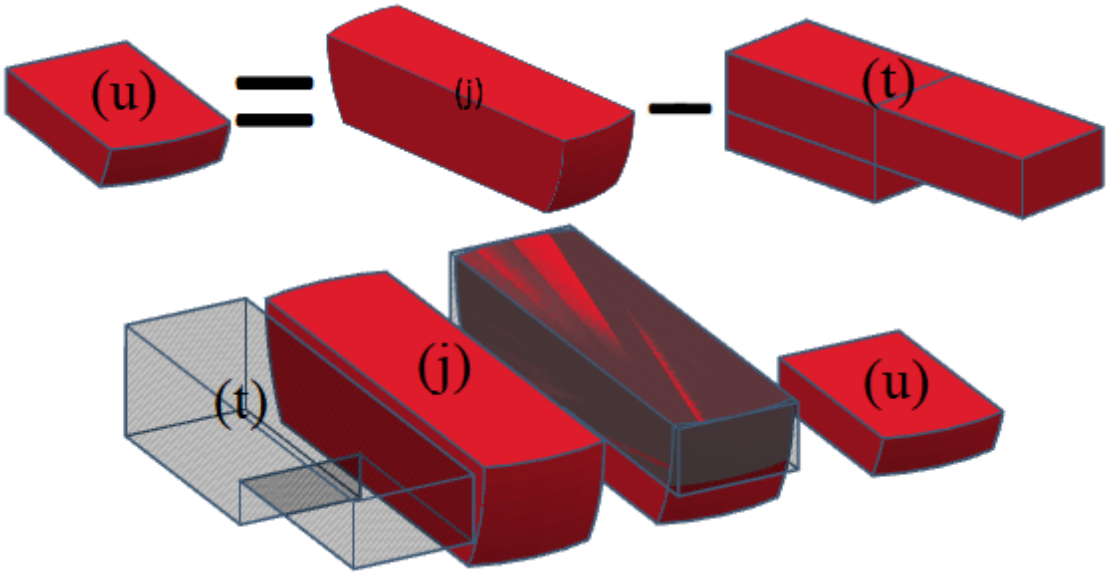


Εικόνα 7.11: Μετασχηματισμός/Δημιουργία σχήματος (t)

(r) ← 200x56x20, (s) ← 100x56x50

(t) ← (r) + (s)

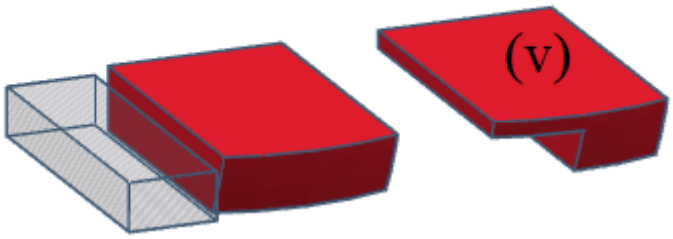
(t) ← 200x56x50



Εικόνα 7.12: Μετασχηματισμός/Δημιουργία σχήματος (u)

(u) ← (j) - (t)

(u) ← 98x56x30



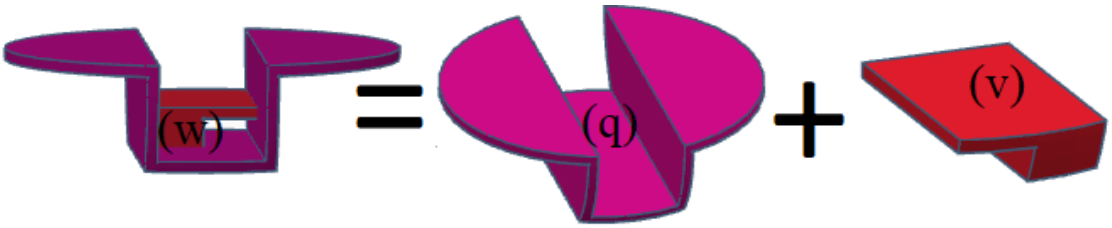
Εικόνα 7.13: Μετασχηματισμός/Δημιουργία σχήματος (v)

(v) ← (u) – (r) modified

(v) ← 98x56x30

7.6.2.4 Μετασχηματισμός για τη δημιουργία του κάτω μέρους του βασικού μοντέλου.

Συναρμολόγηση βάσης πάνω στην οποία στηρίζεται το Arduino και το breadboard.

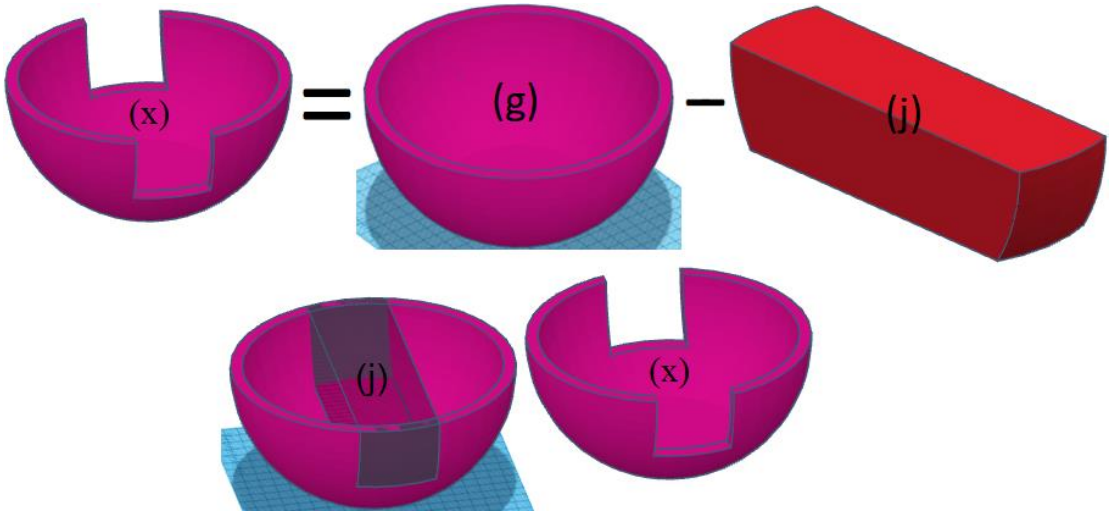


Εικόνα 7.14: Μετασχηματισμός/Δημιουργία σχήματος (w)

(w) ← (q) + (v)

(w) ← 200x191.25x55

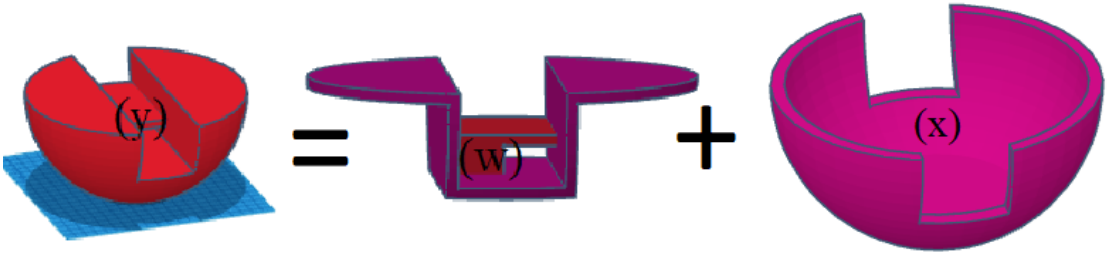
7.6.3 Συναρμολόγηση βασικού σχήματος



Εικόνα 7.15: Μετασχηματισμός/Δημιουργία σχήματος (x)

(x) ← (g) - (j)

(x) ← 200x191.25x92.25



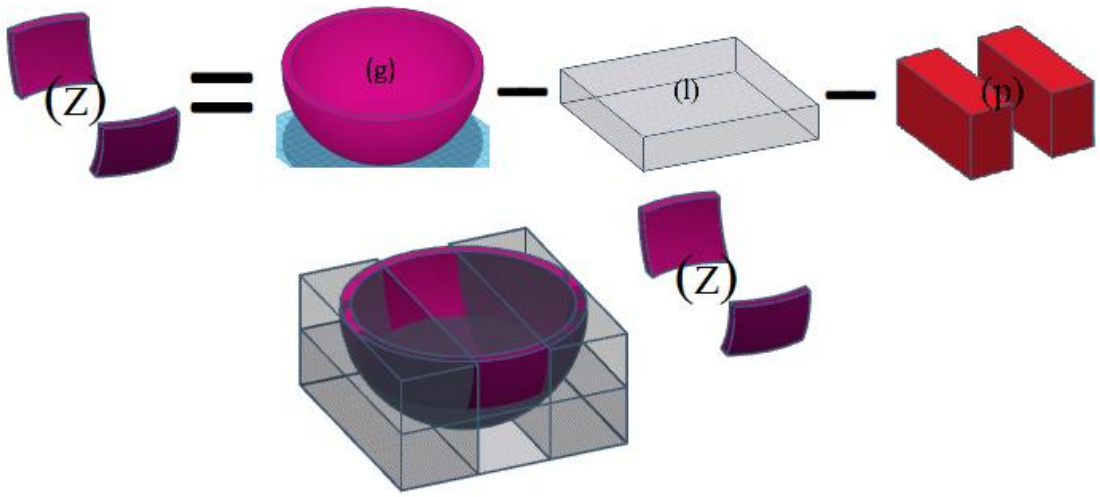
Εικόνα 7.16: Μετασχηματισμός/Δημιουργία σχήματος (y)

(y) ← (w) + (x)

(y) ← 200x191.25x92.25

7.6.4 Δημιουργία επιπλέον σχήματος

Το συγκεκριμένο σχήμα το σχεδιάζουμε κατ' αρχάς για να είναι πιο όμορφο το music box για τον τελικό χρήστη, και επιπλέον για να καταφέρουμε να κρύψουμε τα καλώδια , ώστε να είναι πιο εύκολο στην χρήση και ταυτόχρονα να αποφύγουμε οποιαδήποτε ενδεχόμενη αποσύνδεση κατά τη χρήση του music box.



Εικόνα 7.17: Μετασχηματισμός/Δημιουργία σχήματος (z)

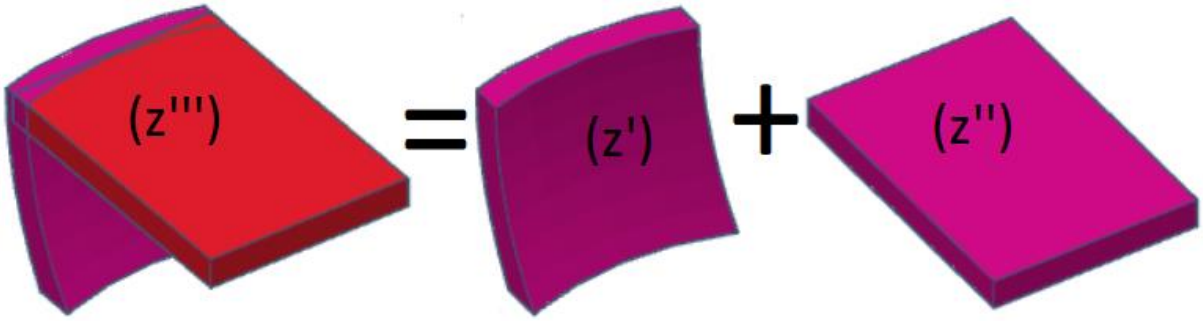
(z) ← (g) - (l) - (p) (στη συγκεκριμένη περίπτωση έχουμε θέσει το ύψος του p σε 100)

(z) ← 200x56x50

Εικόνα 7.18: Σχήμα (z’)

(z’) ← Κρατάμε το ένα απ’ τα δύο μισά του σχήματος (z).

(z’) 30x66x55



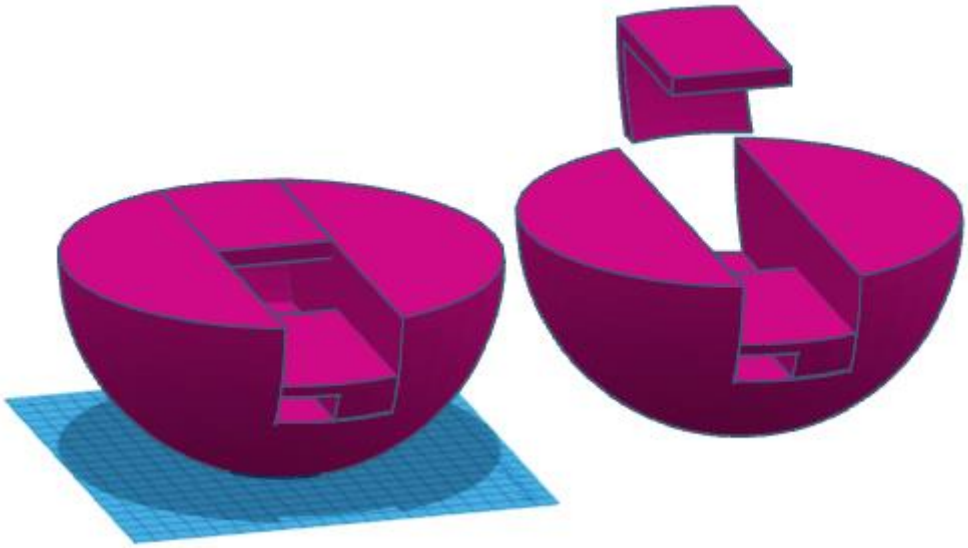
Εικόνα 7.19: Μετασχηματισμός/Δημιουργία σχήματος (z’’’)

(z’’) ← 80x56x7.5

(z’’’) ← (z’) + (z’’)

(z’’’) ← 82.5x56x50

7.6.5 Συναρμολόγηση τελικού σχήματος



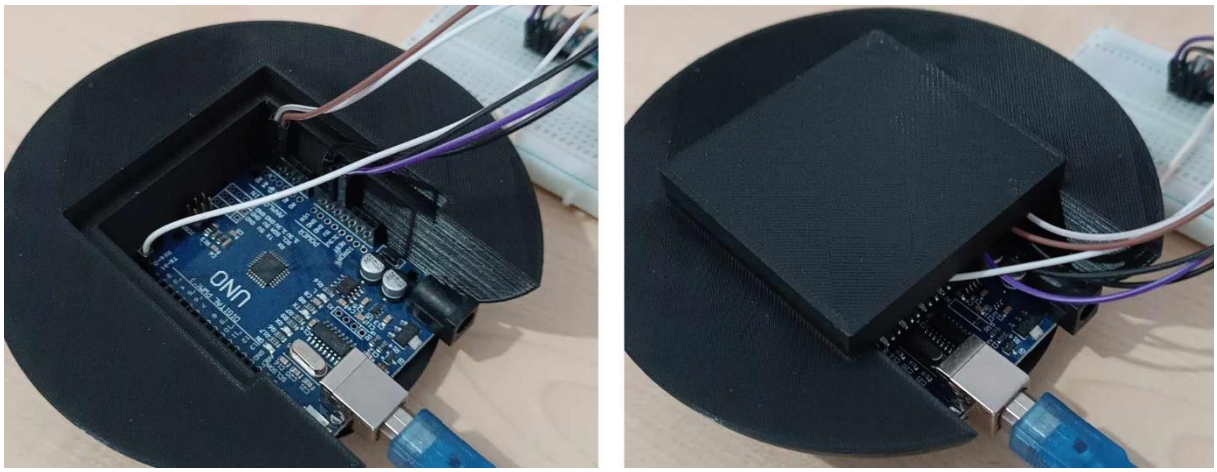
Εικόνα 7.20: Τελικό μοντέλο/βάση για το music box.

Final object ← (z'') + (y)

Final object ← 200x191.25x92.25

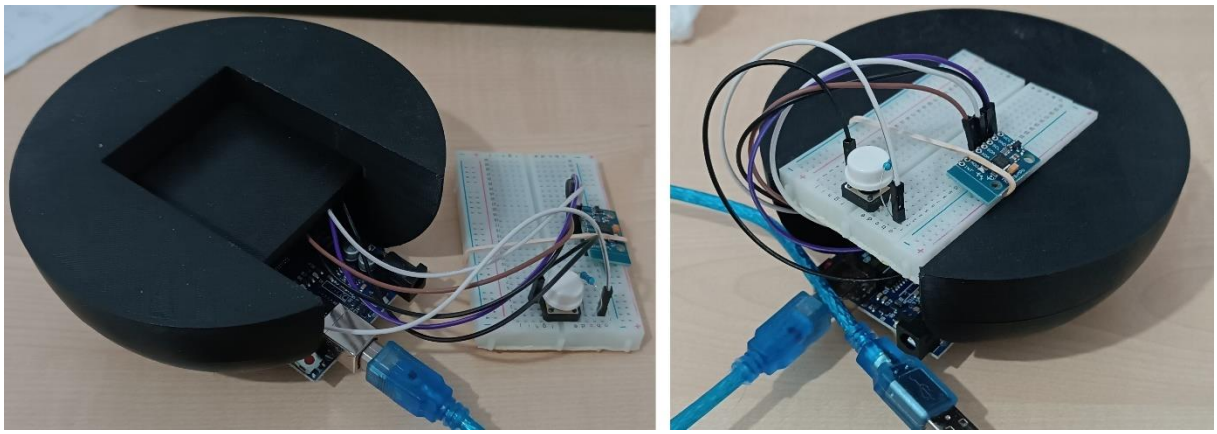
7.6.6 Τοποθέτηση Arduino και Breadboard στο 3D printed σχήμα.

Αρχικά τοποθετούμε το Arduino στο κάτω μέρος του μοντέλου που εκτυπώσαμε, και στη συνέχεια προσθέτουμε το προστατευτικό καπάκι, όπως φαίνεται στην Εικόνα 7.21.



Εικόνα 7.21: Τοποθέτηση Arduino στο 3D εκτυπωμένο σχήμα

Στη συνέχεια συναρμολογούμε το πάνω μέρος του 3D εκτυπωμένου σχήματος και μετέπειτα τοποθετούμε το breadboard πάνω απ' αυτό, όπως φαίνεται στην Εικόνα 7.22



Εικόνα 7.22: Τοποθέτηση breadboard στο 3D εκτυπωμένο σχήμα.

Αξίζει να σημειωθεί ότι η εκτύπωση πραγματοποιήθηκε τμηματικά, ώστε να αποφευχθούν σφάλματα κατά τη διάρκεια της εκτύπωσης, καθώς και να βοηθηθούμε εμείς κατά τη συναρμολόγηση του σχήματος. Όπως αναφέρουμε και στην Ενότητα 8.2 το τελικό σχήμα διαφέρει από το εικονιζόμενο της προηγούμενης ενότητας, λόγω μεγάλου όγκου και κόστους.

7.7 Επίλογος

Με τη συγκεκριμένη διαδικασία μάθαμε ότι με βασικές γνώσεις γραφικών υπολογιστών, ότι η 3D εκτύπωση μπορεί να γίνει πολύ απλή και εύκολη. Δεν απαιτείται η χρήση κάποιου επί πληρωμή λογισμικού που να απαιτούν επαγγελματικού επιπέδου γνώσεις. Μπορεί η διαδικασία που εκτελέσαμε να φαίνεται περίπλοκη, ωστόσο αυτό που κάναμε ήταν απλώς η δημιουργία των σχημάτων (x), (q), (v) και (z’’’) και ένωση αυτών μεταξύ τους.

Κεφάλαιο 8ο: Συμπεράσματα

8.1 Γενικά

Φτάνοντας στο τέλος της εκπόνησης της συγκεκριμένης πτυχιακής εργασίας, έχουμε φτάσει σε σημείο οπού μπορούμε να εκφέρουμε κάποια συμπεράσματα, να αναφέρουμε τις προκλήσεις που αντιμετωπίσαμε, καθώς και να αναπτύξουμε τυχόν μελλοντικές βελτιώσεις με τις οποίες μπορεί να ασχοληθούμε μελλοντικά, ή να προτείνουμε σε άτομα που τους ενδιαφέρει το συγκεκριμένο αντικείμενο και θα ήθελαν να ασχοληθούν περεταίρω.

Το συγκεκριμένο αντικείμενο κατά τη γνώμη μας έχει μεγάλη πρακτική αξία. Αποτελεί ένα πολύ καλό πρότυπο, στο οποίο αν πραγματοποιηθούν μερικές αναβαθμίσεις και τροποποιήσεις, θα αποκτήσει μεγαλύτερη χρηστική αξία. Επιπλέον θεωρούμε ότι στα χέρια κάποιου ανθρώπου με εξειδικευμένες μουσικές γνώσεις μπορεί να προσφέρει πολλές επιπλέον επιλογές, τις οποίες κατά τη διάρκεια εκπόνησης της πτυχιακής δεν είχαμε την ευκαιρία να ανακαλύψουμε.

8.2 Προκλήσεις

Οι προκλήσεις που αντιμετωπίσαμε κατά τη διάρκεια εκπόνησης της πτυχιακής είναι οι εξής:

Χρειάστηκε ένα μεγάλο διάστημα μελέτης των διαφορετικών εφαρμογών με τις οποίες είχαμε μηδενική εμπειρία στο παρελθόν. Μελετήσαμε συστηματικά το λογισμικό MAX/MSP. Αυτό, παρ' ότι θυμίζει σε μεγάλο βαθμό τις συμβατικές γλώσσες προγραμματισμού διαφέρουν από αυτές σε μεγάλο βαθμό. Το θετικό είναι ότι υπάρχει μια μεγάλη κοινότητα στο διαδίκτυο, ενδεχομένως όχι τόσο μεγάλη συγκριτικά με κοινότητες που ασχολούνται με γλώσσες προγραμματισμού όπως η C++ ή η Java. Η κοινότητα αυτή προσέφερε μεγάλη βοήθεια, παρέχοντας προτάσεις για πειραματισμό αι δίνοντας λύσεις σε ό,τι δυσκολία αντιμετωπίσαμε. Το χρονικό διάστημα πειραματισμού ήταν μεγάλο, και ήταν πολλές οι στιγμές που θεωρήσαμε ότι μπορεί και να μην καταφέρουμε να επιτύχουμε το επιθυμητό αποτέλεσμα.

Ο τελικός ήχος που παραγάγαμε θεωρούμε ότι είναι ο καλύτερος δυνατός που μπορέσαμε, δεδομένου του χρονικού περιορισμού που είχαμε. Οι παραμετροποιήσεις που ο χρήστης έχει τη δυνατότητα να πραγματοποιεί κατά της εκτέλεση της εφαρμογής πρέπει να είναι διακριτές ακουστικά, να είναι εύηχες, καθώς και να πληρούν ορισμένους περιορισμούς ορθής παραγωγής ήχου. Για να επιτευχθεί αυτό, που πραγματοποιούσαμε τροποποιήσεις στο πρόγραμμα για μεγάλο χρονικό διάστημα. Η συνεχής έκθεση σε αυτούς τους διαφορετικούς ήχους που δημιουργήσαμε, πολλές φορές μας προκάλεσε δυσφορία. Επίσης υπήρξαν στιγμές στις οποίες ανησυχήσαμε μήπως προκαλέσαμε κάποια βλάβη σε εμάς, ή στο σύστημα στο οποίο δουλεύαμε, κατά τη διάρκεια αναπαραγωγής αυξημένης έντασης ήχου, ή ανεπιθύμητου αποτελέσματος ήχου.

8.3 Μελλοντικές Βελτιώσεις

Κατόπιν ολοκλήρωσης της συγκεκριμένης πτυχιακής θεωρούμε απαραίτητο να εκφράσουμε σε ποια σημεία πιστεύουμε ότι υστερεί αυτή, και σε ποια σημεία πιστεύουμε ότι μπορούμε να βελτιωθούμε ή αναβαθμίσουμε μελλοντικά.

Αρχικά προτείνεται η αγορά μιας έκδοσης Arduino που να υποστηρίζει κάποια ασύρματη τεχνολογία όπως είναι το WI-FI. Αυτή θα οδηγήσει αρχικά σε ένα πιο εύχρηστο προϊόν και επιπλέον θα βοηθήσει και στο να σχεδιάσουμε ένα πιο απλό τρισδιάστατο μοντέλο για εκτύπωση.

Η τωρινή κατασκευή εμπεριέχει μόνο ένα κουμπί που σειριακά στέλνει τιμές σε διαφορετικές μεταβλητές. Η προσθήκη ενός δεύτερου κουμπιού υπάρχει στα μελλοντικά μας σχέδια για «backwards» χρήση. Επίσης η προσθήκη επιπλέον διαφορετικών modules υπήρχε πάντα στο μυαλό μας. Με αυτό τον τρόπο θα είχαμε τη δυνατότητα να επεξεργαστούμε περισσότερες μεταβλητές ταυτόχρονα.

Το κόστος εκτύπωσης του 3D μοντέλου ήταν αυξημένο, και ο σχεδιασμός ενός μικρότερου μοντέλου σε όγκο κρίθηκε σημαντικότερος παράγοντας, με αποτέλεσμα να παραμετροποιήσουμε το τελικό μοντέλο εκτύπωσης.

Τέλος είχαμε σαν πλάνο τη δημιουργία περισσότερων patchers, για την παραγωγή ήχων διαφορετικών ειδών μουσικής, για την περεταίρω επίδειξη των δυνατοτήτων του μουσικού αυτού οργάνου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «About Arduino,» Available: https://www.arduino.cc/en/about.

[2] «Arduino Modules,» Available: https://arduinomodules.info/.

[3] Π. Μποζάνης Δ., Εισαγωγή στην Πληροφορική & τους Υπολογιστές, Τζιόλα, 2016.

[4] «Differences Between Creative Computing and Creative Technology,» Available: https://www.smu.edu/Meadows/NewsAndEvents/News/2023/Differences-Between-Creative-Computing-and-Creative-Technology.

[5] Deitel M. Harvey, Deitel J. Paul, Java Προγραμματισμός, Γκιούρδας Μ., 2015.

[6] Alan Dix, Janet Finlay, Gregory D. Abowd, Beale Russellk Επικοινωνία Ανθρώπου – Υπολογιστή, Γκιούρδας Μ., 2007.

[7] M. PRZYBYLLA, «Physical Computing and its Scope – Towards a Constructionist Computer Science Curriculum with Physical Computing,» *Informatics in Education - An International Journal*, αρ. 2, pp. 241-254, 13/2014.

[8] Lu, Zhang, «Creativity in Intelligent Technologies and Data Science,» *Knowledge Discovery in Creative Computing for Creative Tasks*, τόμ. 535, p. 93–104, 2015.

[9] «What is an Arduino?,» . Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all.

[10] «Arduino Uno (R3), Arduino Nano, Arduino Pro, Arduino Mini,» Available: https://www.elprocus.com/different-types-of-arduino-boards/.

[11] «ATmega328P,» Available: https://content.arduino.cc/assets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

[12] «Arduino Uno Rev3 Store,» Available: https://store.arduino.cc/products/arduino-uno-rev3.

[13] «Aruino Shields,» Available: https://learn.sparkfun.com/tutorials/arduino-shields.

[14] «List of Arduino Modules,» Available: https://arduinomodules.info/.

[15] «Arduino IDE,» Available: https://www.arduino.cc/en/software.

[16] «Arduino IDE,» Available: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics.

[17] «IDE,» Available: https://aws.amazon.com/what-is/ide/.

[18] Wheeler, David A., «Program Library HOWTO,» 2003.

[19] «Arduino Libraries,» Available: https://www.arduino.cc/reference/en/libraries/.

[20] «mpu-6050,» Available: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/.

[21] «How to Interface Arduino and the MPU 6050 Sensor,» Available: https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor.

[22] «MPU6050 Module,» Available: https://nettigo.eu/products/mpu6050-3-axis-gyroscope-and-3-axis-acclerometer.

[23] «MPU-6050,» Available: https://www.circuits-diy.com/mpu6050-3-axis-accelerometer-and-gyroscope-module/.

[24] «Blender,» Available: https://www.blender.org/.

[25] Hearn Donald, Baker M. Pauline, Carithers R. Warren, Γραφικά Υπολογιστών με OpenGL, Τζιόλα, 2010.

[26] «MPU6050 Pinout Configuration,» Available: https://components101.com/sensors/mpu6050-module.

[27] «MPU-6000 and MPU-6050 Product Specification Revision 3.4,» Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[28] «Arduino Tutorial for Complete Beginners: Using a Button,» 2013. Available: https://www.kodeco.com/2740-arduino-tutorial-for-complete-beginners-using-a-button.

[29] «How to Wire and Program a Button Picture,» Available: https://docs.arduino.cc/static/73702ee121860fa04c7f6db5bc77183b/29114/circuit.png.

[30] «How to Wire and Program a Button With a Resistor Picture,» Available: https://docs.arduino.cc/built-in-examples/digital/Button.

[31] «tinkercad,» Available: https://www.tinkercad.com/.

[32] «MPU6050 Library,» Available: https://www.arduino.cc/reference/en/libraries/mpu6050/.

[33] ElectronicCats, «ElectronicCats MPU-6050 Library,» Available: https://github.com/ElectronicCats/mpu6050.

[34] M. Adams, «Michael Adams Button Arduino Code,» Available: https://github.com/madleech/Button.

[35] «Michael Adams Arduino Code,» Available: https://www.arduino.cc/reference/en/libraries/button/.

[36] «Cycling '74 MAX/MSP,» Available: https://cycling74.com/products/max.

[37] «Patcher Reference - Max Documentation,» Available: https://docs.cycling74.com/max5/refpages/max-ref/patcher.html.

[38] «Messy Cords,» Available: https://cycling74-web-uploads.s3.amazonaws.com/58b7280e137a734d61493953/2018-02-28T21:05:22Z/Screen%20Shot%202018-02-27%20at%206.07.49%20PM.png.

[39] «MIDI-TO-NOTE,» Available: https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies.

[40] «makenote,» Available: https://docs.cycling74.com/max5/refpages/max-ref/makenote.html.

[41] Μουστακίδης, Γεώργιος Β., Βασικές Τεχνικές Ψηφιακής Επεξεργασίας Σημάτων, Τζιόλια, 2004.

[42] «ADSR~ Documentation,» Available: https://docs.cycling74.com/max5/refpages/msp-ref/adsr~.html.

[43] «ADSR,» Available: https://mastering.com/adsr/.

[44] Savitch, Walter, C++ Επίλυση Προβλημάτων, Τζιόλα, 2018.

[45] «Serial Communications between Arduino and Max,» Available: https://chariscat.wordpress.com/2020/12/06/serial-communications-between-arduino-and-max/.

[46] «Midi Note Numbers and Center Frequencies,» Available: https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies.

[47] Monson, Hayes H., Ψηφιακή Επεξεργασία Σήματος, Τζιόλα, 2000.

[48] Ngo, Tuan D., «Additive manufacturing (3D printing): A review of materials, methods, applications and challenges,» *Composites Part B: Engineering,* τόμ. 143, pp. 172-196, 2018.

[49] «MAIN COMPONENTS OF DESKTOP 3D PRINTERS,» Available: http://my3dconcepts.com/wp-content/uploads/2017/01/Hardware-components-of-FDM-3D-Printers.jpg.

[50] «3Dexpert,» Available: https://www.3dexpert.gr/.

[51] «Autodesk Tinkercad,» Available: tinkercad.com.

[52] «Arduino UNO R3,» Available: https://docs.arduino.cc/hardware/uno-rev3.

[53] «Cycle~ Documentation,» Available: https://docs.cycling74.com/max5/refpages/msp-ref/cycle~.html.

ΠΑΡΑΡΤΗΜΑ Α : ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ

Για να χρησιμοποιήσουμε σωστά το μουσικό όργανο που κατασκευάσαμε και προγραμματίσαμε πρέπει να ακολουθήσουμε τα ακόλουθα μέτρα.

Arduino IDE

Εγκαθιστούμε τις απαραίτητες βιβλιοθήκες:

- Επιλέγουμε το «Sketch» → «Include Library» → «Manage Libraries».
- Στο Library Manager, αναζητούμε «MPU6050» και εγκαθιστούμε τη βιβλιοθήκη που εμφανίζεται στην οθόνη.

Μεταφορτώνουμε τον επιθυμητό κώδικα στον Arduino:

- Επιλέγουμε το «File» → «Open» → Επιλέγουμε το αρχείο που θέλουμε. Εναλλακτικά δημιουργούμε δικό μας αρχείο.
- Βεβαιωνόμαστε ότι έχουμε επιλέξει το σωστό board και port στο Arduino IDE.
- Πιέζουμε το κουμπί «Upload» για να μεταφορτώσουμε τον κώδικα στο Arduino.
- Ορίζουμε σωστά την τιμή του baud rate σύμφωνα με τον κώδικα της εφαρμογής.
- Παρακολουθούμε την έξοδο του προγράμματος στην σειριακή οθόνη.

MAX/MSP

- Εκτελούμε τον Patcher που έχουμε δημιουργήσει.
- Εκτελούμε το Initialize Subpatch πιέζοντας το ανάλογο Button.
- Βεβαιωνόμαστε πως το port number είναι σωστό πιέζοντας το print αντικείμενο που βρίσκεται εντός του Arduino Subpatch.
- Ενεργοποιούμε το Arduino Subpatch πιέζοντας το ανάλογο Toggle.
- Βεβαιωνόμαστε ότι το subpatch εκτελείται σωστά μέσω του subpatch 6.3.3.
- Στη συνέχεια ενεργοποιούμε οποιοδήποτε από τα First Second Third ή All Instrument Subpatches μας ενδιαφέρει.
- Πιέζουμε το κουμπί που βρίσκεται στο Music Box, και βλέπουμε τις τιμές που παραμετροποιούνται την κάθε φορά.
- Περιστρέφουμε το music box για να παραμετροποιήσουμε τις τιμές σύμφωνα με την αρέσκειά μας.
- Πιέζουμε ξανά το κουμπί ώστε να σταματήσουμε την παραμετροποίηση της τρέχουσας μεταβλητής, και να ξεκινήσουμε να πραμετροποιούμε την επόμενη.
- Μπορούμε συμπληρωματικά να παραμετροποιήσουμε τα επιθυμητά αντικείμενα μέσω της διεπαφής του MAX/MSP που δεν έχουμε ορίσει να επηρεάζονται από τον Arduino.
- Σε περίπτωση αποσύνδεσης του Arduino κατά την εκτέλεση της εφαρμογής απενεργοποιούμε και ενεργοποιούμε το toggle του Arduino Subpatch.

ΠΑΡΑΡΤΗΜΑ Β : ΠΙΝΑΚΕΣ ΚΑΙ ΤΕΧΝΙΚΑ ΔΕΔΟΜΕΝΑ

Πίνακας 5:Arduino Uno R3 Technical Specifications [52]

Board	Name	Arduino UNO R3
	SKU	A000066
Microcontroller	ATmega328P	
USB connector	USB-B	
Pins	Bult-in LED Pin	13
	Digital I/O Pins	14
	Analog input Pins	6
	PWM Pins	6
Communication	UART	Yes
	I2C	Yes
	SPI	Yes
Power	I/O Voltage	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Barrel Plug
Clock speed	Main Processor	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	ATmega328P	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	Weight	25 g
	Width	53.4 mm
	Length	68.6 mm

Πίνακας 6: Midi Note Numbers and Center Frequencies [46]

MIDI note number	Note names	Frequency (tuning at 440 Hz)
127	G9	12543.85
126	F#9/Gb9	11839.82
125	F9	11175.30
124	E9	10548.08
123	D#9/Eb9	9956.06
122	D9	9397.27
121	C#9/Db9	8869.84
120	C9	8372.02
119	B8	7902.13
118	A#8/Bb8	7458.62

117	A8	7040.00
116	G#8/Ab8	6644.88
115	G8	6271.93
114	F#8/Gb8	5919.91
113	F8	5587.65
112	E8	5274.04
111	D#8/Eb8	4978.03
110	D8	4698.64
109	C#8/Db8	4434.92
108	C8	4186.01
107	B7	3951.07
106	A#7/Bb7	3729.31
105	A7	3520.00
104	G#7/Ab7	3322.44
103	G7	3135.96
102	F#7/Gb7	2959.96
101	F7	2793.83
100	E7	2637.02
99	D#7/Eb7	2489.02
98	D7	2349.32
97	C#7/Db7	2217.46
96	C7	2093.00
95	B6	1975.53
94	A#6/Bb6	1864.66
93	A6	1760.00
92	G#6/Ab6	1661.22
91	G6	1567.98
90	F#6/Gb6	1479.98
89	F6	1396.91
88	E6	1318.51
87	D#6/Eb6	1244.51
86	D6	1174.66
85	C#6/Db6	1108.73
84	C6	1046.50
83	B5	987.77
82	A#5/Bb5	932.33
81	A5	880.00
80	G#5/Ab5	830.61
79	G5	783.99
78	F#5/Gb5	739.99
77	F5	698.46
76	E5	659.26
75	D#5/Eb5	622.25
74	D5	587.33
73	C#5/Db5	554.37
72	C5	523.25
71	B4	493.88
70	A#4/Bb4	466.16

69	A4 concert pitch	440.00
68	G#4/Ab4	415.30
67	G4	392.00
66	F#4/Gb4	369.99
65	F4	349.23
64	E4	329.63
63	D#4/Eb4	311.13
62	D4	293.66
61	C#4/Db4	277.18
60	C4 (middle C)	261.63
59	B3	246.94
58	A#3/Bb3	233.08
57	A3	220.00
56	G#3/Ab3	207.65
55	G3	196.00
54	F#3/Gb3	185.00
53	F3	174.61
52	E3	164.81
51	D#3/Eb3	155.56
50	D3	146.83
49	C#3/Db3	138.59
48	C3	130.81
47	B2	123.47
46	A#2/Bb2	116.54
45	A2	110.00
44	G#2/Ab2	103.83
43	G2	98.00
42	F#2/Gb2	92.50
41	F2	87.31
40	E2	82.41
39	D#2/Eb2	77.78
38	D2	73.42
37	C#2/Db2	69.30
36	C2	65.41
35	B1	61.74
34	A#1/Bb1	58.27
33	A1	55.00
32	G#1/Ab1	51.91
31	G1	49.00
30	F#1/Gb1	46.25
29	F1	43.65
28	E1	41.20
27	D#1/Eb1	38.89
26	D1	36.71
25	C#1/Db1	34.65
24	C1	32.70
23	B0	30.87
22	A#0/Bb0	29.14

21	A0	27.50
20		25.96
19		24.50
18		23.12
17		21.83
16		20.60
15		19.45
14		18.35
13		17.32
12		16.35
11		15.43
10		14.57
9		13.75
8		12.98
7		12.25
6		11.56
5		10.91
4		10.30
3		9.72
2		9.18
1		8.66
0		8.18